**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW OF THE PROJECT**

This IWONTMISS is an online shopping website. In real time there is no communication between farmer and buyer. Try to solve those issue by updating the process done through online. The purpose of this site is to provide easy way of shopping facility to get healthy food products on online for consumers. For vendors it provides an easy way to sell the Food products using online. In order to give guaranteed products the system named “IWONTMISS” was introduced. It directly connects the buyer with the seller. The buyer is able to view and order the products from the nearby seller through logging in and if the buyer is new to the site they need to sign up. The seller first need to complete their signup process by contacting the Admin, after getting the login details the seller is able to update the product quantity for sales. In such way it helps the buyers to save time and money along with quality products. IWONTMISS is place the buyers can get all the details of nearby sellers who sell fresh and healthy products like fruits and vegetables. IWONTMISS is easiest way to find nearby seller’s wherever you may be. It allows the buyer to find seller using their current location, it also allows the buyers to search sellers within state, city or zip code.to achieve the above features ionic framework is used to develop the front end and firebase as server.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

In existing system sellers and buyers are used to go the market to sell and buy food products. There is middle person between the farmer and buyer. The farmer produce the food products and handover it to the middle person, the middle person who bring food products to the market, consumer able to get food product only from the market. There is no opportunity to get different kinds of food products from a single place.

**2.2 PROPOSED SYSTEM**

To solving the issues of the existing system I proposed a system that will allow the buyer and seller to buy and sell food product online. In that proposed system there is no intermediate process between farmer and buyer. Buyer can get a healthy and good product from the farmer itself. We also get different types of product from one place. It also reduce the cost for buyers and encourage the farmers.

**2.3 FEASIBILITY STUDY**

A feasibility study is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us whether a project is worth the investment in some cases, a project may not be doable. There can be many reasons for this, including requiring too many resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn’t profitable.

A well-designed study should offer a historical background of the business or project, such as a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements, and tax obligations. Generally, such studies precede technical development and project implementation.

Areas of Project Feasibility

There are three types of feasibility study they are separate areas that a feasibility study examines, described below.

**2.3.1 TECHNICAL FEASIBILITY**

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves evaluation of the hardware, software, and other technology requirements of the proposed system. This system will build using ionic framework as front-end and firebase as backend so it is technically feasible.

**2.3.2 ECONOMIC FEASIBILITY**

This assessment typically involves a cost benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility. That helps for the decision makers to determine the positive economic benefits to the organization that the proposed project will provide. This system will develop using ionic framework and firebase which require less cost for development so it is economically feasible.

**2.3.3 OPERATIONAL FEASIBILITY**

This assessment involves undertaking a study to analyze and determine whether and how well the organization’s needs can be met by completing the project. Operational feasibility studies also analyze how a project plan satisfies the requirements identified in the requirements analysis phase of system development. This system will develop using ionic framework and firebase. The process of installation, development and maintenance are easy. This system is user-friendly so it is operatively feasible.

**2.4 SYSTEM SPECIFICATION**

**2.4.1 HARDWARE REQUIREMENTS**

Processor Quad Core and above

Ram 2GB

Memory Space 50GB

**2.4.2 SOFTWARE REQUIREMENTS**

Operating System Windows, Android and IOS

Framework Ionic

Editor Visual Studio Code, devtools

Server Firebase

**2.5 SOFTWARE DESCRIPTION**

Technology used

Ionic

Firebase

**2.5.1 IONIC**

Ionic is a complete [open-source](https://en.wikipedia.org/wiki/Open-source) [SDK](https://en.wikipedia.org/wiki/Software_development_kit) for hybrid [mobile app](https://en.wikipedia.org/wiki/Mobile_app) development.it is built on top of AngularJs and [Apache Cordova](https://en.wikipedia.org/wiki/Apache_Cordova). The more recent releases, known as Ionic 3 or simply "Ionic", are built on [Angular](https://en.wikipedia.org/wiki/Angular_(application_platform)). Ionic provides tools and services for developing hybrid mobile apps using Web technologies like [CSS](https://en.wikipedia.org/wiki/CSS), [HTML5](https://en.wikipedia.org/wiki/HTML5), and [Sass](https://en.wikipedia.org/wiki/Sass_(stylesheet_language)). Apps can be built with these Web technologies and then distributed through native [app stores](https://en.wikipedia.org/wiki/App_store) to be installed on devices by leveraging Cordova.

Features of Ionic

Ionic provides all the functionality which can be found in native mobile development SDKs. Users can build their apps, customize them for [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) or [iOS](https://en.wikipedia.org/wiki/IOS" \o "IOS), and deploy through [Cordova](https://en.wikipedia.org/wiki/Apache_Cordova). Ionic includes mobile components, typography, interactive paradigms, and an extensible base theme.

Using Angular, Ionic provides custom components and methods for interacting with them. One such component, collection repeat, allows users to scroll through a list of thousands of items without any performance hits.

Besides the SDK, Ionic also provides services that developers can use to enable features, such as [push notifications](https://en.wikipedia.org/wiki/Push_notifications), [A/B testing](https://en.wikipedia.org/wiki/A/B_testing), analytics, [code deploys](https://en.wikipedia.org/wiki/Software_deployment), and [automated builds](https://en.wikipedia.org/wiki/Build_automation).

Ionic also provides a powerful [command-line interface (CLI)](https://en.wikipedia.org/wiki/Command-line_interface), so developers can get started with creating a project with a simple command. The CLI also allows developers to add Cordova plugins and additional front-end packages, enable push notifications, generate app Icons and [Splash screens](https://en.wikipedia.org/wiki/Splash_screens), and build native binaries.

Supported Platforms

Ionic is focused on building for modern Web standards and for modern mobile devices. For Android, Ionic supports Android 4.1 and up. For iOS, Ionic supports iOS 7 and up. Ionic supports the Universal Windows Platform for building [Windows 10](https://en.wikipedia.org/wiki/Windows_10) apps.Ionic Framework, powered by Angular.js also supports [BlackBerry 10](https://en.wikipedia.org/wiki/BlackBerry_10) apps.

Performance

Compared to hybrid applications, mixing Ionic code with native mobile app code in Apache Cordova allows for higher performance of the resulting product.

Utilizing [AngularJS](https://en.wikipedia.org/wiki/AngularJS" \o "AngularJS) (rather than [jQuery](https://en.wikipedia.org/wiki/JQuery)) allows Ionic to rely on native hardware acceleration (rather than extensive [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) manipulation). Ionic leverages CSS transitions and transforms for animation as a way to leverage the [GPU](https://en.wikipedia.org/wiki/Graphical_processing_unit) and maximize

available processor time.

Web technologies

The methods by which computers communicate with each other through the use of markup languages and multimedia packages is known as web technology. Ionic supports following web technologies:

HTML

CSS/SaSS

Angularjs

HTML

Hypertext Markup Language (HTML) is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) for creating [web pages](https://en.wikipedia.org/wiki/Web_page) and [web applications](https://en.wikipedia.org/wiki/Web_application). With [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), it forms a triad of cornerstone technologies for the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).

[Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML documents from a [web server](https://en.wikipedia.org/wiki/Web_server) or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page [semantically](https://en.wikipedia.org/wiki/Semantic_Web) and originally included cues for the appearance of the document.

CSS

Cascading Style Sheets (CSS) is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language) like [HTML](https://en.wikipedia.org/wiki/HTML).CSS is a cornerstone technology of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), alongside HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript).

CSS is designed to enable the separation of presentation and content, including [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and fonts. This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility), provide more flexibility and control in the specification of presentation characteristics, enable multiple [web pages](https://en.wikipedia.org/wiki/Web_page) to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or [screen reader](https://en.wikipedia.org/wiki/Screen_reader)), and on [Braille-based](https://en.wikipedia.org/wiki/Braille_display) tactile devices. CSS also has rules for alternate formatting if the content is accessed on a [mobile device](https://en.wikipedia.org/wiki/Mobile_device).

Sass

Sass(Syntactically awesome style sheets) is a [preprocessor](https://en.wikipedia.org/wiki/Preprocessor) [scripting language](https://en.wikipedia.org/wiki/Scripting_language) that is [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) or [compiled](https://en.wikipedia.org/wiki/Compiled_language) into [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS). SassScript is a simple scripting language used in Sass files.

JavaScript

JavaScript is a [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [programming language](https://en.wikipedia.org/wiki/Programming_language). It is a language which is also characterized as [dynamic](https://en.wikipedia.org/wiki/Dynamic_programming_language), [weakly typed](https://en.wikipedia.org/wiki/Weak_typing), [prototype-based](https://en.wikipedia.org/wiki/Prototype-based_programming) and [multi-paradigm](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language).

Alongside [HTML](https://en.wikipedia.org/wiki/HTML) and [CSS](https://en.wikipedia.org/wiki/CSS), JavaScript is one of the three core technologies of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web). JavaScript enables interactive [web pages](https://en.wikipedia.org/wiki/Web_page) and thus is an essential part of [web applications](https://en.wikipedia.org/wiki/Web_application). The vaSst majority of [websites](https://en.wikipedia.org/wiki/Website) use it, and all major [web browsers](https://en.wikipedia.org/wiki/Web_browser) have a dedicated [JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine) to execute it.

As a multi-paradigm language, JavaScript supports [event-driven](https://en.wikipedia.org/wiki/Event-driven_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming), and [imperative](https://en.wikipedia.org/wiki/Imperative_programming) (including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) and [prototype-based](https://en.wikipedia.org/wiki/Prototype-based_programming)) [programming styles](https://en.wikipedia.org/wiki/Programming_paradigm). It has an [API](https://en.wikipedia.org/wiki/Application_programming_interface) for working with text, [arrays](https://en.wikipedia.org/wiki/Array_data_type), dates, [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), and basic manipulation of the [DOM](https://en.wikipedia.org/wiki/Document_Object_Model), but the language itself does not include any [I/O](https://en.wikipedia.org/wiki/Input/output), such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

AngularJS

AngularJS is a JavaScript-based [open-source](https://en.wikipedia.org/wiki/Open-source_software) front-end [web application framework](https://en.wikipedia.org/wiki/Web_application_framework) mainly maintained by [Google](https://en.wikipedia.org/wiki/Google) and by a community of individuals and corporations to address many of the challenges encountered in developing [single-page applications](https://en.wikipedia.org/wiki/Single-page_application). The JavaScript components complement [Apache Cordova](https://en.wikipedia.org/wiki/Apache_Cordova), a framework used for developing cross-platform mobile apps. It aims to simplify both the development and the [testing](https://en.wikipedia.org/wiki/Software_testing) of such applications by providing a framework for client-side [model–view–controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC) and [model–view–viewmodel](https://en.wikipedia.org/wiki/Model_View_ViewModel) (MVVM) architectures, along with components commonly used in [rich Internet applications](https://en.wikipedia.org/wiki/Rich_Internet_application).

The AngularJS framework works by first reading the [HTML](https://en.wikipedia.org/wiki/HTML) page, which has additional custom [tag attributes](https://en.wikipedia.org/wiki/HTML_attribute) embedded into it. Angular interprets those attributes as [directives](https://en.wikipedia.org/wiki/Directive_(programming)) to bind input or output parts of the page to a model that is represented by standard [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [variables](https://en.wikipedia.org/wiki/Variable_(computer_science)). The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic [JSON](https://en.wikipedia.org/wiki/JSON) resources.

JSON

Java Script Object Notation is an [open-standard](https://en.wikipedia.org/wiki/Open_standard) [file format](https://en.wikipedia.org/wiki/File_format) that uses readable text to transmit data objects consisting of [attribute–value pairs](https://en.wikipedia.org/wiki/Attribute%E2%80%93value_pair) and [array data types](https://en.wikipedia.org/wiki/Array_data_type) (or any other [serializable](https://en.wikipedia.org/wiki/Serialization" \o "Serialization) value). It is a very common [data](https://en.wikipedia.org/wiki/Data) format used for [asynchronous](https://en.wikipedia.org/wiki/Asynchronous_I/O) browser–server communication, including as a replacement for [XML](https://en.wikipedia.org/wiki/XML) in some [AJAX](https://en.wikipedia.org/wiki/Ajax_(programming))-style systems.

**2.5.2 FIREBASE**

Real-time Database

Firebase provides a real-time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The company provides client libraries that enable integration with [Android](https://en.wikipedia.org/wiki/Android_(operating_system)), [iOS](https://en.wikipedia.org/wiki/IOS), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Objective-C](https://en.wikipedia.org/wiki/Objective-C), [swift](https://en.wikipedia.org/wiki/Swift_(programming_language)) and [Node.js](https://en.wikipedia.org/wiki/Node.js) applications. The database is also accessible through a REST API and bindings for several [JavaScript frameworks](https://en.wikipedia.org/wiki/JavaScript_frameworks) such as [AngularJS](https://en.wikipedia.org/wiki/AngularJS" \o "AngularJS), [React](https://en.wikipedia.org/wiki/React_(JavaScript_library)), [Ember.js](https://en.wikipedia.org/wiki/Ember.js) and [Backbone.js](https://en.wikipedia.org/wiki/Backbone.js). The REST API uses the [Server-Sent Events](https://en.wikipedia.org/wiki/Server-sent_events) protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the real-time database can secure their data by using the company's server-side-enforced security rules.

Firebase Authentication

Firebase Authentication is a service that can authenticate users using only client-side code. It supports [social login providers](https://en.wikipedia.org/wiki/Social_login) Facebook, GitHub, Twitter and Google (and [Google Play Games](https://en.wikipedia.org/wiki/Google_Play_Games)). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.

Firebase Storage

Firebase Storage provides secure file uploads and downloads for Firebase apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firebase Storage is backed by Google Cloud Storage.

**CHAPTER 3**

**SYSTEM DESIGN AND DEVELOPMENT**

**3.1 MODULE DESCRIPTION**

This system is proposed based on 3 necessary modules that are follows:

Buyer Module

Seller Module

Admin Module

**3.1.1 BUYER MODULE**

Buyer Module is the place where users can Search and order their needed food items. And the buyers also able to pay the amount when they place the order after placed the order they get the details of the seller. For that they need to sign up using their credentials like name, mobile number, and email id. All the verification is done using their mail id.

**3.1.2 SELLER MODULE**

Seller Module allows the sellers to update the quantity of food items and track the order summary. Seller is also get notification about their orders and the buyer details. For that they need to get the login from the admin.

**3.1.3 ADMIN MODULE**

Admin Module allows the admin to create seller login and monitor the process of buyer and seller.

**3.2 UML DIAGRAMS**

**3.2.1 DATA FLOW DIAGRAM**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. A **data-flow diagram (DFD)** is a **graphical representation** of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (**structured design**).On a **DFD**, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on Fig 3.2.1).

DFDs help system designers and others during initial analysis stages visualize a current system or one that may be necessary to meet new requirements. Systems analysts prefer working with DFDs, particularly when they require a clear understanding of the boundary between existing systems and postulated systems.

DFDs represent the following

1. External devices sending and receiving data

2. Processes that change that data

3. Data flows themselves

4. Data storage locations

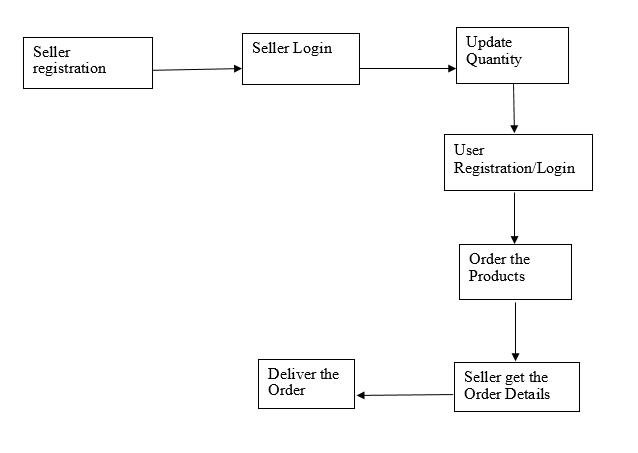


Fig 3.2.1 Data Flow Diagram

**3.2.2 USE CASE DIAGRAM**

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous.

A use case (or set of use cases) has these characteristics

* Organizes functional requirements
* Models the goals of system/actor (user) interactions
* Records paths (called *scenarios*) from trigger events to goals
* Describes one main flow of events (also called a basic course of action), and possibly other ones, called *exceptional* flows of events (also called alternate courses of action)
* Is multi-level, so that one use case can use the functionality of another one.

Use cases can be employed during several stages of software development, such as planning system requirements, validating design, testing software, and creating an outline for online help and user manuals Fig 3.2.2 illustrates the usecase diagram for the system.

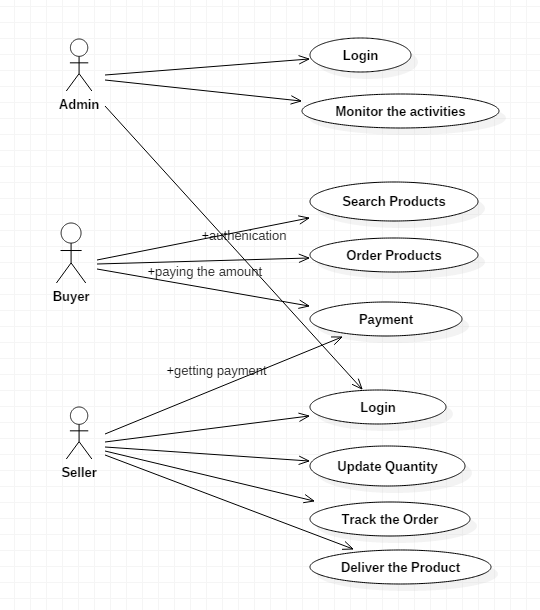


Fig 3.2.2 Usecase Diagram

**3.2.3 CLASS DIAGRAM**

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The concept is several years old but has been refined as OOP Modeling paradigms have evolved.

In a class diagram, the classes are arranged in groups that share common characteristics. A class diagram resembles a flowchart in which classes are portrayed as boxes, each box having three rectangles inside. The top rectangle contains the name of the class; the middle rectangle contains the attributes of the class; the lower rectangle contains the methods, also called operations, of the class. Lines, which may have arrows at one or both ends, connect the boxes. These lines define the relationships, also called associations, between the classes.

Class Diagram provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. It provides a wide variety of usages; from Modeling the domain-specific data structure to detailed design of the target system. With the share model facilities, you can reuse your class model in the interaction diagram for Modeling the detailed design of the dynamic behaviour. Fig 3.2.3 illustrates the classes of the IWONTMISS system with its attributes and methods.

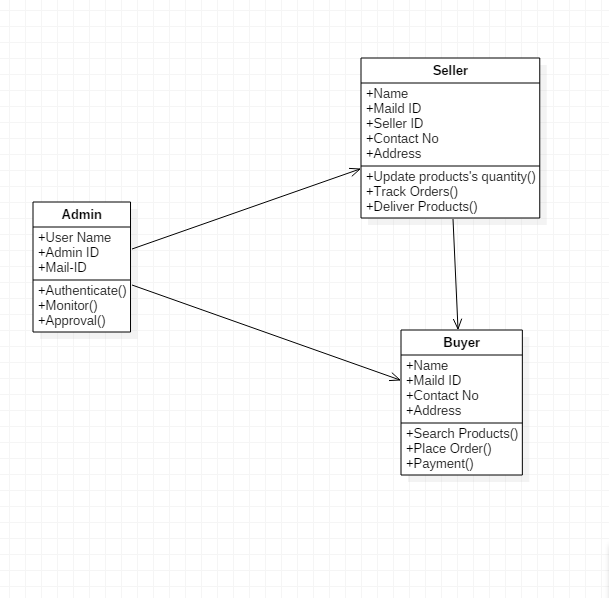
****

Fig 3.2.3 Class Diagram

**3.2.4 SEQUENCE DIAGRAM**

A sequence diagram, in the context of UML, represents object collaboration and is used to define event sequences between objects for a certain outcome. A sequence diagram is an essential component used in processes related to analysis, design and documentation. A sequence diagram is also known as a timing diagram, event diagram and event scenario.

Object interactions usually begin at the top of a diagram and end at the bottom. In a sequence diagram, object interaction occurs through messages on the vertical and horizontal dimensions and are designated by horizontal arrows and message names. The initial sequence diagram message begins at the top and is located on the diagram's left side. Subsequent messages are added just below previous messages. Sequence diagram messages may be subdivided by type, based on functionality.

A lifeline, which indicates a role, is represented by a named rectangular box with a dashed line descending from the centre of the diagram's bottom edge. Lifeline boxes represent participating sequence object instances. Blank instance names represent anonymous instances. The Sequence Diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. Fig 3.2.4 sequence diagram is from the flow of events which you have defined in the use case description.

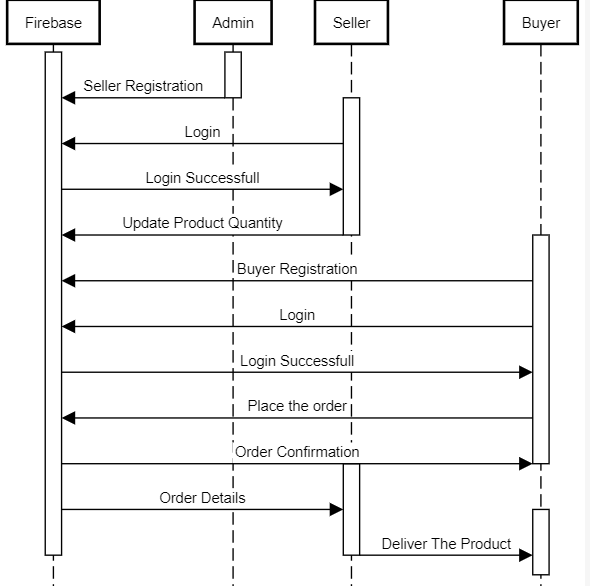
****

Fig 3.2.4 Sequence Diagram

**3.2.5 ER DIAGRAM**

An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system’s entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The three main components of an ERD are:

* The *entity* is a person, object, place or event for which data is collected. For example, if you consider the information system for a business, entities would include not only customers, but the customer's address, and orders as well. The entity is represented by a rectangle and labelled with a singular noun.
* The *relationship* is the interaction between the entities. In the example above, the customer *places* an order, so the word "places" defines the relationship between that instance of a customer and the order or orders that they place. A relationship may be represented by a diamond shape, or more simply, by the line connecting the entities. In either case, verbs are used to label the relationships.
* The *cardinality* defines the relationship between the entities in terms of numbers. An entity may be *optional*: for example, a sales rep could have no customers or could have one or many customers; or *mandatory*: for example, there must be at least one product listed in an order. There are several different types of cardinality notation; *crow's foot notation*, used here, is a common one. In crow's foot notation, a single bar indicates *one*, a double bar indicates *one and only one*, a *circle* indicates zero, and a *crow's foot* indicates many.

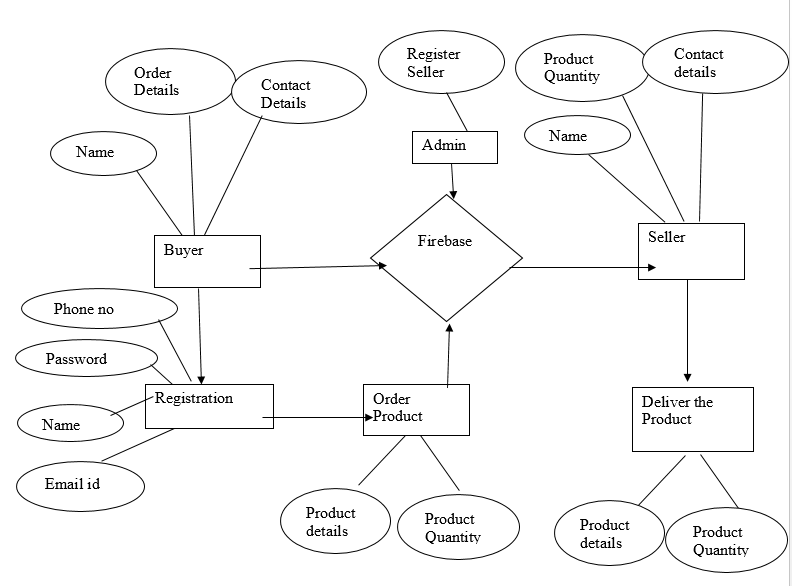


Fig 3.2.5 ER Diagram

**3.3 DATABASE DESIGN**

Database is an integrated collection of data and provides a centralized access to the data and makes possible to treat data as a separate resource. Usually centralized data managing software is called a Relational Database Management System (RDBMS). The most significant different between RDBMS and other type of Data Management is the separation of data as seen by the program and data as store of on the direct access storage device. This is the difference between logical and physical data.

Data Dictionary

The efficiency of an application developed using RDBMS mainly depend upon the database tables, the fields in each table and the way the tables are opened using the contents in them to retrieve the necessary information. Hence a careful selection of tables and their fields are imperative.

The database tables used in this system are created keeping the above points in mind. The tables used are given below.

Table 3.3.1 Buyer Table

|  |  |  |  |
| --- | --- | --- | --- |
| **SERIAL NO** | **FIELD NAME** | **DATATYPE** | **DESCRIPTION** |
| 1 | Bname | Varchar(50) | Buyer Name |
| 2 | Baddress | varchar(50) | Buyer Adress |
| 3 | Bnumber | int(10) | Buyer Mobile Number |
| 4 | Bemail | varchar(50) | Buyer Email address |

Table 3.3.2 Seller Table

|  |  |  |  |
| --- | --- | --- | --- |
| **SERIAL NO** | **FIELD NAME** | **DATATYPE** | **DESCRIPTION** |
| 1 | Sname | Varchar(50) | Seller Name |
| 2 | Saddress | varchar(50) | Seller Adress |
| 3 | Snumber | int(10) | Seller Mobile Number |
| 4 | Semail | varchar(50) | Seller Email address |
| 5 | Sproduct | Varchar(30) | Product details which is the seller wants to sell |

Table 3.3.3 Product Table

|  |  |  |  |
| --- | --- | --- | --- |
| **SERIAL NO** | **FIELD NAME** | **DATATYPE** | **DESCRIPTION** |
| 1 | Pid | Int(50) | Product Id |
| 2 | Pname | Varchar(50) | Product Name |
| 3 | Pprice | int(20) | Price of the Product |
| 4 | Pqunt | int(10) | Product Quantity |

**3.4 OUTPUT DESIGN**

The term output necessarily implies to information printed or displayed by an information system. Following are the activities that are carried out in the output design stage:

* Identification of the specific outputs required to meet the information requirements.
* Selection of methods required for presenting information.
* Designing of reports, formats or other documents that acts as carrier of information.

**Output design objectives**

The output design of an information system must meet the following objectives:

1.)    The output design should provide information about the past, present or future events. The operational control level outputs provide information of the past and present events. On the other hand, required at the strategic planning level provide information of the future events.

2.)    The output design should indicate the important events, opportunities and problems.

3.)    The output design should be designed keeping in mind that an action must be triggered in response to some event. A set of rules is pre-defined for such trigger.

4.)    The output design should produce some action to the transaction. For example, when the telephone bill is received, a receipt is printed.

**Presentation of output**

The next consideration in the output design is the presentation involved with an information system. The presentation of an output is regarded as an important feature of output design. The presentation of an output represented either in tabular or graphical form or in both forms. A tabular format is preferred in the following conditions:

* When the details dominate the content of the output
* When the contents of the output are classified in groups.
* When the output design are to be compared.

A tabular format is also preferred for the detailed reports. Graphical representation is used to improve the effectiveness of the output because some users prefer to view information in graphic form rather than in rows and columns of the tables. The tabular and graphical formats may be combined together to enhance the presentation of the output.

**3.5 INPUT DESIGN**

Similar to the output design, input design is equally important for a system designer. This is because output from a system is regarded as the foremost determinant for defining the performance of a system. The output of the system greatly affects the input design of the system.

**Input Design Objectives**

The input design of an information system must the following objectives:

* The input design of the system must attempt and try reducing the data requirements. It should also avoid capturing unnecessary data such as constant and system-computable data.
* The input design must avoid processing delays during data entry. Capturing automatic data can reduce this kind of delay.
* The input design must avoid data entry errors. This can be achieved by checking the errors in data entry program. This technique of checking data entry program programs for errors is known as input validation technique.
* The input design must keep the process simple and easy to use.

**Input layout**

The layout of the input design must contain the following items.

1. Headings and date of data entry.
2. Data heading and value
3. Data type and width of the column
4. Initials of data entry operator

**CHAPTER 4**

**TESTING**

Testing is the stage before system implementation where the system is made error free and all the needed modifications are made. The system was tested with test data and necessary corrections to the system were carried out. All the reports were checked by the user and approved. The system was very user friendly with online help to assist the user wherever necessary.

**4.1 TEST PLAN**

A test plan is a general document for the entire project, which defines the scope, approach to be taken, and schedule of testing, as well as identifying the test item for the entire testing process, and the personal responsible for the different activities of testing. This document describes the plan for testing, the knowledge management tool.

Major testing activities are

Test units

Features to be tested

Approach for testing

Test deliverables

Schedule

Personal allocation

**4.2 TESTING STRATEGIES**

Some of other testing strategies are follows:

Alpha Testing

Beta Testing

Unit Testing

System Testing

**4.2.1 ALPHA TESTING**

This was done at the developer’s site by a customer. The software is used in a natural setting with the developer “looking over the shoulder” of the user and recording errors and usage problems. Alpha tests are conducted in a controlled environment.

**4.2.2 BETA TESTING**

This was conducted at one or more customer sites by the end-user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta tests, software engineers make modifications and then prepare for release of the software product to the entire customer base.

**4.2.3 UNIT TESTING**

Unit testing is a [software testing](https://en.wikipedia.org/wiki/Software_testing) method by which individual units of [source code](https://en.wikipedia.org/wiki/Source_code), sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

**4.2.4 SYSTEM TESTING**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified [requirements](https://en.wikipedia.org/wiki/Requirements).

Test Deliverables

The following documents are required besides the test plan

Unit test report for each unit

Test case specification for system testing

The report for system testing

Error report

The test case specification for system testing has to be submitted for review before the system testing commences.

**4.3 TEST CASES**

A test case is a set of conditions or variables and inputs that are developed for a particular goal or objective to be achieved on a certain application to judge its capabilities or features. It might take more than one test case to determine the true functionality of the application being tested. Every requirement or objective to be achieved needs at least one test case.

**TEST CASE RESULTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **TESTCASE** | **EXPECTED RESULT** | **ACTUAL RESULT** |
| 1. | Signup With Invalid Username or Password of Gmail in buyer Signup Page | Error message should be display. | Error message is display. |
| 2. | Login with valid username and password in both Seller and Buyer Login Page | Error message should be display. | Error message is display. |
| 3. | Login with valid username and password in both Seller and Buyer Login Page | Menu Page should be open. | Menu page is open. |
| 4. | Register with valid details in Buyer Page | Registration done | User dash board open. |
| 5. | Address Field in order page of buyer is empty | Error message should be displayed | Error message is displayed |
| 6. | After placing order | A success message should be displayed | Order Successfully placed message is displayed |

Table 4.3.1 Test Cases Table

**CHAPTER 5**

**IMPLEMENTATION**

 An implementation is a realization of a [technical specification](https://en.wikipedia.org/wiki/Technical_specification) or algorithm as a [program](https://en.wikipedia.org/wiki/Computer_program), [software component](https://en.wikipedia.org/wiki/Software_component), or other [computer system](https://en.wikipedia.org/wiki/Computer_system) through [computer programming](https://en.wikipedia.org/wiki/Computer_programming) and [deployment](https://en.wikipedia.org/wiki/Software_deployment). Many implementations may exist for a given specification or standard. For an implementation process to be successful, many tasks between different departments need to be accomplished in sequence.

**5.1 IMPLEMENTATION PROCEDURE**

Implementation of a computer system to replace a manual system. The problems encountered are converting files, training users, and verifying printouts for integrity.

Implementation of a new computer system to replace an existing one. This is usually a difficult conversion. If not properly planned there can be many problems.

**5.2 IMPLEMENTATION ISSUES**

This phase is primarily concerned with user training, site preparation and file conversions. During the final testing, user acceptance is tested, followed by user training. Depending in the nature of the extensive user training may be required.

After development and testing has been completed, implementation of the information system can begin. During system implementation system can begin. During system implementation, the project team should be brought back to full strength. During software development stage, project teams end to play passive role as the technical steps of program development and testing evolve. However, organizational representation, accomplished through the project team, is required to complete the system development life cycle.

A key difference between system implementation and all other phases of the lifecycle is that all project activities up to this point have been performed in safe, protected and secure environments, where project issues that arise have little or no impact on day-to-day business operations. Once the system goes live, however, this is no longer the case. Any misuses at this point will almost certainly translate into direct operational and/or financial impacts on the performing organization. It is through the careful planning, execution and management of the system implementation activities that the project team can minimize the likelihood of these occurrences, and determine the contingency plans in the event of a problem.

The entire system was developed using the ionic as Front End and firebase as back end. Ionic is used to design the pages with Input Fields and Output Fields. It also used to Perform Validations. The firebase is the back end tool where the database resides.

Hence the design of the entire system is user-friendly and simple the implementation has been quite easy

**CHAPTER 6**

**CONCLUSION AND FUTURE ENHANCEMENT**

**6.1 CONCLUSION**

The IWONTMISS system is developed using Ionic Framework as Front end and firebase as Server fully meets the objectives of the system which it has been developed. The system has reached a steady state where all bugs have been eliminated. The system is operated at a high level of efficiency and all the buyers and sellers associated with the system understands its advantage.

**6.2 FUTURE ENHANCEMENT**

As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment. Sub admin module can be added to control the traffic. Based on the customer needs the products and its seller will be expanded in this system.

**APPENDIX**

**A1 SAMPLE CODE**

angular.module('app.controllers', [])

.controller('loginCtrl',function($scope,$rootScope,$ionicHistory,sharedUtils,$state,$ionicSideMenuDelegate) {

$rootScope.extras = false; // For hiding the side bar and nav icon

$scope.$on('$ionicView.enter', function(ev) {

if(ev.targetScope !== $scope){

$ionicHistory.clearHistory();

$ionicHistory.clearCache();

}

});

//Check if user already logged in

firebase.auth().onAuthStateChanged(function(user) {

if (user) {

$ionicHistory.nextViewOptions({

historyRoot: true

});

$ionicSideMenuDelegate.canDragContent(true); // Sets up the sideMenu dragable

$rootScope.extras = true;

sharedUtils.hideLoading();

$state.go('menu2', {}, {location: "replace"});

}

});

$scope.loginEmail = function(formName,cred) {

if(formName.$valid) { // Check if the form data is valid or not

sharedUtils.showLoading(); firebase.auth().signInWithEmailAndPassword(cred.email,cred.password).then(function(result) {

$ionicHistory.nextViewOptions({

historyRoot: true

});

$rootScope.extras = true;

sharedUtils.hideLoading();

$state.go('menu2', {}, {location: "replace"});

},

function(error) {

sharedUtils.hideLoading();

sharedUtils.showAlert("Please note","Authentication Error");

}

);

}else{

sharedUtils.showAlert("Please note","Entered data is not valid");

}

};

$scope.loginGmail = function(){

//Gmail Login

};

})

.controller('signupCtrl', function($scope,$rootScope,sharedUtils,$ionicSideMenuDelegate,

$state,fireBaseData,$ionicHistory) {

$rootScope.extras = false; // For hiding the side bar and nav icon

$scope.signupEmail = function (formName, cred) {

if (formName.$valid) { // Check if the form data is valid or not

sharedUtils.showLoading();

//Main Firebase Authentication part

firebase.auth().createUserWithEmailAndPassword(cred.email, cred.password).then(function (result) {

//Add name and default dp to the Autherisation table

result.updateProfile({

displayName: cred.name,

photoURL: "default\_dp"

}).then(function() {}, function(error) {});

//Add phone number to the user table

fireBaseData.refUser().child(result.uid).set({

telephone: cred.phone

});

//Registered OK

$ionicHistory.nextViewOptions({

historyRoot: true

});

$ionicSideMenuDelegate.canDragContent(true); // Sets up the sideMenu dragable

$rootScope.extras = true;

sharedUtils.hideLoading();

$state.go('menu2', {}, {location: "replace"});

}, function (error) {

sharedUtils.hideLoading();

sharedUtils.showAlert("Please note","Sign up Error");

});

}else{

sharedUtils.showAlert("Please note","Entered data is not valid");

}

}

})

.controller('menu2Ctrl', function($scope,$rootScope,$ionicSideMenuDelegate,fireBaseData,$state,

$ionicHistory,$firebaseArray,sharedCartService,sharedUtils) {

//Check if user already logged in

firebase.auth().onAuthStateChanged(function(user) {

if (user) {

$scope.user\_info=user; //Saves data to user\_info

}else {

$ionicSideMenuDelegate.toggleLeft(); //To close the side bar

$ionicSideMenuDelegate.canDragContent(false); // To remove the sidemenu white space\

$ionicHistory.nextViewOptions({

historyRoot: true

});

$rootScope.extras = false;

sharedUtils.hideLoading();

$state.go('tabsController.login', {}, {location: "replace"});

}

});

// On Loggin in to menu page, the sideMenu drag state is set to true

$ionicSideMenuDelegate.canDragContent(true);

$rootScope.extras=true;

// When user visits A-> B -> C -> A and clicks back, he will close the app instead of back linking

$scope.$on('$ionicView.enter', function(ev) {

if(ev.targetScope !== $scope){

$ionicHistory.clearHistory();

$ionicHistory.clearCache();

}

});

$scope.loadMenu = function() {

sharedUtils.showLoading();

$scope.menu=$firebaseArray(fireBaseData.refMenu());

sharedUtils.hideLoading();

}

$scope.showProductInfo=function (id) {

};

$scope.addToCart=function(item){

sharedCartService.add(item);

};

})

.controller('offersCtrl', function($scope,$rootScope) {

//We initialise it on all the Main Controllers because, $rootScope.extra has default value false

// So if you happen to refresh the Offer page, you will get $rootScope.extra = false

//We need $ionicSideMenuDelegate.canDragContent(true) only on the menu, ie after login page

$rootScope.extras=true;

})

.controller('indexCtrl', function($scope,$rootScope,sharedUtils,$ionicHistory,$state,$ionicSideMenuDelegate,sharedCartService) {

//Check if user already logged in

firebase.auth().onAuthStateChanged(function(user) {

if (user) {

$scope.user\_info=user; //Saves data to user\_info

//Only when the user is logged in, the cart qty is shown

//Else it will show unwanted console error till we get the user object

$scope.get\_total= function() {

var total\_qty=0;

for (var i = 0; i < sharedCartService.cart\_items.length; i++) {

total\_qty += sharedCartService.cart\_items[i].item\_qty;

}

return total\_qty;

};

}else {

$ionicSideMenuDelegate.toggleLeft(); //To close the side bar

$ionicSideMenuDelegate.canDragContent(false); // To remove the sidemenu white space

$ionicHistory.nextViewOptions({

historyRoot: true

});

$rootScope.extras = false;

sharedUtils.hideLoading();

$state.go('tabsController.login', {}, {location: "replace"});

}

});

$scope.logout=function(){

sharedUtils.showLoading();

// Main Firebase logout

firebase.auth().signOut().then(function() {

$ionicSideMenuDelegate.toggleLeft(); //To close the side bar

$ionicSideMenuDelegate.canDragContent(false); // To remove the sidemenu white space

$ionicHistory.nextViewOptions({

historyRoot: true

});

$rootScope.extras = false;

sharedUtils.hideLoading();

$state.go('tabsController.login', {}, {location: "replace"});

}, function(error) {

sharedUtils.showAlert("Error","Logout Failed")

});

}

})

.controller('myCartCtrl', function($scope,$rootScope,$state,sharedCartService) {

$rootScope.extras=true;

//Check if user already logged in

firebase.auth().onAuthStateChanged(function(user) {

if (user) {

$scope.cart=sharedCartService.cart\_items; // Loads users cart

$scope.get\_qty = function() {

$scope.total\_qty=0;

$scope.total\_amount=0;

for (var i = 0; i < sharedCartService.cart\_items.length; i++) {

$scope.total\_qty += sharedCartService.cart\_items[i].item\_qty;

$scope.total\_amount += (sharedCartService.cart\_items[i].item\_qty \* sharedCartService.cart\_items[i].item\_price);

}

return $scope.total\_qty;

};

}

//We dont need the else part because indexCtrl takes care of it

});

$scope.removeFromCart=function(c\_id){

sharedCartService.drop(c\_id);

};

$scope.inc=function(c\_id){

sharedCartService.increment(c\_id);

};

$scope.dec=function(c\_id){

sharedCartService.decrement(c\_id);

};

$scope.checkout=function(){

$state.go('checkout', {}, {location: "replace"});

};

})

.controller('lastOrdersCtrl', function($scope,$rootScope,fireBaseData,sharedUtils) {

$rootScope.extras = true;

sharedUtils.showLoading();

//Check if user already logged in

firebase.auth().onAuthStateChanged(function (user) {

if (user) {

$scope.user\_info = user;

fireBaseData.refOrder()

.orderByChild('user\_id')

.startAt($scope.user\_info.uid).endAt($scope.user\_info.uid)

.once('value', function (snapshot) {

$scope.orders = snapshot.val();

$scope.$apply();

});

sharedUtils.hideLoading();

}

});

})

.controller('favouriteCtrl', function($scope,$rootScope) {

$rootScope.extras=true;

})

.controller('settingsCtrl', function($scope,$rootScope,fireBaseData,$firebaseObject,

$ionicPopup,$state,$window,$firebaseArray,

sharedUtils) {

//Bugs are most prevailing here

$rootScope.extras=true;

//Shows loading bar

sharedUtils.showLoading();

//Check if user already logged in

firebase.auth().onAuthStateChanged(function(user) {

if (user) {

//Accessing an array of objects using firebaseObject, does not give you the $id , so use firebase array to get $id

$scope.addresses= $firebaseArray(fireBaseData.refUser().child(user.uid).child("address"));

// firebaseObject is good for accessing single objects for eg:- telephone. Don't use it for array of objects

$scope.user\_extras= $firebaseObject(fireBaseData.refUser().child(user.uid));

$scope.user\_info=user; //Saves data to user\_info

//NOTE: $scope.user\_info is not writable ie you can't use it inside ng-model of <input>

//You have to create a local variable for storing emails

$scope.data\_editable={};

$scope.data\_editable.email=$scope.user\_info.email; // For editing store it in local variable

$scope.data\_editable.password="";

$scope.$apply();

sharedUtils.hideLoading();

}

});

$scope.addManipulation = function(edit\_val) { // Takes care of address add and edit ie Address Manipulator

if(edit\_val!=null) {

$scope.data = edit\_val; // For editing address

var title="Edit Address";

var sub\_title="Edit your address";

}

else {

$scope.data = {}; // For adding new address

var title="Add Address";

var sub\_title="Add your new address";

}

// An elaborate, custom popup

var addressPopup = $ionicPopup.show({

template: '<input type="text" placeholder="Nick Name" ng-model="data.nickname"> <br/> ' +

'<input type="text" placeholder="Address" ng-model="data.address"> <br/> ' +

'<input type="number" placeholder="Pincode" ng-model="data.pin"> <br/> ' +

'<input type="number" placeholder="Phone" ng-model="data.phone">',

title: title,

subTitle: sub\_title,

scope: $scope,

buttons: [

{ text: 'Close' },

{

text: '<b>Save</b>',

type: 'button-positive',

onTap: function(e) {

if (!$scope.data.nickname || !$scope.data.address || !$scope.data.pin || !$scope.data.phone ) {

e.preventDefault(); //don't allow the user to close unless he enters full details

} else {

return $scope.data;

}

}

}

]

});

addressPopup.then(function(res) {

if(edit\_val!=null) {

//Update address

if(res!=null){ // res ==null => close fireBaseData.refUser().child($scope.user\_info.uid).child("address").child(edit\_val.$id).update({ // set

nickname: res.nickname,

address: res.address,

pin: res.pin,

phone: res.phone

});

}

}else{

//Add new address

fireBaseData.refUser().child($scope.user\_info.uid).child("address").push({ // set

nickname: res.nickname,

address: res.address,

pin: res.pin,

phone: res.phone

});

}

});

};

// A confirm dialog for deleting address

$scope.deleteAddress = function(del\_id) {

var confirmPopup = $ionicPopup.confirm({

title: 'Delete Address',

template: 'Are you sure you want to delete this address',

buttons: [

{ text: 'No' , type: 'button-stable' },

{ text: 'Yes', type: 'button-assertive' , onTap: function(){return del\_id;} }

]

});

confirmPopup.then(function(res) {

if(res) {

fireBaseData.refUser().child($scope.user\_info.uid).child("address").child(res).remove();

}

});

};

$scope.save= function (extras,editable) {

//1. Edit Telephone doesnt show popup 2. Using extras and editable // Bugs

if(extras.telephone!="" && extras.telephone!=null ){

//Update Telephone

fireBaseData.refUser().child($scope.user\_info.uid).update({ // set

telephone: extras.telephone

});

}

//Edit Password

if(editable.password!="" && editable.password!=null ){

//Update Password in UserAuthentication Table

firebase.auth().currentUser.updatePassword(editable.password).then(function(ok) {}, function(error) {});

sharedUtils.showAlert("Account","Password Updated");

}

//Edit Email

if(editable.email!="" && editable.email!=null && editable.email!=$scope.user\_info.email){

//Update Email/Username in UserAuthentication Table

firebase.auth().currentUser.updateEmail(editable.email).then(function(ok) {

$window.location.reload(true);

//sharedUtils.showAlert("Account","Email Updated");

}, function(error) {

sharedUtils.showAlert("ERROR",error);

});

}

};

$scope.cancel=function(){

// Simple Reload

$window.location.reload(true);

console.log("CANCEL");

}

})

.controller('supportCtrl', function($scope,$rootScope) {

$rootScope.extras=true;

})

.controller('forgotPasswordCtrl', function($scope,$rootScope) {

$rootScope.extras=false;

})

.controller('checkoutCtrl', function($scope,$rootScope,sharedUtils,$state,$firebaseArray,

$ionicHistory,fireBaseData, $ionicPopup,sharedCartService) {

$rootScope.extras=true;

//Check if user already logged in

firebase.auth().onAuthStateChanged(function(user) {

if (user) {

$scope.addresses= $firebaseArray( fireBaseData.refUser().child(user.uid).child("address") );

$scope.user\_info=user;

}

});

scope.payments = [

{id: 'CREDIT', name: 'Credit Card'},

{id: 'NETBANK', name: 'Net Banking'},

{id: 'COD', name: 'COD'}

];

$scope.pay=function(address,payment){

if(address==null || payment==null){

//Check if the checkboxes are selected ?

sharedUtils.showAlert("Error","Please choose from the Address and Payment Modes.")

}

else {

// Loop throw all the cart item

for (var i = 0; i < sharedCartService.cart\_items.length; i++) {

//Add cart item to order table

fireBaseData.refOrder().push({

//Product data is hardcoded for simplicity

product\_name: sharedCartService.cart\_items[i].item\_name,

product\_price: sharedCartService.cart\_items[i].item\_price,

product\_image: sharedCartService.cart\_items[i].item\_image,

product\_id: sharedCartService.cart\_items[i].$id,

//item data

item\_qty: sharedCartService.cart\_items[i].item\_qty,

//Order data

user\_id: $scope.user\_info.uid,

user\_name:$scope.user\_info.displayName,

address\_id: address,

payment\_id: payment,

status: "Queued"

});

}

//Remove users cart

fireBaseData.refCart().child($scope.user\_info.uid).remove();

sharedUtils.showAlert("Info", "Order Successfull");

// Go to past order page

$ionicHistory.nextViewOptions({

historyRoot: true

});

$state.go('lastOrders', {}, {location: "replace", reload: true});

}

}

$scope.addManipulation = function(edit\_val) { // Takes care of address add and edit ie Address Manipulator

if(edit\_val!=null) {

$scope.data = edit\_val; // For editing address

var title="Edit Address";

var sub\_title="Edit your address";

}

else {

$scope.data = {}; // For adding new address

var title="Add Address";

var sub\_title="Add your new address";

}

// An elaborate, custom popup

var addressPopup = $ionicPopup.show({

template: '<input type="text" placeholder="Nick Name" ng-model="data.nickname"> <br/> ' +

'<input type="text" placeholder="Address" ng-model="data.address"> <br/> ' +

'<input type="number" placeholder="Pincode" ng-model="data.pin"> <br/> ' +

'<input type="number" placeholder="Phone" ng-model="data.phone">',

title: title,

subTitle: sub\_title,

scope: $scope,

buttons: [

{ text: 'Close' },

{

text: '<b>Save</b>',

type: 'button-positive',

onTap: function(e) {

if (!$scope.data.nickname || !$scope.data.address || !$scope.data.pin || !$scope.data.phone ) {

e.preventDefault(); //don't allow the user to close unless he enters full details

} else {

return $scope.data;

}

}

}

]

});

addressPopup.then(function(res) {

if(edit\_val!=null) {

//Update address

fireBaseData.refUser().child($scope.user\_info.uid).child("address").child(edit\_val.$id).update({ // set

nickname: res.nickname,

address: res.address,

pin: res.pin,

phone: res.phone

});

}else{

//Add new address

fireBaseData.refUser().child($scope.user\_info.uid).child("address").push({ // set

nickname: res.nickname,

address: res.address,

pin: res.pin,

phone: res.phone

});

}

});

};

})

**A2 SCREEN SHOTS**

**ADMIN SCREENSHOT**

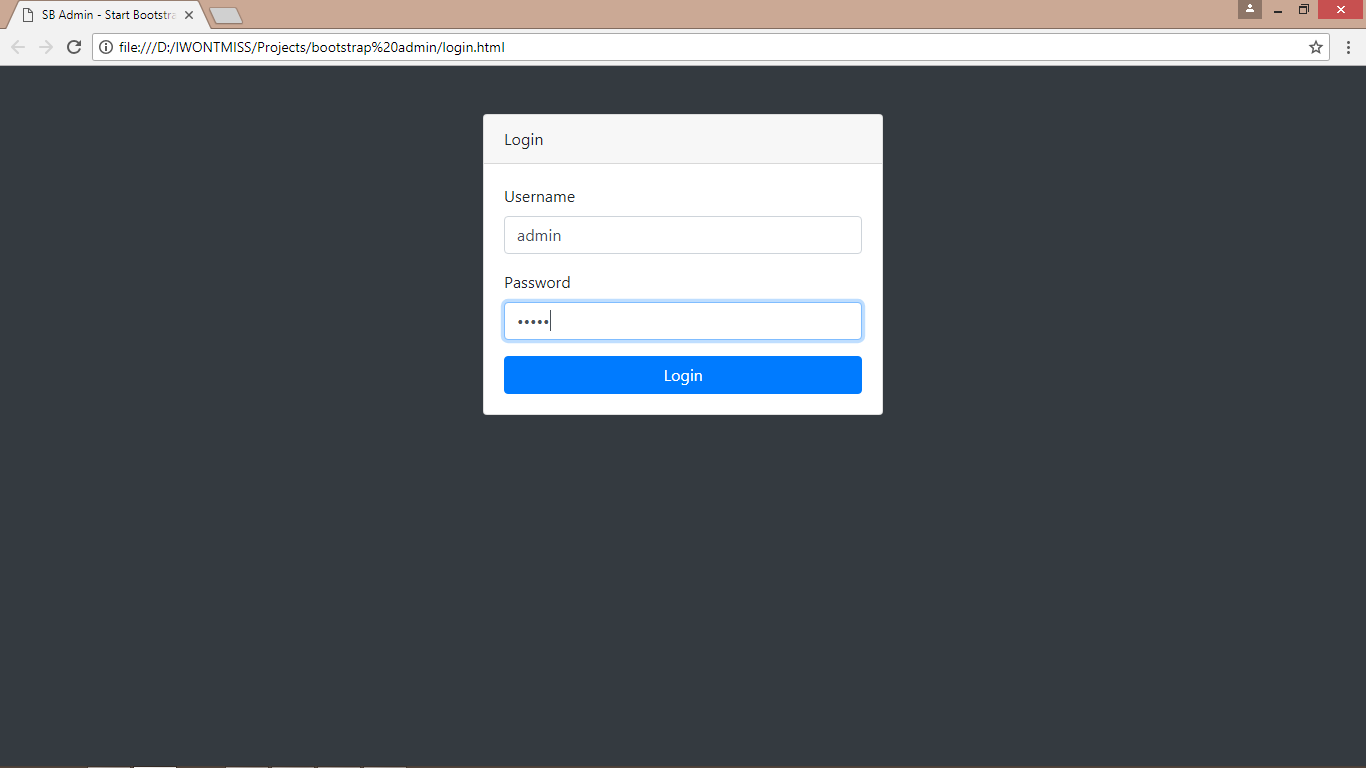
****

Fig A2.1 Admin Login

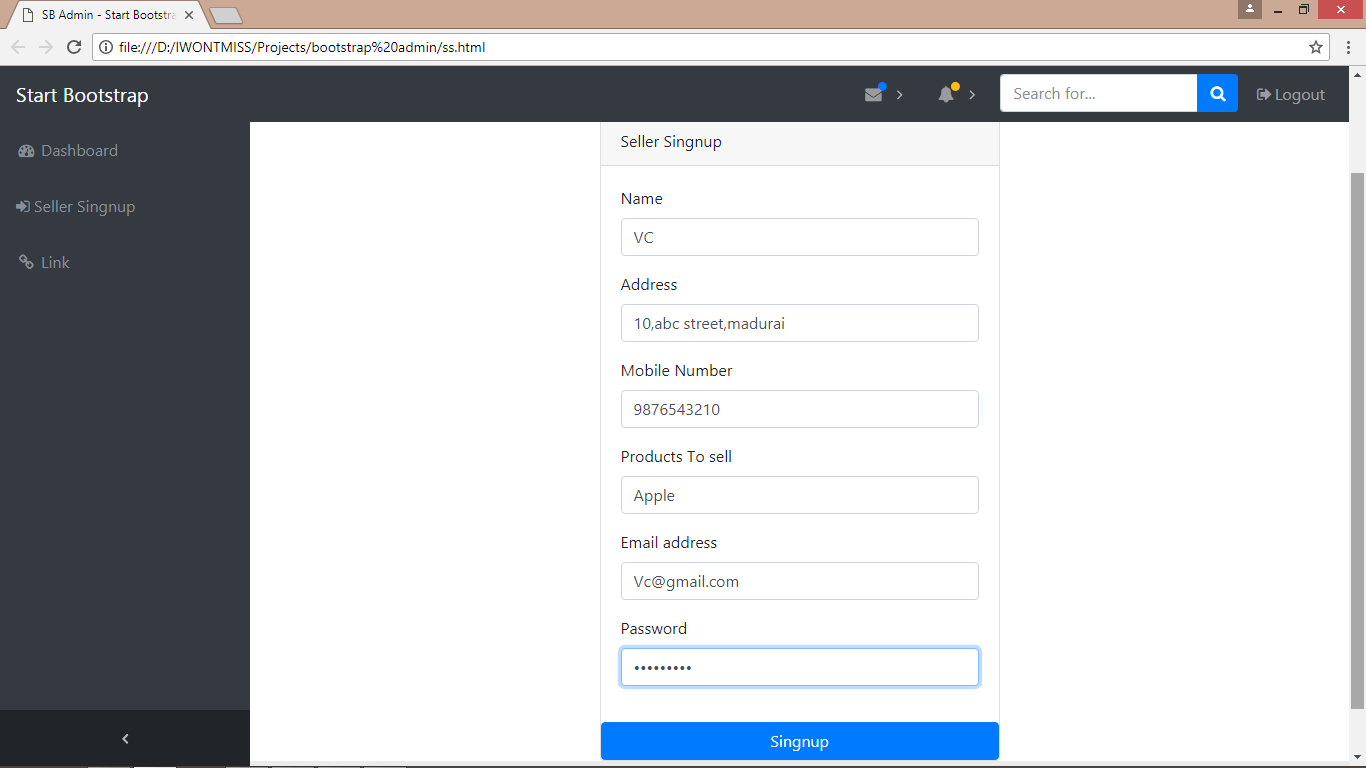
****

Fig A2.2 Seller Signup

**SELLER SCREENSHOT**

****

Fig A2.3 Seller Login

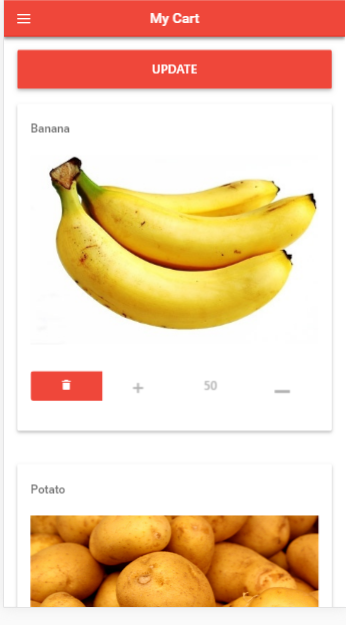
****

Fig A2.4 Seller Update Page

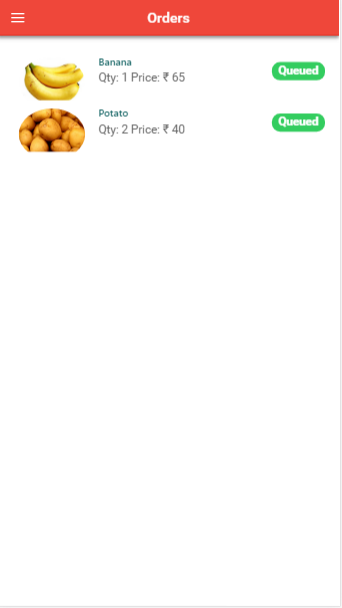
****

Fig A2.5 Order Details

**BUYER SCREENSHOT**

****

Fig A2.6 Buyer Signup

****

Fig A2.7 Buyer Login

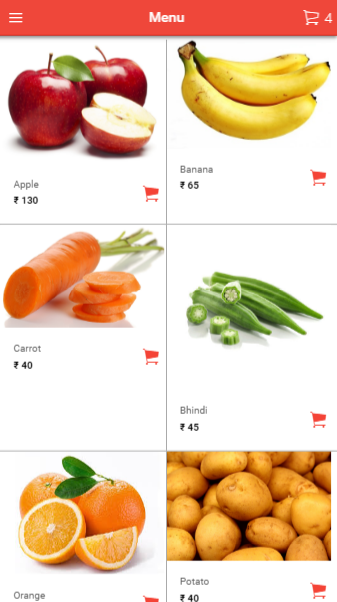
****

Fig A2.8 Menu Page

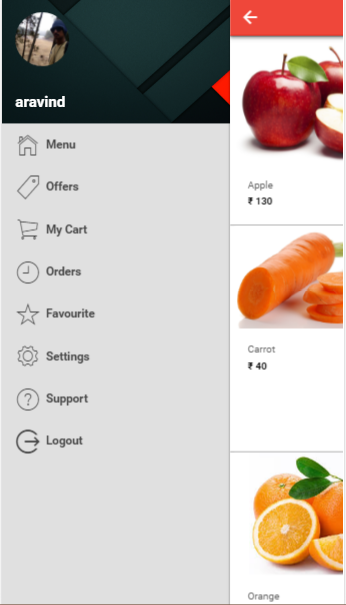
****

Fig A2.9 Slider

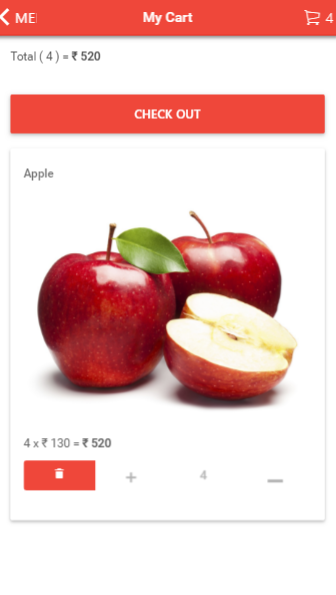
****

Fig A2.10 Buyer Cart

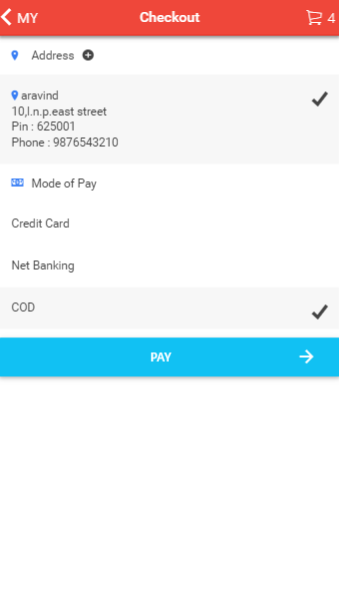
****

Fig A2.11 Checkout

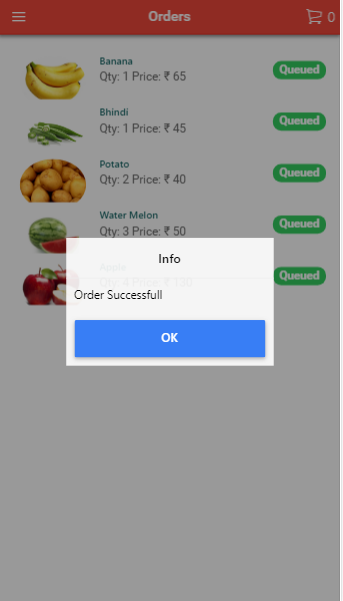
****

Fig A2.12 Orders

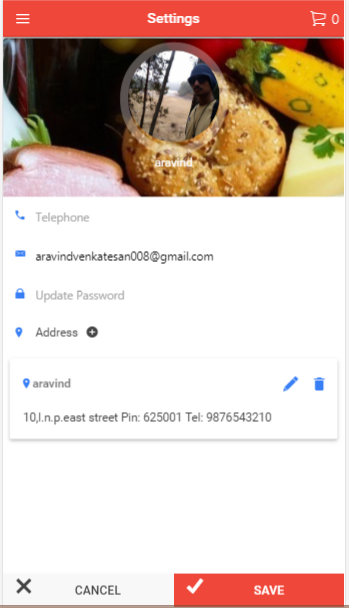
****

Fig A2.13 Buyer Settings

**REFERENCES**

**BOOK REFERENCE**

1. Brad Dayley and Brendan Dayley. (2015) ‘Angularjs,JavaScript,andJQuery’ - Sams Publishing
2. Cory Gackenheimer.(2013) ‘Node.js Recipes: A Problem-Solution Approach’ – Apress
3. Fritz Schneider and Thomas Powell.(2001) ‘Java Script: The Complete Reference’ - McGraw-Hill
4. Thomas Powell.(2010) ‘ HTML & CSS: The Complete Reference Fifth Edition’ - McGraw Hill Professional

**WEBSITE REFERENCE**

[**https://angularjs.org/**](https://angularjs.org/)

[**https://firebase.google.com/docs/**](https://firebase.google.com/docs/)

[**https://ionicframework.com/**](https://ionicframework.com/)

[**https://nodejs.org/en/docs/**](https://nodejs.org/en/docs/)

**YOUTUBE REFERENCE**

1. Angular firebase
2. codedamn
3. Paul halliday