



Theses and Dissertations

2018-12-01

SWAT Online: Development of a Web-Based Decision Support System for the Soil and Water Assessment Tool

Spencer Dean McDonald
Brigham Young University

Follow this and additional works at: <https://scholarsarchive.byu.edu/etd>



Part of the [Civil and Environmental Engineering Commons](#)

BYU ScholarsArchive Citation

McDonald, Spencer Dean, "SWAT Online: Development of a Web-Based Decision Support System for the Soil and Water Assessment Tool" (2018). *Theses and Dissertations*. 8810.
<https://scholarsarchive.byu.edu/etd/8810>

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

SWAT Online: Development of a Web-Based Decision Support System
for the Soil and Water Assessment Tool

Spencer Dean McDonald

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of
Master of Science

E. James Nelson, Chair
Daniel P. Ames
Norman L. Jones

Department of Civil and Environmental Engineering
Brigham Young University

Copyright © 2018 Spencer Dean McDonald

All Rights Reserved

ABSTRACT

SWAT Online: Development of a Web-Based Decision Support System for the Soil and Water Assessment Tool

Spencer Dean McDonald
Department of Civil and Environmental Engineering, BYU
Master of Science

As satellite and climate modelling technology continues to improve and as climatological disasters and issues continue to impact the global community, climate data will increase in size and relevance. With this new influx of information, it is becoming more and more important for scientists to simply and concisely communicate their findings to both decision makers in governments and disaster preparedness organizations and also to the general public. The Soil and Water Assessment Tool (SWAT) is a powerful modelling tool that allows scientists to simulate essentially all of the physical processes involved in the water cycle. The data that SWAT produces can be valuable information as people strive to better plan for and understand various hydrologic events.

The work presented in this thesis represents an effort to overcome some of the limitations of the previously developed SWAT visualization software by creating a set of modular web applications that can be duplicated, customized, and run by any organization or individual interested in visualizing and sharing data from SWAT. By eliminating the technical knowledge barriers that are inherent in running and using SWAT models, this work has the potential to increase SWAT's impact on non-technically trained stakeholders and decision makers in areas where water and climate management is important.

Keywords: Soil and Water Assessment Tool, hydrology, Tethys, decision support tool

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Jim Nelson for his continued support over the last three years, for his allowing me to be involved in the SERVIR project, and for ultimately helping me find funding and a place to work while I continued my thesis work in Bangkok, Thailand. I would also like to thank the other two members of my committee, Dr. Norm Jones and Dr. Dan Ames for their support and willingness to help me complete this thesis remotely.

I am indebted to Dr. John Bolten and his team at the NASA Goddard Space Flight Center for their support in helping me understand SWAT and giving me the opportunity to work with them on this project. I would like to thank Dr. Peeranan Towashiraporn and DDr. Farrukh Chishtie for hosting me at ADPC. I would like to thank my colleagues at ADPC, Chinaporn Meechaiya, Kingkan Chamnan, Ate Poortinga, Nguyen Hanh Quyen, Biplov Bhandari, Alex Barrett, Kittiphong Phongsapan, and Waddee Deeprawat for their friendship, support, laughter, and good food.

A special thanks to Sarva Pulla. He has played a significant role in the development of these tools and has been a great friend to me over the last four or five years. He has been very supportive, often willing to have video calls during non-working hours to help me with stubborn code.

I would also like to thank my parents, siblings, and my wife for their continued support and understanding throughout this roller coaster.

This material is based upon work supported by the SERVIR, NASA Applied Sciences Program, Grant No. NNX16AN45G.

TABLE OF CONTENTS

LIST OF TABLES.....	vi
LIST OF FIGURES	vii
1 Introduction	1
1.1 Big climate data.....	2
1.2 Decision Support Tools.....	4
1.3 The Soil and Water Assessment Tool (SWAT).....	8
2 Background.....	10
2.1 The Soil and Water Assessment Tool (SWAT).....	10
2.1.1 How it works.....	11
2.2 Desktop Applications	18
2.2.1 ArcSWAT	18
2.2.2 MWSWAT	19
2.2.3 QSWAT	20
2.2.4 SWAT-CUP	20
2.2.5 SWAT-GRASS.....	20
2.2.6 SWAT Output Viewer desktop application	21
2.2.7 VizSWAT	22
2.3 Web interfaces and web services	22
2.3.1 WRESTORE.....	23
2.3.2 OWS4SWAT	24
2.3.3 SWATShare	24
2.4 Moving forward.....	26
2.5 Web applications in the Tethys Platform	26
2.5.1 Software organization	28
2.5.2 Tethys app folder structure	29
2.6 Case study area: The Lower Mekong River Basin.....	30
3 Providing access to SWAT inputs and outputs through the Tethys Platform	34
3.1 Introduction	34
3.2 Accessing SWAT inputs through the nasaaccess web application	35

3.3	Viewing and accessing SWAT outputs through the SWAT Data Viewer application ..	40
3.3.1	Home view	40
3.3.2	Feature selection	41
3.3.3	Data modal view	42
3.4	Using output from the SWAT Data Viewer and nasaaccess apps for further analysis ..	50
3.5	Summary	51
4	Discussion.....	54
4.1	Summary	54
4.2	Current limitations and future work.....	55
4.3	Assessing impact	55
	References.....	58
	Appendix A. Technical Documentation: nasaaccess App.....	68
A.1	Data requirements and file structure	69
A.2	App architecture	70
A.3	Map publishing.....	73
A.4	App functionality.....	73
	Appendix B. Technical Documentation: SWAT Data Viewer	85
B.1	Data requirements and file structure	85
B.2	App architecture	87
B.3	Initializing PostgreSQL database and uploading data	89
B.4	Map publishing.....	92
B.5	App functionality.....	93

LIST OF TABLES

Table 2-1: Input files needed to run a SWAT simulation.....	12
Table 2-2: List of nasaaccess functions	14
Table 2-3: Land-phase processes simulated by SWAT for each HRU.....	15
Table 2-4: Routing phase processes simulated by SWAT for each stream reach	16
Table A-1: nasaaccess functions	69
Table A-2: Description of each file in the app package.....	71
Table B-1: Description of all required files in SWAT Data Viewer	86
Table B-2: Description of each file in the app package.....	89
Table B-3: Description of tables in the SWAT Data Viewer PostgreSQL Database	90
Table B-4: Individual functions within upload_new_model.py	91
Table B-5: Description of key-value pairs in time-series data dictionaries.....	99

LIST OF FIGURES

Figure 2-1: Example output.rch file and list of output variables	17
Figure 2-2: Tethys platform software architecture	28
Figure 2-3: Example Model-View-Controller architecture	29
Figure 2-4: Generic Tethys app file structure	30
Figure 3-1: nasaaccess app user interface	35
Figure 3-2: Left pane of nasaaccess interface with user inputs	36
Figure 3-3: Contact Information modal for email submittal.....	37
Figure 3-4: Example email message notifying the user that their data is ready for download.....	37
Figure 3-5: Download Data modal	38
Figure 3-6: Example file structure of nasaaccess output files	39
Figure 3-7: Example GPMswat output file starting on June 1, 2011.....	39
Figure 3-8: Initial view of the SWAT Data Viewer	40
Figure 3-9: Configurations and layer visualizations in the SWAT Data Viewer	41
Figure 3-10: Maps displaying subset spatial data within the data modal	42
Figure 3-11: Initial view of the data modal after feature selection.....	43
Figure 3-12: Tabs within data modal used SWAT Output visualization.....	44
Figure 3-13: Monthly streamflow time series extracted from output.rch.....	44
Figure 3-14: Daily streamflow time series extracted from output.rch.....	45
Figure 3-15: Daily precipitation time series extracted from output.sub	45
Figure 3-16: Daily surface runoff time series extracted from output.sub.....	46
Figure 3-17: Land Use/Land Cover coverage statistics	47
Figure 3-18: Soil type coverage statistics	47
Figure 3-19: nasaaccess interface in the SWAT Data Viewer.....	48
Figure 3-20: Example csv file produced by the SWAT Data Viewer	49
Figure 3-21: The data cart tab in the SWAT Data Viewer	49
Figure 3-22: Hypothetical workflow for using output data from the SWAT Data Viewer	50
Figure 3-23: Hypothetical workflow for using nasaaccess outputs	51
Figure A-1: nasaaccess_data file structure	70
Figure A-2: nasaaccess application architecture.....	71

Figure A-3: MVC framework for nasaaccess application	72
Figure A-4: Main view of nasaaccess application	74
Figure A-5: Model and form structure for uploading new files to the app.....	75
Figure A-6: Code to upload shapefile to geoserver	76
Figure A-7: User input options for accessing nasaaccess functions.....	77
Figure A-8: Contact information modal for notification when functions have completed.....	78
Figure A-9: nasaaccess_run() code.....	79
Figure A-10: Input arguments passed from front end to nasaaccess.py	80
Figure A-11: nasaaccess functions being called based on user inputs.....	80
Figure A-12: File structure for nasaaccess function outputs	81
Figure A-13: send_email() code	82
Figure A-14: Example email sent by the nasaaccess app	82
Figure A-15: Download data modal	83
Figure A-16: download_data() code	84
Figure B-1: Data files required for SWAT Data Viewer to work.....	86
Figure B-2: SWAT Data Viewer application architecture.....	88
Figure B-3: MVC framework for the SWAT Data Viewer	88
Figure B-4: User specified options for upload_new_model.py	91
Figure B-5: Initial view of the SWAT Data Viewer with layer toggling	93
Figure B-6: get_upstreams() code.....	94
Figure B-7: clip_raster() code.....	95
Figure B-8: Layer configurations for maps in the data modal.....	96
Figure B-9: Data modal view.....	97
Figure B-10: Tabs for querying SWAT outputs	98
Figure B-11: Python dictionary created by extraction functions and passed to front end.....	99
Figure B-12: extract_sub() code	100
Figure B-13: Example time-series plot in “Sub Model Outputs” tab	101
Figure B-14: coverage_stats() code for identifying unique values in raster.....	102
Figure B-15: coverage_stats() code for computing % coverage for each unique value	102
Figure B-16: coverage_stats() code for creating lulc dictionary to be sent to front end	103
Figure B-17: coverage_stats() code for creating soil dictionary to be sent to front end.....	104

Figure B-18: Example land cover distribution pie chart shown in the “LULC Data” tab.....	104
Figure B-19: The nasaaccess interface within the SWAT Data Viewer.....	105
Figure B-20: Example time-series csv file written by the app.....	106
Figure B-21: The “Data Cart” tab.....	106

1 INTRODUCTION

The era of Big Data is in full swing. In a review of regularly cited papers on big data, De Mauro et al. proposed the following as formal definition: “Big Data represents the information assets characterized by such a High Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value” (De Mauro et al., 2015). In this definition, “volume” refers to the amount of data, “velocity” refers to the speed at which it is collected, and “variety” refers to the type of data (Laney, 2001). Thanks to the internet, information on an infinite number of unique topics is now available at the click of a button or a few keystrokes. Incidentally, all of those clicks and keystrokes are also adding to the data that is being made available. Data is being collected, stored, and analyzed continuously and is then used to produce even more data. From social media posts/likes/tweets to the daily weather forecasts that you access on your cell phone, the volume and variety of all this data has far exceeded the capabilities of manual data analysis (Provost and Fawcett, 2013). Thus, there is, and will continue to be, a great need for new and more efficient ways to process information in a way that will add value to our world.

All of this data offers valuable information to anyone willing to take on the task of sifting through it (Assunção et al., 2015). Professionals in essentially every field are constantly looking for better ways to access and analyze this data to make more informed decisions (Boyd and Crawford, 2012). Evidence strongly suggests that the implementation of big data technologies to

aid in decision-making processes can greatly improve performance within an organization (Provost and Fawcett, 2013). The climate sciences (i.e. hydrology, meteorology, etc.) are one of the many fields that have an abundance of data available that has the potential to improve the lives of people around the world. This potential has yet to be realized fully, however, as the technical barrier in accessing and understanding the data remains too high for most people to use.

1.1 Big climate data

For the last few years the issue of climate change has come to the forefront of many political and economic issues. Two initiatives of the United Nations, the Sustainable Development Goals (United Nations, 2016) and the Sendai Framework (UNO UNISDR, 2015), are strongly linked to the sustainability of the environment and safety from climatological disasters. In addition to the overly politicized climate change crisis, there are also natural disasters occurring on a very frequent basis. These crises and disasters are generally unavoidable and have led many to question how these global changes (deforestation, global warming, population growth, and natural disasters) will affect the livelihoods of humans around the world (Foley, 2011). These climate issues have emerged into global awareness simultaneous with the growth of big data or, more specifically, the increase in the amount of climate data that is now available (Faghmous and Kumar, 2014). In this paper I argue that with the emergence of better technologies to access, analyze, and share climate information, we can greatly improve the efforts of governments around the world and also people in disaster-prone areas to prepare for, and respond to, the changes in climate and the disasters that will continue to occur.

There are many organizations like NASA in the United States and the European Centre for Medium Range Weather Forecasts (ECMWF) in Europe that are producing volumes of earth

observations datasets. The Global Precipitation Measurement (GPM) mission, which was launched in 2015 by NASA, produces half-hourly and monthly rainfall estimates on a global scale in near-real time (NASA, 2011). Further, local organizations around the world are using improved sensors and crowd-sourcing to collect more and more in-situ observation data (e.g. stream and rainfall data). These in-situ measurements are valuable to organizations, but often lack the spatial resolution to be used on a larger scale (Verdin et al., 2015). Using satellite data to supplement the sometimes-sparse availability of in-situ observation data can help to better identify spatial distributions and anomalies of the earth's various climate processes (Chen and Han, 2016). Most of this data is archived for use in a wide range of projects around the world.

In addition to the massive amount of data from satellite and in-situ observed datasets, hydrologic and climate models are being used to improve the applicability of these observation products and in and of themselves produce large volumes of additional data. There has also emerged another hybrid data source, often called a reanalysis or data assimilation system, that uses both satellite and in-situ data in conjunction with numerical models to address the heterogeneity and inaccuracy that is inherent in the observation datasets (Faghmous and Kumar, 2014). This hybrid data can be used with various climate models to then produce even more information from the archived raw observation data. The Global Land Data Assimilation System (GLDAS) produced by NASA is but one example of this reanalysis data type. GLDAS uses both satellite and ground-based observation as inputs in a complex land surface model to create near real-time data for a broad range of climate variables (e.g. surface water runoff, soil moisture, air temperature, evapotranspiration) on a global scale at a fairly high resolution (~1km grid cells) (NASA Goddard Space Flight Center, n.d.). ECMWF uses a similar methodology to produce

global 15-day ensemble forecasts and historical reanalysis datasets for multiple weather and land surface variables (ECMWF).

Earth observations, both from space and ground, when supplemented with climate models and statistical data assimilation methods have already proven to be powerful tools in forecasting future events, analyzing past events, and performing what-if analyses to assist in disaster preparedness and future infrastructure planning. Efforts are being made within the fields of data science, climatology, hydrology, and hydroinformatics to use all of this data to better understand the management of the world's water and to ensure its fair use. One of the objectives of the relatively new field of hydroinformatics, is to use information and communication technologies to store, manage, and analyze water data so that more people can understand the water cycle and its impact on the environment (Chen and Han, 2016). However, even with new advances in data management techniques and the continuous generation of climate data, we are barely scratching the surface. In a study done by Overpeck et al. (2011), the amount of climate data is projected to reach 350 petabytes (1 PB = 1,000 TB) by the year 2030 but according to Selding only 3-5% of the currently available data is actually being used. The findings of Dahlhaus et al. (2016) suggest that an even smaller amount of that data is actually reaching those that can use it to make decisions.

1.2 Decision Support Tools

The data discussed above was, and continues to be, produced to help people. Within those petabytes of data there are patterns that can help us understand the past and also predict future changes (Assunção et al., 2015). Unfortunately, the people that are in a position to make decisions based on the information from these models generally don't have the infrastructure, the

time, or the technical expertise to sift through and analyze the raw data to make the informed decisions for which the data was produced in the first place (Dahlhaus et al., 2016). There is a wide community of possible stakeholders that are trying to understand and use this climate information to make decisions for themselves, for their families, their land and, in the case of government and disaster preparedness agencies, for whole communities (Overpeck et al., 2011). Climate data in its raw form will never be able to reach everyone in this broad community of stakeholders. Most of this data and the models that go along with it are still being developed for strictly academic and research purposes without much thought to the educational and decision support power that it contains (Janssen et al., 2009; Olsson and Andersson, 2006). For this reason, organizations with the technical expertise and understanding of all this data are now focusing much of their time and efforts to, in effect, transform the data into understandable and actionable information for those less familiar with the science.

Two international organizations which I have collaborated with that are working to achieve this goal of providing actionable information to stakeholders are the International Centre for Integrated Mountain Development (ICIMOD) in Nepal and the Asian Disaster Preparedness Center (ADPC) in Thailand. These two organizations, along with others in Africa and South America, are working with the assistance of NASA and USAID through the SERVIR project (SERVIR Global) to bring state-of-the-art science and data to people and regions prone to climatological disasters (floods, droughts, climate change). Regions that have historically lacked information to help prepare for those extreme events. The SERVIR objective of, “connecting space to village”, encapsulates the goal to make earth data available to everyone.

According to Overpeck et al., there are two things that the success of projects like SERVIR rely on. First, continued funding is a necessity for the improvement of climate sciences,

especially when it comes to the management of data and modeling tools. Without sufficient financial support, these new technologies and data products will never be able to reach their full potential in helping reduce human vulnerability to the changes in the climate and to natural disasters (Overpeck et al., 2011). Second, for this new push to provide information to “the village” to be successful, data needs to be shared freely. While there have been efforts recently to provide open access to data among governments and across professional fields (Goodall et al., 2008), there still remains many restrictions on accessing data from outside sources, particularly when it comes to water resource rights across political boundaries. There needs to be additional effort and openness to make this data available to all (Overpeck et al., 2011). The HydroShare system, developed by Tarboton et al. (2014), is a great example of the efforts being made to share model outputs and observation data across different technical fields.

I will add a third criterion for the success of this new paradigm. Even with the added funding and improved data accessibility, these projects will fall flat without the development of robust, yet simple to use, web-based tools that facilitate quick and easy to understand visualization of information for those in positions to make decisions. For more effective environmental and natural resource management to occur, there needs to be an easier way for decision makers to access and visualize data (Dahlhaus et al., 2016). The open, fast, and interactive nature of the internet makes it a great place to host the data and visualization functionality (Voinov and Costanza, 1999). For this reason, over the last few decades, many organizations have been developing systems called Decision Support Tools (DST) or Decision Support Systems (DSS).

These decision support systems, especially when enabled through web services (Goodall et al., 2008), can have a large impact in helping stakeholders (e.g. government officials, water

managers, farmers, emergency first responders) to make quick and informed decisions (Klug and Kmoch, 2015). There are examples of these decision support systems within just about any climate related organization. Oddo et al., combined the power of MODIS imagery (NASA Goddard Space Flight Center) with population, land use, and infrastructure datasets to produce a web tool showing near-real time flood extent maps and also information on the socioeconomic impacts that the flood is potentially having on a given region. They argue that the information produced by their workflow is valuable to government officials, international agencies, and first responders (Oddo et al., 2018). Similarly, in a study performed by Deltares and Floodtags in the Netherlands, crowd-sourced images of a flood in Indonesia were obtained from twitter posts to supplement other satellite and ground observations to create a more accurate flood extent maps to give to decision makers (Eilander et al., 2016). In Victoria, Australia, The Visualizing Victoria's Groundwater data portal allows for end-users of groundwater data to have direct access to the data that they need, which cuts out the governmental middle man (Dahlhaus et al., 2016). As far back as 2003, Engel et al. developed a web-based what-if-analysis tool that allowed users (many with limited hydrology background) to run hydrologic simulations for different land-use changes in a selected watershed and then to compare data in the web interface (Engel et al., 2003). The Streamflow Prediction Tool, developed by Snow et al., is a web-based decision-support tool that leverages the global runoff 15-day forecast and 35-year historical simulation data produced by ECMWF and the Routing Application for Parallel computation of Discharge (RAPID) (David et al., 2016) to show 15-day river flow forecasts and flood warnings for watersheds around the world. The National Water Model (The Office of Water Prediction, n.d.) produces daily hydrologic forecasts and hindcasts for 2.7 million stream reaches in the National Hydrography Dataset throughout the United States (USGS). Souffront Alcantara et al.

developed a cyberinfrastructure (a REST API) along with a suite of web applications to provide for easier access and analysis of the NWM data within the HydroShare architecture (Souffront Alcantara et al., 2017). These are but a few examples, barely scratching the surface of the kinds of climate related decision support tools that are available around the world.

1.3 The Soil and Water Assessment Tool (SWAT)

One of the commonly used tools for generating climate and hydrologic information to be used by decision-makers is the Soil and Water Assessment Tool (SWAT). SWAT is being used globally to model virtually all of the different physical processes involved in the hydrologic cycle and has proven to be a valuable component to a decision support system for water managers (Muleta and Nicklow, 2005). Through using SWAT, decision-makers and stakeholders can better understand the impacts of both natural and human driven changes on the environment and then make more informed decisions based on that information (Arias et al., 2011). With that said, the full potential of SWAT has yet to be realized. The complexity in obtaining and preparing the necessary input data, running the model, and then making sense of its various output files has been a barrier for those lacking the technical training, software licenses, and computing resources (Roehrig, 2002; Jayakrishnan et al., 2005).

The work presented in this thesis represents an effort to lower that technical barrier for SWAT through using open source web development, web services, and cloud storage technologies. Using the Tethys platform (Swain et al., 2015) as the web framework and the data and model outputs from an upgraded regional SWAT model for the Lower Mekong River Basin in Southeast Asia (Mohammed et al., 2018), I created a SWAT output viewer and online data portal in the form of a modular web application. While the SWAT model used as a case study is

for the Lower Mekong River Basin, the application is completely open source and modular which means that it can be implemented anywhere (Croney et al., 2007; Nelson et al., 2016) for any watershed with SWAT output data available. The work to develop this web interface to SWAT, completed in collaboration with experts at the Asian Disaster Preparedness Center, the Mekong River Commission, and NASA Goddard Space Flight Center, will provide stakeholders in the Mekong region with a simple way of accessing the wealth of hydrologic, climate, and water quality information available from SWAT input and output data products.

2 BACKGROUND

2.1 The Soil and Water Assessment Tool (SWAT)

The Soil and Water Assessment Tool is a physically based, continuous time model focused on estimating the long-term impacts of land management practices and climatological changes on the hydrologic cycle of large and complex watersheds (Neitsch et al., 2009). Originally developed by Dr. Jeff Arnold for the US Department of Agriculture (USDA) Agricultural Research Service (ARS), SWAT has been widely applied by water managers and policy makers throughout the world (Krysanova and White, 2015). To share just a few examples, SWAT has been used to perform a continental simulation of water management scenarios over the entire United States (Srinivasan et al., 1998; Jayakrishnan et al., 2005), an analysis of land cover change impacts on watersheds in East Africa (Baker & Miller, 2013), an assessment of the effects of hydropower dams on water quantity and quality in multiple river basins throughout Asia (Zhang et al., 2011; Bouraoui et al., 2005; Jeong et al., 2013), and to identify areas prone to the negative effects of climate change in watersheds in South America (Grimson et al., 2013) and in India (Narsimlu et al., 2013).

Francesconi et al. have completed an in-depth review of journal articles that have been published about the usefulness of SWAT as a tool for modeling ecosystem services (i.e. the benefits that humans obtain from nature). From their review, they found that SWAT has proven to be a valuable tool for stakeholders and decision-makers in evaluating the effects of various

management and land cover changes on downstream ecosystem services (Quintero et al., 2009; Welderufael et al., 2013). However, because of the complexity of SWAT's data preparation and computations, there is still room for improvement in making it accessible to environmental managers and decision-makers (Vigerstol and Aukena, 2011).

In this chapter, I will review how SWAT works, the data and computational requirements for it to run, and some of the current data visualization methods being used for SWAT model outputs. I will then introduce the case study area that my tools were built for.

2.1.1 How it works

SWAT simulates a wide range of physical processes. For this reason, it is actually a combination of multiple sub-models. The models that have contributed most to the development of SWAT are the SWRRB model (Simulator for Water Resources in Rural Basins) (Williams et al., 1985; Arnold et al., 1990), CREAMS (Chemicals, Runoff, and Erosion from Agricultural Management Systems) (Knisel, 1980), GLEAMS3 (Groundwater Loading Effects on Agricultural Management Systems) (Leonard et al., 1987), and EPIC (Erosion-Productivity Impact Calculator) (Williams et al., 1984). Each of these sub models simulates a specific climatological, hydrologic, or chemical process that contributes to the water cycle. SWAT, being a semi-distributed physically-based model, is driven by a simplified representation of the hydrologic cycle and the assumptions of the water balance equation (Neitsch et al., 2009).

Because SWAT is physically-based and considered to be semi-distributed, it depends on a wide range of input datasets to represent the state of the watershed area during the time to be simulated (Arnold et al., 2009). Table 2-1 gives an overview of what data are required and some methods for obtaining that data. A more in depth description of each of these input datasets and

how to prepare them for a SWAT simulation can be found in the Soil and Water Assessment Tool input/output file documentation by Arnold et al. on the official SWAT website (<https://swat.tamu.edu/documentation/2012-io/>).

Table 2-1: Input files needed to run a SWAT simulation

Input Data	How to obtain	required?
Watershed/stream network	Delineate from DEM	Yes
Land Use/Land Cover	Obtain from local organization, global/regional datasets, or user-defined	Yes
Soil Type	Obtain from local organization, global/regional datasets, or user-defined	Yes
Rainfall Data	Rainfall gauge data from local organization, remote sensed data, or user-defined	Yes
Air Temperature Data	Rainfall gauge data from local organization, remote sensed data, or user-defined	Yes
Other Climate Data	Rainfall gauge data from local organization, remote sensed data, or user-defined	No
Land/Water management methods and schedules	From local organization or user-defined	No
Observed Data for calibration	From local organization managing gauges/sensors	No
Point source information (e.g. Treatment plants)	From local organization or user-defined	No

One of the main obstacles in running SWAT lies in the difficulty of accessing the various input files required by SWAT. Without access to in-situ observed data from local organizations, a user's options are often limited to having to use a global dataset like the Global Precipitation Measurement (GPM) and Global Land Data Assimilation System (GLDAS) datasets. While these datasets are available for free, their global nature requires for a substantial amount of processing before the data can be used in a model like SWAT. nasaaccess is a software tool built in R that streamlines the retrieval and processing of the global NASA earth observation data products (GPM and GLDAS) for use in models such as SWAT (Mohammed et al., 2018). The core functionality of nasaaccess can be summarized as:

- Access the NASA Goddard Space Flight Center (GSFC) servers to download earth observation data
- Clip needed grids to an input shapefile of a user study watershed
- Handle temporal and spatial inconsistencies
- Generate daily climate gridded data files and definition files compatible with SWAT and other models

nasaaccess was built as an R library containing four separate data processing functions which are described in Table 2-2. It is a very efficient system for accessing earth observation data. However, the number of potential users has been limited in part because of the relatively high learning curve involved in using R. Part of the work I will present in the following chapters involves an effort to make the nasaaccess functionality available to a wider audience using a custom web interface.

Table 2-2: List of nasaaccess functions

Function	Definition
GLDASwat	Generates SWAT compatible air temperature files
GPMswat	Generates SWAT compatible precipitation files
GLDAS Poly Centroid	Generates an air temperature station file at the centroid of each polygon within the input watershed boundary
GPM Poly Centroid	Generates a precipitation station file at the centroid of each polygon within the input watershed boundary

Within SWAT, the hydrologic cycle is divided in two separate phases: The land phase, and the routing phase. In the land phase, the watershed is divided into multiple subwatersheds or subbasins and then further into what are called Hydrologic Response Units (HRUs) to better represent the heterogeneity of the watersheds physical properties. This subdivision facilitates easier analysis by the user in comparing the effects of different land uses, soils, and management techniques on the overall watershed response (Neitsch et al., 2011).

An HRU is defined as the total area within a subbasin that has a unique combination of land use class, soil type, and management method (Arnold et al., 2012). Generally, each subbasin will contain 1-10 unique HRUs. Each of the physical processes defined in Table 2-3 are modelled for each HRU separately and then combined to produce totals for the entire subbasin (Neitsch et al., 2011).

Table 2-3: Land-phase processes simulated by SWAT for each HRU

Variable	Description
Canopy Storage	Used to compute maximum storage for each land cover/crop type When computing evaporation, the model first removes the water from canopy storage.
Infiltration	Infiltration = rainfall - surface runoff
Redistribution	“The continued movement of water through a soil profile after input of water (via precipitation or irrigation) has ceased at the soil surface (Neitsch et al., 2011).”
Evapotranspiration	“All processes by which water in the liquid or solid phase at or near the earth's surface becomes atmospheric water vapor (Neitsch et al., 2011).”
Lateral Subsurface Flow	“streamflow contribution which originates below the surface but above the zone where rocks are saturated with water (Neitsch et al., 2011).”
Surface Runoff	SCS curve number method (USDA Soil Conservation Service, 1972) --OR-- Green & Ampt infiltration method (Green and Ampt, 1911)
Tributary Channels	Surface flow that is routed into the main channel. SWAT uses the tributary channels to compute the time of concentration for the subbasin and to compute transmission losses (USDA Soil Conservation Service, 1983)
Return Flow	Volume of streamflow originating from groundwater
Plant Growth	Used to assess removal of water and nutrients from the root zone, transpiration, and biomass/yield production Air temperature inputs are used to determine growing seasons
Erosion	Erosion and sediment yield are estimated for each HRU with the Modified Universal Soil Loss Equation (MUSLE) (Williams, 1975)
Nutrients	The movement and transformation of several forms of nitrogen and phosphorus in the watershed (Neitsch et al., 2011)
Pesticides	“SWAT simulates pesticide movement into the stream network via surface runoff and into the soil profile and aquifer by percolation (Neitsch et al., 2011).” Based on GLEAMS equations (Leonard et al., 1987)

After the land phase calculates the water, sediment, nutrient and pesticide loadings into the main channel, the routing phase begins to move those loadings into the stream network (See Table 2-4 for details on the routing methods used for each of the four types of loadings).

Table 2-4: Routing phase processes simulated by SWAT for each stream reach

Loading	Routing Method
Water/Flood	Muskingum Routing Method (Gill, 1978) --OR-- Variable Storage Coefficient Method (Williams, 1969)
Sediment	Amount of stream bed degradation and sediment transported in a stream segment is a function of the peak channel velocity and available stream power (Williams, 1980)
Nutrients	Nutrient simulation adapted from the QUAL2E model (Brown and Barnwell, 1987). Simulates the movement of Nitrogen and Phosphorus (in their various phases in streams and adsorbed to sediment).
Pesticides	Models the settling, burial, resuspension, volatilization, diffusion and transformation of both dissolved and sediment-attached pesticide compounds.

Each simulation produces a large number of output files which store the large amounts of information that SWAT produces. Each file represents the model outputs for one of the watershed subdivision units outlined above (i.e. full watershed, subbasins, HRUs, and stream reaches). These files are all written in ASCII text format and contain either daily, monthly, or yearly time steps for a wide range of variables for each unit (subbasin, HRU, or stream segment). Among the many output files produced by SWAT, the primary ones are:

- output.sub - time series data for each subbasin
- output.hru - timeseries data for each HRU
- output.rch - timeseries data for each stream reach (Arnold et al., 2012).

Figure 2-1 below is an example of what the three primary output files look like. For more information on the SWAT output files, see the SWAT Input/Output Documentation at <https://swat.tamu.edu/documentation/2012-io/>.

REACH	GIS	MON	AREAKm2	FLOW_INcms	FLOW_OUTcms	EVAPcms	TLOSScms	SED_INTons	SED_OUTtons	SEDCONCmg/kg	ORGN_INkg	ORGN_OUTkg	ORGP_INkg	ORGP_OUTkg	N03_INkg	N03_OUTkg
REACH 1	0	1	0.1078E+04	0.2457E+01	0.2385E+01	0.8626E-01	0.0000E+00	0.7043E-06	0.0000E+00	0.0000E+00	0.3199E-02	0.0000E+00	0.9597E-02	0.0000E+00	0.8221E+01	0.7939E+01
REACH 2	0	1	0.7273E+03	0.1920E+01	0.1895E+01	0.2457E-01	0.0000E+00	0.6803E-06	0.6364E-06	0.1292E-06	0.2240E-02	0.0000E+00	0.6720E-02	0.4090E-02	0.1208E+02	0.1191E+02
REACH 3	0	1	0.3693E+03	0.1115E+00	0.1007E+00	0.1145E-01	0.0000E+00	0.1135E-06	0.0000E+00	0.0000E+00	0.9045E-03	0.5847E-03	0.2714E-02	0.1465E-02	0.9057E-03	0.8003E-03
REACH 4	0	1	0.7224E+03	0.2819E+00	0.2497E+00	0.3700E-01	0.0000E+00	0.1002E-06	0.0000E+00	0.0000E+00	0.1526E-02	0.2991E-03	0.4579E-02	0.2214E-02	0.1529E-02	0.3360E-01
REACH 5	0	1	0.1119E+04	0.3785E+00	0.3189E+00	0.6593E-01	0.0000E+00	0.1634E-06	0.0000E+00	0.0000E+00	0.2503E-02	0.7210E-03	0.7509E-02	0.3353E-02	0.9055E+00	0.1626E+01
REACH 6	0	1	0.2662E+03	0.7860E-01	0.7667E-01	0.1929E-02	0.0000E+00	0.5567E-07	0.5567E-07	0.2909E-06	0.7077E-03	0.6565E-03	0.2123E-02	0.1919E-02	0.7085E-03	0.6894E-03
REACH 7	0	1	0.1189E+04	0.3535E+00	0.3464E+00	0.7090E-02	0.0000E+00	0.1240E-06	0.0000E+00	0.0000E+00	0.9362E-03	0.4762E-03	0.3836E-02	0.3368E-02	0.3445E-01	0.7551E-01
REACH 8	0	1	0.2331E+04	0.6734E+00	0.5827E+00	0.1107E+00	0.0000E+00	0.1579E-06	0.0000E+00	0.0000E+00	0.2241E-02	0.5236E-04	0.8972E-02	0.3905E-02	0.3320E+00	0.7426E+00
REACH 9	0	1	0.1487E+04	0.4512E+00	0.4202E+00	0.3138E-01	0.0000E+00	0.1125E-06	0.0000E+00	0.0000E+00	0.1437E-02	0.5523E-03	0.6201E-02	0.3962E-02	0.7650E-01	0.3303E+00
REACH 10	0	1	0.3098E+03	0.1028E+00	0.1005E+00	0.2320E-02	0.0000E+00	0.1136E-06	0.1136E-06	0.4557E-06	0.7870E-03	0.7318E-03	0.2361E-02	0.2140E-02	0.7879E-03	0.7682E-03
REACH 11	0	1	0.2547E+04	0.7391E+01	0.7154E+01	0.3470E+00	0.0000E+00	0.5159E-06	0.0000E+00	0.0000E+00	0.6583E-02	0.0000E+00	0.1975E-01	0.0000E+00	0.1685E+02	0.2149E+02
REACH 12	0	1	0.2649E+03	0.4580E+00	0.4552E+00	0.2016E-02	0.0000E+00	0.9299E-06	0.9299E-06	0.0001E-06	0.8211E-03	0.0000E+00	0.2463E-02	0.2305E-02	0.1620E+01	0.1618E+01
REACH 13	0	1	0.4031E+03	0.6558E+00	0.6380E+00	0.1736E-01	0.0000E+00	0.4923E-06	0.4923E-06	0.3098E-06	0.1220E-02	0.3480E-04	0.3661E-02	0.3183E-02	0.1169E+01	0.1141E+01
REACH 14	0	1	0.2818E+04	0.7621E+01	0.7610E+01	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 15	0	1	0.2317E+04	0.4646E+01	0.4521E+01	0.1233E+00	0.0000E+00	0.6835E-06	0.0000E+00	0.0000E+00	0.1194E-02	0.0000E+00	0.7671E-02	0.0000E+00	0.2346E+02	0.2281E+02
REACH 16	0	1	0.2542E+03	0.8445E+00	0.8442E+00	0.3420E-03	0.0000E+00	0.2063E-06	0.2063E-06	0.9736E-07	0.7288E-03	0.0000E+00	0.2186E-02	0.1055E-02	0.1818E+01	0.1760E+01
REACH 17	0	1	0.6354E+03	0.3544E+00	0.3234E+00	0.3081E-01	0.0000E+00	0.2381E-06	0.0000E+00	0.0000E+00	0.1859E-02	0.1288E-02	0.5578E-02	0.3749E-02	0.1896E+01	0.1760E+01
REACH 18	0	1	0.3248E+04	0.6087E+01	0.5907E+01	0.2007E-01	0.0000E+00	0.7330E-06	0.0000E+00	0.0000E+00	0.6557E-03	0.0000E+00	0.4737E-02	0.0000E+00	0.4040E+02	0.4700E+02
REACH 19	0	1	0.2692E+04	0.5457E+01	0.5287E+01	0.1907E-01	0.0000E+00	0.5159E-06	0.0000E+00	0.0000E+00	0.6583E-02	0.0000E+00	0.1975E-01	0.0000E+00	0.1685E+02	0.2149E+02
REACH 20	0	1	0.4150E+03	0.5967E+00	0.5797E+00	0.2016E-02	0.0000E+00	0.9299E-06	0.9299E-06	0.0001E-06	0.8211E-03	0.0000E+00	0.2463E-02	0.2305E-02	0.1620E+01	0.1618E+01
REACH 21	0	1	0.8851E+03	0.2632E+00	0.2632E+00	0.1736E-01	0.0000E+00	0.4923E-06	0.4923E-06	0.3098E-06	0.1220E-02	0.3480E-04	0.3661E-02	0.3183E-02	0.1169E+01	0.1141E+01
REACH 22	0	1	0.4670E+04	0.1184E+01	0.1184E+01	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 23	0	1	0.5599E+04	0.1274E+01	0.1274E+01	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 24	0	1	0.2787E+03	0.7711E+00	0.7711E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 25	0	1	0.3690E+04	0.8944E+00	0.8944E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 26	0	1	0.4643E+04	0.9224E+00	0.9224E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 27	0	1	0.3074E+03	0.5722E+00	0.5722E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 28	0	1	0.2568E+03	0.6884E+00	0.6884E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 29	0	1	0.4659E+03	0.1341E+00	0.1341E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 30	0	1	0.8332E+03	0.2974E+00	0.2974E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 31	0	1	0.6430E+04	0.1434E+00	0.1434E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 32	0	1	0.8366E+04	0.1644E+00	0.1644E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 33	0	1	0.8902E+03	0.2044E+00	0.2044E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 34	0	1	0.4893E+04	0.6783E+00	0.6783E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 35	0	1	0.5334E+04	0.1002E+00	0.1002E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 36	0	1	0.4072E+03	0.8609E+00	0.8609E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 37	0	1	0.1165E+04	0.3215E+00	0.3215E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 38	0	1	0.6011E+04	0.1102E+00	0.1102E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 39	0	1	0.4266E+03	0.1074E+00	0.1074E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 40	0	1	0.3211E+03	0.8815E+00	0.8815E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02
REACH 41	0	1	0.1855E+04	0.3754E+00	0.3754E+00	0.1143E-01	0.0000E+00	0.1854E-05	0.0000E+00	0.0000E+00	0.2097E-04	0.0000E+00	0.2448E-02	0.0000E+00	0.2311E+02	0.2333E+02

Figure 2-1: Example output.rch file and list of output variables

As the raw output files for SWAT are just text files with huge arrays of numbers, it can be difficult for people to read and make sense of the information. Because of the distributed, spatial nature of SWAT, GIS tools have been paired with it to provide pre/post processing interfaces to better visualize its data. There are a number of robust GIS software systems that

have been developed in SWAT's relatively short history to assist users in preparing input data, running the models, and then visualizing the model outputs. Currently, however, many of these systems are time and knowledge intensive to use, often are not interoperable with other software, and sometimes rely on license restricted programs like ArcGIS (Giuliani et al., 2013) that must be updated periodically on the user's desktop computers. The goal of this section is to review the most commonly used software systems for SWAT visualization and to then identify some limitations in functionality that this new modular web interface will hopefully fill to make SWAT data available to anyone interested regardless of technical background and financial/computational resource availability.

2.2 Desktop Applications

Throughout most of the time SWAT has been in use, desktop applications have been the only method for processing and viewing SWAT data and model outputs. There are a wide range of applications, software plugins, and standalone software programs that SWAT users can run from a local computer. The basic functionality of the most commonly used desktop programs and software plugins is outlined below.

2.2.1 ArcSWAT

ArcSWAT is an open source interface for SWAT built on ArcGIS (Winchell et al., 2010). This interface allows users of the ArcGIS software to delineate watersheds, and define the various input SWAT parameters from the map interface and to then run the SWAT simulation. The current version of ArcSWAT does not facilitate any output visualization (Dile et al., 2016), but with a complex workaround of exporting the output ArcSWAT data as a csv and then re-importing it into ArcMap using the attribute joining function in ArcGIS, the data can be viewed

spatially (Yu, 2018a). Visualizing the output data in plots requires external software such as MATLAB and Microsoft Excel (Olivera et al., 2006).

While ArcSWAT is currently the most commonly used interface to SWAT, is free as an extension, and is heavily supported by the SWAT community, the fact remains that to use the free extension a license of ArcGIS is required. That it does not facilitate any output viewing limits its effectiveness in making SWAT results accessible to decision-makers and stakeholders. Even for licensed ArcGIS users, ArcSWAT is currently not compatible with the newest version of ArcGIS (Current version of ArcGIS is 10.6, ArcSWAT is only compatible with ArcGIS 10.4 and older) which means that an even smaller amount of people has access to it unless they're willing to downgrade their ArcGIS subscription.

2.2.2 MWSWAT

As a part of a project for the United Nations University and in an effort to provide free, open source access to SWAT's capabilities as a decision support tool, Chris and Leon developed a SWAT interface for MapWindow GIS software (Ames et al., 2008) called MapWindow SWAT or MWSWAT (Chris and Leon, 2008). Very similar to ArcSWAT, MWSWAT was built primarily for input data preparation and to enable users to run the model through the GIS interface and thus lacks the inherent ability to visualize the model outputs without external software and additional processing by the user. The fact that the most recent version is already ten years old suggests additional challenges associated with reliance on desktop solutions that in a relatively short time become out of date and difficult to maintain compatibility with model features.

2.2.3 QSWAT

To solve the software licensing issues apparent in the ArcSWAT interface and the lack of output visualization features in both ArcSWAT and MWSWAT (Francesconi et al., 2016), Dile et al., developed a new GIS interface for the SWAT model called QSWAT that is compatible with the QGIS software platform (Dile et al., 2016). QSWAT was developed to provide improved functionality while maintaining free access to the SWAT model. The QSWAT interface allows users to set up and run SWAT similar to ArcSWAT, but adds functionality to view the model outputs. Output visualization in QSWAT is available both spatially through the map interface through animated color changes for streams/subbasins based on time and modelled output values and also through a plot function that uses an external tool called SWATGraph to plot the timeseries data on a graph (Dile et al., 2018).

2.2.4 SWAT-CUP

The SWAT Calibration and Uncertainty Programs (SWAT-CUP) is another public domain program meant to provide SWAT access to users without an ArcGIS license (Francesconi et al., 2016). SWAT-CUP was not built for output visualization, as its primary purpose is to facilitate calibration, validation, and sensitivity analyses. However, it does contain multiple functions to graph output data for validation and calibration with observed data and also has a map viewer to view basic watershed characteristics (Abbaspour, 2013).

2.2.5 SWAT-GRASS

The Geographic Resources Analysis Support System (GRASS) (US Army, 1991) is yet another public domain software system that has SWAT capabilities through the development of the SWAT-GRASS plugin. Like the QSWAT software, SWAT-GRASS not only provides a GIS

interface for the preprocessing of input data and running of SWAT but it also contains a wide range of features for viewing and analyzing the model outputs. The output interface extracts output data from the ASCII text files and allows viewing of data in the form of time-series graphs, statistical plots, and maps (Srinivasan et al., 1996; Jayakrishnan et al., 2005).

GRASS is a fairly common GIS processing software available on the three major operating systems (macOS, windows, and linux) (Neteler et al., 2012) and it is well documented.

2.2.6 SWAT Output Viewer desktop application

The SWAT Output Viewer (Yu, 2018a) is a Windows compatible desktop application built specifically to read, analyze, and visualize the output files from SWAT model runs. It was built as a solution to the often complicated and time-consuming process of preparing data for visualization in SWAT GIS interfaces like ArcSWAT and QSWAT. It is a robust system that provides users with multiple data visualization and statistical data validation options and even allows for simple model runs all within the easy to use interface unburdened by being connected to a much larger and general-purpose GIS. Within the app, the map window allows users to view the spatial distribution of model outputs through color scaling. The user can then select any stream segment or subbasin boundary in the watershed and view a time-series plot for any of the modelled variables (Yu, 2018b).

While the SWAT Output Viewer is well documented, it appears that it is maintained solely by its original developer, without any apparent external uptake or support from the greater SWAT community. Another issue is that only Windows users have access to the app.

2.2.7 VizSWAT

VizSWAT (Baird & Associates, 2004) is another software tool built strictly for visualizing SWAT model outputs. It is a fully functional GIS interface that allows users to upload their own SWAT output data from any of the SWAT simulation interfaces (i.e. ArcSWAT, SWAT-CUP) and explore the data both spatially and temporally through map animations and time series plots. It also features data analysis functions such as time series aggregation, basic statistics, and correlation, frequency, base flow and flow duration calculations.

Unlike the SWAT Output Viewer VizSWAT is not open source and thus comes at a price. A single use license costs \$250 and an additional \$100 for every new version upgrade. However, VizSWAT is maintained by the core developers of SWAT itself (Srinivasan, 2004), thus the user support for VizSWAT may be worth the price tag for those serious about having a consistent SWAT output visualization software system.

2.3 Web interfaces and web services

The desktop applications and software packages outlined above have and will likely continue to play a key role in SWAT development and visualization, especially for those scientists and developers familiar with the model. However, there are limitations to how useful these tools can be for stakeholders without technical training in SWAT. Operating system compatibility, licensing costs, and software versioning issues can often inhibit users from having access to the newest or most robust technologies. In addition, the 'local' nature of these technologies causes issues when sharing data across disciplines or organizations is required. The technical expertise needed to create a custom SWAT model aside, using these desktop applications requires those interested in SWAT to have the computational and storage resources

locally to run the model and store all of the outputs. This severely limits the potential for data sharing and collaboration between users unless they are all looking at the same computer screen.

In recent years, there have been efforts to bring SWAT functionality to the web. These efforts were due, in part, to the need for more collaboration and data sharing between SWAT model developers and stakeholders. The following sections outline three web-based technologies that have been developed to make SWAT accessible to a wider range of people.

2.3.1 WRESTORE

The Watershed Restoration Using Spatio-Temporal Optimization of Resources (WRESTORE) tool is a web interface that allows members of a community to explore and optimize various conservation and land management practices within a watershed using a pre-calibrated SWAT model. The web interface allows for the user to specify different conservation practices and to then run a specific SWAT model based on those specifications. Complete with map views of the watershed and data charts, WRESTORE then serves as a model output viewer for the user to compare various model runs and decide which conservation practices are most effective on the area of study (Babbar-Sebens et al., 2015).

WRESTORE is a step in the right direction for making SWAT accessible to a broader group of watershed stakeholder communities. The concept of giving stakeholders (in this case farmers) a chance to compare the effects of different management practices on their land is an example of the potential for SWAT and other climate and hydrologic models to have greater impact. However, WRESTORE is currently only available for two watersheds in the United States, which while providing a model for the way forward still severely limits its influence on the global SWAT community.

2.3.2 OWS4SWAT

In an effort to simplify the often-tedious process of preparing SWAT model outputs for geospatial visualization, Giuliani et al. (2013) developed a proof of concept for a data publishing and visualization workflow called OWS4SWAT. This system takes SWAT output data files and publishes the data to database and web service systems that are compliant to the Open Geospatial Consortium (OGC) data sharing standards (Open Geospatial Consortium, 2004). It then uses those services to create geospatial visualizations of the data from both desktop and web-based GIS programs. OWS4SWAT leverages trusted open source and OGC compliant software such as GRASS and R for data processing, PostgreSQL/POSTGIS for data storage, and Geoserver for web map (WMS), feature (WFS), and processing (WPS) services.

The OWS4SWAT theory and methodology is a very valuable development for making SWAT available online to all interested parties. Admittedly, there is still much room for further development as the software package only works to publish and visualize the output.sub SWAT file geospatially and lacks the functionality to display data temporally. Though it is only a prototype and there does not seem to have been any developments since 2013, the OWS4SWAT methodology should be used as a guideline for any future developments in visualizing SWAT outputs through web services.

2.3.3 SWATShare

SWATShare is the gold standard for the current advances in making SWAT model data accessible on the web. Inspired by CUAHSI (Consortium of Universities for the Advancement of Hydrologic Science, Inc., <https://www.cuahsi.org/>) and their development of the Hydrologic Information System (Tarboton et al., 2009), HydroServer (Horsburgh et al., 2010) and

HydroDesktop (Ames et al., 2012) as hydrologic data sharing software, SWATShare was built to allow those with the technical expertise in developing and calibrating SWAT models to share their work with the broader community of scientists and stakeholders. The web interfaces for uploading, viewing, editing, running SWAT models, and then viewing the model outputs in both graphical and geospatial forms makes SWATShare the most complete tool for SWAT developers to share their data and for stakeholders to view it and use it to support their decisions.

SWATShare employs multiple front-end and back-end web services for handling the complexity involved in supporting both the pre and post processing of SWAT data. On the front-end, SWATShare uses the Flex software development kit (<http://flex.apache.org/>) for the user interface, geoserver for map visualization, and python and PHP for data plot generation. On the back-end, MySQL is used for data storage and Tomcat web services are used for communicating user data queries with the database and simulation commands to a high-performance computing resource hosted at Purdue university (Rajib et al., 2015). For those that go through the process of creating an account, SWATShare proves to be a very capable tool providing temporal and spatial visualization options for three of the four primary SWAT output files (output.std, output.sub, output.rch).

SWATShare is an ideal tool for those familiar with the development and use of SWAT. It allows for complete user control of custom SWAT models and provides a way to store and share that data with others. However, because of the breadth of options supported by the tool, information retrieval can be difficult for stakeholders (farmers, policy makers, etc.) who have not been trained in the intricacies SWAT.

2.4 Moving forward

From my review of the most relevant software systems for SWAT, there are two key features (modularity and data sharing) that, when combined will help make SWAT reach a new level of impact in the community of SWAT users. The modularity (i.e. think of the independent mobility of the apps on your smartphone) of the desktop applications allows them to meet the unique needs of specific areas, as there is not a one-size-fits-all solution for all situations, especially in hydrology. Where the desktop applications fall short is in their lack of data sharing capabilities and compatibility across operating systems. On the other hand, the open nature of the web technologies I reviewed facilitates a much higher level of scientist-stakeholder collaboration and data sharing than the technically exclusive desktop applications. However, these web interfaces for SWAT, particularly WRESTORE and SWATShare, are tied to the servers hosting them which limits their potential global impact.

I believe that by creating a new web application that leverages the data sharing capabilities of current web technologies and that can be duplicated, installed, and hosted anywhere, SWAT's ability to meet the unique data and modeling needs of stakeholders in any region will improve significantly.

2.5 Web applications in the Tethys Platform

The Tethys platform was originally built to lower the barrier of web application development for engineers and hydrologists so they could better share their models and data. Tethys combines many of the most commonly used technologies and scripting languages for database management, data processing, and web design into a relatively easy to use framework for developing web-based applications and decision support tools. The architecture used by

Tethys can be divided into three main components: Tethys SDK (Software Development Kit), Tethys Portal, and Tethys Software Suite. Each of these components in Tethys provides developers with all the tools needed to create robust, modular web-based tools for processing and sharing data.

Tethys web apps are developed in the Python programming language and a Software Development Kit (SDK). The SDK provides Python module links to each software component of the Tethys Platform, making the functionality of each component more accessible to “novice” programmers who have a working knowledge of Python but not all of the underlying web technologies used by Tethys. In addition, users can use any of the Python packages that they are accustomed to using in their scientific Python scripts to control their web apps.

The second component, Tethys Portal, is where developed apps are published. It acts as an app landing page for the users to access installed apps. It also includes tools and functionalities to customize features such as user permissions, portal design, etc.

The final component, the Tethys Software Suite contains tools and packages, such as Geoserver, PostgreSQL, OpenLayers, etc., that are used commonly when developing geospatial web apps. Each of these tools has been built in to the overall functionality of Tethys, making it a very powerful tool, not only for displaying data, but also for any number of geoprocessing and database services. Figure 2-2 is a visual representation of the Tethys platform software architecture (Swain, 2015).

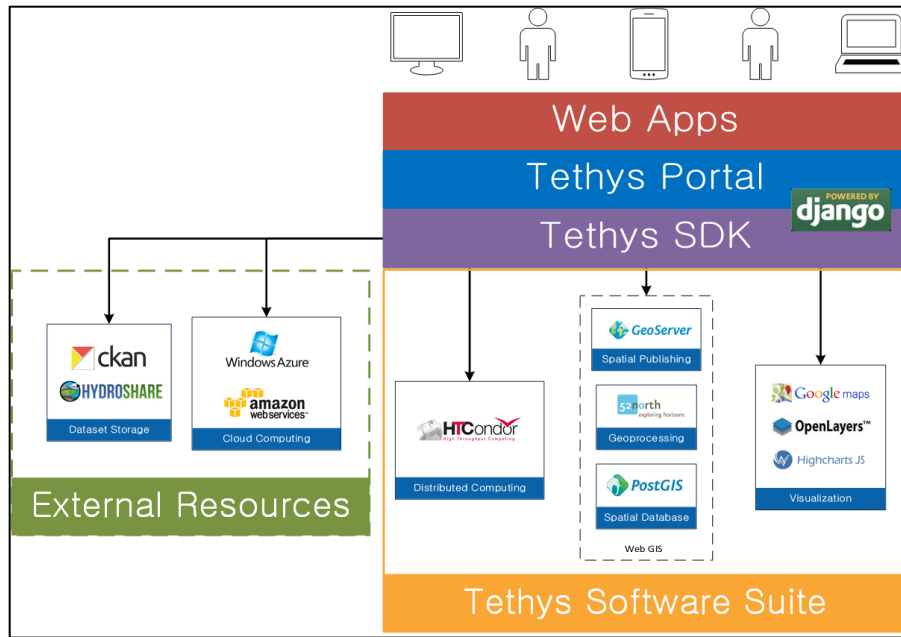


Figure 2-2: Tethys platform software architecture

2.5.1 Software organization

The Tethys application structure follows the Model-View-Controller (MVC) software architecture, see Figure 2-5. This allows for simple, readable and reusable code. The model is responsible for initializing the database and managing the database structure. The controllers are scripts written in Python that handle the logic in the web application and connect the data to the front end. Within these controller functions, scientists are able to leverage the wealth of already developed Python packages (e.g. pandas, gdal, numpy) along with their own customized scripts to control the data retrieval, processing, and analysis functions of an app. The data is then passed from the controller to the views. The views represent the HTML pages that are rendered for the user to see. The data from the controllers is passed in the form of context variables, meaning variables that are created every time the app is initiated, thus ensuring that the data is dynamic.

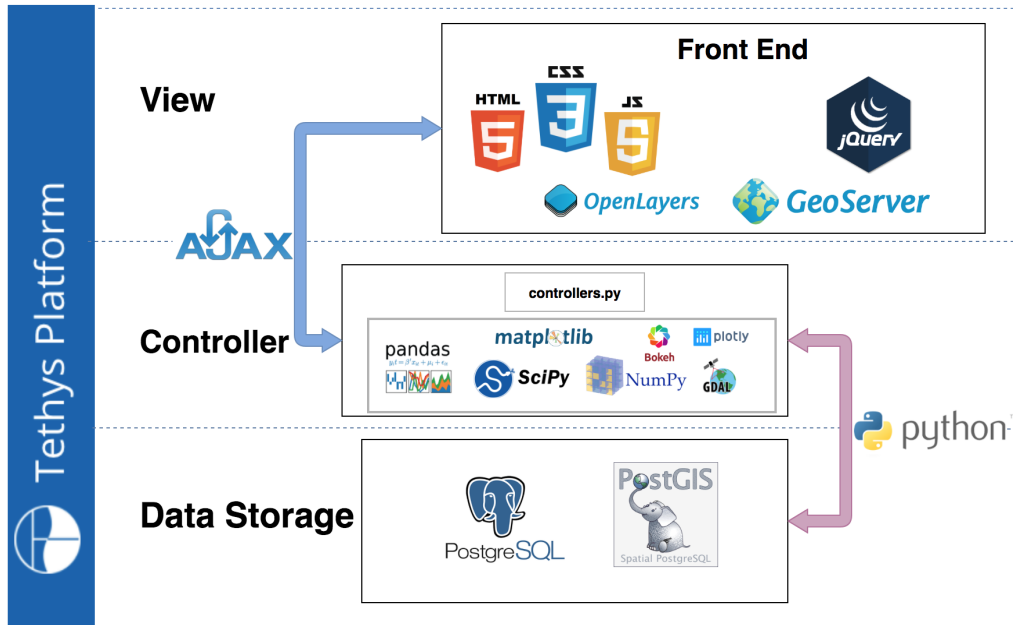


Figure 2-3: Example Model-View-Controller architecture

2.5.2 Tethys app folder structure

The Tethys App folder structure was designed for easy duplication and deployment. In the app package (a generic file structure is shown in Figure 2-6), each file plays a role in the making the app work. The controllers.py file contains all of the code directly related to the initial view of the app. It also manages all of the user actions within the app. In other words, it takes user inputs from the app interface, passes them into the various data processing python functions and then re-renders the app interface based on the output data of the functions called. The templates directory contains the HTML pages that are rendered to the front end. The public directory contains resources that are responsible for rendering the HTML content, such as JavaScript, Cascading Style Sheets (CSS), and images; it also contains any external libraries. The original Django app structure has several moving parts within the MVC architecture, and thus is

not straightforward for novice developers. The Tethys project app structure has all the MVC components within the app directory, making it easier for first-time web developers to leverage the MVC structure.

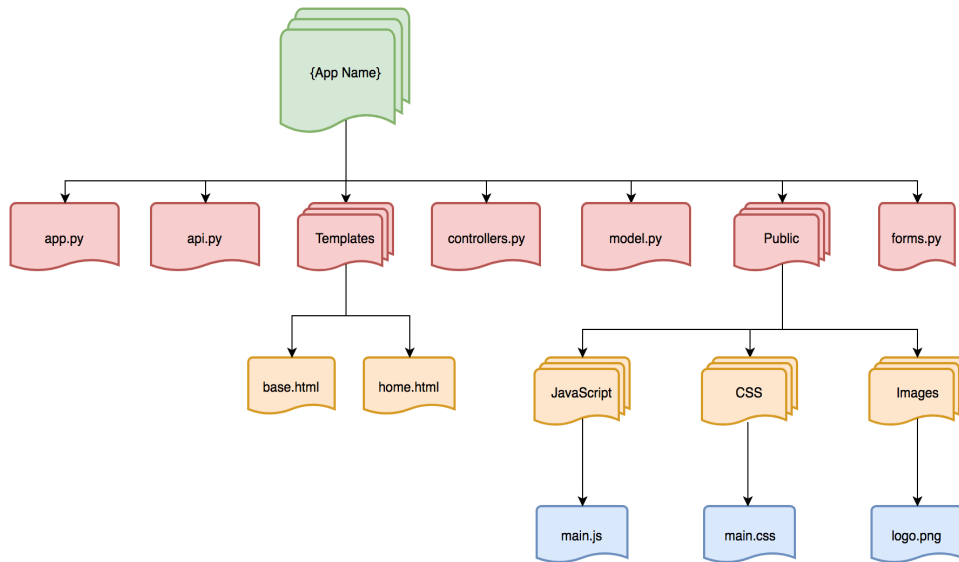


Figure 2-4: Generic Tethys app file structure

The complete documentation for the Tethys platform is available at <http://docs.tethysplatform.org/en/stable/index.html>.

2.6 Case study area: The Lower Mekong River Basin

The Lower Mekong River Basin (LMRB) is one of the largest, most culturally, climatically, geographically and biologically diverse river basins in the world. The Mekong River passes through 6 countries (China, Myanmar, Laos, Thailand, Cambodia, and Vietnam), each of which relies heavily on its flows for agriculture (rice paddies especially), fisheries, natural resources, and transportation. (MRC Mekong Basin, n.d). Because of the Mekong's

breadth of influence and the climatological diversity throughout mainland Southeast Asia, there are a number of issues that threaten the sustainability of its water resources and the diplomacy of the countries that it passes through.

The high seasonality, or disparity between the dry and wet seasons in Southeast Asia is a constant issue as many areas have to deal with heavy flooding and severe droughts on a yearly basis (Jacobs, 2002). According to the Mekong River Commission (MRC), an intergovernmental organization set up as a partnership between Thailand, Laos, Cambodia, and Vietnam to manage the Mekong river, 80-90% of the river's flow occurs during the wet season of June to November (MRC Hydrology, n.d). While flooding causes expensive damages in most areas, there are still many areas throughout Southeast Asia that heavily rely on these floods for rice paddy irrigation. For each area within the region to get the water that it needs and to make sure that areas that don't need it don't get flooded (See 2011 floods in Thailand and 2018 dam break in Laos), effective water storage planning is a necessity (MRC Floods & Droughts, n.d.). With the ever-increasing population and the higher energy and food demands that come with it, this planning is becoming ever the more difficult.

Human factors threaten the resources that the Mekong provides. For example, fisheries along the river provide much of the food consumed throughout the region but increased agriculture demands, domestic water supply, and industrial use (i.e. hydropower dams/plants) threaten to diminish the flow and fish supply to those fisheries (MRC Fisheries, n.d.). The cascades of dams that are being built throughout the region, mainly in the upper regions (Jacobs, 2002; Piman et al., 2013; Robert, 2017) have shown to have significantly decreased river discharge downstream (Räsänen et al., 2016).

With the human and climate changes that are now occurring, it is becoming more and more important for the governments and organizations charged with managing the flows and the resources provided by the Mekong to work together for the common good of the people in the region. The political nature of transboundary water resources makes the sharing of information and planning methods difficult, but the sustainability of the Mekong depends on the diplomacy and willingness to compromise of its riparian countries (Sneddon & Fox, 2006). In the MRC's 2017 annual report they outlined some of the key outcomes they wanted to achieve in the coming year. Two of the outcomes that align with the need for better data sharing and decision support tools discussed in chapter 1 are: (1) "Increased common understanding and application of evidence-based knowledge by policy makers and project planners" and (2) "Basin-wide monitoring, forecasting, impact assessment and dissemination of results strengthened for better decision-making by Member Countries (MRC, 2017)."

SWAT is currently being used for watersheds, large and small, throughout the LMRB. Various organizations, including MRC, have their own SWAT models that they use for basin wide water resource management. Mohammed et al. (2018), a research team from the NASA Goddard Space Flight Center, have developed a new calibrated SWAT model for the entire LMRB to hopefully replace the current version being used by MRC. This new model uses an updated land use/land cover map and satellite precipitation and air temperature observation data as the climate inputs instead of the spatially inconsistent in-situ observation data that are generally used in the region. They argue that using the new land use/land cover map and satellite data instead of in-situ measurements has drastically improved model accuracy for the Mekong River.

The input and output files from this new Lower Mekong model were provided to me by the team at NASA Goddard as a case study watershed for the building of the prototype web applications that will be described in the following chapter.

3 PROVIDING ACCESS TO SWAT INPUTS AND OUTPUTS THROUGH THE TETHYS PLATFORM

3.1 Introduction

In mid 2011, an unusually heavy monsoon season led to months of severe flooding throughout the entire Mekong basin along with other nearby basins. This flooding caused serious financial loss, infrastructural damages, and, in many cases, loss of life throughout the Lower Mekong's riparian countries (USAID, 2011). In the Mekong River Commission's annual report for 2011 they used this flood event to illustrate the need for better data management technologies and systems to increase public awareness about the risks of severe hydrologic events (MRC, 2011).

In this chapter, two new web applications will be introduced. The first application is a web interface for the nasaaccess software which acts as a data portal for those interested in accessing climate data from NASA with limited time or technical background. The second application is a fully customizable interface for visualizing, processing, and downloading SWAT model outputs. I believe that the applications I have created for nasaaccess and SWAT can not only be used by scientists to analyze past events like the 2011 floods but they can also be used to provide decision makers and water managers with simple data visualization tools to use while engaging with stakeholders.

The figures shown throughout this chapter show SWAT input data and outputs for the outlet of the Xe Don River near Pakse, Laos which was one of the areas that experienced heavy flooding during the monsoon season in 2011. This data is part of the Lower Mekong SWAT model created by Mohammed et al. (2018) that was discussed in Chapter 2.

3.2 Accessing SWAT inputs through the nasaaccess web application

The nasaaccess web application features an aesthetic, uncomplicated interface for submitting data processing jobs to the server, and then provides feedback for the user to know that it is working as expected. The main view of the app (Figure 3-1) features two panes. The left pane contains all of the user options, each of which relates directly to one of the input arguments of the nasaaccess functions. The right pane displays a map and the watershed boundary for which the user has selected to access data.

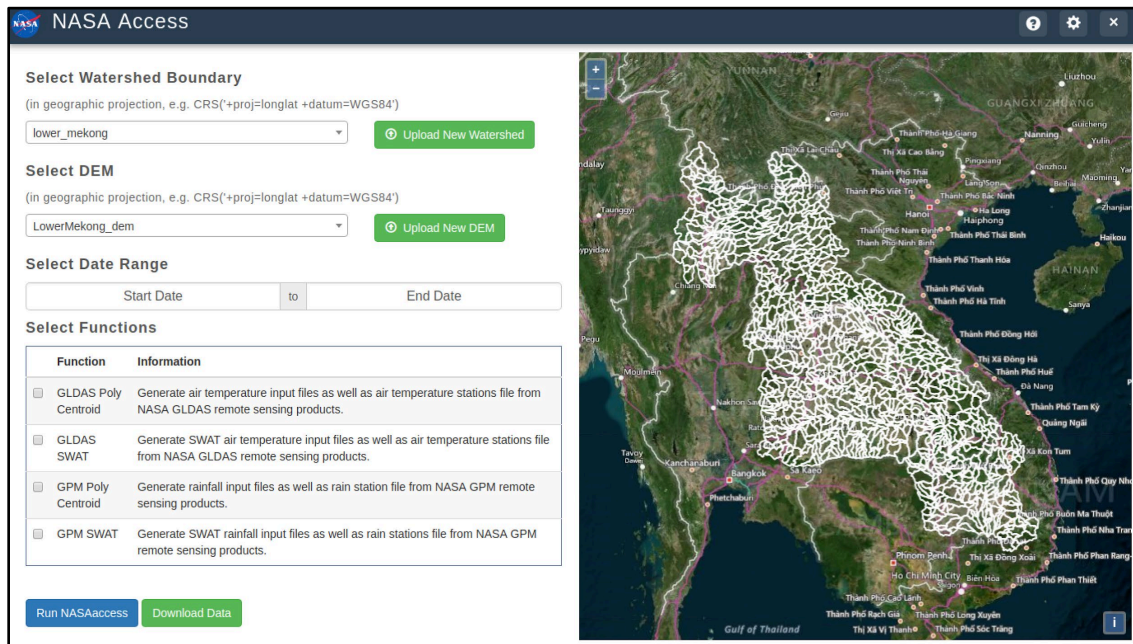


Figure 3-1: nasaaccess app user interface

The main purpose of the nasaaccess web application is to give users access to the nasaaccess data processing functions. Thus, the key functionality of the app resides in the four input elements in the left pane of the app. In the left pane of the app (Figure 3-2), the user may select the watershed boundary, DEM, and date range to be used as input arguments in the nasaaccess functions. The user is also given the option to select one or multiple nasaaccess functions to run using those arguments. The app uses these input elements to pass user requests as input arguments into the nasaaccess functions.

If a user has a shapefile or DEM file for an area not currently available in the app, by selecting the “Upload New Watershed” and “Upload New DEM” buttons there is also an option for them to upload their own files temporarily to the server and use them as inputs in the nasaaccess functions.

The screenshot shows the NASA Access web application interface. At the top, there is a NASA logo and the text "NASA Access". Below this, there are four main sections for user input:

- Select Watershed Boundary:** A dropdown menu with "lower_mekong" selected and a green button labeled "Upload New Watershed".
- Select DEM:** A dropdown menu with "LowerMekong_dem" selected and a green button labeled "Upload New DEM".
- Select Date Range:** Two date input fields with "Jun 1, 2011" and "Oct 31, 2011" entered, separated by a "to" label.
- Select Data:** A table with two columns: "Dataset" and "Information".

Dataset	Information
<input type="checkbox"/> GLDAS Poly Centroid	Generate air temperature input files as well as air temperature stations file from NASA GLDAS remote sensing products.
<input checked="" type="checkbox"/> GLDAS SWAT	Generate SWAT air temperature input files as well as air temperature stations file from NASA GLDAS remote sensing products.
<input type="checkbox"/> GPM Poly Centroid	Generate rainfall input files as well as rain station file from NASA GPM remote sensing products.
<input checked="" type="checkbox"/> GPM SWAT	Generate SWAT rainfall input files as well as rain stations file from NASA GPM remote sensing products.

At the bottom of the interface, there are two buttons: "Run NASAaccess" (blue) and "Download Data" (green).

Figure 3-2: Left pane of nasaaccess interface with user inputs

After selecting inputs and clicking the “Run nasaaccess” button, a modal appears requesting the email address of the user (Figure 3-3). Because the nasaaccess functions can take an extended length of time (depending on the size of the watershed and the date range selected), the app passes the user’s inputs to the server, initiates the nasaaccess functions, and then detaches those functions from the application so that the app isn’t tied up waiting for a single process to run. By obtaining the user’s email address, the app can notify the user once the processing of their data has been completed (Figure 3-4).

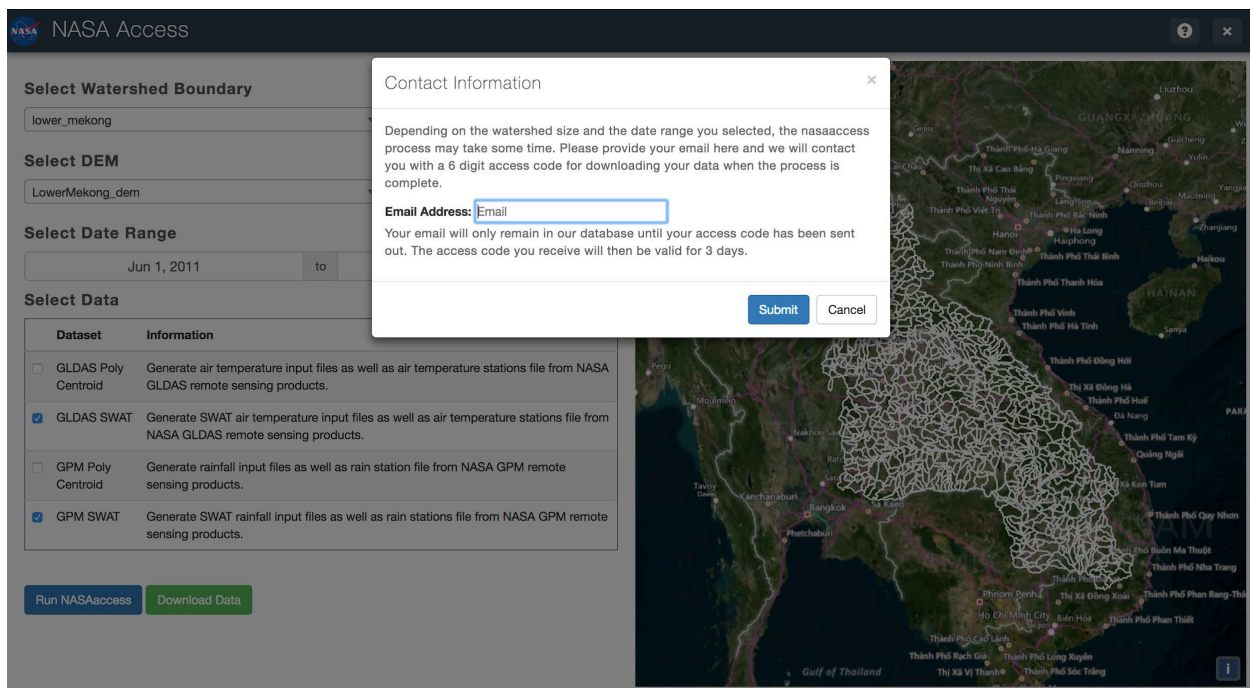


Figure 3-3: Contact Information modal for email submittal

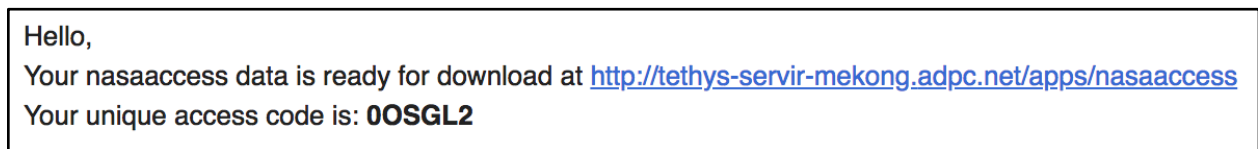
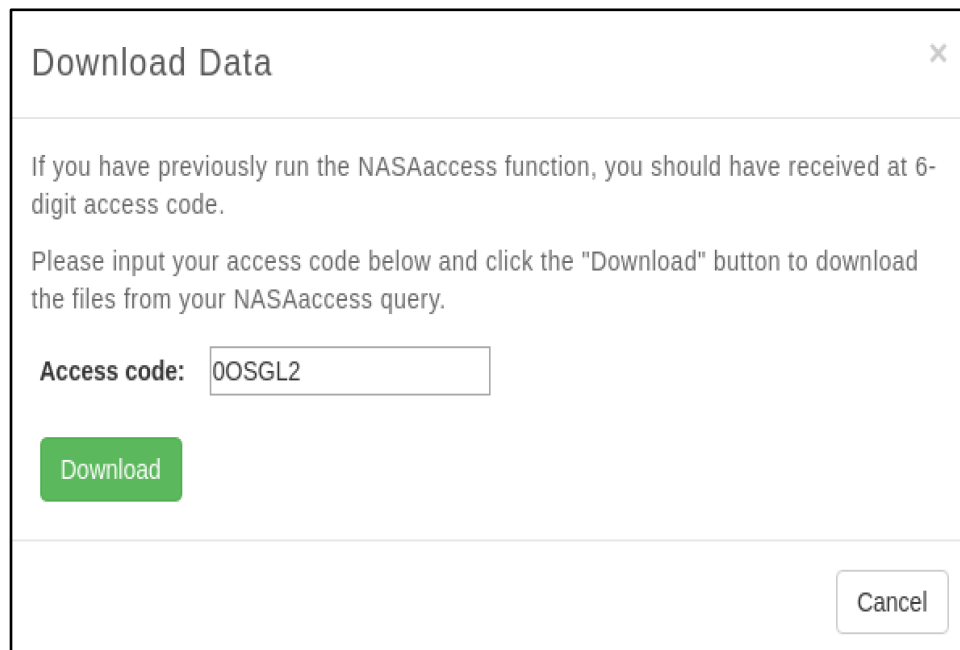


Figure 3-4: Example email message notifying the user that their data is ready for download

Once a user has received an email with an access code, they can return to the app and click the “Download Data” button. This opens the “Download Data” modal (Figure 3-5) that contains a text input field for the user to input the access code from the email. After entering the access code and clicking the “Download” button, the download dialogue box from the user’s web browser appears for the user to specify the location that they would like to save their nasaaccess files.



Download Data

If you have previously run the NASAaccess function, you should have received a 6-digit access code.

Please input your access code below and click the "Download" button to download the files from your NASAaccess query.

Access code:

Download

Cancel

Figure 3-5: Download Data modal

The files are saved as a zip file which contains subfolders containing the actual data files for each nasaaccess function that was completed. Figure 3-6 and Figure 3-7 show an example file structure and an example data file, respectively.

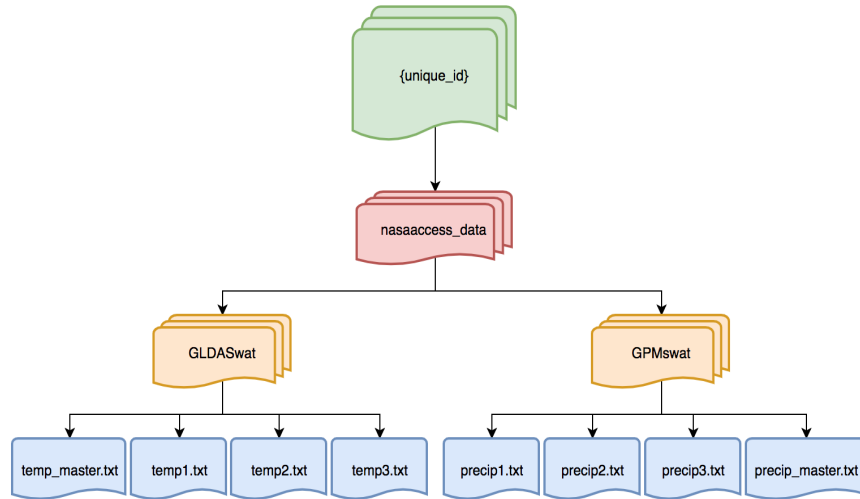


Figure 3-6: Example file structure of nasaaccess output files

```

20110601
1.949999928474426270e+00
0.000000000000000000e+00
2.775000000000000000e+01
2.639999866485595703e+00
6.683999633789062500e+01
1.341000080108642578e+01
7.626000213623046875e+01
5.100000381469726562e+00
0.000000000000000000e+00
5.850000381469726562e+00

```

Figure 3-7: Example GPMswat output file starting on June 1, 2011

An in-depth description of all the code involved in the nasaaccess app can be found in Appendix A.

3.3 Viewing and accessing SWAT outputs through the SWAT Data Viewer application

The SWAT Data Viewer was developed to act as a virtual and open access data store for stakeholders and decision-makers to view and download SWAT model inputs and outputs for their area of interest. Unlike many of the SWAT related web applications reviewed in Chapter 2, this application is completely modular meaning that it can be duplicated, customized and served from any server running the Tethys platform and can display the data from any valid SWAT model.

3.3.1 Home view

The initial view of the app, shown in Figure 3-8, allows the user to select a watershed (the list of available watersheds is produced based on the folders within the “swat_data” file path defined in config.py file) and view the various spatial files (subbasins, stream lines, lulc, and soil) for that watershed. In the navigation pane on the left, the user can toggle the visibility of each of the different layers (Figure 3-9).

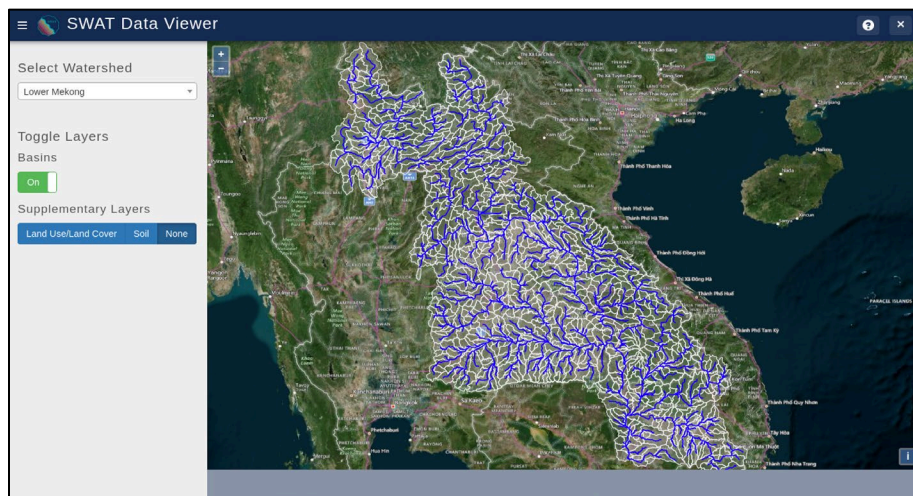


Figure 3-8: Initial view of the SWAT Data Viewer

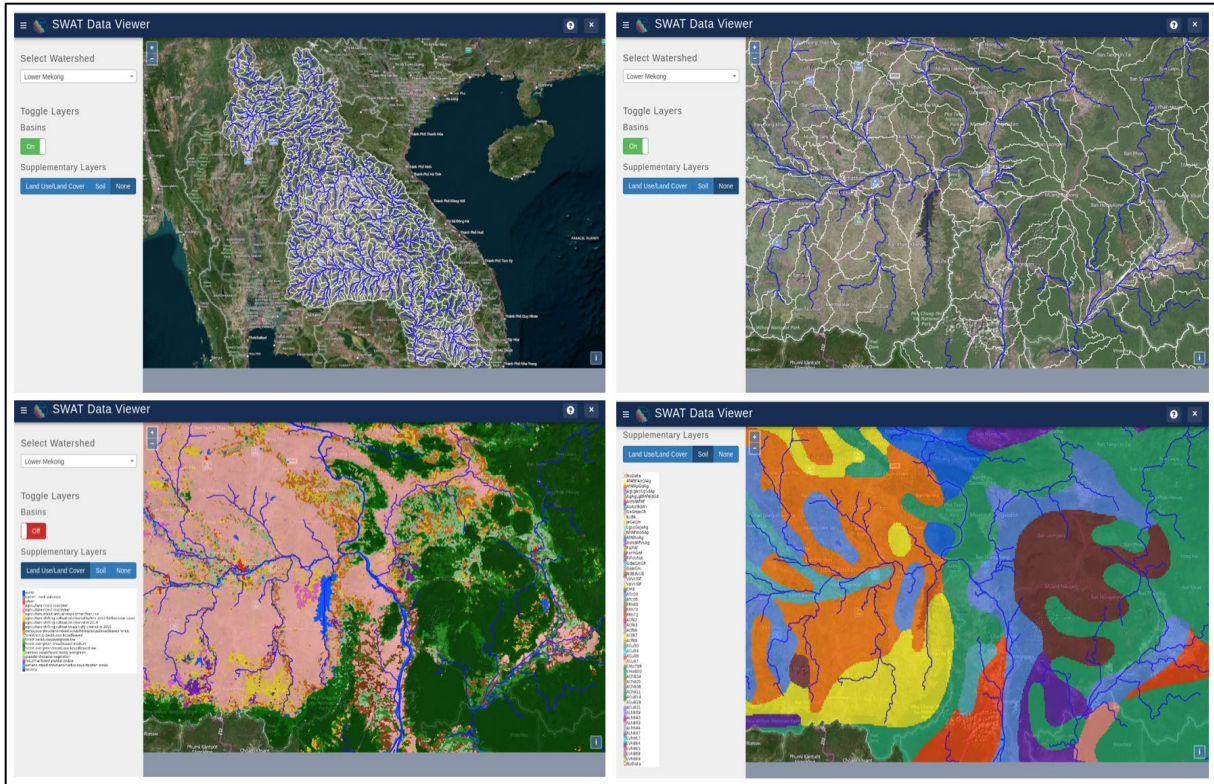


Figure 3-9: Configurations and layer visualizations in the SWAT Data Viewer

3.3.2 Feature selection

Using the WMS capabilities of the subbasins and streams layers, the user can select any feature (subbasin or stream segment) and access further functionality within the app. When a feature is selected, the app reads the obtains the feature ID from its attributes. This ID corresponds to the subbasin and stream ID used in all of the SWAT output files so it is saved in the browser's session storage to be used in any of the user's data queries. With the feature ID, the app creates a list of all of the streams and subbasins upstream of the selected feature and creates new streams, subbasins, land use, and soil layers for the area selected by the user. While the app is creating the new layers, a data modal containing multiple tabs appears at the center of the screen and small maps are initialized on each of the tabs to display the new layers (Figure 3-10).

This modal is where the user can perform all of their data queries from the SWAT model outputs.

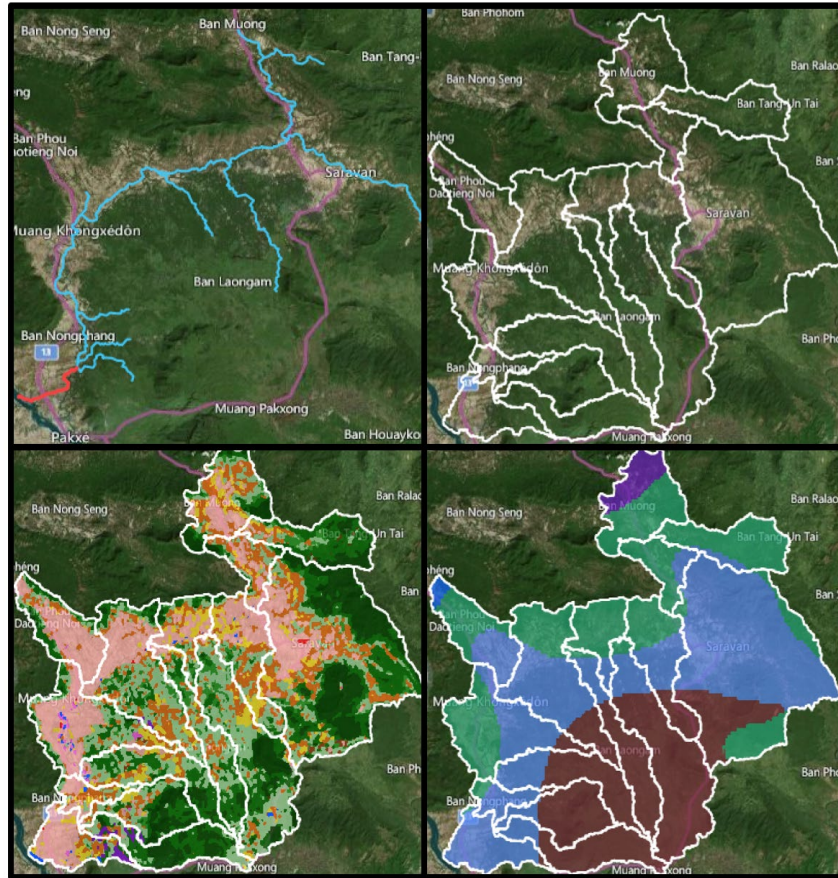


Figure 3-10: Maps displaying subset spatial data within the data modal

3.3.3 Data modal view

Once in the data modal (Figure 3-11), the user is given access to a wide range of data querying and analysis functions for the SWAT model outputs produced for the selected watershed. These functions are separated into multiple tabs within the modal so that the user can easily focus on the data that they're interested in.

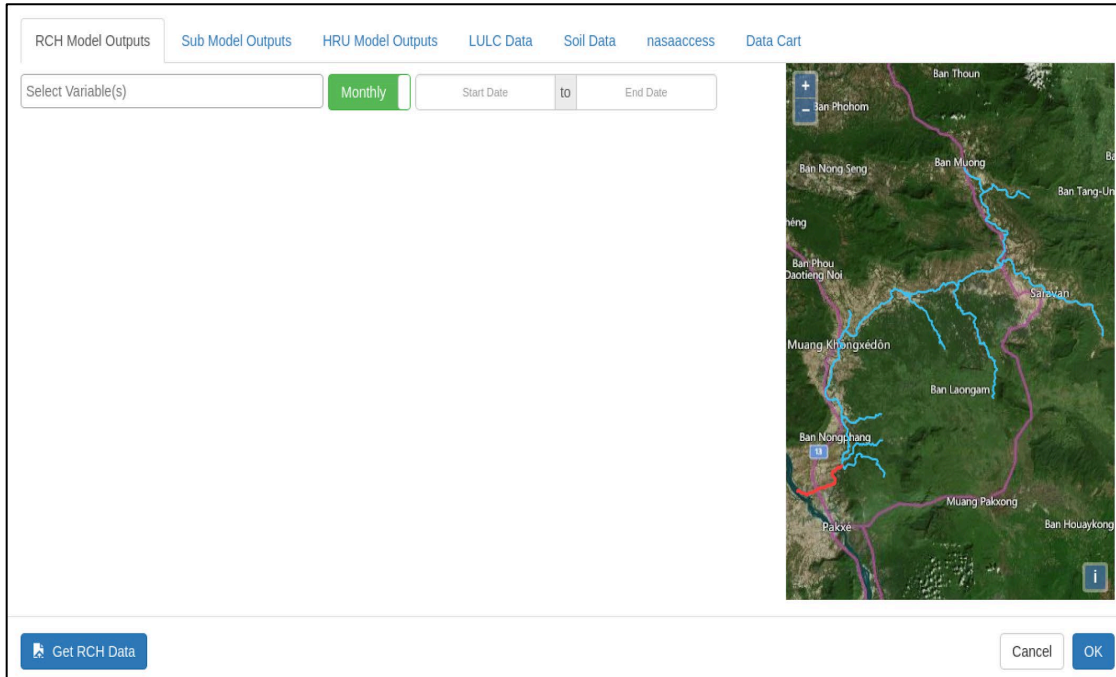


Figure 3-11: Initial view of the data modal after feature selection

The first two tabs correspond directly to the three main SWAT output files, output.rch and output.sub. Because the output files are formatted similarly, the “RCH Model Outputs” and “SUB Model Outputs” function in an almost identical way. Their only differences lie in the model variables that the user can select from.

Within each of these tabs (shown in Figure 3-12), the user will see a “Select Variable(s)” dropdown menu, start and end date selector dropdowns, a map view showing either the upstream stream segments (in RCH tab) or the upstream subbasins (in SUB tab) with the selected feature highlighted in red, and a button at the bottom to “Get {RCH or SUB} Data”.

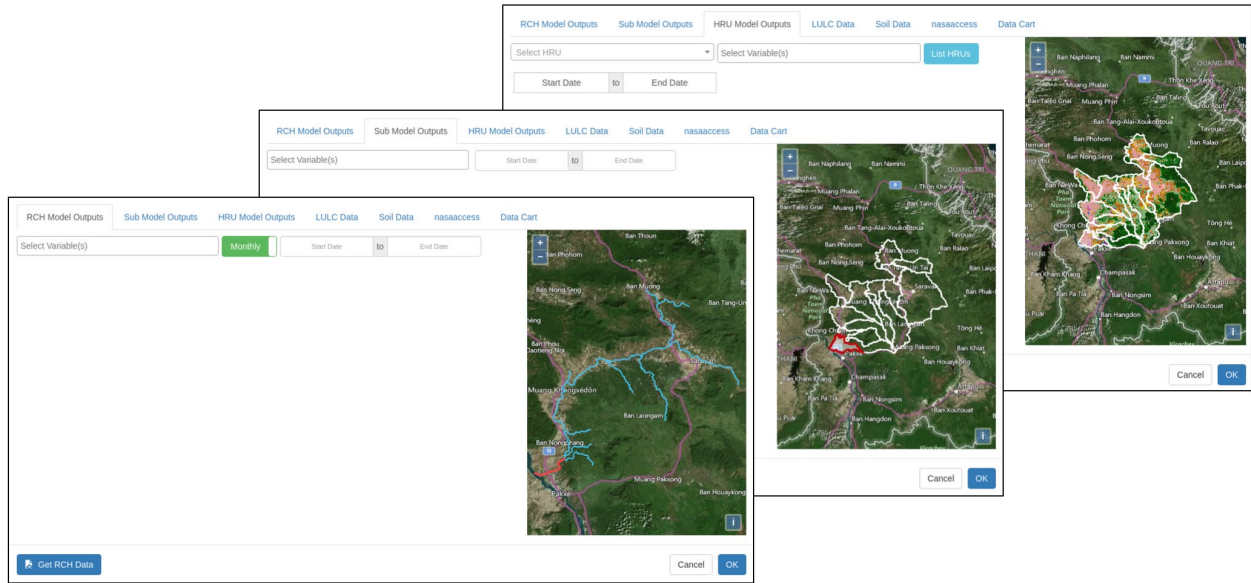


Figure 3-12: Tabs within data modal used SWAT Output visualization

As an example of how the SWAT Data Viewer can be used analyze a flood event, Figures 3-13 to 3-16 show various data queries for the Xe Don River outlet into the Mekong near Pakse, Laos during the 2011 monsoon season.

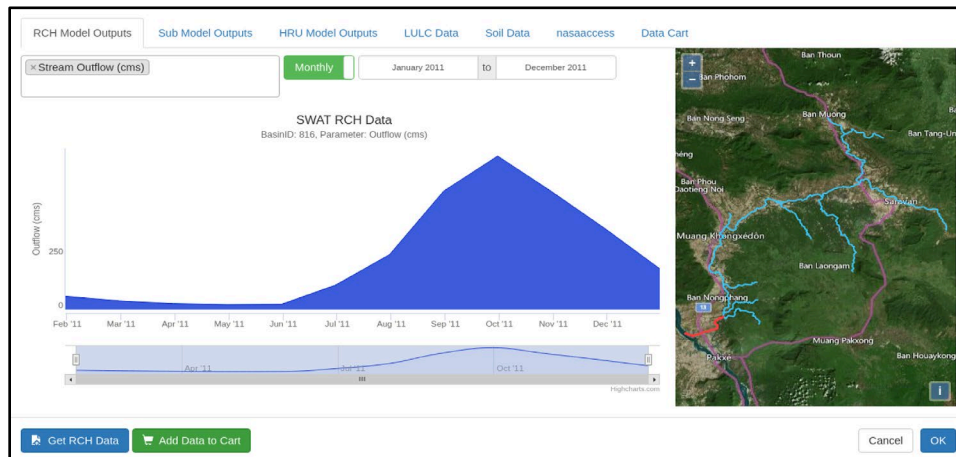


Figure 3-13: Monthly streamflow time series extracted from output.rch

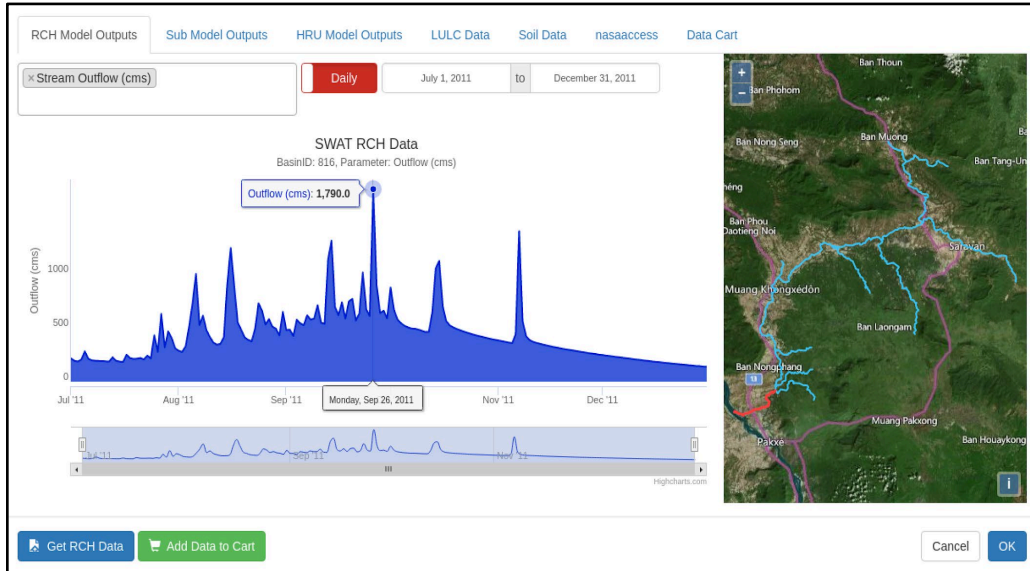


Figure 3-14: Daily streamflow time series extracted from output.rch



Figure 3-15: Daily precipitation time series extracted from output.sub



Figure 3-16: Daily surface runoff time series extracted from output.sub

The next two tabs in the data modal, “LULC Data” and “Soil Data” were built to give the user a better understanding of the land conditions and characteristics within the watershed that they selected. Within these two tabs, the user will see a map on the right-hand side with either the clipped LULC or Soil layer overlaid with the upstream subbasins layer. In the footer of the modal, a “Compute {LULC or Soil} Statistics” button is available for the user to click to compute the percent coverage of each different land use/land cover class or soil type within the selected area. This information is then displayed within the data modal in the form of a pie chart (Figure 3-17 and Figure 3-18). Each section of the pie chart is selectable, allowing the user to “drill down” into a land use class and view the statistics on the various subclasses within that class. For example, in Figure 3-17 below, the majority of the selected watershed is made up of forest (74.1%) and agriculture (25.5%). Of the agricultural land within the watershed, the map and pie chart show that the largest portion is made up of annual rice crops. This information can

be very valuable to stakeholders (policy makers, water managers, and farmers to name a few) as they try to assess the effects that hydrologic events will have on the land that they're responsible for.

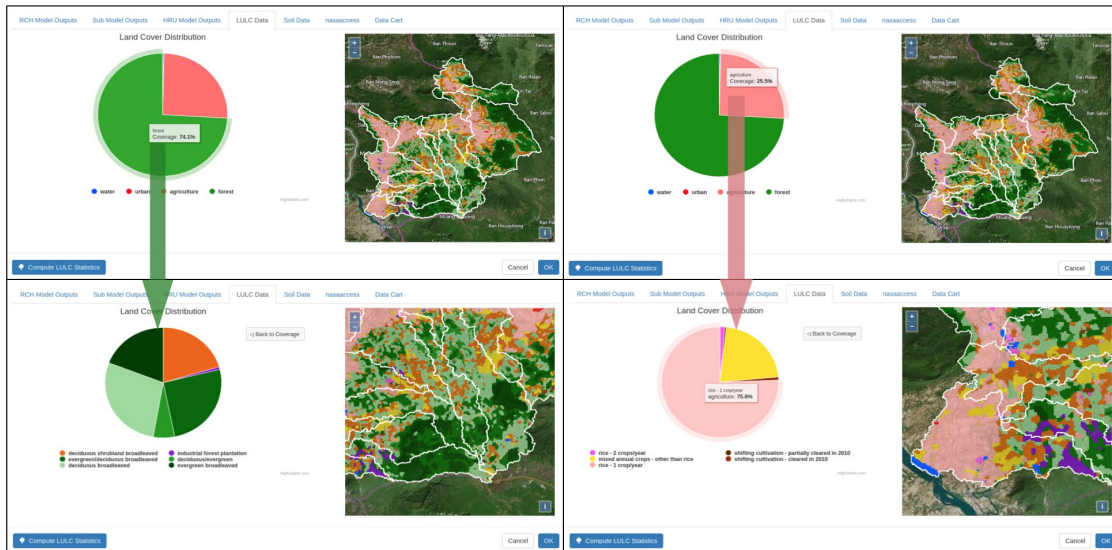


Figure 3-17: Land Use/Land Cover coverage statistics

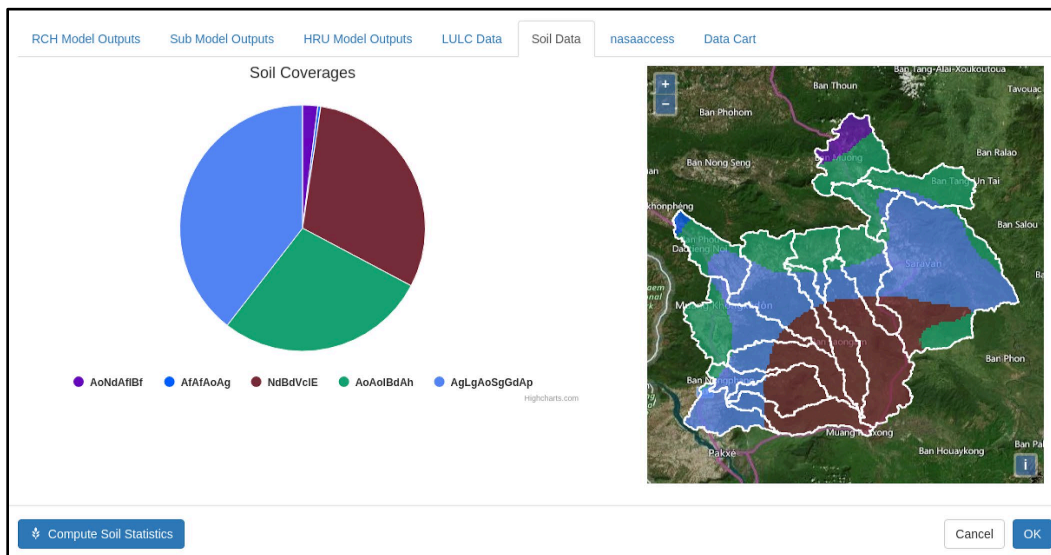


Figure 3-18: Soil type coverage statistics

UTC Offset (sec)	Date (m/d/y)	FLOW_OUTcms
1309478400	7/01/2011	217.2
1309564800	7/02/2011	194.6
1309651200	7/03/2011	187.6
1309737600	7/04/2011	206.7
1309824000	7/05/2011	281.9
1309910400	7/06/2011	214.8
1309996800	7/07/2011	198.6
1310083200	7/08/2011	195.1
1310169600	7/09/2011	193.2
1310256000	7/10/2011	192.7
1310342400	7/11/2011	189.1
1310428800	7/12/2011	188
1310515200	7/13/2011	226.4
1310601600	7/14/2011	194.5

Figure 3-20: Example csv file produced by the SWAT Data Viewer

The screenshot shows the 'Data Cart' tab in the SWAT Data Viewer. It contains two tables of data available for download.

Timeseries data available for download:

Data Type	Parameters	Time Step	Start Date	End Date	Stream/Basin ID
rch	FLOW_OUTcms	Monthly	012011	122011	816
rch	FLOW_OUTcms	Daily	07012011	12312011	816
sub	PRECIPmm	Daily	06012011	10312011	816
sub	SURQmm	Daily	06012011	10312011	816

Spatial data available for download:

Data Type	File Type	Outlet Stream ID
lulc	TIFF	816
soil	TIFF	816
reach_upstream	JSON	816
basin_upstream	JSON	816

At the bottom of the interface, there are buttons for 'Download', 'Cancel', and 'OK'.

Figure 3-21: The data cart tab in the SWAT Data Viewer

An in-depth description of all the code involved in the SWAT Data Viewer can be found in Appendix B.

3.4 Using output from the SWAT Data Viewer and nasaaccess apps for further analysis

The SWAT Data Viewer and nasaaccess applications not only provide the user with spatial and temporal visualization of SWAT related data within the apps, but the data download features within the apps allow for users to leverage other external technologies for further analysis. For example, the output csv files from the SWAT Data Viewer can be used in a wide range of statistical analyses using software like R or Microsoft Excel. Scientists and SWAT model developers can also take the spatial data (stream and subbasin json files and land use/land cover and soil raster files) from the SWAT Data Viewer and the climate data produced by the nasaaccess app and use them as the land and climate inputs for a new subsetted SWAT model for their selected area. Figure 3-22 and Figure 3-23 demonstrate how the data products of these two apps can be used for additional analysis and modeling.

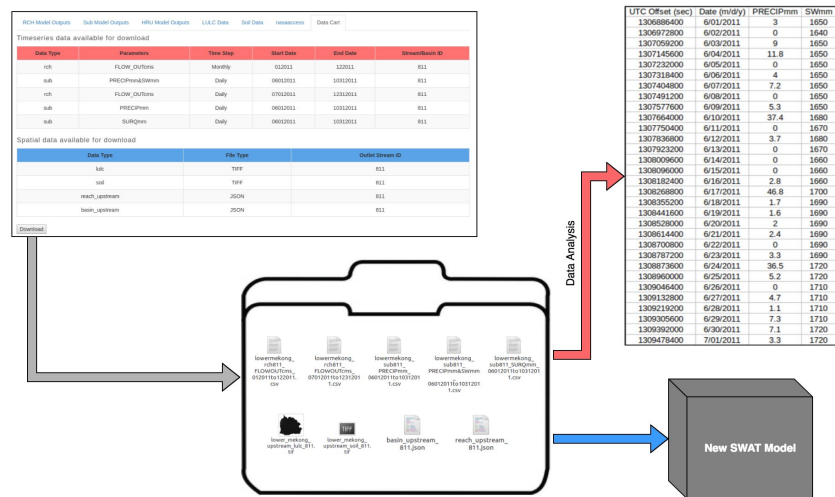


Figure 3-22: Hypothetical workflow for using output data from the SWAT Data Viewer

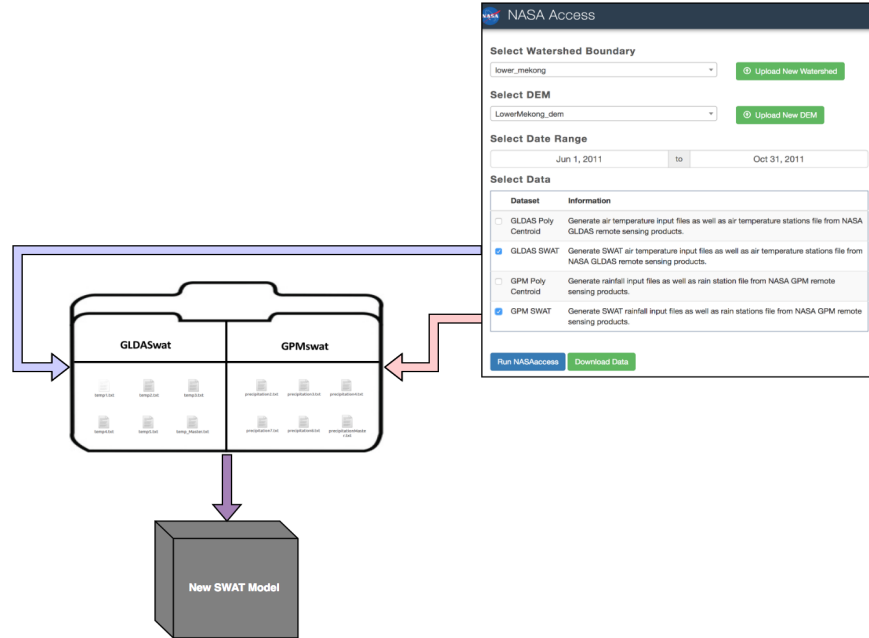


Figure 3-23: Hypothetical workflow for using nasaaccess outputs

3.5 Summary

The information that SWAT uses and produces has the potential to have a significant impact on the future decisions of water and climate scientists, policy makers, and communities around the world. However, the model’s impact has been limited because of the complicated nature of the model. While the technical barrier for creating and running SWAT models will likely never be eliminated completely, improving the way the model inputs and outputs are visualized and shared can increase SWAT’s impact substantially.

For scientists and SWAT developers, accessing inputs for SWAT can pose a challenge. The Global Precipitation Measurement (GPM) mission and Global Land Data Assimilation System (GLDAS), produced regularly by NASA on a global scale, contain valuable information on the current state of the world’s climate and can be used as inputs for a SWAT model.

However, the use of these earth observation datasets, available for free from the NASA earth data website (<https://earthdata.nasa.gov/>), has been limited due to the high technical requirements for accessing, processing, and understanding it. The work done by Mohammed et al (2018) to develop the nasaaccess workflow was an important step in opening these datasets up to those that could benefit from them. However, similar to the limitations of the datasets themselves, the nasaaccess software package has been limited in use because it is available only within the R software environment.

The nasaaccess web application was developed to lower the technical barrier that was keeping interested parties from accessing the powerful capabilities of the nasaaccess package and, ultimately, the information available in the GPM and GLDAS datasets. The nasaaccess web interface frees users from needing any technical background in R or earth observation data processing, yet still allows them to process large amounts of data into small, more manageable data files. The app produces data that is then ready to be used in a wide range of climatological and hydrological uses, including hydrologic modeling programs like SWAT, climate modeling, statistical analysis, and data visualization for decision support.

For decision makers and stakeholders in an area where the hydrologic process plays a key part in environmental, economic sustainability the SWAT model can provide crucial information to future planning and analysis of past events. However, due to high technical requirements in using SWAT, its impact on stakeholders and decision makers has been severely limited. The SWAT Data Viewer was designed to overcome the limitations of existing desktop and web technologies for SWAT visualization outlined in Chapter 2. Instead of being tied down with software licensing and versioning issues inherent in the current desktop applications, the SWAT Data Viewer is completely open source and was built to be resilient to changes in software. It is

also completely modular, unlike other SWAT related web interfaces, which means that the app can be duplicated, installed, and run from any machine (private or public, local or shared) with a Tethys Platform installed, making it adaptable to any specific user needs.

While these applications are limited to some extent in their functionality when compared to the programs and services outlined in Chapter 2, the customizability, modularity, and easy to use interfaces of these two apps have the potential to simplify SWAT data analysis and visualization for those lacking technical training in SWAT model development.

4 DISCUSSION

4.1 Summary

In this era of climate change, natural disasters, and public information it is becoming more and more important for people to have access to accurate and timely information. There is a wealth of information about the earth's climate, changes in weather patterns, and humankind's impact on the environment. This information comes in many forms. Satellite earth observations can provide scientists, policy makers, and stakeholders with both historical and near real-time representations of the earth. Additionally, the in-situ observations (e.g. stream or rainfall gauges) provide even more detailed information on the current and historical state of local systems. Modeling and data assimilation technologies continue to improve, thus increasing the amount of valuable information available for those that know how to use it

With the amount of climate data increasing daily, it leads to question what the end goal of this data is and whether it is reaching its mark? Proving or disproving climate change aside, one of the most apparent potential use cases for all this data is in providing decision makers and stakeholders (e.g. policy makers, disaster first responders, farmers, communities in areas prone to severe climate events) throughout the world with accurate information to guide future decisions. Organizations worldwide are investing exorbitant amounts of time and money into projects trying to transform raw climate data (in all of its forms) into clear, actionable information.

In this paper, I introduced my own efforts to make a very small amount of this data (i.e. model input and output data for the Soil and Water Assessment Tool) available to a wider audience. The two web applications I created, the nasaaccess app and SWAT Data Viewer, leverage the web-development and data processing technologies available in the Tethys Platform to simplify data visualization techniques and lower the technical barrier for accessing and using the information produced by SWAT.

4.2 Current limitations and future work

There is still room for improvement however. A key feature available in most of the SWAT enabled desktop and web technologies is the ability to visualize temporal data spatially through changing the symbology of the subbasin polygons and stream reach segments. This is a widely used method of viewing SWAT data and the SWAT Data Viewer currently is not able to do it. With that said, because the SWAT Data Viewer is already using PostgreSQL database technologies to store SWAT model outputs, spatiotemporal visualization should be seriously considered in future versions of the app.

4.3 Assessing impact

As organizations continue to fund projects like the SERVIR project in developing areas of the world, it is becoming more and more essential for there to be regular evaluations of the observed impacts in their work. The international nature common to these development projects—often with a combination of language, time, technical, and cultural barriers to overcome—can cause major communication problems between those managing the project from afar and the local organizations that they are serving. From my experience, these issues can and

often do lead to frustration on both sides and severely reduces the potential impact that any product might have on the end user.

To overcome these communication barriers and to make sure the needs of stakeholders are met, SERVIR has recently adopted a new service planning approach (SERVIR SCO, 2017). This service-centered approach was developed to better engage stakeholders in a region before, during, and after a project to ensure the work, trainings, and funding are reaching their mark. Before a project begins, needs assessments are completed to identify stakeholder interests and any limits in stakeholder technical knowledge. During a project, regular interaction between project managers, developers/scientists, and stakeholders is important to enable stakeholders to feel that they have a stake in the process and to ensure that the scientists are achieving, not overachieving, the end goal of the project. Following the completion of a project, stakeholders should be given everything that they need, technical training included, to fully adopt the results of the new project. For project managers and scientists alike, engaging stakeholders and implementing their suggestions and feedback throughout the process all but guarantees a project's success.

The theory behind this service planning approach is logical. It's easy to meet the needs of stakeholders when their voices are heard. Not only will their technical needs (i.e. new decision support tools, datasets, etc.) be met, but their investment (both in regards to time and finances) throughout a project will also lead to greater stakeholder buy-in when the project is completed.

With all the new technical developments in the fields of hydrology, earth observations, and software and web development, complicated science continues to become more accessible to a wider group of people. I believe that when organizations and the scientists working for them approach new projects with stakeholder adoption as the end goal, the funding and time put in to

projects will not go to waste. Conversely, if organizations and scientists insist on “knowing best” and do not allow stakeholders to have a true stake in a project, the project will almost always miss the mark.

I believe that the SWAT Data Viewer and nasaaccess applications have the potential to meet the goal of full stakeholder adoption. The apps have significantly simplified the process of accessing the climate inputs and model outputs for SWAT. They can be implemented anywhere, by any interested organization or individual, and can be used for any SWAT model for any watershed. This potential for stakeholder adoption can be realized through an increased effort to inform stakeholders of the current developments, train them in using the apps, and then to involve them in future developments.

REFERENCES

- Abbaspour, K.C. 2013. "SWAT-CUP 2012: SWAT Calibration and Uncertainty Programs—a User Manual." *Eawag: Dübendorf, Switzerland* 103.
- Abbott, M. B. 1991. *Hydroinformatics: Information Technology and the Aquatic Environment*. Avebury Technical.
- Ames, D. P., J. S. Horsburgh, Y. Cao, J. Kadlec, T. Whiteaker, and D. Valentine. 2012. *HydroDesktop: Web Services-Based Software for Hydrologic Data Discovery, Download, Visualization, and Analysis*. Vol. 37. doi://doi.org/10.1016/j.envsoft.2012.03.013. <http://www.sciencedirect.com/science/article/pii/S1364815212001053>.
- Ames, D. P., C. Michaelis, A. Anselmo, L. Chen, and H. Dunsford. 2008. "MapWindow GIS." In *Encyclopedia of GIS*, 633-634: Springer.
- Arias, M. E., T. A. Cochrane, K. S. Lawrence, T. J. Killeen, and T. A. Farrell. 2011. "Paying the Forest for Electricity: A Modelling Framework to Market Forest Conservation as Payment for Ecosystem Services Benefiting Hydropower Generation." *Environmental Conservation* 38 (4): 473-484. doi:10.1017/S0376892911000464. <https://www.cambridge.org/core/article/paying-the-forest-for-electricity-a-modelling-framework-to-market-forest-conservation-as-payment-for-ecosystem-services-benefiting-hydropower-generation/EE4EF042F04B5E1FD7ACF0801B815127>.
- U. S. Army. 1991. "GRASS Version 4.0: User's Reference Manual." *US Army Corps of Engineers, Construction Engineering Research Laboratory, Champaign, Illinois*.
- Arnold, J. G., J. R. Kiniry, R. Srinivasan, J. R. Williams, E. B. Haney, and S. L. Neitsch. 2012. "Soil and Water Assessment Tool Input/Output File Documentation: Version 2012." *Texas Water Resources Institute Technical Report*. <https://swat.tamu.edu/documentation/2012-io/>.

- Assunção, M. D., R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya. 2015. *Big Data Computing and Clouds: Trends and Future Directions*. Vol. 79-80. doi://doi-org.erl.lib.byu.edu/10.1016/j.jpdc.2014.08.003.
<http://www.sciencedirect.com.erl.lib.byu.edu/science/article/pii/S0743731514001452>.
- Babbar-Sebens, M., S. Mukhopadhyay, V. Bhushan Singh, and A. D. Piemonti. 2015. "A Web-Based Software Tool for Participatory Optimization of Conservation Practices in Watersheds." *Environmental Modelling & Software* 69: 111-127.
- Baird & Associates. "VizSWAT for SWAT2000 and SWAT2005.",
<http://www.bairdsoftware.com/vizswat.html>.
- Bouraoui, F., S. Benabdallah, A. Jrad, and G. Bidoglio. 2005. "Application of the SWAT Model on the Medjerda River Basin (Tunisia)." *Physics and Chemistry of the Earth, Parts A/B/C* 30 (8-10): 497-507.
- Boyd, D. and K. Crawford. 2012. "Critical Questions for Big Data: Provocations for a Cultural, Technological, and Scholarly Phenomenon." *Information, Communication & Society* 15 (5): 662-679.
- Brown, L. C. and T. O. Barnwell. 1987. *The Enhanced Stream Water Quality Models QUAL2E and QUAL2E-UNCAS: Documentation and User Manual* US Environmental Protection Agency. Office of Research and Development. Environmental Research Laboratory Athens, Georgia.
- Chen, Y. and D. Han. 2016. "Big Data and Hydroinformatics." *Journal of Hydroinformatics* 18 (4): 599-614.
- Croney, J. K., N. Kothari, M. Harder, G. S. Lindhorst, and A. Sanabria. 2007. "Framework for Creating Modular Web Applications." .
- Dahlhaus, P., A. Murphy, A. MacLeod, H. Thompson, K. McKenna, and A. Ollerenshaw. 2016. "Making the Invisible Visible: The Impact of Federating Groundwater Data in Victoria, Australia." *Journal of Hydroinformatics* 18 (2): 238-255.
- David, C. H. "RAPID Hub." rapid-hub.org., last modified n.d., accessed September 1, 2018,
<http://rapid-hub.org/>.

- David, C.H., J. S. Famiglietti, Z. L. Yang, F. Habets, and D. R. Maidment. 2016. "A Decade of RAPID—Reflections on the Development of an Open Source Geoscience Code." *Earth and Space Science* 3 (5): 226-244.
- De Mauro, A., M. Greco, and M. Grimaldi. 2015. "What is Big Data? A Consensual Definition and a Review of Key Research Topics." AIP, .
- Dile, Yihun T., Prasad Daggupati, Chris George, Raghavan Srinivasan, and Jeff Arnold. 2016. "Introducing a New Open Source GIS User Interface for the SWAT Model." *Environmental Modelling & Software* 85: 129-138.
- Dile, Yihun, R. Srinivasan, and Chris George. 2018. "QGIS Interface for SWAT (QSWAT)." 1.5. <https://swat.tamu.edu/software/qswat/>.
- ECMWF. "Climate Reanalysis.", last modified n.d., accessed September 3, 2018, <https://www.ecmwf.int/en/research/climate-reanalysis>.
- Eilander, D., P. Trambauer, J. Wagemaker, and A. van Loenen. 2016. *Harvesting Social Media for Generation of Near Real-Time Flood Maps*. Vol. 154. doi://doi.org/10.1016/j.proeng.2016.07.441. <http://www.sciencedirect.com/science/article/pii/S1877705816318306>.
- Engel, B. A., J. Y. Choi, J. Harbor, and S. Pandey. 2003. "Web-Based DSS for Hydrologic Impact Evaluation of Small Watershed Land use Changes." *Computers and Electronics in Agriculture* 39 (3): 241-249.
- Faghmous, J. H. and V. Kumar. 2014. "Spatio-Temporal Data Mining for Climate Data: Advances, Challenges, and Opportunities." In *Data Mining and Knowledge Discovery for Big Data*, 83-116: Springer.
- Foley, J. A. 2011. "Can we Feed the World & Sustain the Planet?" *Scientific American* 305 (5): 60-65.
- Francesconi, W., R. Srinivasan, E. Pérez-Miñana, S. P. Willcock, and Marcela Quintero. 2016. "Using the Soil and Water Assessment Tool (SWAT) to Model Ecosystem Services: A Systematic Review." *Journal of Hydrology* 535: 625-636.

- George, C. and L. F. Leon. 2008. "WaterBase: SWAT in an Open Source GIS." *The Open Hydrology Journal* 2 (1).
- Gill, M. A.. 1978. "Flood Routing by the Muskingum Method." *Journal of Hydrology* 36 (3-4): 353-363.
- Giuliani, G., K. Rahman, N. Ray, and A. Lehmann. 2013. "OWS4SWAT: Publishing and Sharing SWAT Outputs with OGC Standards." *International Journal of Advanced Computer Science and Applications* 3 (3): 90-98.
- Goodall, J. L., J. S. Horsburgh, T. L. Whiteaker, D. R. Maidment, and Ilya Zaslavsky. 2008. "A First Approach to Web Services for the National Water Information System." *Environmental Modelling & Software* 23 (4): 404-411.
- Green, W. H. and G. A. Ampt. 1911. "Studies on Soil Physics." *The Journal of Agricultural Science* 4 (1): 1-24.
- Grimson, R., N. Montroull, R. Saurral, P. Vasquez, and I. Camilloni. 2013. "Hydrological Modelling of the Iberá Wetlands in Southeastern South America." *Journal of Hydrology* 503: 47-54.
- Horsburgh, J. S., D. G. Tarboton, K. AT Schreuders, D. R. Maidment, I. Zaslavsky, and D. Valentine. 2010. "HydroServer: A Platform for Publishing Space-Time Hydrologic Datasets." .
- Jacobs, J. W. 2002. "The Mekong River Commission: Transboundary Water Resources Planning and Regional Security." *The Geographical Journal* 168 (4): 354-364.
- Janssen, J. AEB, A. Y. Hoekstra, J. de Kok, and R. MJ Schielen. 2009. "Delineating the Model-Stakeholder Gap: Framing Perceptions to Analyse the Information Requirement in River Management." *Water Resources Management* 23 (7): 1423.
- Jayakrishnan, RSRS, R. Srinivasan, C. Santhi, and J. G. Arnold. 2005. "Advances in the Application of the SWAT Model for Water Resources Management." *Hydrological Processes: An International Journal* 19 (3): 749-762.
- Jeong, H. G., S. J. Kim, and R. Ha. 2013. "Assessment of Climate Change Impact on Storage Behavior of Chungju and the Regulation Dams using SWAT Model." *Journal of Korea Water Resources Association* 46 (12): 1235-1247.

- Klug, H. and A. Kmoach. 2015. "Operationalizing Environmental Indicators for Real Time Multi-Purpose Decision Making and Action Support." *Ecological Modelling* 295: 66-74.
- . 2014. "A SMART Groundwater Portal: An OGC Web Services Orchestration Framework for Hydrology to Improve Data Access and Visualisation in New Zealand." *Computers & Geosciences* 69: 78-86.
- Laney, D.. 2001. "3D Data Management: Controlling Data Volume, Velocity and Variety." *META Group Research Note* 6 (70): 1.
- Mohammed, I. N., J. D. Bolten, R. Srinivasan, and V. Lakshmi. 2018. "Improved Hydrological Decision Support System for the Lower Mekong River Basin using Satellite-Based Earth Observations." *Remote Sensing* 10 (6).
- MRC. 2011. *Annual Mekong Flood Report*. Phnom Penh, Cambodia: Mekong River Commission.
- . 2018. *Annual Report 2017*. Vientiane, Lao PDR: MRC.
- . "Fisheries.", last modified n.d., accessed May 20, 2018, <http://www.mrcmekong.org/topics/fisheries/>.
- . "Flood & Drought." www.mrcmekong.org., last modified n.d., accessed August 1, 2018, <http://www.mrcmekong.org/topics/flood-and-drought/>.
- . "Hydrology.", last modified n.d., accessed May 30, 2018, <http://www.mrcmekong.org/mekong-basin/hydrology/>.
- . "Mekong Basin." [mrcmekong.org](http://www.mrcmekong.org)., last modified n.d., accessed May 20, 2018, <http://www.mrcmekong.org/mekong-basin/>.
- Muleta, M. K. and J. W. Nicklow. 2005. "Decision Support for Watershed Management using Evolutionary Algorithms." *Journal of Water Resources Planning and Management* 131 (1): 35-44.
- Narsimlu, B., A. K. Gosain, and B. R. Chahar. 2013. "Assessment of Future Climate Change Impacts on Water Resources of Upper Sind River Basin, India using SWAT Model." *Water Resources Management* 27 (10): 3647-3662.

NASA. "Precipitation Measurement Missions." nasa.gov., last modified April 1, accessed September 1, 2018, <https://pmm.nasa.gov/>.

NASA Goddard Space Flight Center. "Land Data Assimilation Systems." gsfc.nasa.gov., last modified June 29, accessed August 31, 2018, <https://ldas.gsfc.nasa.gov/gldas/GLDASgoals.php>.

———. "Moderate Resolution Imaging Spectroradiometer." modis.gsfc.nasa.gov., last modified n.d., accessed August 28, 2018, <https://modis.gsfc.nasa.gov/about/>.

Neitsch, S. L., Jeffrey G. Arnold, Jim R. Kiniry, and Jimmy R. Williams. 2011. "Soil and Water Assessment Tool Theoretical Documentation Version 2009." . <https://swat.tamu.edu/media/99192/swat2009-theory.pdf>.

Nelson, J., D. P. Ames, N. Jones, D. G. Tarboton, Z. Li, X. Qiao, and S. Crawley. 2016. "An Extensible, Modular Architecture Coupling HydroShare and Tethys Platform to Deploy Water Science Web Apps."

Neteler, M., M. H. Bowman, M. Landa, and M. Metz. 2012. "GRASS GIS: A Multi-Purpose Open Source GIS." *Environmental Modelling & Software* 31: 124-130.

Oddo, P. C., A. Ahamed, and J. D. Bolten. 2018. "Socioeconomic Impact Evaluation for Near Real-Time Flood Detection in the Lower Mekong River Basin." *Hydrology* 5 (2): 23.

Olivera, F., M. Valenzuela, R. Srinivasan, J. Choi, H. Cho, S. Koka, and A. Agrawal. 2006. "ArcGIS-SWAT: A Geodata Model and GIS Interface for SWAT." *JAWRA Journal of the American Water Resources Association* 42 (2): 295-309.

Olsson, J. A. and L. Andersson. 2006. "Possibilities and Problems with the use of Models as a Communication Tool in Water Resource Management." In *Integrated Assessment of Water Resources and Global Change*, 97-110: Springer.

Open Geospatial Consortium. 2004. "The Havoc of Non-Interoperability." *OGC White Paper*, 7p.

Overpeck, J. T., Gerald A. Meehl, Sandrine Bony, and David R. Easterling. 2011. "Climate Data Challenges in the 21st Century." *Science* 331 (6018): 700-702.

- Piman, T., T. Lennaerts, and P. Southalack. 2013. "Assessment of Hydrological Changes in the Lower Mekong Basin from Basin-Wide Development Scenarios." *Hydrological Processes* 27 (15): 2115-2125.
- Provost, F. and T. Fawcett. 2013. "Data Science and its Relationship to Big Data and Data-Driven Decision Making." *Big Data* 1 (1): 51-59.
- Quintero, M., S. Wunder, and R. D. Estrada. 2009. *For Services Rendered? Modeling Hydrology and Livelihoods in Andean Payments for Environmental Services Schemes*. Vol. 258. doi://doi.org/10.1016/j.foreco.2009.04.032. <http://www.sciencedirect.com/science/article/pii/S0378112709003132>.
- Rajib, M. A., V. Merwade, I. L. Kim, L. Zhao, C. Song, and S. Zhe. 2016. "SWATShare—A Web Platform for Collaborative Research and Education through Online Sharing, Simulation and Visualization of SWAT Models." *Environmental Modelling & Software* 75: 498-512.
- Rasanen, T. A., P. Someth, H. Lauri, J. Koponen, J. Sarkkula, and M. Kummu. 2017. "Observed River Discharge Changes due to Hydropower Operations in the Upper Mekong Basin." *Journal of Hydrology* 545: 28-41.
- Roehrig, J.. 2002. "Information Interoperability for River Basin Management." *Technology Resource Management & Development—Scientific Contributions for Sustainable Development* 2: 141-148.
- Selding, P.. 2012. "US Government-Leased Satellite Capacity Going Unused." *Space News*: 12-03.
- SERVIR Global. "SERVIR Connects Space to Village.", last modified n.d., accessed September 1, 2018, <https://servirglobal.net/>.
- SERVIR SCO. 2017. *SERVIR Service Planning Toolkit*
- Sneddon, C. and C. Fox. 2006. "Rethinking Transboundary Waters: A Critical Hydrogeopolitics of the Mekong Basin." *Political Geography* 25 (2): 181-202.
- Snow, A. D., S. D. Christensen, N. R. Swain, E. J. Nelson, D. P. Ames, N. L. Jones, D. Ding, N. S. Noman, C. H. David, and F. Pappenberger. 2016. "A High-Resolution National-Scale Hydrologic Forecast System from a Global Ensemble Land Surface Model." *JAWRA Journal of the American Water Resources Association* 52 (4): 950-964.

Soil Conservation Service. 1972. "National Engineering Handbook - Hydrology." 4.

Souffront Alcantara, M. A., C. Kesler, M. J. Stealey, E. J. Nelson, D. P. Ames, and N. L. Jones. 2017. "Cyberinfrastructure and Web Apps for Managing and Disseminating the National Water Model." *JAWRA Journal of the American Water Resources Association*.

Srinivasan, R. "Vizswat.", last modified n.d., accessed September 1, 2018, <https://swat.tamu.edu/software/vizswat/>.

Srinivasan, R., J. G. Arnold, and R. Jayakrishnan. 1996. "SWAT/GRASS Interface User's Manual." .

Srinivasan, R., J. G. Arnold, and C. A. Jones. 1998. "Hydrologic Modelling of the United States with the Soil and Water Assessment Tool." *International Journal of Water Resources Development* 14 (3): 315-325.

Swain, N., S. Christensen, J. Nelson, and N. Jones. 2015. "Tethys Platform: A Platform for Water Resources Modeling and Decision Support Web Apps." .

Tarboton, D. G., J. S. Horsburgh, D. R. Maidment, T. Whiteaker, I. Zaslavsky, M. Piasecki, J. Goodall, D. Valentine, and T. Whitenack. 2009. "Development of a Community Hydrologic Information System."

Tarboton, D. G., R. Idaszak, J. S. Horsburgh, J. Heard, D. Ames, J. L. Goodall, L. Band, V. Merwade, A. Couch, and J. Arrigo. 2014. "HydroShare: Advancing Collaboration through Hydrologic Data and Model Sharing." .

The Office of Water Prediction. "The National Water Model.", last modified n.d., accessed August 31, 2018, <http://water.noaa.gov/about/nwm>.

UNISDR, UNO. 2015. "Sendai Framework for Disaster Risk Reduction 2015–2030." UNISDR Sendai, Japan, .

United Nations. 2016. *Sustainable Development Goals Report 2016* UN.

USAID. 2011. *Southeast Asia --- Floods*: USAID.

USDA Soil Conservation Service. 1983. "National Engineering Handbook - Hydrology." 4.

USGS. "Hydrography.", last modified n.d., accessed August 31, 2018, <https://nhd.usgs.gov/>.

Verdin, A., B. Rajagopalan, W. Kleiber, and C. Funk. 2015. "A Bayesian Kriging Approach for Blending Satellite and Ground Precipitation Observations." *Water Resources Research* 51 (2): 908-921.

Vigerstol, K. L. and J. E. Aukema. 2011. *A Comparison of Tools for Modeling Freshwater Ecosystem Services*. Vol. 92. doi://doi.org/10.1016/j.jenvman.2011.06.040.
<http://www.sciencedirect.com/science/article/pii/S0301479711002374>.

Voinov, A. and R. Costanza. 1999. *Watershed Management and the Web*. Vol. 56. doi://doi.org/10.1006/jema.1999.0281.
<http://www.sciencedirect.com/science/article/pii/S0301479799902815>.

Welderufael, W. A., Y. E. Woyessa, and D. C. Edossa. 2013. *Impact of Rainwater Harvesting on Water Resources of the Modder River Basin, Central Region of South Africa*. Vol. 116. doi://doi.org/10.1016/j.agwat.2012.07.012.
<http://www.sciencedirect.com/science/article/pii/S0378377412002053>.

Williams, J. R. 1975. "Sediment Routing for Agricultural Watersheds." *JAWRA Journal of the American Water Resources Association* 11 (5): 965-974.

———. 1980. "SPNM, a Model for Predicting Sediment, Phosphorus, and Nitrogen Yields from Agricultural Basins 1." *JAWRA Journal of the American Water Resources Association* 16 (5): 843-848.

Williams, J. R. 1969. "Flood Routing with Variable Travel Time Or Variable Storage Coefficients." *Transactions of the ASAE* 12 (1): 100-0103.

Winchell, M., R. Srinivasan, M. Di Luzio, and J. Arnold. 2010. "ArcSWAT Interface for SWAT 2009." *User's Guide, Blackland Research Center, Texas Agricultural Experiment Station, Temple*.
<https://www.researchgate.net/file.PostFileLoader.html?id=543e7c21d2fd64cb378b4600&assetKey=AS%3A273668666527744%401442259103080>.

Yu, Z.. 2018a. *SWAT Output Viewer*. Vol. 0.1.2.

———. 2018b. *SWAT Output Viewer User Manual*. Vol. 1.

Zhang, Y., J. Xia, J. Chen, and M. Zhang. 2011. "Water Quantity and Quality Optimization Modeling of Dams Operation Based on SWAT in Wenyu River Catchment, China." *Environmental Monitoring and Assessment* 173 (1-4): 409-430.

Zhang, Y., J. Xia, Q. Shao, and X. Zhai. 2013. "Water Quantity and Quality Simulation by Improved SWAT in Highly Regulated Huai River Basin of China." *Stochastic Environmental Research and Risk Assessment* 27 (1): 11-27.

APPENDIX A. TECHNICAL DOCUMENTATION: NASAACCESS APP

nasaaccess is a software tool for accessing and processing earth observation data products on a global scale developed as part of the work done by Mohammed et al. (2018). The main objective of the nasaaccess software package is to streamline retrieving and processing NASA earth observation data products such as the Global Precipitation Measurement (GPM) mission and the Global Land Data Assimilation System (GLDAS). Moreover, the nasaaccess software package includes a smoothing technique method to address the spatial scale issues prevalent in these earth observation data products. The core functionality of nasaaccess can be summarized as:

- Access the NASA Goddard Space Flight Center (GSFC) servers to download earth observation data
- Clip needed grids to an input shapefile of a user study watershed
- Handle temporal and spatial inconsistencies
- Generate daily climate gridded data files and definition files compatible with SWAT and other models

The inputs needed for the various functions within nasaaccess are: start and end dates for the user's desired earth observation data, a shapefile for the study area of interest, and a digital elevation model (DEM) for the area of interest.

nasaaccess was originally built as an R library containing four separate data processing functions which are described in Table A-1. The number of potential users has been limited in part because of the relatively high learning curve involved in using R. To make the nasaaccess functionality available to a wider audience and to eliminate the need to download, install, and run it from a local machine, a nasaaccess web application was built using the Tethys platform.

Table A-1: nasaaccess functions

Function	Definition
GLDASwat	Generates SWAT compatible air temperature files
GPMswat	Generates SWAT compatible precipitation files
GLDAS Poly Centroid	Generates an air temperature station file at the centroid of each polygon within the input watershed boundary
GPM Poly Centroid	Generates a precipitation station file at the centroid of each polygon within the input watershed boundary

A.1 Data requirements and file structure

The nasaaccess web application is equipped with the full functionality of the nasaaccess R package. For the nasaaccess functions to work properly within the app, there needed to be a file structure established to store the input DEMs and shapefiles. Figure A-1 shows the file structure

that the app recognizes. The config.py file within the app package is where the file path to the “nasaaccess_data” folder can be specified.

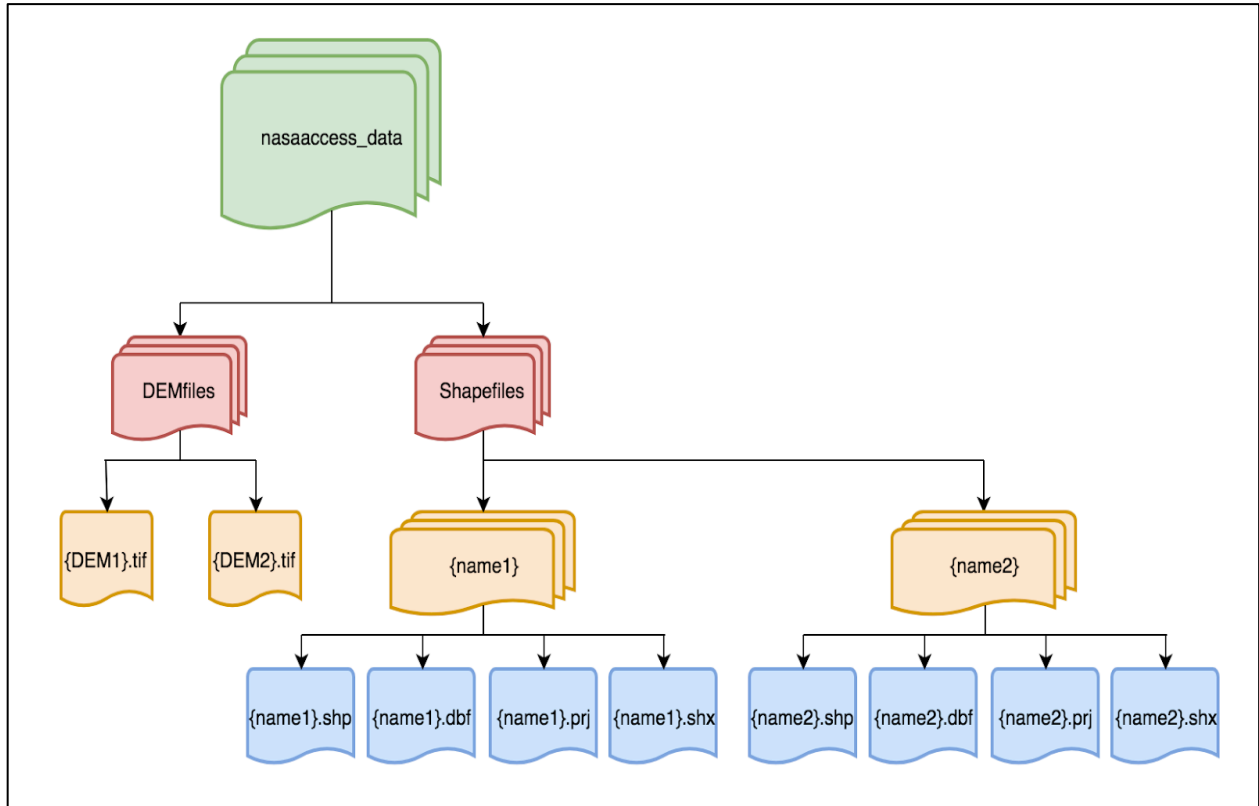


Figure A-1: nasaaccess_data file structure

A.2 App architecture

Linking the web interface of the app with the nasaaccess functions was done using the app package shown in Figure A-2 and the Model-View-Controller (MVC) architecture described in Chapter 2 and shown in Figure A-3. This app package contains html, JavaScript, css files for front end visualization and a number of python scripts to facilitate file uploading and

downloading, and, most importantly, passing user requests to the nasaaccess functions. Table A-2 describes the role of each of the files in the application file structure.

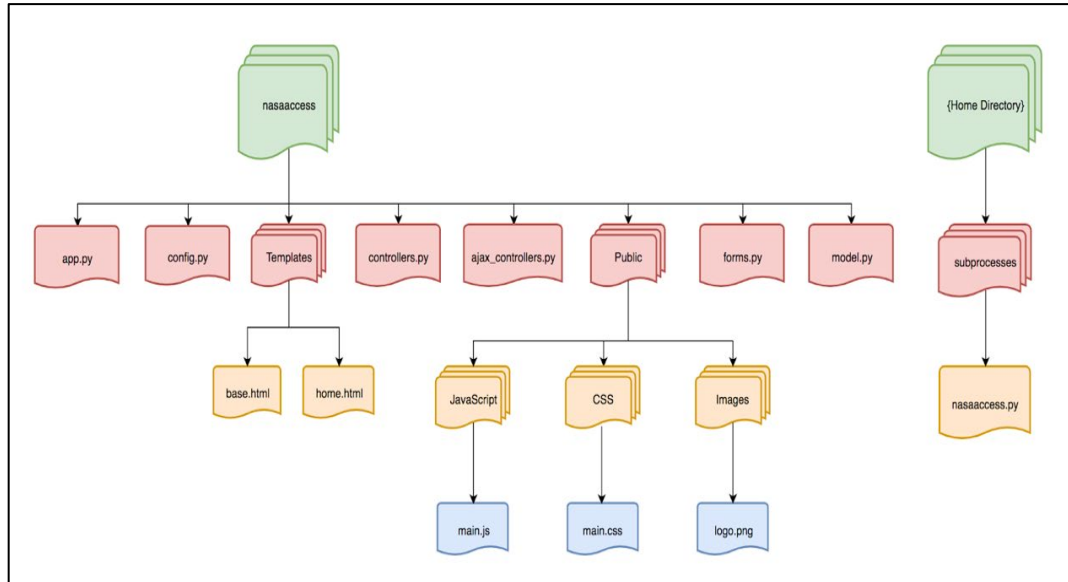


Figure A-2: nasaaccess application architecture

Table A-2: Description of each file in the app package

File	Role
app.py	Basic application information (app name, color, description) Maps URLs to the various controller functions
config.py	Specifies file paths to data stored on the server that the app needs to function
controllers.py	Sets up the initial view of the app including shapefile and DEM selection dropdowns (based on files available in the “nasaaccess_data” folder) and the date pickers
ajax_controllers.py	Manages all user requests passed from front-end to back-end through ajax
forms.py	Initiates Django forms to handle file upload to the server

Table A-2 Continued

File	Role
model.py	Contains all independent python functions for processing data
nasaaccess.py	Python script containing all nasaaccess functions. Note: This file is stored separate from the app package so that it can run independent from the app (This will be described in the following sections)
base.html and home.html	Templates handling all the individual elements in the web interface
main.js	Initiates the map visualization Handles passing user requests to the back-end for processing
main.css	Styling of HTML elements
logo.png	Image file (.jpg, .png, .gif) of the app logo

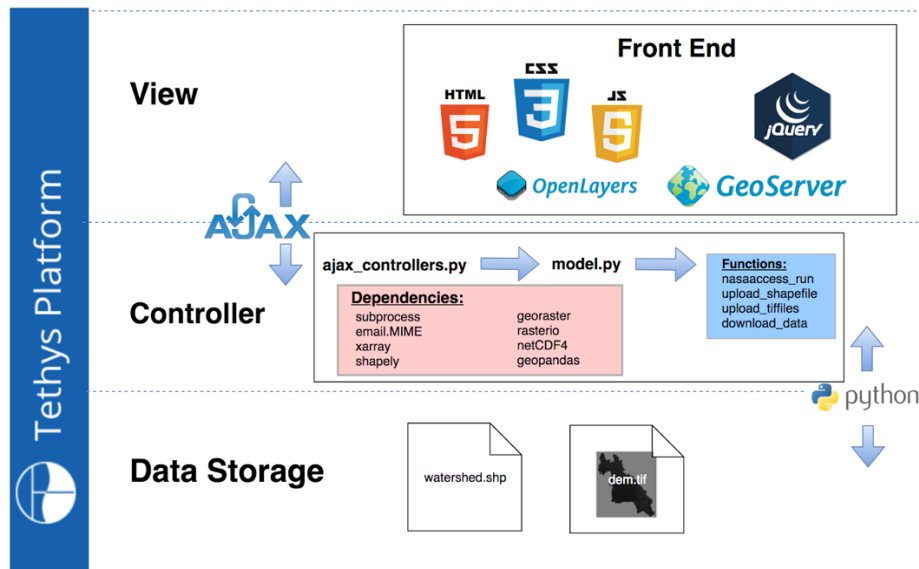


Figure A-3: MVC framework for nasaaccess application

A.3 Map publishing

The nasaaccess application does not feature extensive geospatial data visualization. However, the main view of the app does contain a small map to allow the user to view the extent of the selected watershed boundary. Displaying the watershed boundaries in the app is facilitated through a geoserver. Once uploaded to a geoserver, the watershed basins are available as WMS services, accessible through a URL request similar to the following:

http://216.218.240.206:8080/geoserver/wms/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&FORMAT=image/png&TRANSPARENT=true&LAYERS=nasaaccess:lower_mekong&BBOX=6.204590651114307,91.01387164397141,28.79540934888569,117.98612835602859&WIDTH=1736&HEIGHT=1454

The app can access any geoserver with a public facing IP address by simply changing the `geoserver_url` variable in the `main.js` file and the `geoserver` object in `config.py`.

A.4 App functionality

Because nasaaccess was built for the sole purpose of preparing input data for model applications like SWAT, there is no real need for much data visualization within the nasaaccess web application. Thus, the nasaaccess web application features an aesthetic, uncomplicated interface for submitting data processing jobs to the server, and then provides feedback for the user to know that it is working as expected.

The main view of the app (Figure A-4) features two panes. The left pane contains all of the user options, each of which relates directly to one of the input arguments of the nasaaccess functions. The right pane displays a map and the watershed boundary for which the user has selected to access data.

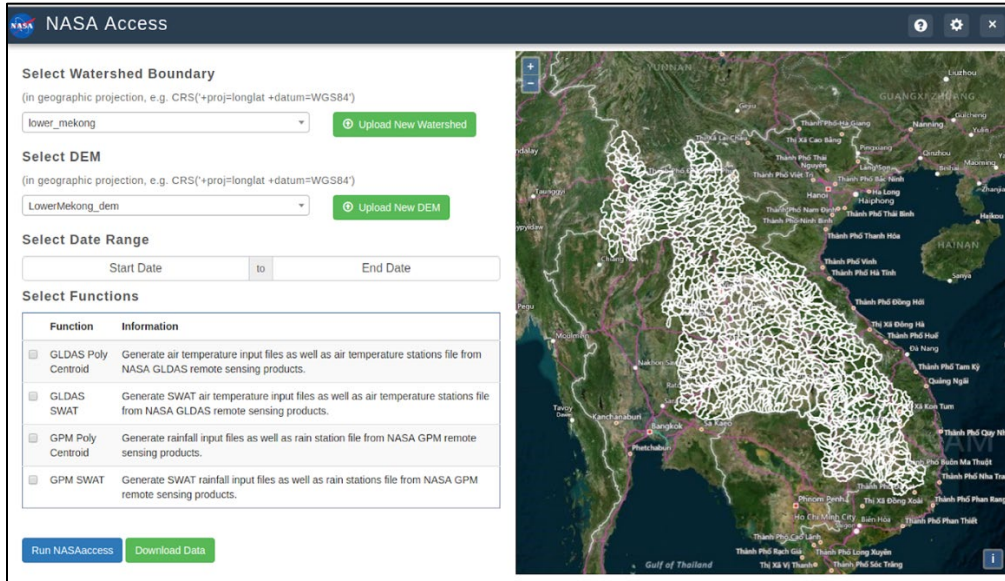


Figure A-4: Main view of nasaaccess application

Users of the nasaaccess application are not limited to running nasaaccess on the currently available watersheds of the app. It also allows users to upload their own shapefiles and DEM files temporarily to the server so that the nasaaccess functions are run for the user’s area of interest.

For users wanting to upload their own files, the user interface features “Upload New Watershed” and “Upload New DEM” buttons which are used to open a new modal view in the app for the user to select files from their local machine and upload them to the app’s hosting server. These modal views were built using Django forms and models to facilitate file upload without the need for excessive python and JavaScript coding.

Model and Form: Shapefiles and DEMfiles

The forms.py file in the app package contains the code that initiates the file upload forms. Using the Django framework, these forms (technically considered ModelForms) are rendered as html elements in home.html through controllers.py. The models in the model.py file dictate the

input type of the fields within the forms and how the forms perform (i.e. file path to folder on server to upload files to). Figure A-5 is a graphical depiction of how each of the files mentioned above combine to handle file uploading using Django forms and models.

When the user has selected a file to upload and then clicks the “Upload New {DEM .tif File or Shapefile}” button, the URLs for the file uploading python functions in `ajax_controllers.py` are called using the “action” attribute in the form html element. The files and requests are then passed to the back end using the POST method.

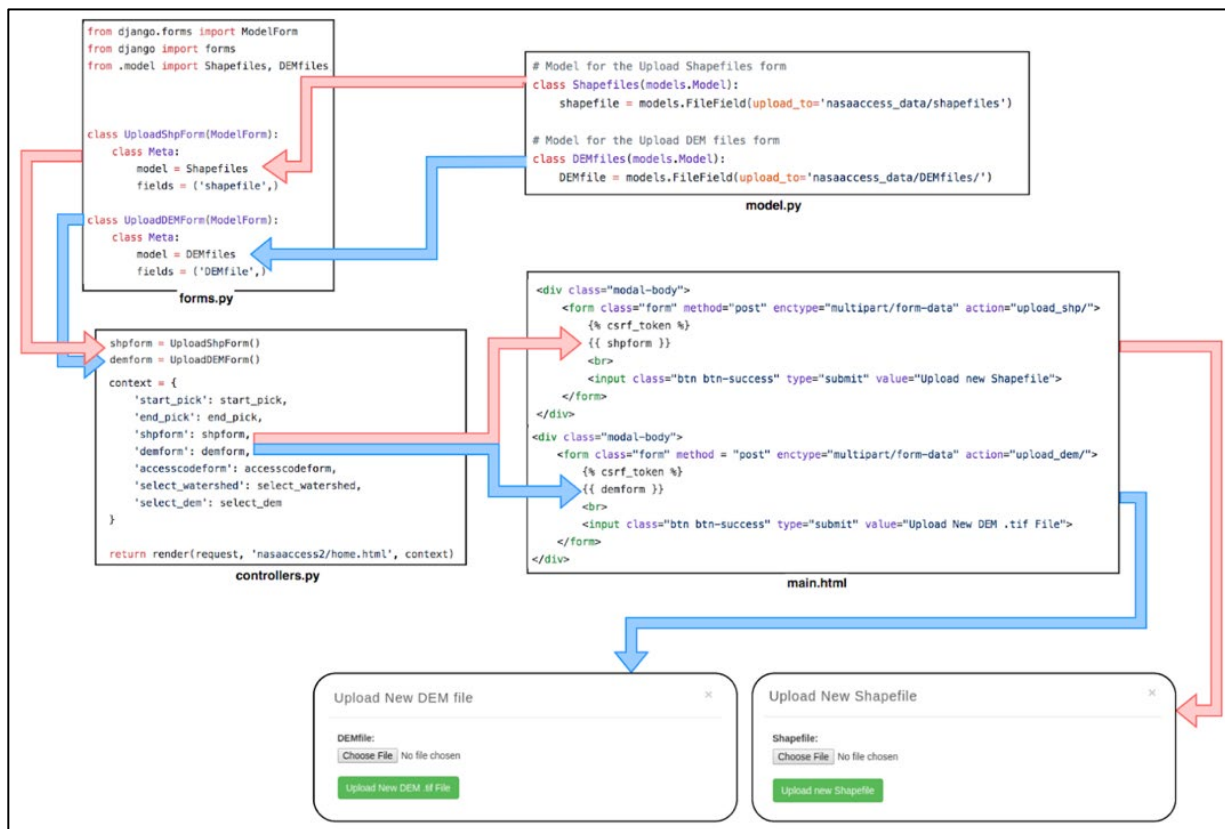


Figure A-5: Model and form structure for uploading new files to the app

Function: upload_tifffiles() and upload_shapefiles()

The upload functions, `upload_tifffiles()` and `upload_shapefiles()`, in `ajax_controllers.py` function in essentially the same way. They first read in the user's POST request, including the user's files, and then save the file. The `save()` function simply saves the users file to the folder on the app's hosting server specified in the "upload_to" argument of the model in `model.py`. After the file has been saved, the functions re-render the app's home page.

Function: upload_shapefile()

When uploading a shapefile, the `upload_shapefile()` function is called in addition to saving the shapefile to the server. This function takes the shapefile and uploads it to the geoserver (`geoserver_url` specified in `config.py`) using the "put" method in the `requests` python module (Figure A-6). Uploading the shapefile to a geoserver allows the user to then select the watershed that they uploaded and view the boundary extent in the map view on the right pane of the home page.

```
request_url = '{0}workspaces/{1}/datastores/{2}/file.shp'.format(geoserver['rest_url'],
                                                             geoserver['workspace'], storename)
requests.put(request_url, verify=False, headers=headers, data=data, auth=(user, password))
```

Figure A-6: Code to upload shapefile to geoserver

The main purpose of the `nasaaccess` web application is to give users access to the data processing functions. Thus, the key functionality of the app resides in the four input elements in the left pane of the app. The app leverages these input elements in conjunction with AJAX and python to pass user requests as input arguments into the `nasaaccess` functions.

In the left pane of the application’s main interface (Figure A-7), the user may select the watershed boundary, DEM, and date range to be used as input arguments in the nasaaccess functions. They are also given the option to select one or multiple nasaaccess functions to run using those arguments. After selecting inputs and clicking the “Run nasaaccess” button, a modal appears requesting the email address of the user (Figure A-8). Because the nasaaccess functions generally take an extended length of time, the app passes the user’s inputs to the server, initiates the nasaaccess functions, and then detaches those functions from the application so that the app isn’t tied up waiting for a single process to run. By obtaining the user’s email address, the app can notify the user once the processing of their data has been completed.

The screenshot shows the nasaaccess application interface. At the top, there is a header with the NASA logo and the text "nasaaccess". Below the header, there are four main sections for user input:

- Select Watershed Boundary:** Includes a dropdown menu for "Select Boundary Shapefile" and a green button labeled "Upload New Watershed".
- Select DEM:** Includes a dropdown menu for "Select DEM" and a green button labeled "Upload New DEM".
- Select Date Range:** Includes two input fields for "Start Date" and "End Date" separated by a "to" label.
- Select Functions:** Includes a table with four rows, each representing a function with a checkbox and a description.

At the bottom of the interface, there are two buttons: "Run NASAaccess" (blue) and "Download Data" (green).

Function	Information
<input type="checkbox"/> GLDAS Poly Centroid	Generate air temperature input files as well as air temperature stations file from NASA GLDAS remote sensing products.
<input type="checkbox"/> GLDAS SWAT	Generate SWAT air temperature input files as well as air temperature stations file from NASA GLDAS remote sensing products.
<input type="checkbox"/> GPM Poly Centroid	Generate rainfall input files as well as rain station file from NASA GPM remote sensing products.
<input type="checkbox"/> GPM SWAT	Generate SWAT rainfall input files as well as rain stations file from NASA GPM remote sensing products.

Figure A-7: User input options for accessing nasaaccess functions

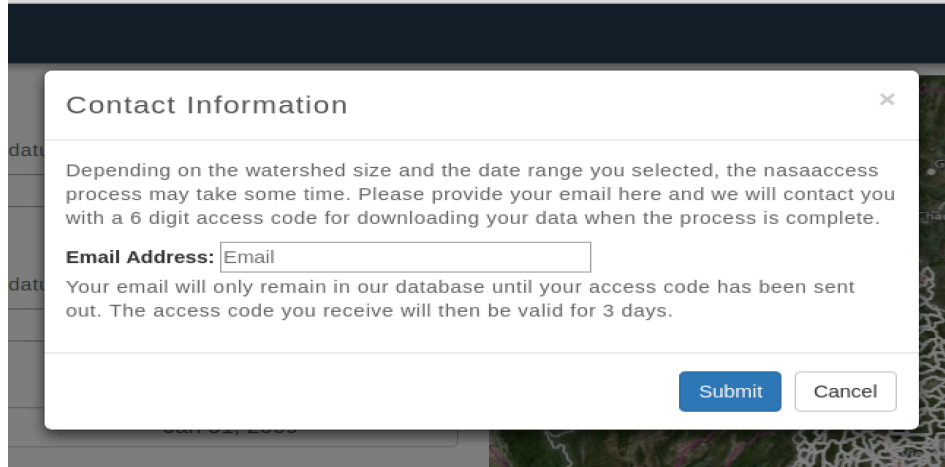


Figure A-8: Contact information modal for notification when functions have completed

By clicking “Submit” in the “Contact Information” modal, the app uses AJAX to pass all of the user’s selected inputs (watershed boundary, DEM, date range, email, and list of nasaaccess functions to run) to the `nasaaccess_run()` function in `ajax_controllers.py` on the server side.

Function: `nasaaccess_run()`

`nasaaccess_run()` (Figure A-9) takes the user submitted information and formats so that it matches the input arguments of the `nasaaccess` functions. The `nasaaccess` functions use file paths as the inputs for the shapefile and DEM arguments. Thus, the names of the watershed boundary and DEM selected by the user are used to find the file path to the shapefile and DEM by the same names. There also needs to be a new temporary folder created to store the outputs of the functions. Each time a user submits a request, a new unique ID is created and a temporary folder is named by the unique ID. Lastly, the `nasaaccess` functions create multiple intermediate files throughout the process. Another temporary folder is created on the server to store this intermediate data.

Once the user inputs from the front end has been formatted and the new folders are created with read and write permissions, `nasaaccess_run()` uses the `subprocess` and `sys` python modules to call the `nasaaccess.py` script to run on the server completely independent of the app. The `subprocess` module also allows all of the file paths and other inputs to be sent to `nasaaccess.py` as arguments. Once the subprocess is called, the front page of the app returns to its original state and is ready to process another data request or to simply be closed. Closing the app will have no effect on the `nasaaccess` functions running on the server.

```
def nasaaccess_run(email, functions, watershed, dem, start, end):
    #identify where each of the input files are located in the server
    shp_path = os.path.join(data_path, 'shapefiles', watershed, watershed + '.shp')
    dem_path = os.path.join(data_path, 'DEMfiles', dem + '.tif')
    #create a new folder to store the user's requested data
    unique_id = ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in range(6))
    unique_path = os.path.join(data_path, 'outputs', unique_id, 'nasaaccess_data')
    os.makedirs(unique_path, 0777)
    #create a temporary directory to store all intermediate data while nasaaccess functions run
    tempdir = os.path.join(temp_path, unique_id)
    os.makedirs(tempdir, 0777)

    functions = ','.join(functions)

    #pass user's inputs and file paths to the nasaaccess python function that will run detached from the app
    run = subprocess.Popen([sys.executable, "/home/ubuntu/subprocesses/nasaaccess.py", email, functions, unique_id,
                           shp_path, dem_path, unique_path, tempdir, start, end])
```

Figure A-9: nasaaccess_run() code

Python file: nasaaccess.py

The `nasaaccess.py` script contains multiple independent functions, including the four `nasaaccess` functions. When the script is called from `nasaaccess_run()`, the first thing that the script does is read in all of the file paths and inputs from the subprocess call. This is done using the `sys.argv` method, which simply identifies the arguments in the subprocess call based on the order they were listed (Figure A-10).

```

#read in file paths and arguments from subprocess call in model.py
email = sys.argv[1]
functions = sys.argv[2].split(',')
unique_id = sys.argv[3]
shp_path = sys.argv[4]
dem_path = sys.argv[5]
unique_path = sys.argv[6]
tempdir = sys.argv[7]
start = sys.argv[8]
end = sys.argv[9]

```

Figure A-10: Input arguments passed from front end to nasaaccess.py

Looping through the list of selected functions (i.e. a combination of GPMpolyCentroid, GPMswat, GLDASpoly Centroid, and GLDASwat), the app is able to pass the input arguments into only the functions that the user requested (Figure A-11). For each function that the user selected, a subfolder within the user’s unique output folder with read and write permissions (‘0777’ gives read and write permissions to all users) is created with the name of the function. All outputs will be saved within that new folder. Figure A-12 shows an example file structure that would be created when all four nasaaccess functions are called.

```

#Run nasaaccess functions requested by user
for func in functions:
    print(func)
    if func == 'GPMpolyCentroid':
        output_path = unique_path + '/GPMpolyCentroid/'
        os.makedirs(output_path, 0777)
        print('running GPMpoly')
        GPMpolyCentroid(output_path, shp_path, dem_path, start, end)
    elif func == 'GPMswat':
        output_path = unique_path + '/GPMswat/'
        os.makedirs(output_path, 0777)
        print('running GPMswat')
        GPMswat(output_path, shp_path, dem_path, start, end)
    elif func == 'GLDASpolyCentroid':
        output_path = unique_path + '/GLDASpolyCentroid/'
        os.makedirs(output_path, 0777)
        print('running GLDASpoly')
        GLDASpolyCentroid(tempdir, output_path, shp_path, dem_path, start, end)
    elif func == 'GLDASwat':
        output_path = unique_path + '/GLDASwat/'
        os.makedirs(output_path, 0777)
        print('running GLDASwat')
        GLDASwat(output_path, shp_path, dem_path, start, end)

```

Figure A-11: nasaaccess functions being called based on user inputs

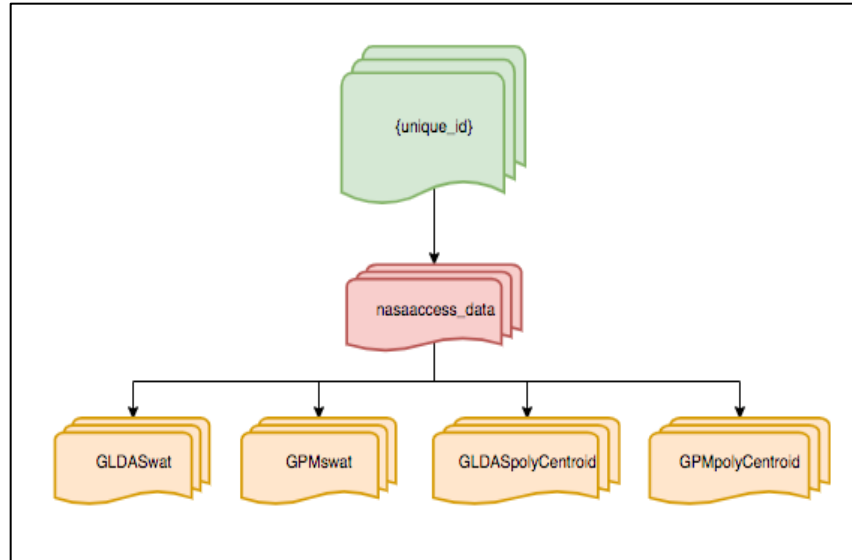


Figure A-12: File structure for nasaaccess function outputs

Function: send_email()

Once all of the functions are completed, the `send_email()` function (Figure A-13) is called. `send_email()` is one of the functions defined within `nasaaccess.py`. It uses the `email.mime` class of the `email` python package to send an email to the user notifying them that their data is ready for download and providing them with the unique ID (the name of the folder containing their requested data) generated when they submit their email and request through the app. Figure A-14 is an example of the email message sent automatically by the `send_email()` function.

Once a user has received an email with an access code, they can return to the app and click the “Download Data” button. This opens the “Download Data” modal (Figure A-15) that contains a text input field for the user to input the access code from the email.

```

def send_email(to_email, unique_id):

    from_email = 'nasaaccess@gmail.com'

    msg = MIMEMultipart('alternative')
    msg['Subject'] = 'Your nasaaccess data is ready'

    msg['From'] = from_email
    msg['To'] = to_email

    #email content
    message = """"\
        <html>
            <head></head>
            <body>
                <p>Hello,
                <br>
                Your nasaaccess data is ready for download at
                <a href="http://tethys-servir-mekong.adpc.net/apps/nasaaccess">
                    http://tethys-servir-mekong.adpc.net/apps/nasaaccess
                </a>
                <br>
                Your unique access code is: <strong>"" + unique_id + ""</strong><br>
            </p>
            </body>
        </html>
    """"

    part1 = MIMEText(message, 'html')
    msg.attach(part1)

    gmail_user = 'nasaaccess@gmail.com'
    gmail_pwd = 'nasaaccess123'
    smtpserver = smtplib.SMTP('smtp.gmail.com', 587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo()
    smtpserver.login(gmail_user, gmail_pwd)
    smtpserver.sendmail(gmail_user, to_email, msg.as_string())
    smtpserver.close()

```

Figure A-13: send_email() code

Hello,
 Your nasaaccess data is ready for download at <http://tethys-servir-mekong.adpc.net/apps/nasaaccess>
 Your unique access code is: **00SGL2**

Figure A-14: Example email sent by the nasaaccess app

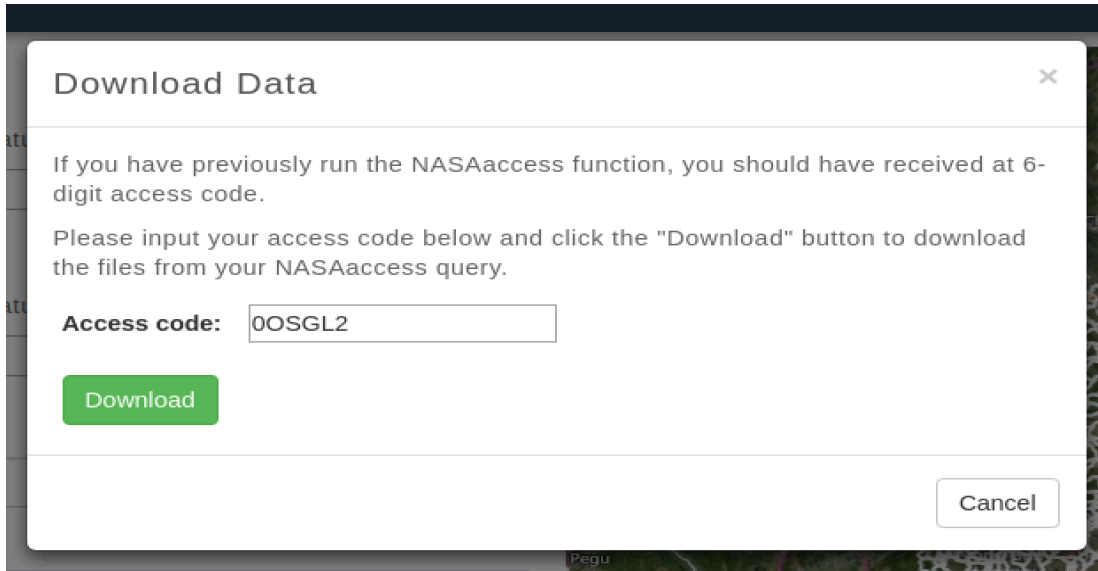


Figure A-15: Download data modal

Model and Form: accessCode

The access code input is handled using a simple Django form and model similar to the file upload forms discussed previously. When the access code is copied into the text box and the “Download” button is clicked, the form is submitted to the `download_data()` function in `ajax_controllers.py` on the back end using the POST method.

Function: download_data()

The `download_data()` function (Figure A-16) takes the access code submitted by the user and uses it to find the data in the folder with the same name. From there, the function compresses the folder (and all of its subfolders) into a zip file. Then, using the Django `HttpResponse` class, the zipped folder is sent to the front as an attachment object for downloading which automatically opens the browser’s download window for the user to specify a destination for the data on their local machine.

```

if request.method == 'POST':
    #get access code from form
    access_code = request.POST['access_code']

    #identify user's file path on the server
    unique_path = os.path.join(data_path, 'outputs', access_code, 'nasaaccess_data')

    #compress the entire directory into a .zip file
    def zipfolder(foldername, target_dir):
        zipobj = zipfile.ZipFile(foldername + '.zip', 'w', zipfile.ZIP_DEFLATED)
        rootlen = len(target_dir) + 1
        for base, dirs, files in os.walk(target_dir):
            for file in files:
                fn = os.path.join(base, file)
                zipobj.write(fn, fn[rootlen:])

    zipfolder(unique_path, unique_path)

    #open the zip file
    path_to_file = os.path.join(data_path, 'outputs', access_code, 'nasaaccess_data.zip')
    f = open(path_to_file, 'r')
    myfile = File(f)

    #download the zip file using the browser's download dialogue box
    response = HttpResponse(myfile, content_type='application/zip')
    response['Content-Disposition'] = 'attachment; filename=nasaaccess_data.zip'
    return response

```

Figure A-16: download_data() code

APPENDIX B. TECHNICAL DOCUMENTATION: SWAT DATA VIEWER

The SWAT Data Viewer was developed to act as a virtual and open access data store for stakeholders and decision-makers to view and download SWAT model inputs and outputs for their area of interest. Unlike many of the SWAT related web applications reviewed in Chapter 2, this application is completely modular meaning that it can be duplicated, customized and served from any server running the Tethys platform and can display the data from any valid SWAT model. This chapter will outline the basic functionality of the application along with an in-depth explanation of the various JavaScript and python functions that make the app run.

B.1 Data requirements and file structure

This application provides access to a wide array of watershed specific information including reach, subbasin, and HRU specific outputs from the SWAT model along with the climate and land model input data used for the SWAT model. All of the data and model output files are stored within the app's hosting server following the file structure shown in Figure B-1 and are ultimately uploaded to the app's PostgreSQL database and geoserver. Table B-1 gives a description of each file within the data file structure. The "swat_data" folder can be stored anywhere on the hosting server as long as the folder is given read and write permissions. The upload_new_model.py and config.py files in the app package allows the user to specify all the file paths that the app needs to find the data.

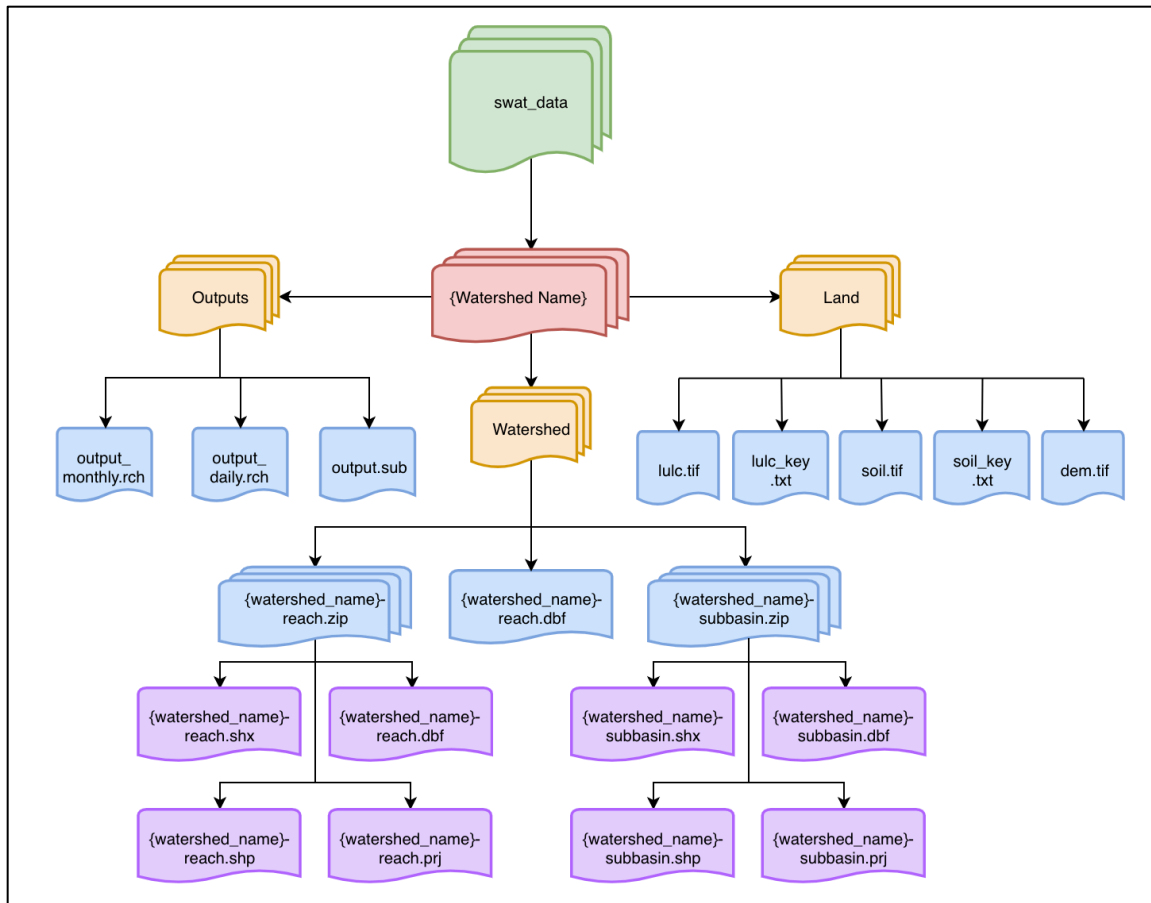


Figure B-1: Data files required for SWAT Data Viewer to work

Table B-1: Description of all required files in SWAT Data Viewer

File name	Description	Purpose
output_monthly.rch	File containing monthly model outputs for each river reach in the model	Time series plots of model outputs
output_daily.rch	File containing daily model outputs for each river reach in the model	Time series plots of model outputs
output.sub	File containing daily model outputs for each subbasin in the model. <i>Note: the app currently only works with the daily .sub file</i>	Time series plots of model outputs
reach.dbf	Database file containing the river connectivity information (i.e. stream IDs, IDs of the upstream and downstream river reaches) <i>Note: If the watershed and streams were delineated using the ArcHydro plugin for ArcMap, these attributes should automatically be there</i>	Identifying upstream river segments and subbasins

Table B-1 Continued

File name	Description	Purpose
lulc.tif	TIFF raster file containing land use and land cover information <i>Note: Coverage extent of this file should match or exceed the boundary of the modelled watershed</i>	Computing and displaying land use/land cover coverage statistics over a user defined subset of the watershed
lulc_key.txt	Text file containing a comma delimited table mapping each land use/land cover code to a land cover category, subcategory, hexadecimal color for the category, and hexadecimal color code for the subcategory. See Figure B-2 <i>Note: The colors are used to match a pie chart of coverage percentages to the colors displayed on the map</i>	Helper file for translating land use/land cover codes into category and subcategory names understandable to the user
soil.tif	TIFF raster file containing soil type information <i>Note: Coverage extent of this file should match or exceed the boundary extent of the modelled watershed</i>	Computing and displaying soil type coverage statistics over a user defined subset of the watershed
soil_key.txt	Text file containing a comma delimited table mapping each soil type code to its name and a display color. <i>Note: The colors are used to match a pie chart of coverage percentages to the colors displayed on the map</i>	Helper file for translating soil type codes into names and colors understandable to the user
dem.tif	TIFF raster file containing the elevation information for the watershed. <i>Note: Coverage extent of this file should match or exceed the boundary extent of the modelled watershed</i>	Used for running the nasaaccess functions through the SWAT app

B.2 App architecture

To make this data as readily accessible and easily readable to the user, the app uses custom python functions that leverage commonly used python modules such as pandas, gdal, and numpy. Using the application project structure shown in Figure B-2 and the Tethys MVC architecture shown in Figure B-3, the SWAT Data Viewer application allows a user to query any combination of SWAT outputs and date ranges within the model simulation date range for any location in the model. All of the user's data query options are available in home.html view as

drop down menus, date pickers, and selectable features on a map. Table B-2 gives a brief description of the role each file in the app package plays in the general functionality of the app.

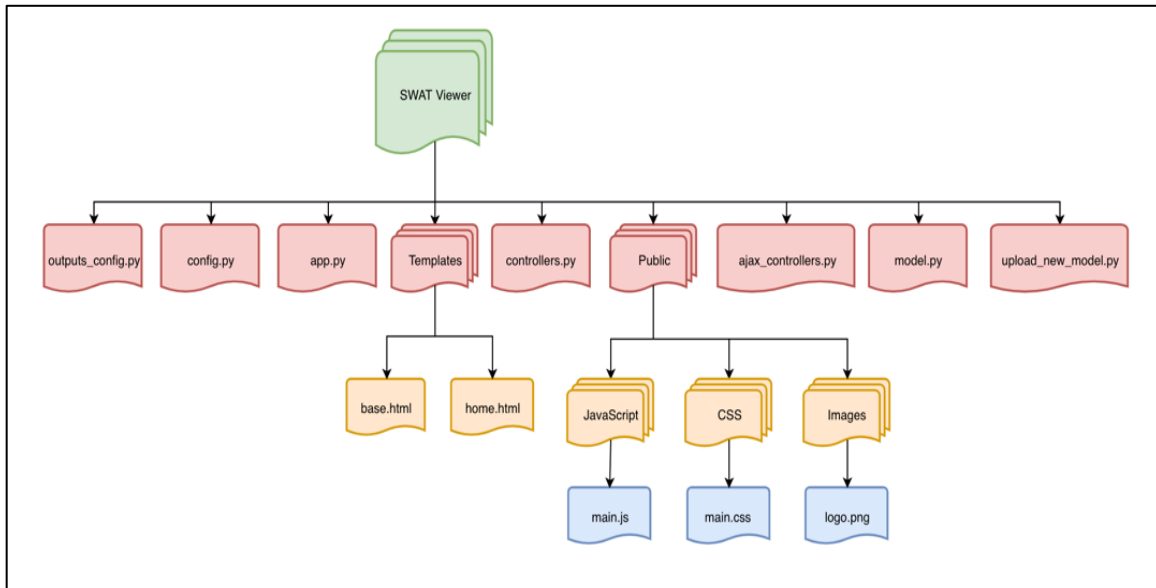


Figure B-2: SWAT Data Viewer application architecture

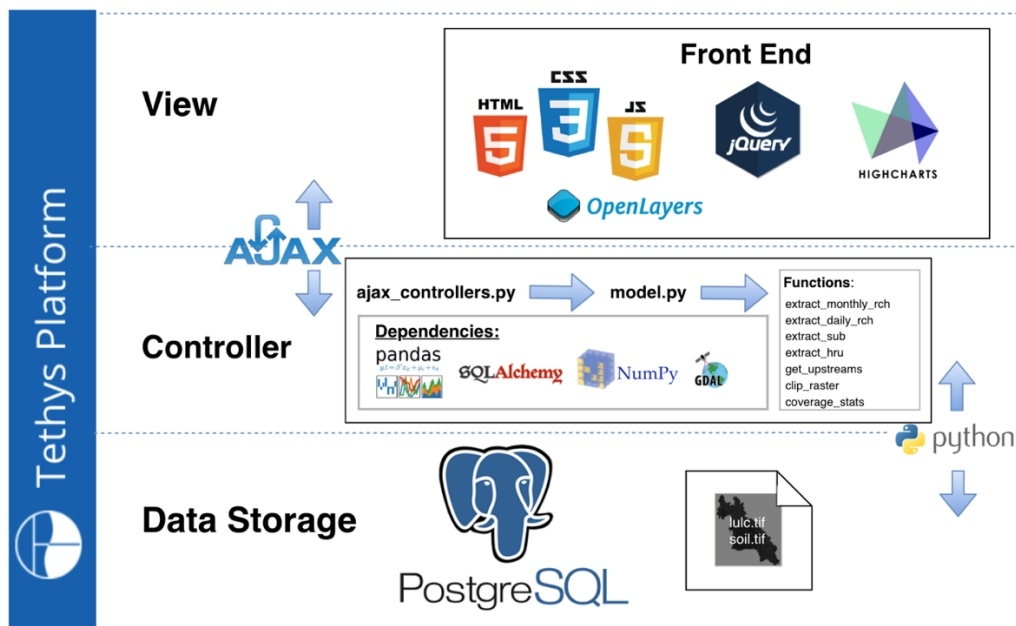


Figure B-3: MVC framework for the SWAT Data Viewer

Table B-2: Description of each file in the app package

File	Role
outputs_config.py	Contains the list of variables and names for the SWAT output files
config.py	Specifies file paths and geoserver URLs unique to hosting server
app.py	Basic application information (name, color, description, URL maps)
controllers.py	Sets up the initial view of the app dropdowns and the date pickers
ajax_controllers.py	Manages communication between front-end and back-end
model.py	Contains all independent python functions for processing data
upload_new_model.py	Uploads all new models data to database and geoserver
home.html	Templates handling all the individual elements in the web interface
main.js	Map visualization and time-series and pie chart plotting
main.css	Styling of HTML elements
logo.png	Image file (.jpg, .png, .gif) of the app logo

B.3 Initializing PostgreSQL database and uploading data

The SWAT Data Viewer uses a PostgreSQL database to store all of its data and to facilitate data querying. Storing all of the data on a database significantly improves app response time during SWAT output data queries and helps to keep the hosting server clean by not having to store all the relevant files permanently.

The first time the app is installed on a new server, the database and all of its tables are automatically created. From there, the app administrator can upload any number of SWAT models to the database which will automatically update the app to allow users to select a model and view data for it. Table B-3 contains a description of each table in the app's database.

Table B-3: Description of tables in the SWAT Data Viewer PostgreSQL Database

Table	Description
watershed	Stores the watershed name and a unique ID for each SWAT model uploaded to the database The ID is used to link all of the other tables.
watershed_info	Stores all the information on available dates and SWAT output variables available for the user to view This table is read each time a user selects a new model to view and updates the date and variable select dropdown menus.
output_rch_day	Stores all the daily data from output.rch
output_sub	Stores all the data from output.sub
lulc	Stores the lookup table information for the land use/land cover raster file
soil	Stores the lookup table information for the soil raster file
stream_connect	Stores the connectivity information for the streams and subbasins

The `upload_new_model.py` file in the app package is what app administrators can use to upload a new SWAT model to the app. The functions within `upload_new_model.py` will read in all of the files in the “swat_data” folder shown in Figure B-1 and upload it to the corresponding data tables in the database (SWAT output files and lookup tables) and Geoserver (.tif and .shp files). To run this script, the app administrator will specify the path the “swat_data” folder containing the new model data, the new watershed/model name (must be unique), the variables from the rch and sub files that need to be uploaded to the database, and the database and Geoserver connection information (Figure B-4).

```

# User specified options
watershed_name = 'lower_mekong' #name of watershed to be used throughout the app (needs to be different from pre-existing
data_path = '/home/ubuntu/swat_data/lower_mekong/' #path to folder containing all data for new model
sub_vars = ['PRECIPmm', 'PETmm', 'ETmm', 'SURQmm', 'GW_Qmm'] #vars from output.sub file to upload to db
rch_vars = ['FLOW_OUTcms', 'SEDCONCmg/kg', 'NO3_OUTkg', 'ORGP_OUTkg'] #vars from output.rch file to upload to db

#database specs
db = {'name': '...',
      'user': '...',
      'pass': '...',
      'host': '...',
      'port': '...'}

#geoserver specs
geoserver = {'rest_url': '...',
             'wms_url': '...',
             'wfs_url': '...',
             'user': '...',
             'password': '...',
             'workspace': '...'}

```

Figure B-4: User specified options for upload_new_model.py

After changing the options mentioned above, the administrator can then run the upload_new_model.py script using the “python upload_new_model.py” command in the terminal. The functions described in Table B-4 below are all run within the upload_new_model.py script. After each function is complete, the SWAT Data Viewer application will automatically update to allow the user to view the newly uploaded data.

Table B-4: Individual functions within upload_new_model.py

Function	Role
new_watershed()	Creates a new watershed name and ID in the watershed table
upload_swat_outputs()	Reads through output.rch and output.sub and creates a row in the output-{rch or sub} table for each day-feature-variable combination in the file
upload_shapefiles()	Uploads the reach and subbasin shapefiles (.zip files) to the geoserver used by the app

Table B-4 Continued

Function	Role
upload_stream_connect()	Reads the {watershed_name}-reach.dbf file and copies the connectivity data (feature ID and to_node) for each feature to the stream_connect table
Upload_tifffiles()	Uploads the soil and lulc TIFF files to the geoserver used by the app
Upload_lulc_key()	Copies the lulc_key.txt file to the lulc table in the database
Upload_soil_key()	Copies the soil_key.txt file to the soil table in the database

B.4 Map publishing

All the geospatial information displayed in the app is facilitated through geoserver. Once uploaded to a geoserver, the watershed basins, streams, land use/land cover layer, and soil type layer are available as WMS services. The app can access any geoserver with a public facing IP address by simply changing the geoserver_url variable in the main.js file and the geoserver object in config.py.

The following is an example of the WMS request that is made to retrieve the subbasin. Using the OpenLayers JavaScript API, WMS requests like this are generated dynamically to update the map view based on user requests within the application interface.

http://216.218.240.206:8080/geoserver/wms/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&FORMAT=image/png&TRANSPARENT=true&LAYERS=swat:lower_mekong-subbasin&BBOX=6.204590651114307,91.01387164397141,28.79540934888569,117.98612835602859&WIDTH=1736&HEIGHT=1454

Each of the layers available for display in this app are styled using Styled Layer Descriptor (SLD), which is an XML-based language available to use within the geoserver interface.

B.5 App functionality

The initial view of the app, shown in Figure B-5, allows the user to select a watershed (the list of available watersheds is produced based on the folders within the “swat_data” file path defined in config.py file) and view the various spatial files (subbasins, stream lines, lulc, and soil) for that watershed. In the navigation pane on the left, the user can toggle the visibility of each of the different layers.

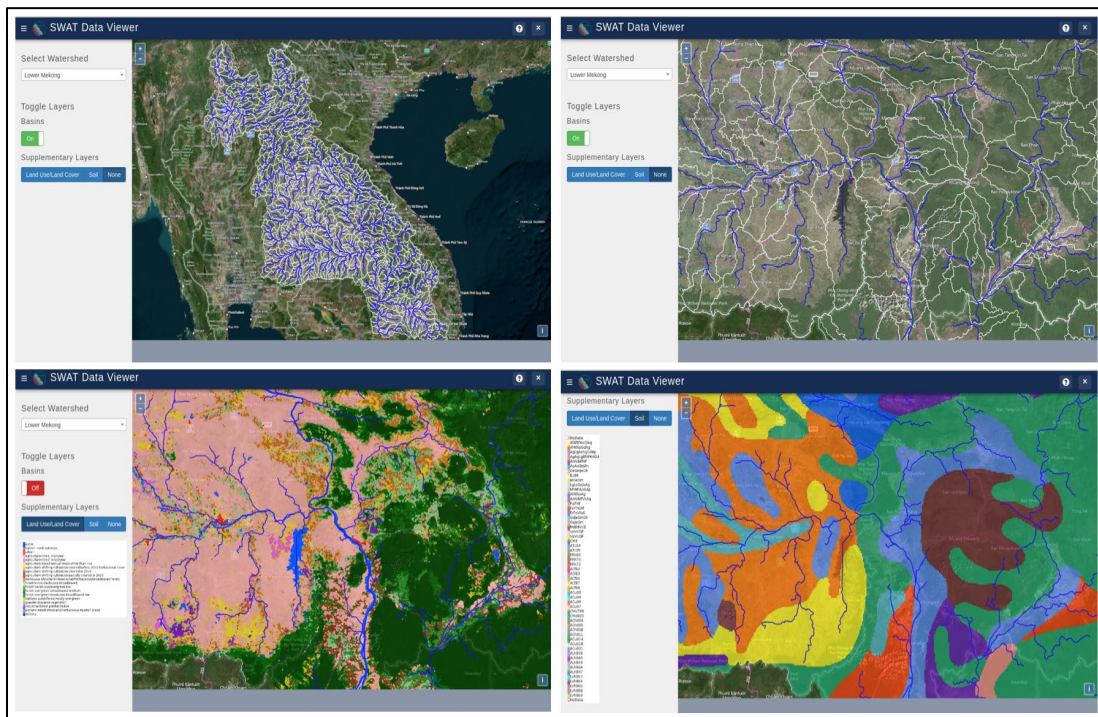


Figure B-5: Initial view of the SWAT Data Viewer with layer toggling

Each time the app is initialized in a browser, a new folder is created and named with unique 5-digit code. This folder acts as a virtual “shopping cart” for the user to add data, both spatial data in the form of geojson and tif files and temporal data as csv files, and to download at the end of their session.

Using the WMS capabilities of the subbasins and streams layers, the user can select any feature (subbasin or stream segment) and access further functionality within the app. When a feature is selected, the app reads the WMS response from the selected layer and obtains the feature ID from its attributes. This ID corresponds to the subbasin and stream ID used in all of the SWAT output files so it is saved in the browser's session storage to be used in any of the user's data queries. Once the selected feature ID is obtained, there are two processes that happen concurrently. On the front end, a data modal appears at the center of the screen and small maps are initialized for each tab within the modal. On the back end, a JavaScript function passes the feature ID to the python controllers (ajax_controllers.py) on the back end using AJAX and runs the `get_upstreams()` function.

Function: `get_upstreams()`

The stream networks used within the SWAT model contain connectivity information (i.e. upstream and downstream IDs) within their attributes which are uploaded from the stream reach .dbf file the `stream_connect` table in the database. The `get_upstreams()` function (Figure B-6) allows the user to access information about the contributing area upstream of the selected reach by creating a list of IDs for all of the streams and catchments upstream of the selected reach.

```
def get_upstreams(watershed_id, streamID):
    Session = Swat2.get_persistent_store_database(db['name'], as_sessionmaker=True)
    session = Session()
    upstreams = [int(streamID)]
    temp_upstreams = [int(streamID)]

    while len(temp_upstreams)>0:
        reach = temp_upstreams[0]
        upstream_qr = """SELECT stream_id FROM stream_connect WHERE watershed_id={0} AND to_node={1}""".format(watershed_id, reach)
        records = session.execute(text(upstream_qr)).fetchall()
        for stream in records:
            temp_upstreams.append(stream[0])
            upstreams.append(stream[0])
        temp_upstreams.remove(reach)
    return upstreams
```

Figure B-6: `get_upstreams()` code

The function reads through the “stream_connect” database table and creates a list of all the streamIDs that are upstream of the selected stream. This list is then passed back to the front end and added to the WMS URLs as a CQL filter for both the streams layer and the subbasin layer so that only the upstream reaches and subbasins are displayed in maps within the data modal. With the new WMS URLs for the upstream streams and subbasins, geojson objects are created and saved into the users “data cart” (the temporary folder with the unique 5-digit name). These json files are used later for various geoprocessing functions.

Function: clip_raster()

Once the geojson files are saved to the user’s unique folder, the clip_raster() function is called. This function takes the original land use/land cover and soil tif files in the “swat_data” file directory and clips them to the extent of the upstream subbasin geojson file using the GDALWARP functionality within the gdal package (Figure B-7). These new subset versions of lulc.tif and soil.tif are saved into the user’s data cart with the naming following name convention: {watershed name}_upstream_{raster type: lulc or soil}_{featureID of outlet stream segment} (e.g. lower_mekong_upstream_lulc_5.tif).

```
subprocess.call(
    'gdalwarp --config GDALWARP_IGNORE_BAD_CUTLINE YES -cutline {0} -crop_to_cutline -dstalpha {1} {2}'
    .format(input_json, input_tif, output_tif), shell=True)
```

Figure B-7: clip_raster() code

The clip_raster() function then checks the current geoserver to see if the clipped raster coverage for that feature has already been uploaded. If not, the function uploads the coverage to geoserver as a coveragestore. From there, the ajax_controller passes a json object containing the

bounding box and coordinate system information from the geojson files back to the front end so that the modal maps can display the subsetted stream, subbasin, lulc, and soil layers zoomed fully to their extent as shown in Figure B-8.

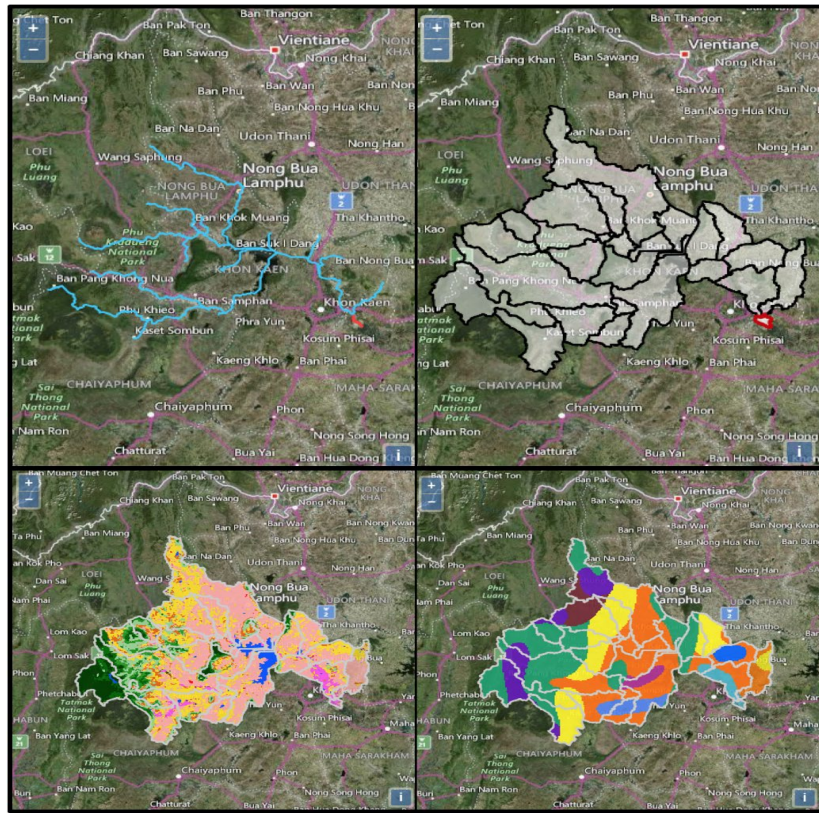


Figure B-8: Layer configurations for maps in the data modal

Once in the data modal (Figure B-9), `get_upstreams()` and `clip_raster()` having finished, the user is given access to a wide range of data querying and analysis functions for the SWAT model outputs produced for the selected watershed. These functions are separated into multiple tabs within the modal so that the user can easily focus on the data that they're interested in.

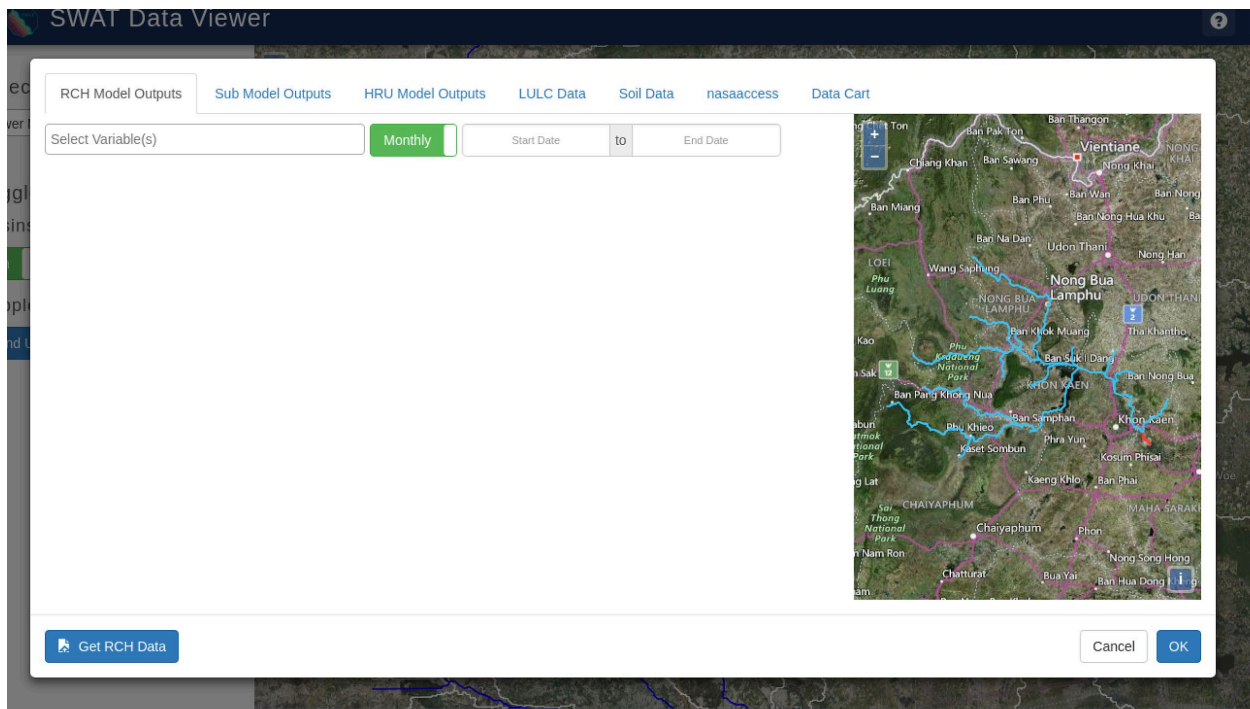


Figure B-9: Data modal view

Because the output.rch and output.sub data is formatted similarly, the first two tabs function in an almost identical way. Their only differences lie in the model variables that the user can select from. Within each of these tabs (Figure B-10), the user will see a “Select Variable(s)” dropdown menu, start and end date selector dropdowns, a map view showing either the upstream stream segments (in RCH tab) or the upstream subbasins (in SUB) with the selected feature highlighted in red, and a button at the bottom to “get data”. The variable and date selector dropdown menus are populated based on the variables and date ranges specified in the “watershed_info” table in the PostgreSQL database.

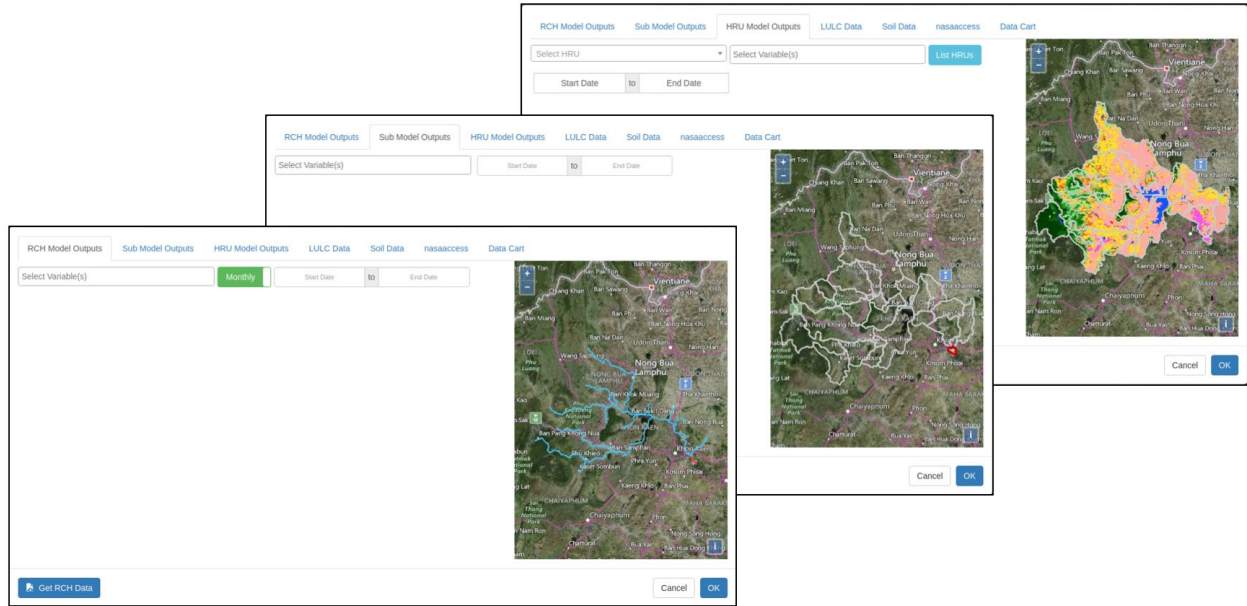


Figure B-10: Tabs for querying SWAT outputs

The drop-down menus at the top of each tab allows the user to select one or multiple variables and a date range that they want to view time series data at the selected stream or subbasin. Once variables and a date range are selected, the “Get {SUB or RCH } Data” button passes all the necessary information (watershed name, stream/subbasin ID, date range, monthly or daily time step) to the ajax controllers which, in turn, initiates a data extraction function.

Function: `extract_rch()` and `extract_sub()`

The complex nature of the SWAT output files is handled by a set of functions dedicated specifically to querying the `output_sub` and `output_rch` tables in the PostgreSQL database. There is a specific function within the `models.py` file for each of the output types. Each of the functions first creates a python dictionary (Figure B-11 shows the dictionary created for SUB outputs and Table B-4 defines each key-value pair within the dictionary) that will contain all of the necessary

metadata and time series data and ultimately be passed back to the front end for visualization using AJAX and the HighCharts JavaScript API.

```
subDict = {'Watershed': watershed,
          'Dates': daterange_str,
          'ReachID': subid,
          'Parameters': parameters,
          'Values': {},
          'Names': [],
          'Timestep': 'Daily',
          'FileType': 'sub'}
```

Figure B-11: Python dictionary created by extraction functions and passed to front end

Table B-5: Description of key-value pairs in time-series data dictionaries

Key	Value	Data Type
Watershed	Name of the watershed as it is in the “swat_data” file system (e.g. “lower_mekong”)	String
Dates	List containing the date at each timestep in the date range (e.g. [“jan 01, 2001”, “jan 02, 2001, ...”])	Array of Strings
ReachID	Feature ID of the selected stream, subbasin, or HRU	Integer
Parameters	List of SWAT variable codes that the user selected (e.g. [“PRECIPmm”, “ETmm”])	Array of Strings
Values	Dictionary that will contain all time series data for each variable in “Parameters” (e.g. {variable1:[(timestep1, value1),...], variable2:[(timestep1,value1),...]}))	Dictionary of tuples
Names	List of human readable variable names (e.g. [“Precipitation (mm)”, “Evapotranspiration (mm)”])	Array of Strings
Timestep	Is the data in daily or monthly time steps	String
Filetype	Did the data come from output.rch, output.sub, or output.hru? (e.g. “sub”)	String

The function then opens the database table (output_rch or output_sub). With the table open, it quickly finds the start date, selected feature ID, and selected variable(s) and writes the

values to a python array. The function then adds the time series data array to the dictionary which is then returned to the ajax_controllers, converted into a JSON response object, and passed back to the front end for visualization.

Figure B-12 shows the key aspects of the extract_sub function. The extract_rch function works in a very similar fashion with only minor adjustments to account for the differences in the output data.

```
def extract_sub(watershed, watershed_id, start, end, parameters, subid):
    dt_start = datetime.strptime(start, '%B %d, %Y').strftime('%Y-%m-%d')
    dt_end = datetime.strptime(end, '%B %d, %Y').strftime('%Y-%m-%d')
    daterange = pd.date_range(start, end, freq='1d')
    daterange = daterange.union([daterange[-1]])
    daterange_str = [d.strftime('%b %d, %Y') for d in daterange]
    daterange_mil = [int(d.strftime('%s')) * 1000 for d in daterange]

    subDict = {'Watershed': watershed,
              'Dates': daterange_str,
              'ReachID': subid,
              'Parameters': parameters,
              'Values': {},
              'Names': [],
              'Timestep': 'Daily',
              'FileType': 'sub'}

    Session = Swat2.get_persistent_store_database(db['name'], as_sessionmaker=True)
    session = Session()
    for x in range(0, len(parameters)):
        param_name = sub_param_names[parameters[x]]
        subDict['Names'].append(param_name)

        sub_qr = """SELECT val FROM output_sub WHERE watershed_id={0} AND sub_id={1} AND var_name='{2}' AND year_month_day BETWEEN '{3}'
watershed_id, subid, parameters[x], dt_start, dt_end)
        data = session.execute(text(sub_qr)).fetchall()

        ts = []
        i = 0
        while i < len(data):
            ts.append([daterange_mil[i], data[i][0]])
            i += 1

        subDict['Values'][x] = ts
        subDict['Names'].append(param_name)
    session.close()
    return subDict
```

Figure B-12: extract_sub() code

Once the JSON object containing the time series data reaches the front end, a function within main.js reads in the information, separates it out into multiple JavaScript variables and

uses those as parameters for HighCharts to use for plotting the data. Figure B-13 shows an example of the time series plots that HighCharts creates.

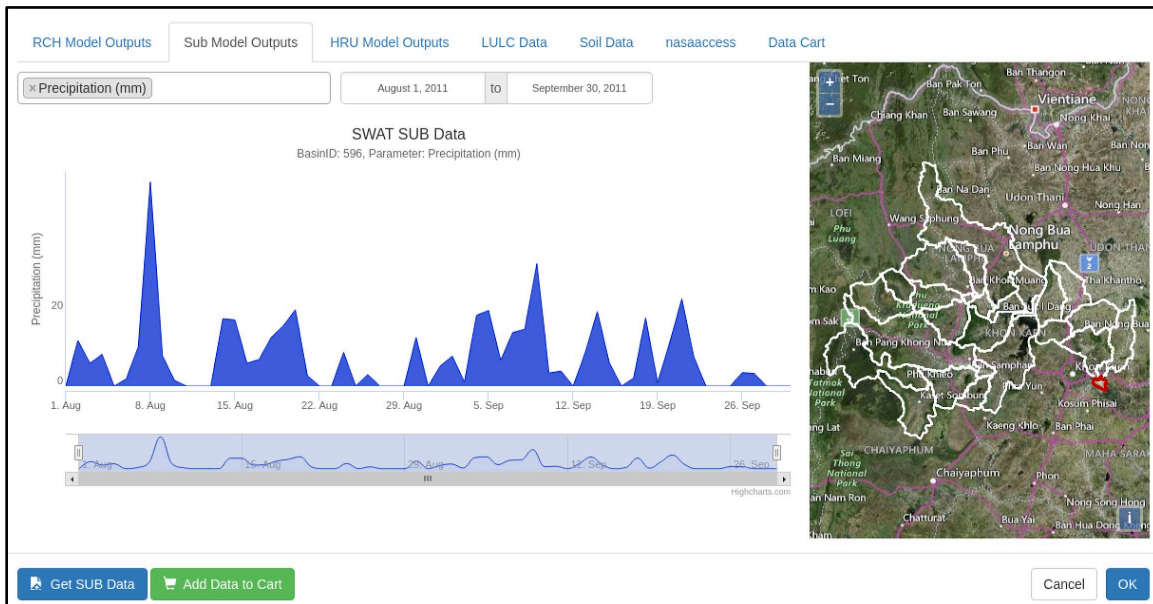


Figure B-13: Example time-series plot in “Sub Model Outputs” tab

The purpose of the LULC and Soil tabs is to give the user a better understanding of the land conditions and characteristics within the watershed that they selected. Within these two tabs, the user will see a map on the right-hand side with either the clipped LULC or Soil layer overlaid with the upstream subbasins layer. In the footer of the modal, a “Compute {LULC or Soil} Statistics” button is available for the user to click and call the coverage_stats() function which computes the percent coverage of each different land use/land cover class or soil type within the selected area.

When the “Compute {LULC or Soil} Statistics” button is clicked, AJAX passes the user’s unique ID, the watershed name, the featureID of the selected feature, and the raster type

(lulc or soil) to ajax_controllers.py to be used as input parameters for the coverage_stats() function.

Function: coverage_stats()

The coverage_stats() function uses the gdal and numpy python packages to read the TIFF raster files (clipped lulc and soil files) within the user's data cart folder and to convert them into an array. From there, all the unique values within the array are identified and a count (# of occurrences) is computed for each value (Figure B-14). Then, using the size (total number of non-"No Data" cells) of the TIFF file a percent coverage is computed for each unique value (Figure B-15).

```
tif_path = temp_workspace + '/' + str(uniqueID) + '/' + watershed + '_upstream_' + str(raster_type) + '_' + str(outletID)
ds = gdal.Open(tif_path) #open user-requested TIFF file using gdal
band = ds.GetRasterBand(1) #read the 1st raster band
array = np.array(band.ReadAsArray()) #create an array of all values in the raster
size = array.size #get the size (pixel count) of the raster
unique, counts = np.unique(array, return_counts=True) #find all the unique values in the raster
unique_dict = dict(zip(unique, counts)) #create a dictionary containing unique values and the number of times each occurs
```

Figure B-14: coverage_stats() code for identifying unique values in raster

```
#compute percent coverage for each unique value
for x in unique_dict:
    if x not in nodata_values:
        unique_dict[x] = float(unique_dict[x]) / size * 100
```

Figure B-15: coverage_stats() code for computing % coverage for each unique value

Another important aspect of the coverage_stats() function is its color-coding ability. The function opens the {lulc or soil} database table and extracts the information needed to map the raw values (integer codes) in the TIFF file to a recognizable name (i.e. the value "16" in the

lulc.tif file corresponds to an “urban” land cover type) and a color that matches the style for the WMS layer. Once all of this information is obtained, it is saved to a python dictionary and sent back to the front end for visualization. There is a slight difference in the value to name and color mapping process between the land use/land cover and soil type processes. For land use/land cover files, coverage_stats() creates a class and a subclass for each lulc type (i.e. “rice crops” and “mixed annual crops - non rice” are their own subclasses but they are also part of the “agriculture” class) (Figure B-16). This is done to “declutter” the data plots on the front end. In the soil type files, on the other hand, each soil type is given its own class (Figure B-17) and is displayed accordingly.

```
if raster_type == 'lulc':

    #lulc is divided into classes and subclasses for easier categorizing and visualization
    lulc_dict = {'classes': {}, 'classValues': {}, 'classColors': {}, 'subclassValues': {}, 'subclassColors': {}}

    for val in unique_dict:
        with open(color_key_path) as f:
            for line in f:
                splitline = line.split(' ')
                splitline = [x.strip() for x in splitline]
                if str(val) not in nodata_values and str(val) in splitline[0]:
                    lulc_dict['subclassColors'][splitline[2]] = splitline[-1]
                    lulc_dict['subclassValues'][splitline[2]] = unique_dict[val]
                    lulc_dict['classes'][splitline[2]] = splitline[1]
                    if splitline[1] not in lulc_dict['classValues'].keys():
                        lulc_dict['classValues'][splitline[1]] = unique_dict[val]
                        lulc_dict['classColors'][splitline[1]] = splitline[-2]
                    else:
                        #add all the % coverage values within a class together
                        lulc_dict['classValues'][splitline[1]] += unique_dict[val]

    return(lulc_dict)
```

Figure B-16: coverage_stats() code for creating lulc dictionary to be sent to front end

```

if raster_type == 'soil':
    #soil type is only divided into soil types and does not have subcategories like lulc
    soil_dict = {'classValues': {}, 'classColors': {}}
    for val in unique_dict:
        with open(color_key_path) as f:
            for line in f:
                splitline = line.split(' ')
                splitline = [x.strip() for x in splitline]
                if str(val) not in nodata_values and str(val) in splitline[0]:
                    soil_dict['classColors'][splitline[1]] = splitline[2]
                    soil_dict['classValues'][splitline[1]] = unique_dict[val]
    return(soil_dict)

```

Figure B-17: coverage_stats() code for creating soil dictionary to be sent to front end

On the front end, the dictionary containing the coverage statistics, names, and colors is retrieved as a JSON object using AJAX. The data in the dictionary is then displayed as a pie chart feature in the HighCharts API. To handle the classes and subclasses for the land use/land cover coverages, HighCharts has “drilldown” capabilities that allow a user to select a certain category in the current pie chart and dynamically create a new pie chart showing all the subcategories within the selected category (Figure B-18).

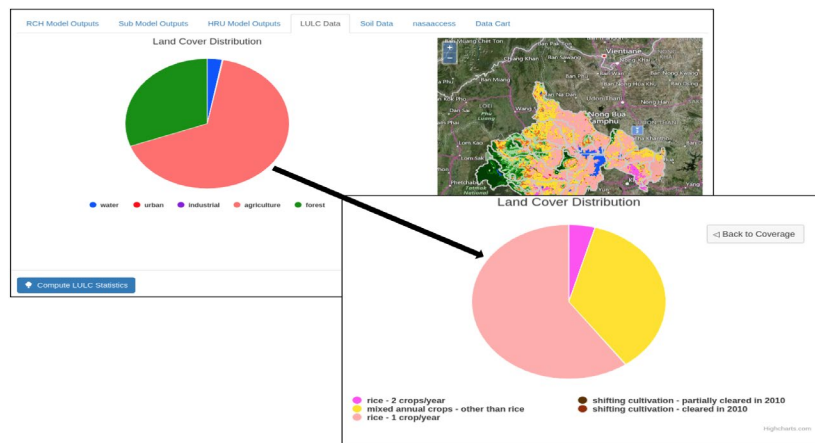


Figure B-18: Example land cover distribution pie chart shown in the “LULC Data” tab

The nasaaccess tab acts as a second interface to initiate the nasaaccess input data processing functions. It does not give the user the ability to upload custom watershed shapefiles

and DEM but does facilitate running the nasaaccess functions on the currently selected upstream subbasin extent as shown in Figure B-19.

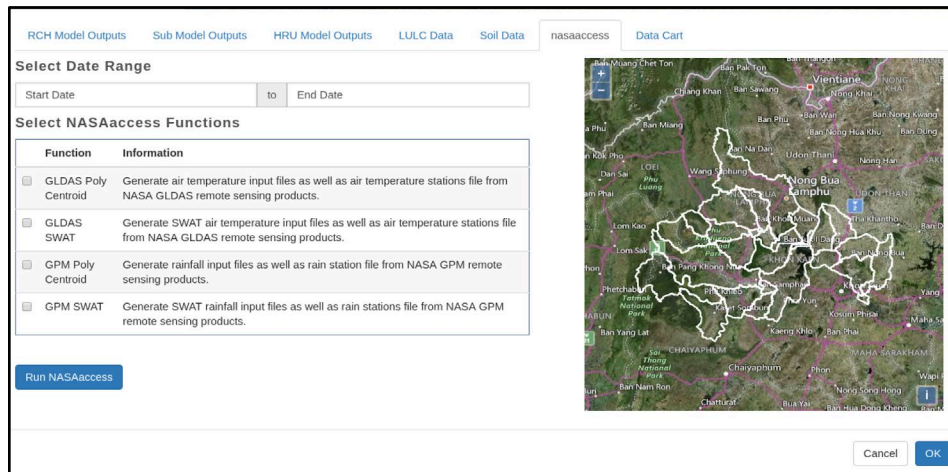


Figure B-19: The nasaaccess interface within the SWAT Data Viewer

The “data cart” has been mentioned multiple times in the above sections. This tab functions similar to the virtual shopping carts on online shopping websites. As the user performs various data queries and data processing requests, they are given the option to add the data to their “data cart”. For example, when viewing a time-series plot of RCH data a button will appear giving the user the option to “Add Data to Cart”.

When the “Add Data to Cart” button is clicked, the timeseries data is sent back to `ajax_controllers.py` in the form of a JSON object and the `write_csv()` function is called.

Function: write_csv()

Using the pandas python package, `write_csv()` simply takes the time-series dictionary and writes the data out line by line using the following format: UTC offset (sec), Date, Variable1, Variable2,... (Figure B-20). This file is saved to the user’s unique folder.

UTC Offset (sec)	Date (m/y)	FLOW_OUTcms	ORGN_OUTkg
1233360000	1/2009	9.232	0
1235779200	2/2009	4.796	0
1238457600	3/2009	6.537	7.398
1241049600	4/2009	6.043	0.1406
1243728000	5/2009	21.32	399.8
1246320000	6/2009	155.6	2143
1248998400	7/2009	368.2	20080
1251676800	8/2009	302.1	2734
1254268800	9/2009	188.1	1818
1256947200	10/2009	81.9	3.86
1259539200	11/2009	28.63	0.3057
1262217600	12/2009	12.48	0
1264896000	1/2010	8.213	2.986

Figure B-20: Example time-series csv file written by the app

Once a file has been written and saved to the user’s unique folder, a JavaScript function adds the file name and information to a list in the data cart tab for the user to keep track of the data already prepared for download (Figure B-21). When the user is satisfied with the data in the cart, they can then select the “download” button. This will compress all the files listed in the data cart tab into a .zip file and then download it to the user’s local computer.

RCH Model Outputs Sub Model Outputs HRU Model Outputs LULC Data Soil Data nasaaccess **Data Cart**

Timeseries data available for download

Data Type	Parameters	Time Step	Start Date	End Date	Stream/Basin ID
rch	FLOW_OUTcms	Monthly	012009	072015	596
rch	FLOW_OUTcms	Daily	08012011	10312011	596
sub	PRECIPmm	Daily	08012011	09302011	596

Spatial data available for download

Data Type	File Type	Outlet Stream ID
lulc	TIFF	596
soil	TIFF	596
reach_upstream	JSON	596
basin_upstream	JSON	596

Download Cancel OK

Figure B-21: The “Data Cart” tab