Bilkent University

Department of Computer Engineering

Senior Design Project

Project Short-name: So FarM So Good

Analysis Report

Giray Baha Kezer, Fazilet Simge Er, Melih Ünsal, Kaan Atakan Öztürk

Supervisor: **Prof. Dr. Halil Altay Güvenir**

Jury Members: **Prof. Dr. Özcan Öztürk, Prof. Dr. Uğur Güdükbay**

Innovation Expert: **Kerem Erikçi**

November 11, 2019

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491.

TABLE OF CONTENTS

1 Introduction

In Turkey, farmers, especially the small ones, have so many problems and this situation causes our production in agriculture to decrease day by day. However, we cannot put ourselves at risk for agriculture because agriculture constitutes around 13% of the country's exports. We also see that agricultural areas and the number of farmers has decreased for the last 5 years. So the problems of agriculture should be resolved one by one and we plan to solve some problems of agriculture by So FarM So Good. This web based platform is going to combine small farmers so that they will be able to sell their goods to the appropriate companies by the virtual cooperative we are going to establish. Normally, companies like Konya Şeker Sanayi and Ticaret A.Ş or Çaykur have some limitations when they are aiming to buy goods from farmers and the farmers who are not able to produce under the threshold that the companies put, sell their crops to the local landowners for less that worth. By our virtual cooperative, they are able to combine their crops and the cooperative is going to sell the crops and pay to the farmers proportional to the amount of products the farmers put into. So FarM So Good is also servicing more secure payment system which uses the new trend: Blockchain

Computer Science world has a new trendy word ‘blockchain’. We met this idea in the early twentieth century. By blockchain, both side of a trade, don’t need any mediator to prove that this trade has occurred. Nowadays, we use banks as a mediator in the trades. However, blockchain convert the trades as a decentralized form so that everyone is able to see all the trades real time and this makes the trades more secure ever than before. In this project, we use blockchain in the trades between the virtual cooperatives and the farmers and also between the farmers in cooperatives and the government. By resolving these 2 problems,, we aim to solve the problems of injustice farmers due to the small amount of the product they produce and thus to make them happy and add value to the agricultural economy of the country by increasing their contribution to total production by making them happy.

2 Current System

2.1 Agrivi

Agrivi is a data-driven farm management software which helps people manage every activity on their farm. It gives the people the opportunity to plan, monitor and analyze their farms from planting to harvesting with only a few clicks. Agrivi’s core features are farm management, powerful analytics, advanced pest detection algorithms and best practice knowledge .

2.2 WeFarm4

Wefarm4 is a cooperation among farmers with the sub-branches of WeFarm4future, WeFarm4planet, WeFarm4Community and WeFarm4Growth. They're aiming to work together, create long-term benefits and empower farmers through cooperatives [3].

Above mentioned applications are established to support to the farmers. However, our projects different than the current ones. Establishing the online cooperatives between farmers is a significant feature that does not exist those. Moreover, we also intend to solve quota problem for the sugar beet farmers which is also new for the current system. On the other hand, our software will be web application. However, entire above mentioned softwares have mobile applications. The project also aims to contribute economy of the Turkey by controlling sugar beet distribution and holding the data in Blockchain system.

3 Proposed System

3.1 Overview

Our Senior Design Project has an innovative engineer solution for the problems listed in the introduction part. In this project, we are planning to resolve 2 problems of the sugar farmers especially who produce less and cannot pass the threshold that the companies like Anadolu Efes, Çaykur, Konya Şeker Sanayi ve Ticaret A.Ş put by themselves. In this platform called So FarM So Good, the sugar farmers will be able to establish their virtual cooperation so that they will be able to unite for large companies to see themselves as business partners other than worthless individuals. For example, a farmer with 30 acres of land may not be noticed by the big companies, but if 5 farmers with 30 acres of land put together their fields, they will make the companies' mouth water.

We also offer a blockchain technology for the paymounts between officials and the farmers in the platform to make the paymounts as secure as possible. To give an instance, a farmer needs 100k Turkish Liras to make harvest, officials give him 100k valued coins instead via blockchain system. These 100k coins can be useable into officials related departments and supplier chains, it means the farmer could only use that valued coins into related expenses and take whatever he needs. Then process will not stop and will be continuous and there is going to be a win-win system belongs farmer, officials. At the end, small farmers will be able to guarantee to sell their crops for their values to the government and there will be no security issue during the payments.

3.2 Functional Requirements

3.2.1 User Functionality Requirements

- The companies, willing to see where the sugar beet fields, should see their locations on the map. It will be serviced by an optimized sugar beet detection model.
- Farmers could have the ability to create cooperation in the website.
- Companies interested in the specific product should have the ability to notify the cooperation consisting of farmers.
- The fields related to the specific product should be shown on the map as a point whose size is proportional to the area of the field.
- The platform should recommend appropriate farmers to the farmers who are searching for a friend to form cooperation.
- The platform should recommend cooperatives to the companies related to their needs in terms of property.
- The platform should recommend farmers to property based sell options to directly companies if they have adequate harvest in terms of different quotas.
- The cooperatives give the rate-based material to the farmers, which can be converted into money after the product sale via taking in proportion as they gave to the companies.
- The farmers takes rate-based material before the harvest sale and then after the harvest sale to companies from cooperatives or farmers himself company pays. After

that payment if two actors, cooperative and company, confirm that phase the transaction inserted into our platform.

3.2.2 System Functionality Requirements

- The platform needs to be a mobile compatible web-based platform.
- There needs to be two different user type, cooperative and farmer.
- Each user needs an account to use the platform.
- Cooperative names must be unique.
- User emails and passwords must be unique.
- A database needs to be set up for the data which will be stored in each profile.

3.3 Non-functional Requirements

3.3.1 Usability

- The platform should have ease of use for our main customers, farmers, which are the most important part of agriculture. Farmers demand more ease of use for this technological move. Thus, they should be able to use the platform easily and willingly.
- Features of the platform and user interface should be easily understandable.

3.3.2 Security

- The platform should ensure the security of data of users and private information about companies by blockchain system.

3.3.3 Scalability

- The platform should be scalable enough to handle the huge number of users and data processing work.

3.3.4 Robustness

- The platform should be robust. Whatever the size of the coming data, the platform should handle it.

3.3.5 Extensibility

- The platform should support easy integrations for possible future features.

3.4 Pseudo Requirements

- The platform will be a web application written in Angular 4-5.
- The platform will be developed using Docker to standardize operations and seamlessly move the platform.
- A database will be set up to store data.

3.5 System Models

3.5.1 Scenarios

Use Case #1

Unique Name: Sign Up

Participating actor/s: Producer/Company

Entry condition: Producer/Company signs up to the system by clicking “Sign Up” button on the main page of the website.

Exit condition: Producer/Company will click the Sign Up button on the Sign Up Page.

Flow of Events:

1. After clicking the Sign Up button on the main page, a popup will appear which the user must choose either of the options producer or company.
2. After selecting, the sign up page will appear.
3. Producer/Company will fill in the blanks and press the Sign Up button.
4. Producer/Company signs up successfully.

Use Case #2

Unique Name: Login

Participating actor/s: Producer/Company

Entry condition: Producer/Company will login to the system by clicking “Login” button on the main page of the website.

Exit condition: Producer/Company will click the Login button on the Login Page.

Flow of Events:

1. After clicking the Login button on the main page, the system will redirect the user to the Login page.
2. Producer/Company will enter their email address and their password, and press the Login button.
3. Producer/Company logs in successfully.

Use Case #3

Unique Name: Change Password

Participating actor/s: Producer/Company

Entry condition: Producer/Company clicks “Change Password” button on their profile page.

Exit condition: Producer/Company presses the “Confirm” button to confirm the new password.

Flow of Events:

1. Producer/Company opens their profile page by clicking their icon on their main page.
2. Producer/Company clicks “Change Password” button.
3. Producer/Company enters their old password and the new password, and presses “Confirm” button.
4. Producer and the company changes password successfully.

Use Case #4

Unique Name: Purchase Product

Participating actor/s: Company

Entry condition: Company will click the “Purchase” button next to the product.

Exit condition: Company will click “Confirm” button on the popup.

Flow of Events:

1. After company click the “Purchase” button, a popup will appear with the details of the product.
2. Company will press “Confirm” button.
3. Company purchases the product successfully.

Use Case #5

Unique Name: View Products On Sale

Participating actor/s: Company

Entry condition: Use Case #2 - Login

Flow of Events:

1. After logging in, the company can see the products on sale on their main page.

Use Case #6

Unique Name: Add Product

Participating actor/s: Producer

Entry condition: Producer will click the “Add Product” button on their main page.

Exit condition: Producer will click the “Add Product” button on the Add Product page.

Flow of Events:

1. After the producer clicks the “Add Product” button, the Add Product page will appear.
2. Producer will fill in the blanks.
3. Producer will click the “Add Product” button.
4. The product is successfully added.

Use Case #7

Unique Name: View Profile

Participating actor/s: Producer/Company

Entry-Exit condition: Producer/Company will click their icon on their main page.

Flow of Events:

1. After Producer/Company clicks their icon, the profile page will appear.

Use Case #8

Unique Name: View Previous Transactions

Participating actor/s: Producer/Company

Entry condition: Use Case #7 - View Profile

Flow of Events:

1. After Producer/Company goes to their profile page, they can see their previous transactions.

Use Case #9

Unique Name: Form Virtual Cooperative

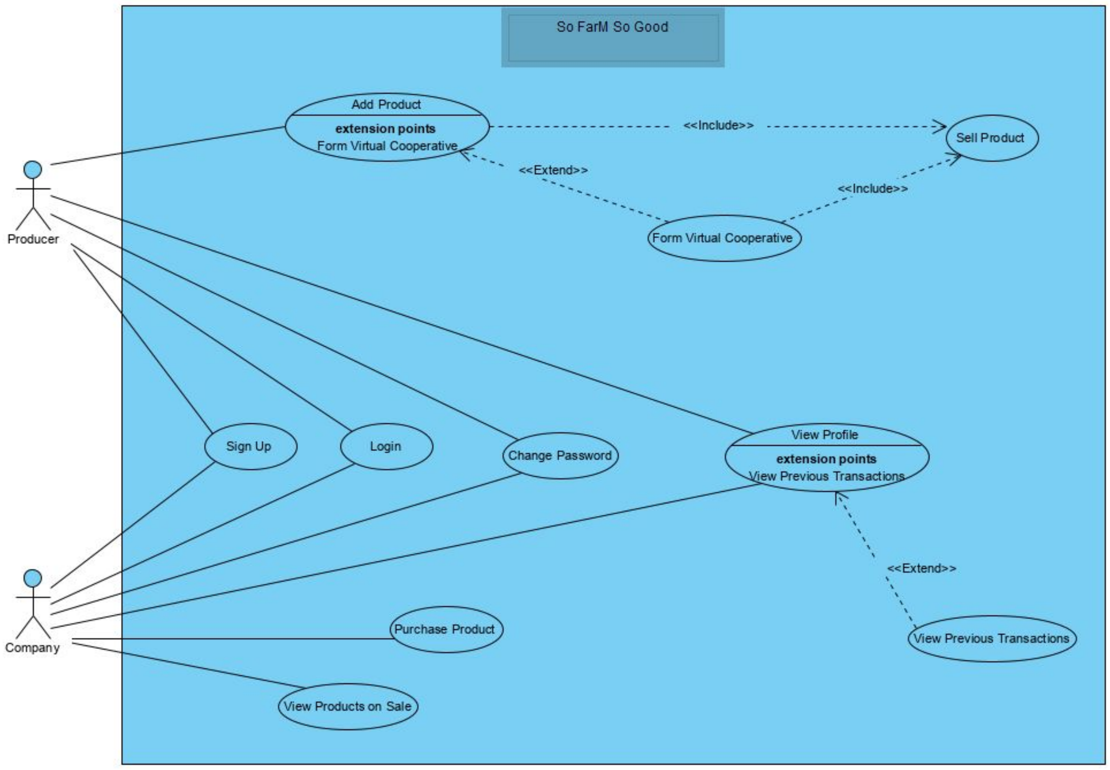
Participating actor/s: Producer

Entry condition: Producer clicks “Form Virtual Cooperative” next to their products on their main page.

Flow of Events:

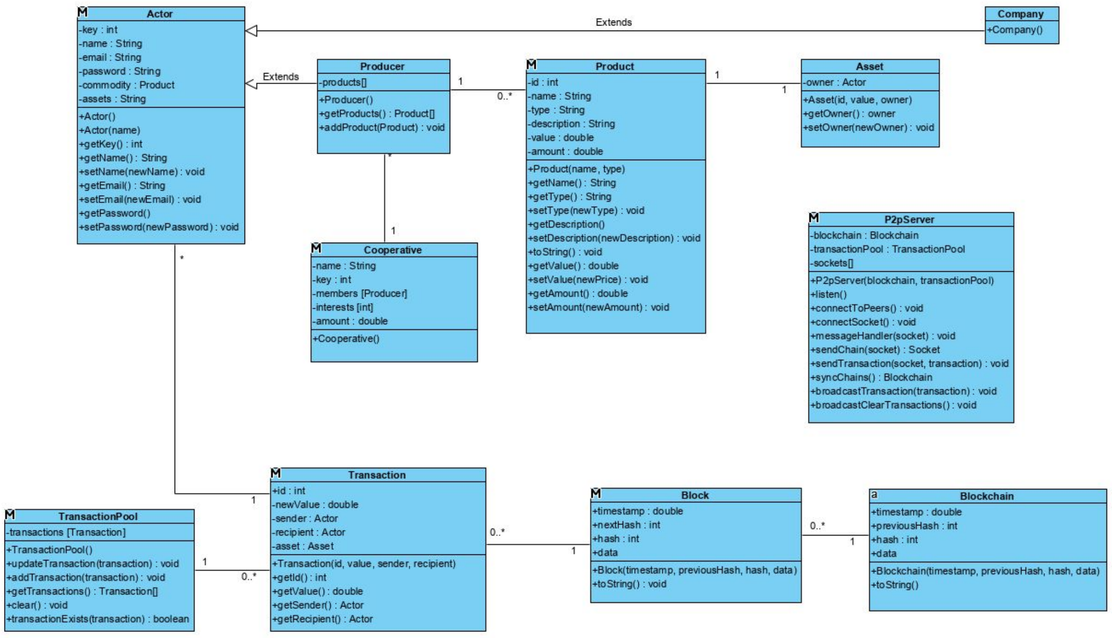
1. After the producer clicks “Form Virtual Cooperative” button, the system will find a suitable Virtual Cooperative for the producer.

3.5.2 Use-Case Model



3.5.3 Object and Class Model

3.5.3.1 Class Diagram



3.5.3.1.1 Class: Actor

Actor class is a mining new blocks for our blockchain platform. It has some private key to be processed and name, some database properties and most importantly commodity and assets information inside. This commodity is a base for transactions and this negotiation will be recorded into our secure blockchain platform.

- ➔ <u>Parameters:</u>
    - ◆ **key:** An integer based private keys to traverse into blockchain blocks.
    - ◆ **name:** A string which holds the name of the actor.
    - ◆ **email:** A string which holds the email of the actor.
    - ◆ **password:** A string which holds the password of the actor.
    - ◆ **commodity:** A product based object to hold property inside value, harvest, or non.
    - ◆ **assets:** A string based made cryptocurrency of actor’s stakes.
- ➔ <u>Operations:</u>
    - ◆ **getKey():** Returns the key of the actor.
    - ◆ **getName():** Returns the name of the actor.
    - ◆ **setName(String newName):** Changes the name of the actor with the given newName.
    - ◆ **getEmail():** Returns the email of the actor.
    - ◆ **setEmail(String newEmail):** It sets new assigned email to string.
    - ◆ **getPassword():** Returns string based password.
    - ◆ **setPassword(newPassword):** It sets password with a new string value.

3.5.3.1.2 Class: Producer

This class extends Actor class. Producer class is responsible for product based operations, it means adding products, selling products and taking Product object are operated by that class. Producer information and product object itself are handled in that class.

- ➔ <u>Parameters:</u>
    - ◆ **products[]:** An array based product objects to hold products inside.
- ➔ <u>Operations:</u>
    - ◆ **getProducts():** A method for taking products.
    - ◆ **addProduct(newProduct):** A method takes new products as parameter used for adding new product amount.

3.5.3.1.3 Class: Company

This class extends one Actor. It is a type of an actor and it is a part of transactions with cooperatives or directly with farmers. Takes products and gives money commodities exchanged with money and product assignments. Its transactions holds on blockchain records.

3.5.3.1.4 Class: Cooperative

It extends Actor class, it is a type of it. Cooperatives takes farmers products to pass the distinct quotas of companies. Its transactions done between farmers and company there is two type of transactions occured with cooperatives. Cooperatives with company and cooperatives with farmers. Cooperatives has interests as commodity at the beginning and after the transactions with farmers interest becomes farmers property and products becomes cooperatives property. Second transaction between company is done by taking products of cooperatives and giving them money, commodities are exchanged among them with product and money variables.

- ➔ Parameters:
  - ◆ **name:** String based name property.
  - ◆ **key:** String based private key property.
  - ◆ **members[Producer]:** Producer based array holds members.
  - ◆ **interests[int]:** Interest based integer array holds interests inside.
  - ◆ **amount[double]:** Amount based double array holds amounts inside.

3.5.3.1.5 Class: Product

Objects of this class represents the real life products produced by producers.

- ➔ Parameters:
  - ◆ **id:** Integer based unique id property
  - ◆ **name:** String based name property
  - ◆ **type:** String based type property
  - ◆ **description:** String based description explaining the product
  - ◆ **value:** Double based value property
  - ◆ **amount:** Double based amount property
- ➔ Operations:
  - ◆ **getName():** A method for taking the name of the product
  - ◆ **getType():** A method for taking the type of the product
  - ◆ **setType(String newType):** A method for setting the type of the product
  - ◆ **getDescription():** A method for taking the description of the product
  - ◆ **setDescription(String newDescription):** A method for setting the description of the product
  - ◆ **toString():** A method for printing the product
  - ◆ **getValue():** A method for taking the value of the product
  - ◆ **setValue(double newValue):**A method for setting the value of the product
  - ◆ **getAmount():** A method for taking the amount of the product
  - ◆ **setAmount(double newAmount):**A method for setting the amount of the product

3.5.3.1.6 Class: Asset

The objects of this class represents a Product with an owner.

- ➔ Parameters:
    - ◆ **owner:** Actor based object showing the owner of the asset
- ➔ Operations:
    - ◆ **getOwner():**A method for taking the owner of the asset
    - ◆ **setOwner(Actor newOwner):**A method for taking the owner of the asset

3.5.3.1.7 Class: Transaction

Objects of this class represent results of exchanges between producers and the companies.

- ➔ Parameters:
    - ◆ **id:** Integer based unique id property
    - ◆ **newValue:** Double based updated value of the transaction
    - ◆ **sender:** Actor based sender property
    - ◆ **recipient:** Actor based recipient property
    - ◆ **asset:** Asset based property
- ➔ Operations:
    - ◆ **getId():** A method for taking the id of the transaction
    - ◆ **getValue():** A method for taking the value of the transaction
    - ◆ **getSender():** A method for taking the sender of the transaction
    - ◆ **getRecipient():** A method for taking the recipient of the transaction

3.5.3.1.8 Class: TransactionPool

This class stores all transactions.

- ➔ Parameters:
    - ◆ **transactions[Transaction]:** Transaction based array property holding the transactions
- ➔ Operations:
    - ◆ **updateTransaction(Transaction transaction):** A method for updating the transaction in the pool
    - ◆ **addTransaction(Transaction transaction):** A method for adding a transaction to the pool
    - ◆ **getTransactions():** A method for taking a transaction in the pool
    - ◆ **clear():** A method for removing all the transactions in the pool
    - ◆ **transactionExists(Transaction transaction):** A method for controlling if a transaction exists in the pool

3.5.3.1.9 Class: Block

Objects of this class stores the Transactions.

- ➔ Parameters:
    - ◆ **timestamp:** Double based time property showing the time of the creation of the block
    - ◆ **nextHash:** Integer based property to make a connection with the next block

    - **hash:** Integer based property to encrypt the block
    - **data:** A property that holding all the information about the block
- Operations:
    - **toString():** A method to write the block

3.5.3.1.10 Class: Blockchain

This class stores the Blocks.

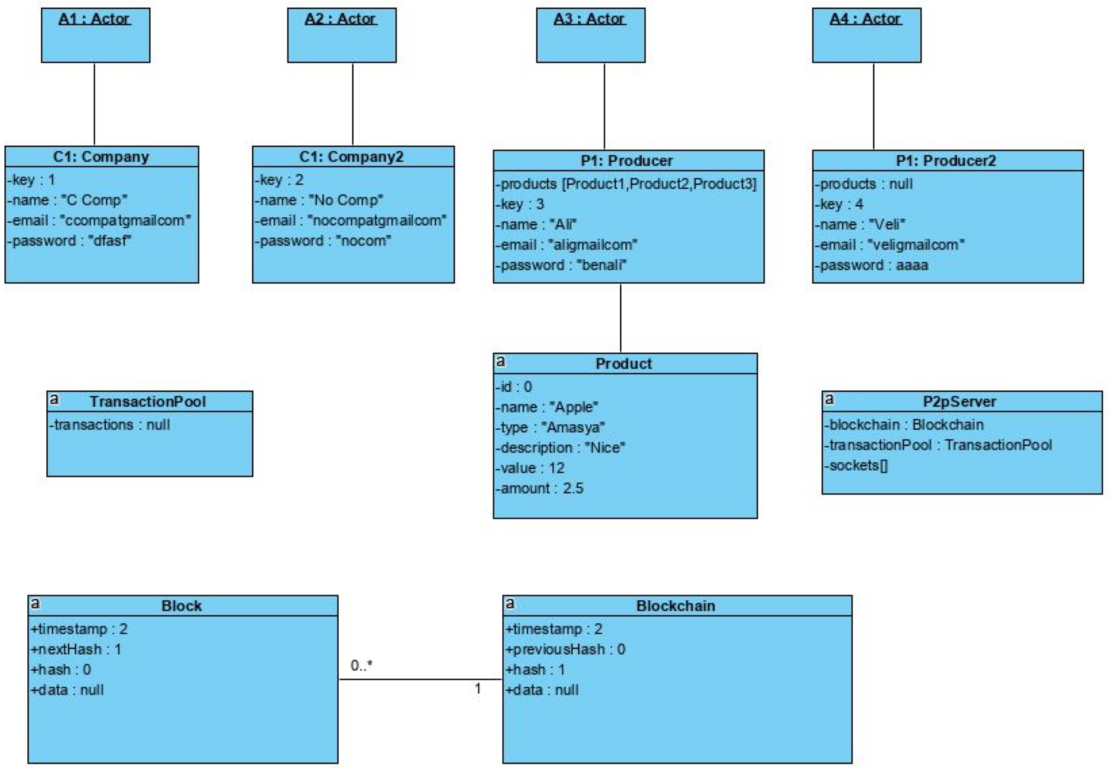
- Parameters:
    - **timestamp:** Double based time property showing the time of the creation of blockchain
    - **previousHash:** Integer based property to make a connection with the previous block
    - **hash:** Integer based property to encrypt the blockchain
    - **data:** A property that holding all the information about the objecy
- Operations:
    - **toString():** A method to write the blockchain

3.5.3.1.11 Class: P2pServer

This class represents a dummy server living in the connection established between the pairs.

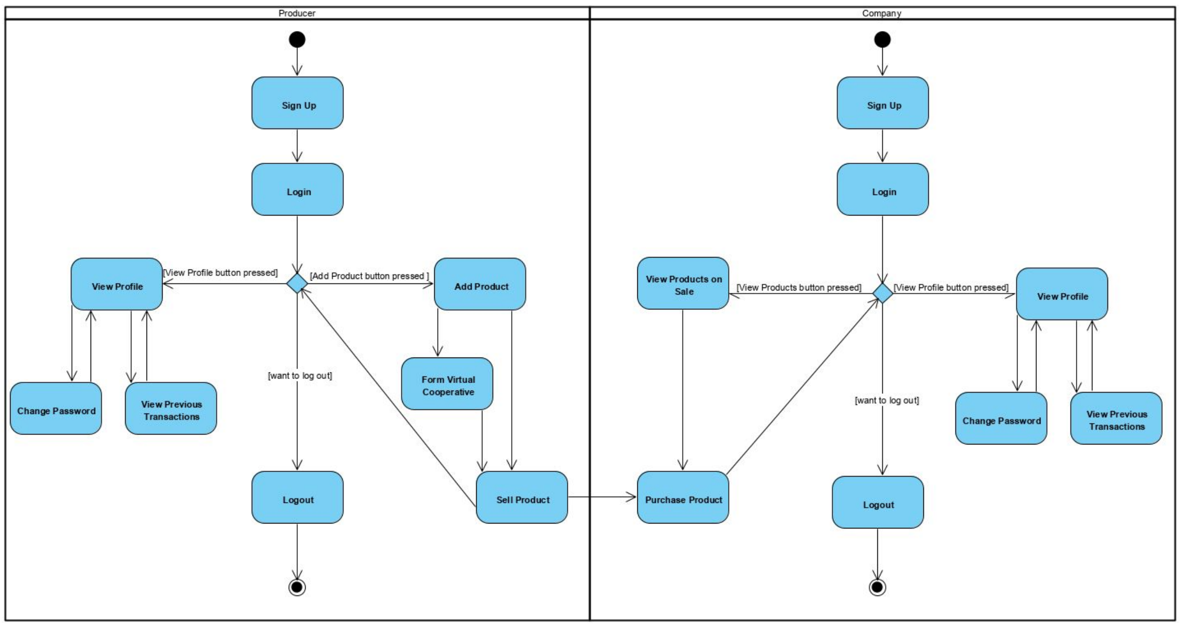
- Parameters:
    - **blockchain:** A blockchain based property
    - **transactionPool:** A trancationPool based object holding all the transactions
    - **sockets[]:** A socket based array property
- Operations:
    - **listen():** A method to listen a socket
    - **connectToPeers():** A method to connect to server
    - **connectSocket():** A method to connect to socket
    - **messageHandler(Socket socket):** A method to handling the message coming from Socket
    - **sendChain(Socket socket):** A method to send the chain to the appropriate socket
    - **sendTransacion(Transaction transaction):** A method to send the transaction
    - **brodcastClearTransactions():** A method to send the clear the transactions

3.5.3.2 Object Model



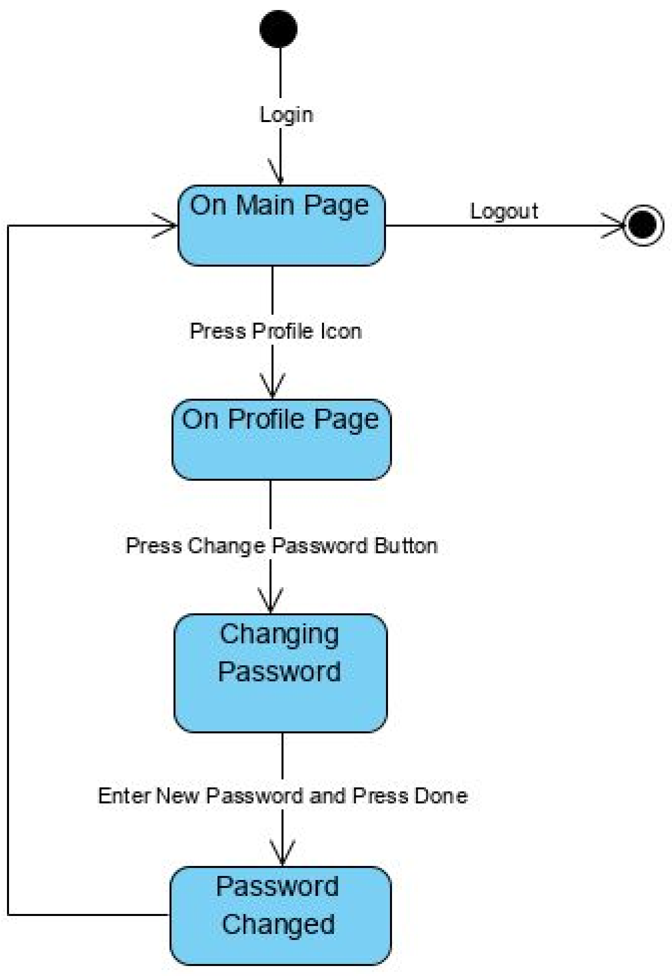
3.5.4 Dynamic Models

3.5.4.1 Activity Diagrams

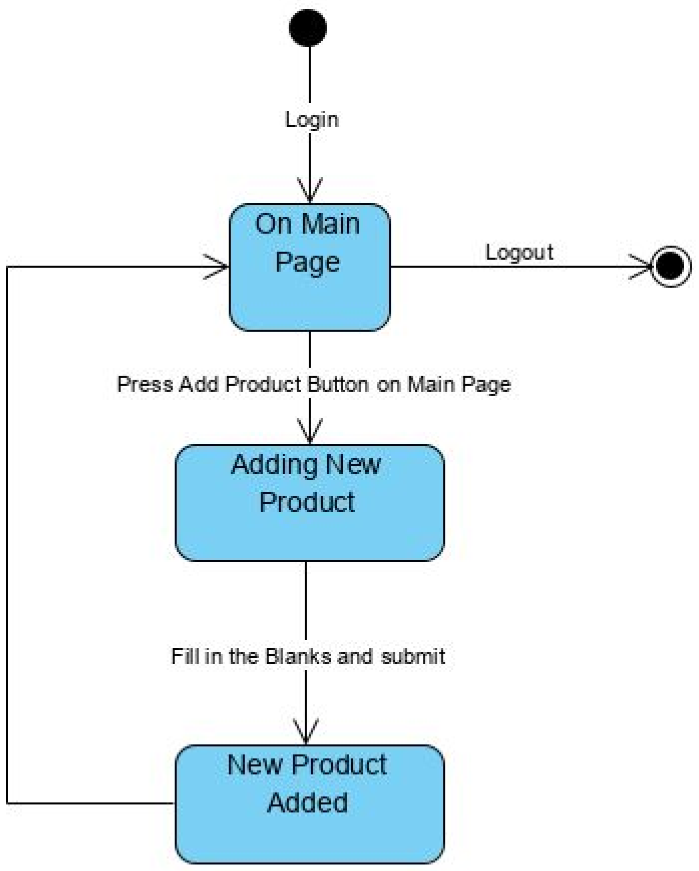


3.5.4.2 State Diagrams

3.5.4.2.1 Change Password

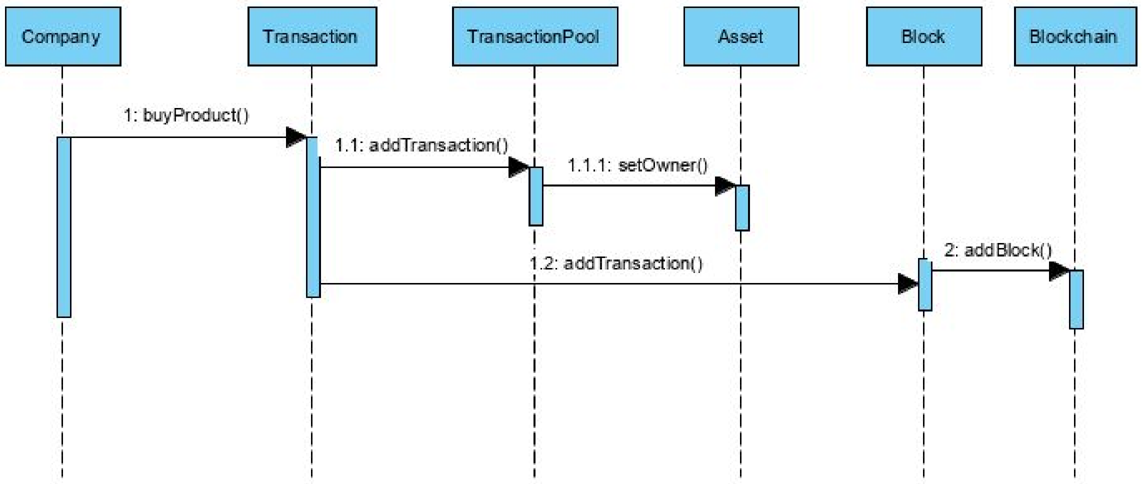


3.5.4.2.2 Add Product


Login
On Main Page
Logout
Press Add Product Button on Main Page
Adding New Product
Fill in the Blanks and submit
New Product Added

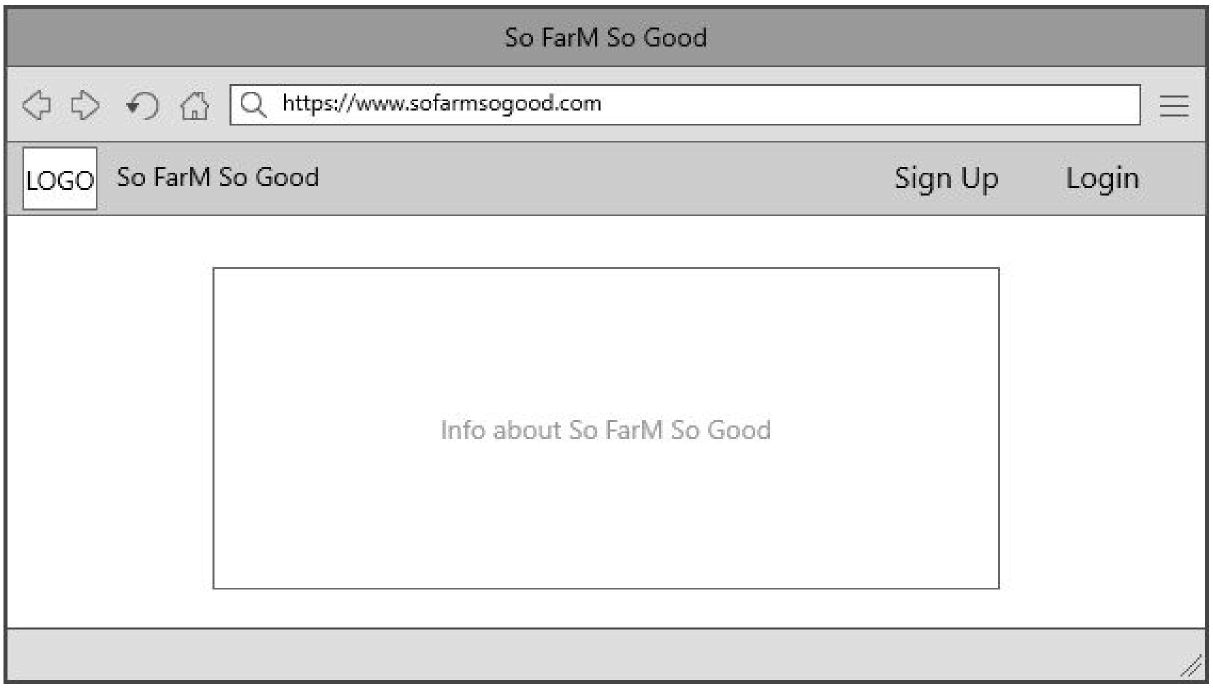
3.5.4.3 Sequence Diagrams

3.5.4.3.1 Buy Product

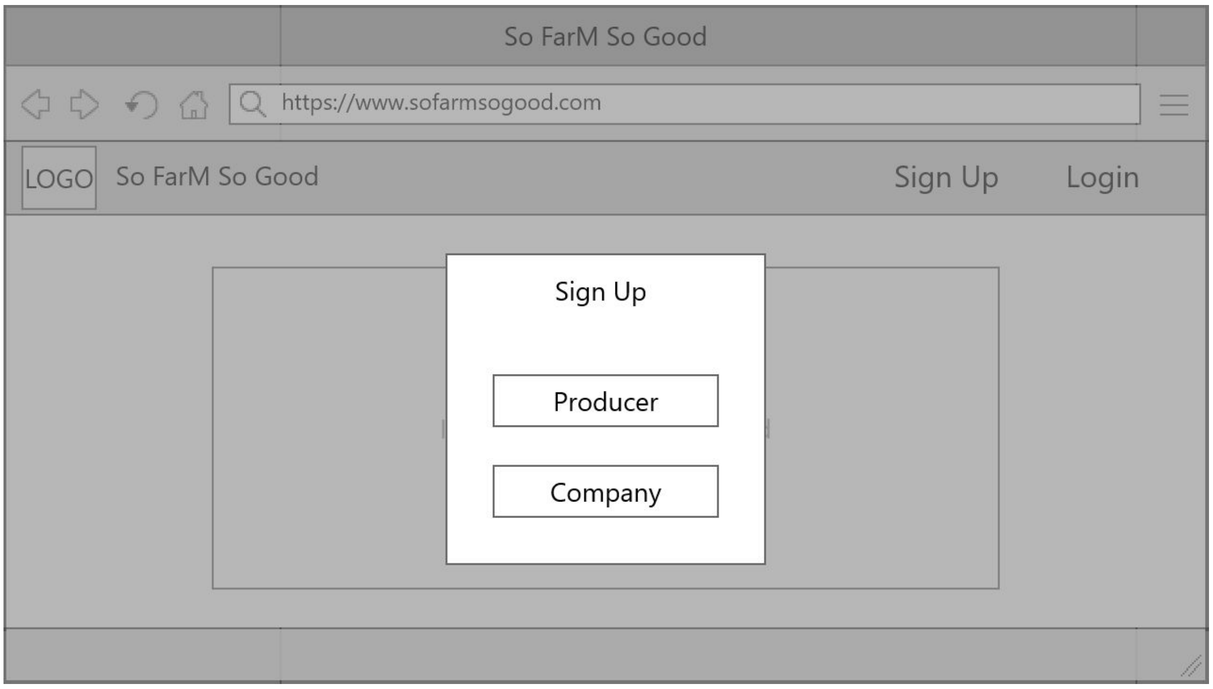


3.5.5 User Interface - Navigational Paths and Screen Mock-ups

3.5.5.1 Home Page

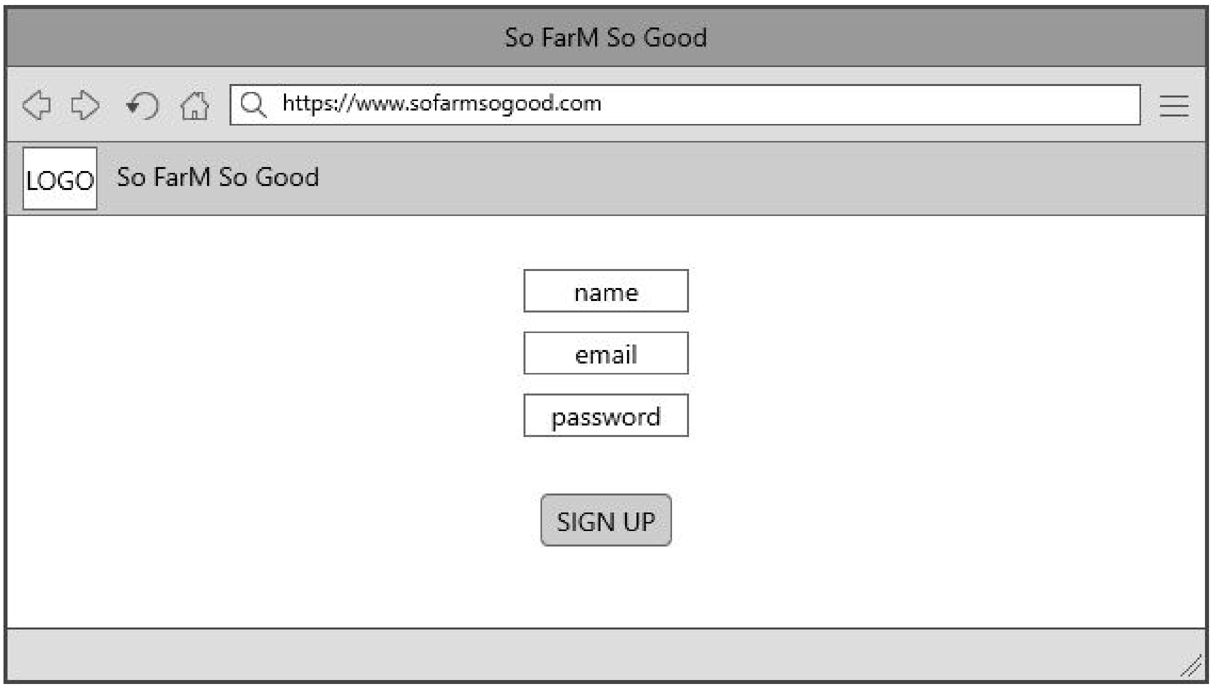


This will be the home page of So FarM So Good. This page will be where the people will get to know our platform and be able to sign up.



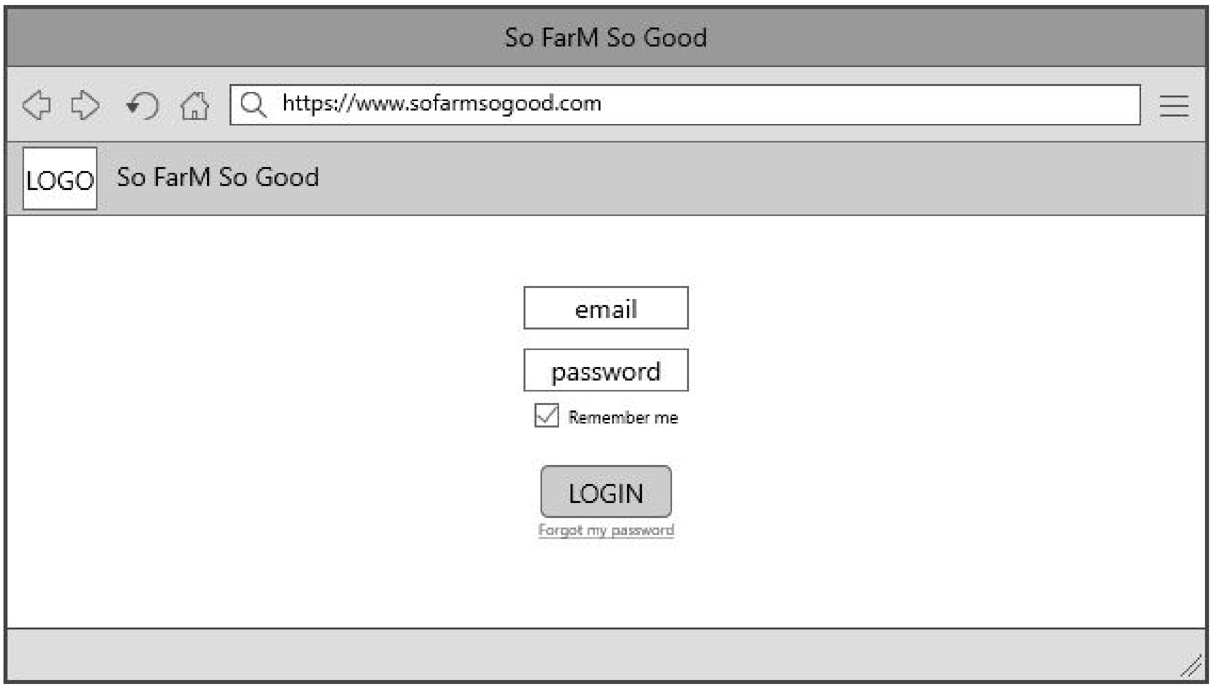
After clicking Sign Up button, a pop-up will appear which the user can select two options two sign up: Producer Sign Up or Company Sign Up.

3.5.5.2 Sign Up Page



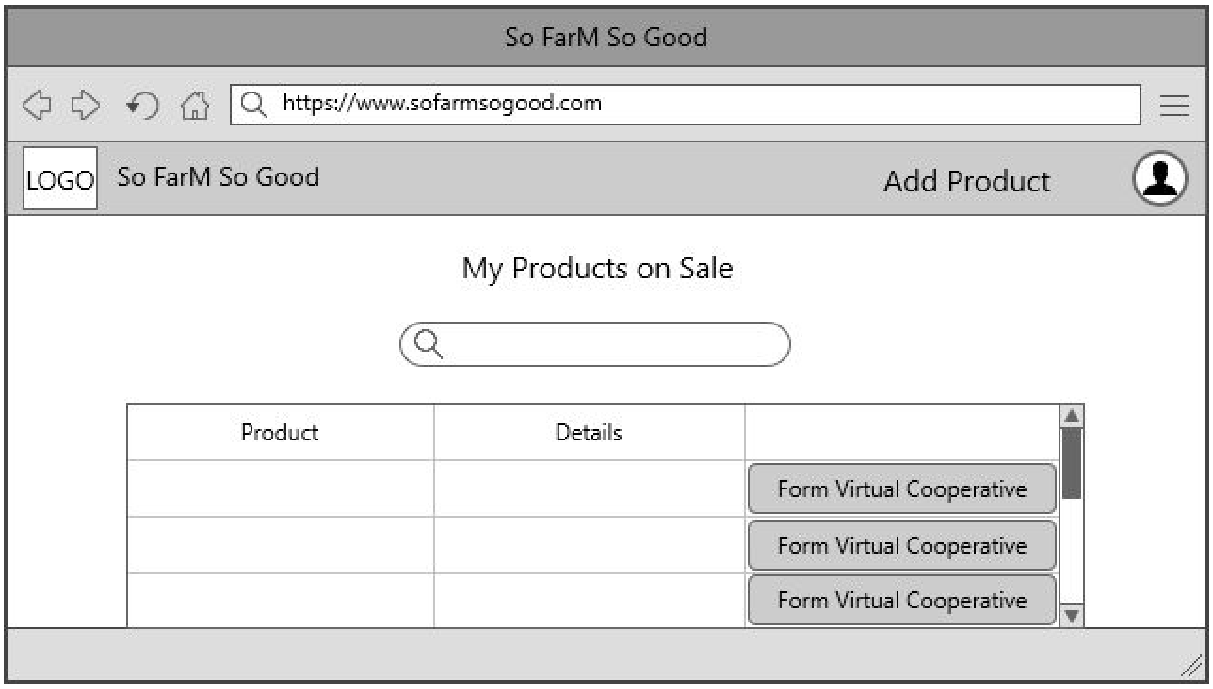
After filling in the blanks, the user will press the Sign Up button and be signed to the platform. This page will be similar for both company user and producer user. There will be design differences such as background-picture and color theme.

3.5.5.3 Login Page



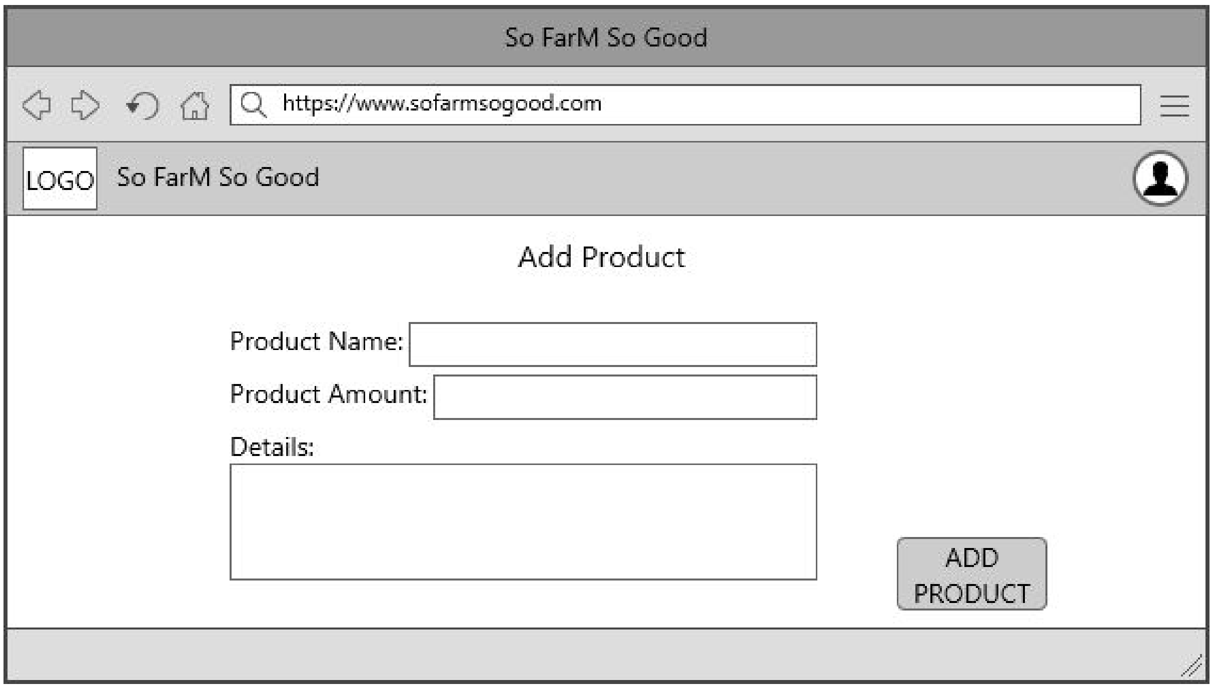
After filling in the blanks, the user will press the Login button and login to the platform. If the user does not remember their password, they can press Forgot my password text and change their password. This page will be similar for both company user and producer user just like the Sign Up page.

3.5.5.4 Producer Main Page



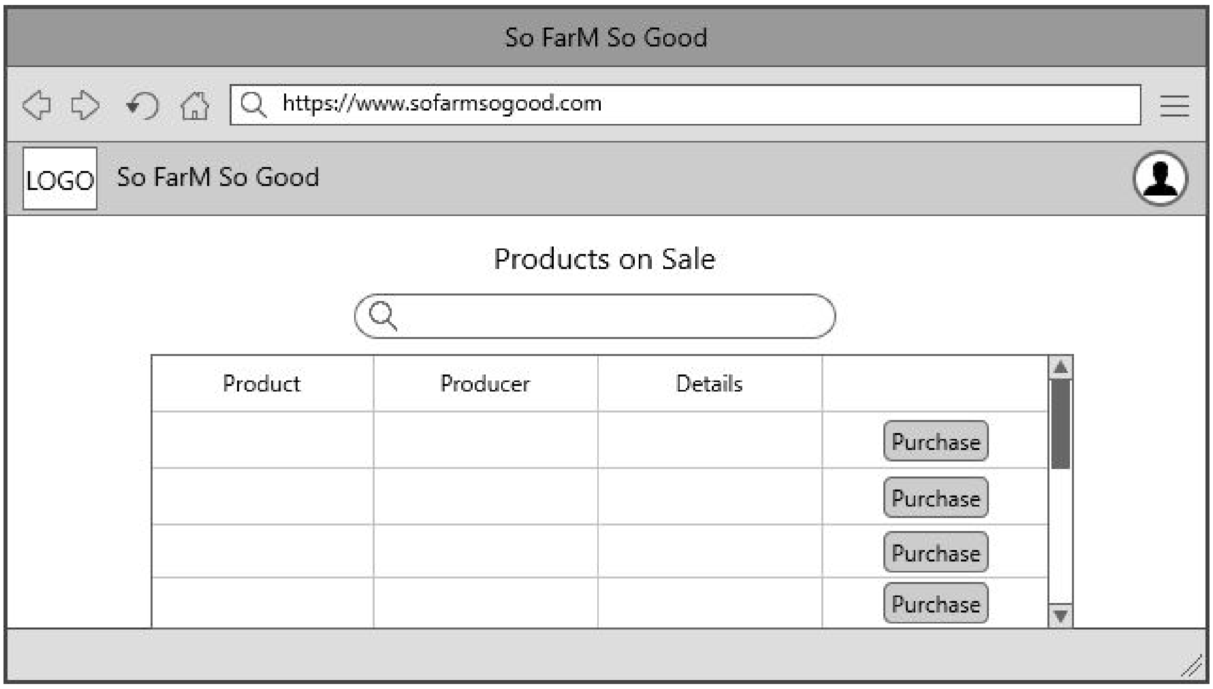
The producer can see their products which they've put on sale on their Main Page. They can search for specific products and Form Virtual Cooperatives for each product. If they want to add a new product, they must press the Add Product button, and the platform will redirect them to Add Product Page. They can also view their profile by pressing their icon on the top right of the page.

3.5.5.5 Add Product Page



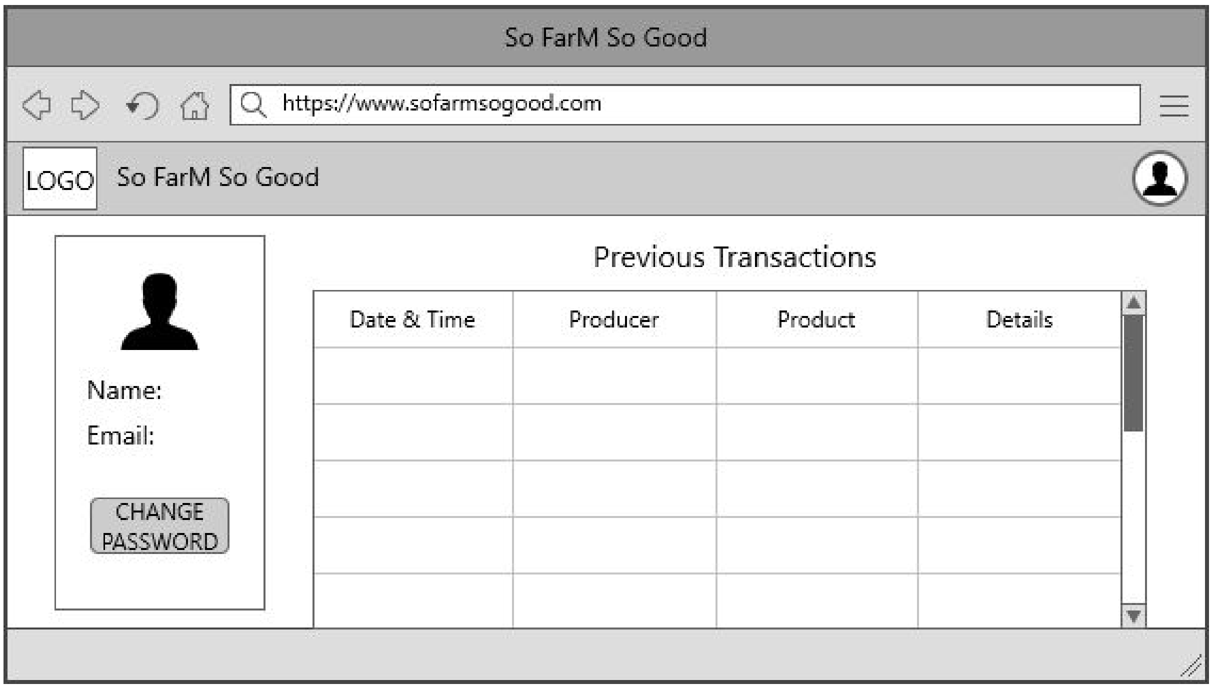
After filling in the blanks and pressing the Add Product button, the new product will be added and can be seen in the producer’s main page.

3.5.5.6 Company Main Page



The company can see products which are on sale on their Main Page. They can search for specific products and purchase them by pressing Purchase button next to each product. They can also view their profile by pressing their icon on the top right of the page.

3.5.5.7 Profile Page



In the profile page, the name, email and the list of previous transactions of the producer/company can be seen. If the user wants to change their password, they must press Change Password button. After that, an email will be sent to their email address which they can then change their passwords.This page will be similar for both company user and producer user. There will be design differences such as background-picture and color theme.

4 Other Analysis Elements

Our senior design project is separated to the two parts. First part is design stage which is decision of implementation techniques and reporting of analysis, high level stages. Second part is implementation of the project as a real working program. Design stage is also separated into 3 three sections in itself. Design stage will be composed of specification, analysis and high level sections. In the specification section, we have already scope and the basic features of the project. Moreover, at the analysis section we will visualize the SoFarmSoGood with using mock ups and diagrams. At the high level design section, we will very close to the implement the project. In this section, the project will be reported with user interfaces and core logic diagrams.

In the implementation part, we need to specify cooperative type because many different cooperatives exist according to the working principle. After that, we should decide on the payment systems according to the peer to peer styles on the Blockchain technology. Our tool will be HyperLedger as composer of the project. This tool is open source and provide us smart contracts and assistive other technologies.

4.1 Consideration of Various Factors

4.1.1 Security

We will consider safety issue by using Blockchain technology. This technology is innately secure because it utilizes powerful cryptography. Each transaction is signed with the private key and then can be further verified with a public key. If transaction data changes, the signature becomes invalid. As a result, the block is ignored and won’t make it to the chain.

4.1.2 Welfare

Our main aim is to support native farmers by establishing cooperatives. Since farming is one of the locomotives of the Turkey’s economy, welfare is directly related to the farming and farmers. In our project, sugar beet farmers will come together and resolving quota problem. After solving quota problem, farmers are not going to sell their products to crop chandlers. These crop chandlers purchase the crops much more cheaper than the government or other sugar beet cooperatives.

4.1.3 Social Factors

SoFarmSoGood will help to the farmers to control and manage crop distribution. Our software will also help the farmers to sell sugar beets even if the amount of the crop is less. Moreover, farmers will hold the data by the Blockchain and make contributions to the economy of the country.

4.2 Risks and Alternatives

First of all, entire group members will learn Blockchain technology. Since this technology is very new to us, we will start from the zero. Learning is a process which may have different responses on the every single person. Realizing and comprehensive understanding will take significant amount of time to implement the project. Because of that , we may encounter a risky situation that may prevents us completing the project before the deadline. In terms of the syllabus we have 2 semesters to implement the project. That is why, we should start the coding section of the project at the very beginning of the second semester. Moreover, the motivations and responsibilities of the each member will be determinant factor for risks. At the beginning of the each stage of the project, the workload is being allocated along team members fairly. In order to succeed the project, each member is responsible of doing his/her work within the specified time interval. Namely, any group member has a potential risk in terms of the probability of escaping allocated works. On the other hand, any team member has a potential of quitting the project completely during the design or implementation stage.

Moreover, the project may be implemented in many different ways. Those implementation and design styles will construct alternatives of the project. Firstly, we can generate various models of the cooperatives. According to the our researches, cooperatives over the world differs as working principle. For instance, some cooperatives get together and become associations in order to export their crops. Moreover, some have own agricultural markets which provides elements to the member farmers. Implementation of the Blockchain is also open to different combinations. The project may evolve in such cases. For instance, direction of the transactions will be remarkable point which. Farmers may take shares from the cooperative when they sell their product. Namely, those shares can be stored on the Blockchain system.

4.3 Project Plan

Our project plan is managed in terms of Agile Methodology Application Lifecycle Management we create our base nature into its Software Development LifeCycle phases. We decided to create our project in Agile Methodology related with our senior design project page analysis report clarification. To exemplify that choice reason, having iterative process requirement is emphasized for project's implementation phase, also they emphasized while doing design and implementation phase there could be a need for going back and edit the past phases again could be possible. These needs required us to use contemporary, less documentation but more focus on product itself methodology as Agile, so we choose Agile Scrum Process Model for our senior design project creation. We need to split project into three main operations to conduct our project in terms of Agile Application Lifecycle Management. To highlight these operations it could be written as Agile ALM bases are Governance, Development and Operations. [1] For the Governance phase we need to advance our idea to make it manageable and for the Development phase we need to develop our idea into some related platforms as implementation step. Last phase is Operations phase and this one only required to have released product, it means this step is some operations after release of our work into market. However, we only have process of development of a product, there will no experience for after releasing it to the market. That is why, we will not need Operations phase in our project's development. We added other two phase of Agile Application Lifecycle Management at the beginning of our Work Breakdown Structure it can be seen below integrated with Gantt Chart.

For having Agile Methodology we need to take several sprints to meet the demand of work splitting and taking actions for demands and requirements. As a process model of Agile Methodology we took Scrum into consideration for our project. Scrum recently is the trendy Process Model for Agile Methodology in comparing between Kandban, XP programming and so on. We planned to have iterative five sprints for Scrum's short iterative sprints as Sprint 1, Sprint 2, Sprint 3, Sprint 4 and Sprint 5 can be seen into our Gantt Chart below. Before splitting it into sprints we needed to think about our work packages which could be thought as smallest manageable unit of project elements. We numbered our work packages into our Work Breakdown Structure and we integrated them into Agile Scrum SDLC. For these manageable work packages, we have six of them, plan, analysis, design, implementation, test and integration. Related with sprints elements and requirements we edited these SDLC work packages illustrated into chart below. Then we assigned ourselves as human resources for work packages and did our work splits. For explanation of this human resources assignment titles and work splitted information can be seen on the WP table below provided by report template. In Agile Methodology Scrum Process Model we need to assign our human resources into needs, it means there is no obvious borders work splitting for project development. For clarification for that way, there is no goal keeper and strikers for our Scrum model into the field most of works can be changed into needs and everyone needs to be capable for doing every related phases of work to be agile in development.

Table 01: Factors that can affect analysis and design.

	Effect level	Effect
Public health	0/10	Our project is not related with the public health.
Public safety	10/10	Blockchain Technology is responsible for the ensuring the data safety.
Public welfare	9/10	By the online cooperatives sugar beet farmers will increase the profit.

Global factors	0/10	As we are not aiming to the export the sugar beet, we do not need to the global factors.
Cultural factors	0/10	No correlation with the cultural factors.
Social factors	10/10	Main aim is to support the farmers.Because of that, social factors has a significant place in the project.

Table 02: Risks

	Likelihood	Effect on the project	B Plan Summary
Having a trouble with learning Blockchain	Unlikely	It totally leads changes all process of our project. Our system needs to be in exactly different path.	Having an proper, regular DB relation between harvest rates of producers as sellers and selling products and companies as buyers relations. e.g. using SQL server
Network problems occurences by Docker	Rare	It leads to having exactly different network solutions. We need to have another programs for usable network between blockchain actors and relations.	Having another network solutions between blockchain relations. e.g. using VirtualBox
Changing requirement related with having coin in system	Possible	It requires total system extension in logic, we need to re-design and re-implement some changes and Angular is not well supported with some bigger coin kinds	Having an more coin-based front-end side extension for our JS frameworks such as Vue or very well supported with Eterium coins, ReactJS,etc

Table 03: List of work packages

WP#	Work package title	Leader	Members involved
WP1	Plan	Giray Baha Kezer	Kaan Atakan Öztürk, Fazilet Simge Er, Melih Ünsal
WP2	Design	Fazilet Simge Er	Giray Baha Kezer, Kaan Atakan Öztürk
WP3	Implementation	Melih Ünsal	Giray Baha Kezer, Kaan Atakan Öztürk, Fazilet Simge Er
WP4	Test	Kaan Atakan Öztürk	Fazilet Simge Er, Giray Baha Kezer

WP 1: *Plan*$_i$ *i=1,2,3,4,5*
Start date: *for Plan Sprint1* **End date:** *ALM Plan:* *10.14.19* *10.14.19* *Sprint Plan:* *10.15.19* *10.15.19* *for Plan Sprint2* *11.25.19* *11.25.19* *for Plan Sprint3*

1.9.20 *1.9.20* *for Plan Sprint4* ***Start date:*** ***End date:*** *2.25.20* *2.25.20* *for Plan Sprint5* *4.10.20* *4.20.20*			
Leader:	*Giray Baha Kezer*	**Members involved:**	*Kaan Atakan Öztürk* *Fazilet Simge Er* *Melih Ünsal*
Objectives: *Plan work packages consists 5 Sprint every Sprint has its own planning phase. Related with Agile SDLC common and iterative one phase is planning. In Sprint 1 we need to decide ALM relevance planning and also first Sprint planning. In addition Project Specification plan is needed to be done that phase. In Sprint 2 we need to plan our Sprint 2 and next step chain.Also we needed to plan our Analysis report in that phase. In Sprint 3 we need to plan our High Level Design report and particular that Sprint in that phase. In Sprint 4 we need to plan our current Sprint in that step. In Sprint 5 we need to plan our current Sprint and also Low Level Design report in that phase. Also in all particular Sprint Planning work packages we need to decide backlogs for other steps.*			
Tasks: ***Task 1.1 <Sprint 1 Plan Task>:*** *This task includes determining the milestones and "Definition of Done" in particular that sprint. Also Sprint backlogs need to be planned in that phase for next steps of Sprint and next strategy for assigned splitted works.* ***Task 1.2 <Sprint 2 Plan Task>:*** *This task includes determining the milestones and "Definition of Done" in particular that sprint. Also Sprint backlogs need to be planned in that phase for next steps of Sprint and next strategy for assigned splitted works.* ***Task 1.3 <Sprint 3 Plan Task>:*** *This task includes determining the milestones and "Definition of Done" in particular that sprint. Also Sprint backlogs need to be planned in that phase for next steps of Sprint and next strategy for assigned splitted works.* ***Task 1.3 <Sprint 4 Plan Task>:*** *This task includes determining the milestones and "Definition of Done" in particular that sprint. Also Sprint backlogs need to be planned in that phase for next steps of Sprint and next strategy for assigned splitted works.*			

Task 1.5 <Sprint 5 Plan Task>: *This task includes determining the milestones and "Definition of Done" in particular that sprint. Also Sprint backlogs need to be planned in that phase for next steps of Sprint and next strategy for assigned splitted works.*			
Deliverables ***D1.1:*** *Project Specifications Planning* ***D1.2:*** *Analysis Report Planning* ***D1.3:*** *High Level Design Planning* ***D1.4:*** *Low Level Design Planning*			
WP 2: *Design$_i$* *i=1,2,3,4,5*			
Start date: *for Design Sprint1* **End date:** *10.25.19* *11.11.19* *for Design Sprint2* *11.26.19* *12.23.19* *for Design Sprint3* *1.15.20* *2.17.20* *for Design Sprint4* *3.4.20* *3.10.20* *for Design Sprint5* *4.14.20* *4.20.20*			
Leader:	*Fazilet Simge Er*	**Members involved:**	*Giray Baha Kezer, Kaan Atakan Öztürk*
Objectives: *Design work packages consists of 5 Sprint phases and every Sprints have their own design phases. Related with Agile SDLC common and iterative one phase is design. In Sprint 1 we need to more focus on designing phase in terms of architectural*			

and technical steps of our project. More commonly technical questions could be part of that phase in particular Sprint1. In Sprint 2 we need to have more rare questions for technical and architectural side for our project. In that phase design part should be more specific than the first one. In Sprint 3 we need more advanced design part for our ongoing work according to technical and architectural side. For our UI parts needed to be advanced than just having mockups in that phase. In Sprint 4 Rare and related questions needed to be that phase. We need to more focus on designing phase in that particular Sprint 4. We need real time testable designing phases in that Sprints more accordingly. These iterative steps cover the previous parts and feeded by them that is why we need more advanced in same side with every next Sprints. In Sprint 5 we need to decide last UI form of our project in that phase. Related with its being last Sprint we need to have solutions of all related architectural and technical questions also UI parts that we want to create.

Tasks:

Task 1.1 <Sprint 1 Design Task>: *This task includes determining architectural phases of particular sprints between project stakeholders. All related technical questions need to be part of Sprint Design part.*

Task 1.2 <Sprint 2 Design Task>: *This task includes determining architectural phases of particular sprints between project stakeholders. All related technical questions need to be part of Sprint Design part. In Sprint 2 we have some ongoing process in that sprint and we need to integrate more complex architectural and technical questions than past Sprint iteratively.*

Task 1.3 <Sprint 3 Design Task>: *This task includes determining architectural phases of particular sprints between project stakeholders. All related technical questions need to be part of Sprint Design part. In Sprint 3 we have some ongoing process in that sprint and we need to integrate more complex architectural and technical questions than past Sprint iteratively.*

Task 1.4 <Sprint 4 Design Task>: *This task includes determining architectural phases of particular sprints between project stakeholders. All related technical questions need to be part of Sprint Design part. In Sprint 4 we have some ongoing process in that sprint and we need to integrate more complex architectural and technical questions than past Sprint iteratively.*

Task 1.5 <Sprint 5 Design Task>: *This task includes determining architectural phases of particular sprints between project stakeholders. All related technical questions need to be part of Sprint Design part. In Sprint 5 we have some nearly done process in that sprint and we need to focus on other phases more than designing phase in that particular Sprint. Because we need to release the product and we are in the last Sprint.*

D1.1: *Project Specifications Designing*

D1.2: *Analysis Report Designing*

D1.3: *High Level Design Designing*

D1.4: *Low Level Design Designing*

WP 3: *Implementation$_i$ i=1,2,3,4,5*			
Start date: *for Implementation Sprint1* *11.13.19*	**End date:** *11.23.19*		
for Implementation Sprint2 *1.1.20*	*1.8.20*		
for Implementation Sprint3 *1.15.20*	*2.24.20*		
for Implementation Sprint4 *2.26.20*	*4.9.20*		
for Implementation Sprint5 *4.14.20*	*5.4.20*		
Leader:	*Melih Ünsal*	**Members involved:**	*Giray Baha Kezer, Kaan Atakan Öztürk,* *Fazilet Simge Er*

Objectives: *In Sprint 1 we have nearly non implementation phase because we do not have enough research on that time and there is not much thing to implemented in a new technology such as Blockchain. In Sprint 2 related with some research and an accurate analysis of project we could start some implementation and its duration much higher than* *the beginning of first iteration can be seen in Gannt Chart. In Sprint 3 we need to have more focus on implementation related with project natural process between ongoing new sprints. In Sprint 4 we need nearly do all of the implementation part and focus only impelementation in terms of projects specification requirements in the last phase. Related feedbacks of client or supervisors will be guiding for the last sprint for releasing the product.*

Tasks:

Task 1.1 <Sprint 1 Implementation>: *Purpose of having that task in this kind of early stage is possible to have some related research for prototyping of new technology such as Blockchain, Docker and so on.*

Task 1.2 <Sprint 2 Implementation>: *Purpose of having that implementation phase is related more advanced research and analysis requires more concrete coding and prototyping in that new technology.*

Task 1.3 <Sprint 3 Implementation>: *Purpose of having that implementation phase is related more advanced research and analysis requires more concrete coding and prototyping in that new technology. Feeded past sprints could require more implementation phase in that sprint we need to be more accurate in coding and implementation in that time. Also related research could be required more implementation into Angular or other types of UI prototyping.*

Task 1.4 <Sprint 4 Implementation>: *Purpose of having that implementation phase is related more advanced research and analysis requires more concrete coding and prototyping in that new technology. Feeded past sprints could require more implementation phase in that sprint we need to be more accurate in coding and implementation in that time. Also related research could be required more implementation into Angular or other types of UI prototyping. We need to have nearly finished in implementation because we only have one sprint and last sprint requires more preparation for releasing the product and last testing phases and so on.*

Task 1.5 <Sprint 5 Implementation>: *Purpose of having that implementation phase is related having only one sprint at that time requires more focused implementation according to your last research and last decisions. It also requires feedbacks based requirements' coding in pure concentration and it is critical for releasing the product into market.*

Deliverables
D1.1: *Project Specifications Implementation*
D1.2: *Analysis Report Implementation*
D1.3: *High Level Design Implementation*
D1.4: *Low Level Design Implementation*

WP 4: *Test$_i$* *i=1,2,3,4,5*			
Start date: *for Test Sprint1* **End date:** *10.16.19* *11.14.19* *for Test Sprint2* *11.26.19* *1.8.20* *for Test Sprint3* *1.10.20* *2.24.20* *for Test Sprint4* *2.26.20* *4.9.20* *for Test Sprint5* *4.14.20* *5.11.20*			
Leader:	*Kaan Atakan Öztürk*	**Members involved:**	*Fazilet Simge Er, Giray Baha Kezer*
Objectives: *Testing is started after plan phase and generally it continues until the end of sprint. In Sprint 1 we have a different case, related with not having adequate research and works testing has some delay to start inside sprint. There is more focus on other phases at that sprint. Except for Sprint 1 we all have same testing manner, testing is started after planning is done and it is spreaded nearly until the end of sprint. Only in*			

Sprint 3 we have exactly finish to finish relation between testing and Sprint 3 itself. Testing starts after planning and continues at the end of particular sprint. Related with having critical changes into implementation we become more needed to testing phases into debugging for coding. Related with coding pattern of project testing becomes more critical.

Tasks:

Task 1.1 <Sprint 1 Testing>: *Purpose of that phase in Sprint 1 is possibility to have some coding advancements and some possible unit testing cases. Also we need some verification and validation research on that particular sprint. However, having testing need is less than other sprints.*

Task 1.1 <Sprint 2 Testing>: *Purpose of that phase in Sprint 2 is possibility to have some more coding advancements and some possible unit testing cases. It could be possible to have some functions and class enhancements related test cases and test process will be needed. Having testing phases need is more than past iterative sprint. It starts with planning ending and continues to nearly end of sprint.*

Task 1.1 <Sprint 3 Testing>: *Purpose of that phase in Sprint 3 is possibility to have more coding advancements. Related with its process but we could need some integration testing, unit testing, some interface testing, and so on. It could be possible to have some functions and class enhancements related test cases and test process will be needed. Having testing phases need is more than past iterative sprint. It has exact the finish to finish relation between sprint 3 itself. It starts with planning ending and continues until the end of that sprint.*

Task 1.1 <Sprint 4 Testing>: *Purpose of that phase in Sprint 4 is possibility to have more coding advancements. Related with its process but we could need some integration testing, unit testing, some interface testing, and so on. It could be possible to have some functions and class enhancements related test cases and test process will be needed. In that scope we need more and more testing because of Scrum's synchronously testing and coding relation. It starts after planning phase and continues nearly to the end of sprint.*

Task 1.1 <Sprint 5 Testing>: *Purpose of that phase in Sprint 4 is possibility to have more coding advancements. Related with its process but we could need some integration testing, unit testing, some interface testing, and so on. It could be possible to have some functions and class enhancements related test cases and test process will be needed. Related with lack of time we have shorter last sprint in our project management lifecycle. Thereby, we have more focus on testing for releasing our final product and lack of time in comparison between other sprints.*

Deliverables

D1.1: *Project Specifications Testing*

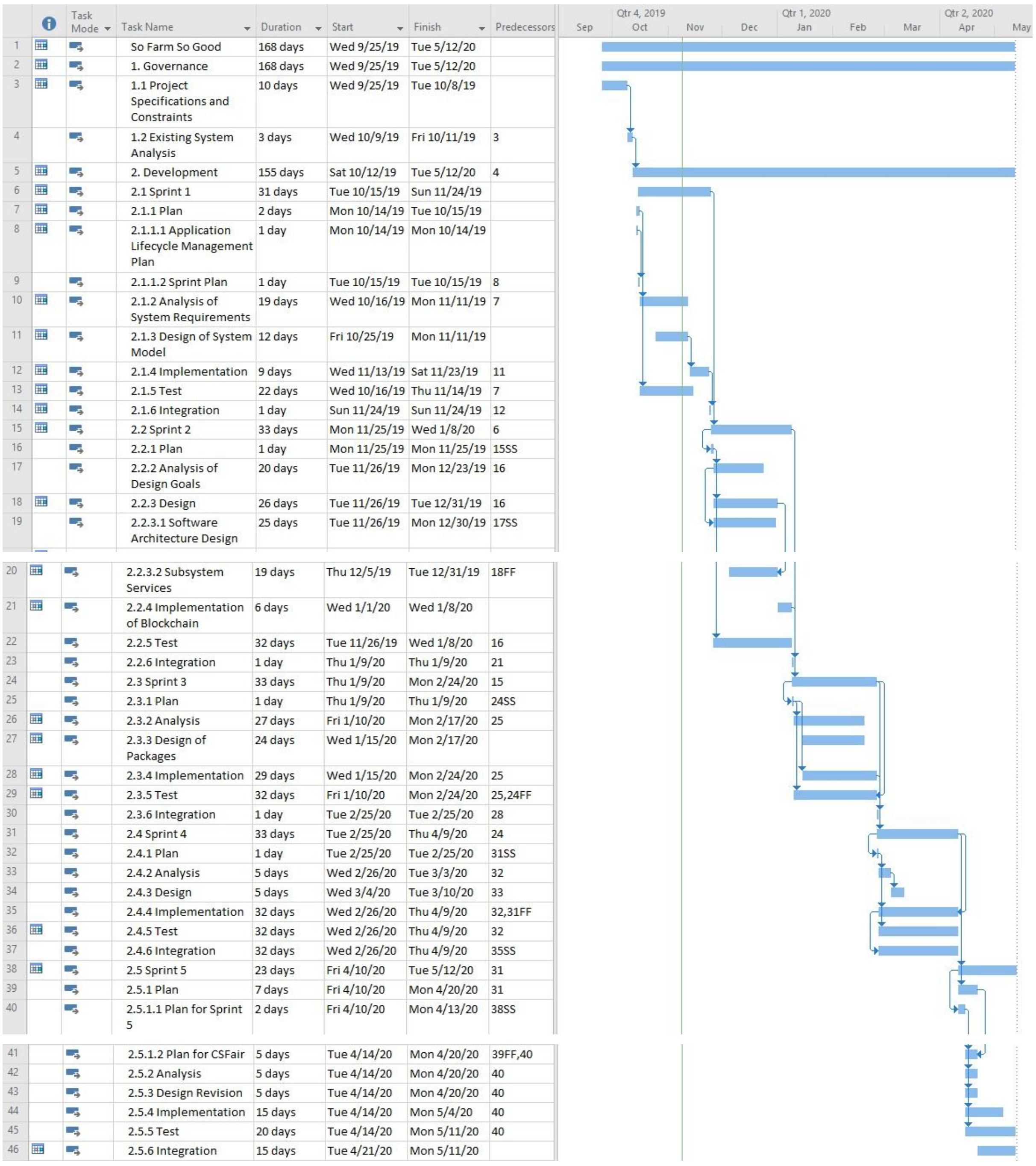
D1.2: *Analysis Report Testing*

D1.3: *High Level Design Testing*

D1.4: *Low Level Design Testing*

4.3.1 Project Management Tools

We prefer Github platform as our project management tool; there is several reasons; github is offering us not only developing our work coordinately online but also it makes our work trackable, easily updatable it means working transparently and on schedule at the same time.[2] We could organize our project and make them splitted into manageable work packages and so arrange the related milestones and after that whole process we could assign them among us easily at the same platform. Tasks could be easily inserted and issues can be easily worked on, labelling could be easily done and also mentions could be easily inserted into working elements while developing coordinately, that is why we choose Github to be able to work faster, we need speed for having Agile Methodology, Scrum Process Model development. At that lines the most important thing is product itself and it requires more speed. We took JIRA as our second helpful choice for project management tool. Related with having Agile Methodology JIRA is one of the best known and usable project management tool in software world.[3] JIRA is integrated with Agile principles and easily usable with sprint permissions, determining issue types, creating related workflows and easily editing custom dashboards.[4] JIRA is also supporting many Scrum charts, figures as Burndown, Sprint report, cumulative flow diagrams and control charts. This support offers us a chance to create and work easily with our project developing side and management side simultaneously. We decided to use Github and JIRA simultaneously in terms of needs, for splitting our projects into work packages as manageable smallest units and created project in its site, so we could send an invitation to our supervisor to check our works. Also JIRA will be used for working with Scrum features as charts and administrating Scrum features while developing the project. In addition to our selected tools, we investigated other ones named ProWorkFlow. Even if it has similar features with JIRA we do not satisfied with its pricing attitudes and we searched for free management tools.



	ℹ	Task Name	Duration	Start	Finish	Predecessors
1	▦	So Farm So Good	168 days	Wed 9/25/19	Tue 5/12/20	
2	▦	1. Governance	168 days	Wed 9/25/19	Tue 5/12/20	
3	▦	1.1 Project Specifications and Constraints	10 days	Wed 9/25/19	Tue 10/8/19	
4		1.2 Existing System Analysis	3 days	Wed 10/9/19	Fri 10/11/19	3
5	▦	2. Development	155 days	Sat 10/12/19	Tue 5/12/20	4
6	▦	2.1 Sprint 1	31 days	Tue 10/15/19	Sun 11/24/19	
7	▦	2.1.1 Plan	2 days	Mon 10/14/19	Tue 10/15/19	
8	▦	2.1.1.1 Application Lifecycle Management Plan	1 day	Mon 10/14/19	Mon 10/14/19	
9		2.1.1.2 Sprint Plan	1 day	Tue 10/15/19	Tue 10/15/19	8
10	▦	2.1.2 Analysis of System Requirements	19 days	Wed 10/16/19	Mon 11/11/19	7
11	▦	2.1.3 Design of System Model	12 days	Fri 10/25/19	Mon 11/11/19	
12	▦	2.1.4 Implementation	9 days	Wed 11/13/19	Sat 11/23/19	11
13	▦	2.1.5 Test	22 days	Wed 10/16/19	Thu 11/14/19	7
14	▦	2.1.6 Integration	1 day	Sun 11/24/19	Sun 11/24/19	12
15	▦	2.2 Sprint 2	33 days	Mon 11/25/19	Wed 1/8/20	6
16		2.2.1 Plan	1 day	Mon 11/25/19	Mon 11/25/19	15SS
17		2.2.2 Analysis of Design Goals	20 days	Tue 11/26/19	Mon 12/23/19	16
18	▦	2.2.3 Design	26 days	Tue 11/26/19	Tue 12/31/19	16
19		2.2.3.1 Software Architecture Design	25 days	Tue 11/26/19	Mon 12/30/19	17SS
20	▦	2.2.3.2 Subsystem Services	19 days	Thu 12/5/19	Tue 12/31/19	18FF
21	▦	2.2.4 Implementation of Blockchain	6 days	Wed 1/1/20	Wed 1/8/20	
22		2.2.5 Test	32 days	Tue 11/26/19	Wed 1/8/20	16
23		2.2.6 Integration	1 day	Thu 1/9/20	Thu 1/9/20	21
24		2.3 Sprint 3	33 days	Thu 1/9/20	Mon 2/24/20	15
25		2.3.1 Plan	1 day	Thu 1/9/20	Thu 1/9/20	24SS
26	▦	2.3.2 Analysis	27 days	Fri 1/10/20	Mon 2/17/20	25
27	▦	2.3.3 Design of Packages	24 days	Wed 1/15/20	Mon 2/17/20	
28	▦	2.3.4 Implementation	29 days	Wed 1/15/20	Mon 2/24/20	25
29	▦	2.3.5 Test	32 days	Fri 1/10/20	Mon 2/24/20	25,24FF
30		2.3.6 Integration	1 day	Tue 2/25/20	Tue 2/25/20	28
31		2.4 Sprint 4	33 days	Tue 2/25/20	Thu 4/9/20	24
32		2.4.1 Plan	1 day	Tue 2/25/20	Tue 2/25/20	31SS
33		2.4.2 Analysis	5 days	Wed 2/26/20	Tue 3/3/20	32
34		2.4.3 Design	5 days	Wed 3/4/20	Tue 3/10/20	33
35		2.4.4 Implementation	32 days	Wed 2/26/20	Thu 4/9/20	32,31FF
36	▦	2.4.5 Test	32 days	Wed 2/26/20	Thu 4/9/20	32
37		2.4.6 Integration	32 days	Wed 2/26/20	Thu 4/9/20	35SS
38	▦	2.5 Sprint 5	23 days	Fri 4/10/20	Tue 5/12/20	31
39		2.5.1 Plan	7 days	Fri 4/10/20	Mon 4/20/20	31
40		2.5.1.1 Plan for Sprint 5	2 days	Fri 4/10/20	Mon 4/13/20	38SS
41		2.5.1.2 Plan for CSFair	5 days	Tue 4/14/20	Mon 4/20/20	39FF,40
42		2.5.2 Analysis	5 days	Tue 4/14/20	Mon 4/20/20	40
43		2.5.3 Design Revision	5 days	Tue 4/14/20	Mon 4/20/20	40
44		2.5.4 Implementation	15 days	Tue 4/14/20	Mon 5/4/20	40
45		2.5.5 Test	20 days	Tue 4/14/20	Mon 5/11/20	40
46	▦	2.5.6 Integration	15 days	Tue 4/21/20	Mon 5/11/20	

Figure [1]: Gantt Chart Visualization of So Farm So Good project

4.4 Ensuring Proper Team-Work

In the early projects, there has been imbalance between the requirements because there were no such a project management plan. In this project, we are going to seperate the project into several packages then for every package there is going to be a package manager that is responsible for this part of the project. Since we have Scrum Process Model, we are going to see who have completed his/her assignment so that we are able to give feedback to any member of the group. Thus, these will increase our performance and also ensure the proper team-work.

4.5 Ethics and Professional Responsibilities

Nowadays, one of the most important problem in the modern world is security and privacy of personal data. Most people are concerning what is this app going to do with our data. In this project, all the transactions and user data will be hold by blockchain technology which is a populer decentralized data technology. By this technology, user data will be secure and cannot be decrypted by any other person since the system automatically updates itself by the speed that no computer can achieve. The user data are not also shared by any other person or company no matter how much money they offer.

4.6 New Knowledge and Learning Strategies

As a new trendy technology we need to learn blockchain. What is this network between ledgers, how its transactions are processed via chains. What is peer cards and what these cards hold as commodities and assets. What is the relation between them? What are the ledgers that form blockchain, what are the ledger types and what type of connection do they have such as distributed and centralized connection? What is blockchain node, what is the importance of the first node and what does it contain? How do nodes make a connection with each other? What is private key for a node and how do the relations between nodes encrypted? How to make a transaction and how does blockchain hold these transactions? What are the actors and what are the relationships between actors? What is docker and why to need it for network? What is Hyperledger, how to use it for blockchain? What is composer and composer generated file? Is there any ethical or juristical concern behind making the payments on the application rather than only holding the transaction data by blockchain? We are going to answer these questions throughout the semester by internet research and interviewing the experienced people around us.

5 Glossary

ALM	Application Lifecycle Management specification,design,development and testing of a software application.	17,20
WP	Work package is the smallest manageable unit.	17,20,21,23
XP	Extreme programming is a agile software development framework that aims to produce higher quality software.	17
HyperLedger	Hyperledger is an open source collaborative effort created to advance cross-industry blockchain technologies.	15
SDLC	Software Development Life Cycle (SDLC) is a process used by the software industry to design,develop and test high quality softwares.	17,20

Glossary for any domain-specific terms you use in your report.

6 References

Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[1] D. Chappell, “What is Application Lifecycle Management?,” *What is Application Lifecycle Management?*, 2014.

[2] “GitHub features: Integrated project management tools,” *GitHub*. [Online]. Available: https://github.com/features/project-management/. [Accessed: 11-Nov-2019].

[3] J. M. D. Santos, “Top 10 Best Project Management Software & Tools in 2019,” *Project*, 08-Nov-2019. [Online]. Available: https://project-management.com/top-10-project-management-software/. [Accessed: 11-Nov-2019].

[4] Atlassian, “Agile tools for software teams - Jira Software,” *Atlassian*. [Online]. Available: https://www.atlassian.com/software/jira/agile. [Accessed: 11-Nov-2019].