Cardiff Metropolitan University

A web-based e-micro farm platform for local agro-growers to effectively connect with consumers' vice-versa addressing the current issues in the agricultural domain and agro-produce foraging.

Submitted on Saturday, 12th June 2022

By

Judah Abshalom Charles Christopher

(Student ID - [st20215381] – [CL/BSCSD/22/44])

This dissertation is submitted in partial fulfillment

Of the requirements for the degree of

Bachelor of Science (Honours)

BSc (Hons) Software Engineering

DECLARATION

This work is being submitted in partial fulfillment of the requirements for the degree of BSc (Hons) Software Engineering and has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed – Judah Abshalom Charles Christopher (Candidate)

Date - 12/06/2022

SUPERVISOR'S DECLARATION STATEMENT

Student Name –

Supervisor's Name -

I acknowledge that the above named student has regularly attended the meeting, and actively engaged in the dissertation supervision process.

Signed - ……. (Supervisor)

Date - …….....

ACKNOWLEDGEMENT

First of all, words cannot express on how I'm thankful to God for guiding me up to this very moment of a successful completion of the dissertation. I'm profoundly thankful to my module lecturer Mr.Induneth De Silva for the commitment he has shown towards this module outcome in guiding us towards choosing a success worthy project. I also could not have come to this point without the guidance of the project supervisor Miss.Upeka Wijesinghe who generously provided feedbacks and support towards taking the right steps during the project timeline.

Additionally I'm also thankful to the academic board and management panel of the campus for providing infrastructural and technical support that impacted the outcome of this project. Last but not least, I am extremely grateful to my family, especially my parents who kept their belief in my academic path which helped to keep my motivation high to reach a successful end-goal in this project.

ABSTRACT

Day by day the challenges faced by local agro-growers and small scale producers are growing at an incremental rate. One of the key challenge they face is the inability to reach out to potential consumers. Currently there is no direct producer-consumer mechanism without the interference of middle-man sellers. This impacts the lives of the small scaled growers at various extents such as post-harvest wastage and unstable income. On the other hand this non-existence of a direct producer-consumer connection also impacts potential consumers who prefer organic fresh, healthy, local produce rather than chemically-altered and store-preserved agricultural products.

The need arises to intersect the challenges of both these parties to create a sustainable solution through which both the needs of consumers and producers could be met Recent advancements of technology has changed the paradigm of daily routines of individual citizens depend on everything to be digitalized shifting the way they consume, communicate and interact. Digital transformations have the potential to solve any problems as such.

The purpose of this project is to implement a web e-micro-farm foraging platform that serves as a centralized node that could connect local growers and potential consumers. Users can manage their agro-produces throughout this platform whereas potential consumers can look for produces around his/her neighborhood on an interactive map and make produce requests if needed. The system itself can serve as a platform to interact with each other in order to build community by at the same time. Ultimately the project intends to encourage local agriculture, making sure it doesn't go to waste

Keywords: *e-micro farm platform, geo-visualization, foraging web-application, producer-consumer interaction*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SDLC	Software Development Life Cycle
ASDP	Agile Software Development Process
OOP	Object Oriented Programming
UML	Unified-Modeling Language
SQL	Structured Query Language
API	Application Programming interface
JWT	JSON Web Token
IDE	Integrated Development Environment
CRUD	Acronym for create, read, update, delete
CSRF	Cross-Site Request forgery
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
JDBC	Java Database Connection
DAO	Data Access
JPA	Java Persistence API
RDBMS	Relational Database management System
MVC	Model-View-Controller

Chapter 1: Introduction

With the growth in technology, the communication trend has turned lives over a new leaf making its way on a connective method with potential increase in the speed of information transfer and secure the business transactions creating new commercial opportunities. Thus this innovation suggest a potential driving force for any sectors or domains towards growth and prosperity. (Dentzel, n.d.) Sri Lanka is a uniquely enriched fertile tropical land with a high diverse in climatic conditions and fairly distributed water sources which allows it to grow a range of crops all year-round. (Chadha, 2019) This suggests a massive potential for the nation to become a self-sufficient country growing diversified agriculture with fruits, vegetables and other agro-produces. This also suggests a well suited environment for agro-producers to carryout local production and on the other hand a benefitted environment for the general public to access fresh produces. Unfortunately in recent times, this expectation becomes unreal due to the growing concerns faced by both agro-growers and consumers. Growers find it difficult to reach out to potential consumers to sell or give away their excess whereas the goodness conscious-consumers find it difficult to reach out producers within their neighbourhood due to the lack of direct connection mechanism and the interference of middleman sellers. (Amir, 2021)

This projects main aim is to provide a web solution that could bridge the producers and consumers, (re)creating a sustainable economy for local growers to share their producers and a platform for consumers to easily access nutritious fresh grown nutritious produces and dominate the scope of local agriculture. Ultimately this will pave the way for the nation to make the best productive economic and social use out of it.

1.1 Background study

Over the past decade the patterns of population growth is increasing rapidly resulting a multifold increase in demands for food and crops. According to the report by United Nations related to world population prospects, it is quoted that Sri Lankan population is projected to surpass nearly 22 million within the next few years. (United Nations, 2019). This growth rate suggests enough opportunity for the agricultural industry to grow at a remarkable rate. But despite the higher demand and opportunities that could occur, it is evident

that there are still problems related to the parties involved in this sector. Furthermore as in recent times, the hike in the price of local-food products itself suggests that the local agricultural ecosystem is unstable. (Amir, 2021)

As a developing Asian country, Sri Lanka's economy highly depend on localized agriculture. Therefore any attempt to solve even minor problems in it contributes to the economic growth of the nation. (Jayathilake, Jayaweera and Waidyasekera, 2010) When local agricultures is given prominence and its dependency is prioritized, unwanted expenditures spent by the government on imports for the same agro-products that could be found locally could be avoided. (Jahufer, 2015) Ultimately there is a higher potential to contribute to the Gross-Domestic Product (GDP) which is the most vital measure of economy of any country as stated by Stobierski, 2021. Thus the citizens should be motivated to rely on local agriculture or may be atleast use their garden or unused land to locally grow local-grown produces because it is resilient that local agriculture dependency will be ought to good use in the post COVID-19 era.

The times where the world depended on agriculture is gone where in turn agriculture is now dependent on the world. Currently as per the mindset of modern citizens, the gravity and the importance of it has started to fade rapidly. According to the current market chain, those involved in local agro-producing such as small-scaled hardworking farmers whose lives run on agriculture are not given much importance and in fact they have no direct mechanism to reach out potential consumers who love to consume fresh food which ultimately is a contributory factor for the crisis. Thus in their point of view they are forced to sell their producers to the middleman. Middlemen are the individuals/organizations that specialize in marketing channels through which they can generate profit out of products given by small scale farmers who reach them. (Vidanapathirana,et al, 2018) According to the studies of analysts, their conclusions convey that if mechanisms exists to skip these middlemen-subsystems it would help the actual producers to retain their maximum profit and vice-versa would enable consumers to access fresh nutirtious produces they deserve at a nominal price. Therefore a system. Therefore a sustainable eco-system of services should exist to support both local producers/farmers and goodness conscious consumers.

1.2 Problem Statement

The most common way in which a grower can reach out to consumers is to showcase their produces in a public market or a dedicated economic center where the producer has to spend more money on fuel and still would be sometimes unable to sell what he produces. Therefore those involved or interested to involve in agro-growing have to rely on intermediate (middle-man) sellers to sell their produces. As per the study of Amir (2021), the intermediate sellers such as retailers, wholesalers, importers and exporters have become the key players of connecting with consumers to make sales. The major problem that arose with this context is that the actual producers or growers do not receive their deserving prices for their valuable harvest. This is currently the biggest issue as this had triggered many farmers/producers to give up on local production and focus on other jobs to earn more income, ultimately putting the agricultural sector at risk.

Another critical problem faced is the impact of middle-man products to consumers. The agro-products sold at stores undergo lengthy preservation procedures on cold storage containers with the help of artificial chemicals. For this purpose, especially the fruits and vegetables sold by the middle-man sellers are plucked even before nutrition has ripened in them. Later on they are induced to ripen and genetically modified with the use of carbo-chemicals when needed. Thereby consumers often tend to receive non-nutritious produces although it looks perfectly alright posing health related risks on the long run. Furthermore it is evident that these sellers imported agro-products that could be actually produced or found locally itself. "Food miles" is the term used to describe the distance that food has traveled. (Allen, n.d.) The more food miles accumulated during food transportation, the more fossil fuels and taxes are consumed, ultimately giving a burden to both country's economy and the environment.

Another growing concern as cited by the All Island Farmers' Federation (AIFF) is Post-harvest wastage. During the past months we have witnessed local-producers yielding surplus production and are unable to sell it all because of not being able to successfully reach out to potential consumers in such surplus cases. In peak seasons, lands were found to be overflowing with fruits and vegetables (like watermelons and pumpkins) that is left unconsumed ultimately found rotten on the ground. Yet just around the corner the same fruit/vegetable is sold at a supermarket with an expensive price labelled due to various taxes and fuel costs that people can't afford. (Chamara, 2019). This problem needs to be addressed because in desperate situations where the economic stability of the country is down, post-harvest wastage is not

something that could be tolerated as there are people and charitable organization currently desperate for food.

1.3 Proposed Objective

The main objective of the project is to build is a solution that could be a centralized node to connect local agro-growers with other growers or goodness conscious consumers' vice-versa making localized produce foraging easy addressing the issues they currently dace. It can be either small/medium scale farming or producing at home garden or private unused lands or even giving away. The users can share about their produces accommodating both buyable and free giveaway produces in case of excess. It will be a great way to find and share produces and build community all at the same time. The expected outcome of the application is to connect the parties without any involvement of middlemen paving a way to have a profitable eco-system. Consumers could also benefit with the choice that thy can make including form whom, where and at what price they would obtain their produces. In this way the solution will encourage local agriculture, making sure it doesn't go to waste and become a motivating factor for the people to do more.

1.4 Scope of the system

The scope of the project is to develop a web-system that is a central node between a producer and a consumer. The system is expected to curate mechanisms to map produces with location boundary matching algorithms in order to aid a potential consumer to look for produces within a neighborhood. The system is not intended to be biased on any of the stakeholders therefore a producer can be a consumer too and a consumer could be a producer too making them as a primary actor. Furthermore the system is expected to support the social connection aspect by providing social networking features on the platform whereby buyers and growers can add each other as 'friends' and interact. Since no middlemen interference is expected to be avoided, functionality is to be curated so that a consumer can make a produce(s) request through the system and pick up the agro products directly from the involved grower/producer and make all payments in terms of funds/produce exchange straight to them ensuring that they retain a stable income obtained from the harvest they produce.

CHAPTER 2: Literature Review

The goal of this literature study is to provide an overview of discussions related to the different aspects of theories/conclusions derived on the domain scope as well as the review on the technologies involved in the similar system development. The areas of context investigated are discussed under different sub-segments such as the consumers' interest on fresh agro-produces, need of consumer-producer connection, E-farming as a solution for local producers, current e-marketing systems, the need of interactive community and comparison of architectures and technologies of web application.

2.1 Consumers interest on fresh agro-produces

In the agro-domain, it is evident that a potential consumer consider an array of product features that is classified intrinsic. In-fact they are reported to frequently associate locally-grown products with greater freshness, reduced spoilage, and increased safety due to the shorter distance travelled to the markets. (Bruce & Som Castellano, 2017; Byker et al., 2012; Webber & Dollahite, 2008) According to an internal survey based-research on consumers conducted by Trobe (2008) many consumers reported that they would like to buy organic fresh produce more often. Moreover about 68% of them cited that they do not buy fresh produces because of the prices labelled on them by super markets. So they mostly go for chemically engineered alternatives which are produced in a short time. Moreover during the research they interviewed some actual farmers and found out that the amount they were paid with was steeply low than what the supermarket sells the same product for. Hence this interference and unnecessary chain of Intermediate person at several stages of the business caused to unnecessary price increases in the consumer markets (Trobe, 2008)

As per the study of (Klopcic, Slokan and Erjavec, 2020) they found out that almost every consumers more likely to emphasize personal health, environment and nutrition when choosing food products. These health and moral concerns are part of their basic human rights. Further the internal research by Trobe (2008) proves this fact because 80% of the consumers responded positively when they were asked if they were actually concerned on how their food is produced. When reasons for this were asked, the most important consumer considerations were the use

of chemicals in the agro produce and the concern over human health. (Klopcic, Slokan and Erjavec, 2020)

2.2 Need of digital consumer and producer connection

Webber and Dollahite's (2008) research highlighted the necessity of relation-building desired between producers and consumers which of course consumes effort from both parties. According to Humphreys and Grayson (2008), this connection is defined as "a recursive loop through which each party gives value to one another". The interference of third-parties (middle-men) in-between may either favour a single party or may disrupt both parties. (Berti & Mulligan, 2016; Perrett & Jackson, 2015). This effect was researched by Abishek et al. (2016). Their critical experiments demonstrated the difference in prices with and without the interference of middle men. Statistically their study proved that a buyer can save a minimum of 10% when purchased directly and also the producer can retain their actual price for their produce.

In the agricultural-marketing domain, success is defined by 3 main values transparency, equity and access. Transparency allows consumers to learn about the production methods, quality and traceability. Equity provides fair wages for small-scale producers while also providing products at reasonable prices to consumers and thus making it accessible to low income people. Access is all about making the produce reach consumers in an orderly manner maintaining a short supply chain. Therefore the producer-consumer relation should embrace this values in order to be beneficial to all. (Berti & Mulligan, 2016).

Setia et al. (2013) argued that the digital technology evolution has shaped up the way we communicate work and consume. Hence the researcher emphasized the need of a digital collaboration in the producer-consumer pattern because it has the potential to connect consumers with a large number of growers as a result of aggregation in a beneficial manner. In the research by Abishek et al. (2016), they proposed a web and mobile application to act as the connection node between the parties where marketing of agricultural products could be effective. The researchers believed that the application can encourage farmers to do more agriculture as they can make sure they get the right labor for their produces.

A Swedish web application Local Food Nodes (2017) was introduced to connect local producers to local food consumers to empower the relationships among them over food. The

application enabled direct transactions and resilient communities. The producers can simply post their food and local consumers can order what they like and payments go straight from consumer to producer ensuring that the producers retain 100% of all revenues from the products that they produce. Deliveries and pick up of food takes place at a predetermined place and time, this is called a node. This is the physical location where the consumers and producers meet. Multiple consumers at the same time while consumers can pick up food from many different producers.

2.2 E-farming systems as a solution to local producers

Pramod et al. (2016) presented a research under the topic of "E-farming solution for local producers". According to the author, agriculture is a prime economic factor for any country. But people who involved in farming are suffering from poverty. The main reason behind the poverty is that after all hard work and sacrifices to their production farmers are unable to get a good market for their harvest as they cheated by the agent. They proposed an E-Farming good market platform for farmers to sell their products across the country and to guide them with current market rates of different products, the total sale and the earned profit for the sold products and access to the new farming techniques through eLearning. With this in practice they cited that most of the economic crisis at the country faces will be changed inversely in no time. (Pramod et al., 2016)

Karunaratne and Vidanagama (2015) proposed a web information system to support the local farmers of Sri Lanka. The system was proposed to have discussion forums where the users can share ideas and comments to support others such as hazardous news, agricultural tips, and awareness of crop diseases. The system maintained a centralized database system through which the users such as farmers, agrarian officers and coordinators respectively.

2.2 Current e-marketing systems related to agro-products

Seed Code Labs (2019) introduced a mobile application namely "Weladapola". The applications main goal was to provide a digital market place to buy and sell fresh local agro products. Moreover the application had a functionality to provide related information necessary

for the agro-producers to market their products such as price indexes of commodities. (Seed Code Labs, 2019)

A cross platform application was proposed by Croptronix (2019) namely "Govipola" to replace the chaotic physical markets. The platforms main goal is a digital market for the farming community where buyers and sellers were able to come together transparently procuring fresh agro commodities. The platform followed an advertisement based listing mechanism where the producers can simply post advertisements to sell their produces. Buyers can simply navigate through the listings according to the needs in order to make a purchase. (Croptronix, 2019)

2.3 Need of an interactive community

Smith et al (2008) argued that both producers and consumers need a medium to share their opinions and foster relationship among them. Ghogare and Monga, (2015) proved that proper communication mechanisms existing in the rural agricultural domain could result in an improvisation of the domain.

These facts reflects the conclusion of an internal research by Opitz et al.,(2019). They studied that consumer-producer community interaction plays a significant role in the economic stability of agriculture. It is because it could be seen as ways through which consumers empower the producer economically. This factor of interpersonal interaction was studied by Bodin, Crona and Ernston (2006) as a correspondent to a knowledge building mechanism. Knowledge can built when ideas and problems are shared on top of responses of others. This is applicable to the producer-consumer community too. (Bodin, Crona and Ernston, 2006; Opitz et al., 2019)

2.4 Architectures of web applications

Bora and Bezboruah. (2015) conducted a study that compared the implementation of RESTful web services to SOAP-based web services. They used a mercury load runner testing tool. They implemented similar functional applications only with the difference of it being REST and SOAP. Both applications were implemented using Java running on the Apache tomcat server and MySQL database server in order to keep other factors constant such as language performance. The findings of this investigation revealed that both REST and SOAP based services are reliable and scalable. But when further analyzed statistically, REST services had a faster response time and higher throughput. (Bora and Bezboruah, 2015)

According to another study by Anshu and Virender (2019), they critically concluded web services. They found out that although both performed well, RESTful services were much faster in handling requests. This was similar to the results obtained by Anil Dudhe et. Al, 2014. They created SOAP and REST web services with jersey and performed tests on Apache Tomcat and Android emulators. Their results proved that REST could be an alternative to SOAP when cross platform ability and scalability is considered. (Ranga and Soni, 2019)

2.5 Comparison of core-technologies in web applications

According to a study by Hamed and Kafri, (2009) on evaluating the performance of Java and .Net applications, they conducted tests using Mercury Load runner 8.0 tool. They found out that .Net application was performing on a stable manner. But in order to understand performances on a three-tiered MVC applications, they critically evaluated similar MVC applications built on Java and .Net in terms of memory utilization and response time. Thus they performed load testing on the login scenario on both the applications using Parasoft-WebKing and HP LoadRunner tools. Conclusively Java EE benchmarked its victory in terms of performance. (Hamed and Kafri, 2009)

Munonye and Martinek, (2018) put forward a performance analysis testing concept using Apache JMeter to test API implemented using MS.NET using entity framework with MSSQL server and Java EE Jersey API using JDBC with MYSQL. They tested the performance with similar HTTP GET request. Conclusively their findings showed that the GET operations perform better with Java and MySQL. (Munonye and Martinek, 2018)

CHAPTER 3: Software Development Process

3.1 Development process models

A software development process model defines how the tasks in the project is processed in an order of sequence. (Sulemani, 2021) In the path of achieving a successful software project, it is crucial to analyze the different available models available by clearly distinguishing it's workflow in-order to select the most suitable model to work with as it plays a major impact on the end-outcome. The most important SDLC models are:

- **Waterfall model:**

  It is a plan-driven process where each phases are dependent on the completion of the previous tasks. The sequential phases include requirement analysis, designing, implementation, testing and maintenance. In this way the requirements should be very clear before even starting to build the software. As a dependent-driven structure it is inflexible to accommodate changes in between and in fact there is no way to test or use the software until the last phase. (Sulemani, 2021)
- **Iterative model:**

  This model splits the software into small fragments that are repetitively built throughout the lifecycle. These fragments are small steps of a large requirement. Each fragment has a designing, development, testing and implementation phases. This aids to review results at early stages of development which can be used to fix functional and design flaws making it a more flexible model than the waterfall model. (Elgabry, 2017)
- **Spiral model:**

  It is a combination of the waterfall and the iterative model. The four main phases, identification, designing, building and evaluation are repeated on a cyclic pattern until the project is completed where each cyclic loop is a phase. It is basically a risk-reduction model in which the risks are assessed at each phase inordeer to reduce the chances of project failures. (Elgabry, 2017)
- **V-model:**

  It is an extended waterfall model where the cycle takes an upward movement of phase in the form of testing and validation Requirements gathered initially cannot change and for each phase there is testing phase and thus known as a as a verification model. (Sulemani, 2021)

- **Agile:**

  The agile approach is a standardized and well-accepted SDLC methodology universally. This method breaks the overall process into small incremental segments where each segment has requirement gathering, designing, developing and testing phase. Hence it is a combined model of iterative and incremental process. With this approach, requirement and solutions evolve concurrently with collaboration. At each iterations features are incremented and the final phase delivers a completed fully functional product. (Brush, 2019)

3.1 Selected process model for the proposed system

The timeline for the entire project is relatively less therefore the cost of delay is high. Moreover an effective delivery should be ensured within this timeframe accommodating all the requirement and functional changes. Hence it is required to go with a concurrent approach where the requirement and the solution is mapped in collaboration. The most suitable methodology is a model that is a combination of incremental and iterative approach. Considering all the process models the life cycle requirement could be attained using the the **Agile Software Development process (ASDP).**

3.2 How ASDP is approached in this project

The commonly used agile methodology is the Agile Scrum. A scrum breaks down each agile **iterations as sprints**. Each of these sprints are between 2-4 weeks long. In this project, the overall objective is broken to these sprints as given in **table 1** below to focus on highly refined prototype delivery in order to complete the project within the given time frame emphasizing the agile approach. Each sprint have a planning, developing and testing phases to deliver a refined prototype.

From an agile perspective, an early release helps to improve the design quality, which means testing for early defect detection and elimination. Because at each iterations features are added or removed and even before the next iteration begins, errors are analyzed and is fixed. (Synopsys, n.d.) At an iteration end, the working prototype is reviewed. Prototype faults are focusingly fixed while speeding up the delivery of the end product. In this way a quickly

delivery of a working product is ensured. As the overall project is broken into individual manageable units, each segmented unit is more likely navigated towards high-quality review, development and testing in conjunction. Quality of the end-product is refined with effective solutions for the defects found in iteration reviews.

Sprint #	Objective/task
Sprint 1	Sprint requirement follow-up
	Designing
	Develop User/ Agro Produce related core modules
	Integrate as Prototype 1
	Testing and Bug fixing
	Reviewing prototype
Sprint 2	Sprint requirement follow-up
	Designing
	Develop interaction/other modules
	Integrate as Prototype 2
	Testing and Bug fixing
	Reviewing prototype
Sprint 3	Sprint requirement follow-up
	Designing
	Develop other modules/extra features
	Integrate as Prototype 3
	Testing and Bug fixing
	Reviewing prototype

Table 1 Agile sprint delivery plan

Chapter 4: Analysis and requirements

4.1 Requirement Gathering

Requirement gathering is the most crucial step in this project because the process involved helps to determine the scope and expected deliverables of the project. These processes are used to be analyzed in deep in-order to gain insights on incorporating functional and non-functional requirements of the system to be developed. Among the several types of requirement gathering techniques, the following techniques were used in this project.

- **Similar system comparison** :

  By analyzing and critically evaluating similar existing systems related in the same domain, it is easy to brighten the ideology of the proposed system. Systems such as Govipola, Agribros and Local food nodes were selected for critical comparison.

- **Interviews:**

  Interviewing aids to felicitate requirements where we can map the system expectations as per the user requirement.

- **Questionnaires**:

  Questionnaires aids to understand what the end-users/public opinion and expectations with regards to the system. A set of questionnaires are circulating questionnaires to the public who are expected to be future users of the system.

The overview of the requirement gathering steps followed and its expected outcomes is as given in **table 1** below.

Technique	Steps/Content	Outcome
Similar system comparison	Analyze the application in terms of provided functionalities and the use of themes etc.	Idea on the required application structure and functionalities
Questionnaires	Survey containing multiple-choice questions compromising both open and close ended questions	Qualitative and quantitative data collection.

Interview	A formal conversation with a potential consumer and an agro-producer with the aid of an interview guide.	Conclude with system end-users expectations

4.1.1 Similar system comparison

Govipola

Govipola is a platform developed by crop-tronix with a main goal of a local digital market for the farming community where buyers and sellers were able to come together procuring fresh agro commodities. The color-schema used in the application is centered on a greenery scale to emphasize a plant based eco-environment. The core functionalities of the system were,

- User registration and login
- Post about the agro-products.
- View other posted products.

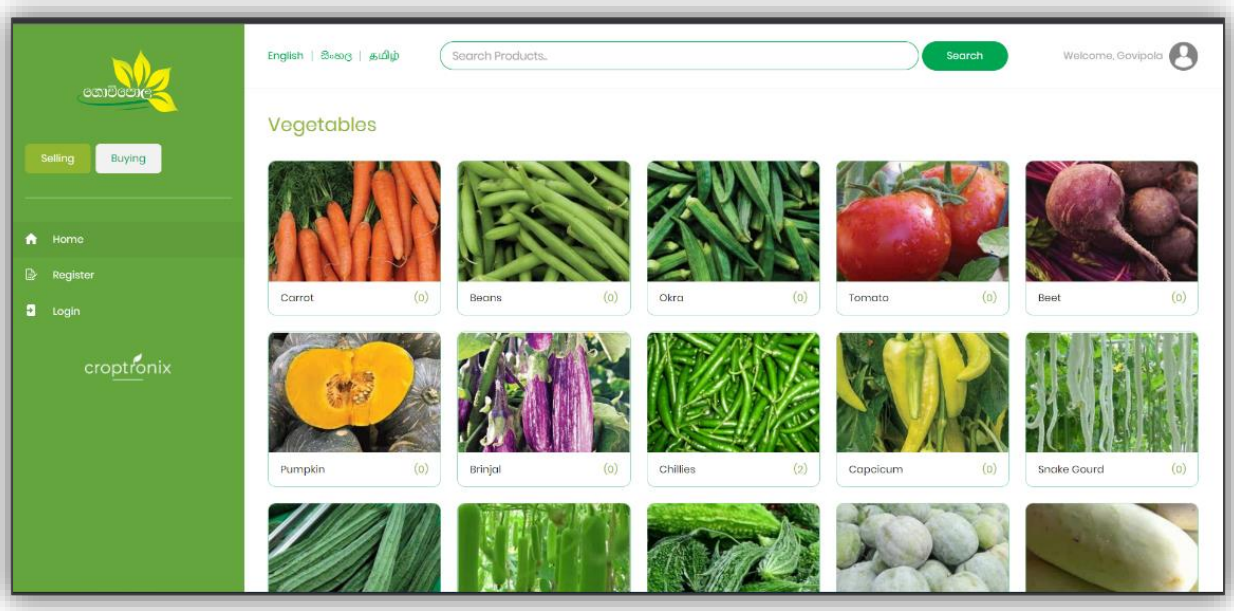


Figure 1 Govipola UI – Source (www.govipola.lk)

Users are also able to post their advertisements to sell their produces by filling in the relevant details. A consumer can go through the produces a produce is searched or navigated through a category, users are able to can navigate through products based on an advertisement based listing mechanism in **figure 2.**

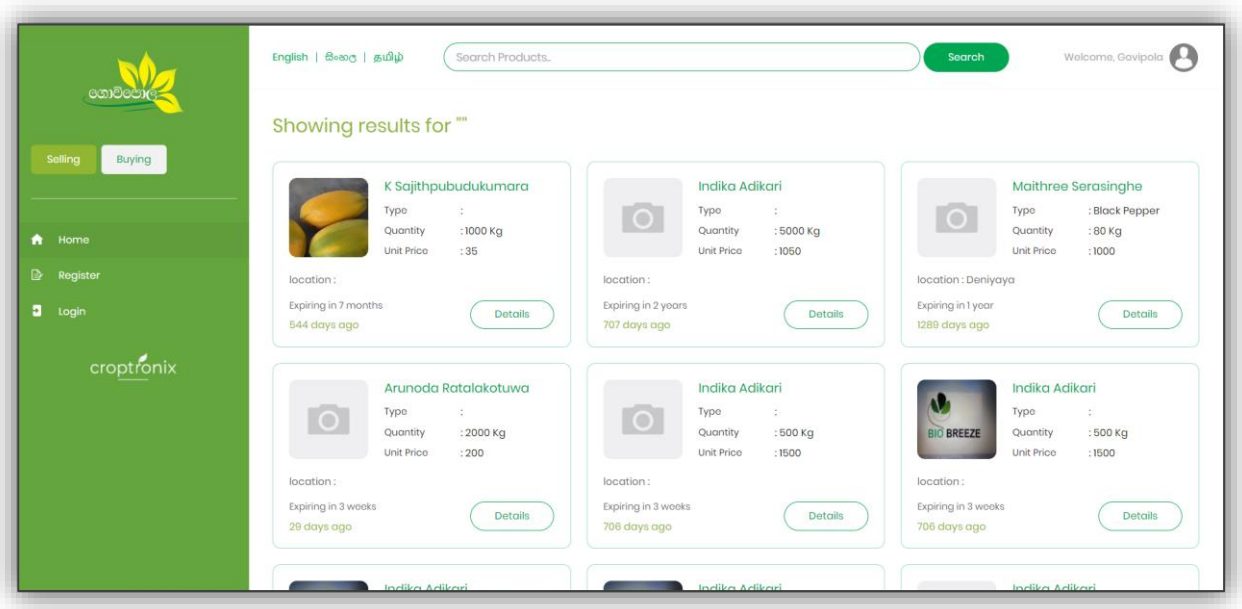


Figure 2 Govipola Product listing – Source (www.govipola.lk)

Agribros

The Agribros market is a digital eb-platform where consumers are able to view fresh agro commodities and make a request to the producer/grower. The color scheme is biased on a greenery scale to emphasize agricultural pleasantness. Users can simply navigate through available products along with the product information and pricing.

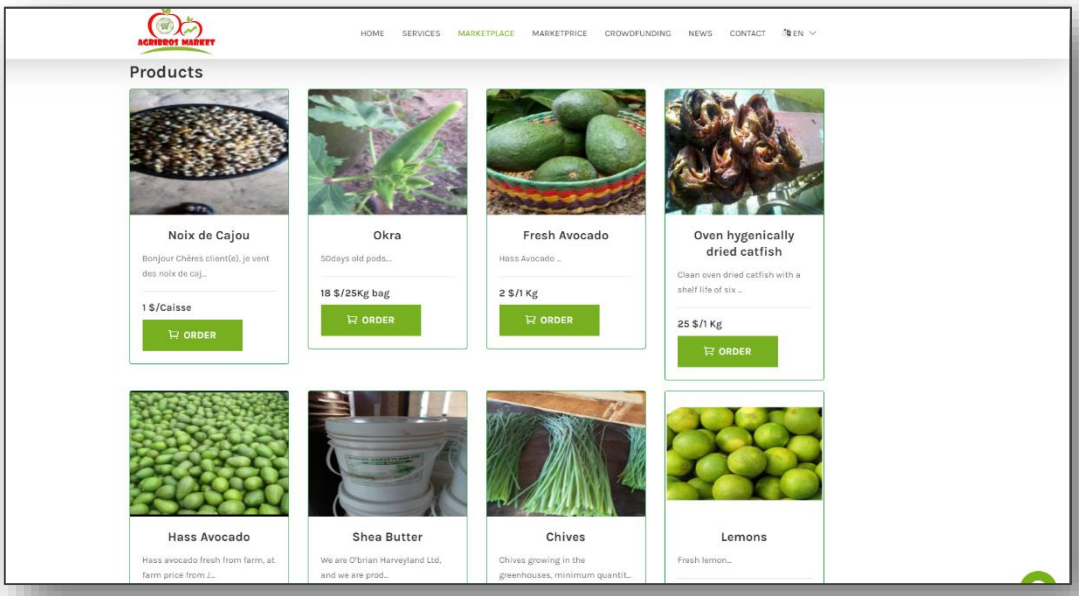


Figure 3 Agribros UI – Source (www.agribros.market)

Inorder to place an order users(consumers) undergo an unreliable proceeding through which they have to fill out a form as below with necessary details and quantity needed which will be sent as a request via email to the product provider.

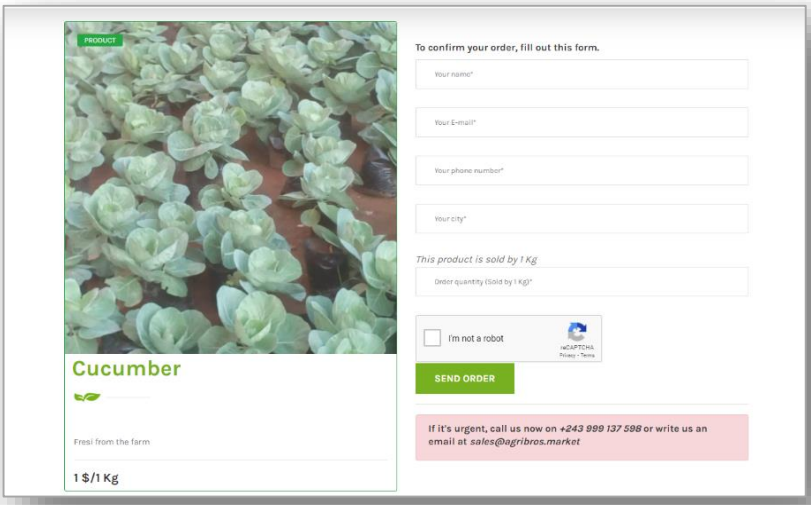


Figure 4 Agribros -Placing order – Source (www.agribros.market)

Agfuse

Agfuse is a community platform specifically for farmers and producers to connect with one another to share information and interact only. Any user can register themselves by providing their area of interest. Once registered users can either add their own post or view posts made by other users, and they can interact with it by commenting or up-voting it. Moreover users canstay connected with other users with a friendship-based functionality similar to many social media sites.

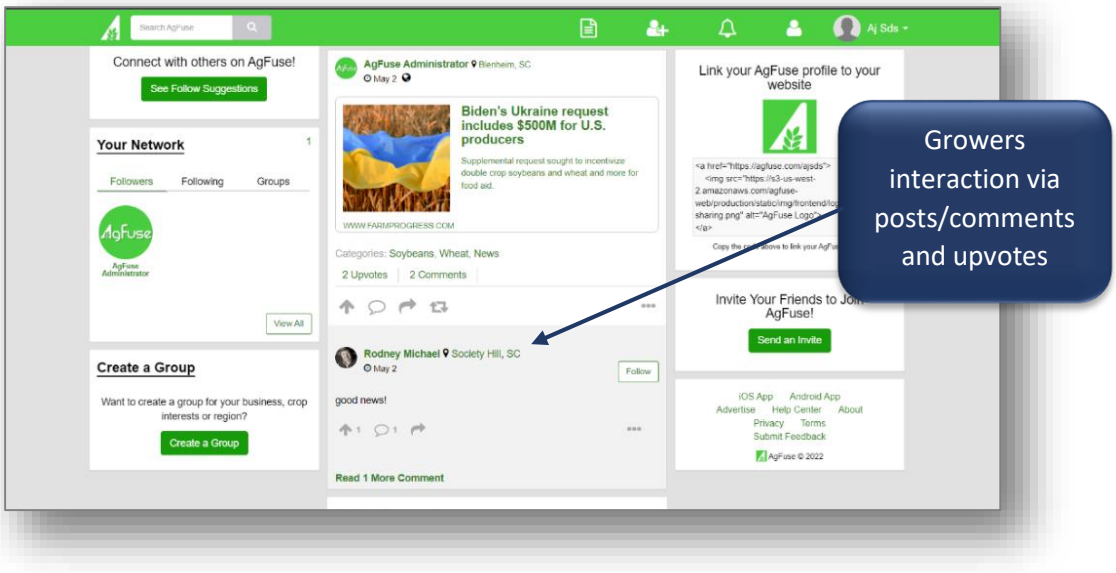


Figure 5 Agfuse – Source (www.agfuse.com)

Local Food nodes

Local Food is a pure-Scandinavian platform for the Swedish producers to show case their producers where consumers can find them. After registration, users are able to view producers or nodes (predetermined place where produce is exchanged) on **a map with the help of markers** placed based on geo data as below. The use of markers in the maps makes it easier for potential consumers to easily locate producers nearby and stay connected to what they grow etc.

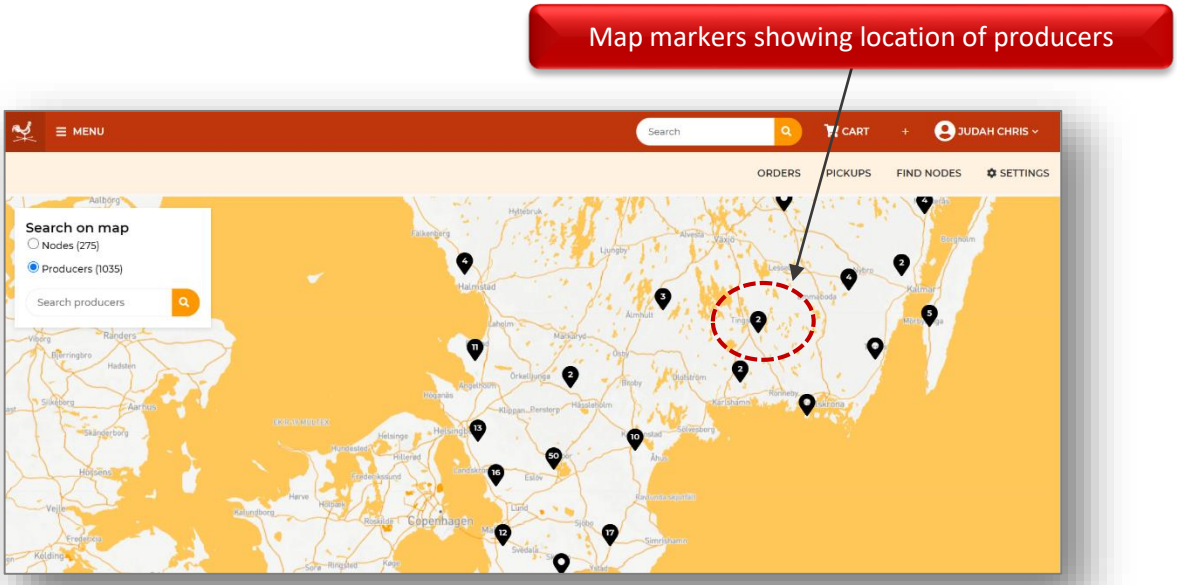


Figure 6 Local Food Nodes – Source (www.localfoodnodes.org)

Any produce-requests made via this application goes straight the actual producer where he may choose to accept or decline the request. Any registered user can be producer and a consumer vice-versa which. However limitations of this application is that a consumer can only place a request, if they pay a yearly usage fee only. In overall terms of navigation and usability, the system is too technical for an ordinary user lacking technical knowledge.

4.1.3 Questionnaires

Gathering information from the future users of the system is very crucial as it aids to deliver insights based on the user's expectation and necessities. A questionnaire type survey with a set of questions made using google sheets was circulated to a group of people in order to receive

their opinions. Using the answers provided, a good understanding of the situation and how to improve the system to meet the user requirements are mapped.

Refer the original questionnaire in **appendix-A**

- **<u>Quantitative-data</u>**

  Inorder to gather quantitative data for the research purposes, close ended questions with pre-defined multiple choices as answers were used as given below. Complete responses received are included in the appendix.

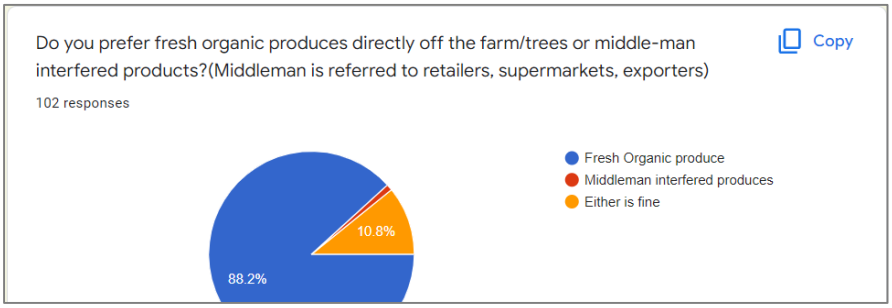


Figure 7 Questionnaire 1

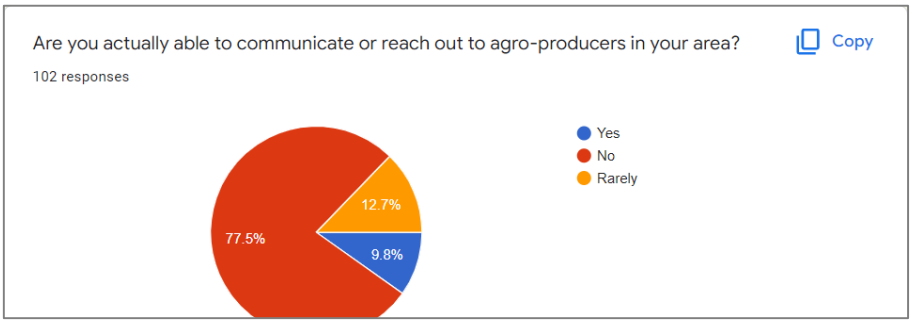


Figure 8 Questionnaire 2

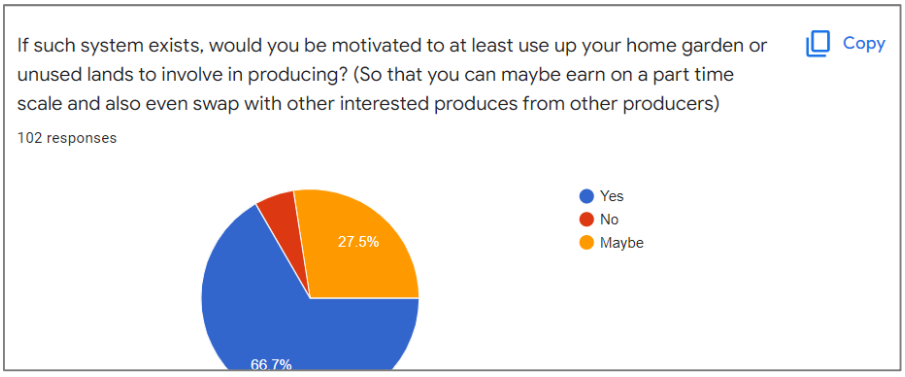


Figure 9 Questionnaire 3

- **Qualitative-data**

The survey also included open-ended questions through which the participants can share their point of expectations with regards to the system. Since they were direct opinions, all their ideas were analyzed as considerations. Some of the responses are given below whereas the complete responses are attached in the **appendix-A**

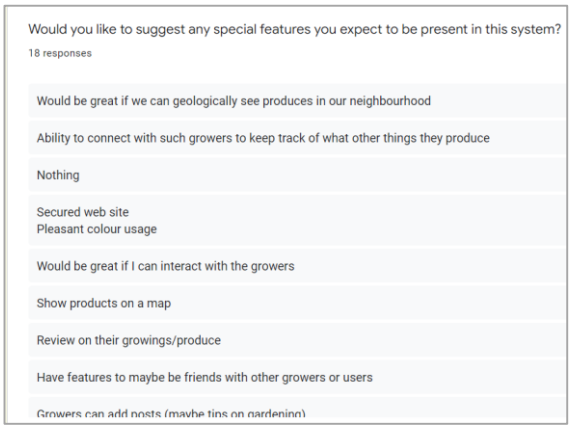


Figure 11 Questionnaire-Open-ended 4

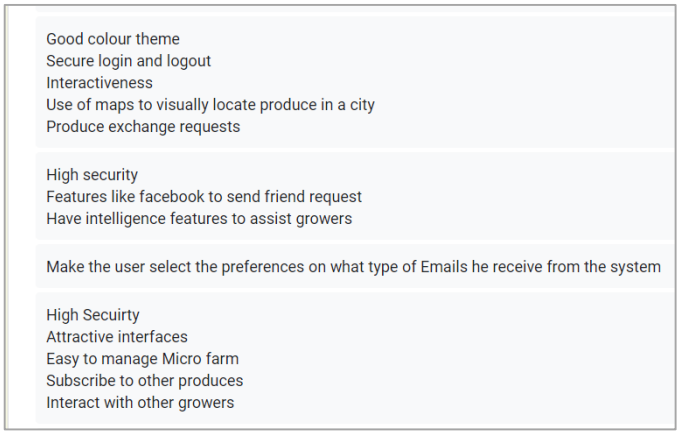


Figure 10 Questionnaire-Open-ended 5

Figure 12 Questionnaire-Open-ended 6

Summary of the questionnaire

A total of **102** responses were collected from the public at the end of this questionnaire method. This method was supportive to get an overview of a large number of audience who is expected to be benefitted by system to be implemented. Their current-issue and expectations were well studied. Conclusively the results suggest that majority of people prefer fresh agro-produces and that they are unaware of actual producers in their community and that **it is hard to reach them**. Moreover many agreed that there is no system in Sri Lanka under the proposed scope. It is also evident that out of the sample population majority voted that they may/maybe interested in agro-growing by using their home garden or unused lands. This validates that the extreme

need of the system. Through the open ended section of survey, it was easy to collect the list of functionalities that needs to be included in the system.

4.1.3 Interview

A structured conversation in the form of an interview was done with related-stakeholder in order to get their inputs in relation to the proposed system. Inorder to avoid possible-biasness potential consumer and a small-scale agro-grower/producer both were interviewed. A pre-structured interview guide as given below in **figure 13 and 14** was given to both parties a day before the session so that they were able to prepare the answers they were to provide.

Interview Guide – Agro-producer/small scale farmer

1. How long have you been involved in producing?
2. What type of produces do you produce?
3. Is this your main source of income?
4. Do you find it easy to reach out to consumers? What is your current process of reaching them?
5. Are you able to sell all your produces easily?
6. If a web solution is to be developed, through which other people around you can easily find out your produces, what are your expectations with this system?
7. Suppose if there is an option to notify people about your excess produce would you agree to give it for a lower rate?
8. What is your overall opinion with this system, do you think it would help you?

Figure 13 Interview-guide (Agro-producer)

Interview Guide – general consumer

1. Are you interested in fresh agro products that is not chemically engineered or altered by any third parties?
2. Do you find it easy to reach out to fresh agro-producers in your area? How do you try to reach them?

   Give a brief introduction of the proposed system
3. Do you think this system would be feasible in the community?
4. Would you be happy to stay connected to fellow growers/producers within your area?
5. As a general consumer what would you like to see in this system? What are your general expectations?

Figure 14 Interview-guide (Potential Consumer)

4.1.4 Overall conclusion on requirement gathering

During the analysis phase the information gathered from the various techniques were compared and combined to critically analyze the system requirements. With similar system comparison, a clear research gap was formulated on how the current related-system which aided to map some functionalities that could be combined in the proposed system to make it a problem solving alternative in the system market. Also system attributes such as color-usage, response times were critically evaluated so that adequate applications of such attributes shall be used in the proposed system.

Based on the responses shared by the public through questionnaires and interview, their expectations/ideas were mapped to draft the required functionalities to aid system designing. A majority of the responses demonstrated the current drawbacks they face and the need of this system to be implemented. A major requirement that was required by them were to have an interaction module through which a consumer and a producer could stay in touch so that a consumer may keep track of what the producer is growing and when the produces will be available for pickup. A majority of respondents of the questionnaire survey also suggested to use mapping features in the system so that location of the produce and the producer can be interacted with, which may be helpful when searching nearby produces in the neighborhood.

4.2 System requirements

4.2.1 Functional requirements

- **User module :**
  All users should be able to easily register to the system to access the functionalities with secured token based login/logout handled.
- **Producer management module** :
  Users should be able to post what they produce and be able to manage all their produces in the *e-micro* farm. Moreover users should also be able manage the status of their produce
- **Finding produces :**
  Users should be able to find locally grown-produces around a specific neighborhood using filtering options. Furthermore, should be able to leave a review on another users produces.
- **Geological visualization :**

  Consumers should be able to visualize what produces are available or growing in a specific neighborhood equipped with required details with the help of geological map plugins.

- **Subscribe/request produce :**

  Potential consumers should be able to subscribe to a produce to keep track of it and make a direct produce request to the producer if needed.

- **Posts feed :**

  Users should be able to be interconnected via posts that maybe provide interactive opportunity to help others or share resources. (Such as effective growing tips or advices related to plant diseases.

- **Friends Module :**
  Users should be able to send friend requests to other users in order to stay connected so that connective node is established between potential consumers and producers. The receiver should be able to accept or decline the request.
- **Email feature**
  Users should be notified with important notifications through the system.

4.2.2 Non-functional requirements

- **Performance :**

  The system shall give quick responses to the end user at all times. The initial load duration of all the interfaces should be relatively low. Moreover data fetched from the database shall be available with high accuracy accommodating minimal delays as possible.

- **Reliability :**

  Reliability is vital to any system. When executing the system functionality, it should work smoothly with minimal failures as possible. High accuracy shall be maintained within data transfers with proper validations. In addition, the system should be tested using several testing techniques to reduce the possible set of the failures.

- **Usability :**

  The system shall be user friendly, where the user interfaces have an easy-to-understructure with proper use of suitable thematic colors which doesn't cause any strains on long term usage.

- **Security :**

  Security is one of the major concern when it comes to the systems which deals with sensitive data of users. System should have token based mechanisms to outlast the requests made from unauthorized users. Moreover sensitive credential information such as user password shall be hashed with effective algorithms before stored onto the database.

- **Maintainability :**

  The system shall be developed using standardized coding conventions in order to increase efficiency when updating functionalities, bug fixes and maintenance. Moreover OOP concepts with modularization should be used throughout effectively to reduce boilerplate coding and coupling.

- **Browser compatibility :**

  The system shall be made operative on most of the standard browsers such as Microsoft Edge, Internet Explorer and Google Chrome.

4.4 Resource requirement and allocation

4.4.1 Software Requirement

- **IDE**: Visual Studio Code, IntelliJ
- **Programming/scripting language**: Java & Angular (HTML/CSS/Typescript)
- **Database**: MySQL
- **VCS:** GitHub
- **API-testing**: Postman
- **Browser**: Google Chrome

4.4.2 Hardware requirement

Hardware Requirement	**Minimum requirement**	**Recommended requirement**
Processor	1.9 gigahertz(GHz) x86-bit or x64 bit dual core processor with 2 GB RAM	3.3 gigahertz (GHz) or faster 64-bit dual core processor with 4GB RAM
Display	Super VGA with a resolution of 1024x768	Super VGA with a resolution of 1024 x 768

Table 2 Hardware requirement

4.5 Feasibility study

4.5.1 Introduction

Feasibility study on a project plays a crucial part during the analysis phase of a project. In simple words with this it is evaluated whether the proposed web system is viable to be completed using the available resources within the given time frame. This study can assess the worthiness of proceeding by evaluating the benefits and the possible drawbacks that could occur before investing time, energy and money in it. The following section of the study is

demonstrated with different perceptions of the feasibility study of the project. (Simplilearn, n.d.)

4.5.2 Technical Feasibility

The proposed solution is a completely dynamic web based solution. The proposed architecture of the solution is to have a client/server-end through which the client consumes REST API exposed by the server-end. The core technologies/frameworks involved in the project such as Java, Spring Boot, HTML, CSS, Typescript, Angular and MySQL are readily accessible open sourced tech and moreover these technologies have an industrial standard **proven to be matured** with high stability. They are capable to serve the expected functionalities of the proposed-system. The technical skills needed to use these technologies are manageable individually as a result of familiarity. The solution development time constraints and the ease of applying these technologies are capable to be synchronized within the given project timeline thus ensuring scalability of the platform technologically.

Moreover the IDE tools such as IntelliJ and Visual Studio Code are required to be used in the project development and these tools have free community editions that can be readily used for development and debugging. These facts validates the resources needed to fulfill the technical requirements of the overall project and confirms that the project is technically feasible enough.

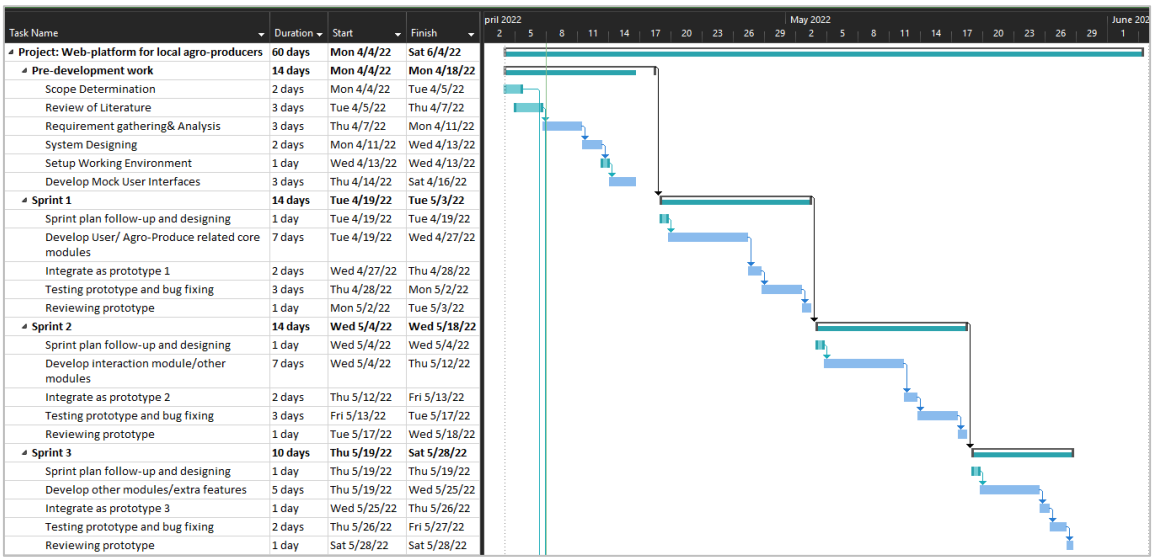
4.5.3 Operational Feasibility

The identified problem domain that the solution solves is something that the society would appreciate, if addressed. It is connecting communities and will become a motivating factor to involve in small scale agro producing. It is because the solution is not intended to be biased on any of the stakeholders because a producer can be a consumer too and a consumer could be a producer too. So if operational this will be contributing factor in all aspects such as economically, socially and environmentally.

Moreover with the growth of the modern era, the world is shifting to a digital paradigm where users depend on everything to be digitalized. Therefore this transformation to digital platform will make it easy for the end users to adapt. These facts validates that the proposed solution will work more than the fact that the system can just work proving the fact that it is operationally feasible within the scope of the user environment.

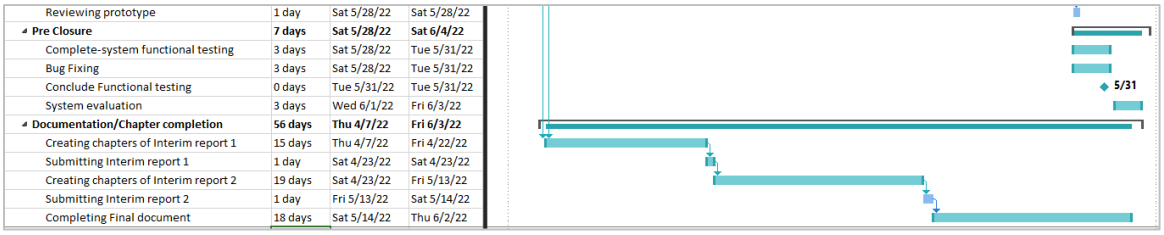
4.5.4 Schedule feasibility

Agile is the chosen methodology to be used in the project. The overall-project is broken to consecutive sprints to focus on prototype releases to complete a viable solution in the given timeline. The expected schedule to design and implement the solution is as below.



Task Name	Duration	Start	Finish
Project: Web-platform for local agro-producers	**60 days**	**Mon 4/4/22**	**Sat 6/4/22**
Pre-development work	**14 days**	**Mon 4/4/22**	**Mon 4/18/22**
Scope Determination	2 days	Mon 4/4/22	Tue 4/5/22
Review of Literature	3 days	Tue 4/5/22	Thu 4/7/22
Requirement gathering& Analysis	3 days	Thu 4/7/22	Mon 4/11/22
System Designing	2 days	Mon 4/11/22	Wed 4/13/22
Setup Working Environment	1 day	Wed 4/13/22	Wed 4/13/22
Develop Mock User Interfaces	3 days	Thu 4/14/22	Sat 4/16/22
Sprint 1	**14 days**	**Tue 4/19/22**	**Tue 5/3/22**
Sprint plan follow-up and designing	1 day	Tue 4/19/22	Tue 4/19/22
Develop User/ Agro-Produce related core modules	7 days	Tue 4/19/22	Wed 4/27/22
Integrate as prototype 1	2 days	Wed 4/27/22	Thu 4/28/22
Testing prototype and bug fixing	3 days	Thu 4/28/22	Mon 5/2/22
Reviewing prototype	1 day	Mon 5/2/22	Tue 5/3/22
Sprint 2	**14 days**	**Wed 5/4/22**	**Wed 5/18/22**
Sprint plan follow-up and designing	1 day	Wed 5/4/22	Wed 5/4/22
Develop interaction module/other modules	7 days	Wed 5/4/22	Thu 5/12/22
Integrate as prototype 2	2 days	Thu 5/12/22	Fri 5/13/22
Testing prototype and bug fixing	3 days	Fri 5/13/22	Tue 5/17/22
Reviewing prototype	1 day	Tue 5/17/22	Wed 5/18/22
Sprint 3	**10 days**	**Thu 5/19/22**	**Sat 5/28/22**
Sprint plan follow-up and designing	1 day	Thu 5/19/22	Thu 5/19/22
Develop other modules/extra features	5 days	Thu 5/19/22	Wed 5/25/22
Integrate as prototype 3	1 day	Wed 5/25/22	Thu 5/26/22
Testing prototype and bug fixing	2 days	Thu 5/26/22	Fri 5/27/22
Reviewing prototype	1 day	Sat 5/28/22	Sat 5/28/22

Figure 16 Gantt chart 1



Reviewing prototype	1 day	Sat 5/28/22	Sat 5/28/22
Pre Closure	**7 days**	**Sat 5/28/22**	**Sat 6/4/22**
Complete-system functional testing	3 days	Sat 5/28/22	Tue 5/31/22
Bug Fixing	3 days	Sat 5/28/22	Tue 5/31/22
Conclude Functional testing	0 days	Tue 5/31/22	Tue 5/31/22
System evaluation	3 days	Wed 6/1/22	Fri 6/3/22
Documentation/Chapter completion	**56 days**	**Thu 4/7/22**	**Fri 6/3/22**
Creating chapters of Interim report 1	15 days	Thu 4/7/22	Fri 4/22/22
Submitting Interim report 1	1 day	Sat 4/23/22	Sat 4/23/22
Creating chapters of Interim report 2	19 days	Sat 4/23/22	Fri 5/13/22
Submitting Interim report 2	1 day	Fri 5/13/22	Sat 5/14/22
Completing Final document	18 days	Sat 5/14/22	Thu 6/2/22

Figure 15 Gantt chart 2

After considering the estimated schedule, it is projected that the allocated-timeframe for SDLC phases such requirement gathering, designing, implementation and testing is potential enough to complete the scope of the project to meet the project deliveries at the given completion date schedules. Therefore the project can be completed within the given time constraint **making a schedule feasible project.**

4.5.6 Economic feasibility

In terms of development, the costs involved is very low. It is because all the environment and the components used in the project are freeware. A detailed description of the related resources required to complete the project and the costs associated with it is given below.

Resource	**Cost**
Visual Studio Code	Free
IntelliJ	Free (Community edition)
Github	Free
Postman API	Free
Mailtrap	Free
Apache Tomcat Server	Free
MySQL	Free
Geocoding/Reverse geocoding (Google API)	1,800 LKR (5 USD)
Wi-Fi network charge	2,000 LKR
Total	**3,800 LKR**

Since all the technologies are open sourced, the end bearable cost on development perspective of the overall project is relatively low. This cost is totally affordable making this project a financially viable project.

4.5.5 Legal feasibility

The development process of the solution is expected to comply all the copyright regulations without damaging any intellectual properties. Moreover the technologies and the libraries that are ought to be used in the project are ensured to be open sourced. Therefore all these facts proves that the project is legally feasible as it brings no violation to any legislative law and regulations.

4.6 Risk management

4.6.1 Risk analysis

A risk is a probable uncertain event that when it happens it may bring about an effect on at least one of the targeted objectives of the project. Thus it should be analyzed mitigation steps should be taken to avoid the most probable risks. Few of the major categories of risks that could occur within this project are,

Technical risks:

- Lack or unavailability of the necessary hardware or software resources to complete the project.
- Hardware malfunction and cyber-malware attacks resulting in loss of project files.

Project management risks:

- Forecasted schedules are subject to change whereas the expected duration to complete an objective may become less due to some factors.
- As the overall project duration is less, any delay to complete an objective may affect the overall project delivery.

Financial risks:

- The forecasted cost on software, hardware and other resources may subject to change ultimately affecting the project budget plan.

External risks:

- Government laws and regulations may change any time such as allocated power cut hours where it may affect the delivery of objectives.

Natural disasters:

- COVID-19 spread is still unpredicted and is likely to become severe at any time. At such severe conditions, the government may enforce travel restrictions which may affect the project quality as the supervision cannot be held physically.
- Other disasters such as thunderstorm, heavy rain, lightning and floods etc. may affect the completion of the project.

4.6.2 Risk assessment

A qualitative risk analysis is done based on the identified risks in contrasts with the risk-matrix. Since avoiding all risks are barely possible, risks are prioritized in-order to ensure scalability of the project.

Risk ID	Risk	Impact	Resolution/Mitigation
R01	Loss of project files and data due to hardware malfunctions	High	Critically evaluate the hardware resources and always maintain updated backups in a version control repository (GitHub)
R03	Malware attacks	Medium High	Use proper internet security to safeguard.
R04	Risks with the software's technologies chosen for the development not able to handle the workload to complete the project	Medium High	Carefully choose industrial standard and stable versions of technologies.
R05	Risks by requirements that are not properly defined	High	Carefully map functions based on requirement gathering techniques
R06	Conflict between commitment towards project and other personal daily activities	Medium High	Forecasting schedule and allocate suitable time for effective development.
R07	Risk of not able to complete the deliverable objective	High	Follow the agile principles to concurrently deliver the objectives
R08	Expected schedules or deadlines changing	Low Medium	Complete the intended tasks on time

R09	Risk of having power cuts	High	Allocate time to work during non-power cut hours.
R10	Expected costs may increase	Low Medium	Thoroughly evaluate the tools and technology used to find about the restrictions and pricing.

Table 3 Risk Assessment

Chapter 5: System Design

5.1 High-level architecture

The high-level architecture of the project is based on a 3-tiered architecture with a presentation layer, application layer and data layer distinctly. This typed-architecture has its benefits in the development process. It makes it possible to make changes on the technology stack of one layer without impacting the other areas of the application vice-versa. Furthermore, it provides scalability to the project.

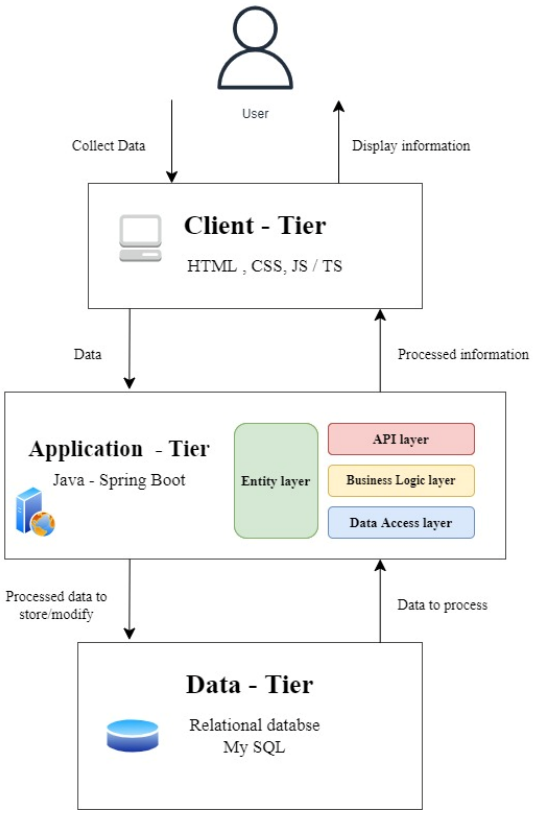


Figure 17 High-level architecture

5.2 Database diagram

5.2.1 ER diagram

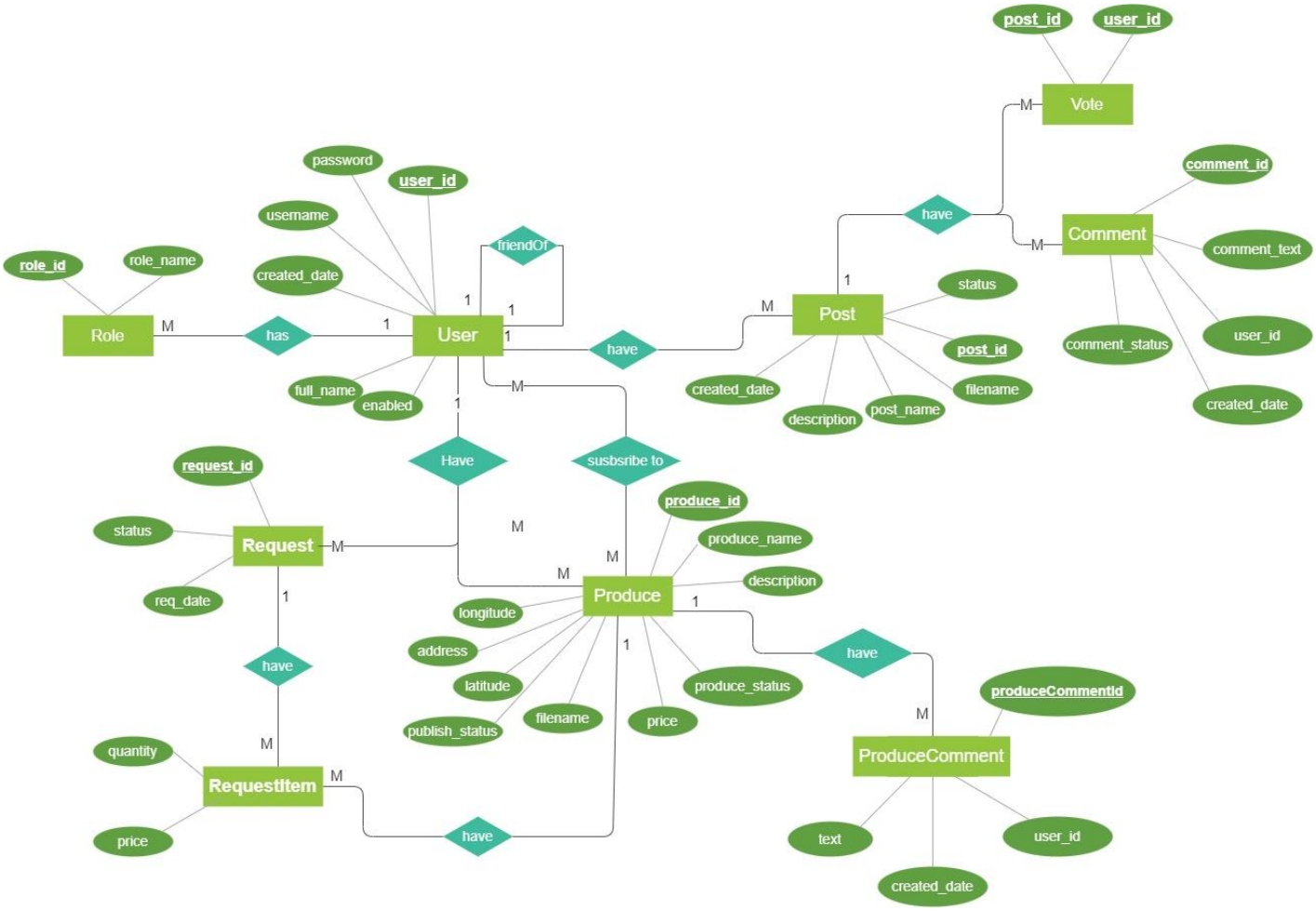


Figure 18 ER-Diagram

Diagram rationale:

- A user is considered to maintain one or many produces whereas a user can also subscribe to many produces which are managed by fellow users.
- User can make/have a request with one or many request items which corresponds to particular produces.
- A user can be a friend of another fellow user showing a recursive relationship.
- Since a community interaction is a requirement in the system, a user can have/post many posts which in turn could be interacted through comments and votes.

5.2 UML Diagrams

5.3.1 Behavioral diagram

5.3.1.1 Use case

The following use-case diagram describes the system high level core functionalities and captures the interaction of the expected actor in handling the system.

Design rationale:

- In this system both producers and consumers are generalized as a user because the concept is that a consumer can involve in producing and vice-versa a producer could be a consumer of another product too.
- The core aspect of the system is to geographically visualize produces around on an interactive map, thus Find-produce use case **includes** visualizing geographically use case.
- A potential consumer is able to request a produce through which the grower receives the request in order to be accepted or declined.

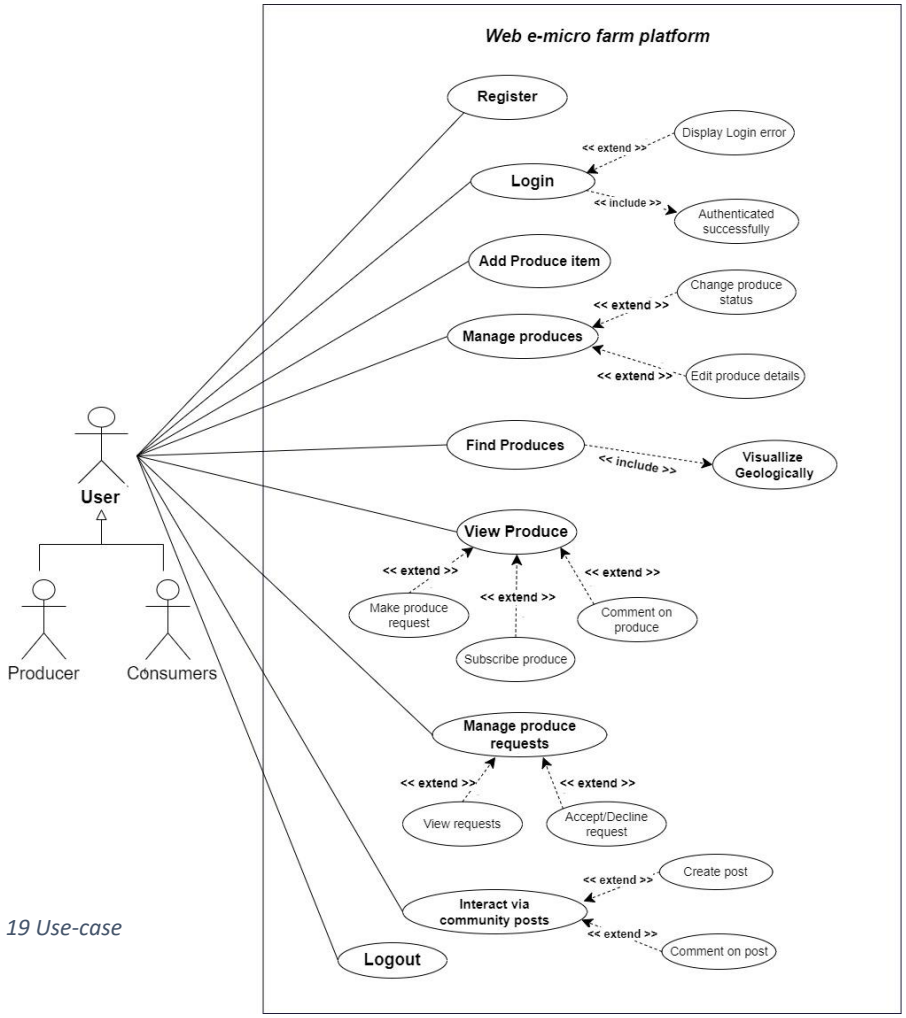


Figure 19 Use-case

5.3.1.3 Sequence diagram

The interaction among the objects of the system and the exchange of messages over time is visualized in the following section. For a better perception of describing the interaction and readability of the program flow, separate sequence diagrams only for the core use-cases are demonstrated below.

❖ **Registration-sequence**

Once a user enters all the required details for registration the system checks **whether the user already exists** in the db. Two alternatives could result where one being a successful registration of user if user not exists or an unsuccessful sequence if exists. After successful generation sending generated email is an Asynchronous flow in order to provide a good user experience.

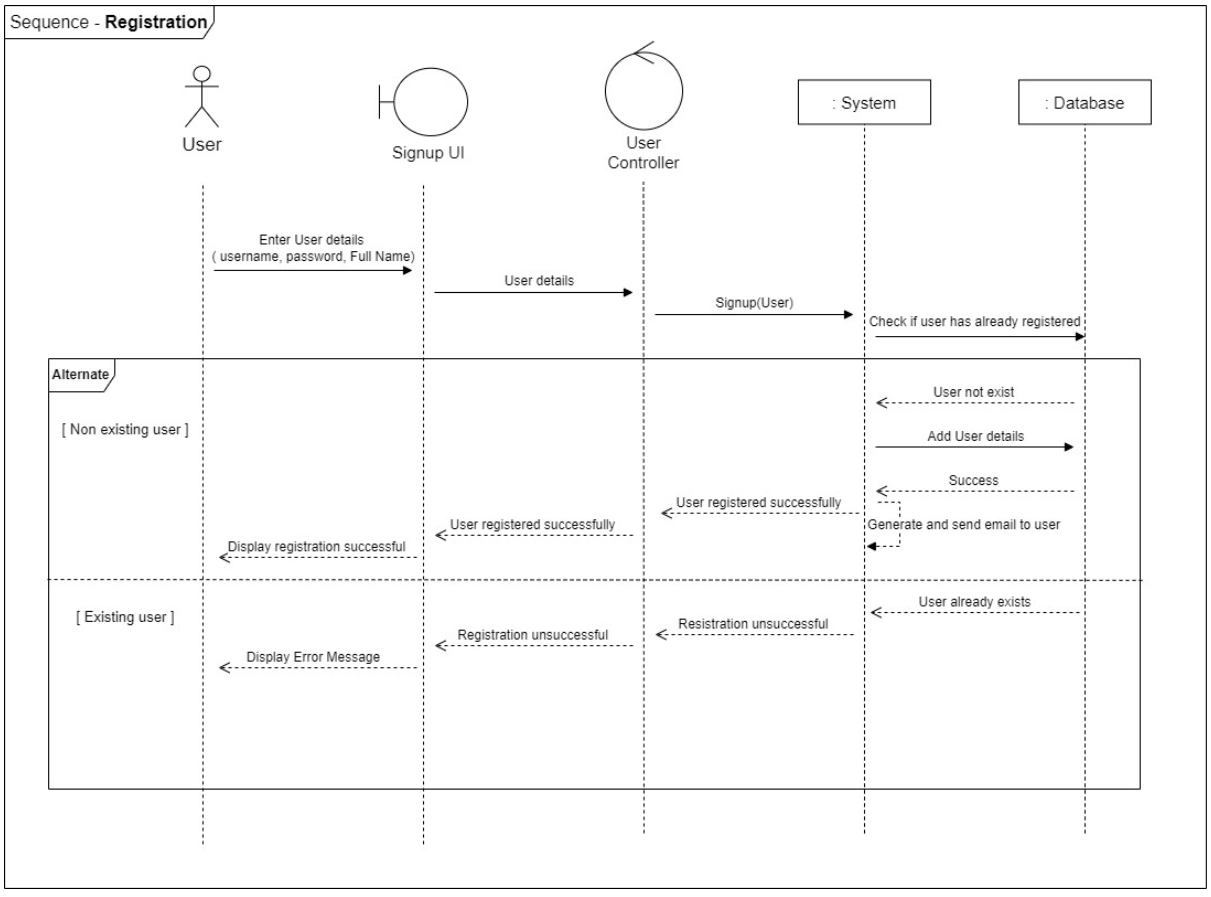


Figure 20 Registration-Sequence

❖ Login-sequence

The entered user credentials is conveyed to the system which cross-checks the database for authentication. Followed by authentication, two alternatives could result where one being a token generation if the user is authenticated or else error message passed if not.

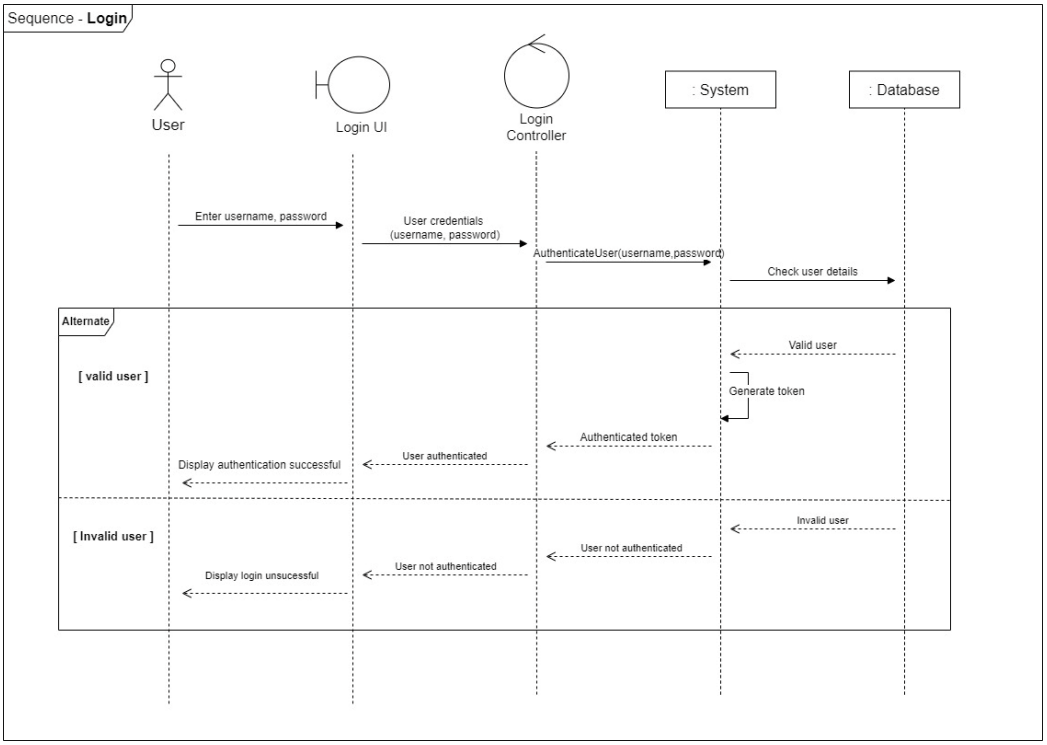


Figure 21 Login-Sequence

❖ Produce-Management-sequence

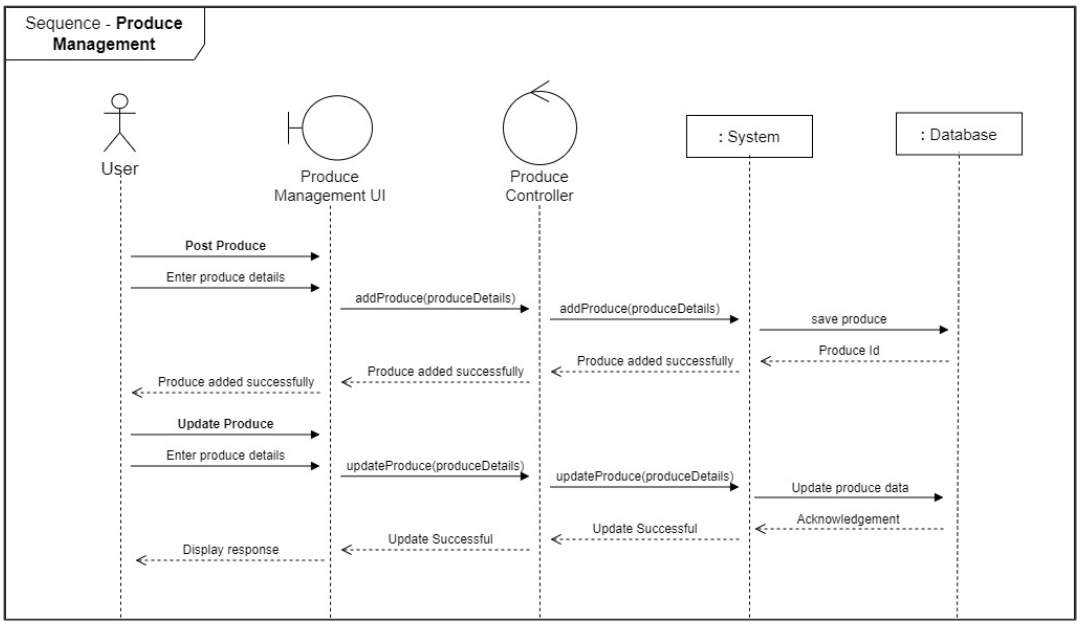


Figure 22 Produce-Management Sequence

❖ View/Search produces-sequence

Viewing and Search is treated as a common sequence because they both display the retrieved results from the database. When fetching produce data from the database the user is able to search by filters. The system fetches the data from the database and performs a synchronous method to generate **GeoJson** data mandatorily as visualizing on a map is a core requirement of the system.

While viewing a produce, the user is provided with two options either to subscribe to the particular produce or to make a request to the produce.

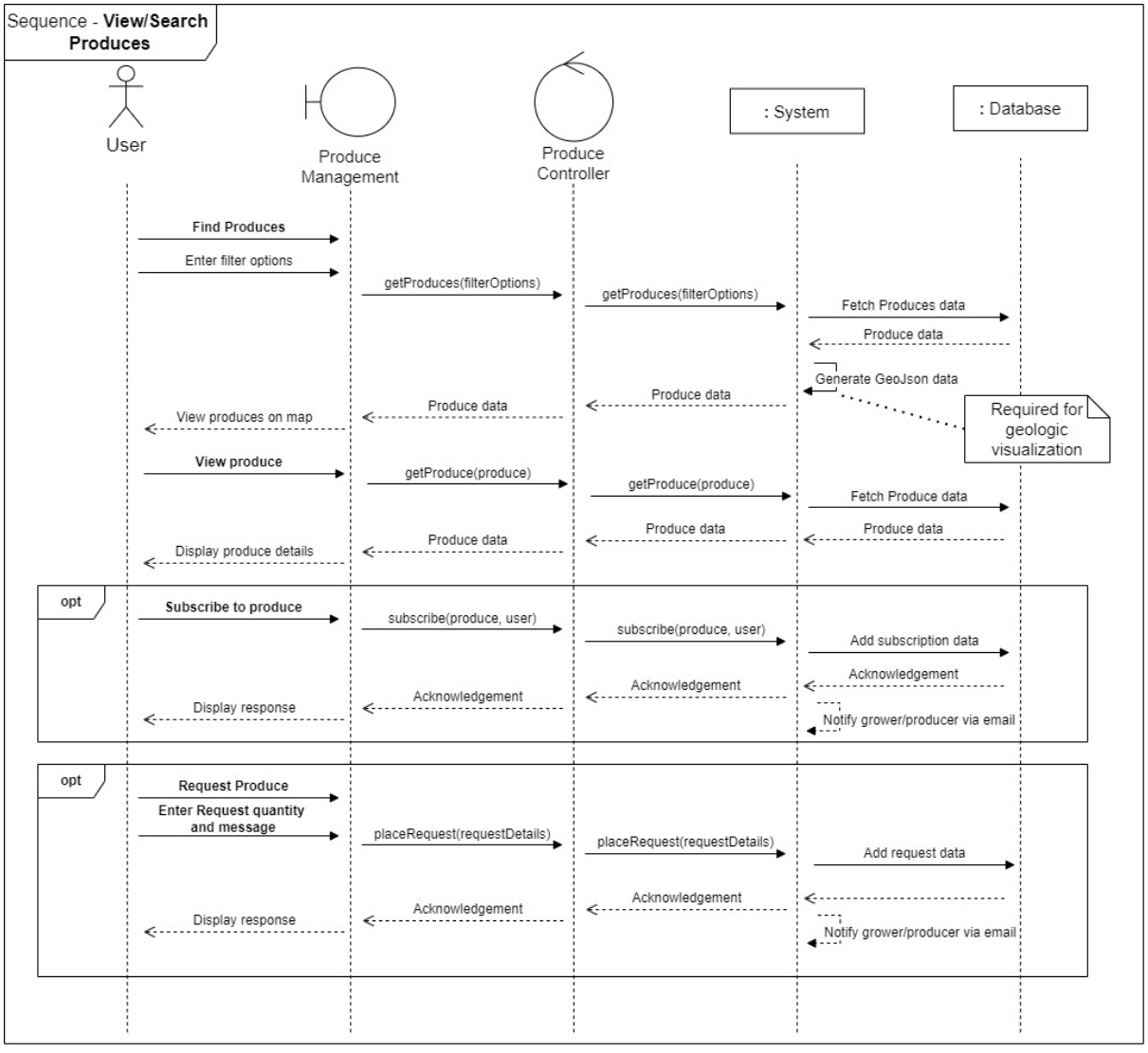


Figure 23 View/Search-Produces Sequence

5.3.2 Structural diagram

5.3.2.1 Class diagram

The following diagram is designed to describe the static view of the application. It shows the objects that resides in the system and the type of interactivity among them. Each class is described with the structural features (attributes) and behavioral features (methods).

Design rationale:

- A user can manage multiple produces whereas a produce belongs to only a user.
- An **enum** class 'ProduceStatus' is used by the produce class and 'RequestStatus' by the request class for constant management of error-prone status.
- A request could be made by a user which may have one or many request items which in turn refers to a corresponding single product. Furthermore a 'requestIem' may belong to only a particular 'request'
- The friend class shows a composition association with users, because logically a friend needs users to exist. If the user class does not exist it may be disrupted.
- Each user may not or may submit many posts while a post must be exactly submitted by a user. A post may contain a collection of comments while a comment is just associated to one instance of the post class.

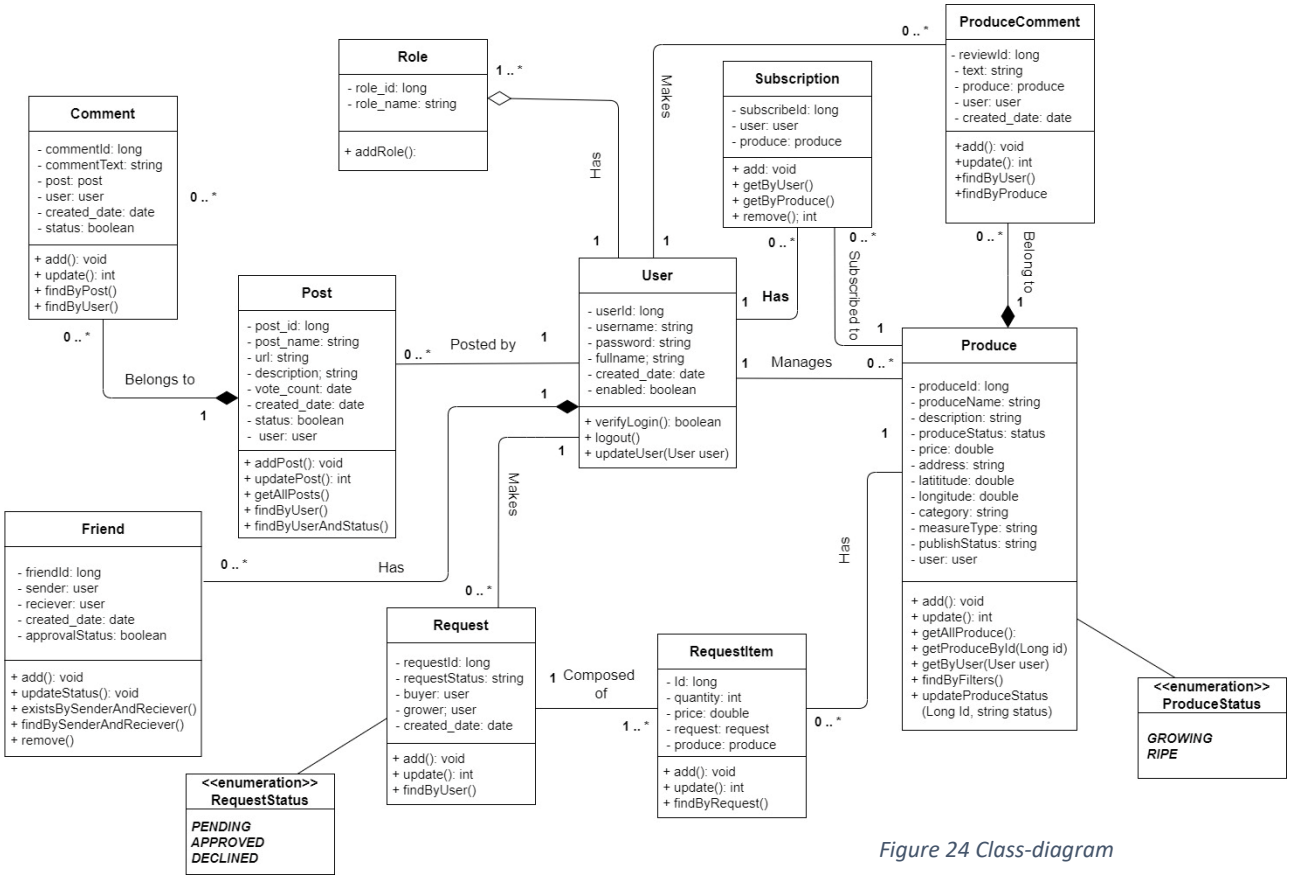


Figure 24 Class-diagram

5.3.2.2 Deployment diagram

The below diagram shows the expected deployment diagram of the entire application. Since the basic architecture chosen for the project is 3-tiered, the system is expected to have a client-side and a server side which corresponds to the client node and application server node in the given diagram. Although the application component and the database components can run on different servers, in this project they are expected to be deployed on one server container. The related artifacts along with execution diagram is clearly depicted below. Furthermore the application server and the client is to be communicated via HTTP and TCP networking protocols.

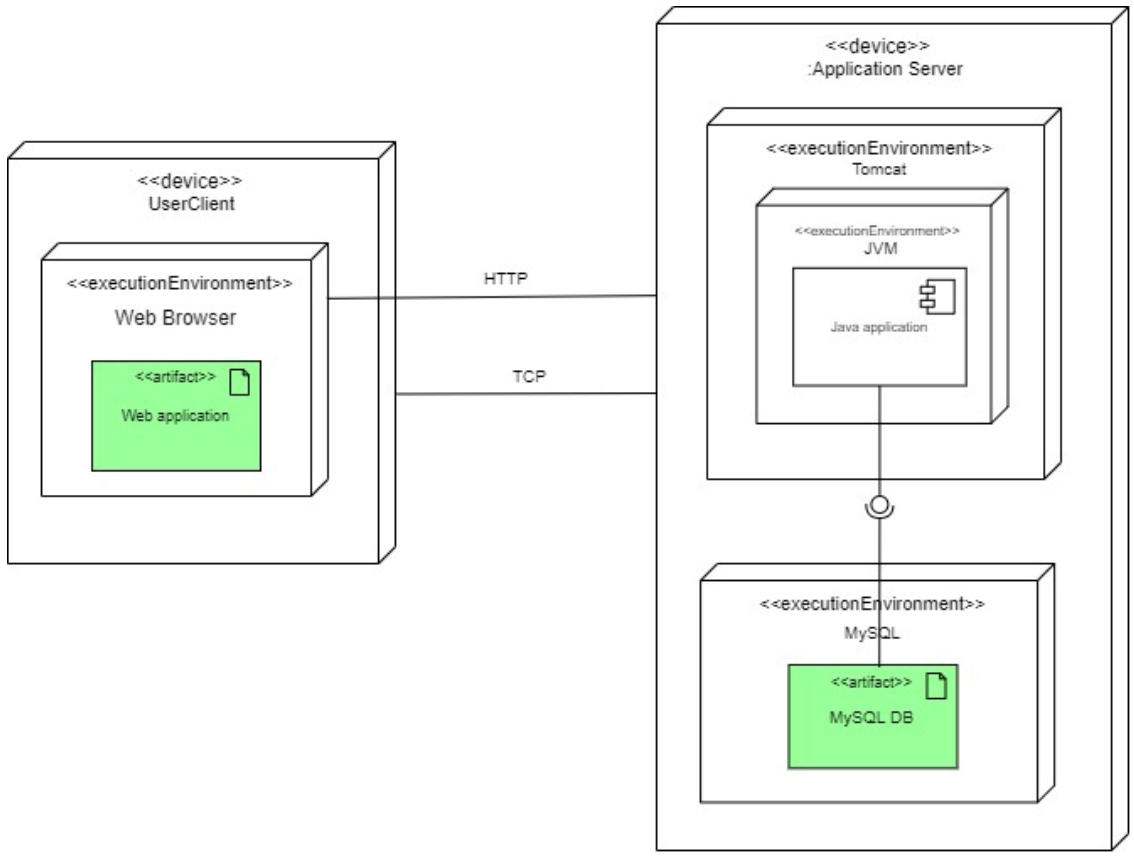


Figure 25 Deployment-diagram

5.2 Wireframe-designs

5.2.1 User Registration

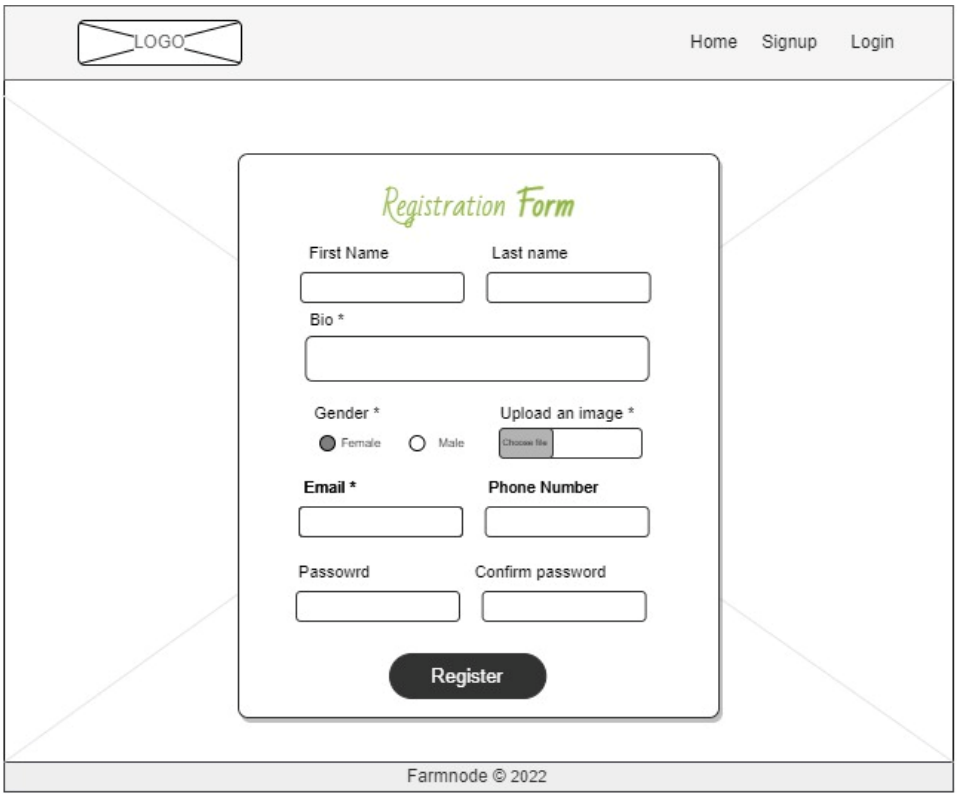


Figure 26 Registration-wireframe

5.2.2 Login

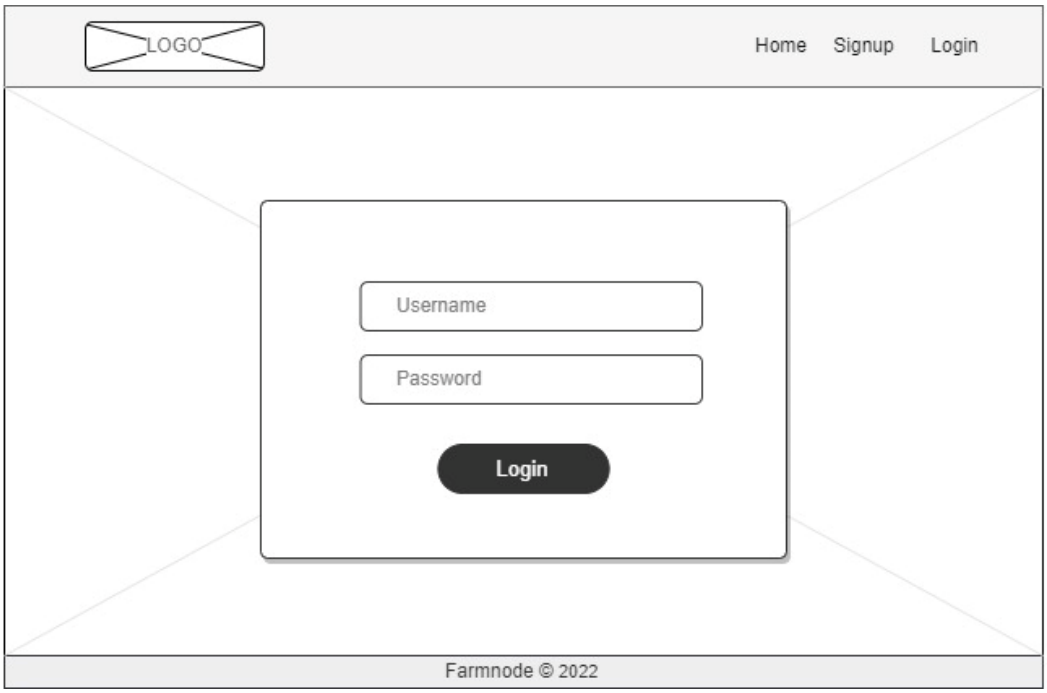


Figure 27 Login-wireframe

5.2.3 Add produces

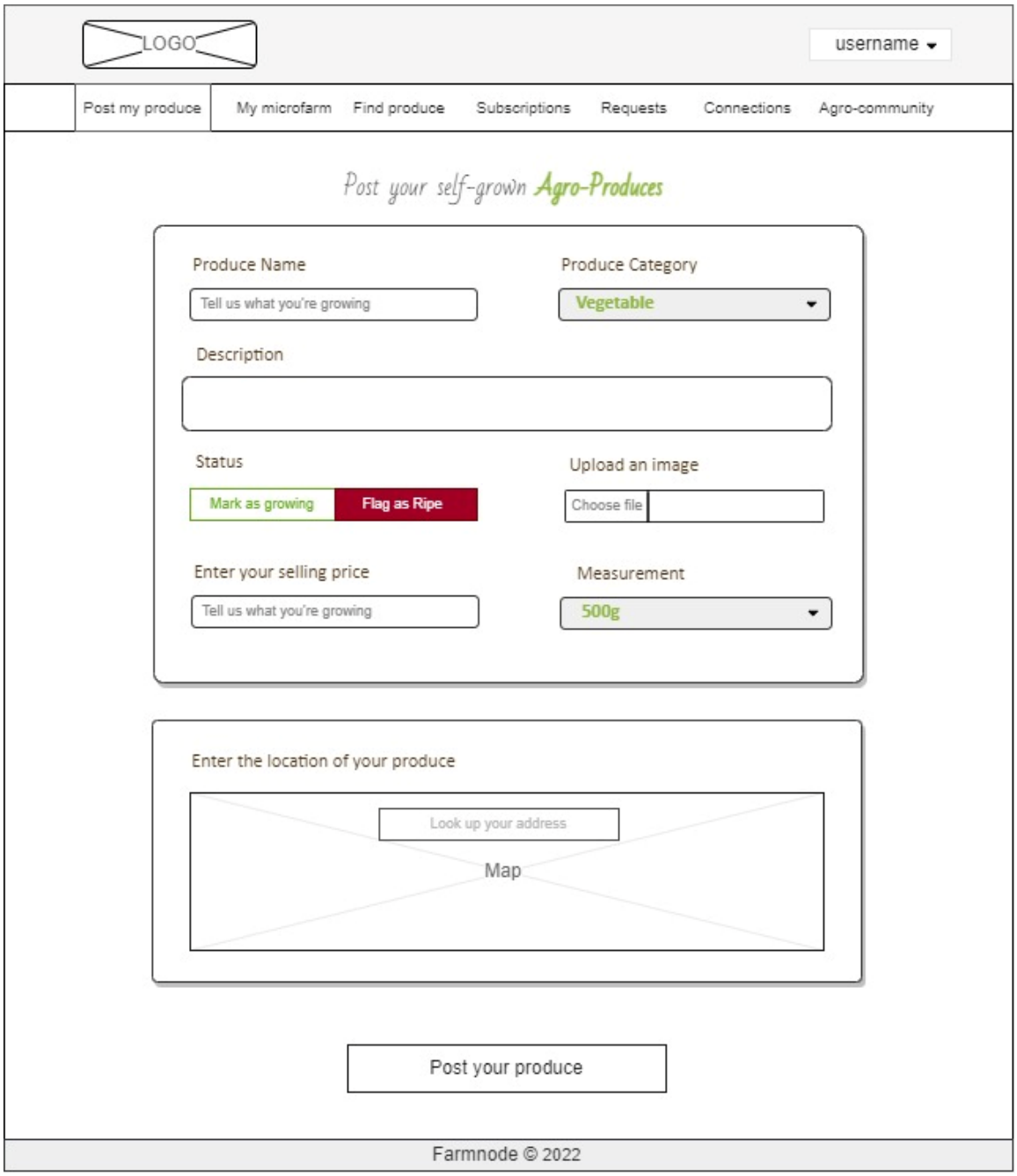


Figure 28 Post produce panel-wireframe

5.2.4 My Micro-farm dashboard

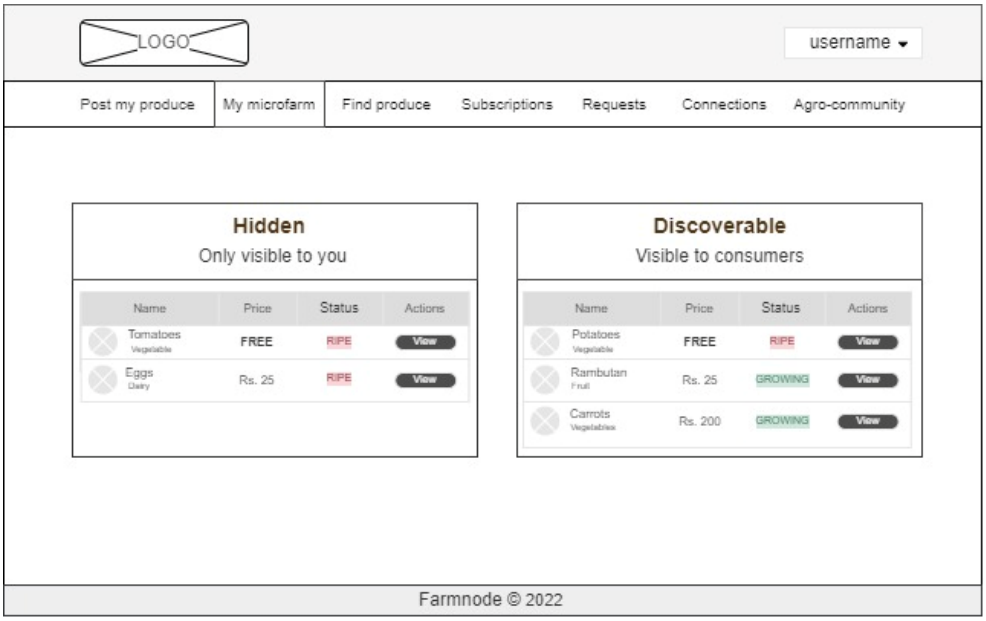


Figure 29 Microfarm-dashboard-wireframe

5.2.5 View produce

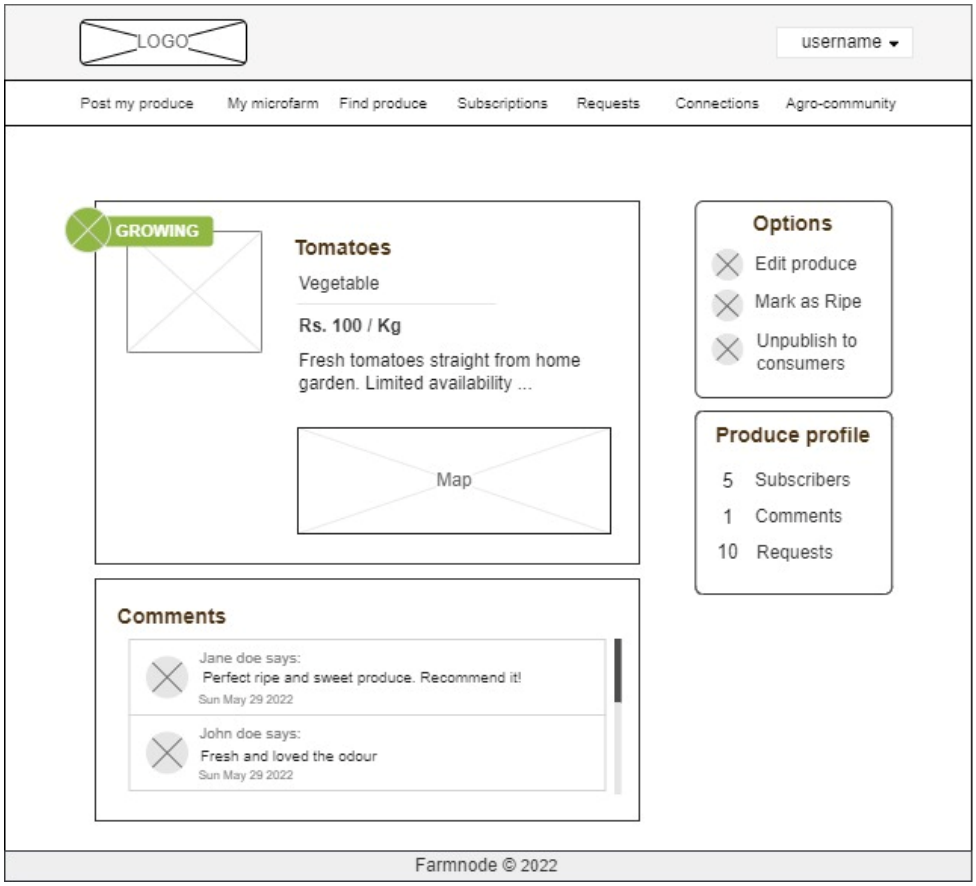


Figure 30 View produce-wireframe

5.2.6 Edit-produce

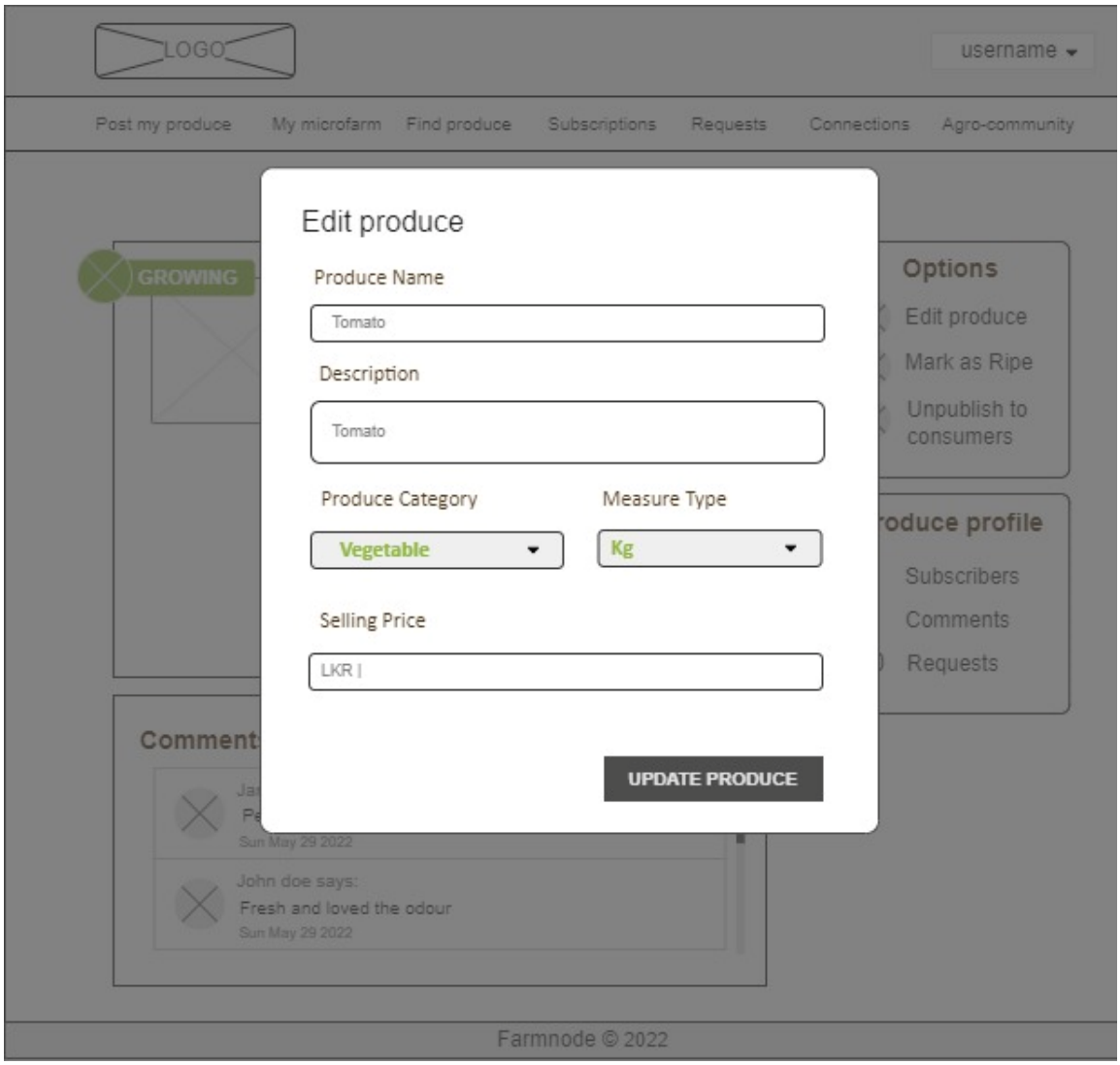


Figure 31 Edit produce - wireframe

.

5.2.7 Find-produces (Geo-visualization panel)

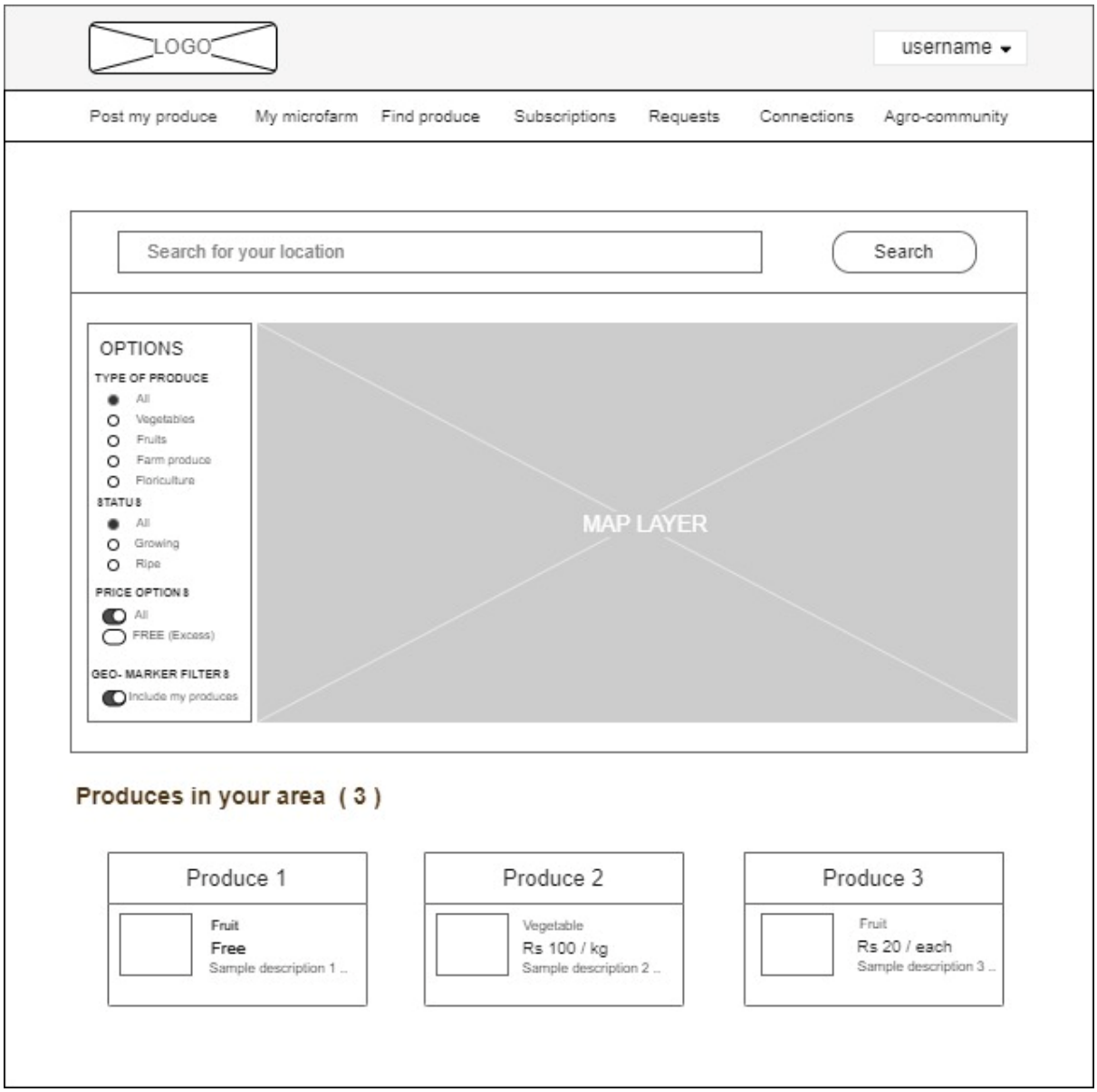


Figure 32 Find produces - wireframe

5.2.8 Others-produce

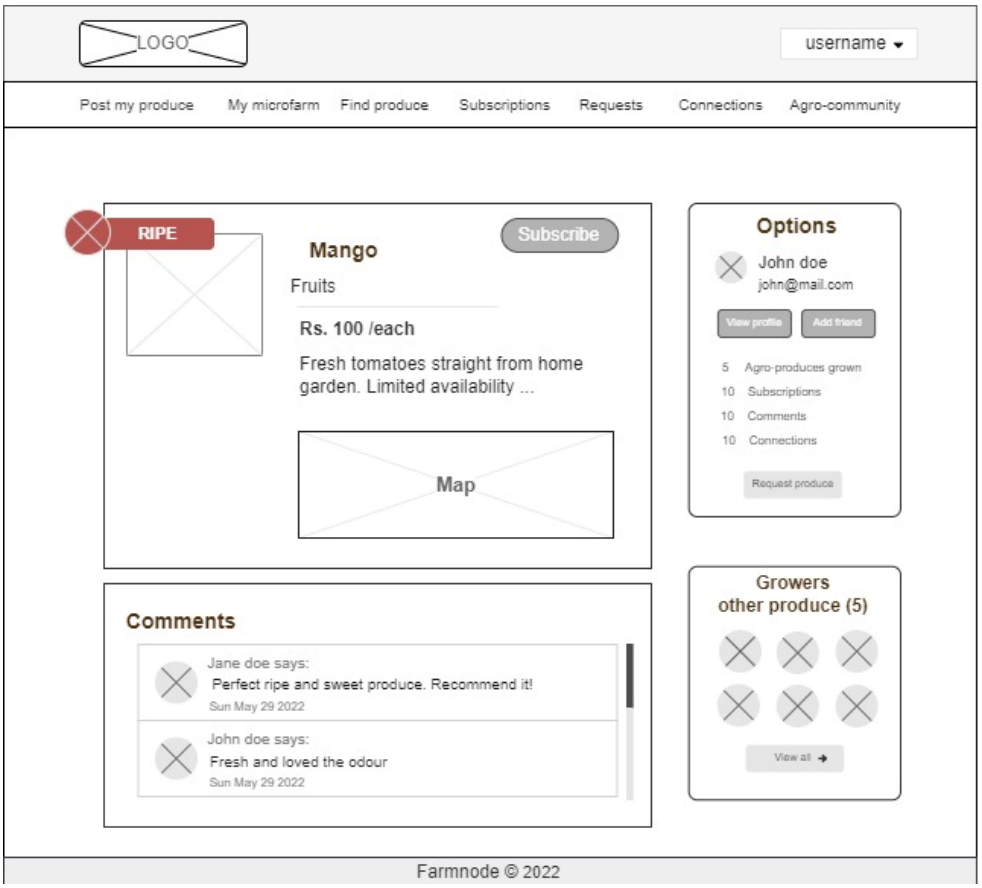


Figure 33 Others produce - wireframe

5.2.9 Subscriptions

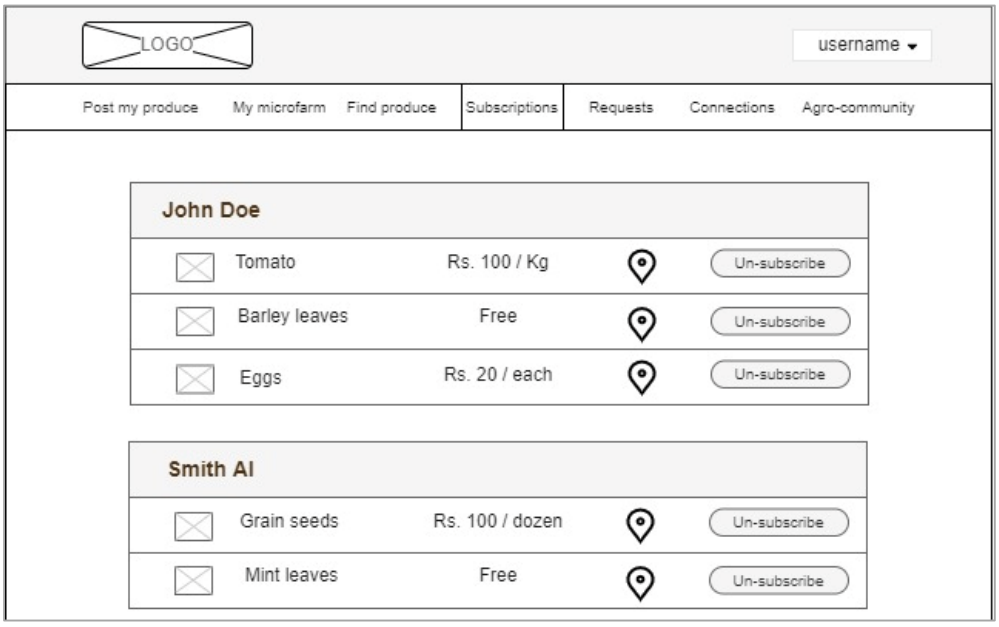


Figure 34 Subscriptions list - wireframe

5.2.10 Produce request

Figure 35 Produce request - wireframe

5.2.11 View inbound/outbound requests

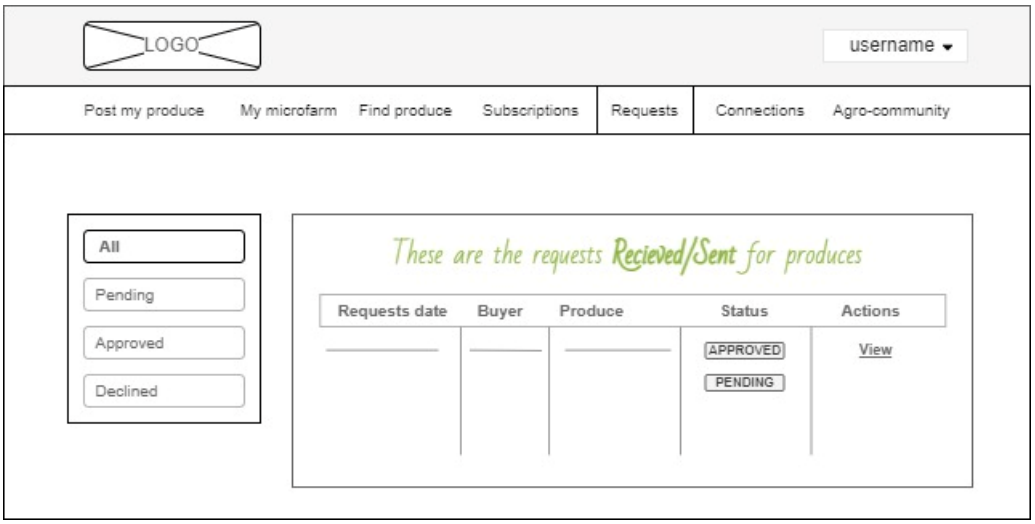


Figure 36 Request listing- wireframe

Chapter 6: Implementation

6.1 Project architecture

The overall strategically architecture of the web application is composed of a client application communicating with the server-side application. Client end consists of an angular project that interacts with the user and makes API calls. The server end consists of a spring boot + JPA (Hibernate) app MySQL as database. This database understands SQL and stores data in the form of tables. The communication protocol relayed in between is chosen to be REST oriented structure communicating over HTTP rather than SOAP. In this way the front-end application communicates with the back-end in a request-response manner as given fig.24 below. It provides an ease of maintenance of the code base, managing presentation code and business logic separately, so that a change to business layer, does not impact the presentation layer and vice versa.

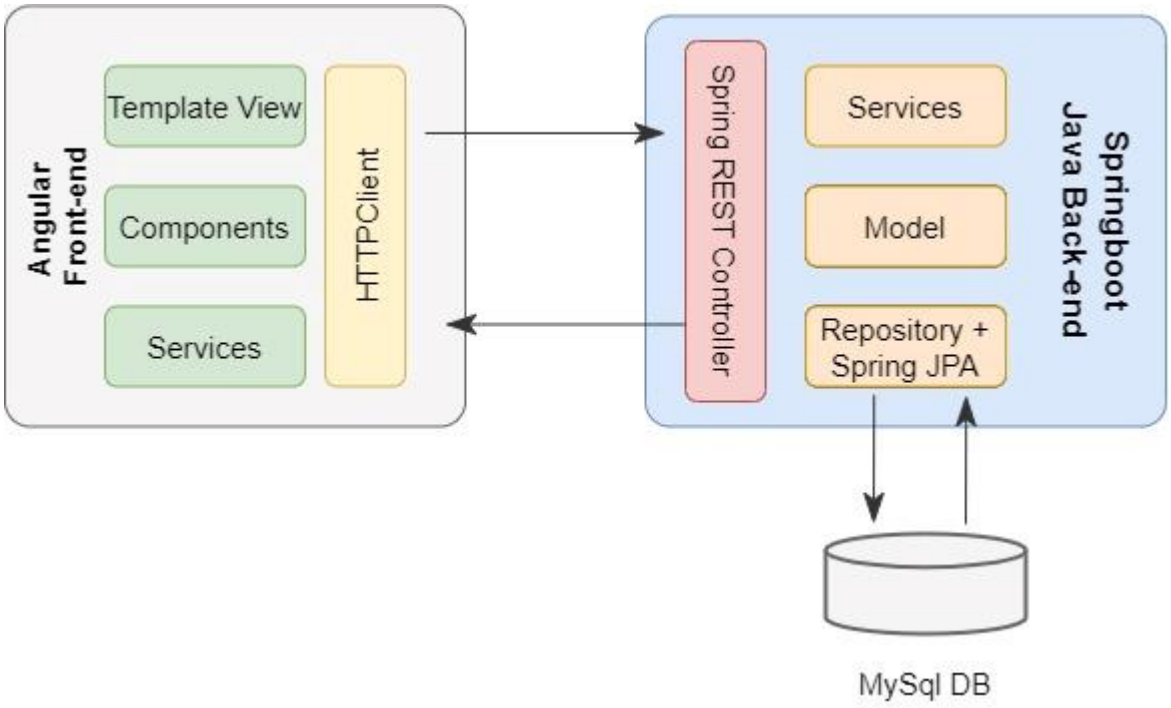


Figure 37 Project-architecture

6.2 Server-side application

The back-end server being the core-part of the application is built using **Java** as the main language as it is a pure OOP-friendly language. The light-weight **Springboot-framework** was favored for implementation considering the several advantages it provides for RESTful web service implementations which will be explained in-detail in the next chapter. The application is built on a pattern close enough to the MVC pattern with the exception that this application does **not** render the view itself but responds to the client as per the API requested data to be processed as the application-view. The flow of the mediating-layers between the client and the database in the request-response mechanism is as given below,

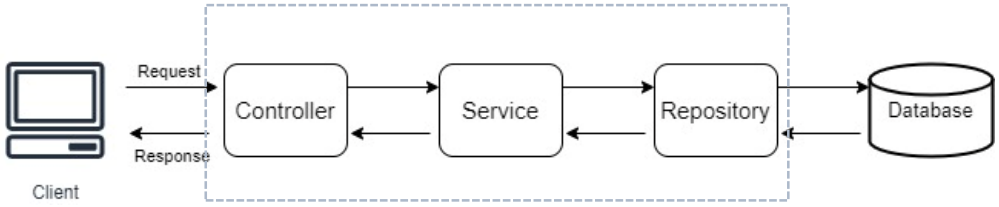


Figure 38 Flow of request-response

6.2.1.1 Controller class

The controller layer is responsible for receiving and disposing the end points through which the client end communicates. In this project there are 8 core controllers which deals with their respective entities. At each end-point of the controller class user requirement and functionality is caught by type of request provided by the annotations (@RequestMapping) along with its specific path. An example of the 'ProduceController' is given below when the user requests to create a new produce. The 'createProduce' method will be called when the users request is a POST request along with the produce payload to the ***'[host URL]/api/produce'***. Once the controller method verifies the request body, the service-layer (business-logic) is triggered and the final output is sent back as JSON response.

```java
@RestController
@RequestMapping("/api/produce")
@AllArgsConstructor
public class ProduceController {
    private final ProduceService produceService;

    @PostMapping
    public ResponseEntity<ApiResponse> createProduce(@RequestBody ProduceDto produceDto, HttpServletRequest request) {
        Principal principal = request.getUserPrincipal();
        String username = principal.getName();
        produceService.save(produceDto, username);
        return new ResponseEntity<>(new ApiResponse( success: true,  message: "Produce has been added"), HttpStatus.CREATED);
    }
```

Figure 39 Controller

6.2.1.2 Service class

The service layer implements all the necessary logic related to a model entity including the CRUD operations corresponding to the similar functional methods in the repository which in turn reflects the actions on the database. In this project separate service classes are implemented for each domains of the system. (eg: UserService/ProduceService/PostService/etc.) Each methods of this class houses the business logic to manage the entities or even return exceptions when the related entity is not available (exception-handling implementation is included in the **appendix-B**). Hence multiple repositories are involved in these methods in order to gather the entity fields or updating changes to related entities. An example of a service class in the project is given below.

```java
@Service
@AllArgsConstructor
@Slf4j
@Transactional
public class ProduceService {
    private final UserRepo userRepo;
    private final ProduceRepo produceRepo;
    private final UserService userService;
    private final ProduceMapper produceMapper;
    private final SubscriptionRepo subscriptionRepo;

    public void save(ProduceDto produceDto, String Username) {
        produceRepo.save(produceMapper.map(produceDto, userService.getUser(Username)));
    }

    @Transactional(readOnly = true)
    public List<ProduceDto> getAllProduce() {
        return produceRepo.findAll() List<Produce>
                .stream() Stream<Produce>
```

Figure 40 Service

6.2.1.3 Repository

In this project, pattern the repository layers are the highest layers after the business logic that persist the data from the domain entity to the **MySQL** database vice versa. The repository is

an interface that sends queries to the database. Every repository is mapped to its own entity, hence as a result it is updated by its own query. The base methods listed in these interfaces are invoked in the service layer as required. In this project, Springboot's JPA specification is used to easily connect to the database and manage relational data besides the use of JDBC. All the repositories **extend** JPARepository with the corresponding entity and the ID type to use the default CRUD operations implemented in it. Thus we don't need to implement any CRUD methods in the repository as it will automatically trigger the related query auto-generated by hibernate. Below is a code example of the produce repository. There is no CRUD operations implemented but only the methods related to custom and derived queries.

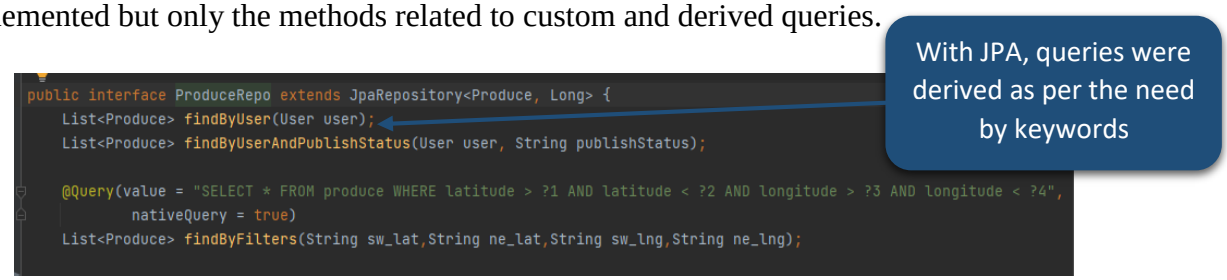


Figure 41 Repository

6.2.1.3 Model-Objects

Each entities of the system have their unique domain classes with their related properties. In-order to avoid the boiler-plate codes of getters, setters and constructors of each encapsulated class, annotations of Lombok - a predefined reliable-library were used as below to improve quality of code. Furthermore the associations of entities such as one-to-one, many-to-one and many-to-many relationships with others are mapped with the aid of Hibernate and JPA.

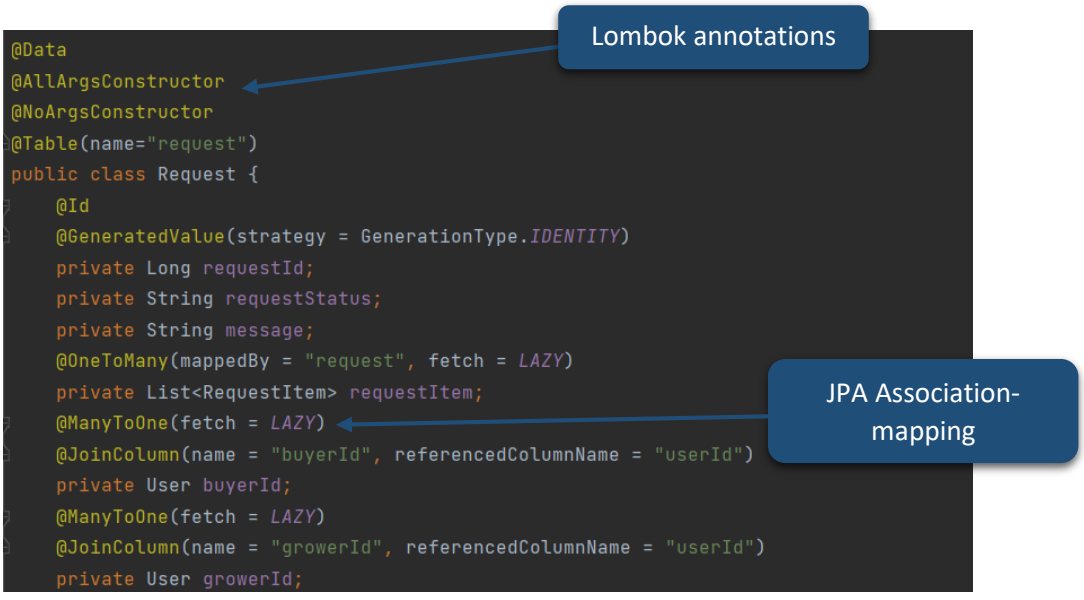


Figure 42 Model-Objects with Lombok and JPA annotations

6.2 Front-end application

The implementation of the front-end was made with Angular framework taking advantage of the benefits and principles of the MVC software design-pattern. The application was divided as individual-components for modularization. Modules made it easy to organize functions segregating as re-usable structures. Each module is implemented having its own controller, view and a model. Distinct models are created to control the data presentation pushed to view. In this way, the application-logic is isolated from the user-interface providing efficiency in focusing on different aspects of the entire application. The views are implemented with HTML, CSS, angular-directives and templates. An overview of component segmentation is as below.

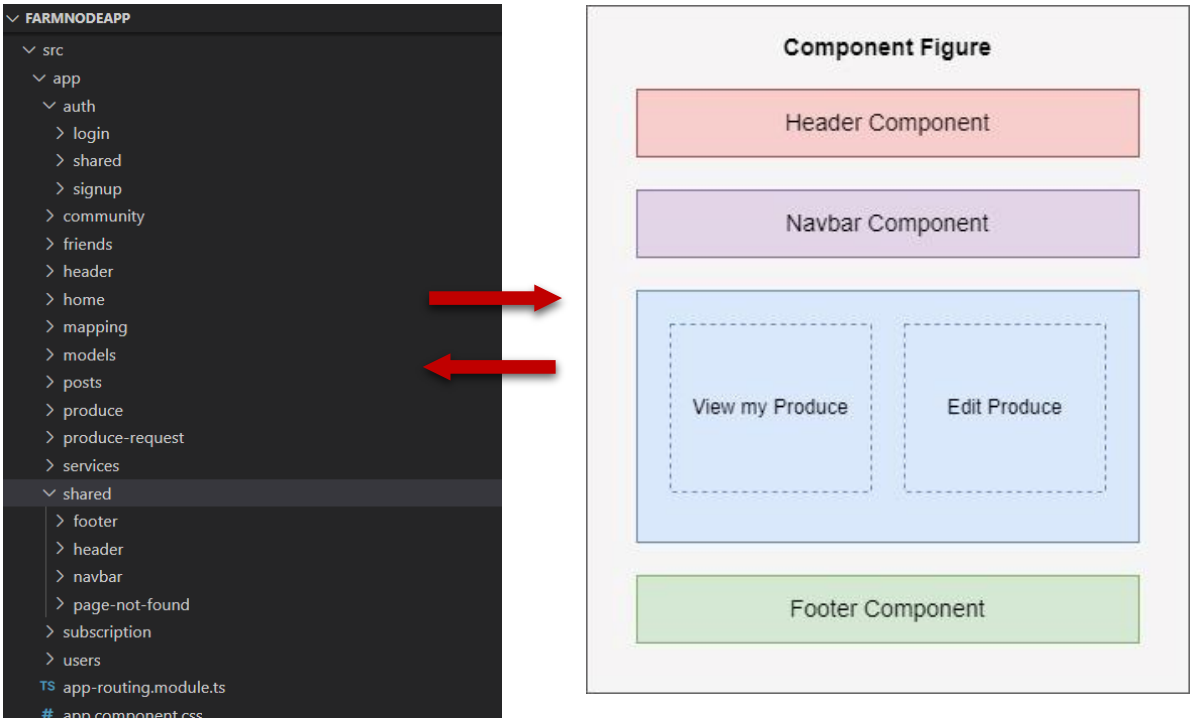


Figure 44 Project components

Figure 43 Component-draft for a page

Each entity has its related service typescript-file which communicates with the back-end server over HTTP request protocols using the AJAX-service class provided by Angular. This gave the ability to effectively map requested response objects to models and manage error handling. Furthermore the methods in these scripts are re-usable in different components enhancing code re-usability and maintainability.

- **Design user interfaces given in Appendix B**

6.2.2 Core functions of the application

6.2.2.1 Security

Since the architecture is based on a REST protocol, stateless means of security should be maintained between the client and server. Thus framework's security is configured to generate and send a valid JSON Web-Token (JWT) containing the user details once a user is authenticated. Each end point of the controllers require this token along with the request in order to identify the user making the request without re-searching the database. In this way, the application is more scalable and can attend many requests without synchronization. Therefore only authorized users are able to request or manipulate resources on the server avoiding CSRF vulnerabilities. Furthermore the token is set with an expiration period so that the application is even secure. Therefore a refresh-token is passed along during authentication to be used to renew the access-token skipping the process of re-authenticating to make the user experience a streamline flow.

- **Refer appendix B for a detailed sequence-diagram for the involvement of JWT in the application**

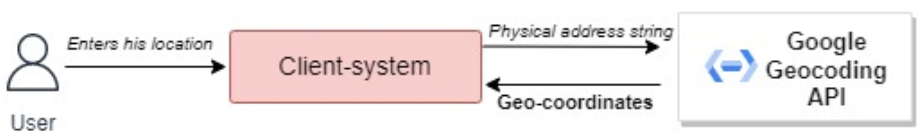
6.2.2.1 Finding produces-module (Geo-visualization on interactive maps)

One of the core-functions of the system is to visualize a produce/product posted by a grower on an interactive map through which potential consumers can locate them. Every location on earth has a global address called the coordinates. (Journey North, 2019) A Coordinate is a point feature geometry in GIS corresponding to a latitude and longitude.

- **<u>While adding a produce:</u>**

  While adding a produce to the listings, the user is asked to place a marker on his location or enter his physical address which will be then used by Google Geocoding API to convert address like *"No.138, Kinsey road, Colombo 03"* into geographic coordinates like "***latitude 37.430 and longitude -122.0837***" to be stored in the database.

Figure 45 Geocoding API



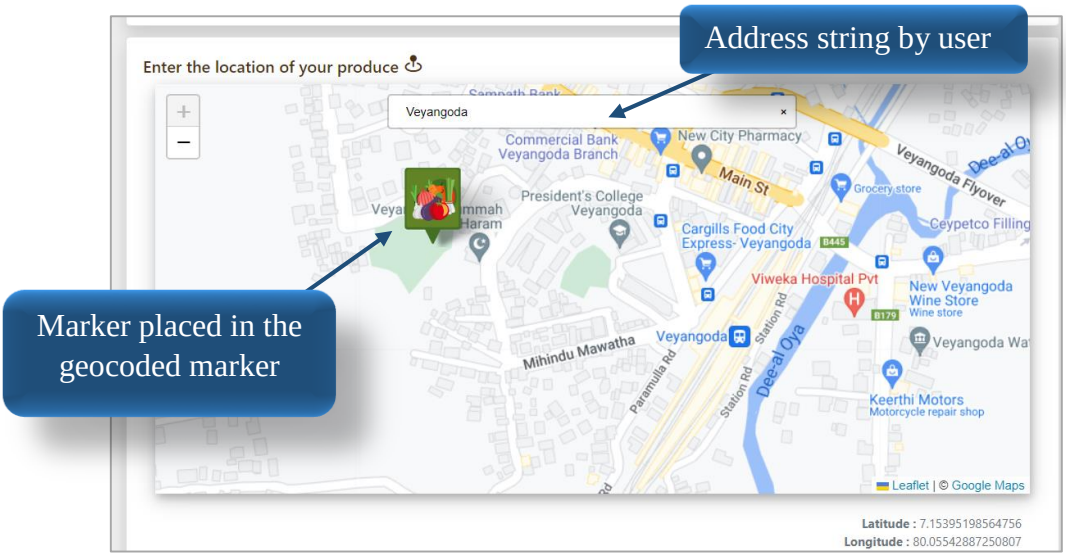


Figure 46 Geocoding while adding a produce

- **While retrieving produces :**

1. For visualizing features on any map, the format through which data is rendered is GeoJSON data which is a global geo-spatial standard. It represents data with its properties and coordinates. Therefore while retrieving the produce details from the database, an algorithm is curated in the backend to generate GeoJSON data. Each produce is a feature and is treated as a point-feature in the data with its own properties, since the purpose is to visualize as point markers in a map. Refer **appendix** C for detailed code-flow

```
//for each result of produce from the DB
for (ProduceDto element : produceDtoList) {
    FeatureDto feature = new FeatureDto();
    PointDto point = new PointDto(element.getLongitude(), element.getLatitude() );
    feature.setGeometry(point);
    feature.setId(element.getProduceId().toString());
    feature.setProperties
            ("{\"Name\": \""+element.getProduceName()+"\", "+
                    "\"description\": \""+element.getDescription()+"\", "+
                    "\"produceStatus\": \""+element.getProduceStatus()+"\", "+
                    "\"price\": \""+element.getPrice()+"\", "+
                    "\"category\": \""+element.getCategory()+"\", "+
                    "\"measureType\": \""+element.getMeasureType()+"\", "+
                    "\"address\": \""+element.getAddress()+"\", "+
                    "\"grower\": \""+element.getGrower()+"\", "+
                    "\"filename\": \""+element.getFilename()+"\", "+
                    "\"publishStatus\": \""+element.getPublishStatus()+"\" "+"}");

    dtoList.add(feature);
}
//set a feature collection
featureCollection.setFeatures(dtoList);
return FeatureCollectionBuilder.getInstance().toGeoJSON(featureCollection);
```

Figure 47 Algorithm to generate GeoJSON data

2. In the Angular front-end, an open source **Leaflet JS an open-source library** is customized to the render the map component consuming the fetched GeoJSON data to use it as a point layer to visualize as individual markers on the map.

```
this.mappingService.getMap(this.findProducePayload).subscribe(
  (data) => {
    this.geoJsonLayer = L.geoJSON(data, {
      pointToLayer: function (feature, latlng) {
        var statusString;

        if (feature.properties.produceStatus == 'RIPE') {
          statusString =
            '<div style="z-index: 10;position: absolute;  left: -16px !i
        } else if (feature.properties.produceStatus == 'GROWING') {
```

Figure 48 Angular code-Point to layer

3. Different markers represent different categories of the products such as vegetables, fruits, dairy products and floriculture. Markers are accurately positioned on the exact coordinate and a popup-box is bind on the marker so that when clicked it shows the relevant produce details such the produce name, price, measure-type and grower's name extracted from the properties of the feature properties on the GeoJSON data.

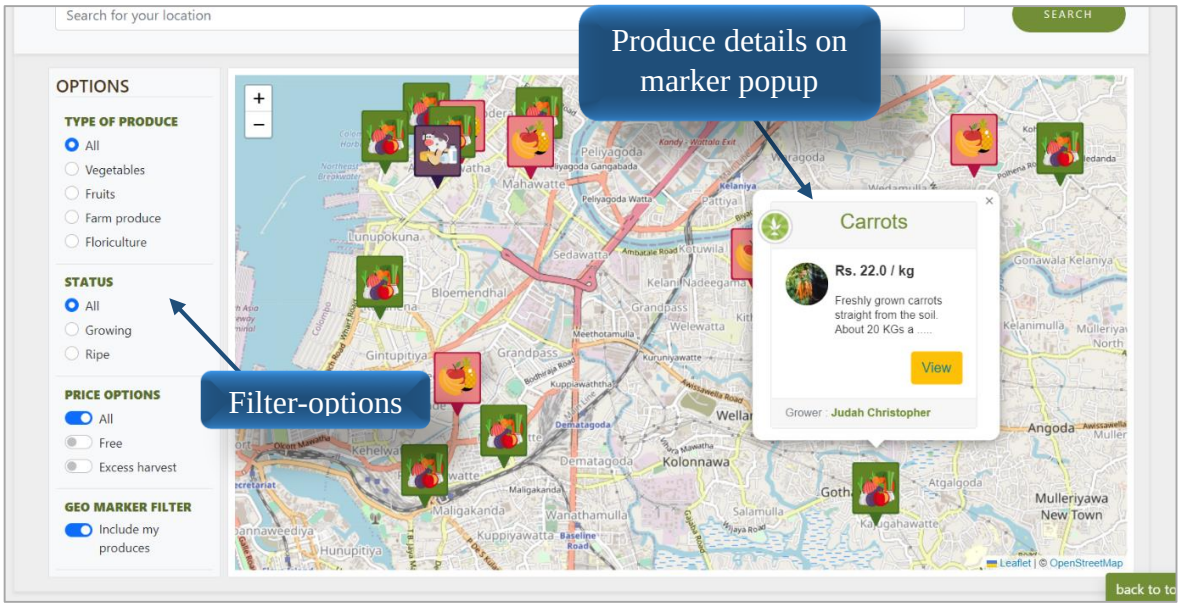


Figure 49 Interactive map with produce markers

4. Inorder to increase efficiency and reduce load, the GeoJSON-data is requested only for the produces within the bounding box that is the North-East Latitude/longitude and the south-west latitude/longitude representing the corners of the visible range in the map.

6.2.2.2 Other Special Design Patterns used

State-Machines: Design pattern

The State-pattern is one of the twenty-three well known GoF design patterns. The core notion of the pattern is to let an object to encapsulate varied behaviors on the same object based on its state. Each of these states represents a status that could change to another state. These state-changes are known as transitions. (Baeldung, 2021) Separate states could be defined and the transition between them could be programmed logically in order to avoid complex if/else statements ultimately making the code cleaner.

In this solution this pattern was applied to the enum-class defining the constants to handle the status of the produces to provide type-safe implementation. A clear logic on the state transition from “GROWING-state” to “RIPE-state” and the return value was coded as given below.

```
public enum ProduceStatus {
    GROWING{
        @Override
        public ProduceStatus nextState() {
            return RIPE;
        }
        @Override
        public String responsibleState() {
            return "GROWING";
        }
    },
    RIPE{
        @Override
        public ProduceStatus nextState() {
            return GROWING;
        }
        @Override
        public String responsibleState() {
            return "RIPE";
        }
    };
    public abstract ProduceStatus nextState();
```

Figure 50 Produce-State machine

Therefore when a user updates on the produces status in the e-micro farm from growing to ripe, the pattern aids to provide a straightforward reusable code to maintain this status ultimately providing a cleaner code.

Chapter 7: Testing and validation

7.1 Introduction to testing

The ultimate objective of this project is to provide a working software solution. Hence testing is the most crucial step in the entire development lifecycle which validates that the implemented solution works as expected to meet our specified requirements. The test results aids to identify and minimize errors as much as possible that may arise in an actual working environment. In this project several testing approaches were used to ensure the delivery of an error-prone working solution

7.2 Testing approach and methodology

There are several approaches when dealing with the testing phase of the SDLC of a system. Below are the main testing rationales that has been adopted in the project.

7.2.1 Functional testing

- **Unit-testing**

  Unit tests are tests that evaluates individual components that make up the application system under isolation and scrutinized independently. (Kolodiy, 2021) This test was carried out a in a **test-driven-development (TDD)** methodology in parallel to development-process. The rationale of this testing underlies the **white-box technique** since deep coding segments are used. These tests made it possible to reduce errors as much as possible when interconnecting units into one whole system. Hence improving the quality of the system.

- **Integration testing**

  Followed by unit-tests, the interaction of groups of components are tested to ensure functional validity.

- **System-testing**

  Once the whole system is implemented, the system was considered as a whole concentrating the major functionalities it performs. This is a black-box testing approach where valid and invalid inputs where fed to the application through which the systems responses behavior was asserted manually to derive test conclusions.

7.2.1 Non-functional testing

- **Security Testing**

  This testing reveals the flaws of security aspect of the application. System responses towards unauthorized access to web pages and functions were evaluated so that loopholes that damages data integrity could be avoided.

- **Compatibility Testing**

  Cross Browser testing was conducted to reveal systems availability in the recommended environments. Browsers such as Google Chrome, Microsoft Edge and Mozilla Firefox was used in evaluation.

7.3 Automated Unit-testing rationale

Since the back-end application does all the processing logic with data in the project, all the core components handling the business logic were subjected to tests. Tests were automated using the **JUnit** an open-source framework which is specially tailored for Java based applications. Since unit tests is independent, all the external-dependencies of the class/method that is under tests are mocked with the help of **Mockito** to maintain a controlled testing environment. Mockito is a mocking framework that is well suited to be used in conjunction with JUnit for mocking and stubbing external classes.

7.3.1 Evidence of unit-tests done in the project

All the business logic layers such as the service layers, object mappers and repository interfaces were subjected to isolated unit-tests. Test cases were drafted to cover all edge cases such as positive outputs and exceptions thrown in each unit were asserted with the help of the assertion methods provided by JUnit. Give below are the examples of test cases mapped to the produce-service module of the system.

ProduceService (Test-class)

All the external dependent classes of the Produce-service are mocked and the methods of it a stubbed in the test environment considering that the dependency unit works fine since they are unit tested separately independently.

```
@RunWith(MockitoJUnitRunner.class)
class ProduceServiceTest {

    @InjectMocks private ProduceService produceService;
    @Mock private ProduceRepo produceRepo;
    @Mock private ProduceMapper produceMapper;
```

Figure 51 Unit test - dependency mock

- Test-case to verify whether a produce is successfully added when a valid produce object is passed. Assertion was made to verify that the exact same produce object is being saved.

```
@Test
void testProduceSaving() {

    //given
    User user = new User( userId: null, name: "Testing", username: "testing@test4434.com", password: "
    Produce produce = new Produce( produceId: null, produceName: "Test", description: "check", produceStatu

    //when
    when(userService.getUser(user.getUsername())).thenReturn(user);
    produceService.save(produce, user.getUsername());

    //then
    ArgumentCaptor<Produce> produceArgumentCaptor = ArgumentCaptor.forClass(Produce.class);
    verify(produceRepo).save(produceArgumentCaptor.capture());

    Produce capturedProduce = produceArgumentCaptor.getValue();
    assertThat(capturedProduce).isEqualTo(produce);
}
```

Figure 52 Automation test case - example -1

- Test-case to verify whether all the produces are the being retrieved from the method.

```
@Test
void testGetAllProduce() {
    Produce produce1 = new Produce( produceId: 1L, produceName: "test", description: "che
    ProduceDto dto1 = new ProduceDto( produceId: 1L, produceName: "test", description: "c
    List<Produce> produceResult = new ArrayList<>();
    produceResult.add(produce1);
    List<ProduceDto> dtoResult = new ArrayList<>();
    dtoResult.add(dto1);

    //when
    when(produceRepo.findAll()).thenReturn(produceResult);
    when(produceMapper.mapToDto(any(Produce.class))).thenReturn(dto1);

    //then
    List<ProduceDto> resultList = produceService.getAllProduce();

    assertEquals( expected: 1, resultList.size()); // check size
    assertEquals(dtoResult, resultList); // check size
    assertEquals(dto1, resultList.get(0)); // check dto result
    Mockito.verify(produceMapper).mapToDto(produce1); //verify the mapper
}
```

Assertion methods

Figure 53 Automation test case - example -2

❖ Test-case verifying that expected exception is thrown when trying to get produces of a non-existing user,

```
@Test
void testExceptionIsThrownWhenNotExistingUserIsUsedToGetProduce() {
    when(userRepo.findByUsername(any())).thenReturn(null);

    assertThatThrownBy(()->produceService.getProduceWithSubscription( id: 1L, username: "testing"))
            .isInstanceOf(UsernameNotFoundException.class)
            .hasMessageContaining("User not found in the database");
}
```

Figure 54 Automation test case - example -3

Produce Repository (Test-class)

Since the repository deals with an external database to manipulate the records and entities, to test it as a unit, **'H2'** which is an in-memory database tool was used representing the task of SQL database in the CRUD operations in-order to avoid coupling an unwanted changes in data.

❖ Test-case to verify whether the findByUser method returns user's produces as expected

```
void testFindingProduceOfExistingUser() {
    //given
    User user = new User( userId: 1L, name: "Testing", username: "testing@test.com", password: "123", filename: "fi
    userRepo.save(user);
    Produce produce= new Produce(
            produceId: 1L, produceName: "test", description: "sds", produceStatus: "RIPE", price: 1.00, category: "ssd", m
    //when
    repoUnderTest.save(produce);

    //then
    assertThat(repoUnderTest.findByUser(user)).isNotNull();
```

Figure 55 Automation test case - example -4

❖ Test-case to validate the updateProduceStatus method whether it updates as expected.

a. Produce is saved
b. Status is updated
c. Verifying whether the new status matches the expected status.

```
@Test
void testUpdateStatusToRipe() {

    User user = new User( userId: null, name: "Testing", username: "testing@test.com", password: "123", filename: "fal
    userRepo.save(user);
    //given
    Produce produce5= new Produce(
            produceId: null, produceName: "test", de          s", produceStatus: "RIPE", price: 1.00, category: "ssd", me
    repoUnderTest.save(produce5);

    //when
    String newStatus ="RIPE";
    repoUnderTest.updateProduceStatus(newStatus,produce5.getProduceId().intValue());

    Produce result = repoUnderTest.getById(produce5.getProduceId());
    //then
    assertEquals(result.getProduceStatus(),newStatus);
```


Figure 56 Automation test case - example -5

7.3.2 Overall automation unit-test results

In overall conclusively all the business logics successfully passed the tests with no flaws ensuring a perfect working solution behaving expectedly on all the forecasted edge cases. Screenshot of result evidence is given below.

Figure 58 Overall test-results 1

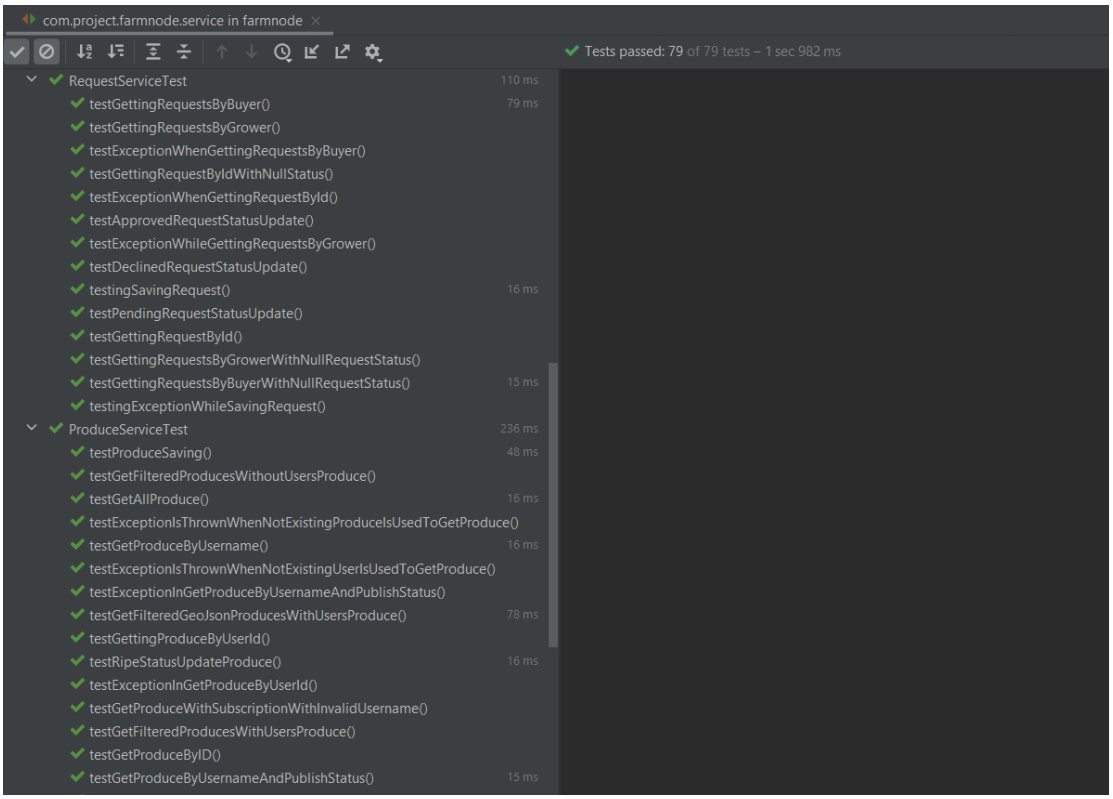


Figure 57 Overall test-results 2

7.4 System-test plan

The test plan is drafted covering all the boundary conditions that fulfill the existence of accurate functionalities including validations, data insertion, update and retrieval. Hence we can make sure that the all the functionalities work as expected with user-interaction.

Test-Case ID	**Test-case**	**Scenario**	**Expected result**
TC001	Test User-registration module	Registering with existing email.	Display error message related to user
TC002	Test user-registration module	Registering as a new user	Display successful registration message
TC003	Test Login module	Login leaving an empty field	Display error message regarding empty fields
TC004	Test Login module	Login with invalid credentials	Display error message
TC005	Test Login module	Login with valid credentials	Navigate to interface displaying successful authentication
TC006	Test adding produces	Add a produce with incomplete fields	Display error message regarding empty fields
TC007	Test adding produces	Upload a non-image file	Display file-type error message
TC008	Test adding produces	Input an invalid address	Display address not found error
TC009	Test adding produces	Input valid address	Marker placed accurately on the location

TC010	Test adding produces	Add a produce with completed details	Display success and produce appears in your produce profile
TC011	Test Produce-management	Viewing all users' produces	Display all the user's produces
TC012	Test Produce-management	Viewing a particular listed produce	Navigate to interface displaying clicked produce details accurately
TC013	Test Produce-management	Updating produce details	Display success and the produce data is updated
TC014	Test Produce-management	Updating produce details to be free	Display success and the produce should be listed as free
TC015	Test Produce-management	Changing produce status to "RIPE"	Display success and the status is updated and email is sent to subscribed users
TC016	Test Produce-management	Changing produce status to "GROWING"	Display success and the status is updated
TC017	Test Produce-management	Changing produce visibility to "False"	Display success and the produce won't be shown on the produce map
TC018	Test Produce-management	Changing produce visibility to "True"	Display success and the produce will be shown on the produce map
TC019	Test Geocoding in find produces module	Search by invalid address	Display error

TC020	Test find produces module	Search by valid city/neighborhood address	The map's bounding box should be set as per the entered address
TC021	Test find produces module	Change map bounds by dragging	Produces result should change dynamically
TC022	Test find produces module	Change map bounds by zooming	Produce results should change dynamically
TC023	Test find produces module	Filter by produce attributes	Produce results should change dynamically
TC024	Test finding produces	Filter by pricing strategy	Produce marker result should change dynamically
TC025	Test finding produces	Viewing produce markers	Popup should appear with relevant accurate produce details
TC026	Test View produce	Subscribe to produce	Display success and the subscribed produce should appear in subscription list
TC027	Test View produce	Unsubscribe a produce	Display success and the subscribed produce be removed from subscription list
TC028	Test View produce	Adding a produce comment	Display success and appears on the comment list
TC029	Test produce-request	Add a produce to the request list	Added produce appears on the request list
TC030	Test produce-request	Click on the same produce to be added	Only quantity changes

TC031	Test produce-request	Update quantity of produce in a request	Reflected lineprice/total price updates
TC032	Test produce-request	Remove produce	Reflects change on the request list
TC033	Test produce-request	Placing a produce request	Display success
TC034	Test produce-request	Viewing sent produce-requests	Display all sent requests
TC035	Test produce-request	Viewing received produce-requests	Display all received requests list
TC036	Test produce-request	Approve a produce-request	Status of request updates
TC037	Test produce-request	Decline a produce-request	Status of request updates
TC038	Test interaction module	Sending connection request to another user	Display request sent and received on the receiver-end
TC039	Test interaction module	Approve a connection request	Display success and updates my-friends list
TC040	Test interaction module	Decline a connection request	Removes the user from inbound request list
TC041	Test community module	Creating a post	Display success and appears on the post feed.
TC042	Test community module	Commenting on a post	Display success and comment appears on the list of comments with accurate time and posted user

Table 4 Test plan

- The test case results included in **Appendix D**

Chapter 8: Tools and technologies

8.1 Implementation-platform

8.1.1 IntelliJ IDEA

IntelliJ is a powerful well-versed IDE for all types of projects with a providence of a free community edition to be used by all. IntelliJ is much easier to be used in comparative with other IDE's due to its faster learning curve and other integrated features. (Walker, 2018) This platform was chosen for the backend-development over other IDE's such as Netbeans, Eclipse and Xcode due to its scalability and support for Java based projects since Java is the core backend-language of this project. The debugging options provided on coding and the user-friendliness of the interface helped to increase programming effectiveness ultimately improving software codeman-ship.

8.1.2 Visual Studio Code

Visual Studio code is a well-versed code editor that supports a great number of programming/scripting languages. It was chosen for the front-end development as it is well suited for front-end development whereas this project relied on Angular framework housing HTML, CSS and Typescript. Another key factor for selection was the availability of codemanship support extensions such as auto code-completions, code-refractor and intellisense syntax mating etc. which was used in the project to provide clean readable code-lines.

8.2 Development tools/technologies

8.2.1 Java

The backend of the system was built on Java which is a widely preferred programming language with a global recognition. (IBM, 2019). The major reason for the choice over other language is the versed-ability to accommodate OOP concepts over other languages such as C#, PHP, Python, Ruby etc. Above all it is very convenient to code and debug as it is quiet straight forward. As a result it's simple to build clean code in the system focusing on individual modules and re-using codes. JDK 11 was configured was configured in the project which included the compilation and debugging dependencies to run a Java programmed platform.

8.2.3 Springboot

Springboot is a well implemented java framework built on top of Spring. It was used in the backend development to provide a state-of-art code architecture. One of the main reasons for the choice of framework was the support it provided for MVC oriented RESTful web service to meet the API requirements of the project. Furthermore it follows a "Convention over configuration" model so that developers have to less worry about dependencies to run a working solution as it is well handled automatically. This is called dependency injection. (Fol, 2021) The other major advantage over other available Java frameworks such as Micronaut and Quarkus is that it embeds a pre-configured tomcat server which can be run directly at a glance. This helped to focus on the outcome of the code-lines rather than spending ample of time on setting up configurations.

8.2.4 Spring Data JPA

Spring data JPA is an explicit repository that makes it easier to implement data access layer of the application without the need of re-writing boiler plating simple usual queries. The repository layer of the back-end application extended all the pre-defined methods to fetch data from the database and also to easily paginate the retrieved responses in order to send limited response to the front-end to make the application run smoothly.

8.2.5 Angular

For the front-end development Angular 13, an open-source framework supported by Google was used in this project. The framework is Typescript oriented which is a latest version and a superset of JavaScript ES6. The typescript domain was more useful in the project as it aided to catch typography errors on primitives in the early process of compiling and eliminate the errors at a glance. The framework is a component split friendly structure where the application could be segregated as different components. Thus this project was modularized into different components where shared codes could be re-used when needed and easily maintained and monitored with easier readability. This helped to achieve maximum productivity throughout the development lifecycle. Moreover the other distinguished feature of Angular **two-way binding** was helpful to achieve a dynamic ecosystem instead of static HTML pages by exchanging data from component to view and vice-versa. Therefore consequent refreshes of web-page was not necessary to update data making the project an effective modern solution.

8.2.6 MySQL

For effective management of database in the project, MySQL an open-source RDBMS engine was used. Relational tables were implemented as per the system needs and were connected to the Springboot application for data manipulation and retrieval. MySQL benchmark itself over other available databases in terms of security, flexibility and scalability.

8.2.7 LeafletJS

LeafletJS is a lightweight JavaScript library to build interactive mapping components in web applications supporting HTML and CSS. It aids to create and render interactive layers on the map tiles using backend or API processed data such GeoJSON data. This library is advantageous over Google Maps API and Bing Maps API due to its Open-sourceness and extensive customizability. In this project it was used in the produce visualization module where individual representative div-markers with popups where rendered using the generated feature GeoJSON data. The mapping tile providers used in the application were Open Street maps and Google maps API used with attributions.

8.2.8 Google-Geocoding API

Google geocoding API is a service provided by Google Maps platform where you can make requests to convert your address strings to coordinates over an HTTP request. Although similar API's such as MapBox, LocationIQ and Geoapify, the results provided by Google shows higher accuracy as it is well built with reliable data. (Rehan, 2022) This was useful in the application when adding a produce or finding nearby produces where our input address is converted into latitudes and longitudes to be used in conjunction with map layer of LeafletJS.

8.3 Testing tools

8.3.1 JUnit

This was crucial for this Hospital management system as each small functionalities played a sensible role and should be carefully implemented. It reduced the test efforts to deliver faster and better quality of a software. Apart from refining the stability of the program, it turned out to mainly reduce the stress and time spent on debugging ultimately contributing to boost up software carpentry skills.

8.3.2 Mockito

Since unit tests requires isolation from external dependencies impact, whike testing the dependencies needed to mocked to make it function as expected. Mockito is a Java-based mocking framework that was used for this purpose while unit testing in conjunction with Junit. In this way unit tests were successful to fix only the unit that is under test avoiding vague results of test results.

8.3.2 H2 database-engine

Another major external dependency of units under tests is the database. Inorder to avoid coupling of data and data mis-usal while testing an in-memory database H2 was used. H2 mocks the JDBC connection providing the exact same copy of original SQL RDMS that is compiled when running tests. In this way testing phase was very successful as only the logic of the unit was focused to be refined.

8.3.4 Postman API

Since the project deals with consuming REST API, it was required to test the responses from the implemented end-points. Postman is simply an API client tool which was used in the project to test these end-points effectively. This platform was chosen over other API clients such as ReadyAPI, Runscope due to the provided free-tier creating workspace environment saving complex HTTP requests as shown in the figure in appendix E. This resulted a less tedious work without the need of re-writing request-url repetitively.

8.3 Others

8.3.1 Git/GitHub

As per the risk analysis phase of the project, it was forecasted that loss of project files or inadequate tracking mechanisms on changes critical changes done to the code-base plays a significant risk impact on the project. Manually doing these may lead to failure or may even result in confusions. To avoid this, GIT a well-known open source VCS was used throughout the project. It is easy to keep track of the changes made to a file and maintain a record of what has been done. In case of trouble, it provides ways to easily revert the problematic changes made to the code. (NobleDesktop, 2018) A private repository was maintained in this project via GITHub to store the project file and the files were updated consequently as individual commits after a code-review.

- **Refer other project technologies in the appendix E**

CONCLUSION

The major underlying research problem highlighting the need for this project was the growing concerns faced by consumers in reaching out fresh agro-produces and local agro-growers finding it difficult to directly reach out a potential consumer. The dominating factor identified to be leading to the problem was the interference of third-parties (middlemen) due to the lack of a direct system mechanism to bridge the parties. Digital technologies especially web applications have taken over the world as part of our daily routines, gradually shaping the way we communicate, reach out and consume. Therefore a web platform is considered to have the potential to bridge the gap of the identified problem. Major research gap was identified in the market of existing applications with no direct use-case that serves to solve the problem.

In this project a web-based interactive e-micro farm platform was implemented as a central connective node to connect a producer and a potential consumer. All the prime objectives of the application set to include modules benefitting both the consumers and growers were implemented successfully. Geo-logical visualization was the major goal of this project where consumers can explicitly view the fresh-agro produces that are available in his/her neighborhood in an interactive map. Potential consumers who wish to get the produces from the growers can make a request through the system that goes straight to the other users (growers) incoming request list to be accepted or rejected. Since no middlemen interference is involved, all payments they receive are retained as their stable income obtained. Furthermore inter-connectivity was another objective of the application where users can be connected to with other users, keep track of the produces that they grow by subscribing to it or even interact via posts or questions in the community module. In this way, the developed application proves to overcome all the early identified issues which will be a great way to find, share fresh produces and build community all at the same time.

Future recommendations and limitations

Since only a limited scope of functionalities were mapped to fit the given project timeline, the solution was a minimum viable product to serve the as an introductory system. At the end of this project many necessary features were identified to be needed. Few of the recommendations that is believed to add value in further implementation are;

- Native language support modules could be integrated to assist general public.

- Deep learning algorithms that can study user's usage pattern and predict more suitable produces to increase self-engagement.
- Prediction models of plant diseases through which growers can study about diseases they encounter while involving in producing.
- Proper administrative panel to manage application content insights and get analytics should be implemented.
- Data cleanup mechanisms should be integrated.
- Furthermore a mobile application could be implemented using the exposed REST API endpoints to enable cross-platform availability.

REFERENCES

Allen, P., n.d. *The facts about food miles.* [Online]
Available at: https://www.bbcgoodfood.com/howto/guide/facts-about-food-miles
[Accessed 28 March 2022].

Amir, A., 2021. *Small-holding farmers and exploitative role of intermediaries.* [Online]
Available at: https://www.emeraldgrouppublishing.com/news-and-press-releases/small-holding-farmers-and-exploitative-role-intermediaries
[Accessed 13 March 2022].

Arrowquip, 2017. *TOP BENEFITS OF BUYING LOCALLY GROWN FOOD.* [Online]
Available at: https://arrowquip.com/blog/animal-science/top-benefits-buying-locally-grown-food
[Accessed 26 April 2022].

Bacinger, T., n.d. *What is Bootstrap? A Short Bootstrap Tutorial on the What, Why, and How.* [Online]
Available at: https://www.toptal.com/front-end/what-is-bootstrap-a-short-tutorial-on-the-what-why-and-how
[Accessed 25 October 2021].

Baeldung, 2021. *Implementing Simple State Machines with Java Enums.* [Online]
Available at: https://www.baeldung.com/java-enum-simple-state-machine
[Accessed 01 May 2022].

Berti, G., & Mulligan, C. (2016). Competitiveness of small farms and innovative food supply chains: The role of food hubs in creating sustainable regional and local food systems. Sustainability, 8(7), 616. https://doi.org/10.3390/su8070616

Bodin, Ö., Crona, B., & Ernstson, H. (2006). Social Networks in Natural Resource Management: What Is There to Learn from a Structural Perspective? *Ecology and Society*, *11*(2). Available at: http://www.jstor.org/stable/26266035
[Accessed 20 April 2022].

Bora, A. and Bezboruah, T., 2015. *A Comparative Investigation on Implementation of RESTful versus SOAP based Web Services*. 10.14257/ijdta.2015.8.3.26

Brush, K., 2019. *Agile Software Development.* [Online]
Available at: https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development
[Accessed 1 May 2022].

Bruce, A. B., & Som Castellano, R. L. (2017). Labor and alternative food networks: Challenges for farmers and consumers. Renewable Agriculture and Food Systems, 32(5), 403–416. https://doi.org/10.1017/s174217051600034x

Byker, C., Rose, N., & Serrano, E. (2010). The benefits, challenges, and strategies of adults following a local food diet. Journal of Agriculture, Food Systems, and Community Development, 1(1), 125–138. https://doi.org/10.5304/jafscd.2010.011.013

Chadha, N., 2019. *Are Sri Lanka's agricultural policies starving our farmers?.* [Online]
Available at: https://www.themorning.lk/are-sri-lankas-agricultural-policies-starving-our-farmers/
[Accessed 5 April 2022].

Chamara, S., 2019. *Post-harvest crop wastage in Sri Lanka.* [Online]
Available at: https://www.themorning.lk/post-harvest-crop-wastage-in-sri-lanka/
[Accessed 15 March 2022].

Codeinstitute, 2022. *What is Java and why is it important?.* [Online]
Available at: https://codeinstitute.net/blog/what-is-java/

Croptronix, 2019. *Govipola.* [Online]
Available at: http://www.govipola.lk/en
[Accessed 17 March 2022].

Dentzel, Z., n.d. *How the Internet Has Changed Everyday Life.* [Online]
Available at: https://www.bbvaopenmind.com/en/articles/internet-changed-everyday-life/
[Accessed 20 April 2022].

Elgabry, O., 2017. *Software Engineering — Software Process and Software Process Models (Part 2).* [Online]
Available at: https://medium.com/omarelgabrys-blog/software-engineering-software-process-and-software-process-models-part-2-4a9d06213fdc
[Accessed 23 April 2022].

Fol, P., 2021. *Spring vs. the World: Comparing Spring Boot Alternatives.* [Online]
Available at: https://www.jrebel.com/blog/spring-boot-alternatives
[Accessed 20 May 2022].

Gadekar, P., Swati, B., Dhanashri, B. and Smita, B., 2016. *E-Farming*. International Journal for Research in Applied Science and Engineering Technology (IJRASET), [online] 4(1), pp.204-207. Available at: https://www.ijraset.com/fileserve.php?FID=3773 [Accessed 12 March 2022].

Ghogare, S. and Monga, P., 2015. International journal of advanced research in computer science and software engineering. *International Journal of Advanced Research in Computer Science and Software Engineering*, [online] 5(1), pp.44-47. Available at: http://www.ijarcsse.com [Accessed 10 April 2022].

Hamed, O. and Kafri, N., 2009. Performance testing for web based application architectures (.NET vs. Java EE). *2009 First International Conference on Networked Digital Technologies,*. 10.1109/ndt.2009.5272178

IBM, 2019. *What is Java Used For?.* [Online]
Available at: https://www.ibm.com/cloud/learn/java-explained
[Accessed 20 AMay 2022].

IBM, 2020. *Java Spring Boot.* [Online]
Available at: https://www.ibm.com/cloud/learn/java-spring-boot#:~:text=Spring%20Boot%20helps%20developers%20create,app%20during%20the%20initialization%20process.
[Accessed 10 May 2022].

Jayathilake, H., Jayaweera, B. and Waidyasekera, E., 2010. ICT Adoption and Its' Implications for Agriculture in Sri Lanka. *Journal of Food and Agriculture*, 1(2), 10.4038/jfa.v1i2.1799

Jahufer, A., 2022. *Finding suitable statistical model for gross domestic product and agriculture related factors*. [online] Ir.lib.seu.ac.lk. Available at: http://ir.lib.seu.ac.lk/handle/123456789/2352 [Accessed 5 April 2022].

Journey North, 2019. *Understanding Latitude and Longitude.* [Online]
Available at: https://journeynorth.org/tm/LongitudeIntro.html
[Accessed 01 June 2022].

Kolodiy, S., 2021. *Unit Tests, How to Write Testable Code and Why it Matters.* [Online]
Available at: https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters
[Accessed 20 May 2022].

Local Food Nodes, 2017. *Local Food Nodes.* [Online]
Available at: https://localfoodnodes.org/en
[Accessed 21 March 2022].

Munonye, K. and Martinek, P., 2018. Performance Analysis of the Microsoft. Net- and Java-Based Implementation of REST Web Services. *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY),* 10.1109/SISY.2018.8524705

NobleDesktop, 2018. *What Is Git & Why Should You Use It?.* [Online]
Available at: nobledesktop.com/blog/what-is-git-and-why-should-you-use-it#:~:text=Git%20is%20the%20most%20commonly,be%20merged%20into%20one%20source. [Accessed 26 April 2022].

Opitz, I., Zoll, F., Zasada, I., Doernberg, A., Siebert, R. and Piorr, A., 2019. Consumer-producer interactions in community-supported agriculture and their relevance for economic stability of the farm – An empirical study using an Analytic Hierarchy Process. *Journal of Rural Studies*, 68, pp.22-32. 10.1016/j.jrurstud.2019.03.011 [Accessed 22 March 2022].

Perrett, A., & Jackson, C., 2015. Local food, food democracy, and food hubs. Journal of Agriculture, Food Systems, and Community Development, 6(1), 7–18.
https://doi.org/10.5304/jafscd.2015.061.003

Pramod, G., Baghat, S., Bhusare, D., Botre, S and Patel, S., 2016. 'E-farming', International Journal for Research in Applied Science and Engineering Technology, 4(1), pp. 204–207.
Available at: http://www.ijraset.com/fileserve.php? FID=3773. [Accessed 15 March 2022].

Rehan, A., 2022. *12 Best Geocoding and Maps API Solution for Your Applications.* [Online]
Available at: https://geekflare.com/geocoding-maps-api-solution/
[Accessed 20 May 2022].

Ranga, V. and Soni, A., 2019. API Features Individualizing of Web Services: REST and SOAP. *International Journal of Innovative Technology and Exploring Engineering*, 8(9S), pp.664-671. 10.35940/ijitee.I1107.0789S19

Seed Code Labs, 2019. *weladapola.* [Online]
Available at: https://weladapola.lk/
[Accessed 20 March 2022].

Stobierski, T., 2021. *WHAT IS GDP & WHY IS IT IMPORTANT?.* [Online]
Available at: https://online.hbs.edu/blog/post/why-is-gdp-important
[Accessed 6 April 2022].

Synopsys, n.d. *Synopsys.* [Online]
Available at: https://www.synopsys.com/glossary/what-is-agile-sdlc.html
[Accessed 23 March 2022].

Sulemani, M., 2021. *What is a software process model? Top 7 models explained.* [Online]
Available at: https://www.educative.io/blog/software-process-model-types
[Accessed 20 April 2022].

Trobe, H., 2008. Farmers' markets: consuming local rural produce. *International Journal of Consumer Studies,* 25(3), pp.181-192. https://doi.org/10.1046/j.1470-6431.2001.00171.x

United Nations, 2019. *Sri Lanka Population.* [Online]
Available at: https://www.worldometers.info/world-population/sri-lanka-population/
[Accessed 20 April 2022].

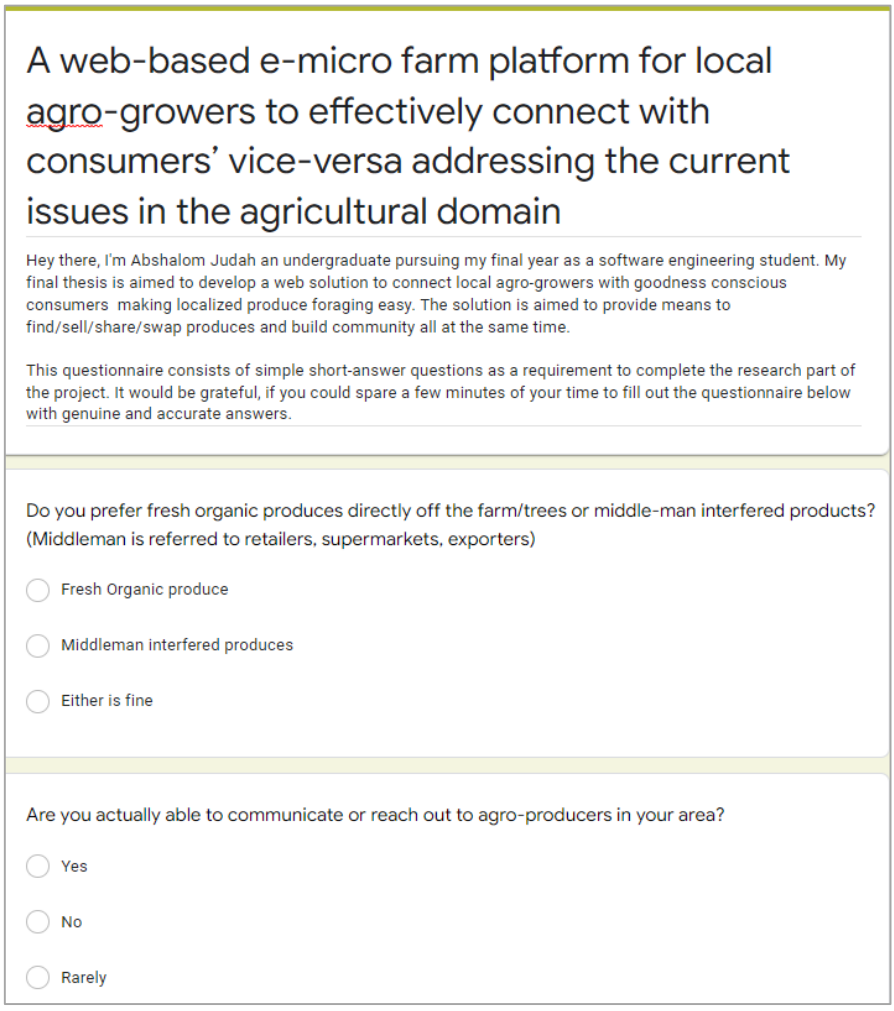
Walker, A., 2018. *8 Best Java IDEs for 2019.* [Online]
Available at: https://www.g2.com/articles/java-ide
[Accessed 28 May 2022].

Webber, C. B., & Dollahite, J. S., 2008. Attitudes and behaviors of low-income food heads of households toward sustainable food systems concepts. Journal of Hunger and Environmental Nutrition, 3(2/3), 187–205. https://doi.org/10.1080/19320240802243266

APPENDICES

Appendix A: Requirement gathering

Complete Questionnaire



A web-based e-micro farm platform for local agro-growers to effectively connect with consumers' vice-versa addressing the current issues in the agricultural domain

Hey there, I'm Abshalom Judah an undergraduate pursuing my final year as a software engineering student. My final thesis is aimed to develop a web solution to connect local agro-growers with goodness conscious consumers making localized produce foraging easy. The solution is aimed to provide means to find/sell/share/swap produces and build community all at the same time.

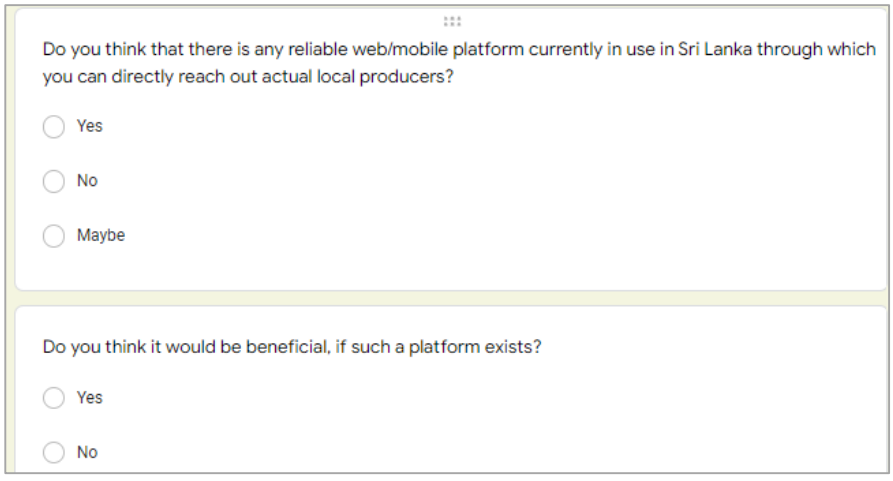
This questionnaire consists of simple short-answer questions as a requirement to complete the research part of the project. It would be grateful, if you could spare a few minutes of your time to fill out the questionnaire below with genuine and accurate answers.

Do you prefer fresh organic produces directly off the farm/trees or middle-man interfered products? (Middleman is referred to retailers, supermarkets, exporters)

- Fresh Organic produce
- Middleman interfered produces
- Either is fine

Are you actually able to communicate or reach out to agro-producers in your area?

- Yes
- No
- Rarely



Do you think that there is any reliable web/mobile platform currently in use in Sri Lanka through which you can directly reach out actual local producers?

- Yes
- No
- Maybe

Do you think it would be beneficial, if such a platform exists?

- Yes
- No

Would you be interested to be notified via E-mail when there is excess harvest given away free within a neighborhood?

- Yes
- No
- Maybe

If such system exists, would you be motivated to at least use up your home garden or unused lands to involve in producing? (So that you can maybe earn on a part time scale and also even swap with other interested produces from other producers)

- Yes
- No
- Maybe

Do you agree that we can become a self-sufficient country when dependency on local agriculture is given a recognition and ultimately become a contributing factor to solve the economic crisis? *

- Yes
- No

Would you like to suggest any special features you expect to be present in this system?

Long answer text

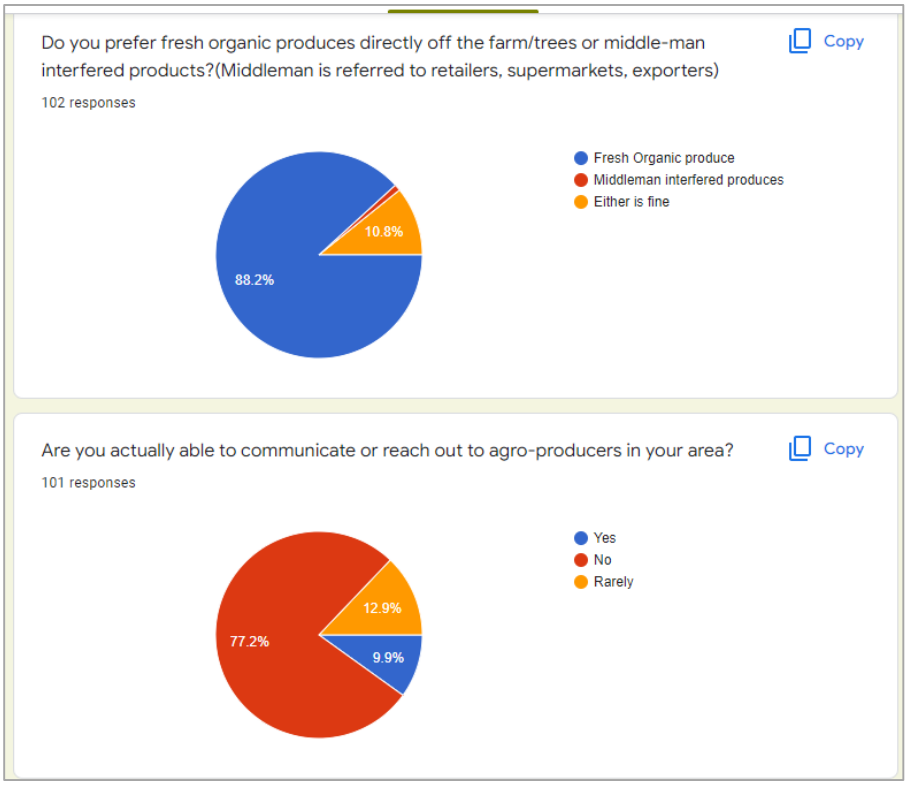
Do you agree that we can become a self-sufficient country when dependency on local agriculture is given a recognition and ultimately become a contributing factor to solve the economic crisis? *

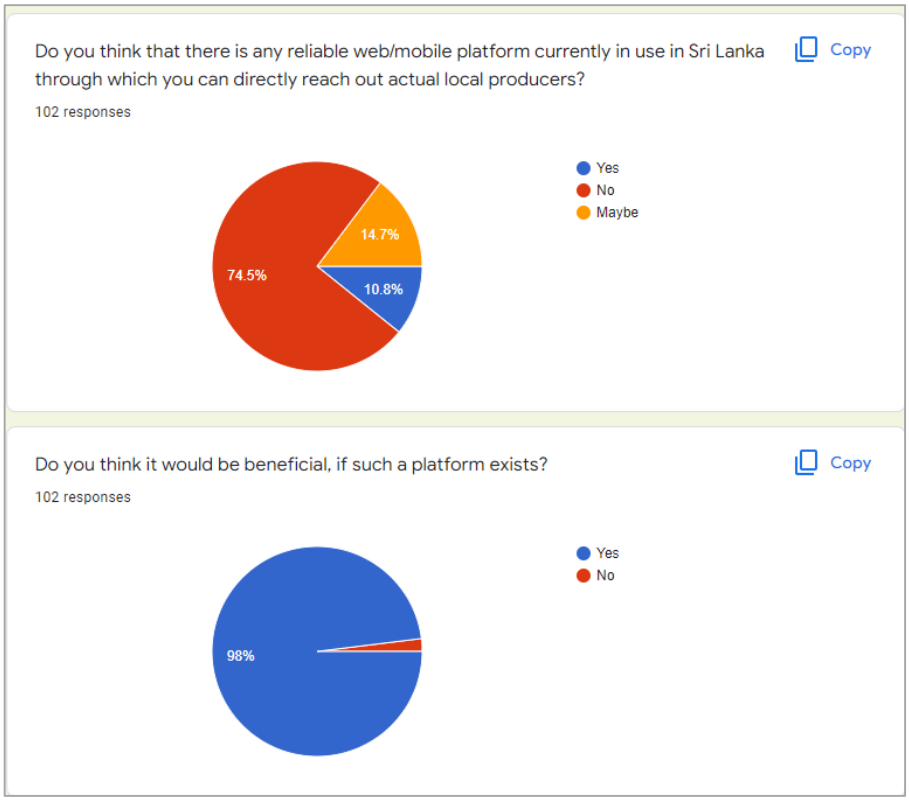
- Yes
- No

Would you like to suggest any special features you expect to be present in this system?

Long answer text

Complete Questionnaire responses

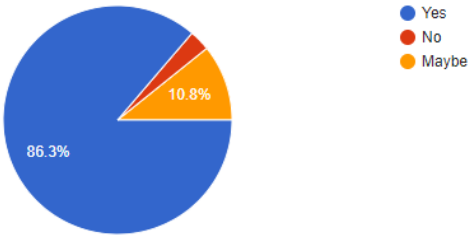




Would you be interested to be notified via E-mail when there is excess harvest given away free within a neighborhood?

Copy

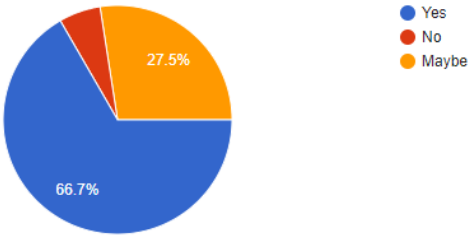
102 responses



If such system exists, would you be motivated to at least use up your home garden or unused lands to involve in producing? (So that you can maybe earn on a part time scale and also even swap with other interested produces from other producers)

Copy

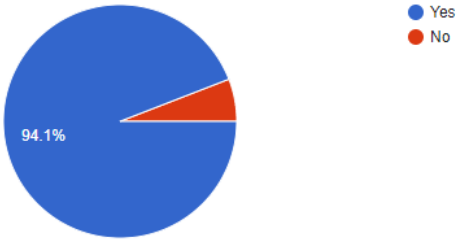
102 responses

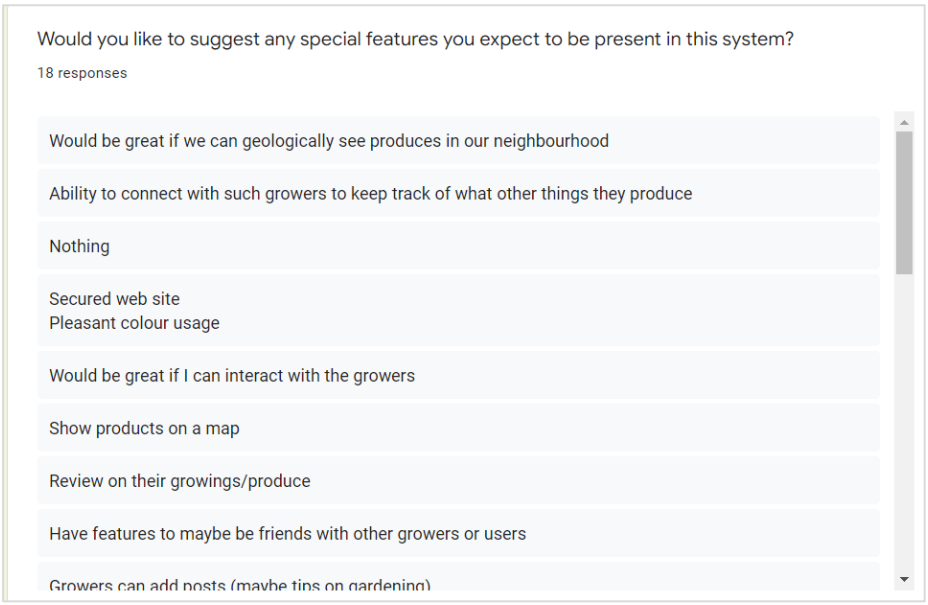


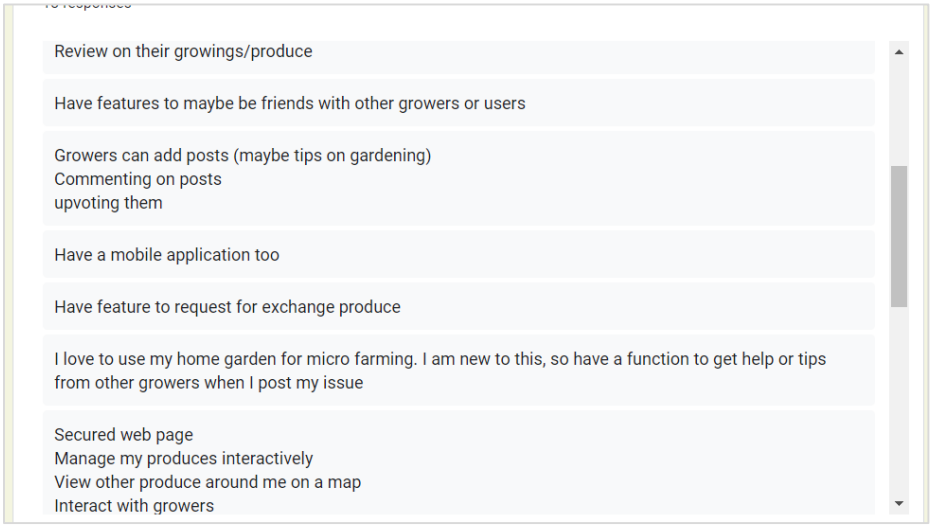
Do you agree that we can become a self-sufficient country when dependency on local agriculture is given a recognition and ultimately become a contributing factor to solve the economic crisis?

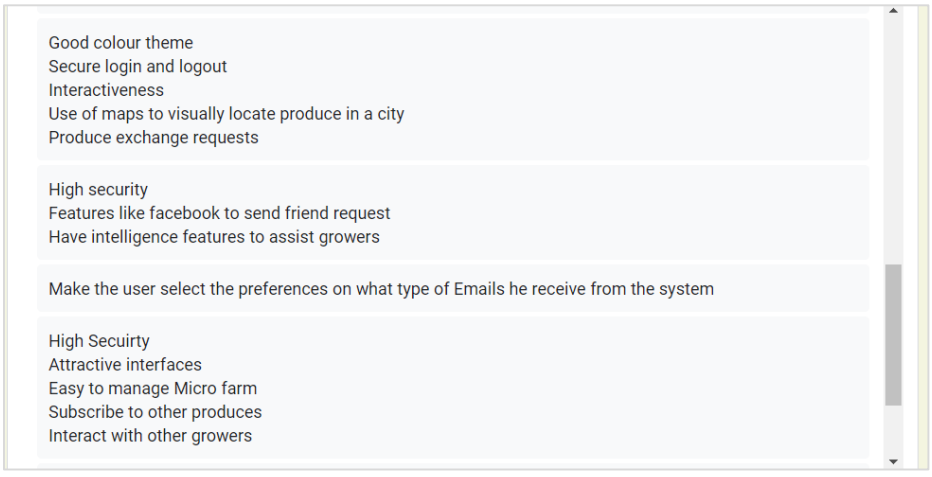
Copy

102 responses



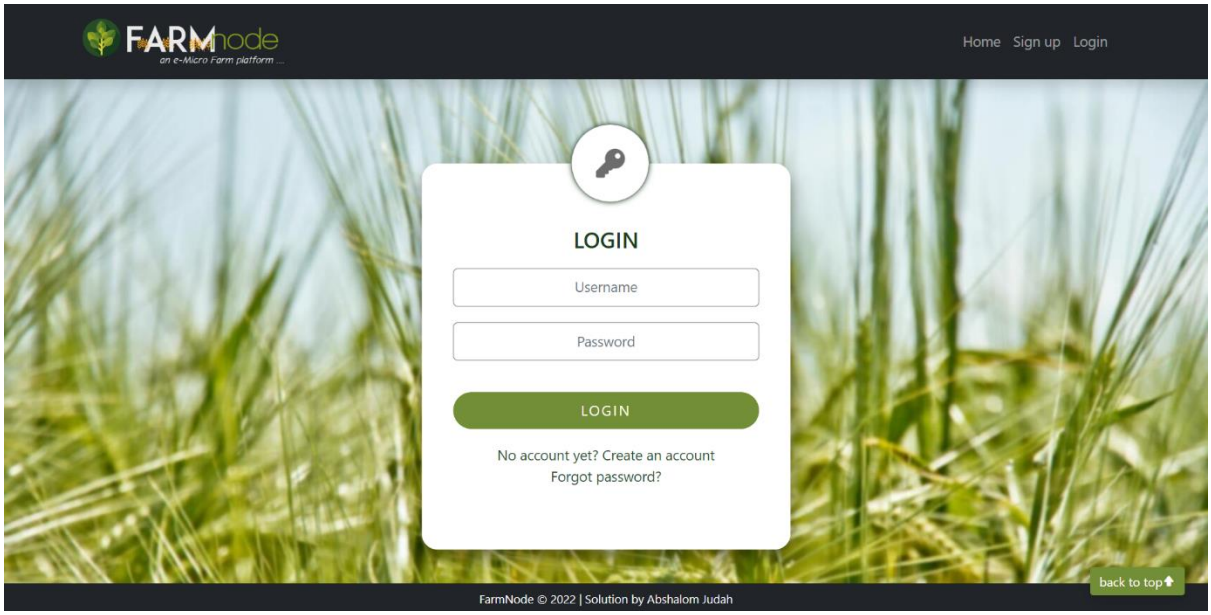

Would you like to suggest any special features you expect to be present in this system?
18 responses
Would be great if we can geologically see produces in our neighbourhood
Ability to connect with such growers to keep track of what other things they produce
Nothing
Secured web site
Pleasant colour usage
Would be great if I can interact with the growers
Show products on a map
Review on their growings/produce
Have features to maybe be friends with other growers or users
Growers can add posts (maybe tips on gardening)


Review on their growings/produce
Have features to maybe be friends with other growers or users
Growers can add posts (maybe tips on gardening)
Commenting on posts
upvoting them
Have a mobile application too
Have feature to request for exchange produce
I love to use my home garden for micro farming. I am new to this, so have a function to get help or tips from other growers when I post my issue
Secured web page
Manage my produces interactively
View other produce around me on a map
Interact with growers

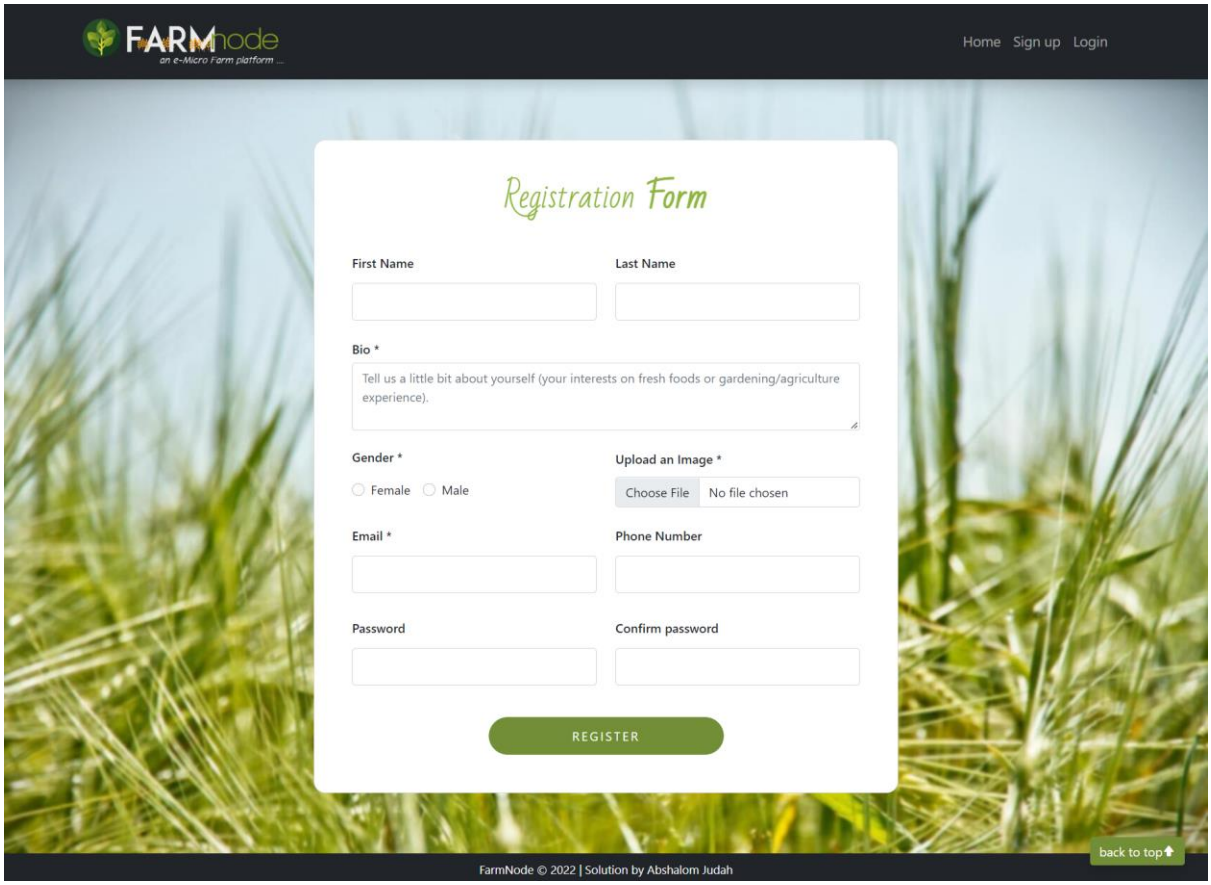

Good colour theme
Secure login and logout
Interactiveness
Use of maps to visually locate produce in a city
Produce exchange requests
High security
Features like facebook to send friend request
Have intelligence features to assist growers
Make the user select the preferences on what type of Emails he receive from the system
High Secuirty
Attractive interfaces
Easy to manage Micro farm
Subscribe to other produces
Interact with other growers

Appendix B: Core UI Implementation

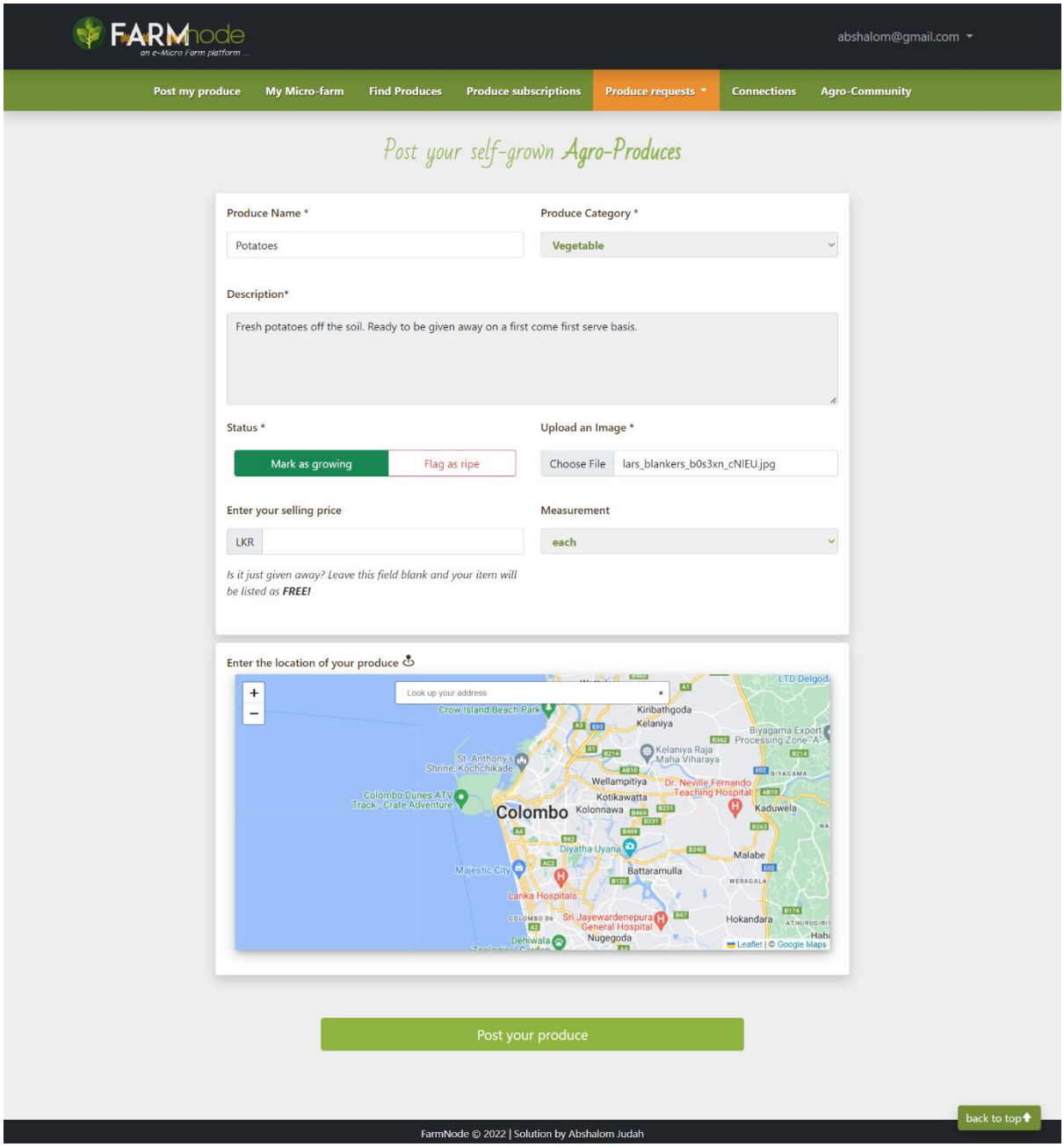
❖ Login - UI



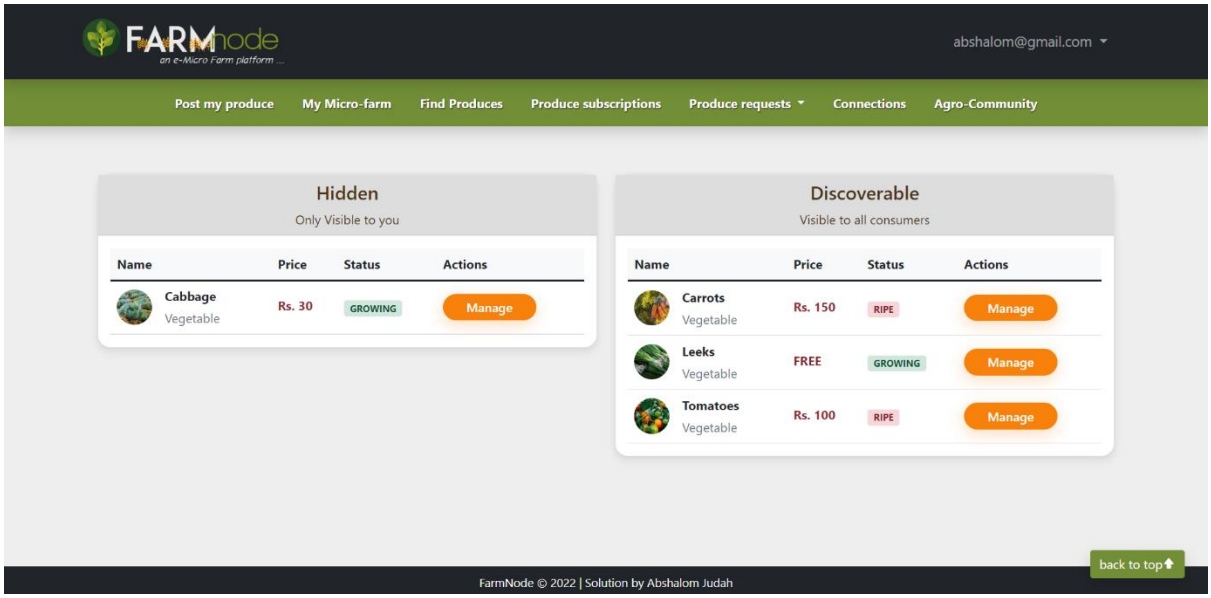
❖ Registration - UI



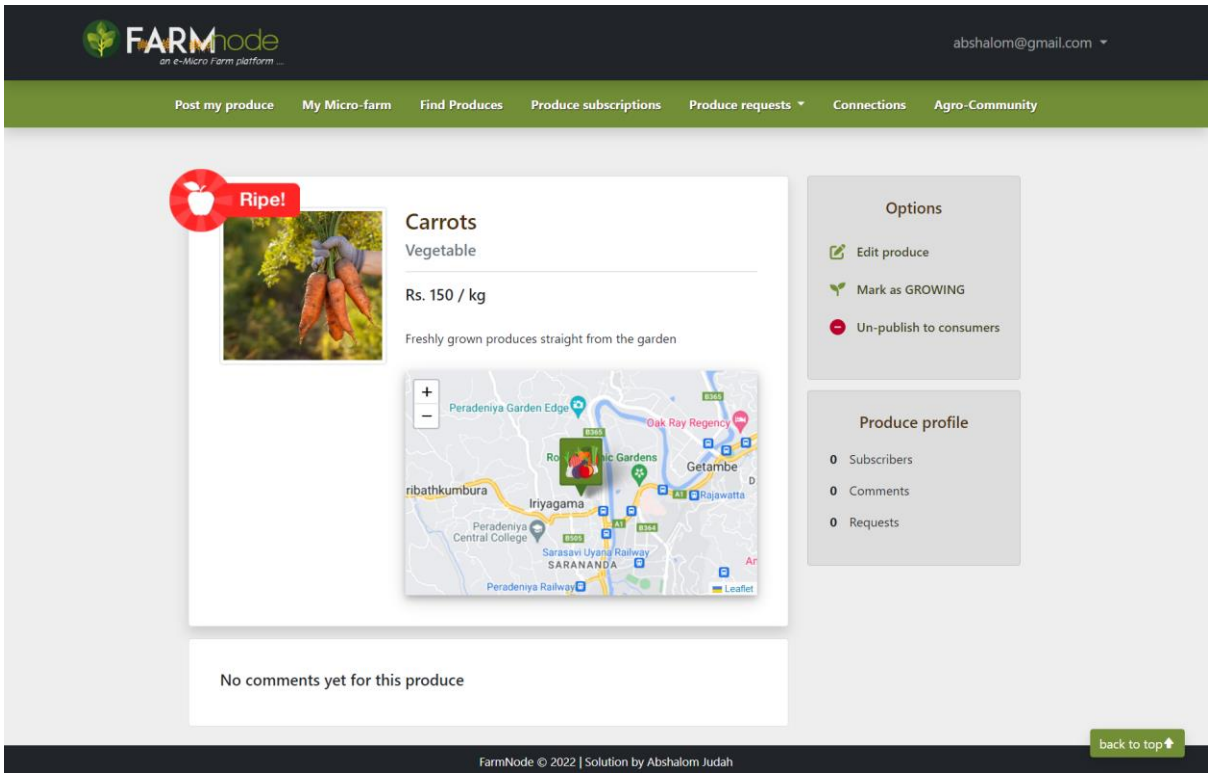
❖ Post produces - UI



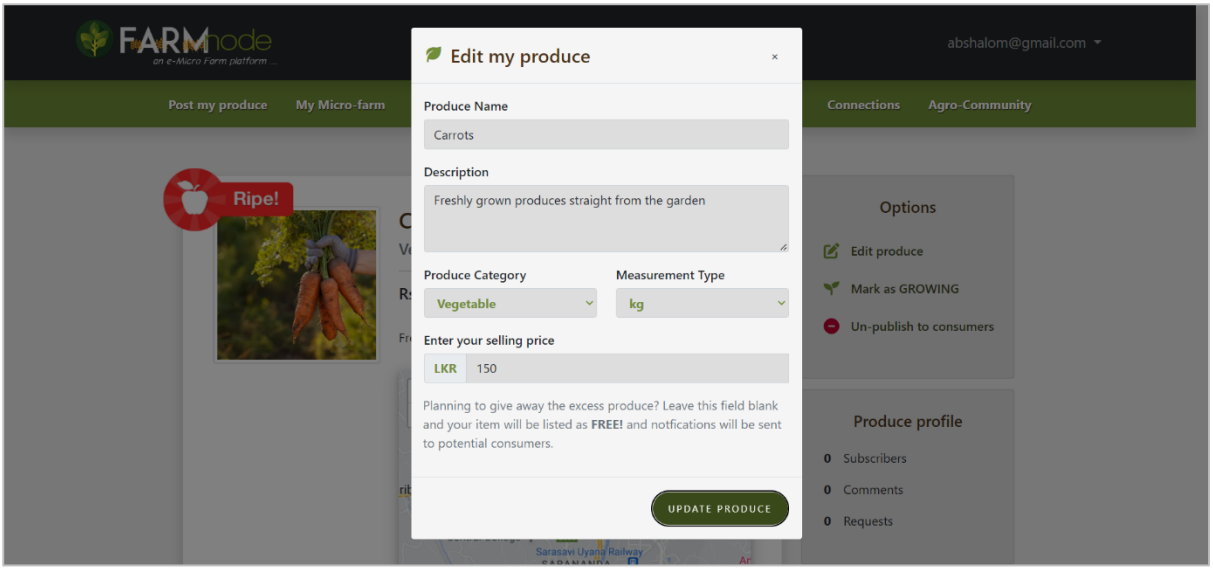
❖ E-micro farm dashboard



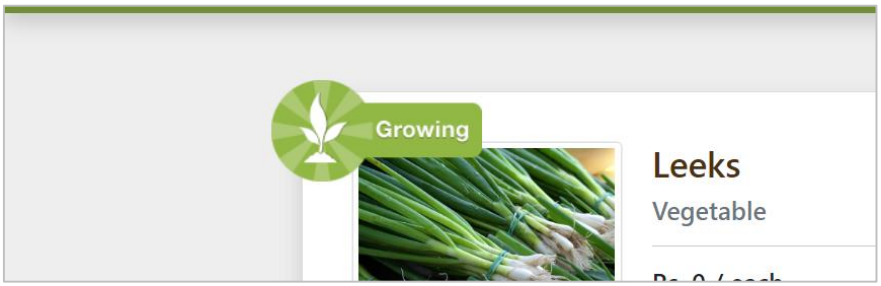
❖ View my produce

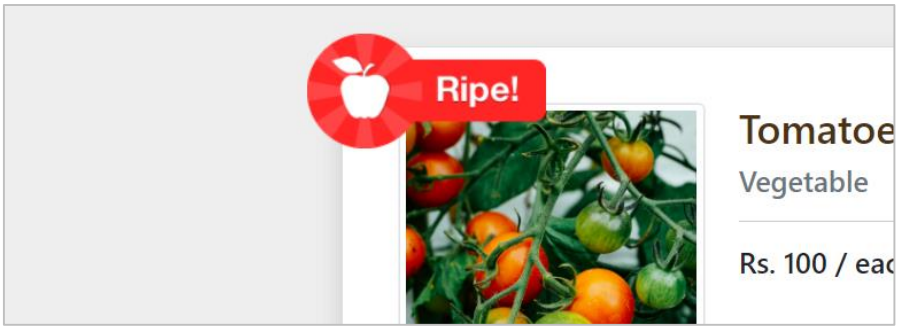


❖ **Update produce - UI**

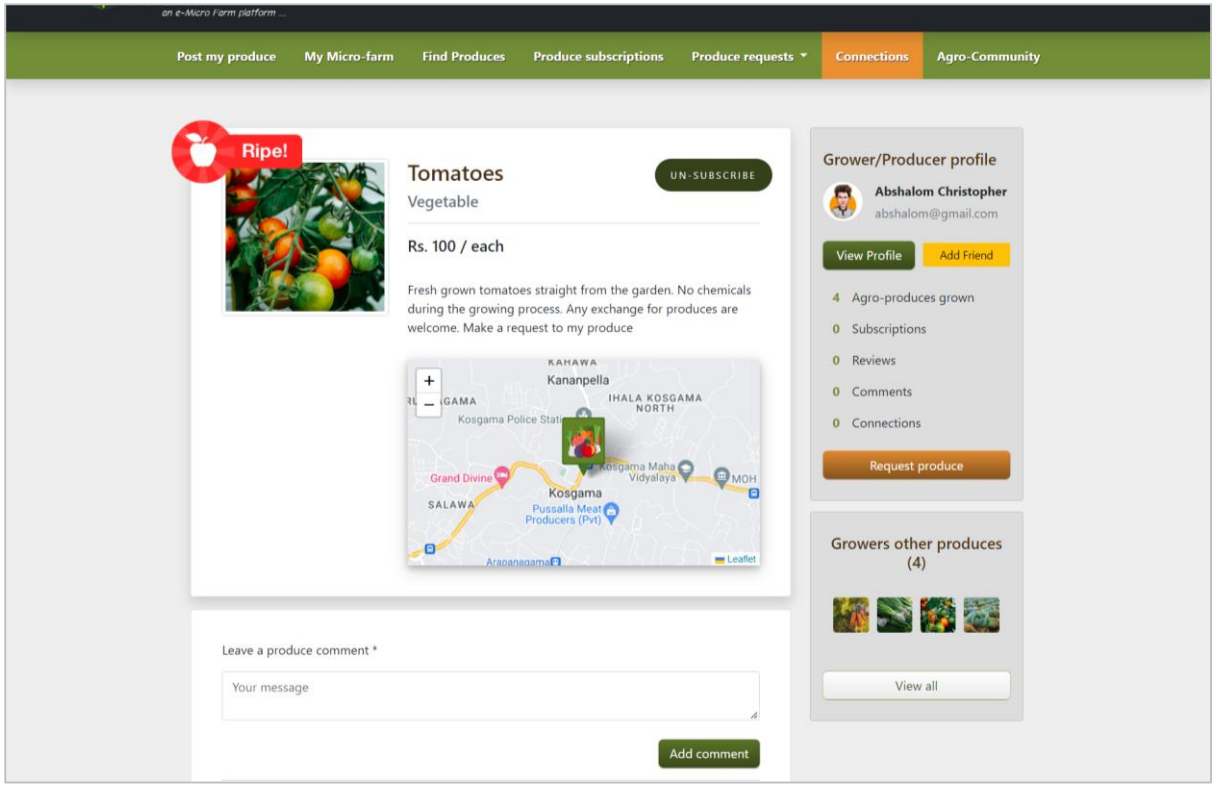


❖ **Growing vs Ripe statuses maintained**

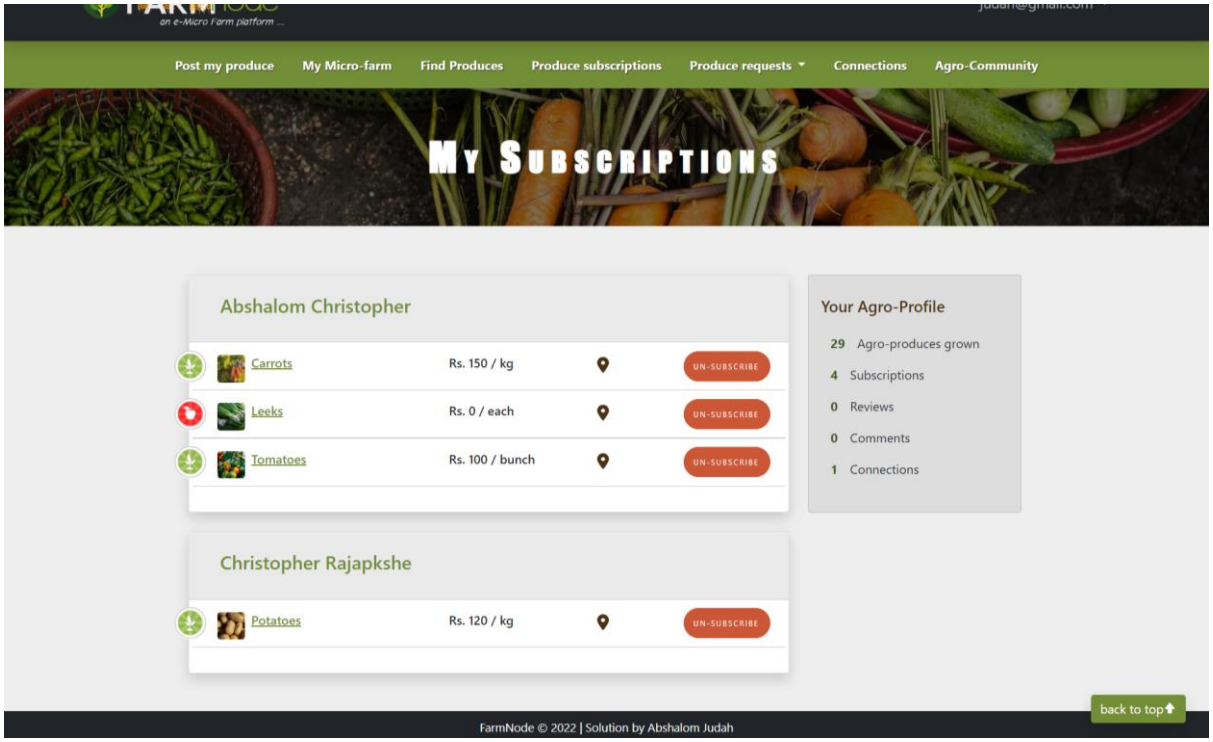




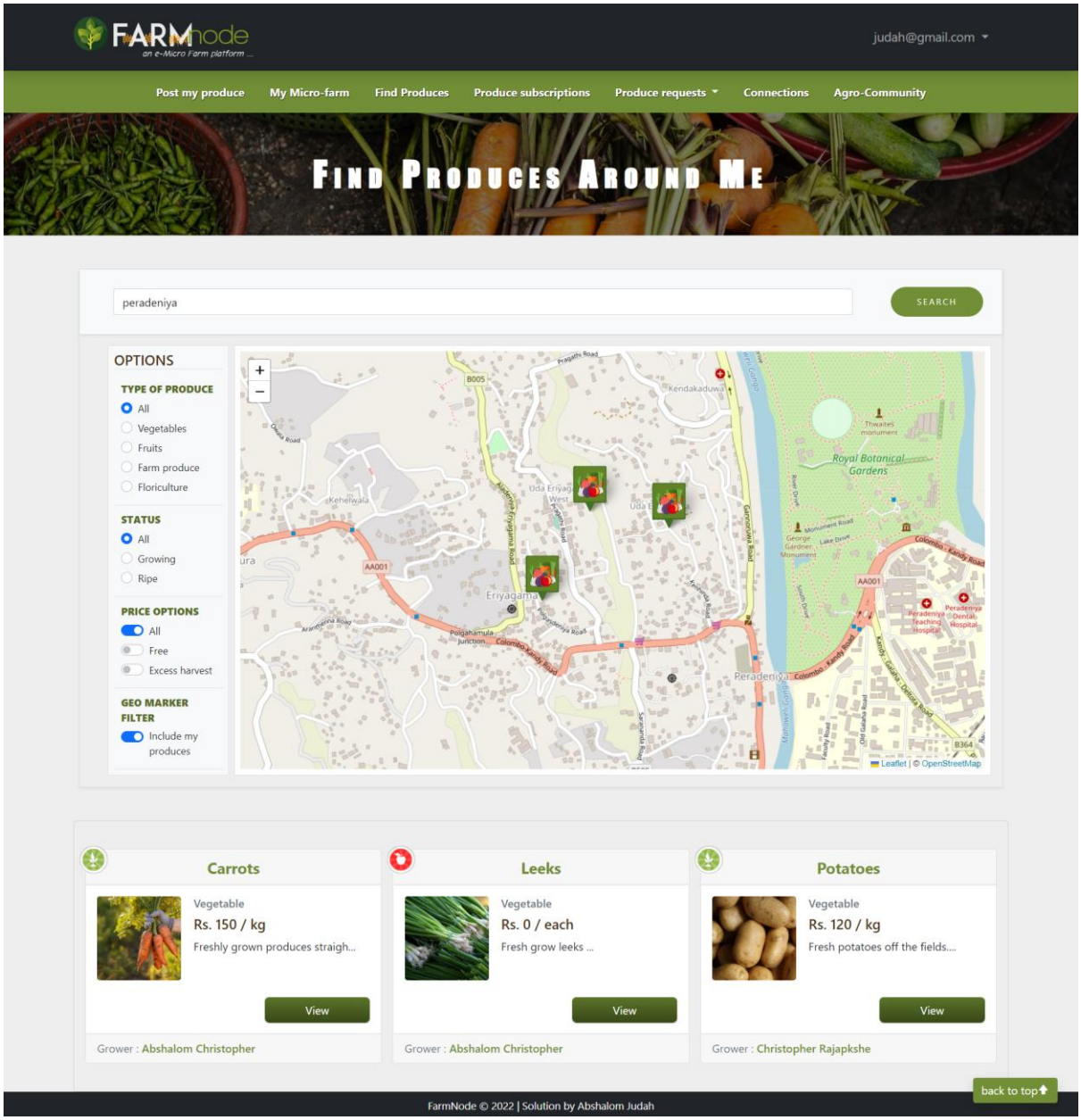
❖ **Others produce profile**



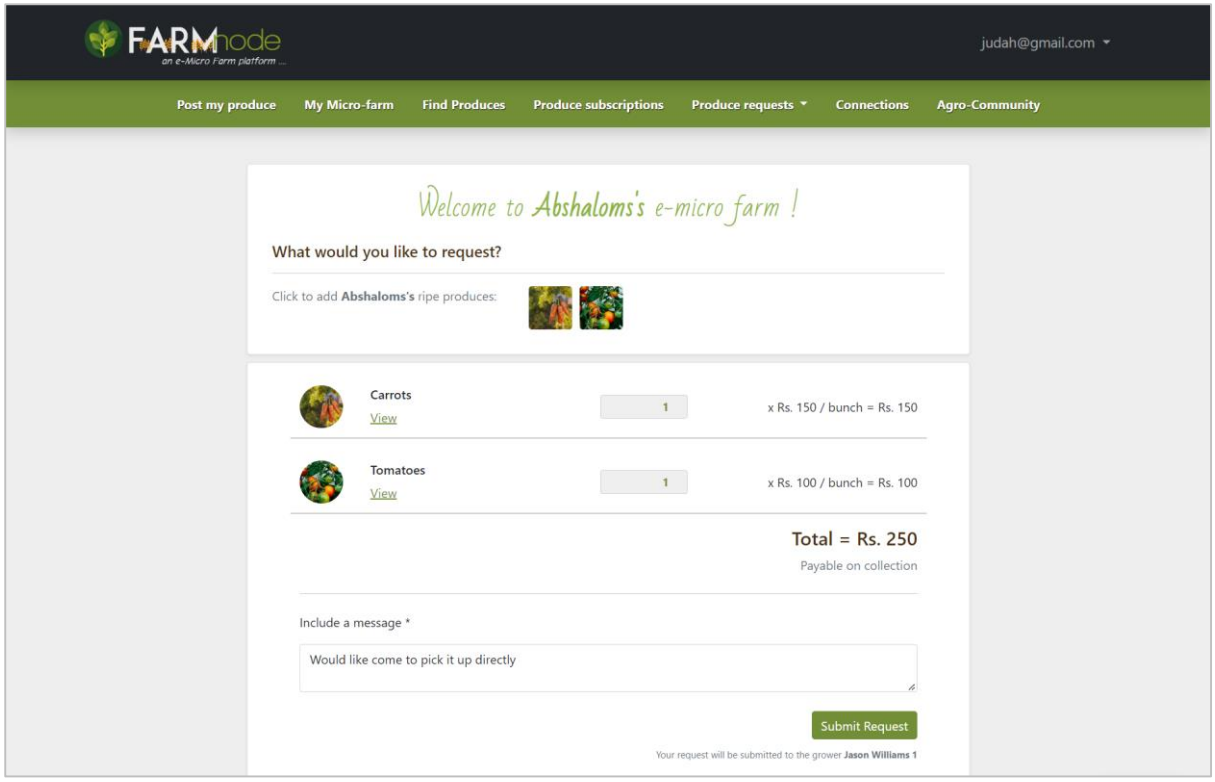
❖ **My produce subscriptions panel**



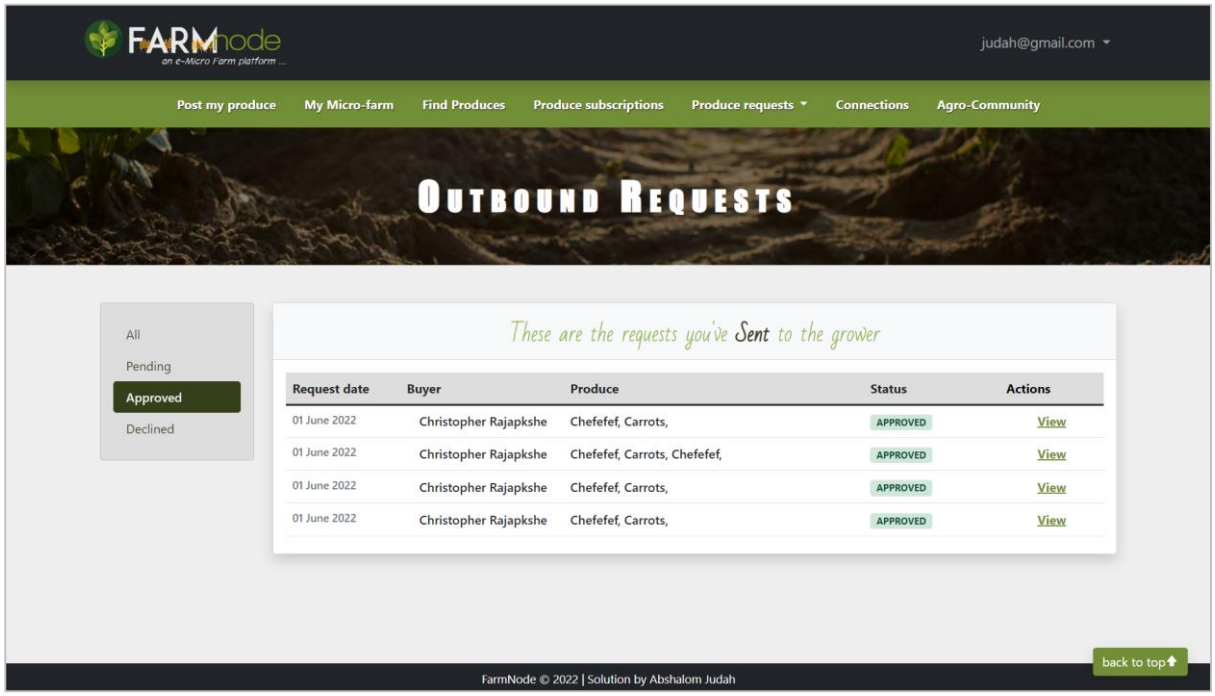
❖ **Find produces/geo-visualization**



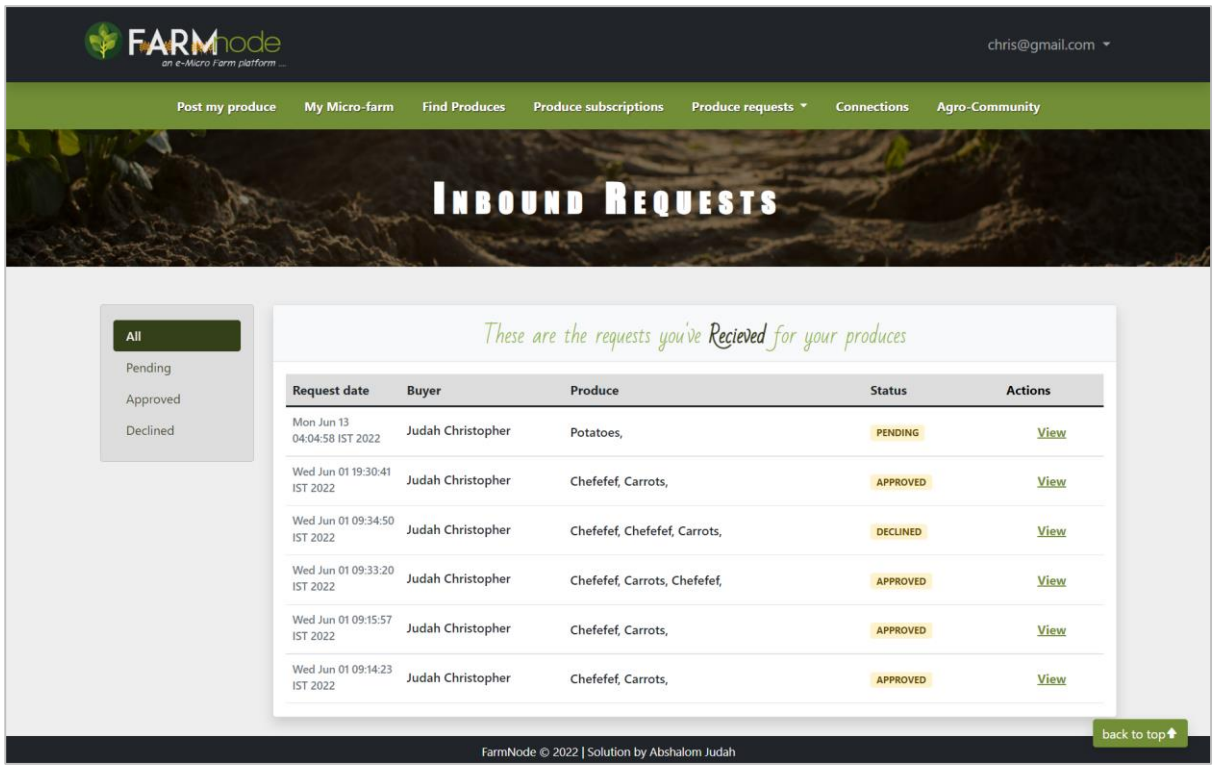
❖ Making a produce request

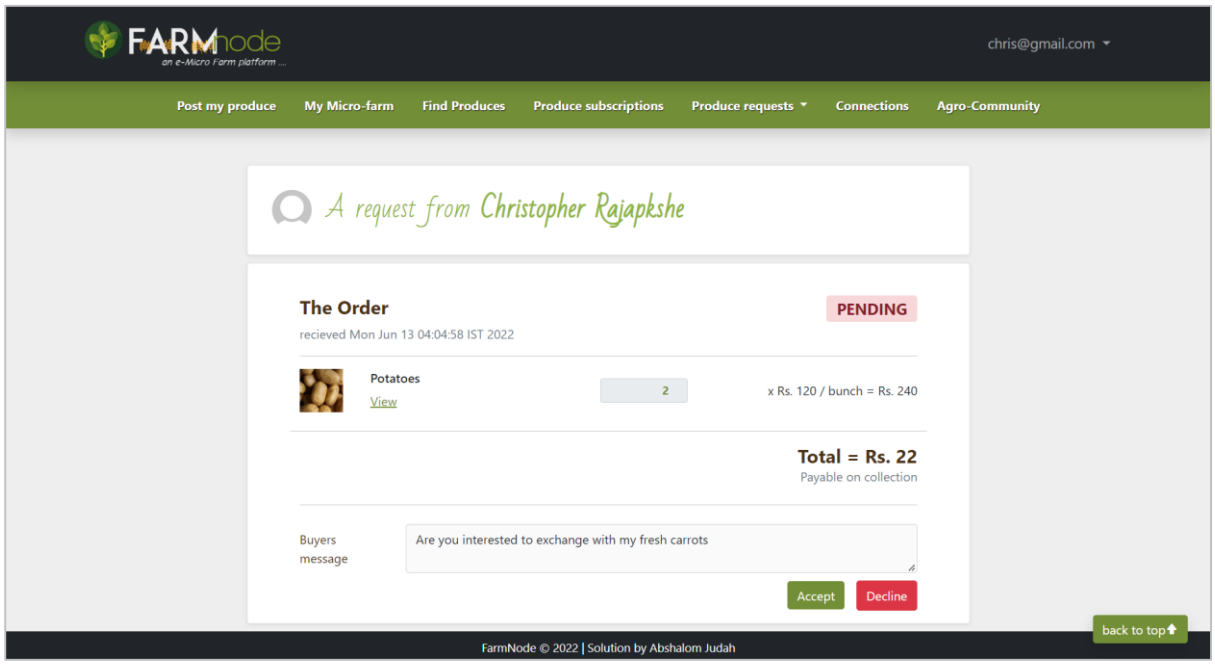


❖ Managing sent requests

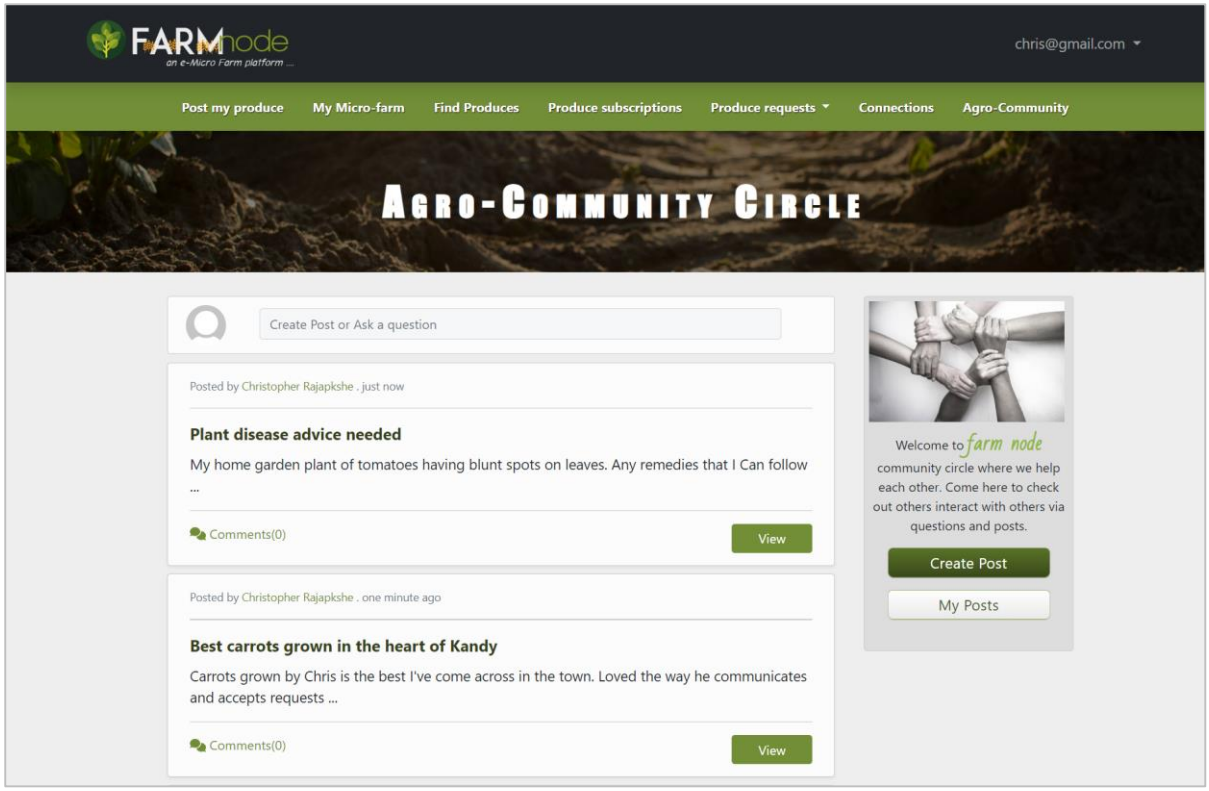


❖ Managing received request

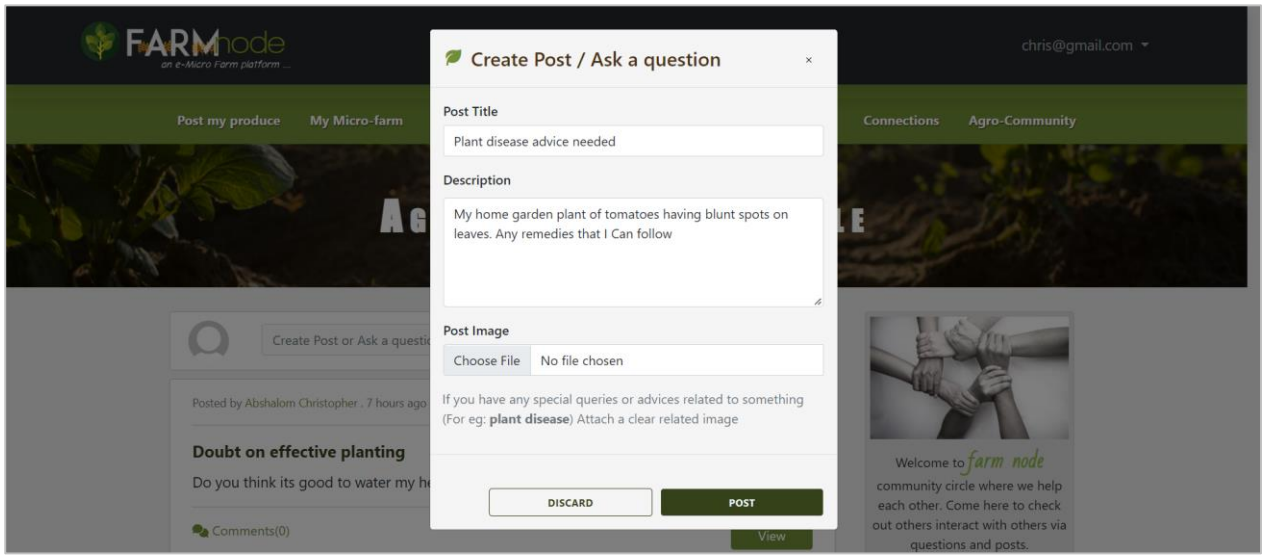




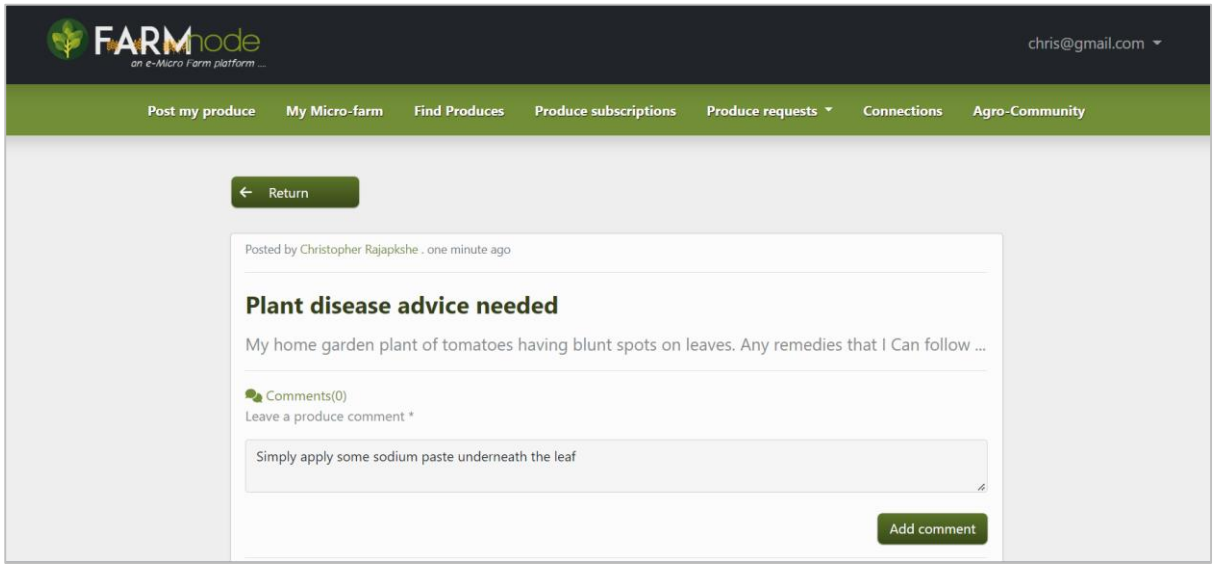
❖ **Interactive community UI**

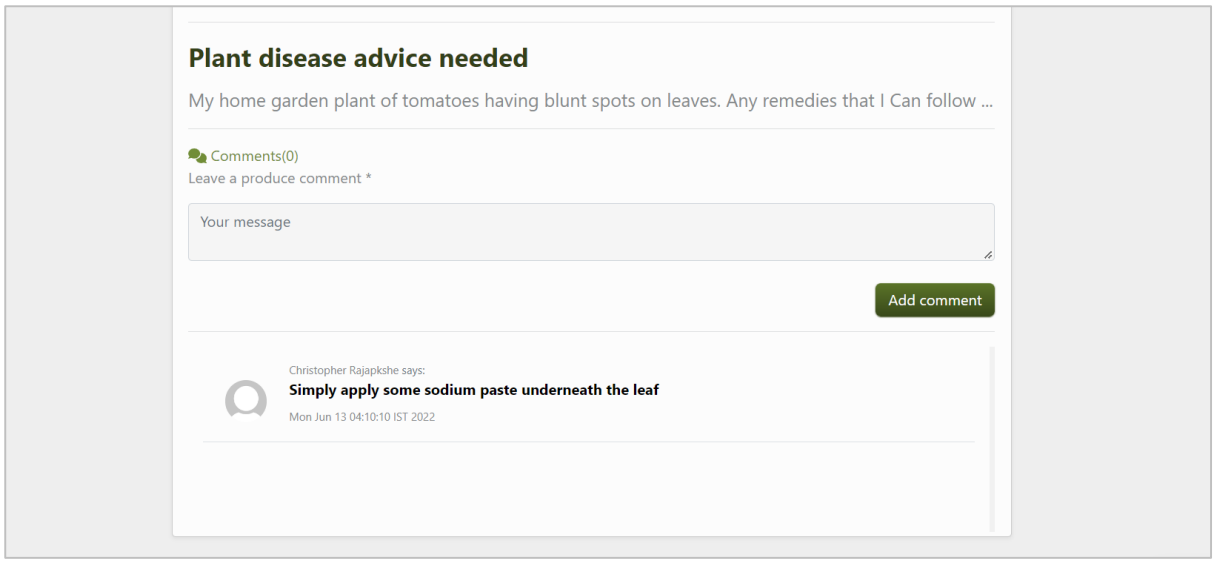


❖ **Posting a question / sharing**



❖ Interacting on others questions or posts





Appendix C: Subsidiary Implementation logics

Exception Handling

Any exception or errors may disrupt the normal flow of the application and may sometimes tend to be vague in reason. Therefore handling such errors in API's and sending proper response makes this project a user-friendly effective system. Thus all expected exceptions are handled individually. Each classes for a particular exception is defined and are defined in a global handler class to specify the exception constructors.

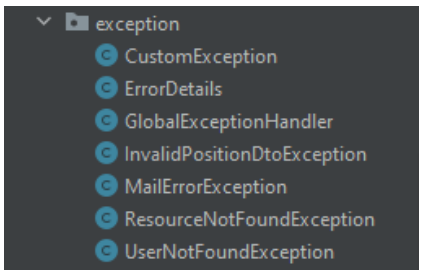
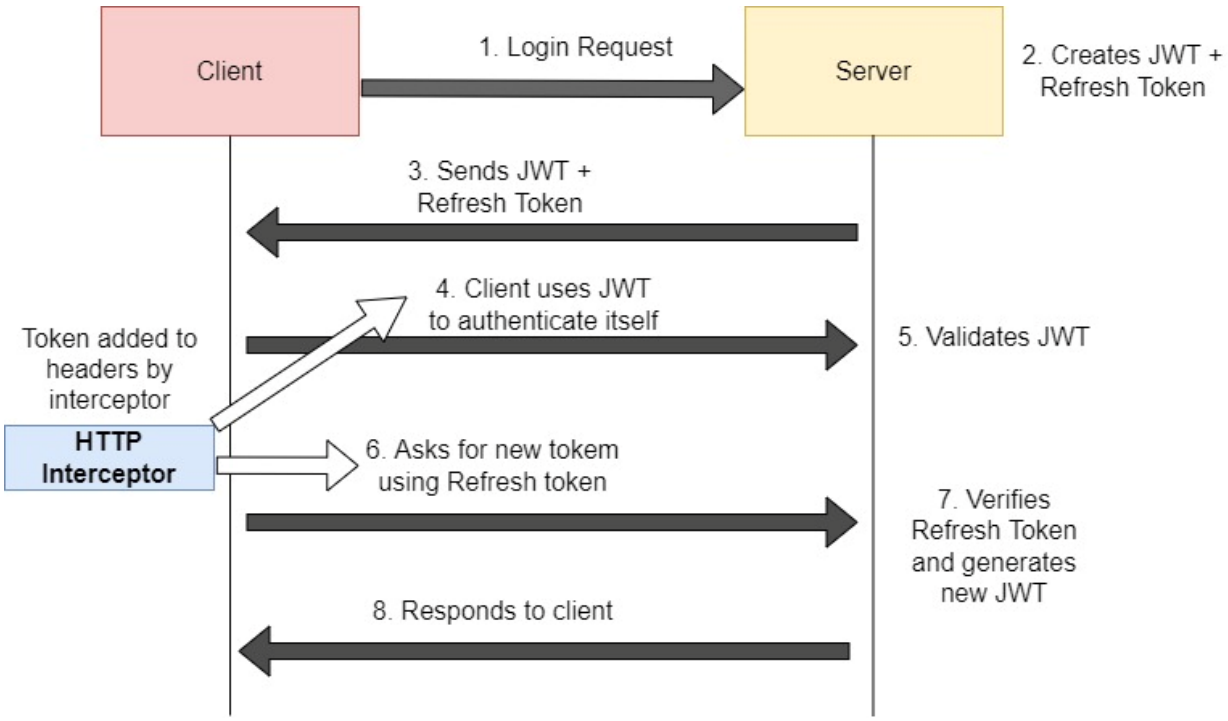


Figure 59 Exception-handling classes in project

Below is a code sample of such approach used to create an exception and use it in a business logic layer expecting the possible error that could occur,

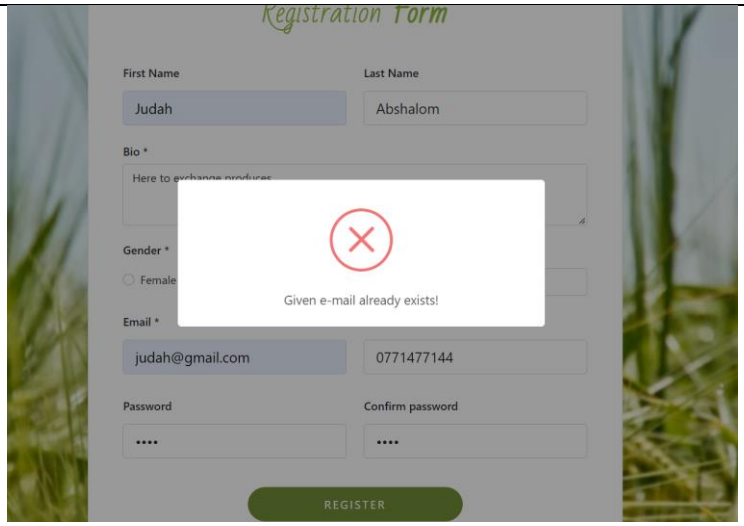
Figure 60 Exception-handling sample

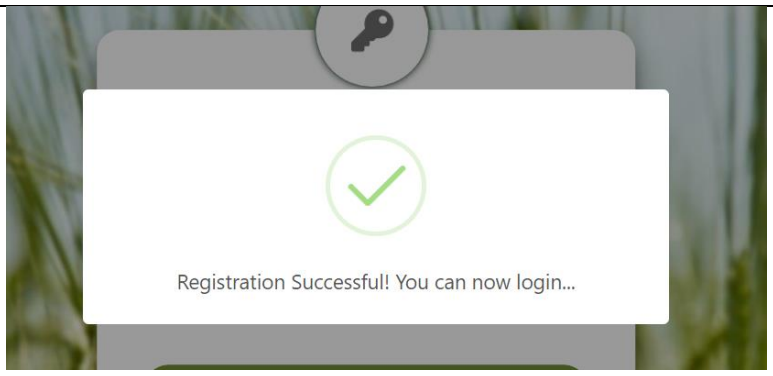
❖ **Client-Server request response flow with JWT token**

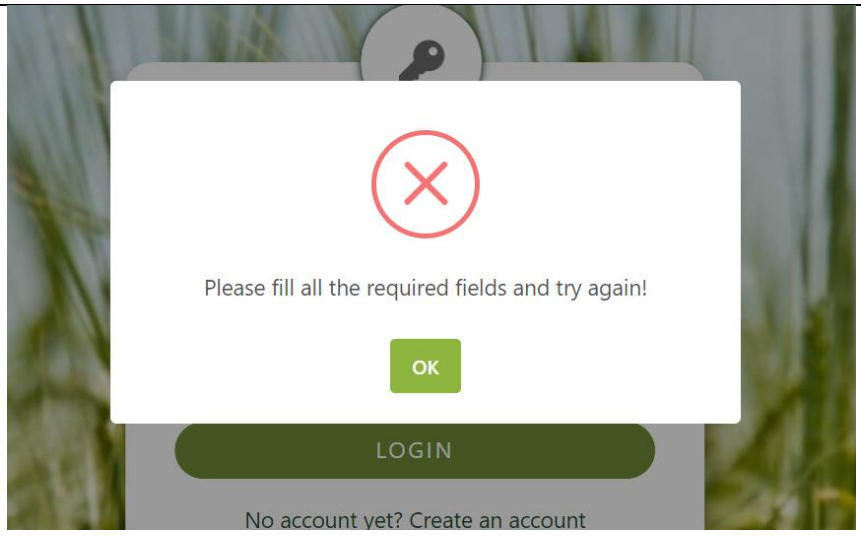


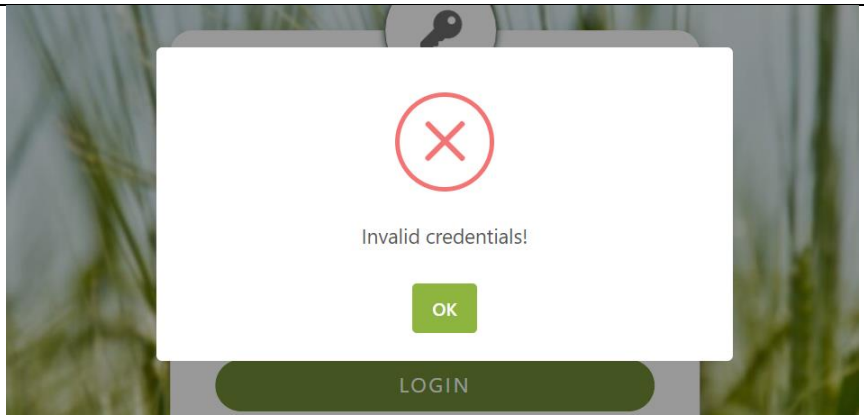
Appendix D: Testing

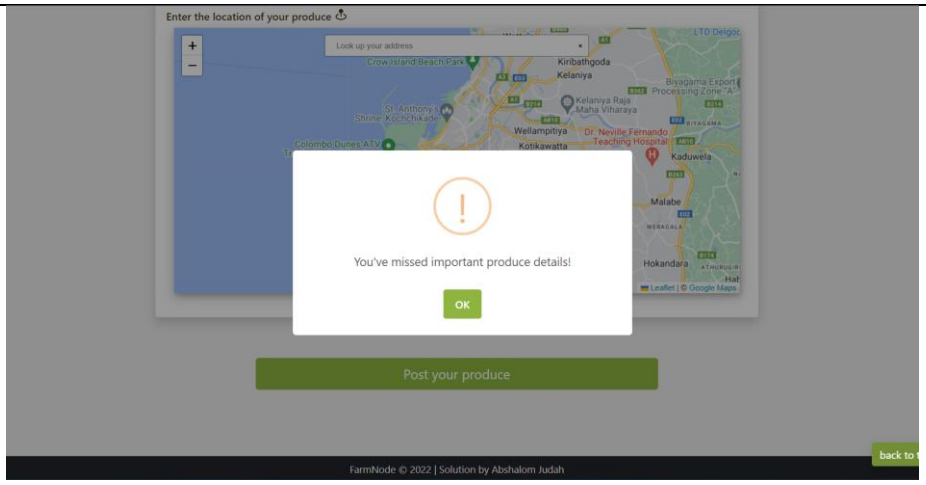
Test cases for overall system testing

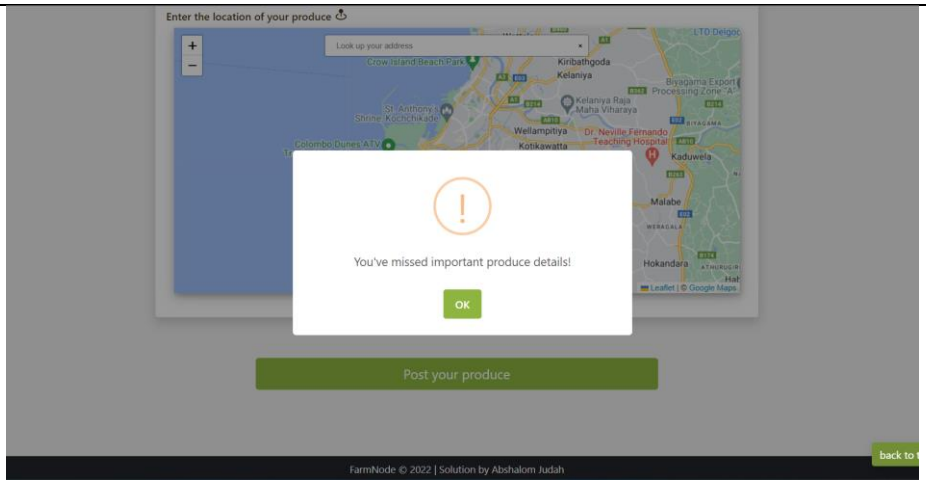
Test Case ID	TC001
Test objective	Registering with existing email as username
Test data	Enter ‘judah@gmail.com’ and click on ‘Register’
Expected Result	Display error message of existence of email
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

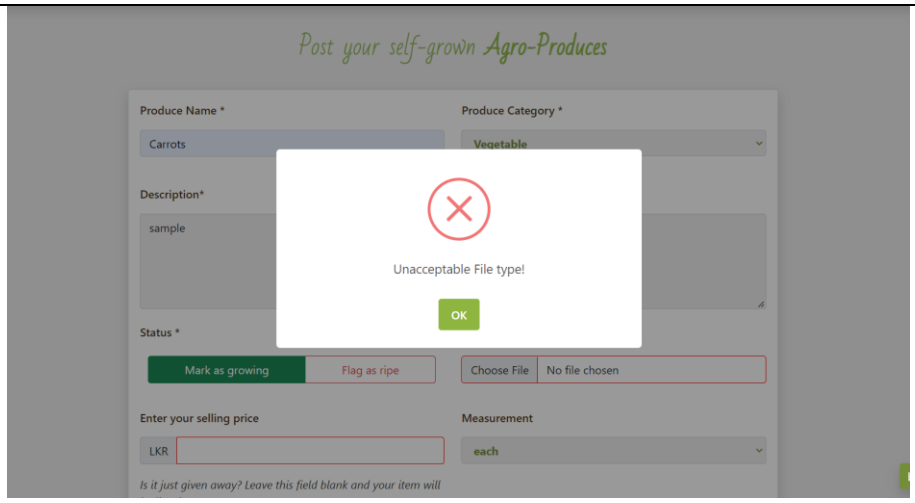
Test Case ID	TC002
Test objective	Registering with existing email as username
Test data	Enter ‘abshalom@gmail.com’ and click ‘Register’
Expected Result	Display successful registration message
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

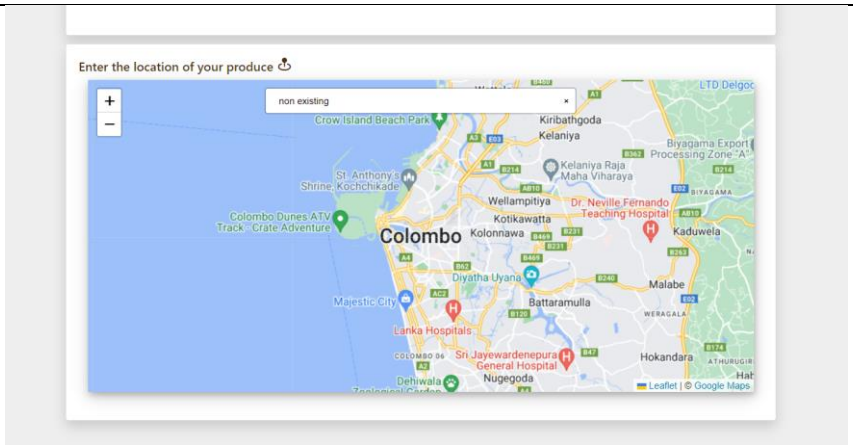
Test Case ID	TC003
Test objective	Attempt login with empty fields
Test data	Click ‘Login’
Expected Result	Display error message regarding empty fields
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

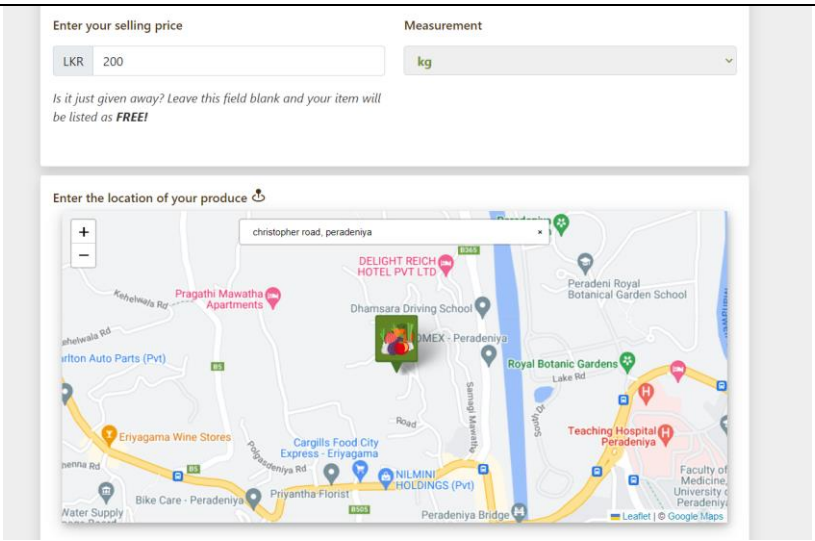
Test Case ID	TC004
Test objective	Attempt login with invalid credentials
Test data	Enter
Expected Result	Display error message
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

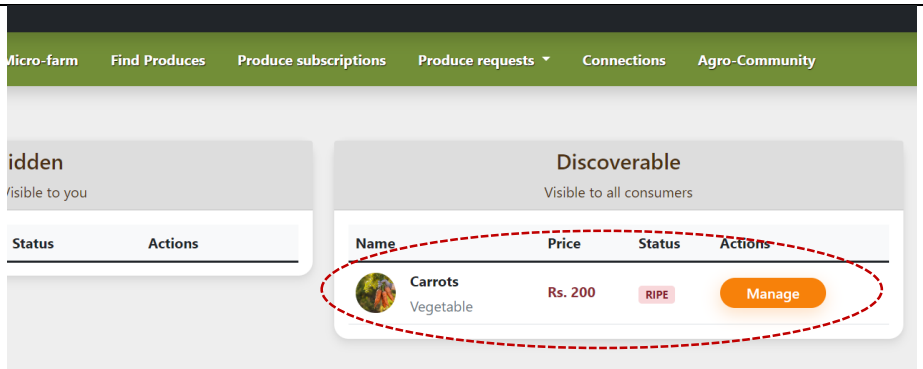
Test Case ID	TC005
Test objective	Attempt login with valid credentials
Test data	Enter username : abshalom@gmail.com, password: abshalom20 and click ‘Login’
Expected Result	Navigate to interface displaying successful authentication
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

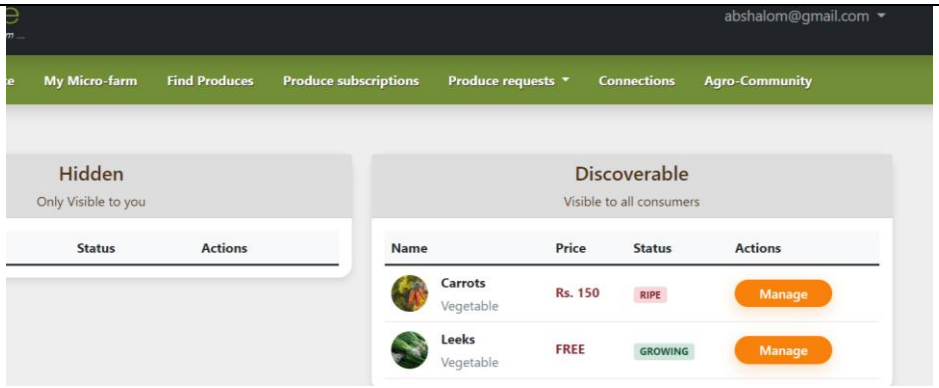
Test Case ID	TC006
Test objective	Adding produce with incomplete fields
Test data	-
Expected Result	Display error message regarding empty fields
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

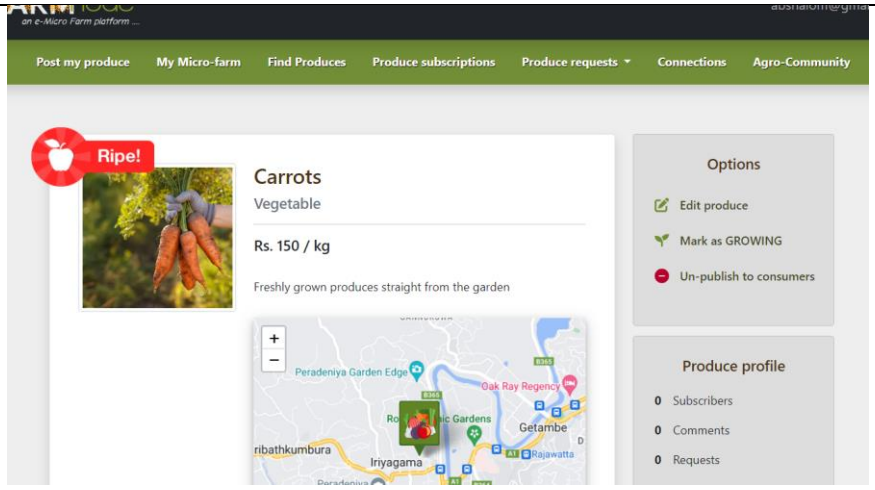
Test Case ID	TC007
Test objective	Upload a non-image file while adding produces
Test data	Select a pdf file during upload
Expected Result	Display file-type error message
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

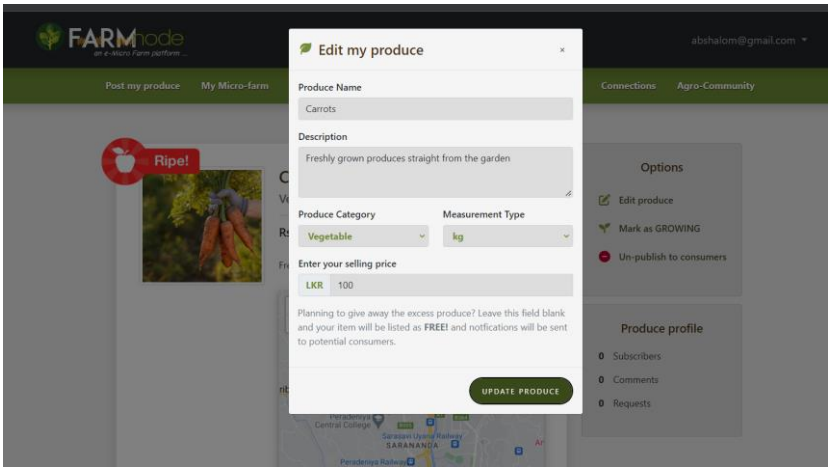
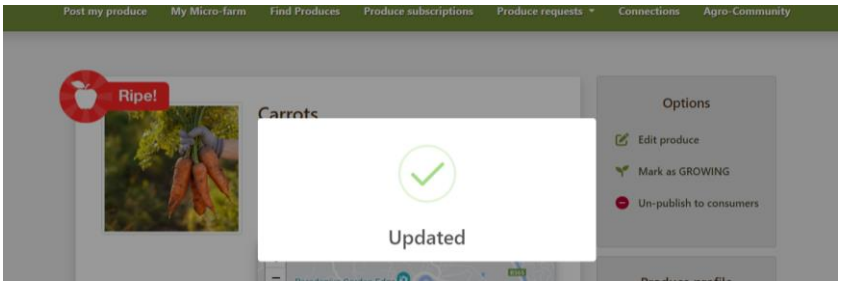
Test Case ID	TC008
Test objective	Adding an incorrect address while adding a produce
Test data	Enter address: ‘non existing’
Expected Result	Marker won’t be placed on the marker
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

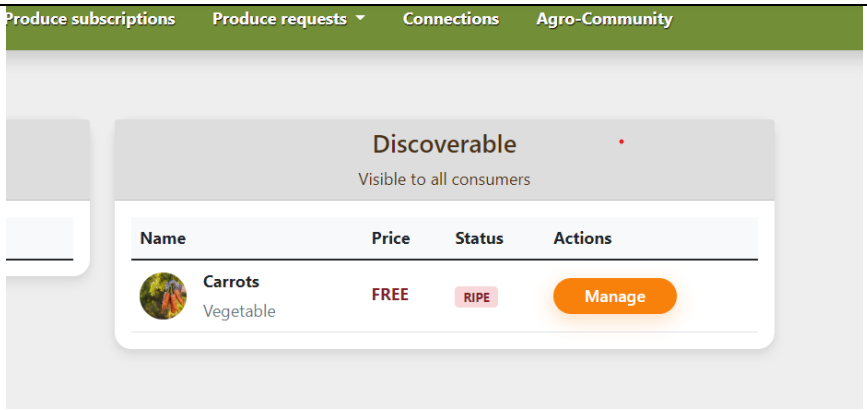
Test Case ID	TC009
Test objective	Adding an valid location while adding a produce
Test data	Enter address: ‘christopher road, Peradeniya
Expected Result	Marker placed accurately on the location
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

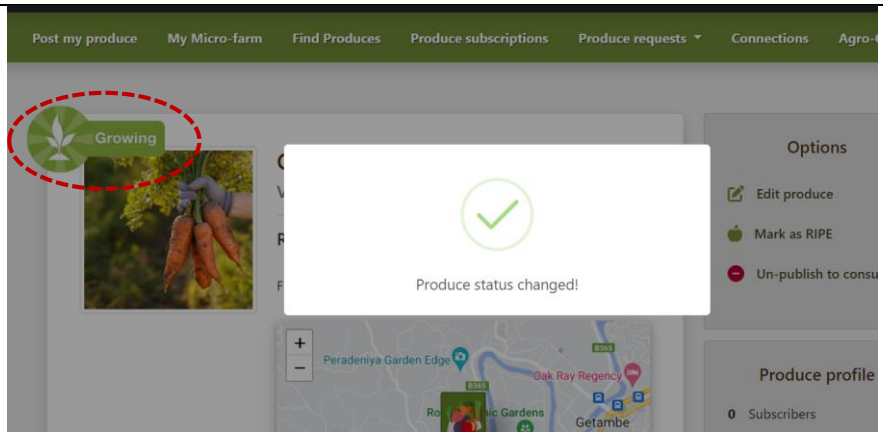
Test Case ID	TC010
Test objective	Adding a produce with valid produce details
Test data	Enter produce name = ‘Carrot’, description = ‘Fresh grown produce’, price = 200, status = ‘Ripe’ and click ‘Post produce’
Expected Result	Display success and produce appears in your produce profile under discoverable tab.
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

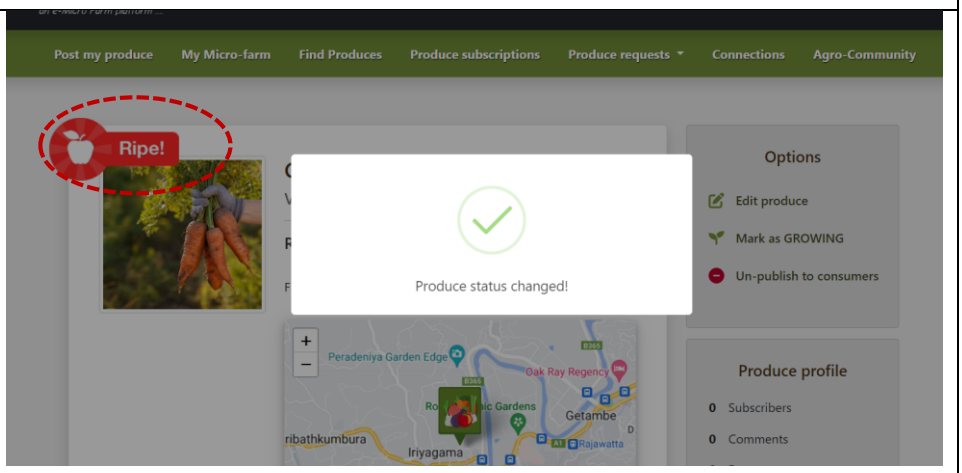
Test Case ID	TC010
Test objective	Viewing all users' produces
Test data	Click on 'My-Micro farm'
Expected Result	Display all the user's produces
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

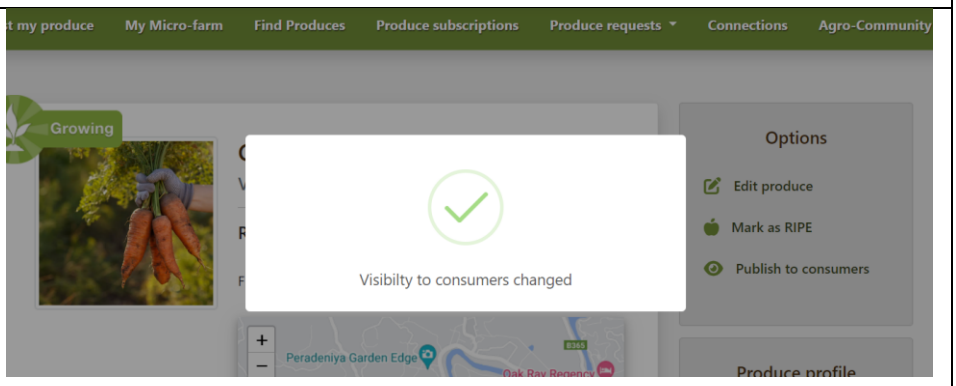
Test Case ID	TC012
Test objective	Click on a listed produce
Test data	Click on 'Manage' corresponding to first produce
Expected Result	Navigate to interface displaying clicked produce details accurately
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

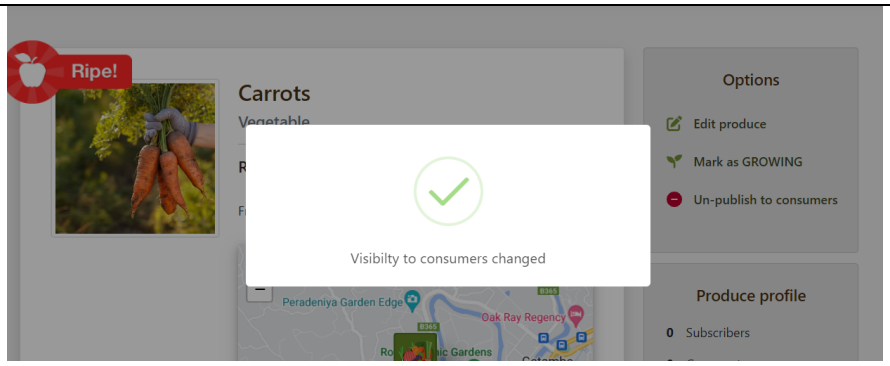
Test Case ID	TC013
Test objective	Updating produce details
Test data	Enter price = ‘300’ and click ‘Update’
Expected Result	Display success and the produce data is updated
Actual Result	Same as Expected
Screenshot	 
Conclusion	Works successfully

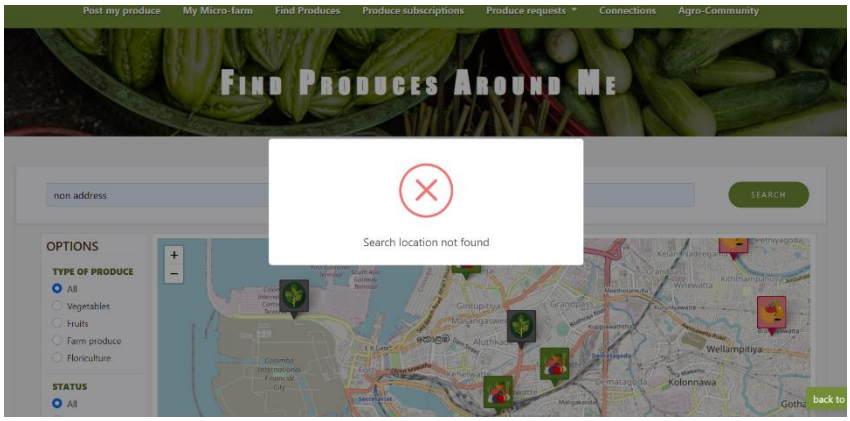
Test Case ID	TC014
Test objective	Updating produce details to be free
Test data	Enter price=0 and click ‘update’
Expected Result	Display success and the produce should be listed as free
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

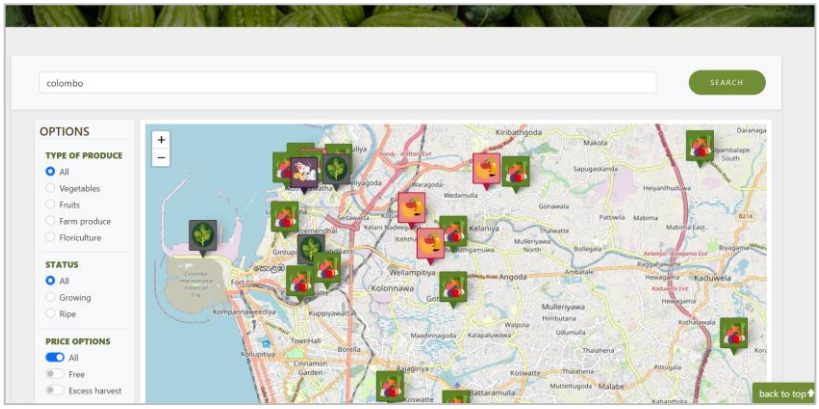
Test Case ID	TC015
Test objective	Changing produce status to “GROWING”
Test data	Click ‘Mark as Growing’
Expected Result	Display success and the status is updated
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

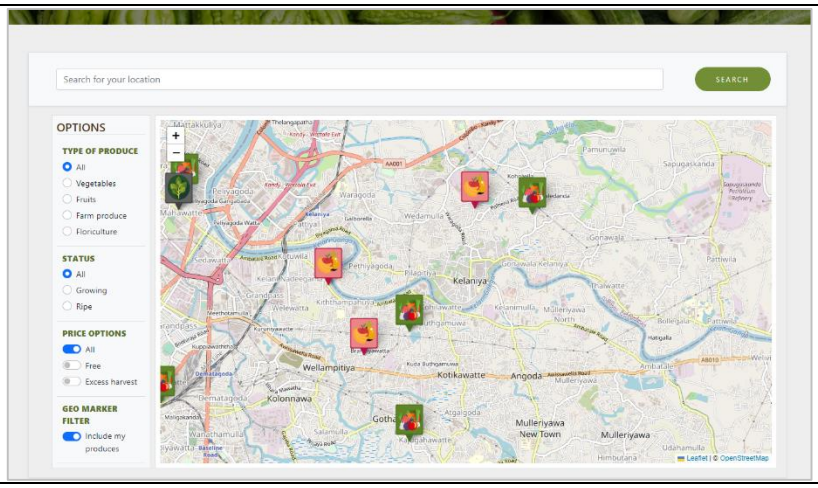
Test Case ID	TC016
Test objective	Changing produce status to “RIPE”
Test data	Click ‘Mark as Ripe’
Expected Result	Display success and the status is updated
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

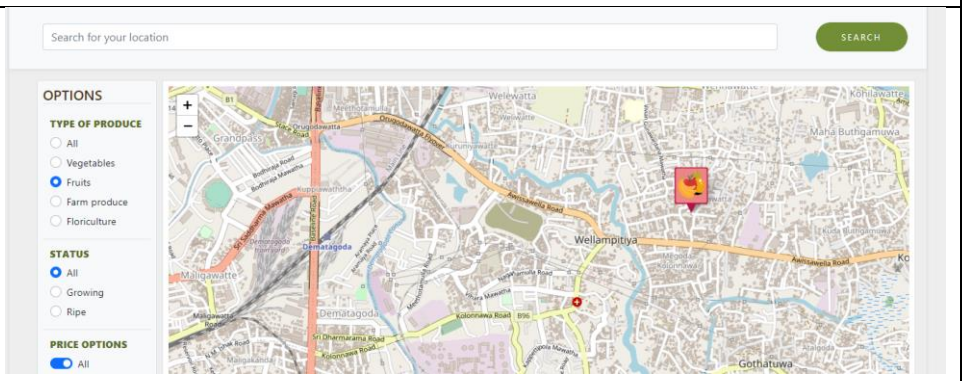
Test Case ID	TC017
Test objective	Changing produce visibility to hidden
Test data	Click ‘Un-publish to consumers
Expected Result	Display success and the status is updated
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

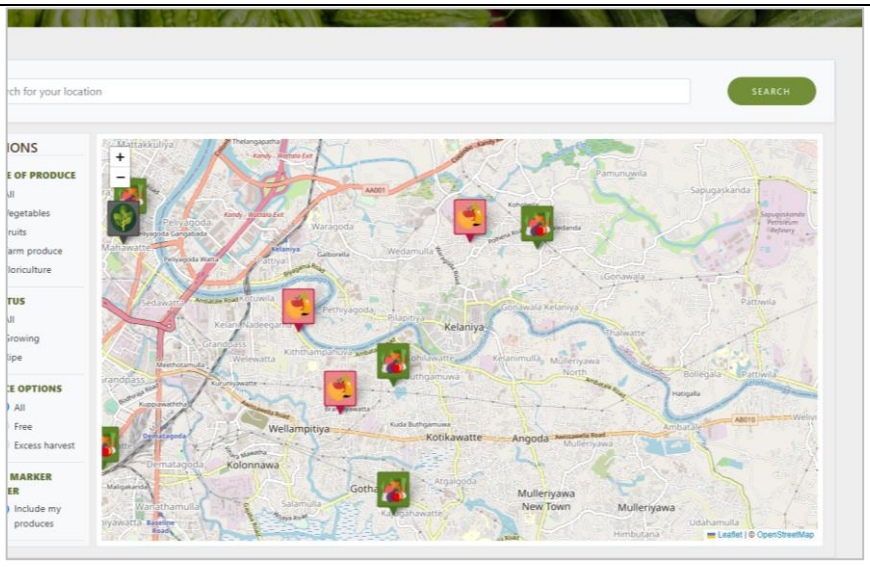
Test Case ID	TC018
Test objective	Changing produce visibility to be visible
Test data	Click ‘Un-publish to consumers
Expected Result	Display success and the status is updated
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

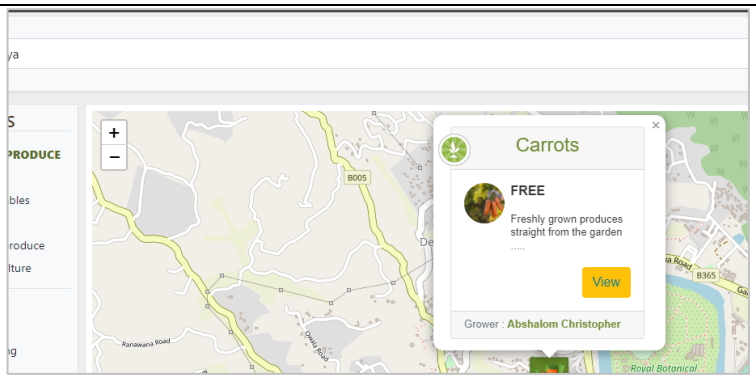
Test Case ID	TC019
Test objective	Search by invalid address when finding produces
Test data	Enter
Expected Result	Display error
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

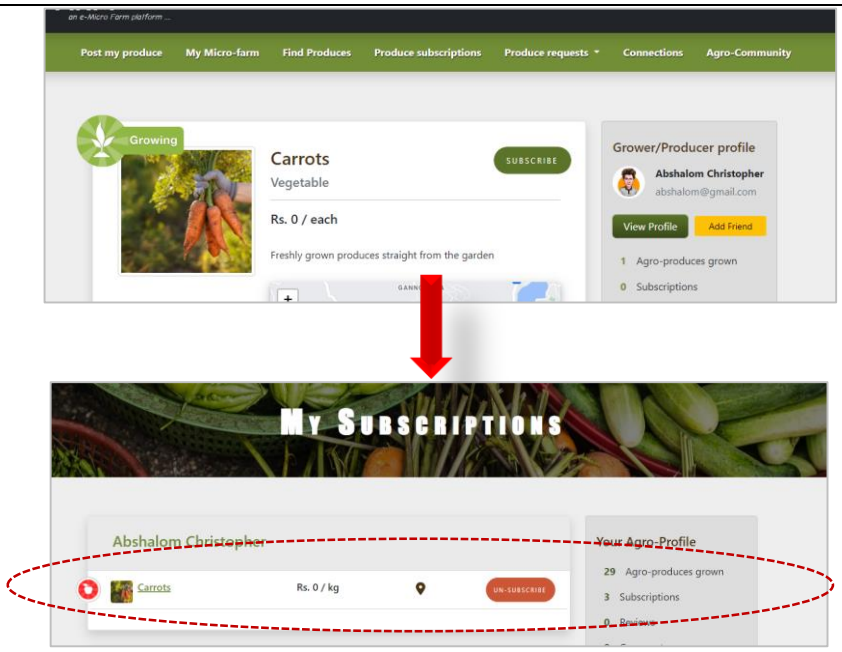
Test Case ID	TC021
Test objective	Change map bounds by dragging when finding produces
Test data	Drag map
Expected Result	Produces result should change dynamically
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

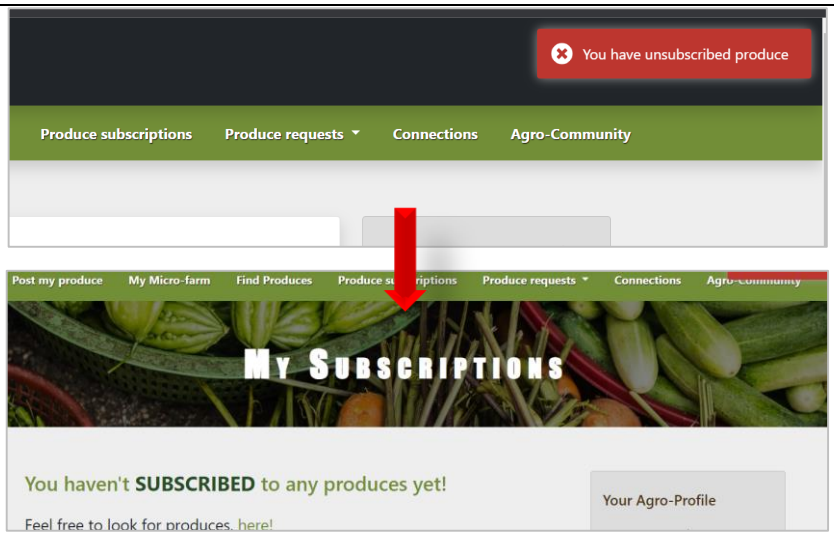
Test Case ID	TC022
Test objective	Change map bounds by zooming when finding produces
Test data	Drag map
Expected Result	Produces result should change dynamically
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

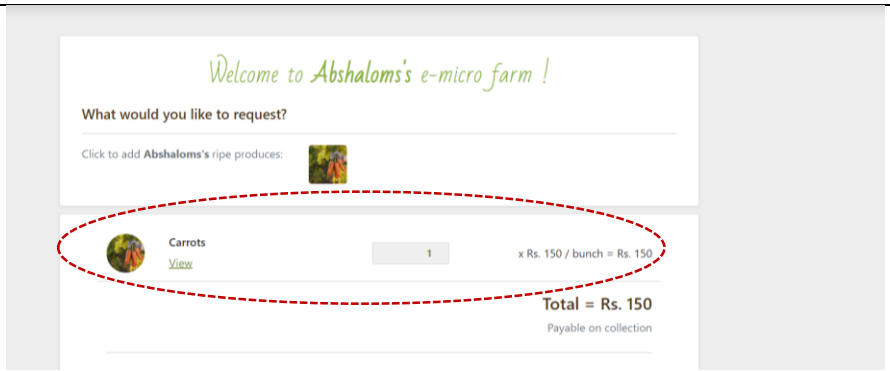
Test Case ID	TC023
Test objective	Filter by produce attributes when finding produces
Test data	Select ‘Fruits’
Expected Result	Only fruits results are shown
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

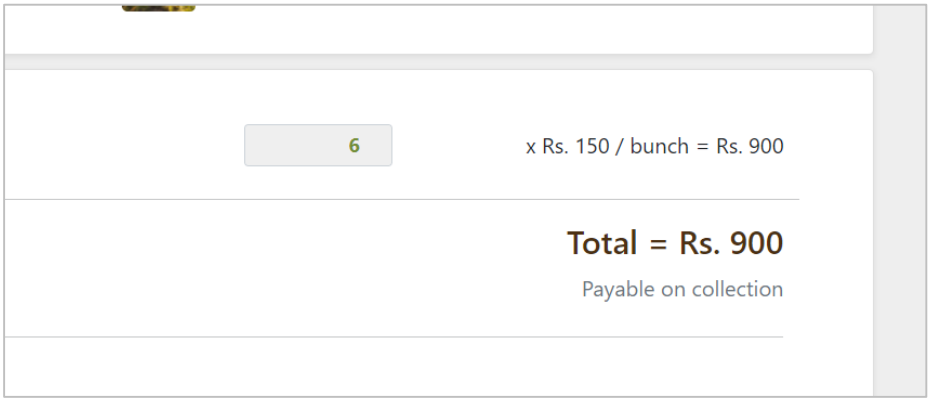
Test Case ID	TC024
Test objective	Filter by pricing strategy
Test data	Select ‘Free
Expected Result	Only free produces are shown
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

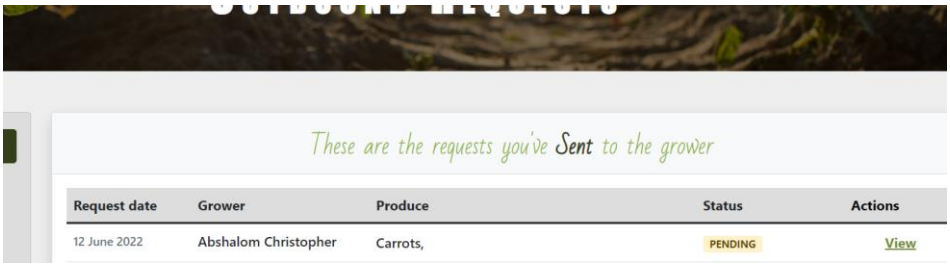
Test Case ID	TC025
Test objective	Viewing produce marker in finding produces
Test data	Click on produce marker
Expected Result	Popup should appear with relevant accurate produce details
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

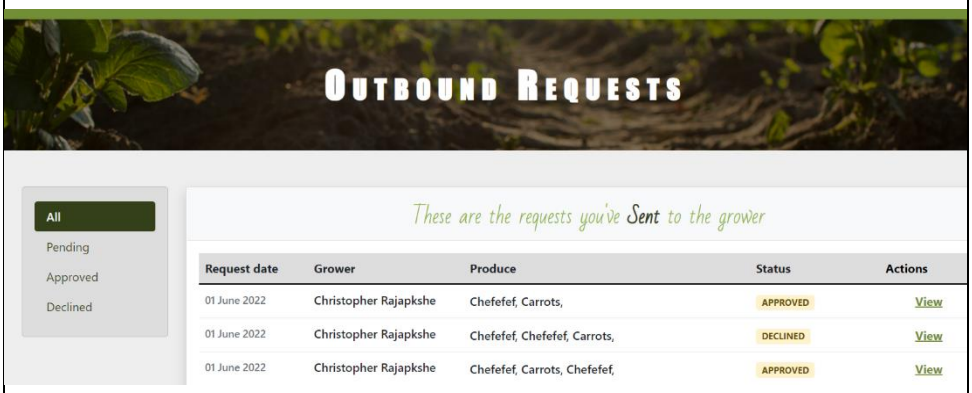
Test Case ID	TC026
Test objective	Subscribing to produce
Test data	Click on ‘Subscribe’
Expected Result	Display success and the subscribed produce should appear in subscription list
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

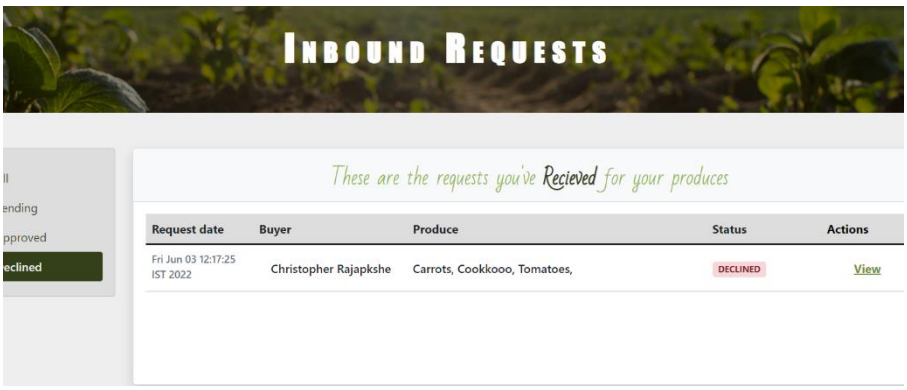
Test Case ID	TC027
Test objective	Unsubscribe a produce
Test data	Click on ‘Unsubscribe’
Expected Result	Display success and the subscribed produce should appear in subscription list
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

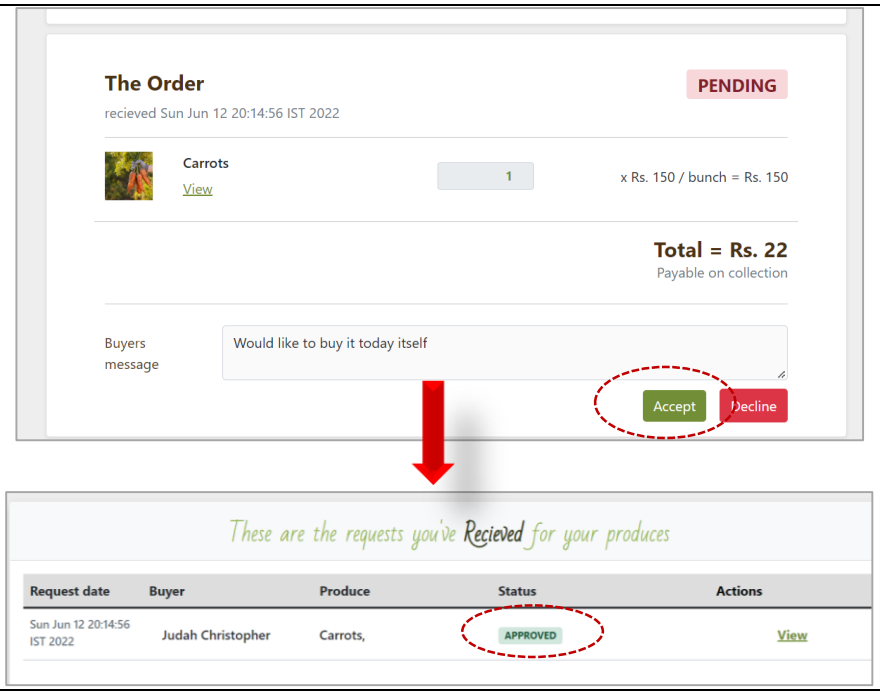
Test Case ID	TC029
Test objective	Adding a produce to the request list
Test data	Click on ‘Request produce’
Expected Result	Added produce appears on the request list
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

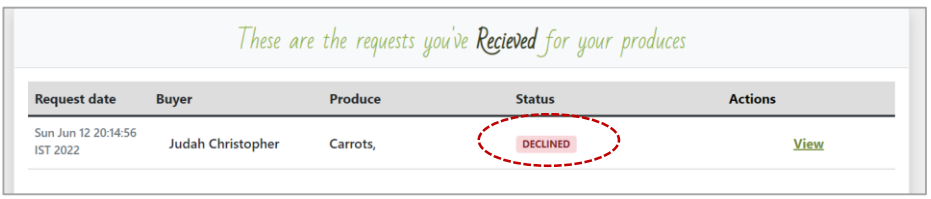
Test Case ID	TC031
Test objective	Update quantity of produce in request
Test data	Enter quantity = 6
Expected Result	Reflected line price/total price updates
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

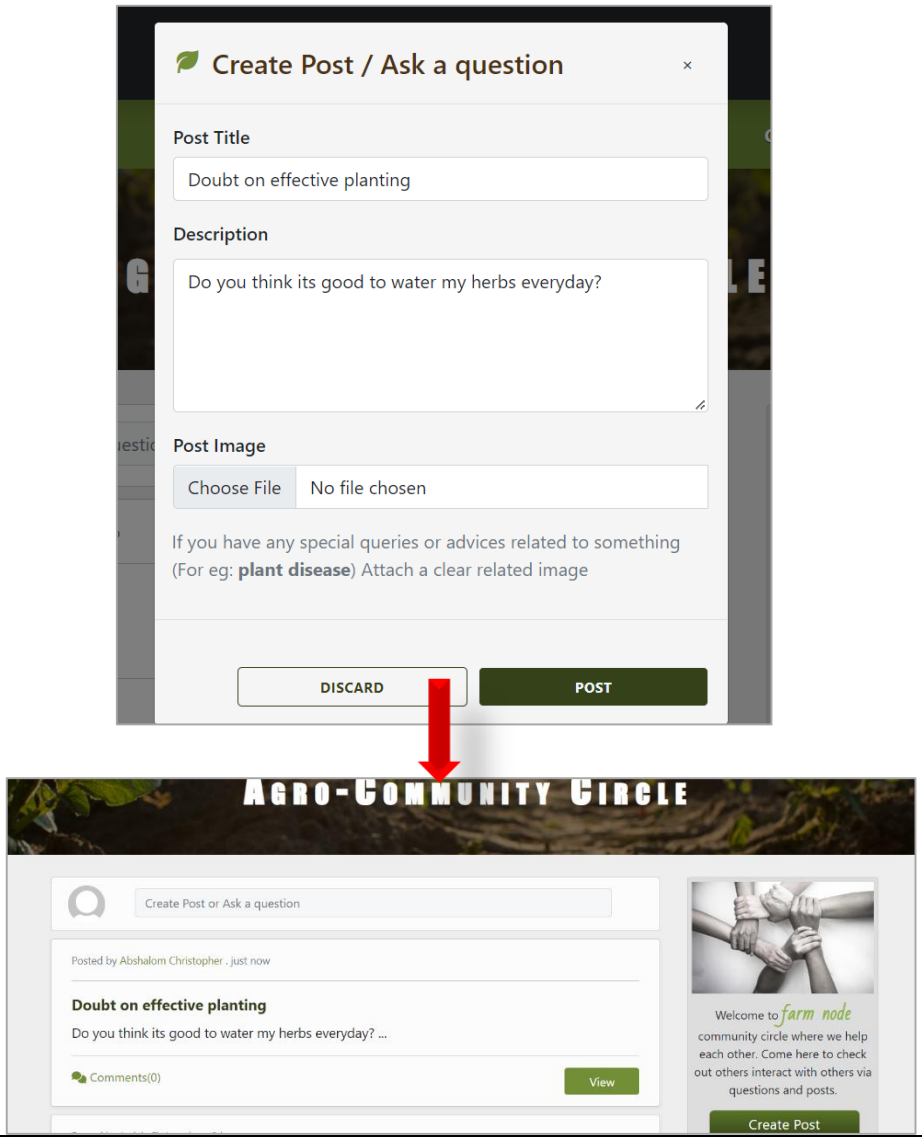
Test Case ID	TC033
Test objective	Placing a produce request
Test data	Click on ‘Place request’
Expected Result	Display success and request appears on the outbound request list.
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

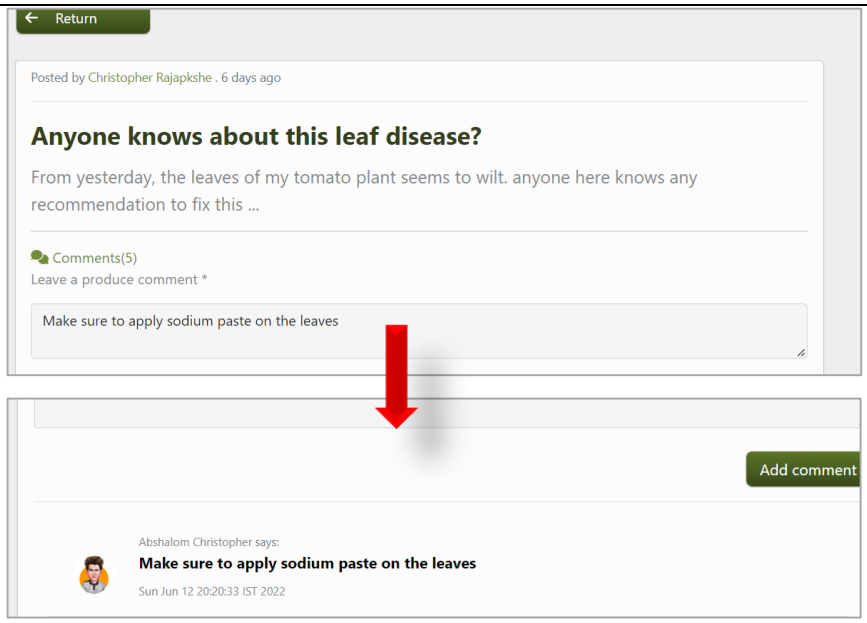
Test Case ID	TC034
Test objective	Viewing sent produce-requests
Test data	Click on ‘Outbound requests’
Expected Result	Display all sent requests
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

Test Case ID	TC035
Test objective	Viewing received produce-request
Test data	Click on ‘Inbound requests’
Expected Result	Display all received requests list
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

Test Case ID	TC036
Test objective	Approve a produce-request
Test data	Click on ‘Accept’
Expected Result	Status of request updates
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

Test Case ID	TC037
Test objective	Decline a produce-request
Test data	Click on ‘Decline
Expected Result	Status of request updates
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

Test Case ID	TC041
Test objective	Creating a post in the community
Test data	Enter post title with description and click ‘Post’
Expected Result	Display all received requests list
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

Test Case ID	TC041
Test objective	Commenting on a post in the community
Test data	Enter comment = ‘ Make sure to apply sodium poste on the leaves” and click add comment
Expected Result	Display success and comment appears on the list of comments with accurate time and posted user
Actual Result	Same as Expected
Screenshot	
Conclusion	Works successfully

Appendix E: Other tools and technologies used in the project

❖ Maven

Maven is an open-source project management tool that is mainly used to maintain and build all project related dependencies within a POM (Project Object Model) file. Since the system used many dependencies such as Lombok, security plugins, mapstruct and other critical libraries, Maven automate this process defining the dependencies in the POM file effectively rather than downloading and placing the jars manually in to the project. Maven requires internet to download the jars, which is a one-time process

❖ Bootstrap

Code re-usability was a prime goal in the project. There an open-source styling library Bootstrap 5.0 was used in the project to reuse styling components of its CSS libraries. In the project pre-defined sizing and style classes were used to avoid lumps of boiler-plated CSS coding. On a developer perspective, it helped to concentrate on focusing on important tasks without worrying on solutions that are already readily available. Ultimately this helped to improve the software carpentry aspect of the project.

❖ NPM

Node Package Manager generally known as NPM have been utilized to make the development process of this application more streamlined. When it came to the Angular framework, the NPM was used to manage all of the front-end dependencies, JavaScript libraries, and CSS frameworks in order to compile the assets and manage all of the packages from a single point where removal as well as update / upgrades of the packages can be performed in very little time and developing the application so much less stressful because everything was taken care of by these package managers from the start.

❖ Swagger

Since the application consumed a backend API it was necessary to study the input and output patterns of an API-endpoint. To serve this purpose Swagger framework was configured in the Java project to easily find and understand all the end-point behavior without accessing the code lines frequently.

Appendix F: Glossary of terms

Agro-Produce	Referred to a product grown by an agricultural or agro-producer.
GeoJSON	Universal standard to represent geospatial data.
Architecture	The design or structure of any kind of system
CSS	This is a language that is used for make attractive a content of the web pages.
Entity	A single object which can be store data
Module	Separate component of a system which are interact with each other
OOP	A software design method that models the real objects using classes and objects.
Server	A computer or computer program. It can manages access to a centralized resource in a network
UML	A modeling language which can be used to visualize a design of the system

Appendix G: Supervision logs sheets

Student's Name: Judah Abshalom ..... **Cardiff Number:** st20215381

Date: 28/04/202.2 **Meeting No:** 01 **Intake:** June /202.1

Project Title: A web based e-micro farm platform for local agro-growers to effectively connect with consumers

Supervisor's Name: Miss Upeka Wijesinghe **Supervisor Signature:** .....
Program Manager: Mr. Shalika Caldera **Program Manager's Signature:**.....

Work progression as to date (noted by student BEFORE mandatory supervisor meeting):

Items for Discussion (noted by student BEORE mandatory supervisor meeting):

1. Requirement gathering and UML diagrams
2. Documentational structures (Introduction and literature review)
3. Best practices in image uploading and retrieving

Student's Name: Judah Abshalom ..... **Cardiff Number:** st20215381

Date: 06/05/202.2 **Meeting No:** 02 **Intake:** June /202.1

Project Title: A web based e-micro farm platform for local agro-growers to effectively connect with consumers

Supervisor's Name: Miss Upeka Wijesinghe **Supervisor Signature:** .....
Program Manager: Mr. Shalika Caldera **Program Manager's Signature:**.....

Work progression as to date (noted by student BEFORE mandatory supervisor meeting):

Items for Discussion (noted by student BEORE mandatory supervisor meeting):

1. Extent of similar system comparison and other requirement gathering techniques
2. Feasibility study amendments
3. System design requirements and expected explainations

Student's Name: Judah Abshalom ..... **Cardiff Number:** st20215381

Date: 14/05/202.2 **Meeting No:** 03 **Intake:** June /202.1

Project Title: A web based e-micro farm platform for local agro-growers to effectively connect with consumers

Supervisor's Name: Miss Upeka Wijesinghe **Supervisor Signature:** .....
Program Manager: Mr. Shalika Caldera **Program Manager's Signature:**.....

Work progression as to date (noted by student BEFORE mandatory supervisor meeting):

Items for Discussion (noted by student BEORE mandatory supervisor meeting):

1. UML Diagrams reviewing
2. Implementation chapter inclusions and exclusions
3. System related discussion

Action List (to be attempted by student by the NEXT mandatory supervisory meeting – TO BE FILLED SUPERVISOR):

Student's Name: Judah Abshalom ..... **Cardiff Number**: st20215381

Date: 30/05/2022 **Meeting No**: 04 **Intake**: June/2021

Project Title: A web based e-micro farm platform for local agro-growers to effectively connect with consumers

Supervisor's Name: Miss Upeka Wijesinghe **Supervisor Signature**: .....
Program Manager: Mr. Shalika Caldera **Program Manager's Signature**:.....

Work progression as to date (noted by student BEFORE mandatory supervisor meeting):

Items for Discussion (noted by student BEORE mandatory supervisor meeting):

1. Software development process reviewing
2. Get feedback on documentation so far
3. Advice related to source code submission

Student's Name: Judah Abshalom ..... **Cardiff Number**: st20215381

Date: 09/06/2022 **Meeting No**: 05 **Intake**: June/2021

Project Title: A web based e-micro farm platform for local agro-growers to effectively connect with consumers

Supervisor's Name: Miss Upeka Wijesinghe **Supervisor Signature**: .....
Program Manager: Mr. Shalika Caldera **Program Manager's Signature**:.....

Work progression as to date (noted by student BEFORE mandatory supervisor meeting):

Items for Discussion (noted by student BEORE mandatory supervisor meeting):

1. Testing exclusions
2. Get feedback on documentation so far
3. Deployment