Team 21: Transforming Drainage Project- Design Document

Jacob Conley, Brandon Kent, Alexis Williams, Jordan Hagedorn

Purpose

The Transforming Drainage Project aims to research, develop and improve upon various types of water drainage storage systems and their associated technologies and practices. The Reservoir Planning Tool is a web-based application that calculates and displays the potential benefits of various sizes of irrigation reservoirs across the Midwest. This tool is has been implemented for engineers, farmers, state/local governments, researchers, and contractors/consultants along with anyone else who needs to be able to calculate variables such as inflow, outflow, and pond water depth, (among others) which are needed to determine reservoir size and the type of water drainage storage system that would be best applied. Our project is an extension of this existing system, where we will provide additional features, ranging from minor updates to output charts to making an automated way to calculate all data points from the grid system (approx. 10,000) within the database and create multiple layers of charts on top of a regional map.

Most software currently available that concerns reservoir development only focuses on an engineering/construction standpoint and not from an agricultural standpoint; meaning that they don't consider environmental factors like precipitation, evaporation, etc. since their reservoirs are designed for residential, commercial, and indoor use.

Functional Requirements

1. Users can download results from the data calculation

   As a user, I would like to:

   A. more easily download results of individual computations
   B. have the option to download results in different file formats
   C. see more than just the average in the result charts
   D. have the option to download each layer of results of all grid cells separately
   E. have the option to download each layer of results of all grid cells together
   F. have an option to save my data to the website database for later use/for other people to use

2. Users will have several approaches to viewing data calculation

   As a user, I would like to:

   A. have an option to calculate all grid cells at once
   B. have an extra tab/external link to results of all grid cells calculation

C. see all results plotted onto a map of grid cells

D. have options to toggle on/off individual layers of results on the map

E. have the option to download each layer of results of all grid cells separately

F. have the option to download each layer of results of all grid cells together

3. Users will have the option to pull from other users

As a user, I would like to:

A. have an option to use other people's data from the website database

B. see a pop-up that explains the quality of other people's data if I choose to use theirs

C. have an option to save my data to the website database for later use/for other people to use

4. Users will have more advanced tools than currently available

As a user, I would like to:

A. have access to more sophisticated graphing capabilities than what is currently present

B. have the option to submit metric measurements as inputs

C. have the option to see charts and results in metric units

Non-Functional Requirements

1. Architecture and Performance

A. We will be following the same platform made by the previous group, which is a combination of Javascript (and Node.js as well), HTML, and CSS as the main basis of the website, along with Python for the database setup and a MySQL server.

B. Our system must continue to be portable to other machines, easy to read and understand, as well as easy to modify and maintain.

C. In regards to performance, steps must be taken to alleviate the computational strain of calculating and graphing the entire 10,000 grid points on the regional Map.

2. Usability

A. Website should be easy to navigate through and pleasing to look at.

B. Make sure that it is intuitive to modify and read the information displayed, as well as download.

C. Large calculations going on in the background must not inhibit the usability of the website by the users, meaning that lag in excess of 10 seconds, or loading time in excess of 2 minutes, cannot be introduced by the main calculation and multiple users' calculations

3. Hosting/Development

A. Will ideally be hosted on a Wordpress site, if that is not possible, our other options include hosting on Apache HTTPD/PHP, IIS 7.5/ASP.NET, or Apache Tomcat 6

B. Currently, the stakeholders are working with a paid developer and Purdue ITAP to make the site publicly available, which is something that we will help with as well.

Core Project Requirements

1. Mass Data Calculation

- The data residing in all 10,000+ grid cells across the region will be ran through the Transforming Drainage Project's algorithm to calculate the various outputs which will then be displayed in layers over the original gridded map

2. Data Input

- The user will be presented with an added option to calculate all data points contained in the database
- The user will also have an option to use data from other users who have uploaded their data to the database

3. Mapping System for Mass Data Calculation

- The results of the grid calculation will be mapped in layers over the entirety of the gridded regions on the map
- Users will be able to toggle on and off the various layers displaying different information

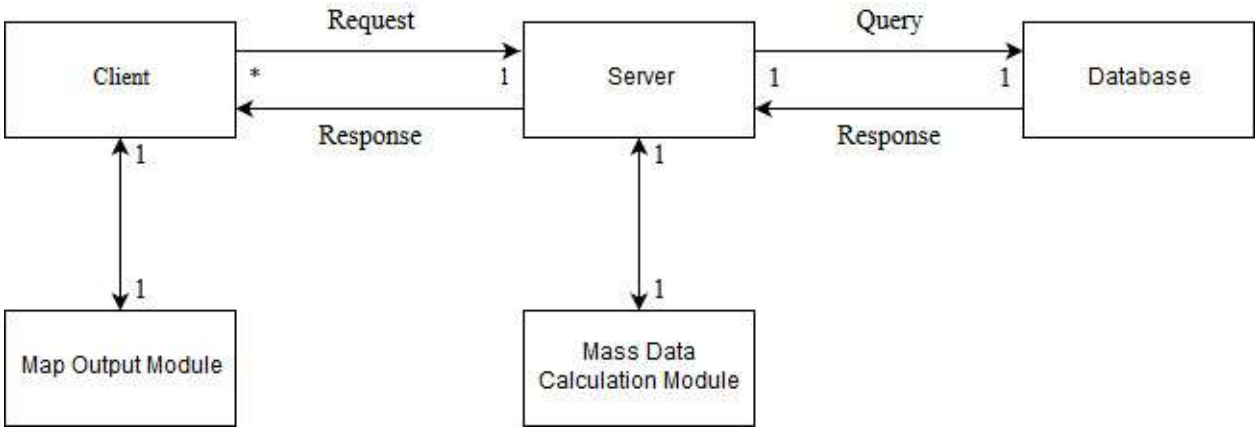
4. Chart Enhancements

- Existing chart capabilities should be enhanced to allow the user more information such as maxima and minima.
- Users should have the option to download and export the charts as .csv file

Design Outline

High level overview

The best strategy for our project is to use a client-server model. The client will gather user inputted data and send it to the server to process. The server will request additional data from the database to compute the given algorithms. With the computations complete the server will send the data back to the client where it will plot the data on maps and graphs and allow the user to manipulate the data shown. Two new modules are being added to the server, the mass data calculation module and the map output module. The former will be used to calculate all of the data points available from the database, which the server will query for, and will then be sent to the map output module to output the information. The latter will take the information from the mass data calculation module and create the layers to be displayed on the original gridded map.



1. Client
    a. Sends data acquisition requests to the server
    b. Receives output data computed by server
    c. Creates maps and charts using data received
    d. Gathers data inputted by the user

2. Map Output Module
    a. Gathers results sent from the server
    b. Plots the information as a series of layers
    c. Displays the layers over the original gridded map

3. Server
    a. Communicates between client and database
    b. Utilizes client's requests in conjunction with items in database to perform necessary calculations
    c. Sends results of calculations for interpretation by client

4. Mass Data Calculation Module
    a. Requests all of the data available from the database
    b. Performs the calculations based on the algorithms for each data point
    c. Returns the results to the client to be interpreted

5. Database
    a. Store irrigation and reservoir data that is provided by the project stakeholder
    b. The server will request information from the database, this will either be a single location or the entire map

Design Issues

1. Map Type
    - Option 1: Leaflet
    - **Option 2: Google Maps**
    - Option 3: RGIS Software

    Leaflet is open source, flexible, lightweight, and provided many features that are essential to what we need to implement (layers, grid overlays, etc.). The same thing goes for RGIS, which is software that is used by our stakeholder in other areas, it provides a very powerful tool for making the layers that would be placed on our map.

    Ultimately, however, we ended up going with Google Maps, which had been implemented previously, so it ultimately made the most sense to continue using Google Maps. RGIS was a serious consideration though, unfortunately we determined that it would be a lot of effort to try and incorporate very specific software into the project and to work out the process for allowing us to use this software in the web application.

2. Mass Data Calculation Output Selection Options
    - Option 1: Extra button
    - **Option 2: Extra tab to choose from at the top**
    - Option 3: Link to external website

    The link to an external website would work as well, but having just a random link would not work well with the design layout, and there's no real reason to have the map be its own separate entity, which would require hosting it as well. The extra button would work

as well, but may cause confusion as to what the user has to input, since the map will need to have very specific input sets for each map's resulting data. The tab is the best choice in this case, since the project stakeholders would like to have certain preset values for three different scenarios, which would be too complicated to have the user manually enter.

3. Charts Library
    - **Google Charts**
    - D3
    - ChartJS
    - Chartlist

D3 is a powerful open source charting library for javascript. It offers multiple ways to customize charts and their data. However, it offers functionality that we don't necessarily need like fancy chart transitions and the ability to enter or exit the chart. ChartJS was another option however they only have six types of charts and functionality is limited on older browsers so this wouldn't fit the need of our project stakeholder. The final option was Chartlist, similar to ChartJS its functionality is limited on older browsers and you have to jump through more hoops to customize the charts.

We decided to use Google Charts since this was already implemented in the previous version of the website. This will allow us to optimise the current charts and add additional data that the project stakeholder requires. The stakeholder told us that the charts needed to hold additional data and allow for users to be able to customize the charts from the website. With this information we determined that we didn't need to redo the charts as a whole, we just had to add additional functionality to the charts.

4. Where should the map be displayed?
    - In a separate tab
    - In a pop-up
    - **Below Inputs**

Using a separate tab to display results would work well aesthetically, as it would not clutter the main page and not cause users to wonder where their results are. Functionally, it also makes sense as it would separate the inputs from the outputs so the users would always know where they need to go to modify the inputs, and view the changes. Pop-ups are a valid option, however they can be annoying for users to deal with, and pop-up blockers may become a very frustrating issue. Currently, results are displayed below the inputs, and the stakeholders like the design, so we made the decision to display the resulting map below the input area.

5. How should the Layers of the map be displayed?
    - Color in each grid square
    - **Overlay precise colors**

The map is set up right now with a grid overlay that allows the user to select a square and see the data for that section. Because of this a choice needs to be made to display a single color in each square and sacrifice accuracy or to overlay precise colors across the map. Because accuracy is important to our stakeholder we decide we need to find a way to layer the colors across the map precisely using included features of Google Maps.

6. How should we handle user saved files?
    - Individual user logins
    - **Open file system**

Users will be able to check a box allowing the file they uploaded to be stored in the database. Originally, we thought about having individual user logins so that each user can use their files without having to re-download them every time; however, the stakeholders did not want to have user logins, but they liked the idea of having shared access to user uploaded data files. So instead we decided to go with an open file system, which is like a dropbox where users can save their files, and other users will have access to them as well. Also, we will have to make sure the data in their .csv file is usable and then rename the file based on the location and time of upload so it can be reused again at a later date with ease.

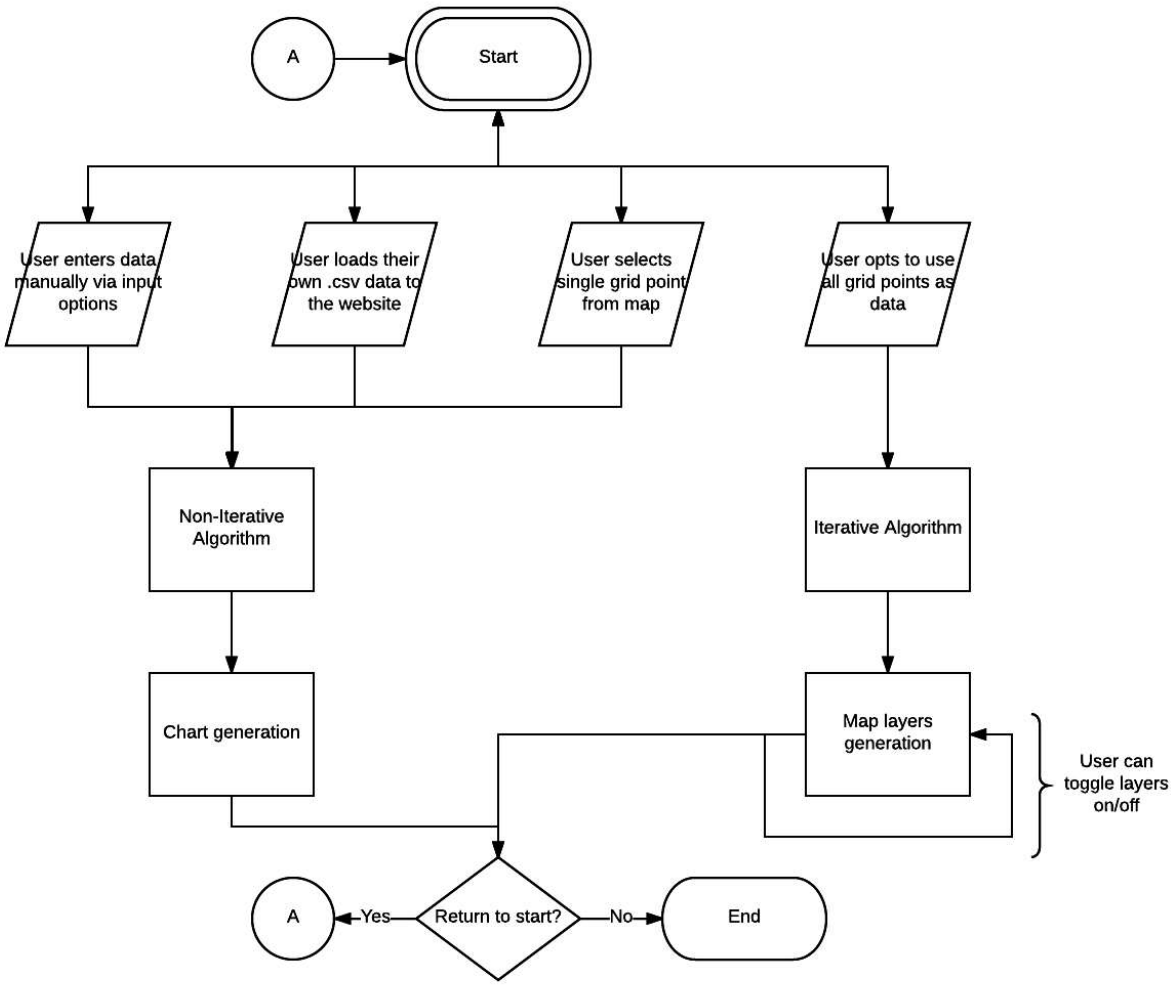
7. How are we going to handle security?

We will practice defensive programming. We are going to make sure that important elements are private in order to limit access to data that isn't necessary. Also through defensive programming we will ensure high quality code to reduce the number of unforeseen bugs and other issues.

Design Details

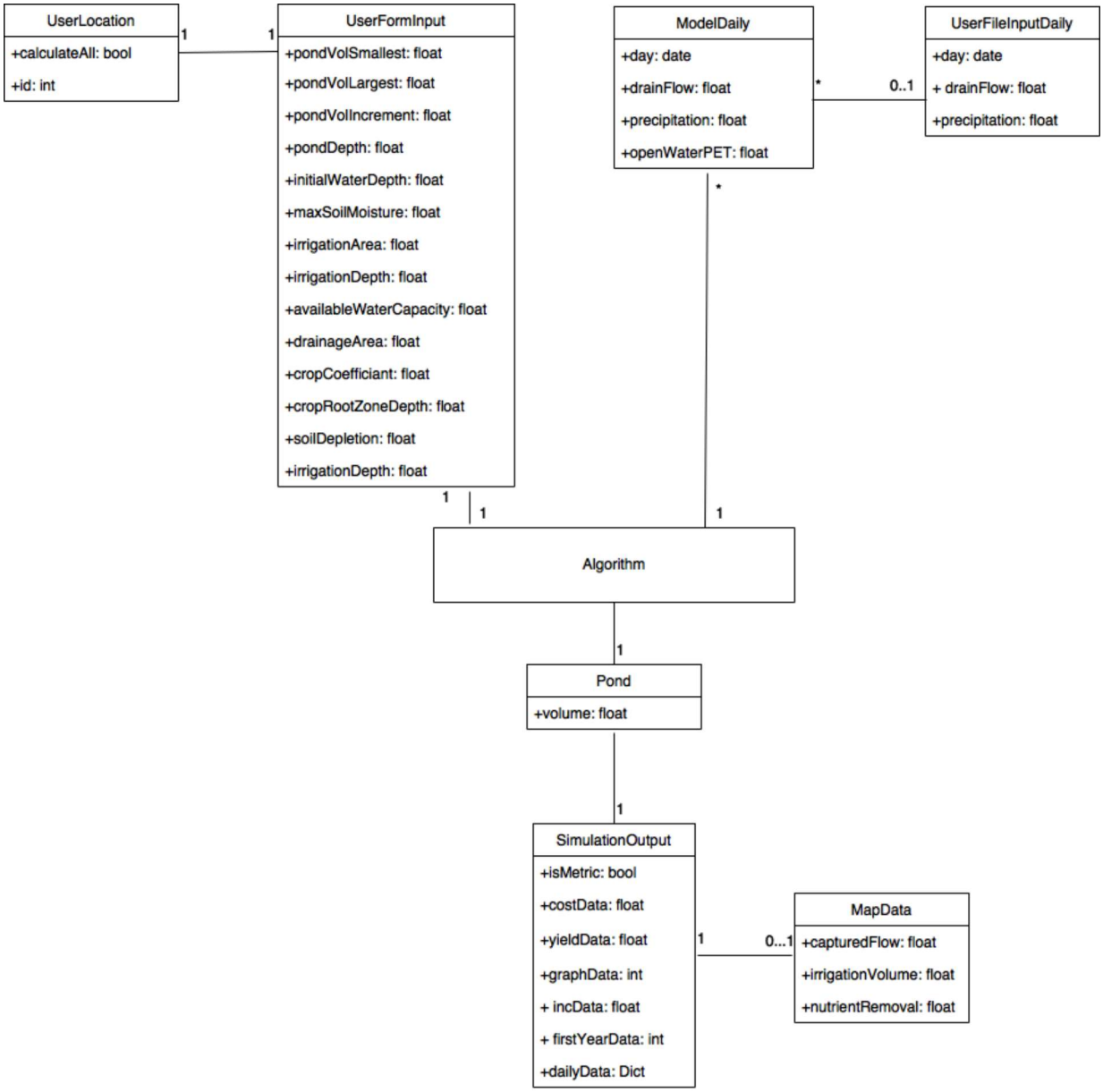
Activity Diagram

The activity diagram below displays the course of a typical user interaction with the web application. At the start, the user has four options to input data: they can use the various input options to manually enter in data, they can download their own, they can specify a specific point from the gridded map, or they can choose to select all grid points on the map. This is where the diagram splits off to show the differences in calculations. The non-iterative algorithm applies the calculations just once, and then generates a series of charts based on the results whereas the iterative algorithm has to apply the calculation to each grid point’s dataset, save each subsequent result, and return all the results to the next step. The map layers generation creates the different map charts which are then layered over the original gridded map. On the other side, the chart generation creates the various plotted graphs. At the end, the user has the option to return to the start.



Class Details

Since we are increasing the functionality of the Transforming Drainage Project's Pond Sizing Tool, some applicable classes already exist. Our features will rely on those classes, as well as additional ones to achieve our primary goal of calculating reservoir benefits for the entire region. User inputs will be stored in the class UserFormInput, and once parsed, the relevant database data is contained in the ModelDaily class. Output information will be located in our MapData class, as well as the SimulationOutput class.



Inputs

These will be provided by the user, either via the user manually entering data, uploading their own data as a .csv file, selecting a single grid point, or selecting an option to use the data of every grid point.

UserFormInput

- Contains user's custom inputs which have been manually inputted using the web application

    - These inputs will be fed into the algorithm to calculate pond depth, incoming and retained water, and costs vs yields and respective impacts, which will then be displayed as charts/graphs
- If user chooses to pick a grid cell (or all grid cells), UserFormInput will not be explicitly stated by the user, but instead will use individual grid cell data from the database
- Every instance of UserFormInput has a UserLocation object tied to it, and is passed into the algorithm alongside ModelDaily to produce instances of Pond, SimulationOutput, and MapData

UserLocation

- User inputs this value by selecting a grid cell on a map
    - The user may instead choose to run the computation with every grid cell at once (across the entire region)
- Holds an ID which corresponds to a GeoJSON layer
- Database contains all information related to each grid cell
- One UserLocation exists for each UserFormInput, and is passed into the algorithm along with ModelDaily to produce our output objects

ModelDaily

- Contains one day's data parsed from The Transforming Drainage Project's database
    - Data includes drainflow, precipitation, and open water potential evapotranspiration
- Each variable listed will be given in millimeters
- One of these objects exists for each day in the range specified by the user
- If the user uploads a CSV file, the ModelDaily instances will take on those values instead of the ones in the database
- ModelDaily's data is passed with the UserFormInput into the algorithm iteratively to account for each instance of ModelDaily, and produces instances of our output classes

UserFileInputDaily

- Only created if the user chooses to upload their own .csv file
- User can specify their own drainflow and precipitation measurements
- .csv is parsed, and will be used in the ModelDaily values in place of database values
    - Only one UserFileInputDaily will be present, and will result in any number of ModelDaily objects

Outputs

After successfully running the algorithm, these objects will be created and are used to construct visuals for the user, which may include: charts and graphs if the user manually inputs data through the web application, the user inputs their own .csv data file, or if the user selects a single location from the map. Map layers will be visible on the regional map with grid cells if the user chooses to use all of the grid cells as data.

Pond

- Contains information about annual and daily output, as well as pond volume of its respective simulation
- Values are dependant on those of the UserFormInput
- The user can choose to specify further information about the pond to generate more detailed charts after the primary algorithm executes

SimulationOutput

- Contains data needed for chart generation in the Google Charts API
    - This data is yearly and daily
    - Includes results such as water bypass volume and storage deficit volume
- One instance of SimulationOutput is created every time user submits their UserFormInput
- Also includes cost and yield impacts to generate even more results

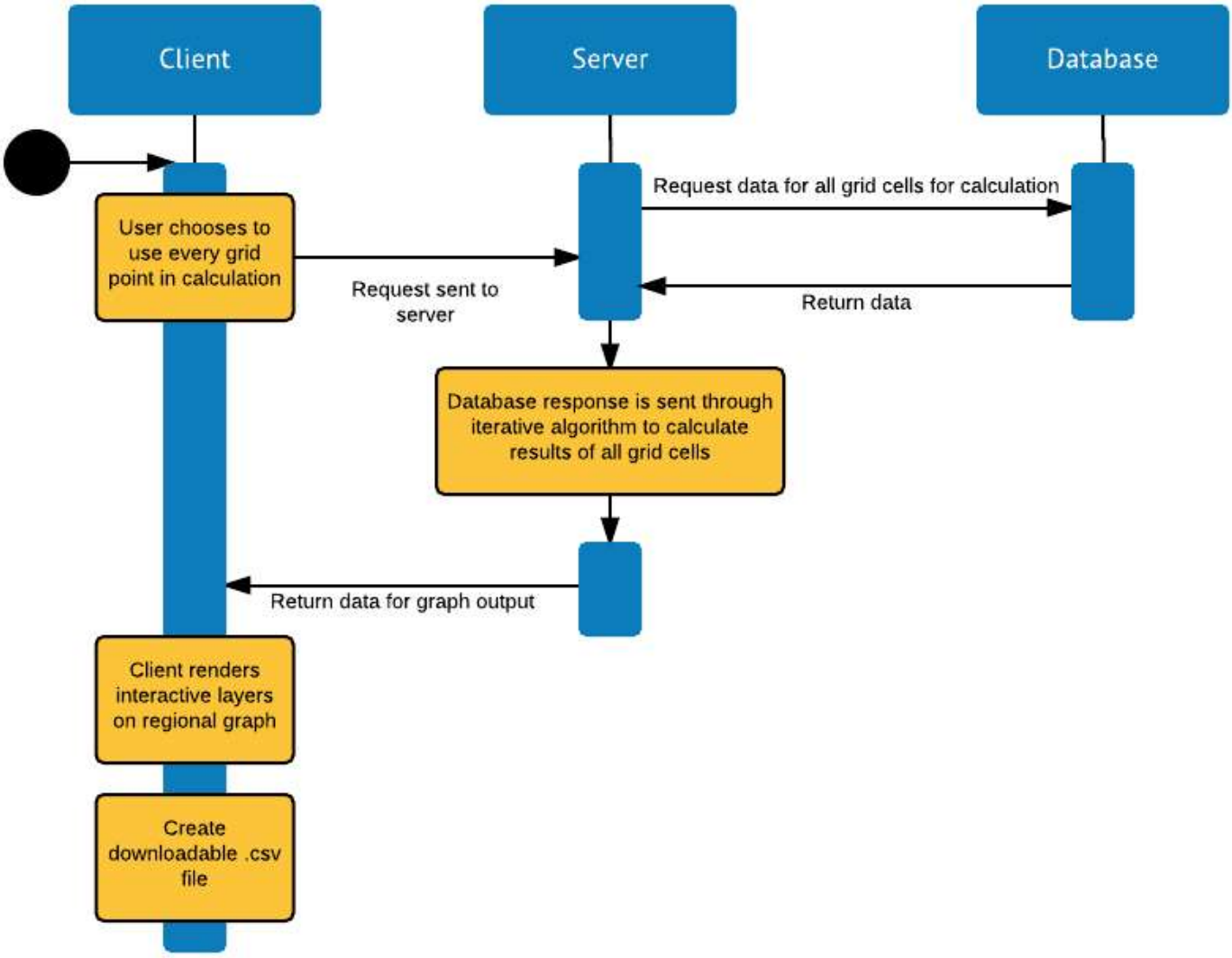
MapData

- Contains data values used to represent map layers in our map output
    - Layers include captured flow, irrigation volumes, and nutrient removal effectiveness
- One instance of MapData is created every time the user submits their UserFormInput, and is also dependant on the ModelDaily values

Sequence Diagrams

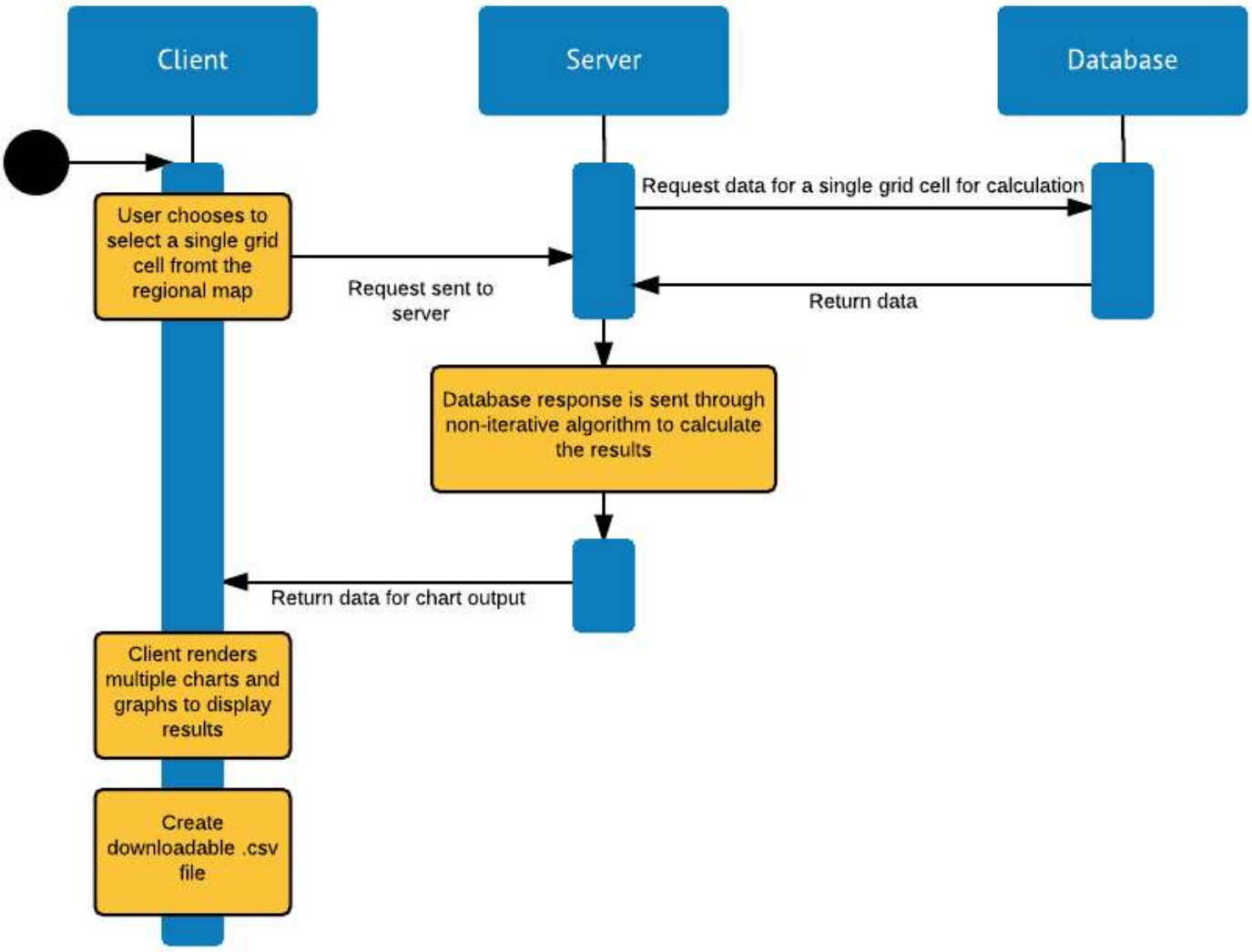
Sequence Diagram For Mass Grid Cell Calculation

To calculate all grid cells at once and see the resulting map, the user will enter inputs on the existing form, but instead of choosing a single cell on the map, they will click a button that takes all grid cells into account. The server obtains the necessary data from the database and sends it through the Transforming Drainage Algorithm. The server sends the algorithm's results to the client, and the client will render a map with toggleable overlays. The user can also choose to save their results as a CSV file.



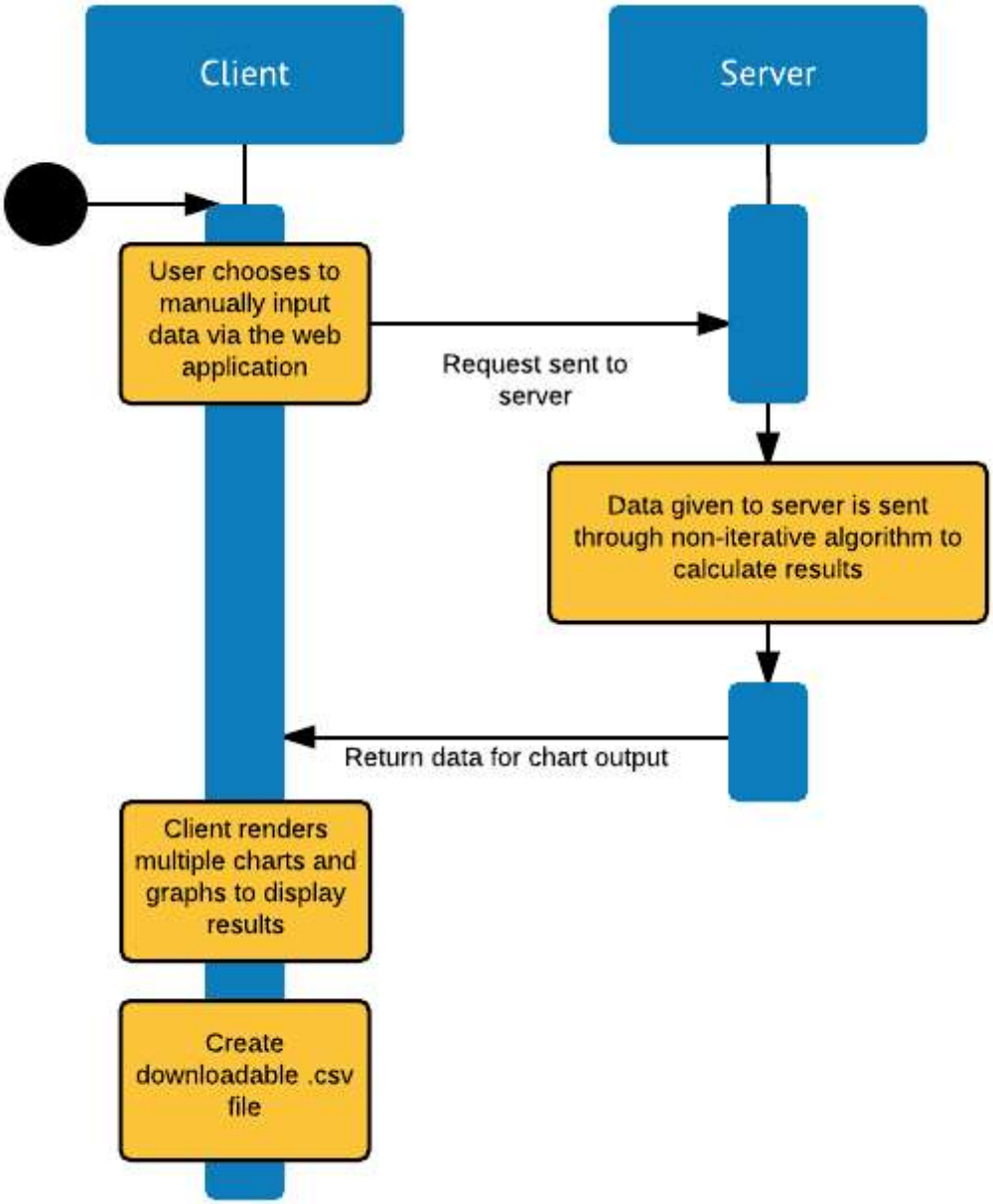
Sequence Diagram For Single Grid Cell Calculation

To calculate a single grid cell, the user must select a single grid cell from the map and then choose to use that data as the inputs. The server will then receive the request and then subsequently request and retrieve data from the database, which will then be sent through the Transforming Drainage Algorithm. The server then sends the results to the client, which renders interactive charts and graphs for the user to visualize the results and download as a .csv file if they want to.



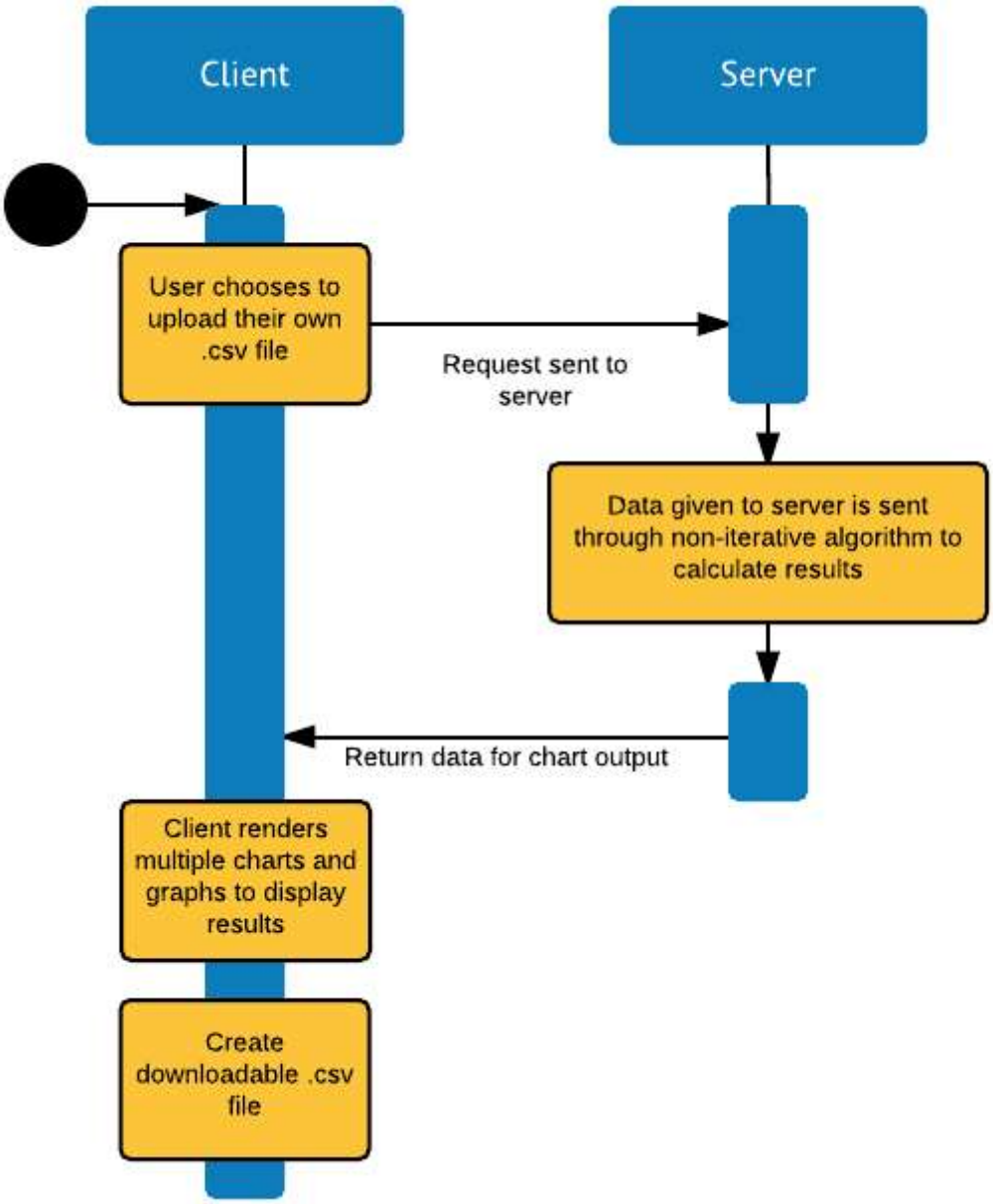
Sequence Diagram For Manual Input

In this case the user will directly input data using the web application's input options available. The resulting data will be sent to the server, which will receive the request for computation, and will send the data through the Transforming Drainage Algorithm. In this case there is no need to access the database, since all of the input information is already given. The result is then sent back to the client to be rendered into interactive charts and graphs which can be downloaded if the user chooses to do so.



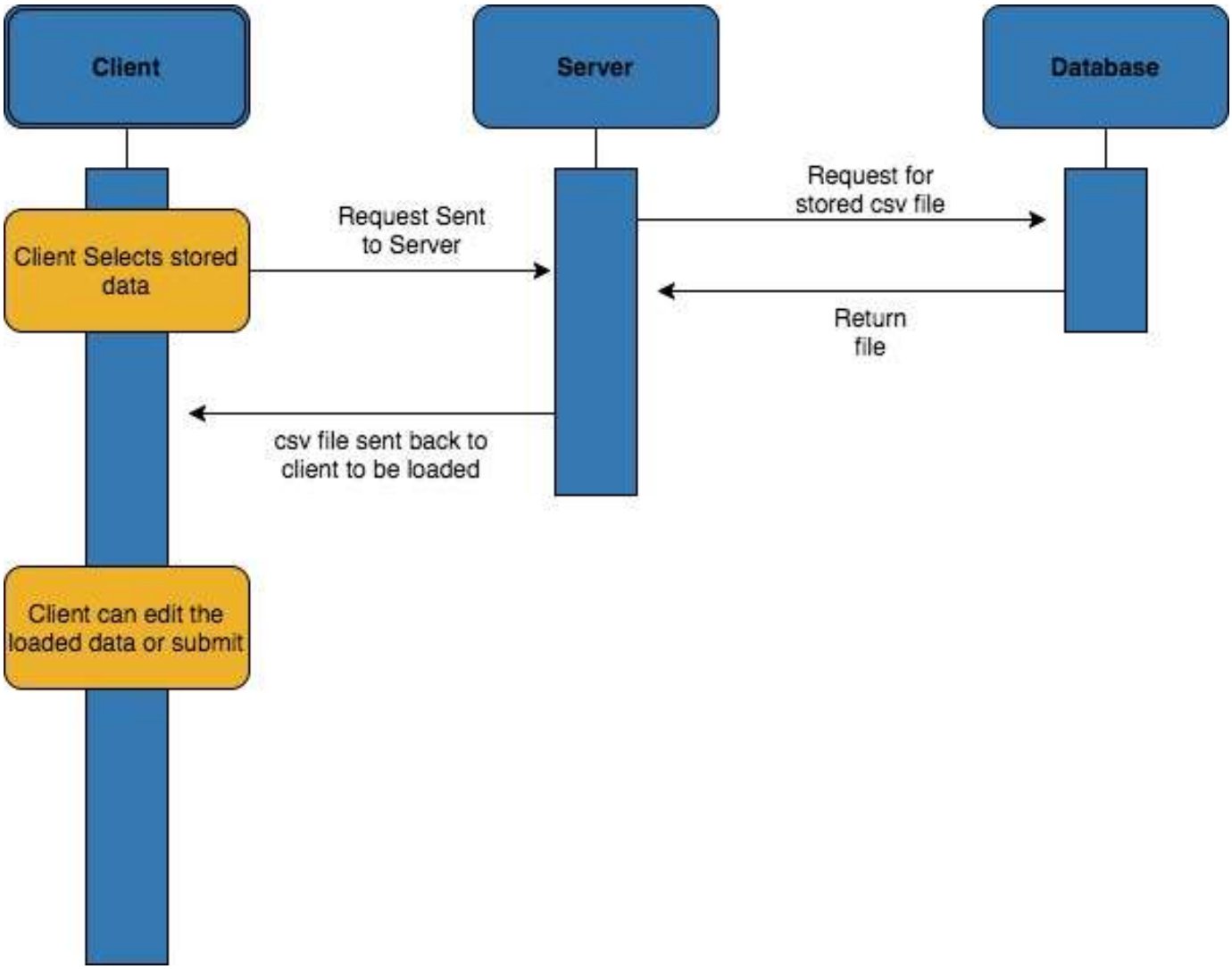
Sequence Diagram For User Uploaded .CSV File

In this case the user has opted to upload their own .csv file for calculation. The information is sent to the server which will receive the request for computation, and will send the data through the Transforming Drainage Algorithm. Just like in the manual input case, there is no need to access the database as it is assumed that the .csv file will contain the information needed for the calculations. The result is then sent back to the client to be rendered into interactive charts and graphs which can be downloaded if the user chooses to do so.



Sequence Diagram For User Selecting A .CSV File From The Database

A user has selected to use one of the .csv files that can be taken from the dropbox-esque .csv file database. The request is sent to the server, which then requests the stored .csv file from the database, which then returns the file to the user. At this point the client can either choose to edit the .csv file's data, or choose to use the .csv file as it is to be computed as data. The file's data will then go through the same process as the sequence diagram above as though the .csv was a user uploaded file.



UI Mockup

There will be another tab to choose from which will have the results ready to display on the map