UM–SJTU Joint Institute

App Development for Entrepreneurs
(Ve441)

Final Report

Aug 7, 2020

Name: Zhu, Haoxuan	Student ID: 516370910157
Name: Li, Chenhao	Student ID: 516370910143
Name: Shi, Yiming	Student ID: 516370910108
Name: Yuan, Yongwei	Student ID: 516370910098
Name: Zhou, Jiaxi	Student ID: 516370910110

Contents

1 Introduction

The team has developed an mobile application called "Salinity Map." This report illustrates considerations of different aspects when designing the product. The ultimate goal is to develop a mobile application that will show water salinity information on the map. The potential customers include not only researchers whose focus is on climate change but also those residents who live in poor countries and suffer from unsafe water source. Researchers may take advantage of our map to gain insight from the data while residents could find the safest water source around them. The detailed implementation will cover functions for different groups of customers.

The report will describe the progress of project from five dimensions, including features and user story, interface design, initial software design, software architecture and implementation considerations.

2 Features and User Story

The ultimate goal of mobile applications is to benefit the stakeholders with features that fulfill their requirements. Hence, finalizing the entrepreneurial idea from a requirement engineering perspective guards the consistency between products and users' requests. Our mobile application is inspired by requests from coastal residents who suffer from high water-salinity problems. Drinking water with high salinity damages the wellness of people. However, the local lack tools to track the water quality nearby. On the other hand, ocean researchers require a platform to store and process data. Another group of stakeholders are government and non government organizations (NGO) who concerns climate change and public health. Providing salinity data together with other weather and climate index help government officials conduct urban planning. Although the business problem has been identified, requirements elicitation exists. Requirement engineering is applied to extract essential and concrete features required by stakeholders.

To derive features and user stories that shape our application, we decompose the requirement into sets of features categorized by different groups of stakeholders. "Connextra Notation" and "SMART principle" are applied to extract fine-grained features, and eventually user stories.

2.1 General Requirement and Generic Stakeholders

The general requirement is to make updated salinity data available to users. As a vague statement, features can be extracted by splitting the general requirements based on different categories of stakeholders. For our application, there are only two main categories of people: local residents and researchers. With surveys and online poll, we gathered many complex features, including but not limited to

- Local residents:

1. As a local resident, I can see the water salinity level near me.
2. As a local resident, I can tell which locations are best to take water
3. As a local resident, I can compare water salinity in locations that I chose
4. As a local resident, I can zoom in to certain region for detailed information on water salinity
5. As a local resident, I can get route to the nearest safe water source
6. As a local resident, I can get route to any water source that I choose

- Researchers:

1. As a researcher, I can input new water salinity data I gathered
2. As a researcher, I can zoom in/out to see water salinity data in different scopes
3. As a researcher, I can delete wrong salinity data
4. As a researcher, I can modify inaccurate salinity data

- Governments and NGOs:

1. As a government/NGO official, I can see real-time weather info on the map
2. As a government/NGO official, I can see real-time wind index on the map
3. As a government/NGO official, I can see current status on the map
4. As a government/NGO official, I can see prediction of future weather status

Here, we process the interviewees' opinions by splitting them into concrete features, each involves at most one action. In this way, essential functions of the application are highlighted.

2.2 Applying "Connextra Notation" and "SMART principles"

"Connextra Notation" is a method to rewrite features with regard to the user group, goal and task. SMART principle further highlights the importance of features being specific, measurable, achievable, time-boxed and relevant. In this way. Both methods contribute to translating abstract requirements into practical and appreciable features. Following is an example of rewriting complex features based on "Connextra Notation" and "SMART principles".

The mechanism of "Connextra Notation" is to make request-feature pairs. providing user cases to support the necessity of certain features. "Connextra Notation" restricts generic ideas to deliver practical features. With "Connextra Notation", the complex features can be re-expressed as

> As a local resident (role) so that I can find a safest water source with normal salinity, I want to search water salinity rates of all water sources near me.

As a researcher (role) so that I can track water salinity in a certain region, I want to observe data visualized on the map.

Then, we apply SMART principle to enhance features:

- **Specific**: Since "normal salinity" is a vague description of water, the phrase should be rewritten as local residents can view the the water salinity data from all nearby water sources and verify that the salinity level of water from the specified source is around the average of those in all nearby water sources.
- **Measurable**: Adding numbers and constraints to the goal sets up a testable quantity. In this example, changing "near me" to "within "10 kilometers from my current location" quantized the abstract feature and set a standard for the application.
- **Achievable**: Some features are easier said than done. In the design phase, non-achievable features should be identified, and be either downgraded or excluded. In above features, always getting "the latest water salinity level in all places" are impossible. Hence, it can be downgraded to "updating water salinity level once a day", as water salinity level is relatively stable in a short amount of time. Reducing the update frequency does not hurt or miss user requirements.
- **Relative**: Each feature should serve a purpose. Besides fulfill portions of user requirements, developers should also consider the underlying business values. The sixth features mentioned in the local residents section is a supplement to the general requirement. The efforts and cost to develop this feature is not profitable. Therefore, according to this principle, developing the comment-review feature should have the lowest priority compared with other features.
- **Time-boxed**: Every user requirement has a time span. Therefore, to take a great share in certain business market, applications need to be delivered as fast as possible while maintaining a high quality. Hence, during design phases, developers need to perform time-aware engineering, ensuring that the major features can be realized first. This principle adds time as a factor to our evaluation.

  Since "achievable" and "time-boxed" can be represented by the rest three principles, during design phases, developers should focus on processing features to be specific, measurable and relative. In this way, features can be ranked based on necessity and cost-efficiency.

For salinity map, the major focus is on data visualization. Therefore, after applying mentioned techniques, features are ranked based on whether adding them help user read and report water salinity level in locations of their interest.

2.3 User story

User stories put applications into all kinds of potential scenarios to refine the extracted features based on their importance. Immersing into different circumstances assist developers in designing the hierarchy of the application and provide references for future phases of

projects, including use case diagrams, I/O interfaces, and user-friendly UI prototypes.

Connecting features with scenario guarantees feature rationale, and accurate estimate of progress. It also helps stakeholders understand how a single feature is used and why it is essential to be developed. A detailed scenario usually emphasizes on condition, event, action, and consequence. For example,

- **Feature** read water salinity data by tapping dots on the map
  - **As a** local resident
  - **So that** I can compare different water sources to choose the safest place to collect water
  - **I want to** know water salinity of different locations that I choose
  - **Given** that I open the application
  - **Then** the application will locate my position, show the nearby water sources with colored dots, whose color indicates the water salinity level
  - **Then** I can know places with green dots on the map indicate they are ideal water sources with healthy water salinity rate
  - **When** I tap a dot on the map
  - **Then** a window pops up and I can see detailed data of the corresponding location

2.4 Interview Schedule and Results

User-oriented design is burned into the backbone of our app. We adapt the Agile development methodology to guarantee that user requirements can be accurately extracted and corresponding features can be developed accordingly to fulfill users' requests. During the developing process,customers are always included to our prototype evaluation and iterations.

Time	**Interviewee**	**Prototype**	**Takeaways**
6/5/2020	Mentor	Design	1. Set basic requirements for a map app 2. Tailor features for different stakeholders
6/30/2020	Online Volunteer	Demo 1	1. Use color for salinity value instead of size 2. Add nearest data point navigation
7/26/2020	Data Scientists	Demo 2	1. Add authentication for data credibility 2. Easy gesture to get data
7/30/2020	Mentor	MVP	1. Add real-time weather related information 2. Interface and UI suggestions

Table 1: Interview Schedule and Feedback

2.5 Features and user stories for salinity map

Initial high-level requirements is raised by stakeholders. General requirements are gathered via online surveys. After performing the above procedure, the developing team select the following features to implement, together with user stories to convince stakeholders.

2.5.1 Basic Information System

- **Feature** Show user's current location
  - **As a** general user
  - **So that** I can locate myself
  - **I want to** see my location on the map app
  - **Given** that I open the application
  - **Then** the application will locate my position, zoom in to my region, show the recorded water sources near me on the map
  - **Then** I can know my current location and water sources nearby
- **Feature** Show points on the map to indicate locations that have salinity data
  - **As a** local resident
  - **So that** I can know where to get water
  - **I want to** know the latitude and longitude of all tested water sources near me
  - **Given** that I open the application
  - **Then** the application will locate my position, zoom in to my region, show the recorded water sources near me on the map
  - **Then** I can know the location of the nearest water sources to get water
  - **As a** local researcher
  - **So that** I can examine all water sources of my interest
  - **I want to** know which places haven't been tested
  - **Given** that I open the application
  - **Then** the application will locate my position, zoom in to my region, and use dots to show where already have data
  - **Then** I can tell which water sources have not been examined, and go there for investigation and research
- **Feature** Apply heat map to denote whether salinity is high or low
  - **As a** local resident
  - **So that** I can ensure that the water I take is not over-salty

  - **I want to** know how the water quality is
  - **Given** that I open the application
  - **Then** the application will show different colors for water sources to indicate salinity level
  - **When** I see a red dot on the map
  - **Then** I know the salinity level at that place is too high
  - **When** I see a green dot on the map
  - **Then** I know the water salinity level is within the standard at that place

  - **As a** local researcher
  - **So that** I can have a broad understanding of the salinity distribution and trend
  - **I want** the data to be intuitive and vivid
  - **Given** that I open the application
  - **Then** the application will use colored dots to indicate salinity level of all the water sources
  - **Then** I can tell which regions have higher water salinity and which ones don't to verify my assumptions of current movement

- **Feature** show detailed water information on the map after tapping a data point
  - **As a** local resident
  - **So that** I can know the specific readings and the test date
  - **I want to** have detailed information when necessary
  - **Given** that I open the application
  - **Then** there are colored dots on the map with different opcaity
  - **When** I tap a data point on the map
  - **Then** the dot will pop up a window, listing all the detailed information, including salinity level data, last modified time, water capability, price, etc

- **Feature** show real-time weather and current information on the map
  - **As a** researcher/government official
  - **So that** I can conduct urban planning and policy making according to the environment
  - **I want to** have a real-time weather, wind, and current report
  - **Given** that I open the application
  - **Then** the app will show real-time wind, current and ocean salinity info on the map
  - **When** I zoom in

  - **Then** detailed reading will appear on the map

- **Feature** show predictions on future weather and salinity information on the map
  - **As a** researcher/government official
  - **So that** I can make precaution and emergent policies to overcome foreseeable high-salinity outbreak
  - **I want to** know the trend of weather and salinity
  - **Given** that I open the application
  - **Then** the app will show weather prediction result visually on the map

2.5.2 Navigation System

- **Feature** show routes to the chosen water source
  - **As a** local resident and/or researcher
  - **So that** I can go to a water station that I choose
  - **I want to** know the direction/route to the chosen water source
  - **Given** that I open the application
  - **Then** the application will zoom to user's current region and show the nearest water sources
  - **When** I tap and hold a point on the map
  - **Then** the application will set my current location as starting point, set the chosen point as destination, and show the optimized route on the map
  - **When** I drive/walk along the route
  - **Then** the map will update the route accordingly

- **Feature** show routes to the nearest water source with acceptable salinity level
  - **As a** local resident
  - **So that** I can go to the water source without examining all water sources
  - **I want to** view the direction to the nearest water source that passes salinity test
  - **Given** that I open the application
  - **Then** the application will zoom to user's current region and show the nearest water sources
  - **When** I tap and hold a point on the map
  - **Then** the application will set the point as starting point, locate the nearest water source that has drinkable water whose salinity level meets the standard, and show the direction on the map, and a window with options
  - **When** I choose another place
  - **Then** the map will re-decide the optimistic routine that meets the restricts

2.5.3 User Account System

- **Feature** Account sign up
  - **As a** researcher
  - **So that** I can report and upload my newly measured salinity level results
  - **I want to** have an account
  - **Given** that I open the application
  - **Then** the app will show water sources near me
  - **When** I double click a data point
  - **Then** a new page will be rendered, asking users to sign in/up.
  - **When** I type sign up
  - **Then** a form will pop out asking the user to enter username, password, institution, and other information
  - **When** I fill out the form and tape sign up
  - **Then** the app will check account validation and allow the user to modify data
- **Feature** Account sign in
  - **As a** researcher
  - **So that** I can report and upload my newly measured salinity level results
  - **I want to** sign in with my account
  - **Given** that I open the application
  - **Then** the app will show water sources near me
  - **When** I double click a data point
  - **Then** a new page will be rendered, asking users to sign in/up.
  - **When** I fill out the form for the sign in process
  - **Then** the app will verify account info and allow the user to modify data

2.5.4 Data Managing System

- **Feature** Add new data to database
  - **As a** researcher
  - **So that** I can report and upload my newly measured salinity level results
  - **I want to** have a form to add new data
  - **Given** that I open the application
  - **Then** the application will show a plus button on the bottom right

  - **When** I click the plus button
  - **Then** a form will pop out and ask the user to fill in the blanks.
  - **When** I fill in the form and submit it
  - **Then** the uploaded new data will be added to database, and a dot will appear on the provided location.

- **Feature** Modify salinity data
  - **As a** researcher
  - **So that** I can report and upload my newly measured salinity level results
  - **I want to** have a form to add new data
  - **Given** that I open the application and sign in
  - **Then** the application will show a plus button on the bottom right
  - **When** I double click a data point
  - **Then** a pop-up window will be rendered and ask the user to input new salinity data
  - **When** I fill in the form and submit it
  - **Then** the uploaded new data will be added to database, and the data dot will be updated simultaneously.

3 Interface Design

3.1 For Local Residents

3.1.1 Water Salinity Map

The most basic feature in the project is a water source salinity map. Anyone who care about the salinity of water source at some point should be able to get an idea from the map clearly. The following shows a sample layer which expresses the data with different color.

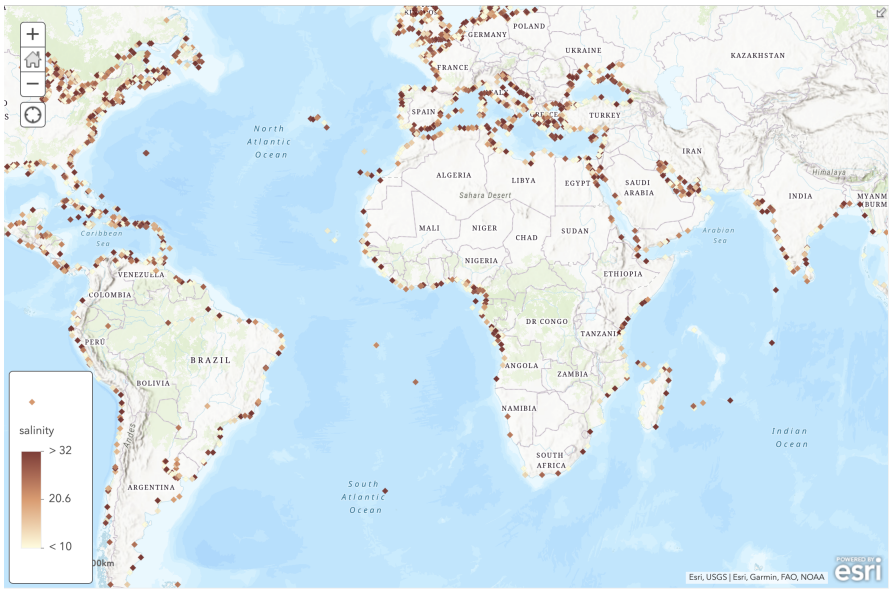


Figure 1: Water salinity map with data points of different color

The data points come from a open-source sample data which contains the information of ports in the world. The data is added with one more feature which assumes we know the salinity of water at each port. With the data points of different color, people can have a direct image whether the salinity of drinking water around them is at a healthy level.

Another way to express the value of water salinity is through the bigness of certain point. The sample interface is as following.

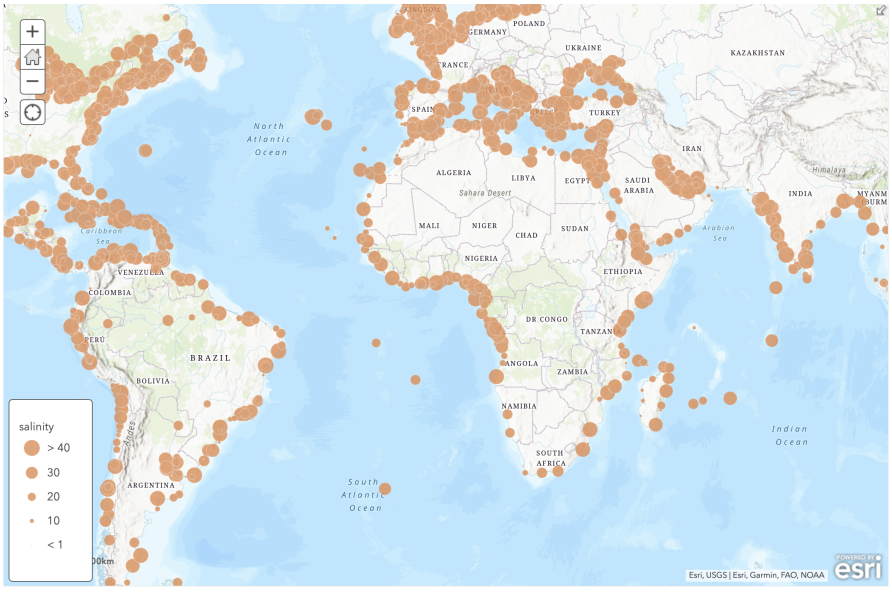


Figure 2: Water salinity map with data points expressed by bigness

After discussion with mentor and potential customers, the water salinity map in minimal

viable prototype expresses the salinity with different colors. The map is also combined with other real-time information such as prediction of precipitation in the following day, wind information including direction and speed, and information of ocean salinity and current. The back-end base map stored in ArcGIS online is as following.

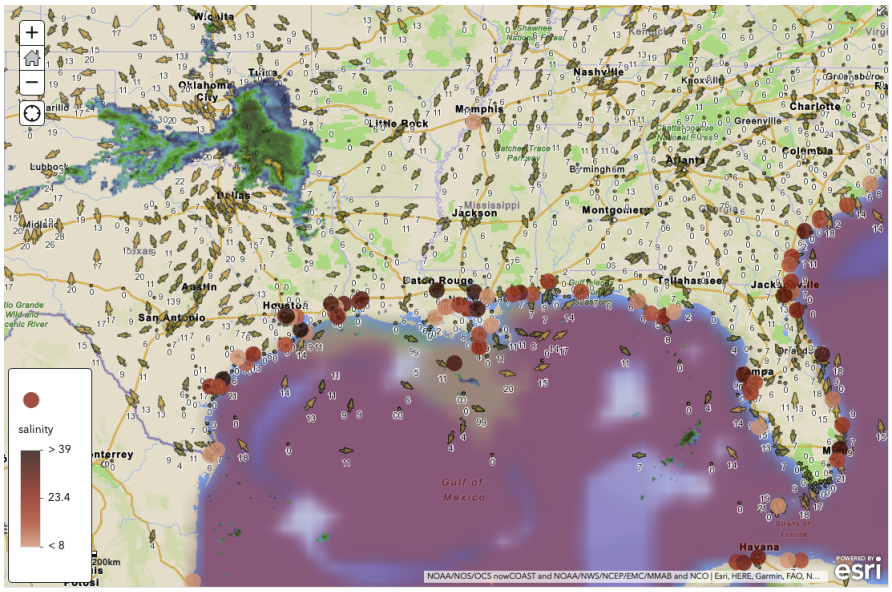


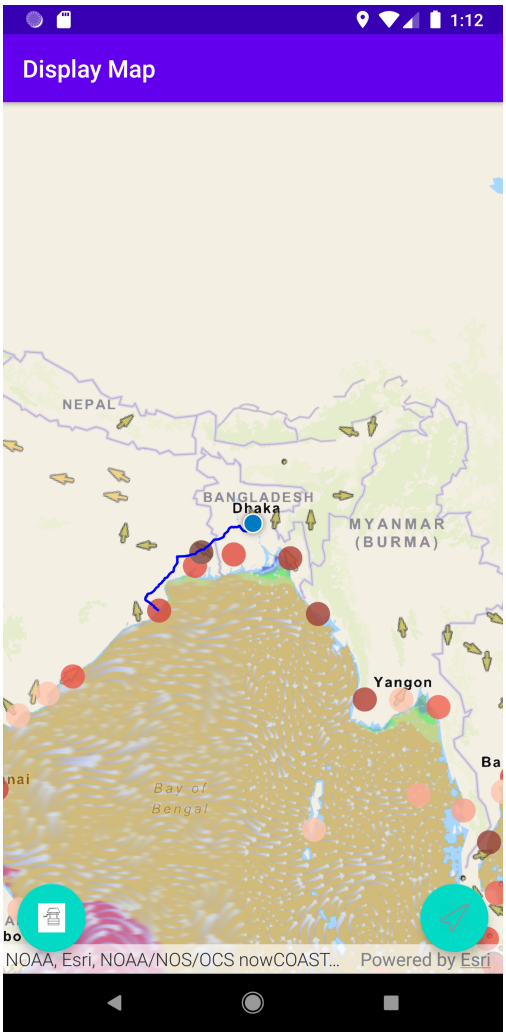
Figure 3: MVP water salinity map

3.1.2 Routing

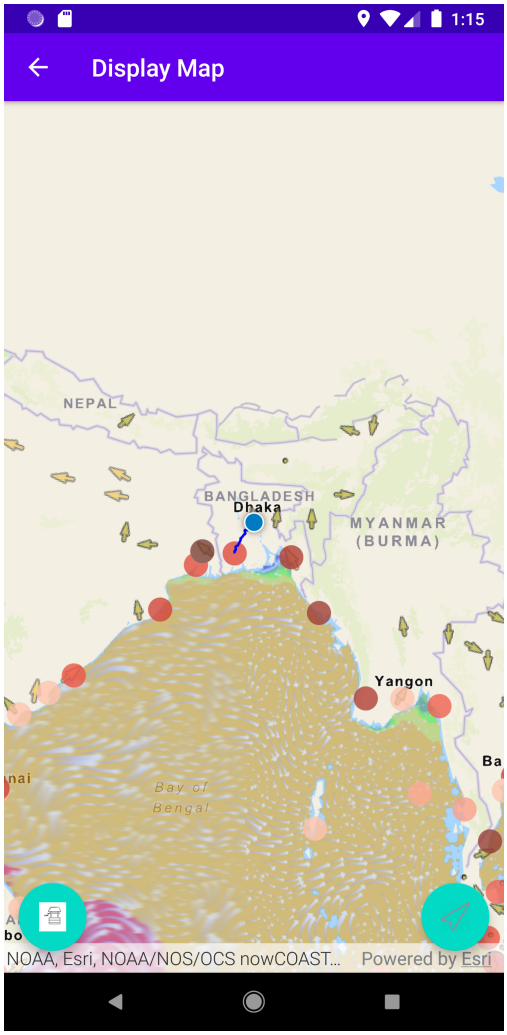
Besides showing the salinity of water source like well, we allow our users to navigate to specific water source by double tapping on a point(Figure 4a).
In the MVP prototype, the function of navigation is updated. Since local residents may only care about drinking safe water, we have a "well" button on the bottom left corner as a shortcut for residents to route to the safest water source within a certain range of their current location(Figure 4b). The app would provide the navigation to the data point with the lowest salinity value among a fixed range. On the mobile end, the app will query to the back-end database to get samples within a range from the current location. Based on the query results, the app will choose the data point with the lowest salinity value and sends it to ArcGIS routing service. Then there will be a route graphic rendered on the base map.

3.1.3 Show Detailed Info in Pop-up Window

Local residents could see where the selected well is located and the water salinity level statistically as shown in Figure 5. The style of the pop-up window is rudimentary at this stage though, which is something we can improve.



(a) Routing to selected well



(b) Adaptive routing based on water salinity and distance to water sources

Figure 4: Routing to safest

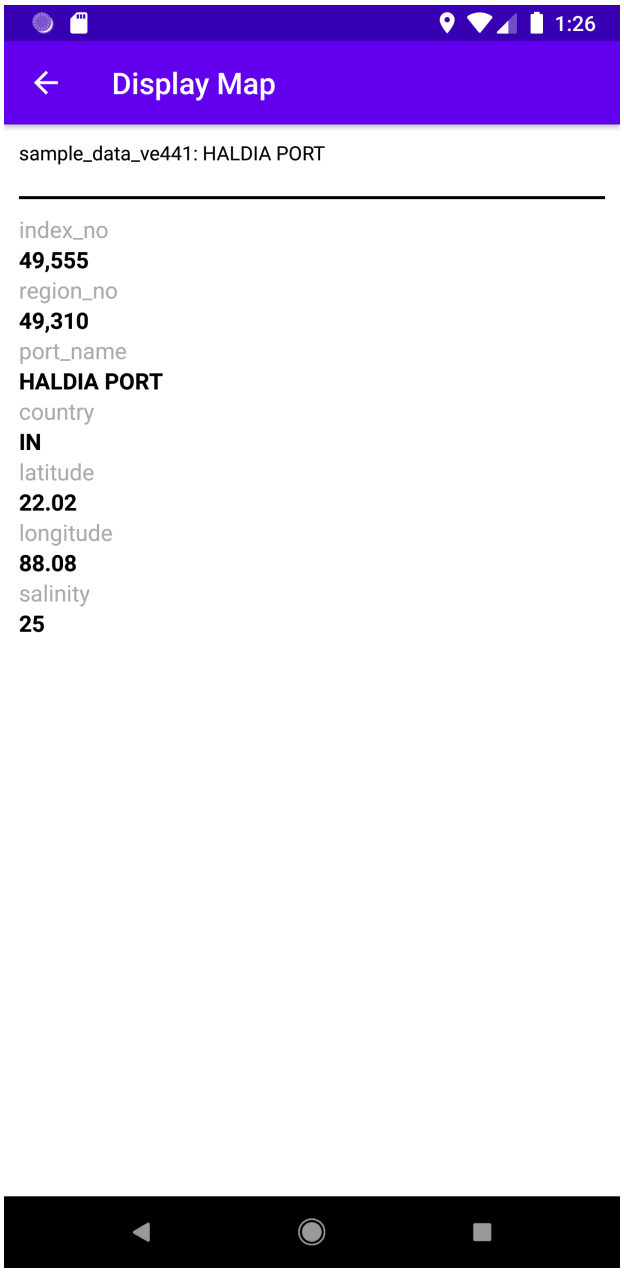


Figure 5: Detailed Status of Well

3.2 For Researchers

3.2.1 Show Trend of Water Salinity

Different from ordinary people, researchers might be more interested in the meaning of the data behind the scene. In our MVP, we only show the trend of water salinity in the same pop-up window in Figure 5 when some well is selected in a line chart.

3.2.2 Update Water Salinity Data

By long pressing on the well, researchers could easily update the water salinity map in our app and the database would sync with the updated status of wells. Figure 6 show the pop-up window where researcher could enter the updated water salinity.

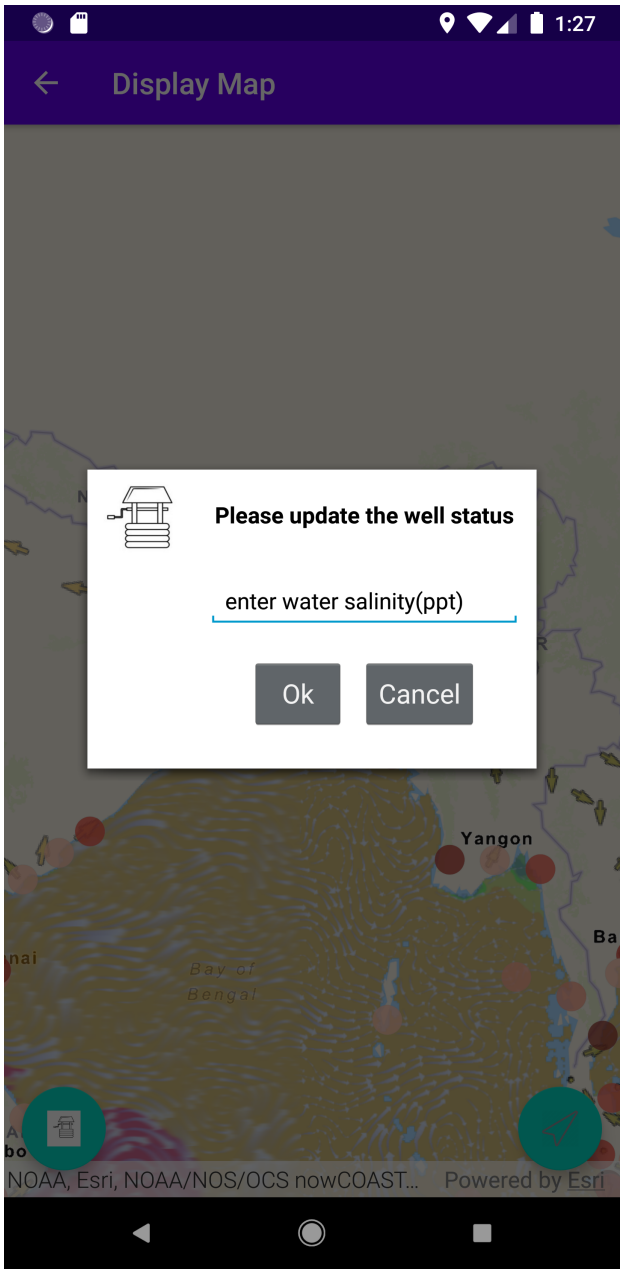


Figure 6: Update Well Status

But for the first time they try to update the data, we will pop a login window to verify their credentials so that the data won't be messed up with by random users. Figure 7 show the pop-up window where we verify the identity of researchers.

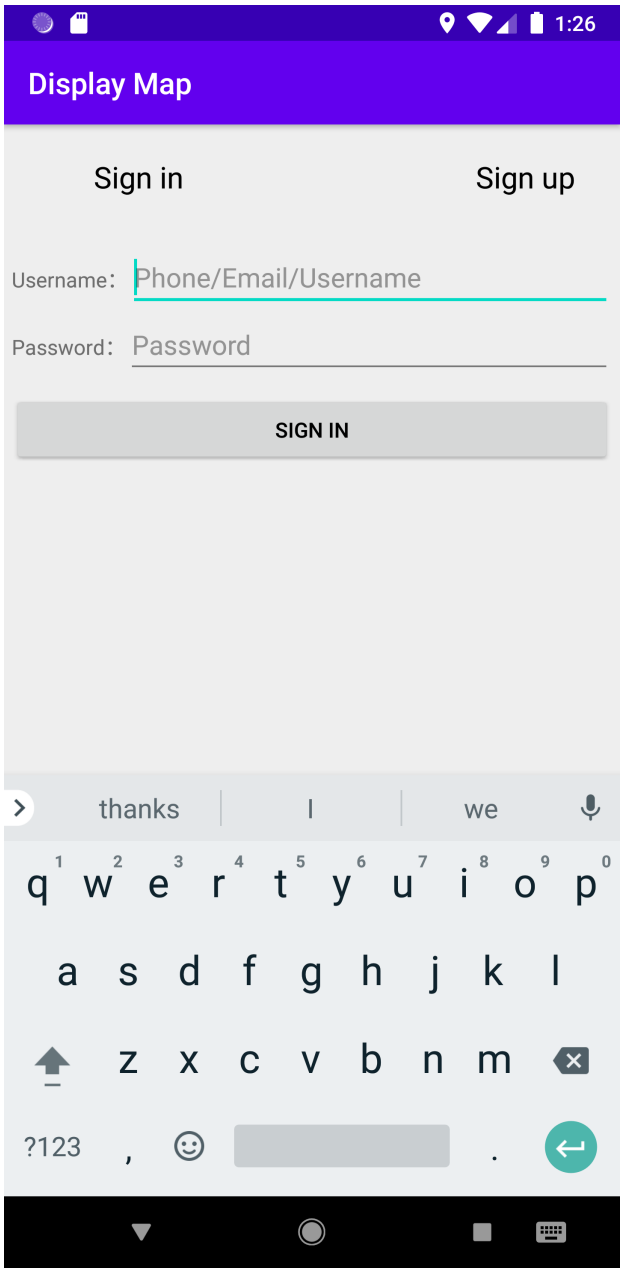


Figure 7: Login interface

4 Initial Software Design

After analyzing all the requirements from potential users, we create an initial software design for the water salinity map application based on the selected key features and user stories in section 2.4. The following UML sequence diagrams vividly illustrate the software design for each user story. They help show the object interactions and message exchanges involved in each scenario.

4.1 Basic Information System

- When a user wants to see the salinity data of nearby water sources, he/she needs to open the application and allow it to access his/her location via mobile phone. Once the app is opened, the user's location will be sent to ArcGIS. Then, ArcGIS will retrieve the salinity data of water sources that are close to this location from its database, and generate a map with data points on it. This map and data will be sent back to the Android platform and displayed on the screen. Thus, users will get access to the information of salinity data nearby after opening the mobile app.

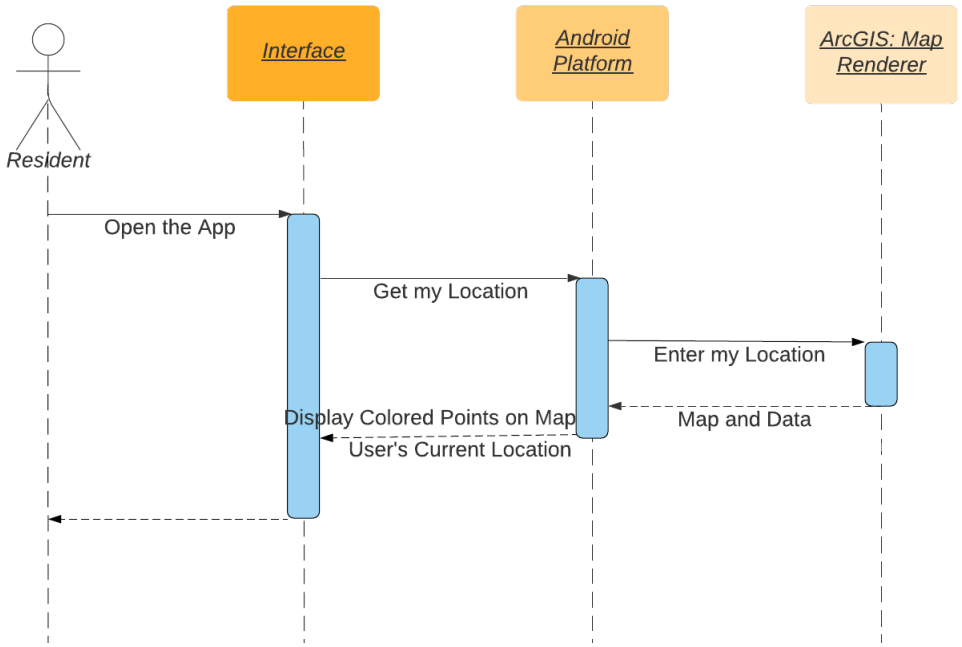


Figure 8: Sequence diagram of showing data points around users' location

- When a user wants an intuitive visual to show the salinity concentration of nearby water sources, he/she needs to open the application and look at the heat map of data points. The points will automatically appear in dark or light colors according to their values of salinity. The points with lower salinity concentration are in lighter colors while those with higher salinity concentration (not passing the standard for drinkable water) becomes darker. The colored points on the map will be sent back to the Android platform and displayed on the screen. This visual helps users to know about the water sources nearby and choose the best one. This feature shares the same logic and sequence diagram with the previous one.

- When a user wants to know detailed information of a certain water source, he/she needs to open the app and tap on a point of water source. After the user taps on a certain point, the ArcGIS will get the location of this point and retrieve its detailed information in the online database, then send the information back to the pop-up system in Android platform. A window with detailed information of this point will

pop up on the screen. Users will get more detailed information after tapping on a specific point on the map.

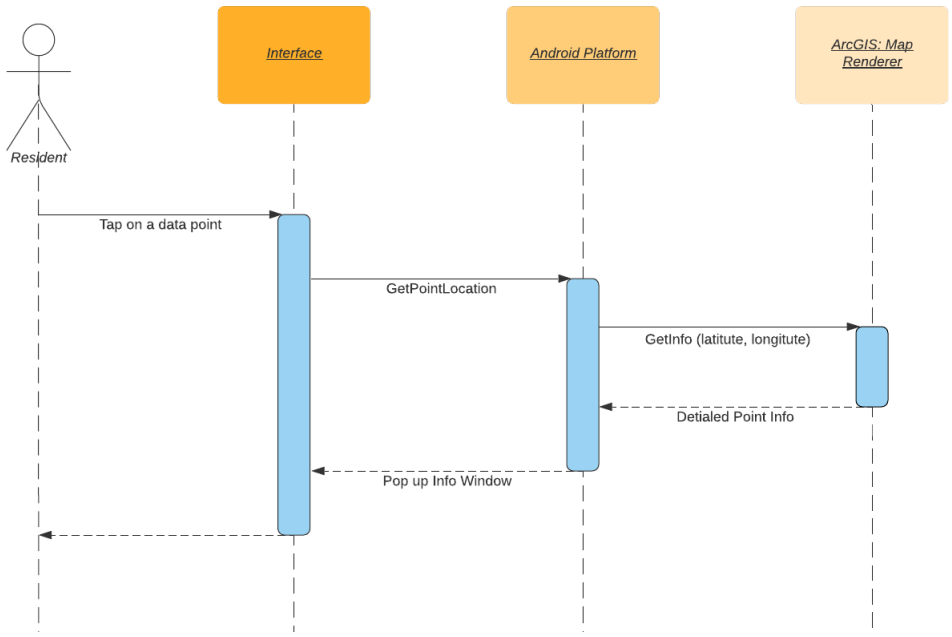


Figure 9: Sequence diagram of showing detailed information in a pop-up window

- When a user, especially a researcher or a government official, wants to know about real-time information of weather and current, he/she needs to open the application. Then, ArcGIS will retrieve the weather and current info from its database, and generate a map with arrows on it. This map and data will be sent back to the Android platform and displayed on the screen.

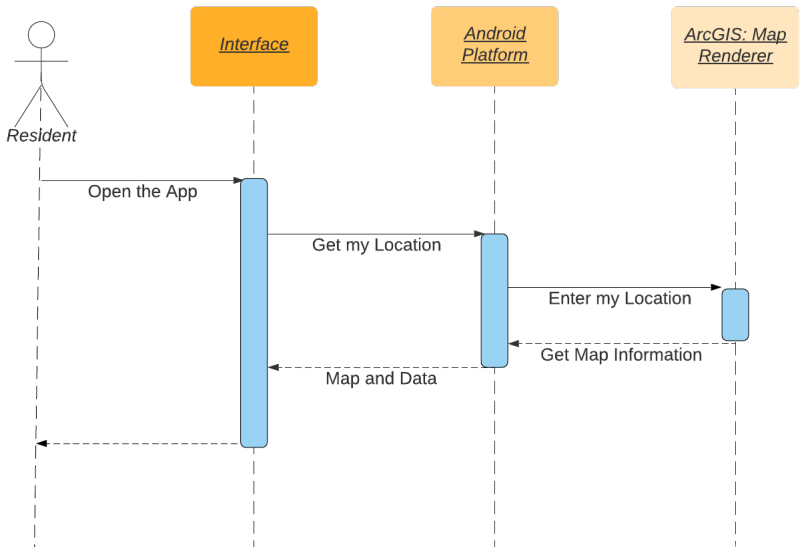


Figure 10: Sequence diagram of displaying real-time weather info

4.2 Navigation System

- When a user wants to know the optimal route to a certain point of water source from the current location, he/she needs to open the app and tap and hold on this point. This point will be set as the destination with the user's current location as the starting location in the ArcGIS calculator service system. Then the calculator service will find the optimal route and send the route back to the map render service. All the information of a map with routes and points on it will be sent to the Android platform. Finally, the routes will be displayed on the map. Users will get the routes to the chosen water source in this way.

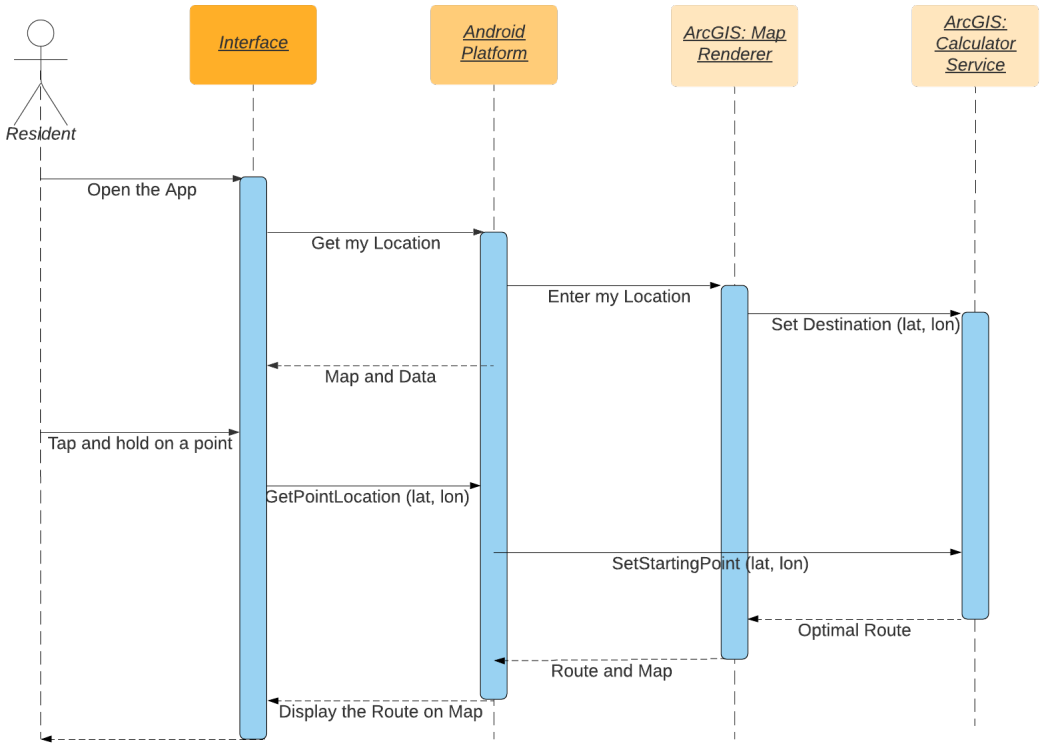


Figure 11: Sequence diagram of routing to a chosen water source

- When a user wants to know routes to the nearest water source with acceptable salinity level from a certain point location, he/she needs to open the app and click on the bottom left button on the screen. The user's current location will be set as the starting location in the ArcGIS calculator service system, and the ArcGIS calculator service will find the nearest available point of water source. Then the calculator service will find the optimal routes and send the routes to the map render service. Finally, the routes will be displayed on the map. Users will get the routes to the nearest acceptable water source in this way.

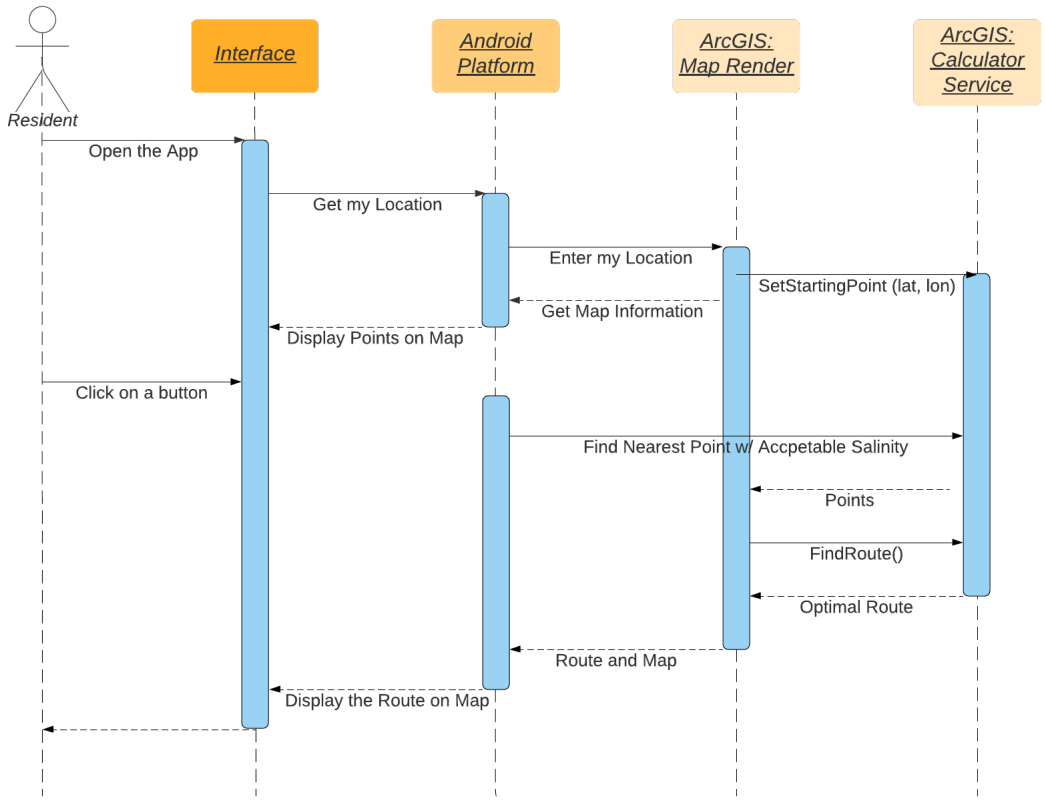


Figure 12: Sequence diagram of routing to the nearest safe water source

4.3 User Account System

- When a researcher wants to sign up for an account to create or edit his/her own database, he/she needs to input his/her personal information in a pop-up window for sign up. The account info will be stored in the online database.

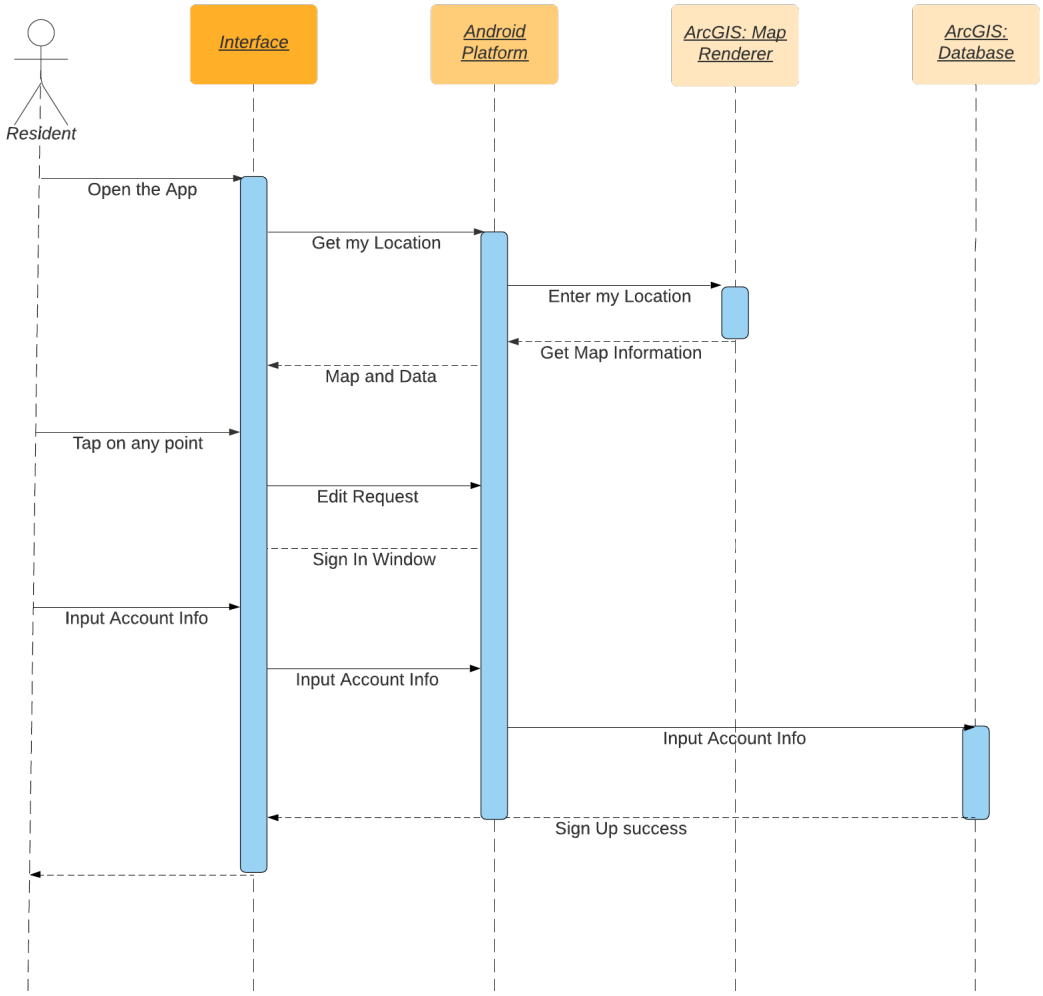


Figure 13: Sequence diagram of user sign up

- When a researcher wants to sign in for an account to see more information or make further changes, he/she needs to input his/her account information in a pop-up window to sign in.

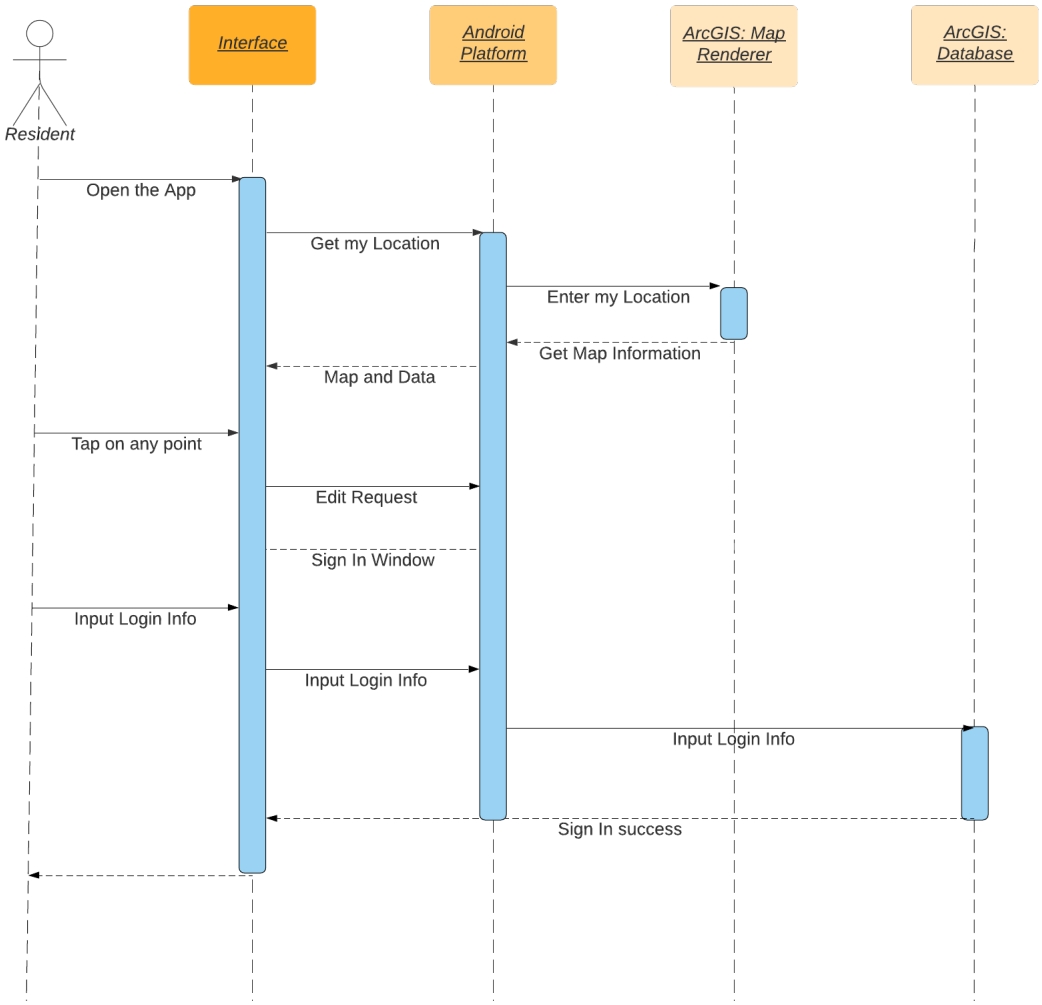


Figure 14: Sequence diagram of user sign in

4.4 Data Management System

- When a researcher wants to edit detailed information of a certain water source, he/she needs to open the app and tap on a point of water source. After the user taps on a certain point, the ArcGIS will get the location of this point and retrieve its detailed information in the online database, then send the information back to the pop-up system in Android platform. A window with detailed information of this point will pop up on the screen. The researcher can then click on the data or blanks to change or add data. These updates will be sent to the online storage.

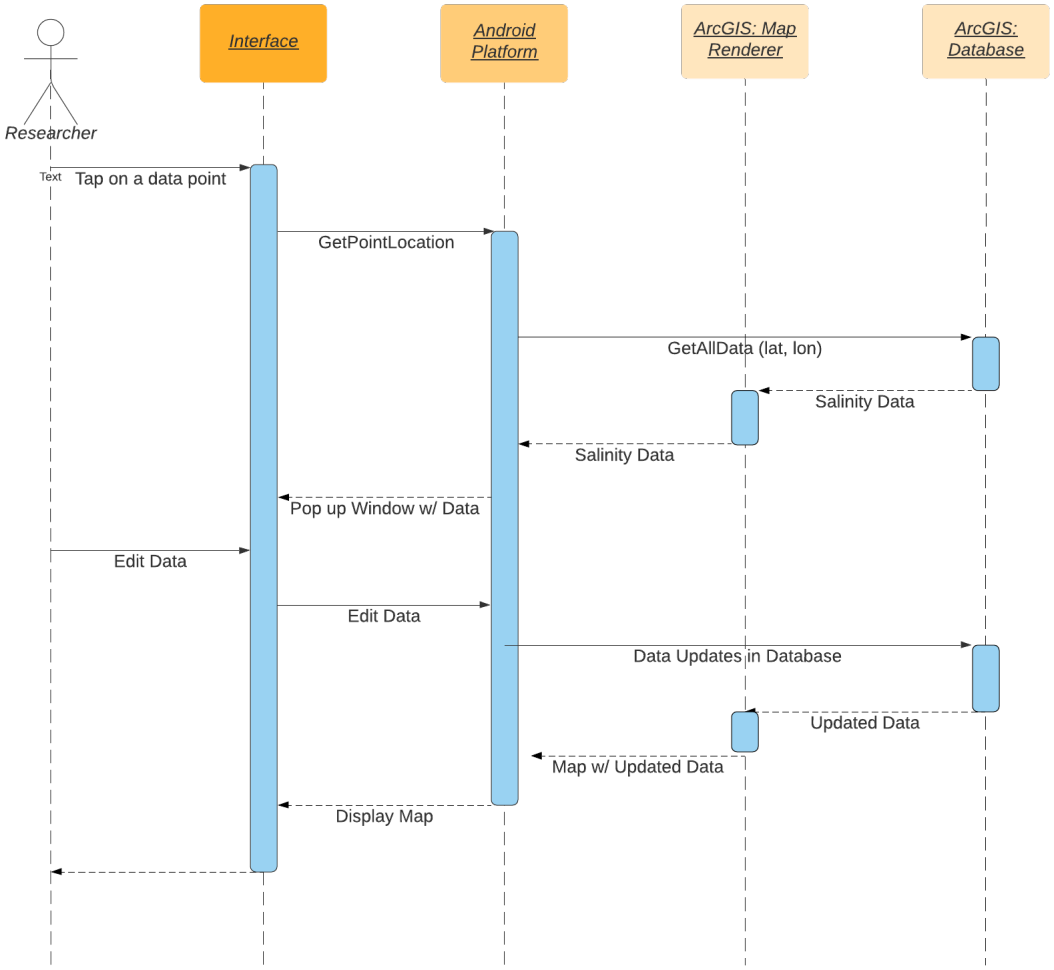


Figure 15: Sequence diagram of editing data

- When a researcher wants to add new data to the online database, he/she needs to click on the '+' button and input the information of this data point in a pop-up form and submit it. After '+' button is clicked, the request of adding data points will be sent to the ArcGIS database, the Android platform will pop up a form for detailed information. Then, after the database receives the information, the new points will be successfully added to the database and sent to the map render. The Android platform will display a map with new points on the screen.

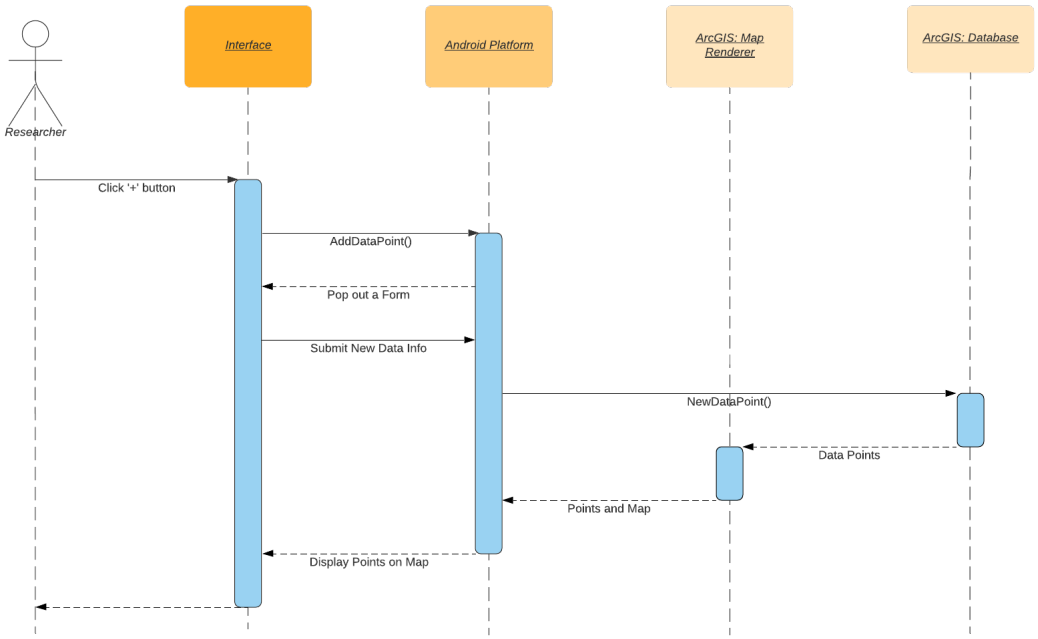


Figure 16: Sequence diagram of adding data

5 Software Architecture

The salinity water map product mainly contains three parts, which include the physical layer, presentation layer and back-end layer. The overall architecture is as following.

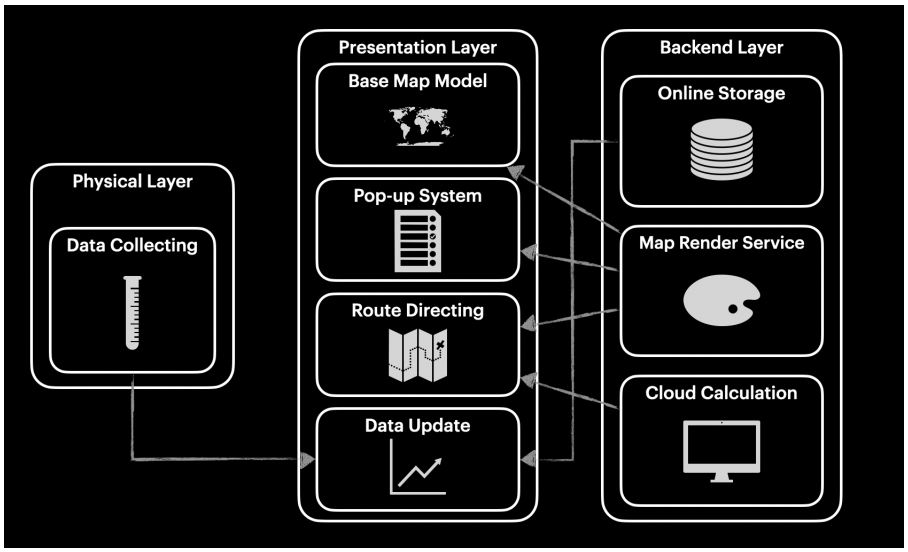


Figure 17: Salinity water map architecture

5.1 Description

First, the physical part in the project mainly involves the device that can collect the salinity of certain water sources. In our case, the water sources are mainly wells. In some cases, users of the application may need to upload new data to the database. We may take advantage of **Bluetooth** to receive latest data from the device or rely on users to manually update the data in our app as shown in Figure 6.

The second one is the presentation layer, which means the interface and functions that are open to users. The main systems are as following.

1. Base map model mainly provides the basic water salinity information to users. The interface contains a map with data points that represent water source with salinity data. This system cooperates with ArcGIS Map render service through run-time ArcGIS API to get map from backend.
   The base map model can be configured through the ArcGIS online service. Besides the basic world map, other layers such as real-time precipitation or wind information can also be rendered to the base map. The real-time layers are mostly provided by ArcGIS service for free, and the database behind is also maintained by ArcGIS.
   The communication between Android mobile end and the ArcGIS database is through integrated APIs provided by Esri libraries.

2. Pop-up system gives users the option to see further information of certain data points. The information will be showed in a pop-up window once the user touch the data point. The pop-up window is constructed by Android mobile end, and it needs to fetch the detailed attributes from the background database. The communication between ArcGIS online database and the mobile end is completed by integrated libraries provided by Esri.

3. Destination directing function gives that users input a destination, and the application will find the nearest water source with the lowest salinity within a fixed distance. The directing is completed by ArcGIS routing service through the corresponding API.

4. Data update system prepares for the case where users want to update the data of certain water source. The mobile end receives data from outside device or get input directly from the user, and then it will renew the data to backend ArcGIS online database.

The last layer is the backend, which provides the main calculation, visualization, database for the mobile end.

1. ArcGIS online storage is one of ArcGIS service that can serve as a database. In the database stores the water salinity data for different locations.

2. Render part is also completed by ArcGIS service. It will generate visualized layer based on the data in database. The generated layers will also be stored in database.

3. Cloud calculation service integrates some general functions that may be often-used in map application including routing function and so on.

5.2 External System

In the project, parts that require implementing mainly fall on the presentation layer. The external system, or Web stack, we use is ArcGIS, which integrates many APIs for map developing, together with a online database system.

For the presentation layer, we implement the mobile app in the Android environment, which is the other main Web stack in our project.

5.3 Design choice

The architecture follows a layer design pattern, which divides the software mainly into three parts, from front end to background.

The first benefit is that implementation can also fall into three parts following the architecture. The front end, which is the presentation layer, falls into Android development environment, while the back end might be completed through the help of ArcGIS web stack. The process implementation will be more clear.

Secondly, each layer contains some main functions or systems that need to be completed. Each of them has the least connection with each other in the same layer. In this way, the process of implementation of each system can become independent from each other, and even if development of one system fails, it would not affect the process of other parts.

6 Implementation Considerations

In order to make our product more accessible to our users and easier for developer to implement and debug, we take many factors into consideration when implement out product and analyse both their advantages and disadvantages. In this section, we illustrate our implementation considerations in two aspects, framework and platform.

6.1 Framework Choice and Relating Components

The core functionality of out product is provided by geographic information system (GIS) framework. Thus the choice of which GIS frameworks we use can affect development process and user experience. In order to find the most suitable GIS framework for our product, we compared several popular GIS frameworks, Mapbox Studio, ArcGIS, and Leaflet. We mainly focused on data processing, interaction experience, functionality supported, platform supported, and so on when we evaluates the GIS frameworks.

6.1.1 Framework choice

In this project, we evaluate three different GIS frameworks, Mapbox Studio, ArcGIS, and Leaflet. There features that closely related to our product components are collected and listed in the tables below.

1. **Mapbox Studio**

<table>
<tr><td>Framework</td><td>Mapbox Studio</td></tr>
<tr><td>Data Format Supported</td><td><code>MBTiles</code>, <code>KML</code>, <code>GPX</code>, <code>GeoJSON</code>, <code>ShapeFile(zipped)</code>, <code>csv</code></td></tr>
<tr><td>Interaction Experience</td><td>Allow the following interactions: pinch, rotation, single tap, double tap, two-finger tap, pan, two-finger drag, One-finger zoom.</td></tr>
<tr><td>Functionality Supported</td><td>• Offline maps: Able to do more without a data connection by storing your maps offline.
• There are several approaches that can be used to add markers and shapes to a map, which allows us to add marker on the map easily.
• Runtime styling: Mapbox's runtime styling features allow you direct control over every layer in your maps with code. It's now possible to create dynamic maps and visualizations that aren't possible with other mobile maps SDKs.
• Support augmented reality navigation.</td></tr>
<tr><td>Platform Supported</td><td>iOS</td></tr>
<tr><td>Price</td><td>Measured by Monthly active users:
<table><tr><td>Monthly active users</td><td>Cost per 1,000</td></tr><tr><td>Up to 25,000</td><td>Free</td></tr><tr><td>25,001 to 125,000</td><td>$4.00</td></tr><tr><td>125,001 to 250,00</td><td>$3.20</td></tr><tr><td>250,001 to 1,250,000</td><td>$2.40</td></tr></table></td></tr>
</table>

Table 2: GIS framework: Mapbox Studio

2. **ArcGIS**

Framework	ArcGIS
Data Format Supported	`csv`, `TXT`, `GPX`, `GeoJSON`, `ShapeFile`
Data Processing	The Data Management toolbox provides a rich and varied collection of tools that are used to develop, manage, and maintain feature classes, datasets, layers, and raster data structures.
Interaction Experience	Able to Rotate map, set initial center and scale, show map extent, show map scale, swipe, swipe basemapes, navigate from view model, and etc.
Functionality Supported	ArcGIS Desktop is available at different product levels, with increasing functionality. The general functionality includes: • 2D and 3D visualization • Search and geocode • Advanced cartography and symbology including support for rotation, offset, vector/font based symbols and military symbology • Directions and Routing • Geometry operations like clip, buffer, intersect, simplify, union, and split • Offline data access and app usability
Platform Supported	Android, iOS, Windows, macOS, Linux
Price	Essentials Plan contains: 10,000 route transactions per month and over 40 GB of tile and data storage for free. Upgrade to builder plan cost $125 per month.

Table 3: GIS framework: ArcGIS

3. **Leaflet**

Framework	Leaflet
Data Format Supported	`GeoJSON`
Interaction Experience	Mouse/touch interaction enabled. Zoom and attribution controls.
Functionality Supported	• Can add markers, polylines, polygons, circles, and popups. • Supports Popups. • Support event handling.
Platform Supported	Since it uses JavaScript. We need webbrowser to launch it.
Price	Free.

Table 4: GIS framework: Leaflet

After taking all these factors into consideration, we decide to chose ArcGIS as our major framework. ArcGIS is the framework that able to satisfy all of our requirements for implementation as well as to achieve high accessibility to client. Thus, to build the entire system, we use Java and Android Studio to build our frontend, ArcGIS as the core for our backend, and we use APIs provided by ArcGIS to connect the frontend and the backend.

6.1.2 Relating Choice to Components

The core functionality for our salinity maps is uploading customized data and displaying salinity data on our map. Our salinity water map architecture contains three layers, physical layer, presentation layer and backend layer.
According to the Table 1, 2, and 3 showed in last section. The reason we choose ArcGIS framework is illustrated as the following part:

- **Physical Layer** In the physical layer our main task is collecting salinity data which will later be sent to our GIS framework. The data is provided by our customer and we would like to have a framework that have less restrictions on data format that our customer provided. According to the tables, we noted that Mapbox studio and ArcGIS supported much more data format than Leaflet. Thus, for this layer we prefer using Mapbox studio and ArcGIS.

- **Presentation Layer** In the presentation layer, we will design an interface for our user. In our perspective, we would like to develop on a platform that we are familiar with, easy to debug, and easy to be deployed. Since our team members have experience in Android development and we have available android devices, implement our app using Java and Android studio become our top choice. Thus we need to select a GIS system that support Android development. In this case, Mapbox studio and ArcGIS becomes our top choice.

  Besides the platform, to give our customer a better interaction experience with the salinity map. We also take interaction experience into consideration. Among all three

GIS framework, ArcGIS provides more functions to interact with the map and more options to display the map.

- **Backend Layer** In the backend layer, the main functionality we need is:

  1. Store data online.
  2. GIS map render service.
  3. GIS calculation service.

  All three GIS framework are able to do these. Moreover, Mapbox studio and ArcGIS support offline data access and app usability. This function enable our user to use our product without internet access. This will help our customer in rural area.

- **Price** As a VE441 course project, our budget is limited. Mapbox studio is more suitable for large company with greater monthly active user. Leaflet is free but the functionality and supported platform is limited. ArcGIS is a platform with good price, since currently the free essentials plan is enough support for our project.

Thus based on the above reason, we chose ArcGIS as our main framework.

6.1.3 Choice of Deployment Platform

For this project, we chose Android as our deployment platform over iOS. The reason is listed below.

- We are targeting a broad global audience and Android have the greatest global market share. According to the map, people in developing country uses Android more and they are our target user who have high demand to track water salinity nearby.

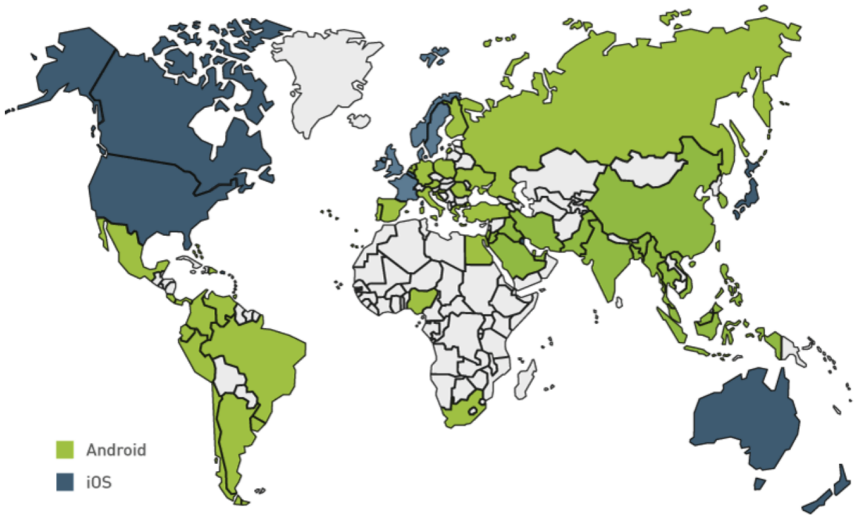


Figure 18: Domain platform used in particular area.

- Our developer is more familiar with Android development which will make our development process more efficient.
- Developing an App for Android allows more flexibility with features. Because Android is open source, there's more flexibility to customize your app — building the features and functions that your audience wants.

Thus according to the above reasons, we decided to use Android as our platform.

7 Team Organization and Appraisal

7.1 Organization of the team

All team members have been collaborating much on this project and actively involved in the app development.

1. Chenhao Li: Technical programmer. Collaborated on the Software architecture design. Constructed the back-end base map on ArcGIS online. Added the real-time layers from ArcGIS public service to the back-end base map. Set the sample salinity data. Implemented the pop-up window on the Android mobile end. Added the navigation function on Android mobile end for double clicking.

2. Yongwei Yuan: Technical programmer. Designed and implemented the navigation system of the salinity map. Collaborated on the implementation of the pop-up window on the Android Platform. Responsible for the User Interface Design. Also collaborated on the Software architecture design.

3. Haoxuan Zhu: Technical programmer. Designed and implemented the welcome page. Implemented user account system using backend database. Implemented sign in/up pages and integrated into the app. Also studied the results of customer interviews and analyzed the customer requirements to give insights for the project.

4. Yiming Shi: Technical programmer. Designed and implemented the navigation system. Researched and studied the tools and resources for this project. Designed and created the MVP video for the Salinity Map App.

5. Jiaxi Zhou: Project manager. Designed and implemented the navigation system. Drew the Sequence Diagrams for users' requirements and designed the software architecture. Planned for customer interviews. Designed the MVP video for the Salinity Map App.

Our team collaborated effectively and efficiently to design and create the Salinity Map application throughout this project. One or two members take turns to lead the whole group moving forward.

7.2 Project outcome

1. Extendable back-end base map. The base map of the product is flexible to adjust the layers on it. As the base map is stored on ArcGIS online, it is easy to add other public layers containing weather related information to the product.

2. Intuitive mobile map. On the mobile end, the product gives a rich information to users including the current location, real-time precipitation and wind information. Clicking on the data point will give users the detailed information and a better idea of the climate at that point.

3. Utility functions. The app provides users navigation to the data points they want to reach. Combined with the data update system, the app gives users better experience to interact with the back-end database.

7.3 Issues encountered

- We heavily rely on ArcGIS. If ArcGIS panics, our app panics.
- AcrGIS server in US may run slow in other countries.
- Currently anyone could modify the system and we didn't distinguish different users and only grant access to those user with high credibility.
- Cache data to boost render speed.
- Waiting time is not stable and user may experience longer lag due to the long waiting time for ArcGIS.
- Public data base may be easily corrupted on ArcGIS online.

7.4 Future plans

- The Salinity Map App is still in the MVP stage. It needs further development in the future.
- Use web framework to build app (so that we can cover all iOS, ANDROID, and computer users.)
- Add interface to connect with salinity reader so that researchers can add data directly from the tool.
- Allow users to export current data.
- Change or add massive salinity data via uploaded files (csv files.)