CA400 Final Year Project

Functional Specification

Smart Garden

Names	Daniel Pereira - 15364491 Jacob Byrne - 15492172
Date Finished	22/11/19

Ollscoil Chathair Bhaile Átha Cliath
Dublin City University

1. Introduction

1.1 Overview
The Smart Garden system is an automated plant monitoring and caring system that aims to provide the most hands off experience for the end user. The system is made up of multiple sensors and controllers that will live alongside a plant. The sensors will pick up on any changes to the plants environment and the controllers will be able to manipulate the conditions in response. It is up to the users to decide if these changes are made by the user or automatically by the system

All data collected from the sensors will be sent via web requests to a secure database and displayed via a web UI. The UI will provide a dashboard for the user containing live up-to-date information sent by the sensors. It will also be able to provide information on historical data by reading from our database mentioned above.

The client side Web UI will be hosted on an AWS cloud instance and will make requests to the back end (Raspberry PI/Database) in real time in order to provide the live up-to-date information mentioned above.

Due to the nature of the project there are huge amounts of communication between each part of the system. There are also a lot of external communication needs with external providers, for example our hosting provider, AWS as well as ensuring successful hardware communication between sensors and our Raspberry Pi.

1.2 Business Context
The smart garden system is a scalable proof of concept. The project has the potential to be marketed towards large greenhouses in garden centres. It also can work as a home solution targeting different niches in the plant growing markets. Our target will be towards plant lovers with a potential reduced motor ability and for those who might not be able to regularly check up on their plants. Taking this into account, the smart garden system will be designed and developed with the end users ease of use in mind.
Through market research we have been in contact with one potential customer that maintains a large number of plants in a greenhouse style environment and has expressed interest in this project as a proof of concept.

The nature of the project is to be automated, this will allow us to remove the need of constant maintenance for homeowners for their houseplants. Our primary target for this project due to the nature of it being built as a proof of concept are the aforementioned homeowners.

1.3 Glossary
Sensor - Device which detects and measures a physical property (in our case soil moisture, PH etc.)

Raspberry Pi - Small, single board, low cost computer we will be using to interface with sensors.

Pump - Motor to transfer water from a storage unit to the plant base.

Flask - Micro web framework written in Python.

NoSQL - Non-relational database.

AWS - Cloud Computing Provider.

EC2 - AWS service that allows you to launch virtual instances in the Cloud.

GDPR - Guideline Data Protection Regulation. European Law on protection of data and privacy for all individuals in the EU.

Alexa - Virtual Assistant Developed by Amazon.

2. General Description

2.1 Product / System Functions

The smart garden system aims to provide real time and historical information of a plants environment. The system will use multiple sensors to monitor the soil and air around the plant. It will them display this on a web UI giving the user the ability to login and check up on their plant from anywhere they see fit. All information is saved to a database where given a range, a user can check historical data overlaid onto graphs. An alternative method for users to check on the live information of their plant is through the custom Alexa skills that have been integrated into the system.

The Smart Garden will also be able to manipulate the environment it is in. This can be done by a user via the web UI, where users can select specific attributes to alter. Alternatively using Alexa, a user can ask about the conditions of their plant and then tell Alexa to change it. If the User chooses, manipulation of the environment can be done automatically in response to any changes picked up from the sensors.

Users will have 2 modes of operation for the system, automated and manual. In automated mode, the system will continuously monitor the environment and make alterations as needed. Alternatively, in manual mode, rather than the system making alterations, the onus will be on the user to make the required alterations themselves. To ensure the system is still viable in manual mode, users will be notified by email when a particular parameter (soil moisture for example) falls below the desired threshold.

2.2 User Characteristics and Objectives

The targeted users for this project will be aimed at those who have plants at home and either travel too often to care for or have mobility issues and are unable to provide constant care to their plants. With this in mind our project will take into account that some users may have very little experience with complex technologies and as such, our UI will be designed with Schneiderman's 8 Golden Rules in mind to be as user friendly as possible.

We expect users to have some brief experience with technologies and will be able to enter the web UI URL and sign in. This should cover the basic operations for using the system. These include the viewing of the live data to see the condition of the plant

We are also operating under the assumption that users will also be vaguely familiar with the needs of the plants they wish to keep.

2.3 Operational Scenarios

2.3.1 User Login
Objective: Gain access to the application.
Actions: User inputs username and password into a text box to authenticate.
Success End Condition: User is redirected to the application home screen
Fail End Condition: User inputs incorrect data and is given error message to request they try again.

2.3.2 User View Historical Data
Objective: Check previous data of the plant they are maintaining.
Actions: Use the view historical data portion of the UI home screen
Success End Condition: User is presented with historical data in graph form for the previous entries for the plant since the system is created.
Fail End Condition: User is presented with error message that historical data is not available.

2.3.3 User View Live Data
Objective: View the current statistics to do with the plant's environment.
Actions: Use the View current data link on the system home screen
Success End Condition: User is redirected to current information page where they are presented with all readings from the associated sensors with information being provided by the sensors displayed.
Fail End Condition: One or more of the sensors are not online and users are informed accordingly.

2.3.4 User Switch on/off Automatic Mode
Objective: Switch from automated to manual monitoring mode.
Actions: Toggle on/off automated mode using a button on the home screen.
Success End Condition: User will be notified when automated mode is **off**. If mode switched back on user will be notified on the GUI.
Fail End Condition: Automated mode not available and error message shown accordingly on the GUI.

2.3.5 User Manually Alter Conditions
Objective: Alter the plants environment by for example raising the moisture content of the soil.

Actions: Raspberry pi controlled water pump switched on for specific amount of time to water the plant.
Success End Condition: Plant's environment altered by the addition of water.
Fail End Condition: Server unable to communicate with the raspberry pi and error message shown accordingly.

2.3.6 Precondition: Automated Mode off - User notified
Objective: When automatic monitoring is switched off, notify the user when a certain parameter drops below a specified threshold.
Actions: System will notify by email when a sensor detects a parameter below the specified threshold that's defined as healthy.
Success End Condition: User receives email reminding them to check on their system.
Fail End Condition: Notification displayed on UI rather than by email.

2.4 Constraints

The main constraints we will face going forward with the project will included and aren't limited to the following:

- Hardware design (Raspberry Pi connection with multiple sensors)
- Driving Water pump/lamp with 5 volt power output with the Raspberry Pi.
- Handling login/out of web interface
- Time management with University Modules and exams
- Communication with client server from Pi
- Keeping the plant alive
  - Different plants require different environments to flourish. Due to the nature of this project as a proof of concept, we will be focusing on one particular species and research as such that species' requirements and cater to them with our system.
- Designing the application to run smoothly on all browsers including mobile browsers
- Error handling
- UI Design
  - As mentioned in section 2.2, users may not necessarily be technical. With this in mind we will be implementing several design principles to ensure ease of use.
- Raspberry Pi internet access

3. Functional Requirements

3.1 External Requirements

3.1.1. Software: For our project we will be developing using Jetbrains PyCharm IDE. A Raspberry pi running python will take and send data to and from the sensors and controllers.

The Raspberry Pi will also be running a flask web server. This will allow us to send and receive web requests with the cloud based server. Without the flask server, no data will be able to be transferred to the cloud and no command can be received. The pi will implement caching in the event that information cant be sent to the cloud server. The design of the server will have to be “effective” allow it to communicate consistently and effectively with the cloud server.

The second flask server will run on an Amazon EC2 instance. This server will communicate with the raspberry pi via web requests and host the user interface. The data it receives will be pushed to a database to be stored. This flask server is a critical component of the system and will be developed in such a way that it can be redeployed in such an event of a failure occurring.

The web UI will implement AWS Cognito, providing a secure and effective login method for users. The UI will use python and javascript plugins (JustGauge & Matplotlib respectively) to display gathered information to users. It will also take in user inputs such as what mode a users chooses and specific requests to alter environment conditions. There are no systems that depend on the web UI component.

We will be using a non-relational-relational database to store historical data for the purposes of retrieval later on. The web UI relies on the database to display historical data on the graphs.

3.1.2 Hardware: A huge component of this project is the hardware powering our system. As mentioned this will consist of a raspberry pi and a number of sensors as well as a water pump that will be readable/controllable by the Pi. There are a number of potential issues here that we will keep in mind during the dev process. Namely the internet connectivity on the pi which we have rectified by making the device wifi enabled and ethernet enabled.

3.1.3 Third Party Providers: Due to the nature of the project we will be using Third-Party Providers for the purposes of computing/hosting. We will be using AWS as it is one of the largest Cloud Computing Providers in the world. This will be a critical part of the project and as such we are choosing AWS as the major service we are using (EC2) is guaranteed at >=99.99% Availability. We are also relying on this component to communicate with our backend (raspberry pi etc.)

3.2 Interface Requirements

3.2.1 Ease of Use: Ease of use is a primary requirement of our project. Users are expected to have a rudimentary knowledge of computers, with this in mind, the interface will be minimal and focus particularly on the Schneiderman principle of ‘reduce short term memory load’. This will require careful planning and will be an important part of our overall design.

3.3 Reliability Requirements

3.3.1 Constant Uptime: As part of our project is automated, we will need a reliable wifi signal to the raspberry pi. As such, our hardware is fitted with an ethernet port as well as a wifi dongle to ensure it remains active with an internet connection for as long as possible. This is also essential as the Front End will be making regular requests to the back end for historical information

3.4 System Functions

3.4.1 Alter Plant Environment: With the use of a water pump, heat and light lamp the system we plan on implementing will be able to modify the conditions of a plants environment. All these controllers will be connected to a raspberry pi using relays and power control units

3.4.2 Display Data: Our GUI will display both live and historical data depending on what view is selected. This is a core function as it will let the user know at that moment the state of the plant's environment and if any alterations need to be made for example. This data retrieval is important as it will make up a significant portion of our GUI.

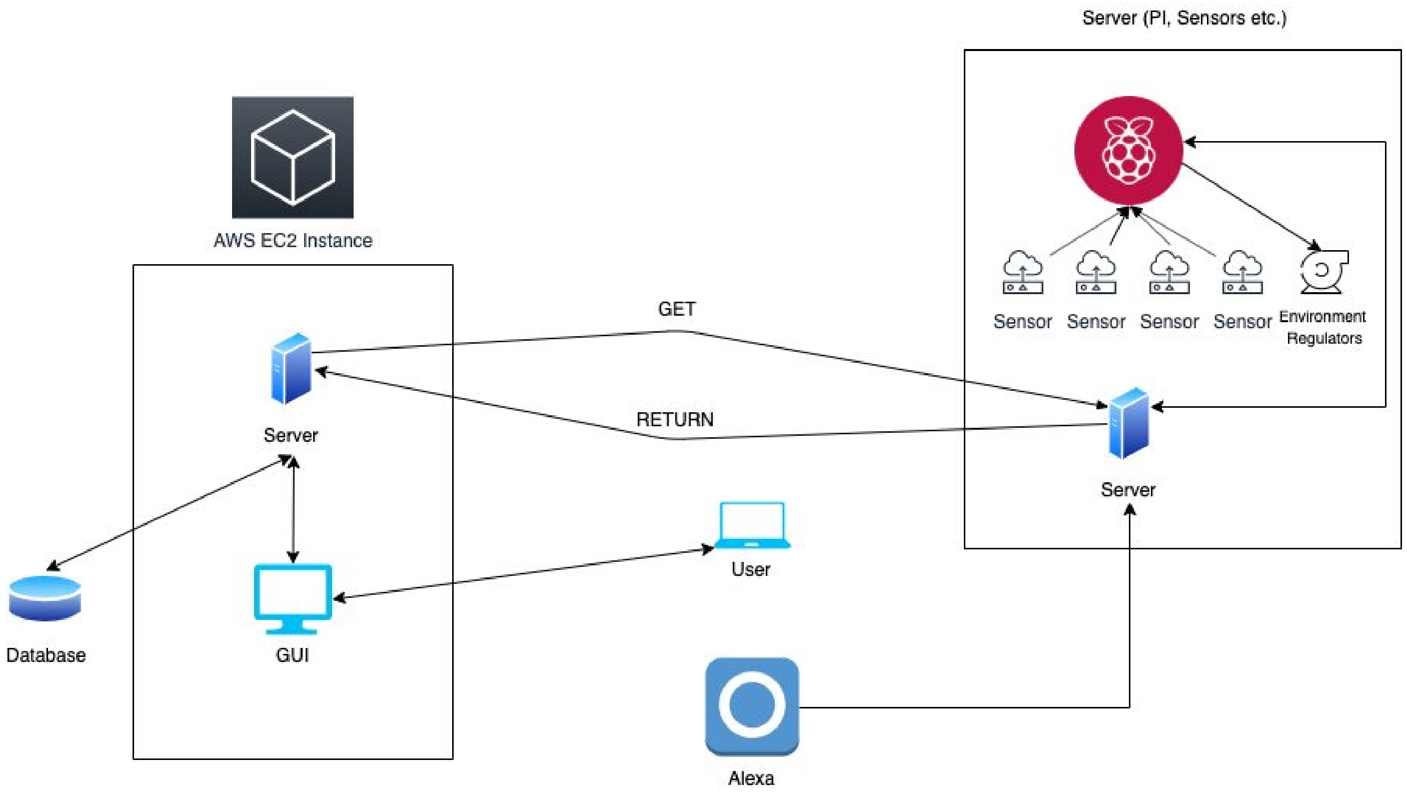
3.4.3 Record Data: The data that is taken in by the cloud server will be saved using a non-relational/relational database model also on the cloud. This will allow efficient access to historical data at all times. This will require database writes whenever recordings are taken. There are potential issues here to do with storage. As the system is up for an extended period of time, memory for the recorded data could become problematic, as such we will be provisioning plenty of disk space as well as monitoring the DB size throughout to ensure we are unaffected by this.

3.4.4 Automated Care: The system we are developing will have the ability to use the current state of conditions of the environment and use these to determine if any are outside given tolerances. Once it works out what factors should be changed, the system can then alter them.

3.4.5 Scalability: Once the system is setup and implemented on a small scale, there is a potential to introduce more raspberry pi's, add tailored sensors and even integrate with existing controllers. This would give us the potential to implement the system up to the scale of a botanic garden centre.

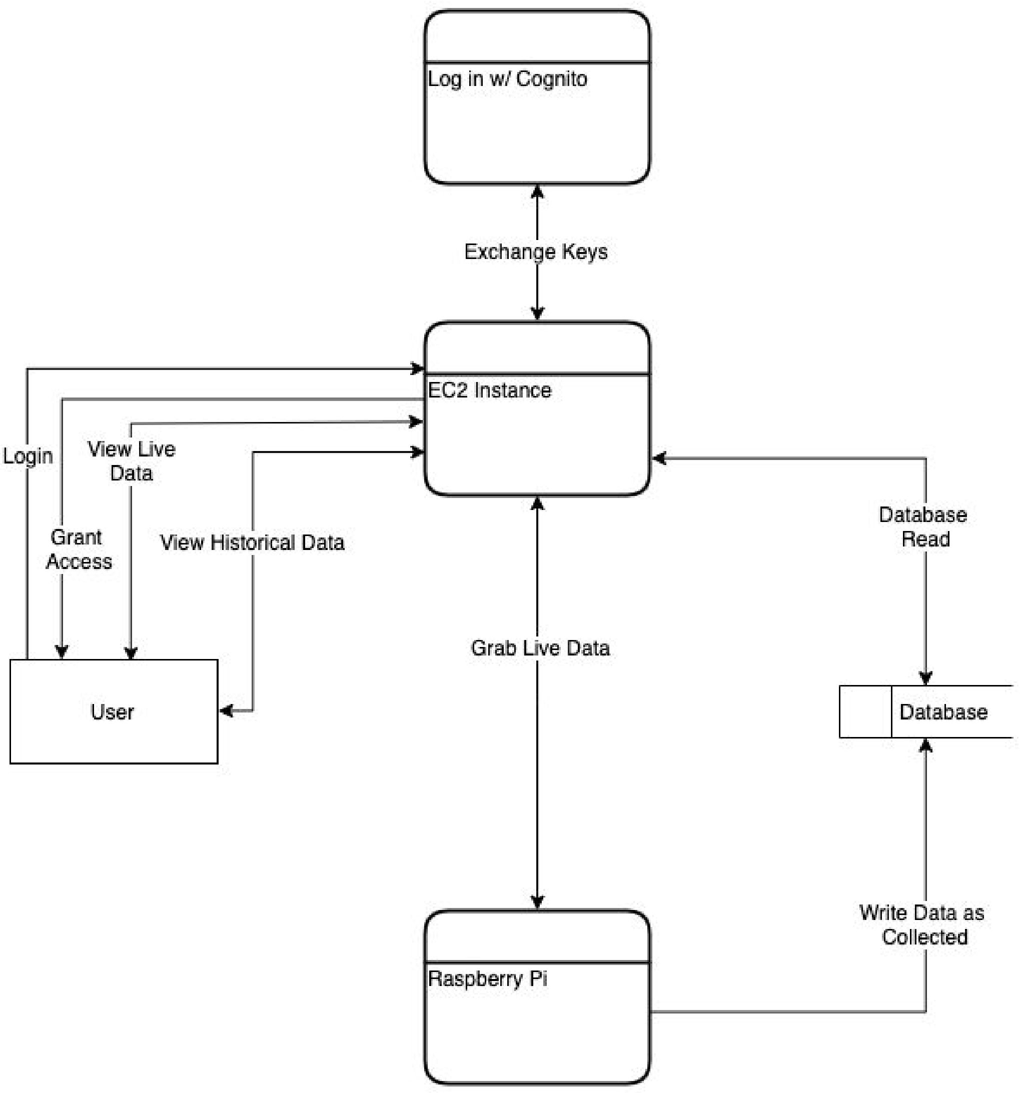
4. System Architecture

4.1 System Architecture Diagram

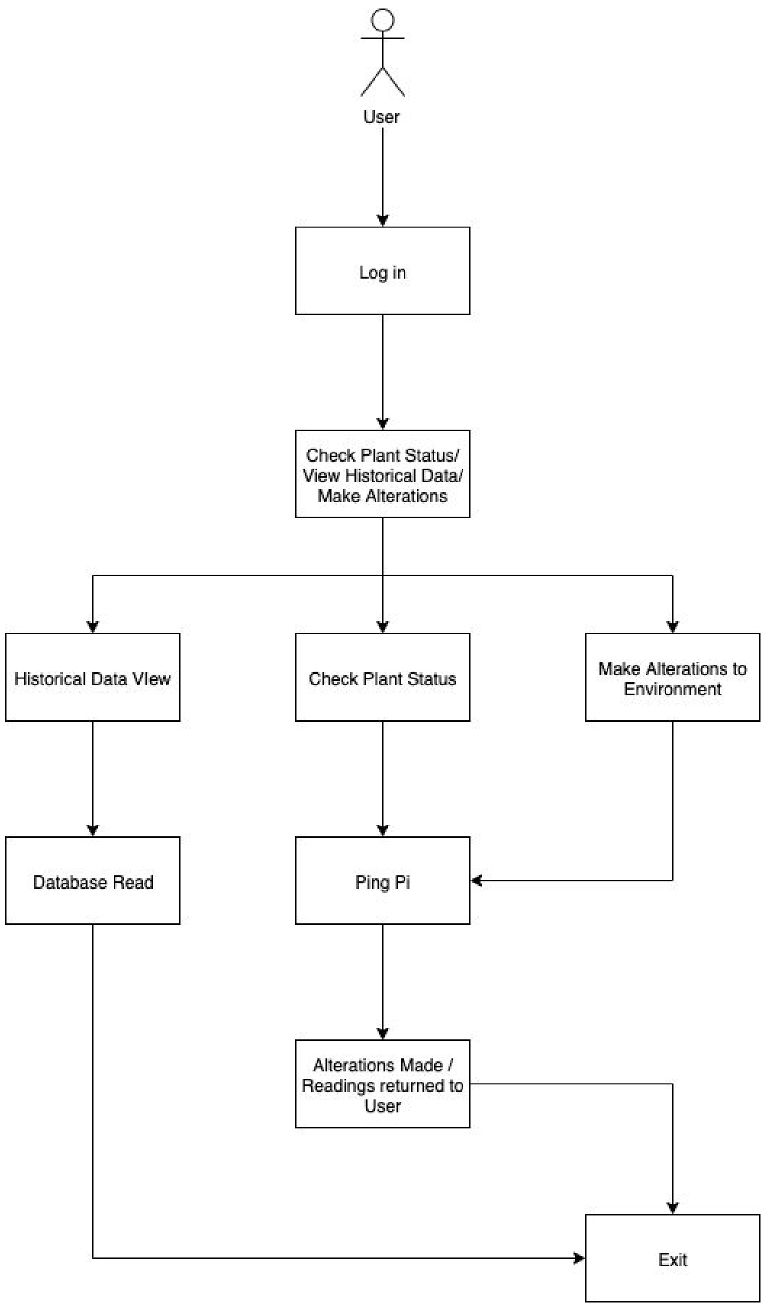


5. High-Level Design

5.1 Data Flow Diagram



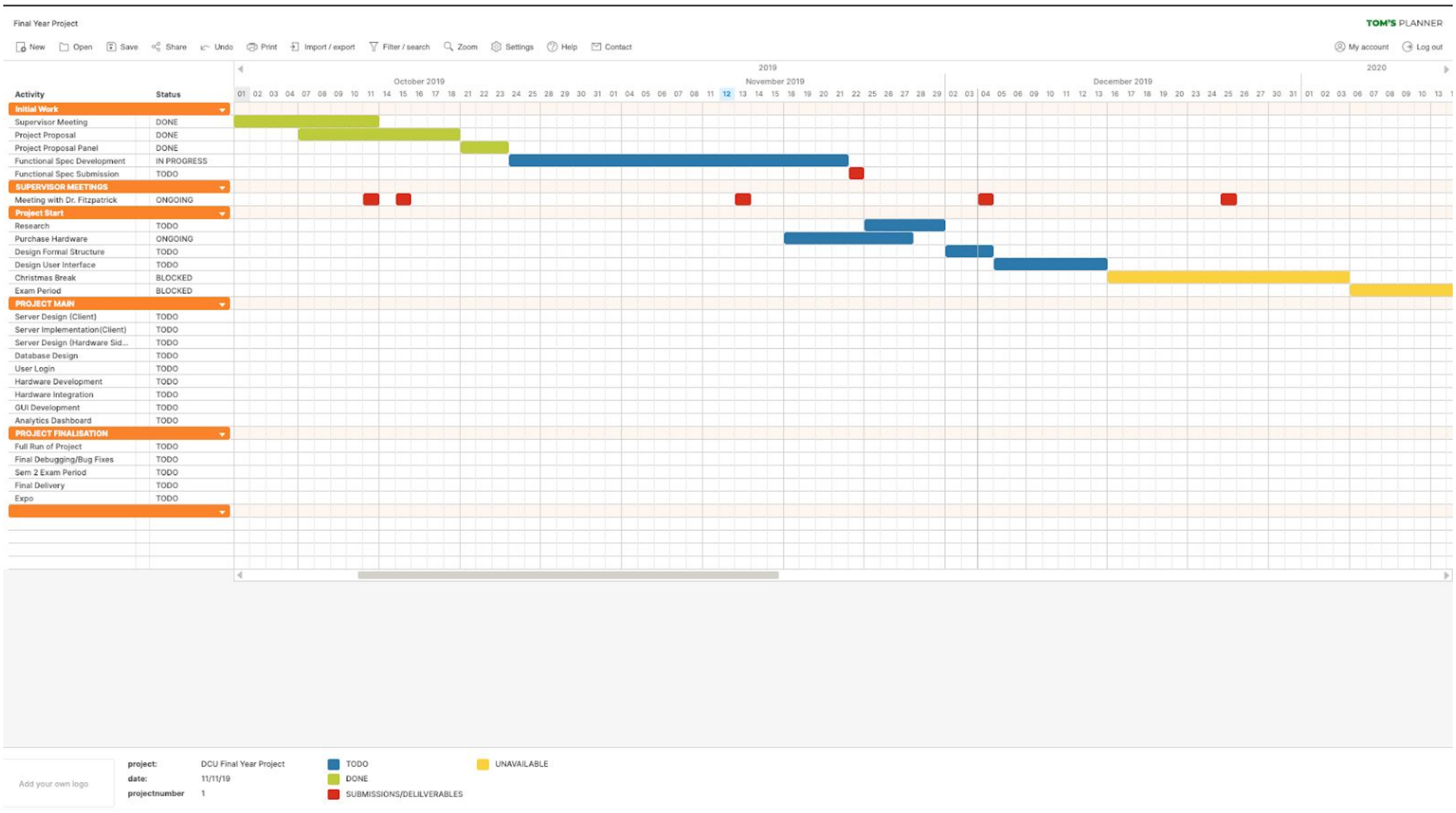
5.2
Activity Diagram


User
Log in
Check Plant Status/ View Historical Data/ Make Alterations
Historical Data VIew
Check Plant Status
Make Alterations to Environment
Database Read
Ping Pi
Alterations Made / Readings returned to User
Exit

6. Preliminary Schedule

Highlighted below is the timeline for our Project. This timeline is preliminary but we will be following it as strictly as possible following the Sprint methodology, treating each allocated time slot as a sprint.

6.1.1 Gantt Diagram: October-December

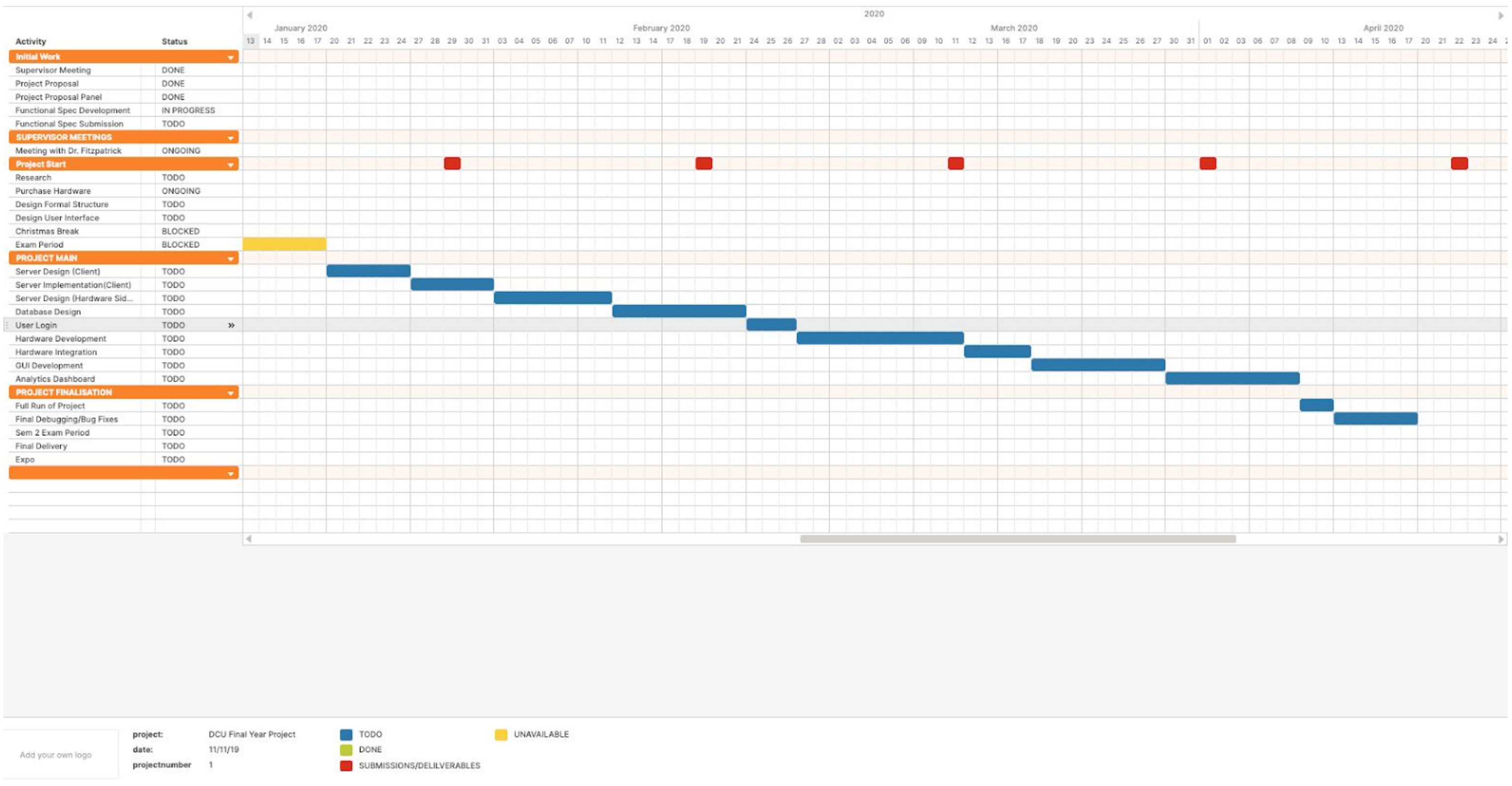


Activity	Status
Initial Work	
Supervisor Meeting	DONE
Project Proposal	DONE
Project Proposal Panel	DONE
Functional Spec Development	IN PROGRESS
Functional Spec Submission	TODO
SUPERVISOR MEETINGS	
Meeting with Dr. Fitzpatrick	ONGOING
Project Start	
Research	TODO
Purchase Hardware	ONGOING
Design Formal Structure	TODO
Design User Interface	TODO
Christmas Break	BLOCKED
Exam Period	BLOCKED
PROJECT MAIN	
Server Design (Client)	TODO
Server Implementation(Client)	TODO
Server Design (Hardware Sid...	TODO
Database Design	TODO
User Login	TODO
Hardware Development	TODO
Hardware Integration	TODO
GUI Development	TODO
Analytics Dashboard	TODO
PROJECT FINALISATION	
Full Run of Project	TODO
Final Debugging/Bug Fixes	TODO
Sem 2 Exam Period	TODO
Final Delivery	TODO
Expo	TODO

project:	DCU Final Year Project
date:	11/11/19
projectnumber	1

🟦 TODO
🟩 DONE
🟥 SUBMISSIONS/DELILVERABLES
🟨 UNAVAILABLE

6.1.2 Gantt Diagram: December-March

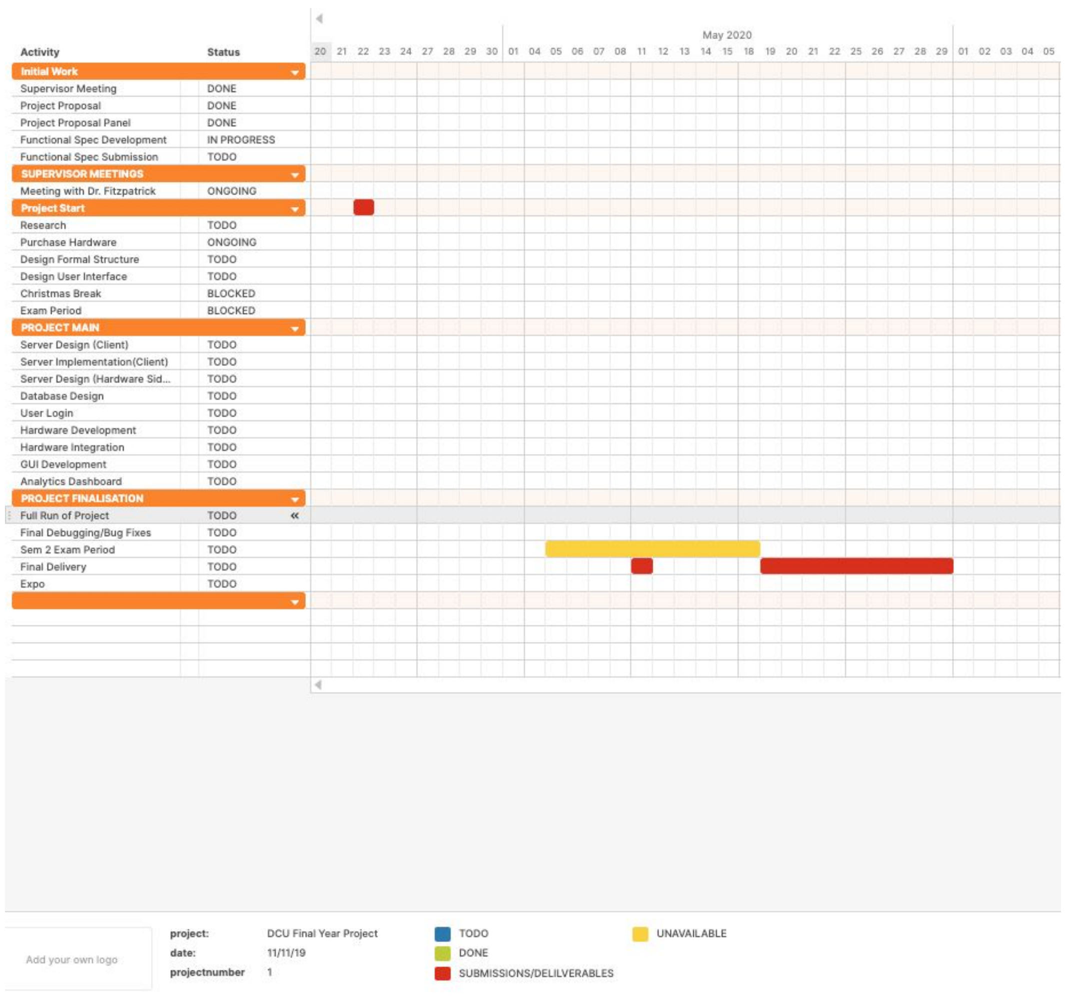


Activity	Status
Initial Work	
Supervisor Meeting	DONE
Project Proposal	DONE
Project Proposal Panel	DONE
Functional Spec Development	IN PROGRESS
Functional Spec Submission	TODO
SUPERVISOR MEETINGS	
Meeting with Dr. Fitzpatrick	ONGOING
Project Start	
Research	TODO
Purchase Hardware	ONGOING
Design Formal Structure	TODO
Design User Interface	TODO
Christmas Break	BLOCKED
Exam Period	BLOCKED
PROJECT MAIN	
Server Design (Client)	TODO
Server Implementation(Client)	TODO
Server Design (Hardware Sid...	TODO
Database Design	TODO
User Login	TODO
Hardware Development	TODO
Hardware Integration	TODO
GUI Development	TODO
Analytics Dashboard	TODO
PROJECT FINALISATION	
Full Run of Project	TODO
Final Debugging/Bug Fixes	TODO
Sem 2 Exam Period	TODO
Final Delivery	TODO
Expo	TODO

Add your own logo

project:	DCU Final Year Project	🟦 TODO	🟨 UNAVAILABLE
date:	11/11/19	🟩 DONE	
projectnumber	1	🟥 SUBMISSIONS/DELILVERABLES	

6.1.3 Gantt Diagram: March-May



Activity	Status
Initial Work	
Supervisor Meeting	DONE
Project Proposal	DONE
Project Proposal Panel	DONE
Functional Spec Development	IN PROGRESS
Functional Spec Submission	TODO
SUPERVISOR MEETINGS	
Meeting with Dr. Fitzpatrick	ONGOING
Project Start	
Research	TODO
Purchase Hardware	ONGOING
Design Formal Structure	TODO
Design User Interface	TODO
Christmas Break	BLOCKED
Exam Period	BLOCKED
PROJECT MAIN	
Server Design (Client)	TODO
Server Implementation(Client)	TODO
Server Design (Hardware Sid...	TODO
Database Design	TODO
User Login	TODO
Hardware Development	TODO
Hardware Integration	TODO
GUI Development	TODO
Analytics Dashboard	TODO
PROJECT FINALISATION	
Full Run of Project	TODO
Final Debugging/Bug Fixes	TODO
Sem 2 Exam Period	TODO
Final Delivery	TODO
Expo	TODO

Add your own logo

project:	DCU Final Year Project	🟦 TODO	🟨 UNAVAILABLE
date:	11/11/19	🟩 DONE	
projectnumber	1	🟥 SUBMISSIONS/DELILVERABLES	

7. Appendices

7.1 Appendices List

- https://developer.amazon.com/docs/custom-skills/steps-to-build-a-custom-skill.html - Alexa Custom Skill Tutorial
- https://docs.aws.amazon.com/lambda/index.html - Lambda Documentation
- https://docs.aws.amazon.com/cognito/index.html - Cognito Documentation
- https://github.com/boto/boto3 - AWS SDK for Python
- https://flask-doc.readthedocs.io/en/latest/ - Flask Documentation
- https://www.tomsplanner.com/ - Gantt Diagram Creating Application
- https://www.netguru.com/codestories/nginx-tutorial-basics-concepts - NGINX Tutorial
- https://www.python.org/ - Python Programming Language
- https://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html - Schniedermans 8 Golden Rules detailed