PRODUCT REQUIREMENTS DOCUMENT v.1

CS Capstone | Fall 2020

Team

	Daniel Kluzner, *Lead*	danielkluzner@ucsb.edu
	Erick Rios, *Scribe*	erickrios@ucsb.edu
	Sai Kathika	saiprem@ucsb.edu
	Huanhua Xu	huanhua_xu@ucsb.edu
	Kelly Yeh	kellyyeh@ucsb.edu

Project Title SmartFarming

Company AgMonitor
Team Name PowWow++

Introduction

PROBLEM STATEMENT

Farming and agriculture play a critical role in the modern world, providing crops for a growing population as well as counteracting world hunger. According to research, by 2100 the world will need to have increased its calorie intake by eighty percent to feed the population, suggesting farming efficiency will be an important issue. Already with the COVID-19 pandemic, and California droughts in recent years, farming inefficiency is an underlying issue that has affected many people.

Additionally, currently in the farm worker hierarchy there is contention between agronomists and ranch managers, due to a lack of data. Agronomists are a third party entrusted with analyzing data in the farms and advising the ranch managers on how to use their resources properly. Because the data is not easily accessible, ranch managers can be skeptical of the agronomist's opinions, especially since the agronomist is not typically in the field everyday, like the ranch manager. The technology we propose will fill the data gap and assist farmers with all these issues.

BACKGROUND

AgMonitor is an agriculturally focused software company that creates technology to help farmers keep track of data, manage tasks, communicate with their teams, and schedule upcoming

irrigation/fertigation times. AgMonitor provides a plethora of agriculture monitoring software with specific focuses on food, water, and energy, helping users ranging up and down the farming hierarchy. In the context of this project, AgMonitor is focusing on providing the resources for us to expand upon their mobile app.

GOAL/IMPLEMENTATION

Our goal with SmartFarming is to create a mobile app using AgMonitor's existing codebase and sensor data to help farms with water efficiency and fertigation issues in hopes of providing an application that can effectively reduce the waste of resources in fields. AgMonitor already has a robust web app as well as a mobile app with limited features. We plan on incorporating new features and expanding the current mobile app to include functionality available on the web app.

One of our main focuses is providing charting features that will display sensor data for different time intervals so that users can be better advised on how their field is using resources and yielding crops. Currently, graphs are available on the web app only, so providing them in a mobile version will greatly benefit farmers. For chart API's we plan on using either Victory or React-Native-Charts, as both these API's are relatively simple to use in a React-Native heavy environment and provide endless customization options. There will also be a calendar feature in the app that will provide an easy way for farmers to manage upcoming tasks and schedule them for certain times. When the app is booted up, there will be a login screen where users can sign in with their credentials with the help of Firebase. From there, users can choose to either view the map or view the calendar. The map screen will allow users to move around and select certain areas of the field to inspect and evaluate, or select certain sensors to view data in charts. From the calendar screen the user will be able to view/edit their schedule.

From a technological standpoint, the app will be coded with React Native for versatility that allows functionality in IOS and Android. The React Native framework is proven to be an excellent tool for creating intuitive user interfaces, and is already used heavily in the current mobile application. The app will work in tandem with a Google Maps API to provide the mapping functionality so users can choose which field and which sensor they want to analyze. This Google Maps functionality is also already in use in the current mobile app. For the sensor data, API calls will fetch data from our firebase database, using either Flask or Django. Flask and Django are both Python frameworks that help with handling large amounts of data transfers in a quick and efficient manner. Firebase will store all our sensor data, as well as user credentials for login purposes.

CURRENT SOLUTIONS

Agronomists typically use pen and paper to tabulate data when conducting field inspections, or resort to bringing their laptops out onto the field to use the web application.

Another way they gather data is by extracting from memory cards inside the on-site sensors. On top of that inconvenience, agronomists are restricted to only looking at the data every two weeks.

As mentioned before, AgMonitor has an existing mobile application that is optimized for irrigators but does not show data that would be useful to ranch managers and agronomists. There are also features that are only available on the web application, such as the charts, which makes it difficult for ranch managers to upload data and edit plans for their crew on the field. There exist a few other competitors that have similar products, but their charts are less intuitive and harder to use.

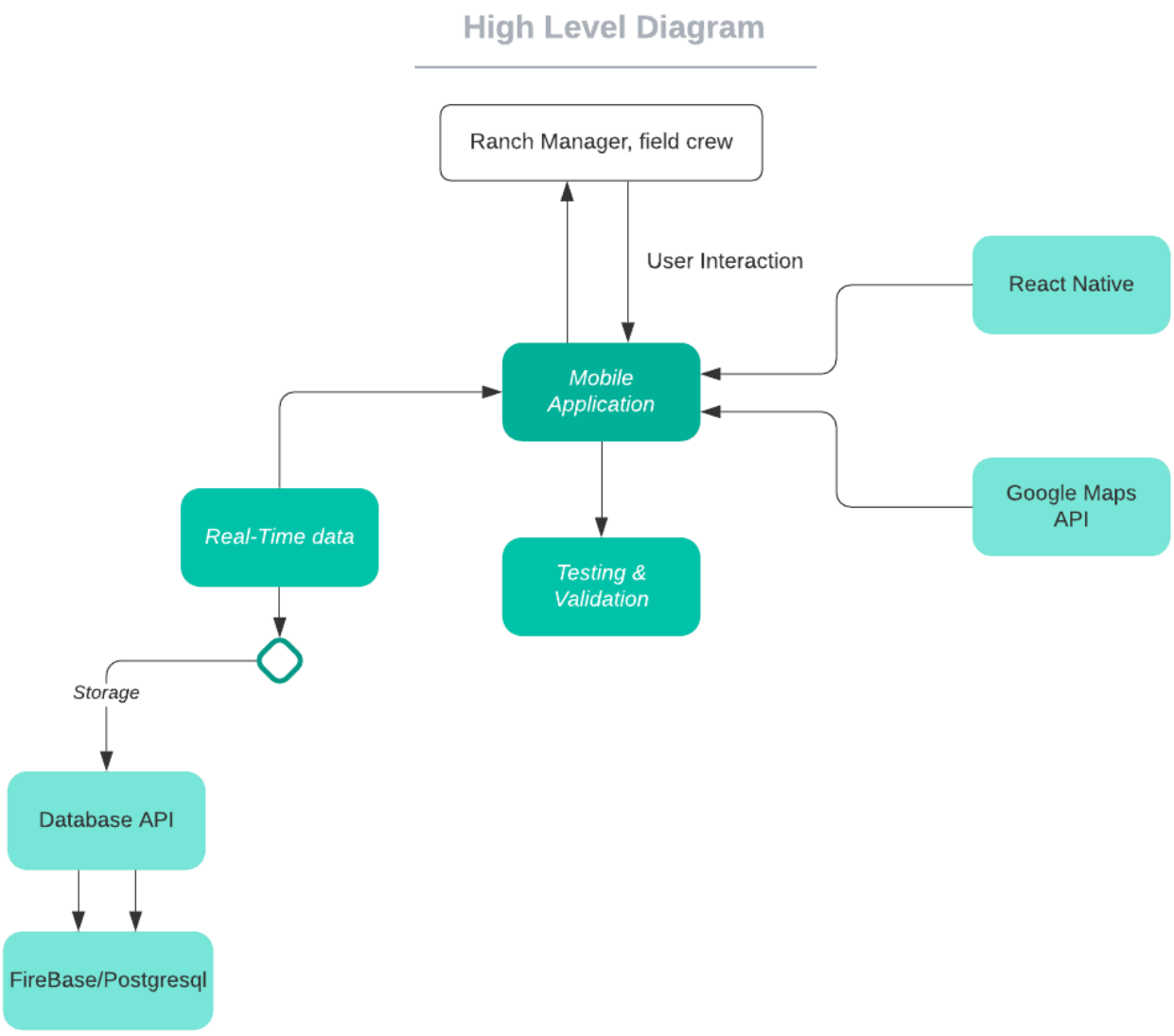
In the context of the current solutions available, a robust mobile application would be greatly beneficial for farm workers, especially ranch managers and agronomists.

Assumptions

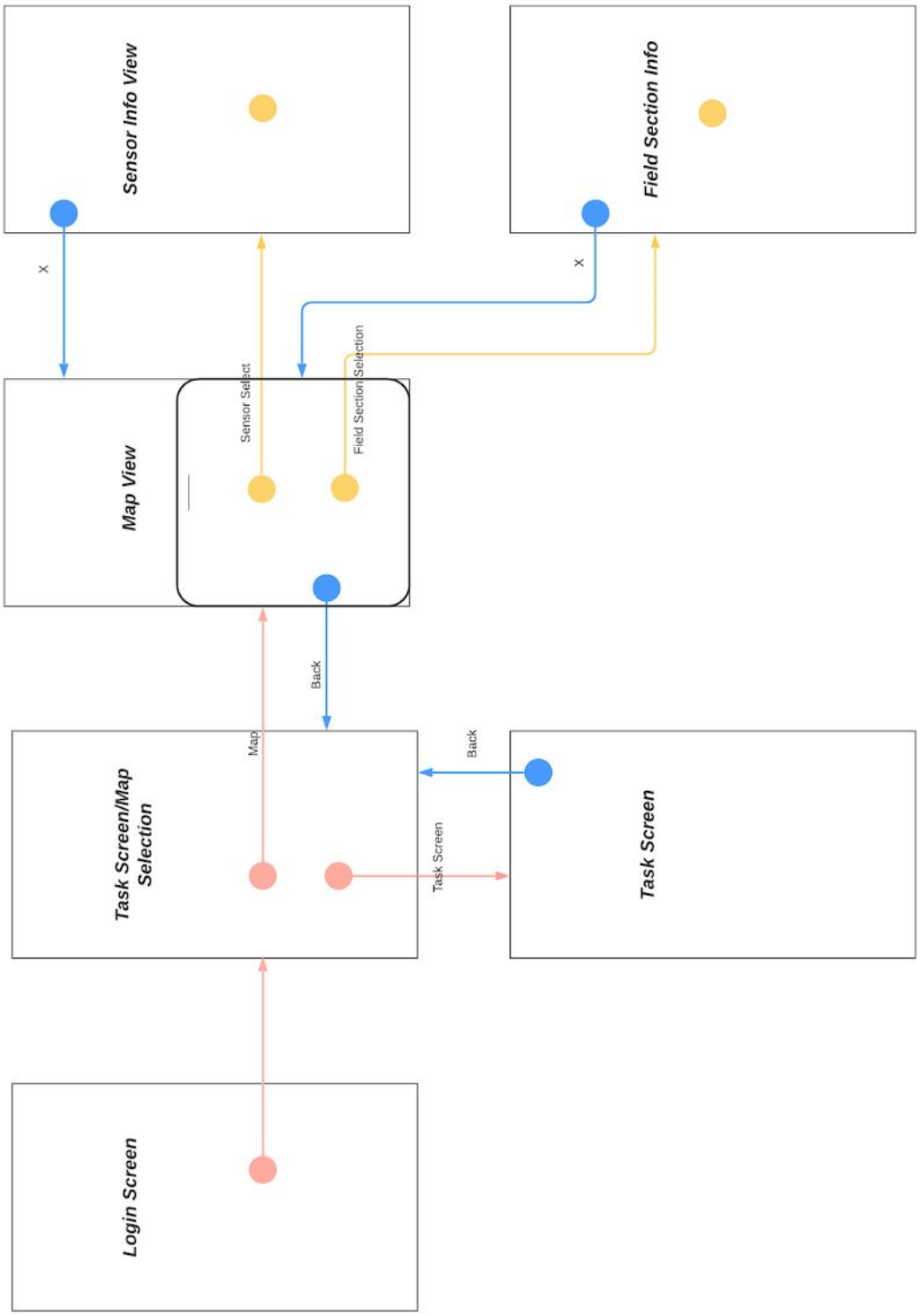
- Our target users will be farm owners and workers in California (only provided data for farms in California)
- Users will need to login and create to access the data
- Data will be publicly available for anyone to see and use as long as they sign up
- The user-interface will be easy to use and intuitive
- Users will be able to quickly comprehend the meaning of the data through charts and other visual tools

System Architecture Overview

HIGH LEVEL DIAGRAM



MOBILE APP UI DESIGN



User Stories

---

1. As a user, I can register an account to save personal information and access the data
    - Acceptance Criteria: After creating an account, the user can use the account name and password he or she has set up to login the application again. If a wrong username or password is entered, then an error message will show up and prompt the user to try again

2. As a registered user, I can check the information I saved (specific map area, specific sensors on the field, etc.)
    - Acceptance Criteria:
        i. Scenario #1 (User with registered account): Given the user already registered an account, when the user logs in to the application with correct username and password, then the user can check their saved information
        ii. Scenario #2 (User with registered account): Given the user already registered an account, when the user denotes a specific area or sensor as a 'favorite' item, if something is updated inside the favourite items, the the user will receive notification
        iii. Scenario #3 (User without account): Given the user hasn't created an account yet, when the user tries to login, the user will be prompted to sign up first

3. As a user, I can zoom in on the maps and check specific sensors on the mobile phone so that I can get the real-time data about local situation
    - Acceptance Criteria: After logging in to the application, the user can select a specific field to view as a map. The map will allow users to zoom in or zoom out, and drag to move the view. Also, when the user taps on a specific senor on the map, the user is able to check the data for that sensor (https://github.com/danielkluzner/SmartFarming/issues/2)

4. As a user, I can choose different time intervals (last day, last two weeks, last month, last year, last two years) and view the data corresponding to that interval
    - Acceptance Criteria: While checking a specific sensor, the user can select different time intervals which will display the data for the corresponding time period (https://github.com/danielkluzner/SmartFarming/issues/1)

5. As a user, when viewing a chart I can zoom in and view the data for smaller intervals of time, depending how much I zoomed in

- Acceptance Criteria: The user can zoom in and zoom out on the chart by pinching with two fingers. The more the user zooms in, the smaller the intervals of time will become (https://github.com/danielkluzner/SmartFarming/issues/3)

6. As a user, I can choose what kind of data to display on the graph (elevation, pressure, etc.) so that I can clearly see the trend of development
    - Acceptance Criteria: There are tabs that indicate different variables, and the user can press any of them so that the data of the chosen variable will be shown on the graph (https://github.com/danielkluzner/SmartFarming/issues/4)

7. As a user, I can check the hours since the last irrigation and show irrigation status so that I can see where water is used inefficiently and adjust water use
    - Acceptance Criteria: The user can select a specific sensor. By clicking on the sensor, it will show the irrigation schedule and the water use (https://github.com/danielkluzner/SmartFarming/issues/5)

8. As a ranch manager, I can edit the calendar and add new events for my crew using the mobile app
    - Acceptance Criteria: if the user is an admin, then the user can update the calendar by adding new events, then it will notify the group about the update (https://github.com/danielkluzner/SmartFarming/issues/6)

9. As a user, I can search for a specific area or sensor and display their data
    - Acceptance Criteria: By entering specific information such as serial number on the search bar, the user can directly access the data of a specific area or sensor. If the information and serial number don't exist, it will return an error message and prompt the user to enter correct information (https://github.com/danielkluzner/SmartFarming/issues/7)

10. As a user, I can select a field on the map and displays the water use and water efficiency in this specific area instead of the whole map
    - Acceptance Criteria: By selecting a specific area on the map, the user can select the area he want to know on the map, and it will display a graph that indicates the water use and water efficiency over time for the selected area (https://github.com/danielkluzner/SmartFarming/issues/8)

Project Specific

FRONTEND

- Using React Native to develop on Android and iOS platforms simultaneously
- Implement login page and registration page where users can enter credentials to enter into main interface of application
- The charts will be displayed on the frontend using a React Native API
- Chart API that we plan on utilizing: Victory or React-Native-Charts
- The app will enter the frontend through App.js where it will be routed to different screens— Login/Register based on users actions

BACKEND

- We do not have access to the backend yet but we presume it will use either Flask or Django to interact with the database
- The backend will interact with the frontend, pull data from the database, and perform all the machine learning analysis and calculations necessary

DATABASE

- Will be using Firebase along with Firestore as our database
- Currently stores various sensor data
- Research an API that can automatically format and consolidate all of the data to be stored in database

LOGIN AUTHENTICATION

- Using Firebase authentication to verify and store user credentials
- Created persistent login feature where users will not have to sign in again after they exited the app
- Firebase backend stores username, email, and hashed password

Appendix A: Technologies Used

DATA STORAGE AND WEB APP

- PostgreSQL
- Firebase
- Python, NumPy, Pandas, Scikit-Learn
- Django, Flask
- Javascript

MOBILE APPLICATION

- React Native
- Android, iOS

OTHER TOOLS

- API's: Google Maps
- Docker
- Windows WSL
- LucidChart
- Firestore
- Android SDK

When the app is booted up, there will be a login screen where users can sign in with their credentials with the help of Firebase. For chart API's we plan on using either Victory or React-Native-Charts, as both these API's are relatively simple to use in a React-Native heavy environment and provide endless customization options.

From a technological standpoint, the app will be coded with React Native for versatility that allows functionality in IOS and Android. The React Native framework is proven to be an excellent tool for creating intuitive user interfaces, and is already used heavily in the current mobile application. The app will work in tandem with a Google Maps API to provide the mapping functionality so users can choose which field and which sensor they want to analyze. This Google Maps functionality is also already in use in the current mobile app. For the sensor data, API calls will fetch data from our firebase database, using either Flask or Django. Flask and Django are both Python frameworks that help with handling large amounts of data transfers in a quick and efficient manner. Firebase will store all our sensor data, as well as user credentials for login purposes.