**Chapter 3: Design**

# Introduction to design

The third stage of software development that can be considered as a bridge between the analysis of a software requirement and implementation is known as design. This stage helps transform the gathered system requirement into suitable logical form with an intent to help the programmers in coding and implementation. This is one of the crucial step of software development that usually answers the **“How?”** aspects of software development. (i.e. how the system should look like, how it should perform etc.) Various diagrammatic models are created using different tools so as to answer the above questions.

The importance of this step on my project can be highlighted from the following points.

* It clarifies the ways and paths to be taken during the coding and implementation of project Ilam Tea Garden.
* Prototyping used in this process will allow the local people of my place to visualize what the system is going to look and function like.
* It makes uses of the different classes and objects generated during analysis and helps to create different diagrams which eventually helps to ease the programmers task.

For project Ilam Tea Garden, I will be designing 4 different models which are listed below.

* Structural model
* Behavior Model
* Database Modelling
* Architectural Model
* Prototyping

## **Structural Model**

Models that shows how different components, (usually objects and classes) and their relationships are organized in a system is known as structural modelling. Structural model diagrams reflect the static relationship of the different components in a system. Below are different structural model diagrams.

## Class Diagram (Final)

This class diagram makes use of relationships of objects at a greater depth and is more informative than an initial class diagram shown in the analysis stage. It is also called the blueprint of the system since the whole system is based on this diagram.

The justification for creating an in-depth class diagram for my project is given below.

* It makes my system more robust since the classes, objects and their interactions are pre-analyzed and determined without having to actually write programming codes.
* It acts as a structural foundation for writing the programming codes.
* Maintaining the system will be easy by referring to class diagrams instead of going through each lines of codes which will save a lot of time.

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Notations Used | Notation Name | Description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Data Flow Diagram

The diagram that represents the flow of data within a system in an informative manner is known as Data Flow Diagram (**DFD).** It is a traditional approach to structural modelling which can be manual, automated as well as both at a time.

The justification for creating a Data Flow Diagram for my project is given below.

* The local people will clearly understand what the system represents because of the efficient communication that DFD provides.
* Data Flow diagrams are clearly understood by technical as well as non-technical audiences.
* It helps to describe the boundaries and scope of the system in a neat and clear way.

The notations used for creating data flow diagram of my project is explained below.

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Notations Used | Notation Name | Description |
| 1 |  | External Entity | It represents any entities outside of the system to operate the functionality of the system. |
| 2 |  | Process | It represents a certain functionality carried out in the system |
| 3 |  | Data Store | It represents a location that has different data. |
| 4 |  | Data Flow | It represents the directional flow of action from one point to another. |

The data flow diagram for my project is given below.

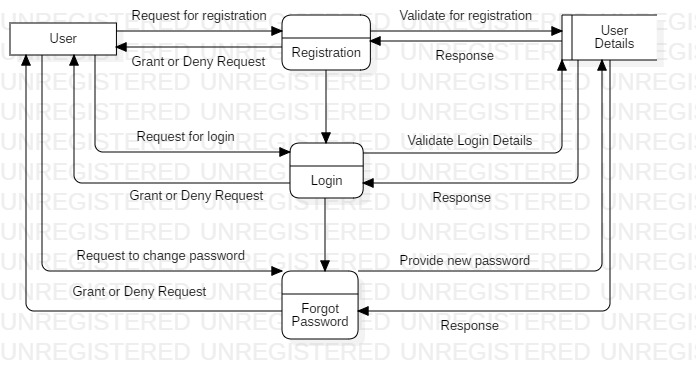


Figure Registration and Login DFD

In the **DFD** above users need to register at first to create an account associated to the system by providing user details. Then they can gain access to the system by providing required and registered details. If in any case the user forgets their login password, the system will guide them to create a new password after validating your ownership of account.

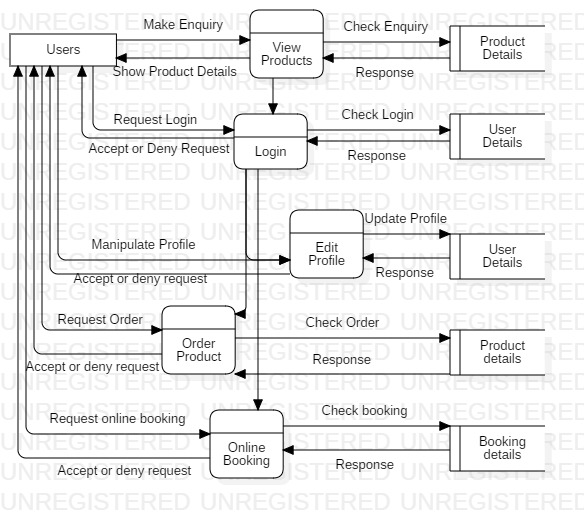


Figure User Functionality DFD

In the **DFD** above, all users will have the ability to view products as well as the registered details associated with it. To use any other functionality though, they need to login to the system providing their login details. They will be able to manipulate their personal details by using the edit profile process, order products after viewing their details as well as make online booking of products by providing booking details. Each process has to accept the request of the user to successfully carry out the function.

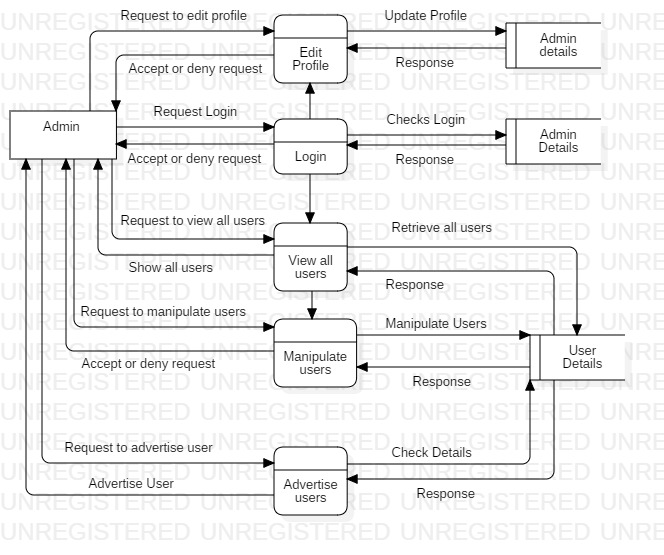


Figure Admin Functionality DFD

In the **DFD** above, admins will be able to login and make changes to their profile just like users. In addition to this, admins will be able to view all registered users of the system including their details and can manipulate the user details. One more functionality of the admin is that they can request the system to advertise certain users and take some part of their details to be shown to every other users.

# Behavior Modelling

The type of modelling that shows how objects interact with each other to produce a particular behavior often specified by a use case is known as behavior model. They represent the overall dynamic system behavior during the execution of different functions. When a certain interaction is made, they show what happens or what should happen.

## Activity Diagram

A type of flowchart that is used to represent the flow of activity from one operation to the other in a dynamic manner is known as an activity diagram. It is crated to show the flow of message from one activity to other rather than one object to other, which other diagrams usually do.

The justification for creating an activity diagram from my project are.

* It will be easier to make the local people as well as every end user understand the work flow concept of the system.
* They can depict multiple conditional scenarios and actors by showing the workflow on every possible scenario.

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Notations Used | Notation Name | Reason |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Sequence Diagrams

The diagram used to represent the interactions of different classes and objects of a system in a time sequence is known as sequence diagrams. It depicts the logical flow of the system in a visual manner.

The justification of using sequence diagram in my project are as follows.

* It creates a common ground to help the developers and business analysts understand the system.
* It helps to visualize what will happen during the execution of each use cases created in the project.

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Notations Used | Notation Name | Reason |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Database Modelling

A database modelling is the process of representation of data in a manner in which they are fundamentally stored, organized as well as manipulated so as to provide the logical structure of a database. One of the popular type of database modelling is a relational database modelling which makes use of a table format for data.

## Data Dictionary

A data dictionary can be defined as a set of information used to describe the structure, contents of the database as well as the format it is made upon. It also represents the relationship of the different elements and helps in the manipulation of the database.

The justification for using a data dictionary for my project are

* It helps to manage all the important details that my project goes through i.e. user details, admin details, product details etc.
* It gives a common meaning to different elements of my project which is easy to understand.
* It helps to analyze the characteristics of my project from a foundation level which helps to locate errors easily and make changes in a comparatively efficient way.

Below are the different elements of my data dictionary.

Admin Registration Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Datatype** | **Length** | **Constraint** | **Null** |
| Admin\_ID | integer | 10 | Primary Key | Not Null |
| First\_Name | varchar | 255 | - | Null |
| Last\_Name | varchar | 255 | - | Null |
| Phone\_Number | integer | 10 | - | Null |
| E-mail | varchar | 255 | - | Null |
| Location | Varchar | 255 | - | Null |
| Date\_of\_Birth | Date |  | - | Null |
| Password | varchar | 255 | - | Null |

User Registration Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Datatype** | **Length** | **Constraint** | **Null** |
| User\_ID | integer | 10 | Primary Key | Not Null |
| First\_Name | varchar | 255 | - | Null |
| Last\_Name | varchar | 255 | - | Null |
| Phone\_Number | integer | 10 | - | Null |
| E-mail | varchar | 255 | - | Null |
| Location | varchar | 255 | - | Null |
| Date\_of\_Birth | Date |  | - | Null |
| Password | varchar | 255 | - | Null |

Product Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Datatype** | **Length** | **Constraint** | **Null** |
| Product\_ID | Integer | 10 | Primary Key | Not Null |
| Product\_Name | Varchar | 255 | - | Null |
| Product\_Quantity | Integer | 10 | - | Null |
| Product\_Price | Integer | 10 |  | Null |
| Manufacture\_Location | Varchar | 255 | - | Null |
| Manufacture Date | Date |  | - | Null |

Online\_Booking Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Datatype** | **Length** | **Constraint** | **Null** |
| User\_ID | integer | 10 | Primary Key | Not Null |
| First\_Name | varchar | 255 | - | Null |
| Last\_Name | varchar | 255 | - | Null |
| Phone\_Number | integer | 10 | - | Null |
| E-mail | varchar | 255 | - | Null |
| Location | varchar | 255 | - | Null |
| Product\_ID | integer | 10 |  | Not Null |
| Product\_Name | Varchar | 255 | - | Null |
| Product\_Quantity | integer | 10 | - | Null |
| Date\_of\_Delivery | Date | - | - | Null |

Community Forum Questions Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Datatype** | **Length** | **Constraint** | **Null** |
| Question\_ID | integer | 10 | Primary Key | Not Null |
| Question | varchar | 255 | - | Null |
| User\_ID | integer | 10 | Foreign Key | Not Null |

Community Forum Answers Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Datatype** | **Length** | **Constraint** | **Null** |
| Answer\_ID | integer | 10 | Primary Key | Not Null |
| Answer | varchar | 255 | - | Null |
| Admin\_ID | integer | 10 | Foreign Key | Not Null |
| User\_ID | integer | 10 | Foreign Key | Not Null |

Product Feedback Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Datatype** | **Length** | **Constraint** | **Null** |
| Feedback\_id | integer | 20 | Primary Key | Not Null |
| Product\_id | integer | 20 | Foreign Key | Not Null |
| Description | Varchar | 255 | - | Null |
| User\_id | integer | 10 | Foreign key | Not Null |
| Admin\_id | integer | 10 | Foreign key | Not Null |

## Entity Relationship Diagram

An entity relationship diagram (**ER Diagram**) is a diagrammatic or graphical representation of different entities and their relationship between each other which is used for organizing data in a database. Each entities holds attributes and their individual data types and are related by different types of relationship like one to one, one to many and many to many.

The justification for using the entity relationship diagram in my project are.

* It acts as a blueprint for the database of my project which helps to create a documentation used to understand the core of my database.
* It helps in the design process to create a format of my database before even creating my database tables.
* It helps to communicate the logical structure of my database and helps the users to organize data.

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Notations Used | Place of use | Reason |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Prototyping

Prototyping is the process of creating a guiding model of a system to test the processes and concept of a system based on which a system is built. Prototyping can be of various types but the type that I am using for my project is mockup prototyping. This type of prototype is helpful only for the visual concept of the system and does not include the processes involved.

The mockups prototypes of project IlamTeaGarden is presented below.

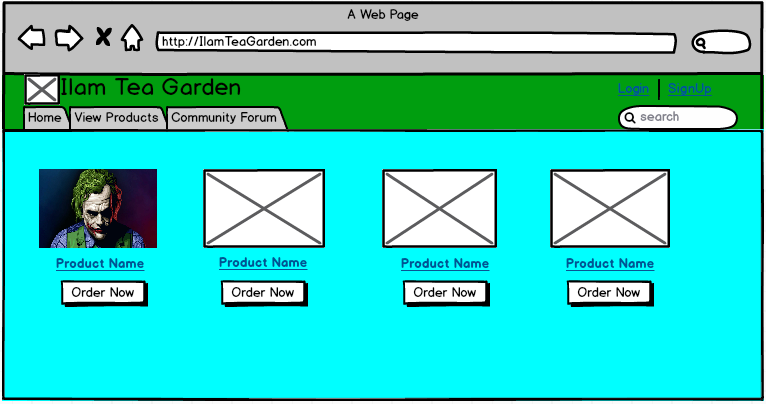


Figure Homepage

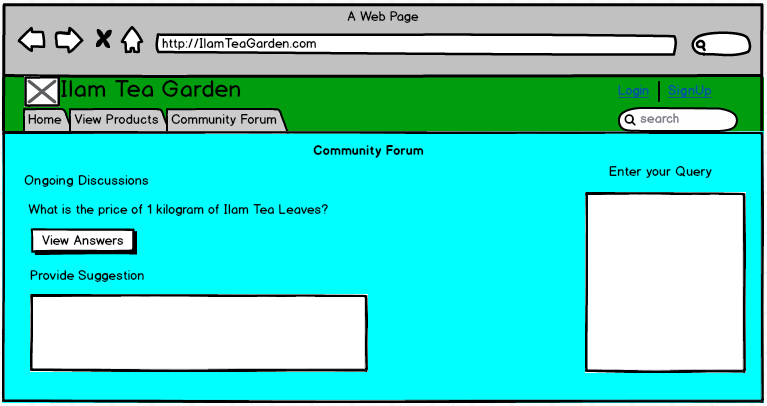


Figure Community Forum

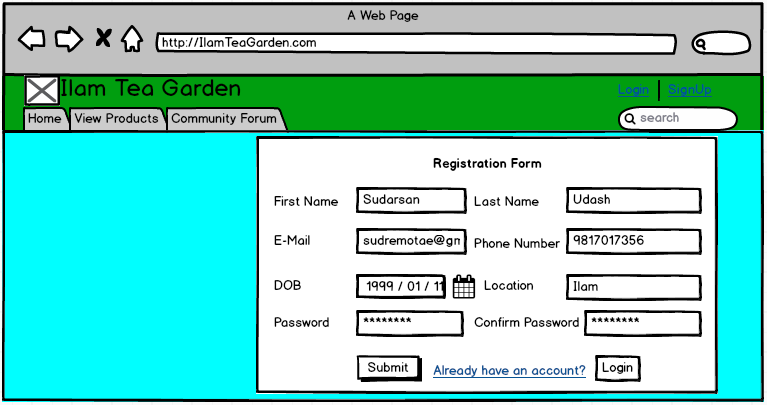


Figure Registration

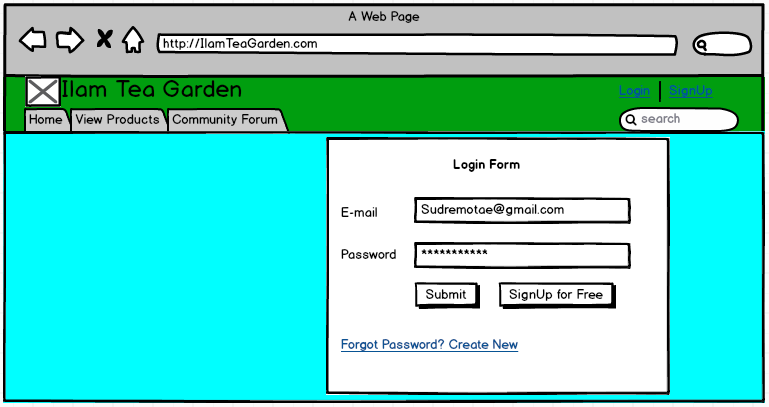


Figure Login

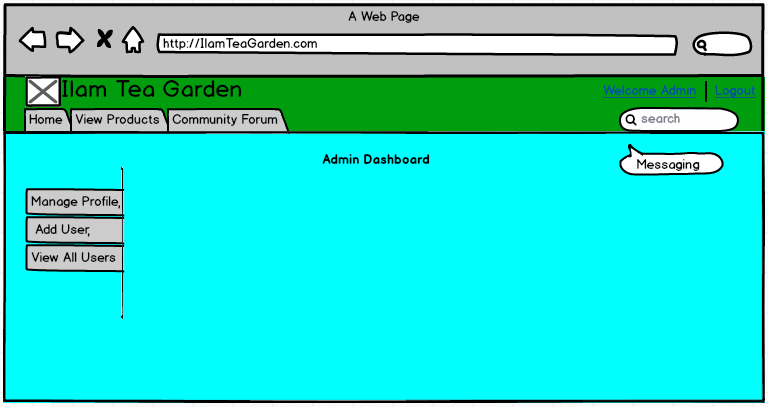


Figure Admin Dashboard

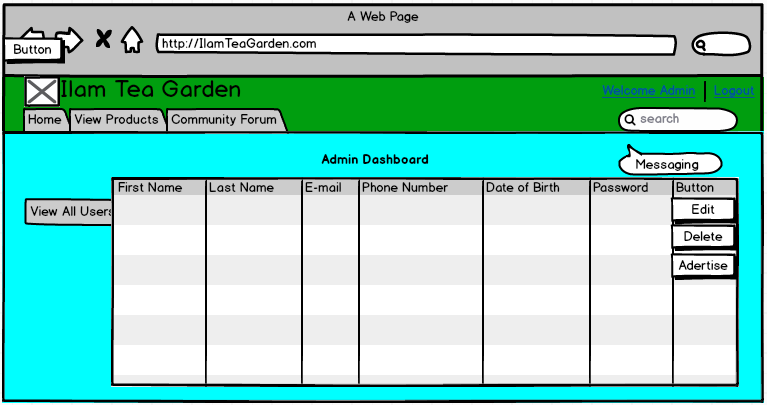


Figure Admin view all users

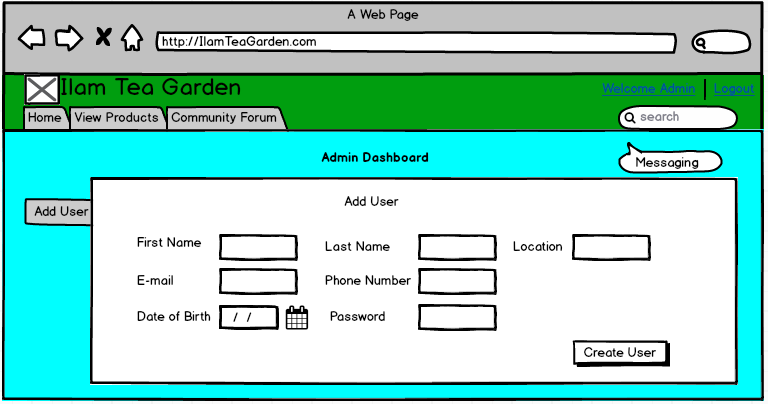


Figure Admin add users

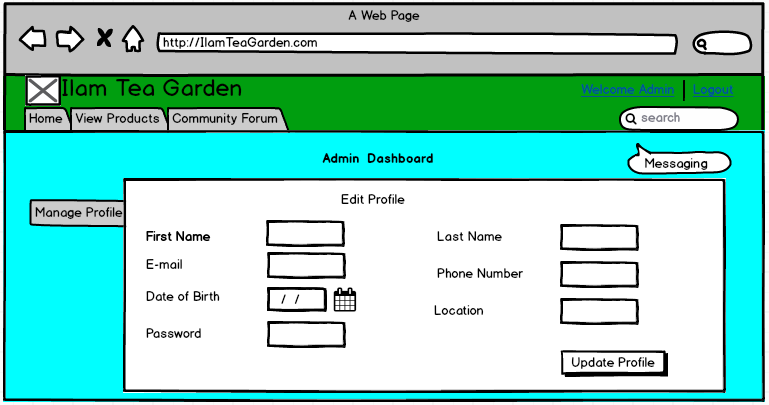


Figure Admin Manage Profile

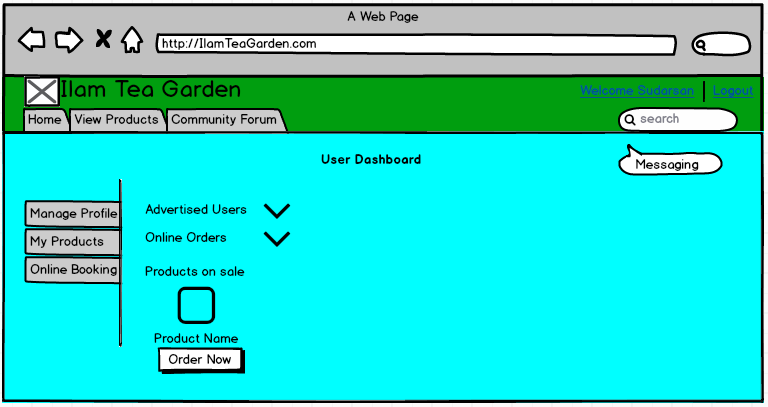


Figure User Dashboard

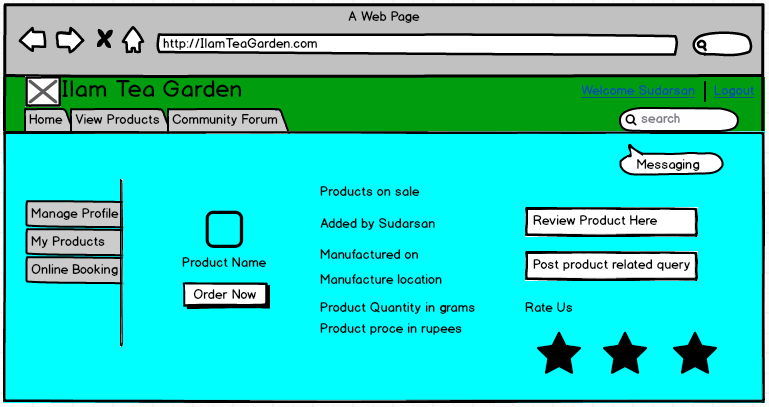


Figure Product page

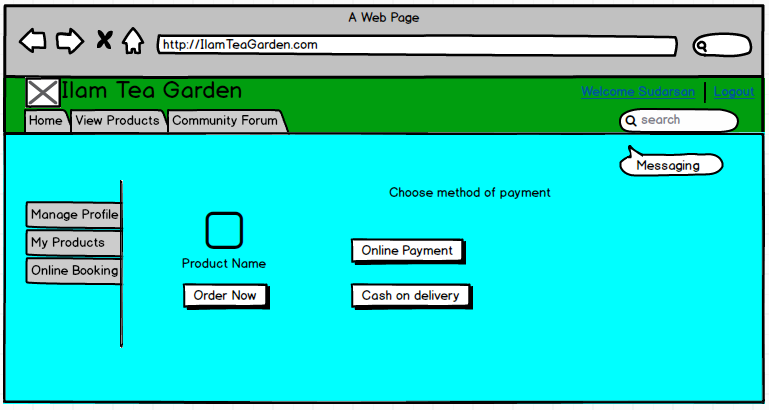


Figure Payment type

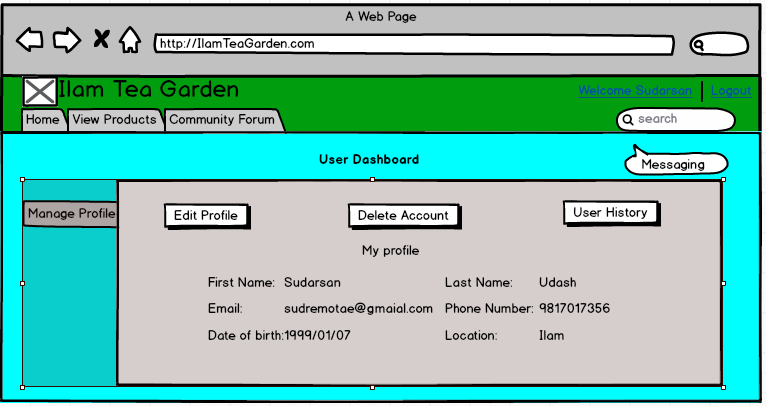


Figure User Manage Profile

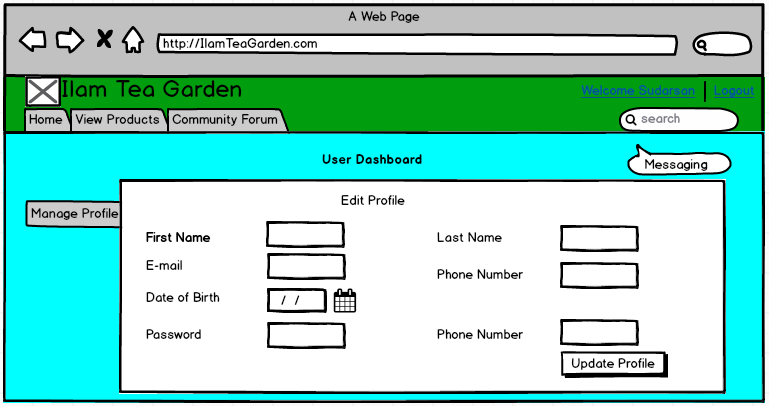


Figure User edit profile

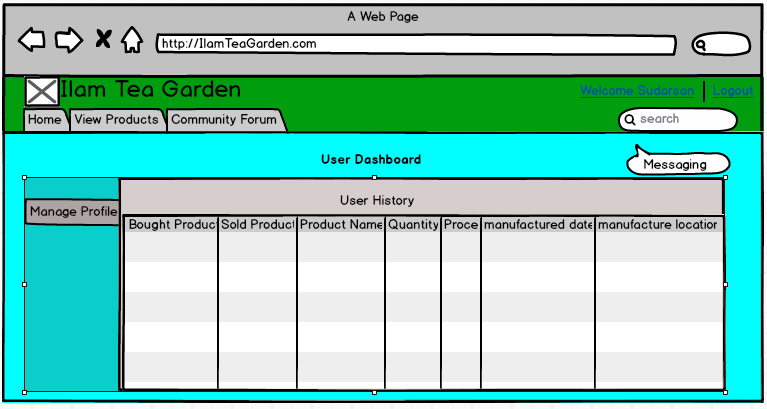


Figure User History

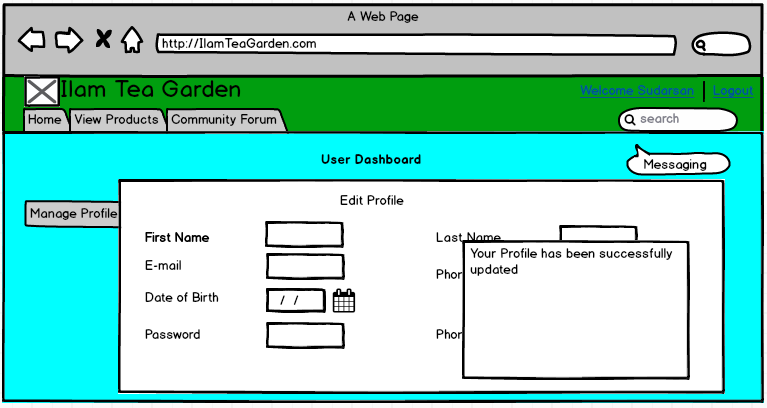


Figure Update successful

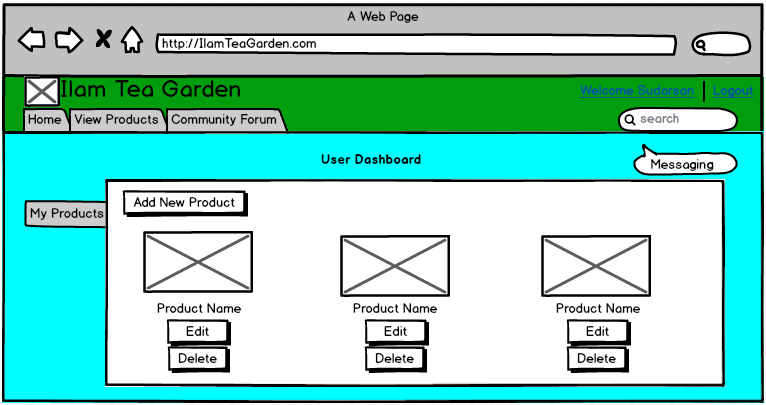


Figure My products

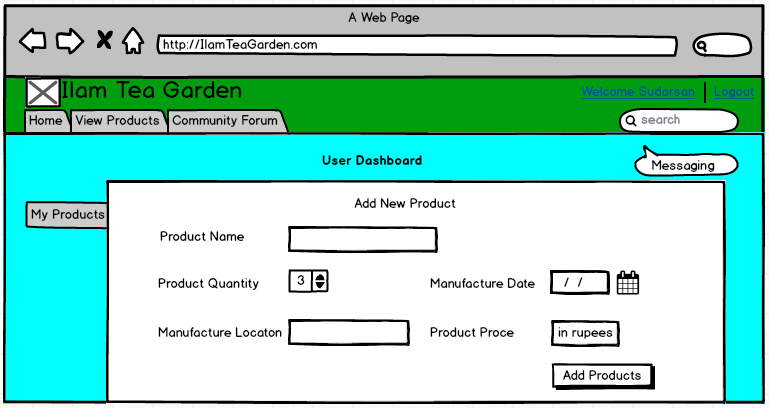


Figure Add new product

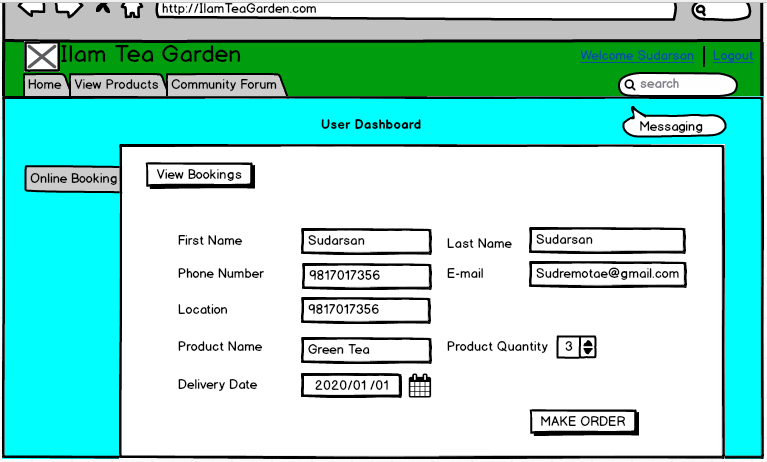


Figure Product Booking