# Software Design Document

for

# Augmented Reality for Hydrology (Version 2)

**Version 2.0 approved**

**Prepared by Mher Oganesyan, Leonardo Obay, Gilberto Placidon, Anthony Soto, Refugio Arroyo**

**Jet Propulsion Laboratory (JPL)**

**March 12, 2018**

## Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| First Draft | 11/18/2018 | Initial draft of document | 1.0 |
| Final Draft | 03/12/2019 | Final draft of document | 2.0 |
| | | | |
| | | | |

# 1. Introduction

## 1.1 Purpose

The purposes of this document are to:

1. Identify the requirements for Augmented Reality for Hydrology (Version 2) which will cover all aspects of the software; providing enough information for the reader to understand the purpose of the project, how the user will interact with the application, and how the application will interact with external services.
2. Describe concepts that will help clarify the operations for Augmented Reality for Hydrology (V2).
3. Provide requirements that are necessary for the completion and operation of the product.

## 1.2 Document Conventions

Typographical conventions used in this document include black text color, and font Times New Roman. The title, sections, and subsections use a bold text for emphasis; font sizes are 32, 20, and 14 respectively. Table of contents, and contents within subsections have a font size of 12. No other document conventions were used when creating this document. In addition, each requirement statement has its own priority.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for developers, testers, project managers, users, and the customer to understand the purpose, operation, and requirements of the project. The specific intentions for for the various readers and the suggested reading sequences are the following:

- Product Manager - Understand use and requirements of the product to help guide the development of said product; Recommended Reading: Section(s) 2, 4, 6
- Developers - Understand what is required of the product and guidance to the implementation of the requirements; Recommended Reading: Section(s) 3 - 9
- Testers - To understand the purpose and requirements of the product to guarantee proper functioning of the product; Recommended Reading: Section(s) 6, 7, 10
- Customer - Guarantee that the product meets the customers needs; Recommended Reading: Section(s) 2, 9

## 1.4 System Overview

Augmented Reality for Hydrology (Version 2) is an Android application. The software retrieves scientific data from Jet Propulsion Laboratory (JPL) Watertrek database. Watertrek is a database that stores hydrology data retrieved from sensors in rivers, mountains, wells, etc or it calculates predictions given a set time period.  With the implementation of Augmented Reality (AR), the software will provide a visualization of the various datasets provided by Watertrek.  AR objects, which are points of interest (POI) in the real world,  rendered in the scene will open a door to a wider array of data.  When a POI is selected, the program will then display a graph, a set of data related to the graph, and filters representing the selected POI's history data, allowing the user to filter time periods and different information for a given POI.  Aside from the POI, the software will provide a way for the user to point their device in a specific direction and retrieve the height of the land within that line of sight.  In the AR view the software will also generate a two and a half dimensional mesh that will mimic and blanket the surrounding land around the device.

# 2. Design Considerations

## 2.1 Assumptions and Dependencies

2.1.1 Android  Device is assumed to have Android API level 15 or higher.

2.1.2 Android Device is assumed to have a working camera and location sensor.

2.1.3 Users are expected to have credentials from JPL to access the database on their devices.

2.1.4 User is expected to have a stable internet connection to retrieve data from the database.

2.1.5 Developers assume all necessary API calls are provided by JPL ,and are operating, for their database.

2.1.6 Developers assume access to the OpenStreetMaps API calls are continuously operating.

## 2.2 General Constraints

2.2.1 Sensor Accuracy - Various devices have different quality sensors

2.2.2 Weak/Unstable Network Connectivity - May cause issues in retrieving data from database

2.2.3 Watertrek database - Database being down or REST calls that are needed are not provided

2.2.4 Device Performance - Need adequate performance from the device to render meshes and/or any other computer generated objects

2.2.5 Device Storage - Need for caching large amounts of data retrieved from database

2.2.6 Budget - Open source application program interfaces are the only option; paid services would help to improve performance

2.2.7 Resource availability - May slow down production time; may need to develop own software

2.2.8 Environment - Watertrek only collects data for the Western half of the United States; many states and other countries will not have functioning data.

2.2.9 Security - The application requires credentials to operate. This limits the user base of the application to the developers and JPL staff that have credentials for Watertrek.

## 2.3 Goals and Guidelines

2.3.1 The application shall have an offline mode in case a stable connection is  inaccessible

2.3.2  The application shall superimpose a mesh, that isn't obstructed, in given users line of sight

2.3.3 The application shall provide historical records of various data types

2.3.4  The application shall provide a graphical representation of historical data

2.3.5 The application shall display a billboard based on range provided by the user

2.3.6 The application shall  display a map that allows the user a eagle eye point of view of POI nearby

## 2.4 Development Methods

An Agile Development approach was taken while developing this product.  The process of the project is as follows:

- The liaisons from JPL provided us a set of high level requirements which lists out what they want the product to include.
    - We analyzed the requirements, along with the datasets provided to us,  and created new ones in more detail and determined use cases for the application.
    - Once approved by the customers, we begin development of the application, regularly keeping in touch to report the progress of the development or to ask them to provide us with a particular program or database call that is necessary to develop the application.
    - We test each component of the software independently.
    - Then we test the components as part of the whole system.
    - The liaisons provide us with any alterations or additions to the project, if there are any.
    - Repeat the process as needed.

# 3. Architectural Strategies

3.1 The programming language will be Java, as we are using Android Studio. The other option being Kotlin, as it is also supported by Android Studio. Java shall be used over Kotlin as it has more available support.

3.2 Preload of data, on application startup, for better user experience.

3.2.1 DEM for meshdata and map tiles for OSM take a couple seconds to load on their own. For better user experience, corresponding data shall be acquired on application startup.

3.2.2 DEM data downsampled for faster computation and less internal storage usage.

3.3 Storing of meshdata as csv file on users internal storage for future use.

3.4 Use of OpenGL ES 2.0 for higher user phone compatibility.

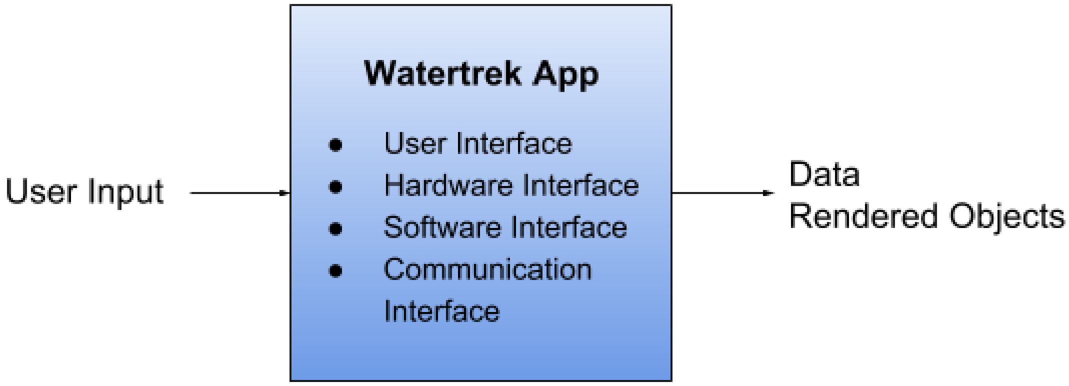
3.4.1 Although version 3.0 is available, with some added features, it wouldn't provide the most user phone compatibility.

3.5 Shall keep as much computational needs on server side to conserve battery life of phone.

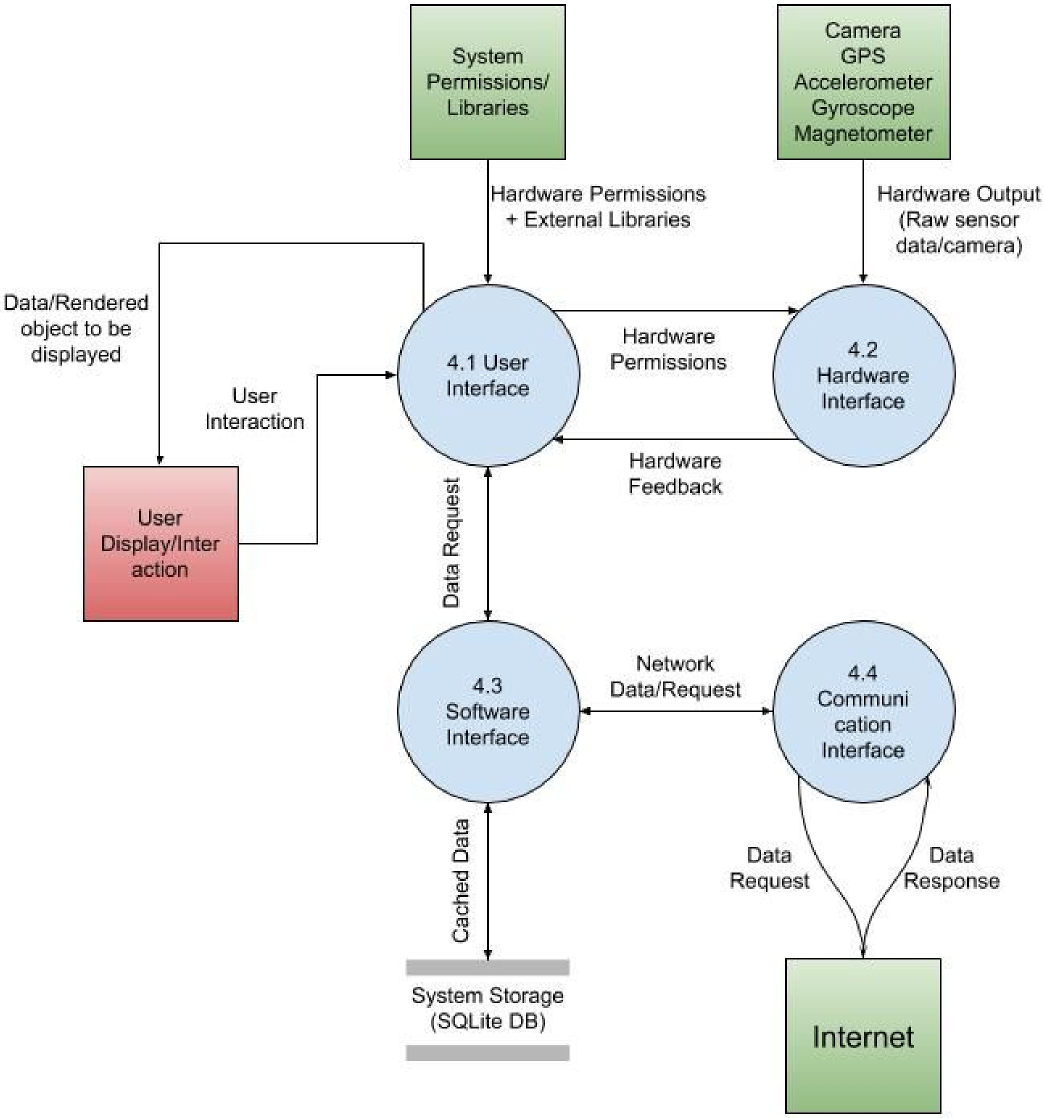
# 4. System Architecture

The architecture for the application is self contained and is split into four major modules.  Each module handles the various components needed for the operation of the program.  The overall application is based on user input and taking that input, a specified output will be displayed to the device with relevant Watertrek data.

The diagram below (DFD level 0) is a high level representation of how the application operates:



Below is a diagram (DFD level 1) that portrays how the different components interact and communicate with each other to have a single functioning application. The four modules are described in detail in section 6.  This is a brief overview of how they work together:


System Permissions/ Libraries
Camera GPS Accelerometer Gyroscope Magnetometer
Hardware Permissions + External Libraries
Hardware Output (Raw sensor data/camera)
Data/Rendered object to be displayed
4.1 User Interface
Hardware Permissions
4.2 Hardware Interface
User Interaction
Hardware Feedback
User Display/Inter action
Data Request
4.3 Software Interface
Network Data/Request
4.4 Communi cation Interface
Cached Data
Data Request
Data Response
System Storage (SQLite DB)
Internet

**4.1 User Interface (UI):** allows the user of the application to interact with and navigate the application.  It takes in user inputs and processes them using the necessary hardware readings and network calls to output the relevant data to the user.
**4.2 Hardware Interface:** continuously outputs sensor data.  That information is grabbed by the user interface whenever a user interacts with a UI element that needs sensor data to process network calls.
**4.3 Software Interface:** handles network calls by either grabbing cached data or requesting a network call from the Communication Interface and caching the data for later use when requested by the user.
**4.4 Communication Interface:** takes in network requests from the Software Interface, executes them, and retrieves the output from the internet to pass into the Software Interface for caching of data or instant use by the User Interface.

# 5. Policies and Tactics

## 5.1 Choice of which specific products used

5.1.1 Android Studio SDK

5.1.2 JVM(Java Virtual Machine)

5.1.3 OpenStreetMaps API

5.1.3.1 Google Maps was considered, but was not used because the API was not open source.

5.1.4 OpenGL ES 2.0

5.1.5 WaterTrek API

## 5.2 Plans for testing the software

5.2.1 Unit tests can ensure a function is responding accordingly as we can test the output of a function independently from the entirety of the code.

5.2.2 A third party site(www.whatismyelevation.com) was used to test whether or not the elevation based on line of sight was accurate relative to the user.

5.2.3 A third party app called Euler Compass was used on an Android device to verify the application's outputs for yaw, pitch, and roll.

## 5.3 Engineering Trade-Offs

5.3.1 To accommodate the use of API level 15, some more advanced features to assist in tracking surfaces in AR could not be used. Instead, mesh objects will be represented in an estimated manner.

5.3.2 In addition to a restriction to a lower API level, only open source software could be used in assisting with development of the application.

## 5.4 Coding Guidelines and Conventions

5.4.1 The coding convention of Object Oriented Programming(OOP) was followed.

## 5.5 The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

5.5.1 An algorithm for constructing mesh objects from a TIFF file was implemented based off previous work from a CSULA graduate student (Alvin Quach).

## 5.6 Plans for maintaining the software

5.6.1 Services such as the line of sight calls and OpenStreetMaps are based online and are assumed to continue functioning as some of the application's features depend on them for functionality.

5.6.2 JPL is expected to maintain their WaterTrek API.

5.6.3 JPL is expected to maintain their Line Of Sight API

## 5.7 Interfaces for End-Users, Software, Hardware, and Communications

5.7.1 The end-user interacts with the software through a GUI that contains a menu for switching on/off which hydrology components are rendered.

## 5.8 Hierarchical organization of the source code into its physical components (files and directories)

5.8.1 The underlying augmented reality functions are located within its own folder separated from higher level components of the application.

- 5.8.1.1 Manifests directory
- 5.8.1.2 Java components
    - 5.8.1.1.1 edu.calstatela.jplone.arframework Package
        - 5.8.1.1.1.1 Graphics3d
        - 5.8.1.1.1.2 Landmark
        - 5.8.1.1.2 Sensor
- 5.8.1.3 Resource components
    - 5.8.1.3.1 Drawable
    - 5.8.1.3.2 Values

5.8.2 The Main Application activities were located

5.8.2.1 Manifests folder

5.8.2.2 Java components

5.8.2.2.1 edu.calstatela.jplone.watertrekapp Package

5.8.2.2.1.1 Activities

5.8.2.2.1.2 Adapters

5.8.2.2.1.3 Billboardview

5.8.2.2.1.4 Data

5.8.2.2.1.5 DataService

5.8.2.2.1.6 Helpers

5.8.2.2.1.7 NetworkUtils

5.8.2.3 Resource components

5.8.2.3.1 Anim

5.8.2.3.2 Drawable

5.8.2.3.3 Layout

5.8.2.3.4 Menu

5.8.2.3.5 Mipmap

5.8.2.3.6 Values

# 6. Detailed System Design

## 6.1 User Interface (Module)

### 6.1.1 Responsibilities

User interface module is comprised of several classes that help navigate through the application to validate user and filter through different data types and be able to view more detailed information of said POI in a particular range in time.

### 6.1.2 Constraints

Hardware Interface contains constraints regarding to the particular device that the user is using and is limited by the sensors capabilities of the users device. Device needs to have a properly working camera built in.

### 6.1.3 Composition

User interface component provides parameters that are passed onto various network calls used to retrieve POI data.

### 6.1.4 Uses/Interactions

Users will be able to access data from JPL servers to view various data such as height, longitude latitude and changes of time in certain data fields.

### 6.1.5 Resources

JPL servers

### 6.1.6 Interface/Exports

## 6.2 Hardware Interface (Module)

### 6.2.1 Responsibilities

Hardware Interface module allows us to retrieve certain angles using the devices sensors that are later used in implementation of line of sight.

### 6.2.2 Constraints

Hardware Interface contains constraints regarding to the particular device that the user is using and is limited by the sensors capabilities of the users device. Device needs to have a properly working camera built in.

### 6.2.3 Composition

Hardware interface component provides angles obtain through various sensors such as accelerometer, gyrometer and magnetometer that are later used in implementation of line of sight using the devices built in camera.

### 6.2.4 Uses/Interactions

Developers and users will be able to see the current roll and pitch angle of the device in realtime and be able to use this data to help determine if there's a obstruction view in a particular direction.

### 6.2.5 Resources

JPL servers

### 6.2.6 Interface/Exports

## 6.3 Software Interface

### 6.3.1 Responsibilities

The Software interface allows us to retrieve different values of data from various POI in our  app as well as allows us to load a interactive map based  image based on users location and stores values temporary in the cache.

### 6.3.2 Constraints

Software Interface contains constraints such as JPL's servers being down and OpenStreetMaps servers being down or being inaccessible.

### 6.3.3 Composition

Software Interface provides POI data and  Open Street Maps information in real time.

### 6.3.4 Uses/Interactions

The use will be able to get real time data  of various POI and be able to see there current location in Open Street Maps with POI near them.

### 6.3.5 Resources

JPL servers, Open Street Maps

### 6.3.6 Interface/Exports

## 6.4 Communication Interface

### 6.4.1 Responsibilities

Provides access to JPL’s database which allows us to retrieve various types of data.

### 6.4.2 Constraints

Communication with JPL’s servers  might be inaccessible.

### 6.4.3 Composition

Various HTTP GET request that allow us to  retrieve POI data such as Wells, Reservoirs, Snotels and height map data as well.

### 6.4.4 Uses/Interactions

Updated information in real time according to JPL’s database  depending on JPLs Sensors.

### 6.4.5 Resources

JPL servers

### 6.4.6 Interface/Exports

# 7. Detailed Lower level Component Design

7.1 ReservoirService.java

7.1.1 Class

7.1.2 Used to gather primary and secondary data from JPL's Watertrek resources. Only provides data for Reservoirs.

7.1.3 No interface

7.2 SnotelService.java

7.2.1 Class

7.2.2 Used to gather primary and secondary data from JPL's Watertrek resources. Only provides data for Snotel.

7.2.3 No interface

7.3 WellService.java

7.3.1 Class

7.3.2 Used to gather primary and secondary data from JPL's Watertrek resources. Only provides data for Wells.

7.3.3 No interface

7.3.4 Uses provided latitude and longitude to generate a polyline(circle). Also compensates for the variances in meridian and parallels along the poles and equator.

7.4 SoilMoistureService.java

7.4.1 Class

7.4.2 Used to gather primary and secondary data from JPL's Watertrek resources. Only provides data for Soil Moisture.

7.4.3 No interface

7.5 Reservoir.java

7.5.1 Class

7.5.2 Custom Java object used to store Reservoir type data. Contains fields and attributes specific to Reservoir type.

7.5.3 No interface

7.6 Snotel.java

7.6.1 Class

7.6.2 Custom Java object used to store Snotel type data. Contains fields and attributes specific to Snotel type.

7.6.3 No interface

7.7 Well.java

7.7.1 Class

7.7.2 Custom Java object used to store Well type data. Contains fields and attributes specific to Well type

7.7.3 No interface

7.8 SoilMoisture.java

7.8.1 Class

7.8.2 Custom Java object used to store Soil Moisture type data.

7.8.3 No interface

7.9 River.java

7.9.1 Class

7.9.2 Custom Java object used to store River type data.

7.9.3 No interface

7.10 DatabaseHelper.java

7.10.1 Class

7.10.2 Used to operate database, create/delete tables, insert/extract data from database.

7.10.3 No interface

7.10.4 Transfer data from hydrology data classes to local database.

7.11 Network Task

7.11.1 Class

7.11.2 Custom class used to delegate asynchronous tasks and perform http requests with Watertrek resources.

7.11.3 No interface

7.12 Credentials Activity

7.12.1 Class

7.12.2 Used to help store and verify that correct username and passwords were used

7.12.3 No interface

7.13 DatePickerFragment

7.13.1 Class

7.13.2 Used to create the alert Dialog to let use select a month, day and year

7.13.3 No Interferface

7.14 History Activity

7.14.1 Class

7.14.2 Used to display selected dates that user has selected and a list of values for that selected date as well as a graphical representation of that given information

7.14.3 no interface

7.15 Elevation Obstruction Service

7.15.1 Class

7.15.2 Used to help determine if there is an obstruction within the viewers field of view and to get a pints elevation values.

7.15.3 no interface

7.16 Main Activity

7.16.1 Class

7.16.2 Used to obtain roll and pitch angle of Device

7.16.3 Used to Navigate in which POI the user wants to be displayed

7.16.4 Used to display POI based on device current location

7.17 Splash Activity

7.17.1 Class

7.17.2 Used to check for credentials, permissions, load DEM data

7.17.3 Loading screen of application

7.17.4 Uses latitude and longitude of device to gather DEM data from database

7.18 MeshService.java

- 7.18.1 Class
- 7.18.2 Used to read mesh file and generate necessary information
- 7.18.3 No interface

7.19 Azimuth_RecyclerViewAdapter

- 7.19.1 Class
- 7.19.2 Used to create compass for azimuth angle
- 7.19.3 Display as compass on Main Activity

7.20 Pitch_RecyclerViewAdapter

- 7.20.1 Class
- 7.20.2 Used to create compass for pitch angle
- 7.20.3 Display as compass on Main Activity

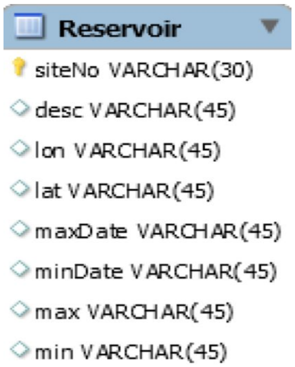
7.21 NetworkTaskJson Authentication

- 7.21.1 Class
- 7.21.2 Used to process JSON data that needs Authentication to be retrieved
- 7.21.3 No Interface
- 7.21.4 Uses credentials to communicate with database to retrieve JSON

# 8. Database Design

This app has one database that includes four tables, but could be extended to more.

Table Reservoir:

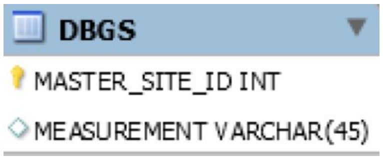


Reservoir

siteNo VARCHAR(30)
desc VARCHAR(45)
lon VARCHAR(45)
lat VARCHAR(45)
maxDate VARCHAR(45)
minDate VARCHAR(45)
max VARCHAR(45)
min VARCHAR(45)

Table reservoir has elements that represent a Reservoir object in hydrology.

Table DBGS:

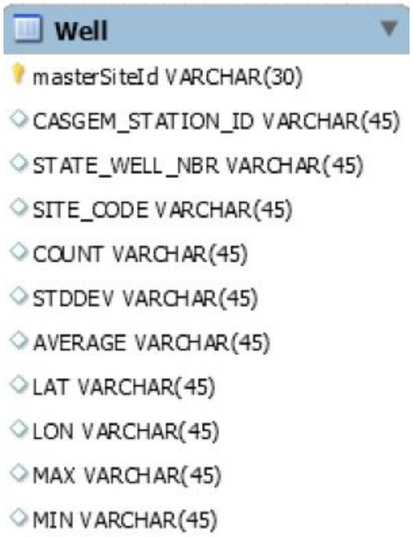


DBGS

MASTER_SITE_ID INT
MEASUREMENT VARCHAR(45)

Table DBGS is a supplemental table of Table Well that store a list of measurement data of a single well.

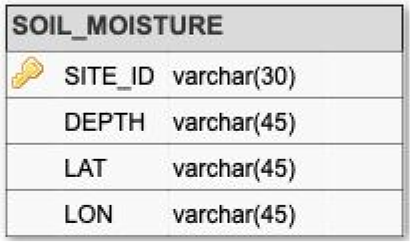
Table Well:



| | Well | |
|---|---|---|
| 🔑 | masterSiteId | VARCHAR(30) |
| ◇ | CASGEM_STATION_ID | VARCHAR(45) |
| ◇ | STATE_WELL_NBR | VARCHAR(45) |
| ◇ | SITE_CODE | VARCHAR(45) |
| ◇ | COUNT | VARCHAR(45) |
| ◇ | STDDEV | VARCHAR(45) |
| ◇ | AVERAGE | VARCHAR(45) |
| ◇ | LAT | VARCHAR(45) |
| ◇ | LON | VARCHAR(45) |
| ◇ | MAX | VARCHAR(45) |
| ◇ | MIN | VARCHAR(45) |

Table Well has elements that represent a well object in hydrology.

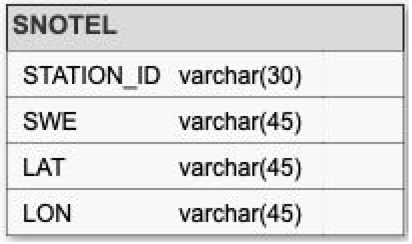
Soil Moisture:



| SOIL_MOISTURE | | |
|---|---|---|
| 🔑 | SITE_ID | varchar(30) |
| | DEPTH | varchar(45) |
| | LAT | varchar(45) |
| | LON | varchar(45) |

Table Soil Moisture has elements that represent a soil moisture object in hydrology.

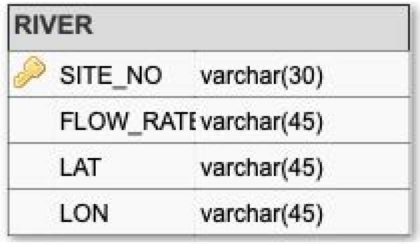
Snotel:



| SNOTEL | |
|---|---|
| STATION_ID | varchar(30) |
| SWE | varchar(45) |
| LAT | varchar(45) |
| LON | varchar(45) |

Table Snotel has elements that represent represent a snotel object in hydrology.

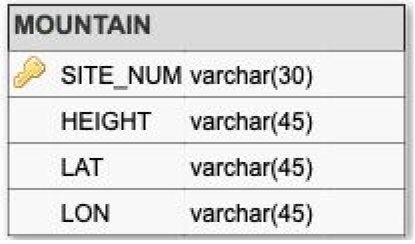
River:



| RIVER | | |
|---|---|---|
| 🔑 | SITE_NO | varchar(30) |
| | FLOW_RATE | varchar(45) |
| | LAT | varchar(45) |
| | LON | varchar(45) |

Table River has elements that represent represent a river object in hydrology.

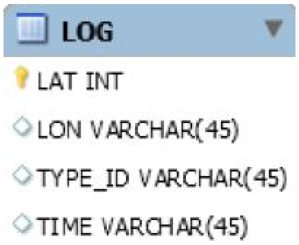
Mountain:



| MOUNTAIN | | |
|---|---|---|
| 🔑 | SITE_NUM | varchar(30) |
| | HEIGHT | varchar(45) |
| | LAT | varchar(45) |
| | LON | varchar(45) |

Table Mountain has elements that represent a mountain object in general (not directly related to hydrological data).

Table Log:



LOG ▼

🔑 LAT INT

◇ LON VARCHAR(45)

◇ TYPE_ID VARCHAR(45)

◇ TIME VARCHAR(45)

Table Log records when and where the application make a rest-api call. User can use the record as reference to decide whether a new api call is going to fire or not.

# 9. User Interface

## 9.1 Overview of User Interface

The main objective of the user interface (UI) is to present the user with a sense of realism through augmented reality. This is portrayed by drawing interactive three-dimensional objects onto the device's camera. The objects, called billboards, represent points of interests (PoIs) such as mountains, rivers, reservoirs, wells, and soil moisture. The billboards are placed on the main segment of the PoIs. The user will be able to click on billboards to view descriptions of the PoIs.

Upon launching the application, the user will be presented with the welcome screen, a scenic view of mountains and river. An icon, the AR Droplet, will fade into the middle of the scene giving the application time to set up its UI. If it is the first time the user launches the application, permissions will be prompted in the launch screen.

After the application loads, the login page for the Watertrek credentials will be prompted, allowing the user to login to access its data.

When the camera view loads, a gear on the top left corner of the screen will represent the page to the settings. In addition, there will be an icon on the screen that is used to display the various hydrological data types; a crosshair in the center of the screen will be use to associate where the device is pointed.

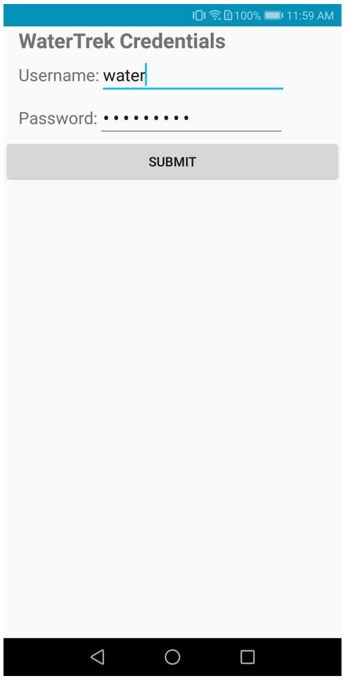
When clicking the gear, the user will be prompted with the settings page. There will be an option to login or log out of the Watertrek database. In addition, the user can set the range for the POI's in the view, mesh shading toggle, and camera toggle.

The second icon on the main screen when selected, will display the different data types that the user can turn on or off, the superimposing of the mesh, and the line of sight call.

When a PoI is selected, a page will be displayed containing a graphical representation of the data for the selected PoI. There will be filters for the start date, end date, and the specific dataset to be displayed for the particular data type. When swiping from the bottom of the screen in an upward motion, the data will be revealed in a list (scrollable) view.

## 9.2 Screen Frameworks or Images

Loading Screen

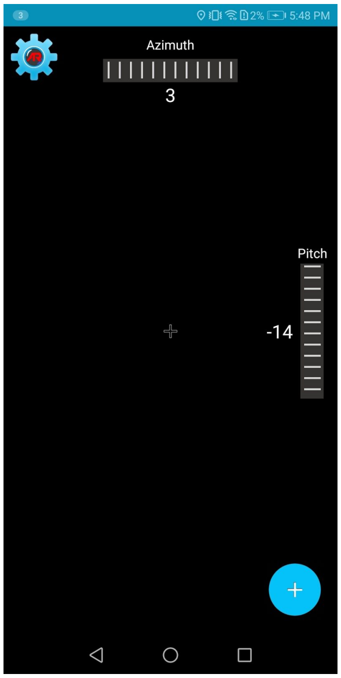


WaterTrek Credentials

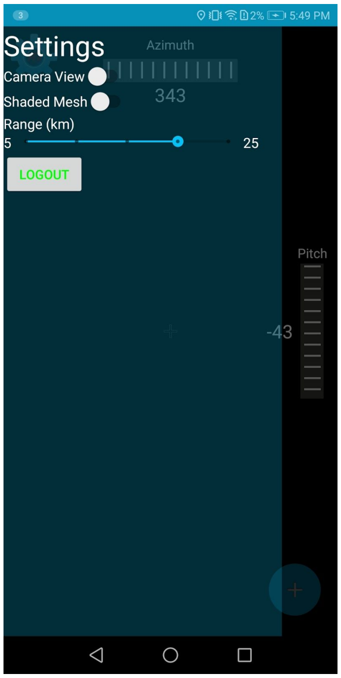
Username: water

Password: •••••••••

Credentials/Login



Camera View



Settings

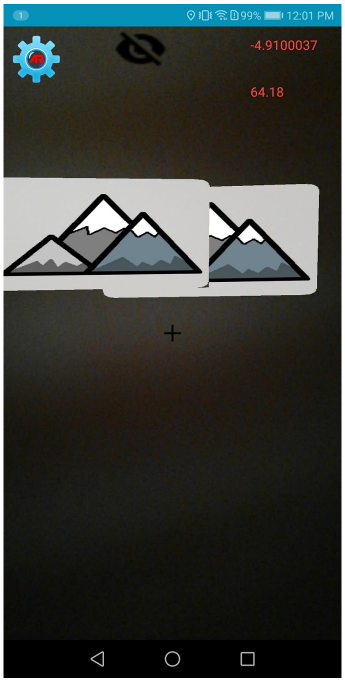
Camera View ●

Shaded Mesh ●

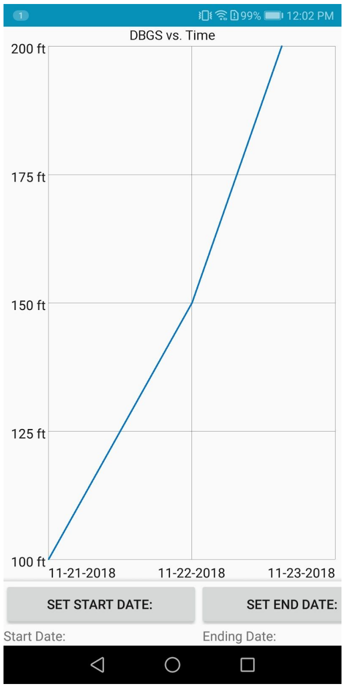
Range (km)

5 25

Preferences



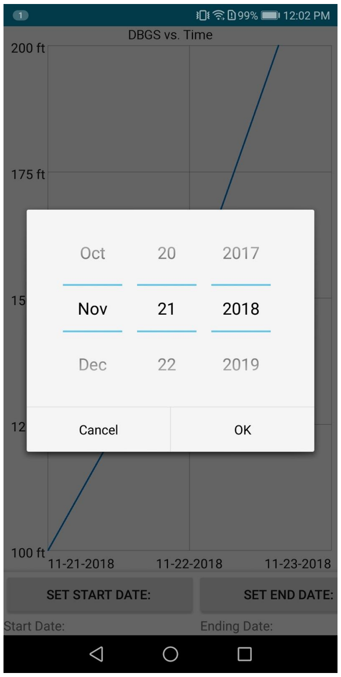
Billboards



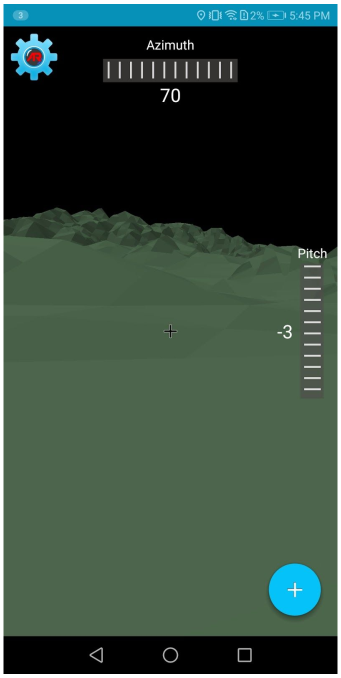
Start Date:

Ending Date:

Graph View of Data

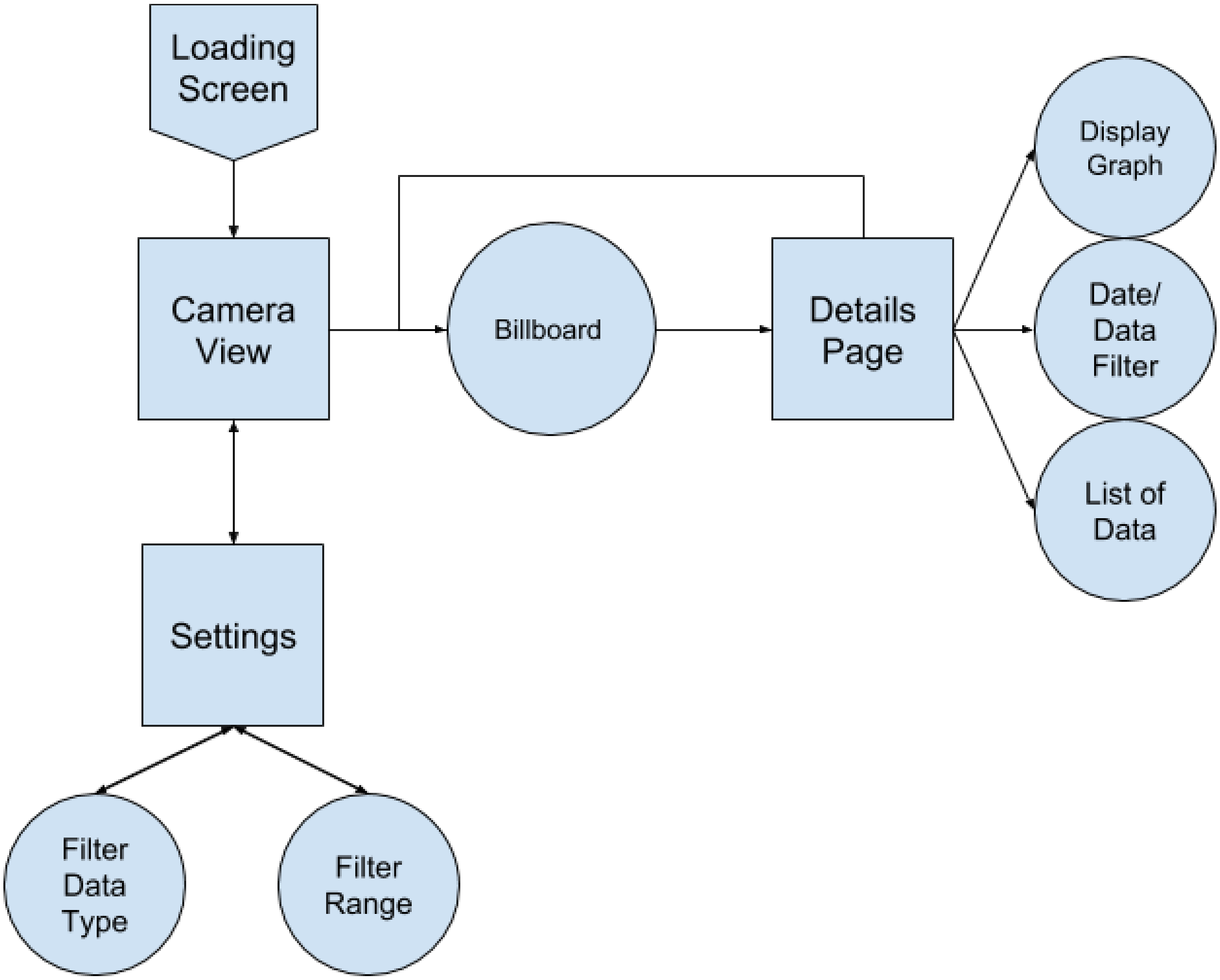


Date Filter



Mesh Example

## 9.3 User Interface Flow Model


Loading Screen
Camera View
Billboard
Details Page
Display Graph
Date/ Data Filter
List of Data
Settings
Filter Data Type
Filter Range

# 10. Requirements Validation and Verification

| Requirement | Component/Module | Test |
|---|---|---|
| 4.1.1. Application Requirements | --- | |
| 4.1.1.1 The application shall run on an Android device | | |
| 4.1.1.2 The application shall process permissions upon first launch. | SplashActivity Class | Manual Testing |
| 4.1.1.2.1 The application shall prompt for location permissions | SplashActivity Class | Manual Testing |
| 4.1.1.2.2 The application shall prompt for camera permissions | SplashActivity Class | Manual Testing |
| 4.1.1.2.3 The application shall prompt for storage permissions | SplashActivity Class | Manual Testing |
| 4.1.1.3 The application shall provide an interface for credentials | CredentialsActivity Class | Manual Testing |
| 4.1.1.3.1 The application shall disable functions that need the credentials if the credentials are incorrect | MainActivity Class | Manual Testing |
| 4.1.1.4 The application shall provide an interface to select data types | MainActivity Class | Manual Testing |
| 4.1.1.4.1 The application shall provide river data | River and RiverService Class | Observation of Code and Manual Testing |
| 4.1.1.4.1.1 The application shall provide river flow rate information | River and RiverService Class | Observation of Code and Manual Testing |

| | | |
|---|---|---|
| 4.1.1.4.2 The application shall provide soil data | SoilMoisture and SoilMoistureService Class | Observation of Code and Manual Testing |
| 4.1.1.4.3 The application shall provide reservoir data | Reservoir and ReservoirService Class | Observation of Code and Manual Testing |
| 4.1.1.4.4 The application shall provide well data | Well and WellService Class | Observation of Code and Manual Testing |
| 4.1.1.4.4.1 The application shall provide depth below ground surface information | Well and WellService Class | Observation of Code and Manual Testing |
| 4.1.1.4.4.2 The application shall provide WSEL information | Well and WellService Class | Observation of Code and Manual Testing |
| 4.1.1.4.4.3 The application shall provide water table depth information | Well and WellService Class | Observation of Code and Manual Testing |
| 4.1.1.4.5 The application shall provide snotel data | Snotel and SnotelService Class | Observation of Code and Manual Testing |
| 4.1.1.5 The application shall provide an interface to filter the range for the data | MainActivity Class | Manual Testing |
| 4.1.1.6 The application shall provide a billboard for the data types | MainActivity Class | Manual Testing |
| 4.1.1.6.1 The application shall allow the user to select the billboard | MainActivity Class | Manual Testing |
| 4.1.1.7 The application shall provide a separate activity for the selected billboard | HistoryActivity Class | Manual Testing |
| 4.1.1.7.1 The application shall provide a graph to represent time range data | HistoryActivity Class | Manual Testing |
| 4.1.1.7.1.1 The graph shall represent mean respect to a range of time | HistoryActivity Class | Manual Testing |

| | | |
|---|---|---|
| 4.1.1.7.1.2 The graph shall represent standard deviation respect to a range of time | HistoryActivity Class | Manual Testing |
| 4.1.1.7.2 The application shall provide a tool to filter dates | DatePickerFragment Class | Manual Testing |
| 4.1.1.7.3 The application shall provide a tool to filter data information for the data types | HistoryActivity Class | Manual Testing |
| 4.1.1.7.4 The application shall provide a panel to view history data | HistoryActivity Class | Manual Testing |
| 4.1.1.8 The application shall superimpose mesh objects onto terrain | MeshDemoActivity Class | Manual Testing |
| 4.1.1.9 The application shall provide an implementation of OpenStreetMaps. | DetailsActivity Class | Manual Testing |
| 4.1.2. Mesh Function | --- | |
| 4.1.2.1 The mesh function shall generate a mesh from the provided DEM data. | MeshDemoActivity Class | Manual Testing |
| 4.1.2.1.1 The mesh function shall retrieve a Tiff file type | MeshDemoActivity Class | Observing Code and Manual Testing |
| 4.1.2.1.2 The mesh function shall process the Tiff file | MeshDemoActivity Class | Observing Code and Manual Testing |
| 4.1.2.1.3 The mesh function shall subsample the data from the Tiff file | MeshDemoActivity Class | Observing Code and Manual Testing |
| 4.1.2.1.4 The mesh function shall send the processed data to OpenGL to be rendered | MeshDemoActivity Class | Observing Code and Manual Testing |
| 4.1.3 Line of Sight Function | --- | |

| | | |
|---|---|---|
| 4.1.3.1 The line of sight function shall retrieve the devices roll | MainActivity Class | Observing Code and Manual Testing |
| 4.1.3.2 The line of sight function shall retrieve the devices pitch | MainActivity Class | Observing Code and Manual Testing |
| 4.1.3.3 The line of sight function shall retrieve the devices yaw | MainActivity Class | Observing Code and Manual Testing |
| 4.1.3.4 The line of sight function shall retrieve the devices coordinates | MainActivity Class | Observing Code and Manual Testing |
| 4.1.3.5 The line of sight function shall retrieve where the device is point at provided the previous parameters | ElevationObsrtructionService class | Observing Code and Manual Testing |
| 4.1.3.6 The line of sight function shall display the elevation of land that it’s pointing to | MainActivity Class | Observing Code and Manual Testing |

# 11. Glossary

**Application Program Interface (API):** Functions or methods for accessing software services or libraries.

**Augmented Reality (AR):** A technique in computer graphics that superimposes (places) a computer generated object into a devices camera view to alter the perception of the real world.

**Depth Below Ground Surface (DBGS):** Data provided by JPL's Watetrek database that is part of Well data. The *depth to groundwater* is the measured vertical distance to water in a well from a defined reference point. The DBGS is the depth to groundwater minus the distance from the reference point to the ground surface. Often the reference point is the top edge of the well casing, which is commonly above the ground surface. The DBGS can be a negative number if the water level in a well casing is above the ground surface.

**Digital Elevation Map (DEM):** Data provided by JPL's Watertrek database, in a tiff format, that represents the elevation of each latitude and longitude.

**Framework:** One of the software deliverables from Augmented Reality for Hydrology (Version 1) that provides components to assist in the development of the AR objects in the application.

**Hydrology:** A study of the properties of earth's water, focusing on the movement of the water in relevance to the land.

**JVM (Java Virtual Machine):** A virtual machine that converts java code into bytecode that the system can understand and execute.

**OpenGLES 2.0:** A 2D/3D graphics API used for rendering the augmented reality scene.

**Point of Interest (POI):** A place in the real world that is represented by a computer generated object in the AR view of the program.

**REST API:** An API practice that uses HTTP or HTTPS request to GET, PUT, POST, and DELETE data from a database.

**Snotel:** Sensors in the hills that measure the weight of snow pillows to estimate the depth of the snow.

**Watertrek:** A database provided by JPL that stores Hydrology data and is accessible using a REST API.

**GUI:** Graphical user interface

**HTTP:** Hypertext Transfer Protocol

**UI:** User Interface

# 12. References

Title: Software Design Document for Augmented Reality for Hydrology
Authors: Wilbert Veit, Christopher Hung Nguyen, Ernesto Padilla, Cuong Pham, Kaichen Zhou
Date: September 5, 2017

https://csns.calstatela.edu/department/cs/project/resource/view?projectId=6059915&resourceId=6566869

Android Sliding Up Panel

https://github.com/umano/AndroidSlidingUpPanel

Graphview

http://www.android-graphview.org/

OpenStreetMaps

http://osmdroid.github.io/osmdroid/