The American University in Cairo
School of Sciences and Engineering

plantiful
AgriTech Solutions

CSCE 4981
Senior Project II

Group 6
Software Requirements Specifications v2.0

Written By
Farah Seifeldin - 900160195
Lobna ElHawary - 900160270
Maram Abbas - 900153570
Mariam ElZiady - 900161869
Samer Basta - 900150910

Supervised By
Dr. Mohamed Shalan
Dr. Tamer ElBatt

May 22, 2021

1. Introduction

1.1 Purpose

This Software Requirements Specification (Revision v2.0) document is meant to be provided as a general overview for the entire **plantiful** system, essentially, by describing the set of requirements, constraints, and features.

1.2 Document conventions

The text is all Times New Roman.
Heading 1 is of font 18 and bold. It signifies the start of a section.
Heading 2 is of font 16. It signifies the start of a subsection.
Heading 3 is of font 14 and underlined. It signifies a subsection within subsections (Heading 2).
Heading 4 is of font 13 and in bold. It signifies a subsection within subsections of subsections (Heading 3).
Normal text is of font 12. When bold is used here, it is to reference sections or subsections.
Figure numbering for the attached figures will be of font 12 and in bold. The first number will denote which section and the following numbers denote which subsections and the final number will denote the order of the figure in the section. So, **Figure 4.3.5** means this is the 4th section, 3rd subsection and this is the 5th photo in this subsection. So, **Figure 5.3.5.1** means this is the 5th section, 3rd sub-section, 3rd sub-sub-section and this is the 1st photo in this subsection. **Figure C** means this is in appendix C.

1.3 Intended audience

The intended audience for this SRS is split into subgroups. The first subgroup is future software engineers or developers looking to expand this project further by enhancing features and improving our constraints. The second consists of smart agriculture professionals that would want to understand the requirements set for this project. The third and last group is clients of our project, i.e. CARES' employees (where we are implementing our idea initially as a proof of concept) and any future users.

1.4 Additional information

Currently, we are collaborating with CARES as they already have an equipped greenhouse where they run their experiments, which serves as a great central hub for our own testing. Their current experiment is on cucumbers, so that's the crop type we'll be focusing on. So, the CARES' greenhouse is our proof of concept, and some of our decisions and design implementations currently are biased towards their physical structure. However, most of our components are generalized to work with greenhouses of multiple sizes and could even work outdoors (granted there are Internet and sources of electricity to supply to our hardware components).

1.5 Contact information/SRS team members

Farah Seifeldin	-	farahseifeldin@aucegypt.edu
Lobna ElHawary	-	lobna1443@aucegypt.edu
Maram Abbas	-	maram_abbas@aucegypt.edu
Mariam El Ziady	-	mariamelziady@aucegypt.edu
Samer Basta	-	samerr@aucegypt.edu

2. Overall Description

2.1 Product perspective

The context of our project is that it is a community deployment product. It is here to replace the existing domain of manually doing the agonizingly tedious jobs of agricultural researchers and experts who face the following problems:
Firstly, researchers have to manually check sensor readings to note them down throughout the day which opens room for human error.
Secondly, especially for a large greenhouse, manually noting the sensor readings of plants and examining each plant's health status can be exceptionally labour intensive and time consuming. This same logic applies to overlooking the leaves of each crop.
Finally, researchers are trying to find the minimum amount of resources, such as water, needed for optimal growth, which requires consistent, detailed, around-the-block micro-analysis, which is difficult to accomplish properly with humans with our human error.

By utilizing IoT for the sensor readings and ML for plant health status and CV for plant stage detection, our project can aid researchers in cutting down on the labour intensity, time, and long term cost it would take to run the operation manually.

2.2 Product functions

- Automatic temperature, humidity, pH and soil moisture sensor readings every set time interval (default is 10 minutes).
- Automatic photo capturing every day.
- A user-friendly web application that includes signing up, logging in, adding and managing agricultural projects, witnessing the latest sensor readings, viewing the sensor readings in a graph format over time, and downloading the sensor readings. The application can serve to view the latest photo taken of the cameras, the current growth stage, and if there are any unhealthy plants detected and to notify if current sensor readings are outside the ideal range or in case of finding an unhealthy plant.
  The current implementation is in **Appendix B: Web Application Use-Case**.

- Growth stage detection to detect the growth stage the plant is in.
- Health detection to detect any unhealthy plants.

2.3 User classes and characteristics

2.3.1 Agricultural Experts

These experts form the working force behind agriculture that are looking to enhance their fields by having on-the-go data sensor readings of their fields. We seek to help them ensure that their crops are within the ideal parameters required for ideal crop growth and yield.
Their main requirements are availability and simplicity. If the information is not readily available for them, it will quickly have them turn back to their old methods of operations. They are typically comfortable in their own bubble of agricultural knowledge, as it is vast and wide, so anything more complicated in this field will just deter them away from our project, hence the simplicity.

2.3.2 Agricultural Researchers

These researchers mainly focus on finding breakthroughs in their changing of growing parameters. They favour having more powerful tools at their disposal in order to analyze, compare and contrast data, then come up with findings.
Their main requirements would be availability and user-friendliness. If data is not always readily available to them, their potential findings will be jeopardized, causing their avoidance of our project. User-friendliness is needed to use the powerful, numerous tools that shall be provided. If the interface is not user-friendly, the researchers will get lost, again causing their abandoning of the project.

2.4 Operating environment

Our system uses the hardware platform of Arduino IDE where the ESP32 is programmed to respond to the RPi 4. The operating system that hosts the process of sensor reading and transmitting is the Linux on the RPi 4. The geographical location of the system's hardware implementation, the sensors and microcontrollers, is CARES' premises while the software implementation, the GUI server, exists on a Google Cloud server located in Indonesia.

2.5 User environment

The user will interact with the system through the GUI where s/he can view readings and results of the growth stage and health diagnoses. More details are given in **Section 6** in our SDD.

2.6 Design/implementation constraints

2.6.1 Restriction on Technologies

Also, after proving so experimentally, BLE cannot be used to transmit requests from the RPi to the ESP32 and data in the other direction. That is due to the distance between the two microcontrollers being on the boundary of the BLE transmission range which is 17 meters.

2.6.2 Restricting Business Rules

We do not have an access card to the CARES premises as the institution only gives those to researchers and staff. This compels us to ask for access each time we want to visit the greenhouse, which shall affect the efficiency of our testing of the remote sensing system.

2.6.3 Restriction on Hardware

2.6.3.1 Timing

The transmission time needed every ten minutes is 0.14 seconds for every ESP32. Thus, we must keep the ESP32 on for a larger duration than this transmission duration.

2.6.3.2 Memory

As for memory, we need 6MB per day to be stored in SQL, so a database server which will allow for storage for the duration of the experiment (usually 90 days) is key. We will not need any substantial memory storage on the RPi since the camera footage is streamed and not stored and the taken then saved snapshots are deleted once successfully stored in the database.

2.6.3.3 Processor

The processor we need for our remote sensing computer, the RPi4, is one which must be both robust and reliable in order to not add to the transmission time of the greenhouse data.

2.6.3.4 Size

The size of the hardware kept in the greenhouse cannot be so large that it takes up experiment space. That is unlikely, however, since the microcontrollers and sensors are reasonably small.
As for cost, we must ensure that we do not spend more on the hardware system in order to make room in our funds for the software system.

2.6.4 Restriction on Formats

Also, the data files downloaded from the GUI must be in CSV format to allow for easy parsing in case the user wishes to run a program of his/her own on the data.

2.7 Documentation

An electronic user manual will be provided for the remote sensing system and distributed to CARES by email and provided on our GitHub link. The manual is to explain how to operate and maintain the hardware.
As for the GUI, a tutorial will be offered the first time the user uses the application in order to facilitate navigating through the website.

2.8 Assumptions

2.8.1 Greenhouse architecture

We are assuming that the greenhouse’s spatial architecture will remain the same, as to not obscure the automated car that has our camera attached to it.

2.8.2 Greenhouse conditions

We are assuming the climate control system, of the greenhouse, will not suffer any failures for the duration of the experiment. In case water from the cooling pads collects at the middle, not the walls, of the greenhouse, it might fall on and thus damage the remote sensing system. Also, if the temperature control unit falters, the hardware components might be exposed to high temperature, thus increasing the components' failure rate and compromising the reliability of the whole system.

2.8.3 Network Speed

We are assuming that the speed of the network established at CARES will always exceed a minimum set value. This assumption goes into calculating the transmission time needed, so if that speed is not attained, transmission problems might occur given the set on-time of the sensor reader (the ESP32).

3. External Interface Requirements

3.1 User Interface

3.1.1 GUI style

We are implementing a custom GUI style that we think will be the most attractive to our users.

3.1.2 Stylistic standards

The font used for the GUI is Arial. The icon on all webpages at the top left is the **plantiful** logo which we designed. Button labels vary from the function of the webpage; examples of labels are SIGNUP, APPLY, and NEW PROJECT. Non-clickable text is green, black and white, and clickable text is blue. Buttons are filled with green. Tabs are also filled with green and placed at the top. Examples of our webpages are inserted in our SDD.

3.1.3 Screen constraints

Our website is adapted to all screen layouts and will appear the same no matter the device.

3.1.4 Standard items

Each webpage has a navigation bar and side bar. This contains buttons redirecting to logout, notifications, create projects, and edit projects/groups.

3.1.5 Messages

Messages appear in a listed form on the notifications web page from the navigation bar.

3.1.6 Software localization

The about page overviews the importance of agricultural research in a way that is relevant to all users across the globe. Our mission and vision are stated without involving any culture-specific problems.

3.1.7 Accommodations

See Section **5.4.1** for details on this.

3.2 Hardware interfaces

Each ESP32 reads from the sensors using GPIO. The readings consist of a float value for each sensor.

3.3 Software interfaces

3.3.1 Linux

The Linux on the RPi4 hosts all code needed for remote sensing and transmitting. A program is run by the operating system that sends a request to each and every ESP32 every ten minutes. After each ESP32 replies back with the data, the program inserts that data into the SQL database with the current time and date.

Also, another program runs on Linux at two known times each day. This program starts a streaming session with each of the cameras and snaps an image of the stream. The image is then uploaded to the SQL database with the current time and date.

3.3.2 Heroku

This is where our web application is deployed. It has proper integrations with GitHub, Django and Postgresql, which we were already utilizing for our project. As well as the feature to add a Machine Learning model for our plant analyzer algorithms.

3.3.3 Heroku's PostgreSQL Database

The database used for the whole project is that of Heroku's basic PostgreSQL Database package. The database contains user details, project profiles/settings, sensor readings, notifications and captured images, among other entities. More explanation is offered in **Section 4.1** of our SDD.

3.3.4 Heroku’s Cloud Storage

This is where the images taken by the IP camera are stored to be used with our plant analyzer algorithms. An add-on on Heroku is used for cloud storage.

3.4 Communication protocols and interfaces

The communication protocols between our components go as follows:

- Between each ESP32 and the RPi 4: Wi-Fi (100 KB/s) using HTTPS requests
- Between RPi 4 and Heroku: Wi-Fi (100 KB/s) using SSL certificates for security
- Between RPi 4 and camera: IP

4. System Features

4.1 User Sign up

4.1.1 Description and priority

For the user to have access to and interact with our system, s/he must sign up for an account on the system's web application. This feature is an integral part of the system.

4.1.2 Action/Result

When a user signs up for an account, his/her email, username, and password are saved in the system's database, so the user can login using these credentials. In addition, the provided security token will be verified before saving his/her information in the database to ensure that this user has access to the system.

4.1.3 Functional Requirements

- Application allows users to sign up.
- Security token is verified.
- Application saves user credentials in the database.
- Application reports errors to users.

4.2 User Login

4.2.1 Description and Priority

This feature allows the user to login to the web application in order to start using its features. This feature is an integral part of the system.

4.2.2 Action/Result

When the user's credentials are entered, authentication happens by comparing the entered data with the decrypted stored data in the system's database. If the credentials match the ones retrieved from the database, the user will have access to the application. In case the credentials are incorrect, an error message will be prompted to the user and s/he will be requested to enter them again or sign up in case the username does not exist in our database.

4.2.3 Functional Requirements

- Application allows users to login.
- Application authenticates user credentials.
- Application reports back errors to users.

4.3 Create Project

4.3.1 Description and Priority

This feature allows users to create new projects in the system's web application that will be linked to the sensor-system. This feature is an integral part of the system, since it is only through a project that the user can define the group settings and view the group's collected data. Essentially, the project is the interface of the web application to the hardware system.

4.3.2 Action/Result

If no errors occur in defining the project settings when creating it, the user is redirected to the "Create Group" page to continue setting up the project.

4.3.3 Functional Requirements

- Application allows users to create new projects.
- Application reports back errors to the user.

4.4 Create Group

4.4.1 Description and Priority

This feature allows users to create new groups in the project they just created. This feature is an integral part of the system, since it is only through a group that the user can enter the settings and the system can begin collecting sensor readings for it. Creating a group involves setting the group's settings or to retrieve settings from previous groups defined by the same user.

4.4.2 Action/Result

If no errors occur in defining the group settings when creating it, the user is redirected to the "Sensor Nodes" page to continue setting up the project by providing names for the sensor nodes.

4.4.3 Functional Requirements

- Application allows users to create new groups, settings and sensor nodes.
- Application reports back errors to the user.

4.5 Edit Project

4.4.1 Description and Priority

Users can modify the settings of an existing project. This feature is of moderate importance since the system can function properly without it. It exists to make the projects more flexible to changing requirements.

4.4.2 Action/Result

If the user has the privileges to do so for the specified project, s/he can edit the project by changing any of the project settings. If the edits are successful, the project settings will be modified and saved in the system's database. The application will return the user to the project dashboard. In case errors occurred while modifying the settings, the system will report this to the user and ask him/her to try again.

4.4.3 Functional Requirements

- Application allows privileged users to edit created projects.
- Application saves new settings in the database.
- Application displays error messages to the user.

4.5 View Numeric Sensor Data

4.5.1 Description and Priority

This feature allows users to view the sensor data for every block in their created projects. The data is shown for every sensor (temperature, humidity, pH, and soil moisture). Latest readings are displayed first, then the user can choose to see line charts showing overall data. The tabular form can be downloaded in csv format. Users can select the start and end date of the data they want as well as pick one or more sensors. This feature is an integral part of the system.

4.5.2 Action/Result

When the user clicks on a project, they can then select a group from that project. Then, the group's average sensor data is displayed. The user can then pick the generate line graph option to see the overall trend of the data for an individual sensor. The result of this action is that the data is fetched from the database from the start of the project until that specific point in time and displayed in the form of a line chart. The rest of clicking download data is that the selected data will be retrieved from the database and written in a .csv file that will be automatically downloaded locally on the user's device.

4.5.3 Functional Requirements

- ESP32 reads sensor data from sensors.
- RPi4 retrieves sensor data from corresponding ESP32.
- RPi4 uploads sensor data to the database.

- Camera uploads images to the cloud storage.
- Application retrieves sensor data from the database.
- Application displays sensor data for a specific project group to the user.
- Application allows the user to download sensor data in csv format.
- Application creates a csv file and writes sensor data to it.
- Application downloads csv file locally on user’s device.

4.6 Generate Security Token

4.6.1 Description and Priority

An admin user has the privilege to invite users to the system. This happens by generating a security token for every invited user. This token can be used by the invited users to register for an account in the system. This feature has moderate importance as it regulates who has access to our system since our system requires that only users with the system’s installed hardware devices are allowed to create accounts.

4.6.2 Action/Result

The generated security token can be shared privately by the admin to other users through offline or encrypted online means. The users can then use this token to sign up for an account as explained in **Section 4.1**.

4.6.3 Functional Requirements

- Application allows admin users to generate security token per invited user.
- Application saves this security token temporarily in the database.

4.7 View Notifications

4.7.1 Description and Priority

When sensor readings go outside the specified boundaries, the system sends notifications to the user. The user can then view these notifications and act accordingly. This feature is high priority because it saves the user time to check sensor data every day.

4.7.2 Action/Result

The notifications appear to the user in the project dashboard in the notifications tab.

4.7.3 Functional Requirements

- Application checks if sensor data lies within the specified ranges.
- Application sends notifications to users when data is not in specified ranges.

5. Other Nonfunctional Requirements

5.1 Performance requirements

5.1.1 Response Time

The average response time should be about 5 seconds.

5.1.2 Scalability

The software shall be built with scalability in mind. Although for now, our intended clients will not exceed 100, our system has the potential to grow and should be able to accommodate a growing user base.

5.1.3 Power Usage

The power usage of our web application and our hardware sensor system should be kept as low as possible. For the web application, this will be achieved through following power saving best practices such as not loading unnecessary resources when the application is idle. For our hardware system, this is done by putting the ESP32 boards in low power sleep mode when they are not reading data which will significantly decrease power consumption. The estimated lifetime for each sensor node taking readings every 10 minutes is 14 days. Batteries will need to be recharged after that.

5.1.4 Memory Usage

The application shall not inefficiently consume the memory of the phone/laptop.

5.2 Safety requirements

The hardware system should not disrupt the greenhouse nor negatively impact the growth of the crops. The sensor circuits should be properly connected to ensure that no hazardous electrical problems occur.

5.3 Security requirements

- Authentication of parties and proper login mechanisms shall be used. Each user will have a unique username and password. The password will have a minimum number of 8 characters with at least 1 number, 1 uppercase letter, 1 lowercase letter and 1 special character. It will also be encrypted in the database. The user's credentials will be required during login.
- The system shall not be accessible to hackers, trojans and malicious users.
- Data shall be backed up every 24 hours on the cloud.
- Data shall be transmitted securely between the server and the client using end-to-end encryption. Data should be transmitted without any changes.
- Confidentiality of user and project information.
- The client shall be able to retrieve their account in case s/he forgets their username and password via other identification methods like sending a verification code through their mobile phone number or through email.
- The system shall be self-protecting and resistive of attacks, penetrations and unauthorized modifications.

5.4 Software quality attributes

5.4.1 Accessibility

The system shall be easily accessible to people with disabilities.

- The web application shall display the interface with a large font without degrading the user-experience or truncating displayed text.
- Displayed information shall be equally recognizable to colour-blinded and healthy users.
- All media shall include alt text and descriptions associated with them.
- The web application shall be available in English for now and Arabic in the future.
- Title tags, skip links, and layout sectioning shall be utilized to ensure that our application can interface easily with assistive technologies.

5.4.2 Reliability

- The system shall be available 99.8% of the time.
- The system shall be available 24 hours per day, 7 days per week.

5.4.3 Maintainability

The system shall be developed in such a way that changes are easily made in the future whether for fixing bugs or for adding functionalities and features. The system shall keep track of hardware components to make sure that they are functioning properly and should report anomalies to users, so they can be fixed.

5.4.4 Ease of Use

Our system shall accommodate users with all levels of computer knowledge especially as our intended users are employees in the agricultural sector with varying computer knowledge.

These objectives will be achieved by enforcing the following measures.

- Friendly graphical user interface
- Informative error messages
- Informative notifications to track projects' progress
- Online help service

Customers shall be able to use the web application without prior training as it shall be self-explanatory.

The following are quantifiable ease-of-use goals that the application shall achieve.

- 1 in every 200 clients might have difficulty using the application.
- 95% of the clients shall be able to easily use the application on their first encounter.

5.4.5 Portability

- The system shall be portable enough to be used on mobile devices, computer devices and on web browsers.

- The application shall operate in different conditions, like different network types, different network bandwidth situations and different screen size ranges.
- The system shall be portable to the extent that if the current hosting becomes too restrictive or expensive, the system can be moved to a new server with minimal changes and minimal downtime.

5.4.6 Traceability

The user shall be able to trace his/her projects' data and progress.

5.4.7 Robustness

- The system shall not need regular manual intervention
- The solution to failures shall be stable and start automatically
- The system shall be able to recover from disasters and problems quickly.
- The system shall not lead to any security breaches in case of failure.
- Time to restart after failure shall be less than 5 minutes
- Percentage of events causing failure shall be 1 in 100

5.5 Project documentation

Project documentation will be included in our GitHub link.

5.6 User documentation

Proper user documentation will be provided to the users of our web application, as well as being included in our GitHub link.

Appendix A: Terminology/Glossary/Definitions list

API

Application Programming Interface. A piece of software that is used to interact with an existing computer system

ERD

Entity Relationship Diagram. A diagram explaining the structure of a database

ESP32

A system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth

IDE

Integrated Development Environment. A program that allows for the running and debugging of applications

IoT

Internet of Things. The field of computer engineering that deals with the embedding of real-life objects with sensors

IP

A network protocol that enables devices to communicate with the Internet

ML

Machine Learning. The field of computing that deals with automated data analysis

CV

Computer Vision. The field of computing that deals with allowing computers to gain understanding from digital images.

RPi 4

Raspberry Pi 4.0. A microcontroller with an operating system and Wi-Fi module

Sensor Block

A block of the sensors of our project: DHT11 (temp and humidity), moisture, and pH

SSL

Secure Sockets Layer. A protocol that ensures data exchanged between a server and client is not compromised through encryption

TLS

Transport Layer Security. A protocol that ensures the security of data transmitted online through encryption

Wi-Fi

A network protocol used to communicate over the Internet

Appendix B: Web Application Use Case

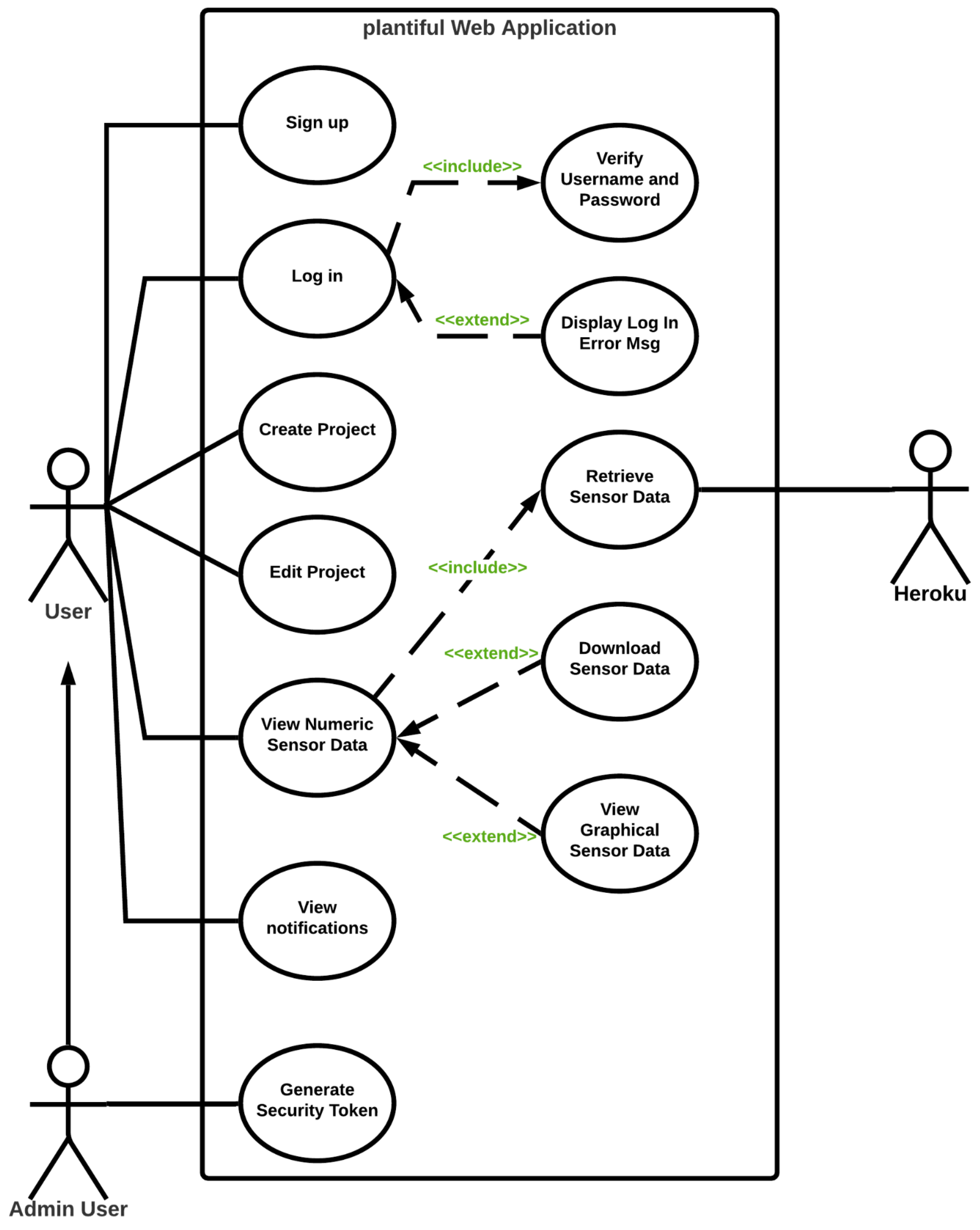

Figure B

Appendix C: Creating Project Activity Diagram

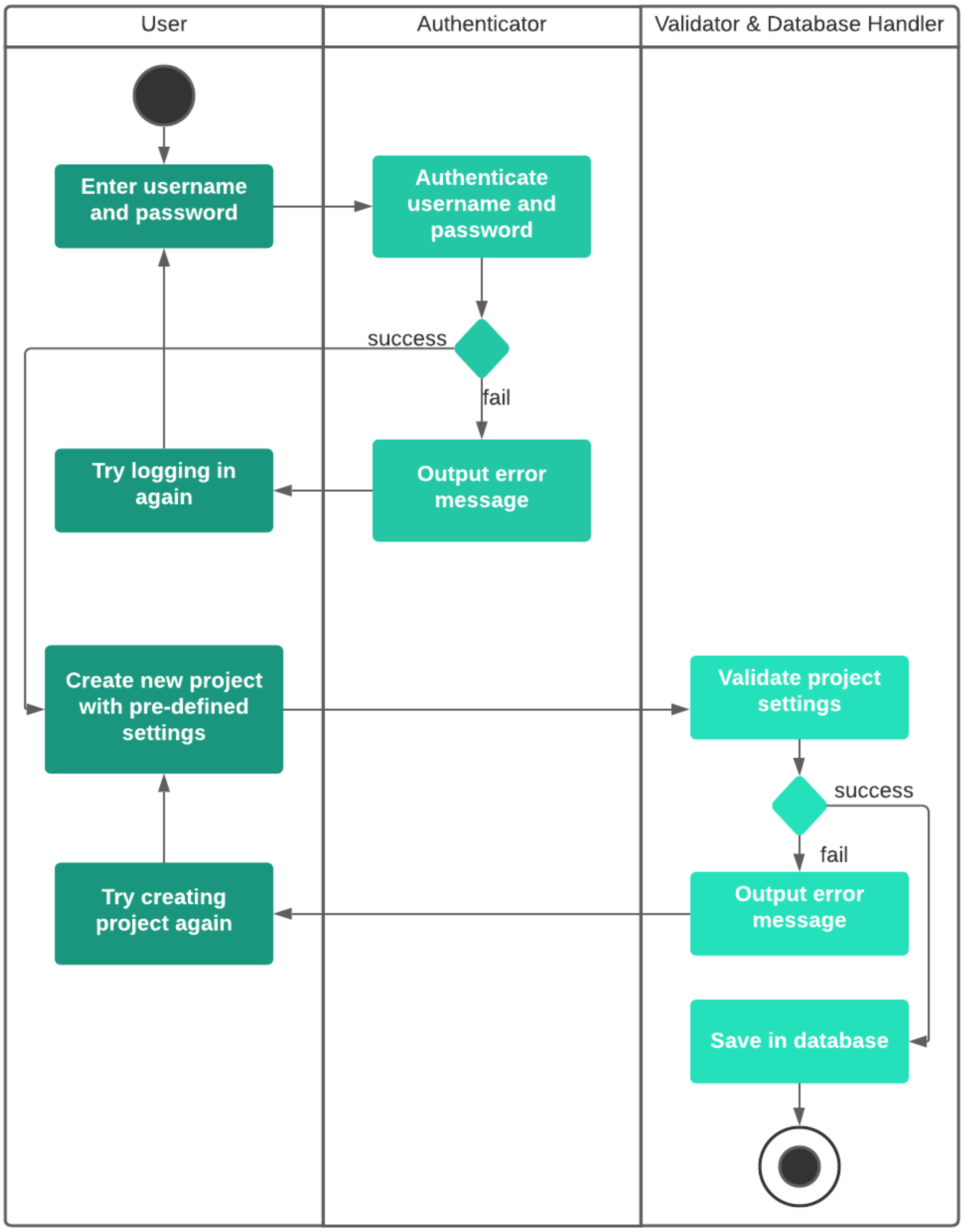
User
Authenticator
Validator & Database Handler
Enter username and password
Authenticate username and password
success
fail
Try logging in again
Output error message
Create new project with pre-defined settings
Validate project settings
success
fail
Try creating project again
Output error message
Save in database

Figure C

Appendix D: Generating Line Graph Activity Diagram

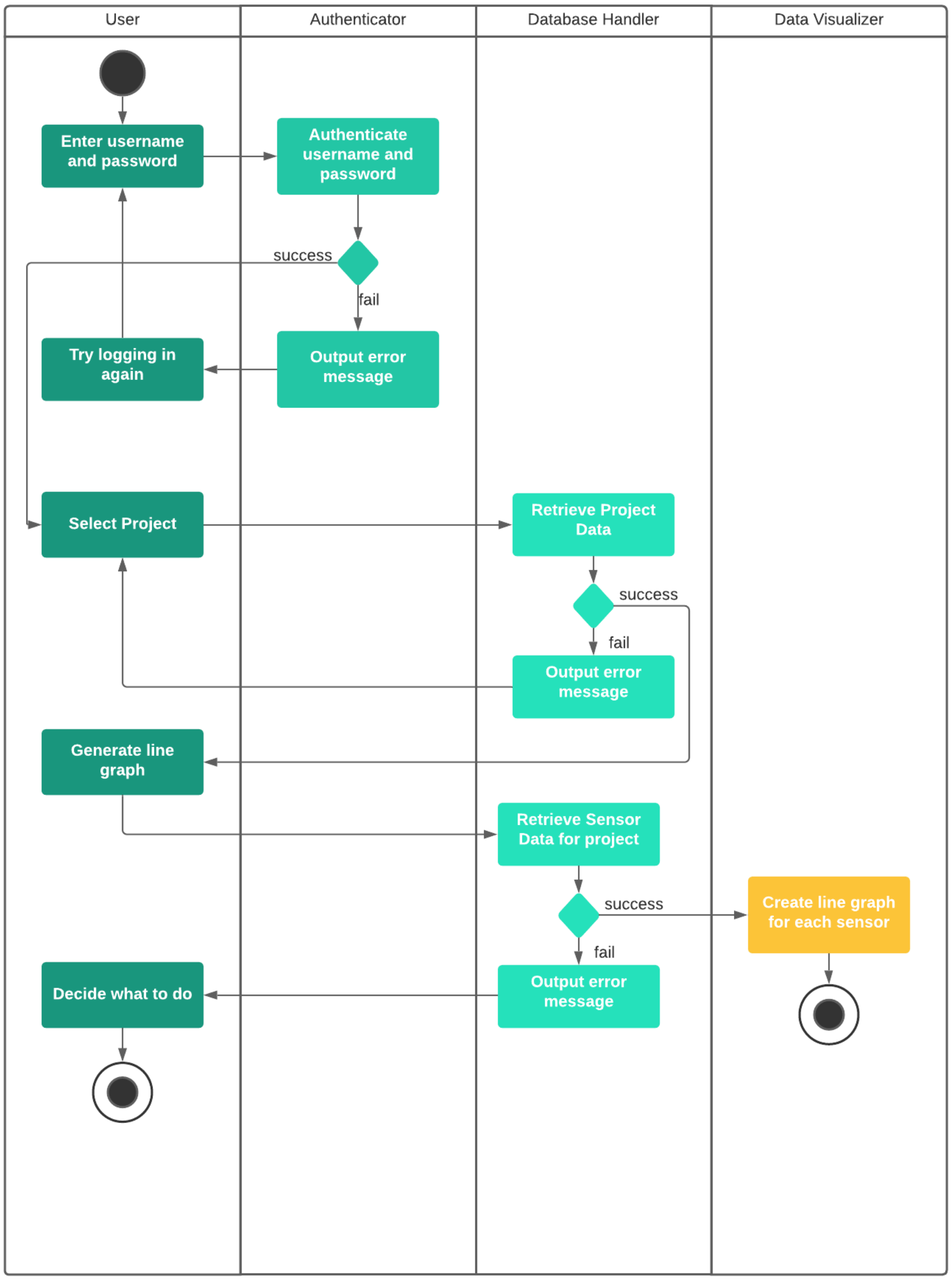

Figure D