SKIP THE QUEUE

A project report submitted in partial fulfilment of the requirement for the Award of the Degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

by

A.SRI PRAGNA (160716733005)

E.BHARGAV (160716733017)

K.MANASA (160716733026)

Under the Guidance of

Mr. K.KISHORE KUMAR,

Assistant Professor, Dept. of CSE

Department of Computer Science and Engineering

Methodist College of Engineering and Technology, King Koti, Abids, Hyderabad-500001.

2019-2020

SKIP THE QUEUE

A project report submitted in partial fulfilment of the requirement for the Award of the Degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

by

A.Sri Pragna (160716733005)

E.Bhargav (160716733017)

K.Manasa (160716733026)

Under the Guidance of

Mr. K.Kishore Kumar, Assistant Professor, Dept. of CSE

Department of Computer Science and Engineering

Methodist College of Engineering and Technology, King Koti, Abids, Hyderabad-500001.

2019-2020

Methodist College of Engineering and Technology, King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

DECLARATION BY THE CANDIDATES

We, **A.Sri Pragna (160716733005), E.Bhargav (160716733017), K.Manasa (160716733026)** students of Methodist College of Engineering and Technology, pursuing Bachelor's degree in Computer Science and Engineering, hereby declare that this project report entitled "**Skip The Queue",** carried out under the guidance of **Mr. K.Kishore Kumar** submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science. This is a record work carried out by us and the results embodied in this project have not been reproduced/copied from any source.

A.Sri Pragna (160716733005)

E.Bhargav(160716733017)

K.Manasa(160716733026)

Methodist College of Engineering and Technology, King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

CERTIFICATE BY THE SUPERVISOR

This is to certify that this project report entitled "**Skip The Queue",** being submitted *by* A.Sri Pragna (160716733005), E.Bhargav (160716733017), K.Manasa (160716733026), submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering, during the academic year 2019-2020, is a bonafide record of work carried out by them.

Mr. K.Kishore Kumar
AssistantProfessor,
Dept. of CSE

Methodist College of Engineering and Technology, King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that this project report entitled "**SKIP THE QUEUE**" *by* A.Sri Pragna (160716733005), E.Bhargav (160716733017), K.Manasa (160716733026), submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering of the Osmania University, Hyderabad, during the academic year 2019-2020, is a bonafide record of work carried out by them.

Dr. P. Lavanya
Associate Professor &
Head of the Department

Methodist College of Engineering and Technology, King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

PROJECT APPROVAL CERTIFICATE

This is to certify that this project report entitled "**SKIP THE QUEUE**" by **A.Sri Pragna (160716733005), E.Bhargav (160716733017), K.Manasa (160716733026)**, submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering of the Osmania University, Hyderabad, during the academic year 2019-2020, is a bonafide record of work carried out by them.

INTERNAL **EXTERNAL** **HOD**

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project guide **Mr.K.Kishore Kumar, Assistant Professor,** for giving us the opportunity to work on this topic. It would never be possible for us to take this project to this level without his innovative ideas and his relentless support and encouragement.

We would like to thank our project coordinator **Mr. Sandeep Ravikanti, Assistant Professor,** who helped us by being an example of high vision and pushing towards greater limits of achievement.

Our sincere thanks to **Dr. P. Lavanya, Associate Professor** and **Head of the Department of Computer Science and Engineering,** for her valuable guidance and encouragement which has played a major role in the completion of the project and for helping us by being an example of high vision and pushing towards greater limits of achievement.

We would like to express a deep sense of gratitude towards the **Dr. Prabhu G. Benakop, Principal, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

We would like to express a deep sense of gratitude towards the **Dr. M. Lakshmipathi Rao, Director, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

We are indebted to the Department of Computer Science & Engineering and Methodist College of Engineering and Technology for providing us with all the required facility to carry our work in a congenial environment. We extend our gratitude to the CSE Department staff for providing us to the needful time to time whenever requested.

We would like to thank our parents for allowing us to realize our potential, all the support they have provided us over the years was the greatest gift anyone has ever given us and also for teaching us the value of hard work and education. Our parents have offered us with tremendous support and encouragement, thanks to our parents for all the moral support and the amazing opportunities they have given us over the years.

Department of Computer Science & Engineering

Vision & Mission

VISION

To become a leader in providing Computer Science & Engineering education with emphasis on knowledge and innovation.

MISSION

M1: To offer flexible programs of study with collaborations to suit industry needs
M2: To provide quality education and training through novel pedagogical practices
M3: Expedite high performance of excellence in teaching, research and innovations.
M4: To impart moral, ethical valued education with social responsibility.

Program Educational Objectives

Graduates of Compute Science and Engineering at Methodist College of Engineering and Technology will beable to:

PEO1: Apply technical concepts, Analyze, Synthesize data to Design and create novel products and solutions for the real life problems.

PEO2: Apply the knowledge of Computer Science Engineering to pursue higher education with due consideration to environment and society.

PEO3: Promote collaborative learning and spirit of team work through multidisciplinary projects

PEO4: Engage in life-long learning and develop entrepreneurial skills.

Program Specific Outcomes

At the end of 4 years, Compute Science and Engineering graduates at MCET will beable to:

PSO1: Apply the knowledge of Computer Science and Engineering in various domains like networking and data mining to manage projects in multidisciplinary environments.

PSO2: Develop software applications with open-ended programming environments**.**

PSO3: Design and develop solutions by following standard software engineering principles and implement by using suitable programming languages and platforms

PROGRAM OUTCOMES

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in

ABSTRACT

Now-a-day shopping in mall is increasing rapidly .While shopping customers have to face some difficulties. First, they go at the billing counter for payments of their total expenditure but there are many people standing in queue for billing purpose. This is unnecessary waste time for customer because cashier scans each item bought with barcode. Barcode technology is time consuming process. Second , customer can't predict their total expenditure so overall purchase total might be greater than budget. Thus, the app overcomes all of these drawbacks faced by customers. This app has been introduced to avoid waiting in billing queue. It uses the camera of the customer's phone as barcode scanner and an online billing system is provided. So, the customer can pay for scan and pay for his items at ease skipping the hassle to wait in a long queue.

Contents

List of Figures

1. Introduction

1.1 Problem Statement

Frequently, people encounter a problem of spending too much of their time waiting in queues for billing their purchases in different shopping centres or supermarkets. Waiting in queues negatively affects human morale and may cause misunderstandings or conflicts amongst people, for instance, when someone breaks the line and stands in front of other people .This project aims to eliminate this problem by introducing a novel alternative to traditional billing methods, speeding up the payment process.

1.2 Bar Codes

The vast majority of modern super markets use barcode system to identify products and check-in customers waiting in queue. Barcodes represent a series of vertical black lines of different thickness and separate distance which can be coded on product packaging[5]. The barcode reader reads the data represented by barcodes .In modern supermarkets; this data involves a unique ID of each product. All information describing a particular product such as full name, cost, weight, etc. are stored in primary software database, and this product is addressed by unique ID that is read from the barcodes .This way, one can easily get all information about a particular product just by scanning the barcode that is printed in some area on product packaging. Hence, existing system of barcodes in supermarkets induces possibilities such as automatic bill calculation of total cost of purchased items, generating bills and item listings.

The main problem with the existing system of barcode billing is the fact that each product is scanned only one at a time so that scanning time grows gradually when there are plenty of purchased products. Therefore, people tend to line up in queues in front of cashier's desk.

1.3 Solution

One measure to reduce the waiting time of customers is to introduce a Self-checkout[4] App. Through which checkout is done in three simple steps

1. Scan
2. Pay
3. Checkout

The Scanning of the products is done through the customer's phone camera the backend of the application will have a predefined database populated with products and their details. Another app is created for backend to handle products and its details. This could be accessible only through the store manager or its employees.

The App uses Razor Pay payment gateway to perform online payments .The gateway provides various options such as UPI, net banking etc. If the customer chooses to pay through cash a bill is generated is printed through store's wifi- printer. The customer can deposit the total amount at cash desk and leave.

1.4 Benefits

In the community perspective, this project grants obvious benefits as it has potential to decrease significantly the queuing time of customers and save much of precious time of every individual shopper. For instance, according to a research of British researchers, average queuing time in stores is 13 minutes and maximum queuing time sometimes even lasts 40 minutes. Which is sufficient to miss most of the important activities such as airplane departure .On the other hand, in the market owner's and stakeholders perspective , this app will be beneficial regarding attracting more customers, since their market will provide fast service and save shoppers time .Moreover , the new shopping experience and emerging technology attract more people, especially the young generation.

2. Literature Survey

2.1 Authors: Mr. P. Chandrasekhar, Mrs. T. Sangeetha, "Smart Cart with Automatic Billing, Product Information, Product Recommendation Using RFID & ZigBee with Anti-Theft," Procedia Computer Science Vol 79, pp 792- 800, March 2016.

A supermarket is a place where customers come to purchase their daily using products and pay for that. So there is a need to calculate how many products sold and generate the bill for the customer. When we go to shopping mart for shopping, customers have to work for selecting the right product. Also, after that it is hectic to stand in line for billing all the goods. Hence, a smart shopping cart system is proposed which will keep the track of purchased products and also online transaction for billing using RFID and ZigBee. The system will also give suggestions for products to buy based on user purchase history from a centralized system. In this system, every product in Mart will have RFID tag, and every cart will be having RFID Reader and ZigBee attached to it .There will be centralized system. There will be a centralized system for the recommendation and online transaction. Moreover, also there will be RFID reader at the exit door for anti-theft.

2.2 Authors: Mr.Raju Kumar ,Mr. Gopala Krishna, Mr. K. Ramesha "Intelligent Shopping Cart ", IJESIT Vol 2, Issue 4, July 2013.

An innovative product with societal acceptance is that one that aids the comfort, convenience and efficiency in everyday life. Purchasing and shopping at big malls is becoming daily activity in metro cities. We can see big rush at these malls on holidays and weekends. People purchase different items and put them in trolley. After completion of purchases, one needs to go to billing counter for payments. At billing counter the cashier prepare the bill using the barcode reader which is very

time consuming process and results in log queue at billing counter. So the intelligent cart system is developed to minimize this trouble .The main objective of this system is to provide a technology oriented, low cost, easily scalable , and rugged system for assisting shopping in person. The developed system consists of 3 key components

(a) Server Communication component(SSC)

(b) User Interface and display component(UIDC)

(c) Automatic Billing component(ABC)

SSC establishes and maintains the connection of shopping cart with main server. UIDC provides the user interface and ABC handles billing in association with SSC. These three modules are integrated into an embedded system and are tested to satisfy the functionality.

2.3 Authors: Mr. Narendra Kumar Nagam, Mr. K.V.S. Sarma, "Smart Phone Self Checkout Payment in Super bazaar", IEEE, 2019 Fifth International Conference February 2019.

With the increasing super bazaar purchases by the customers in cities there is a problem of bottle neck for billing the customers at peak times, to solve this problem we propose a secure way for billing the purchased goods by the customer with their Smartphone's .There are other self checkouts that are present such as 'Amazon go', 'Perpule 1 Pay', and self checkout terminals , some of the solutions uses RFID tags, these solutions are costlier and difficult to implement in countries such as India. 'Amazon go' uses computer vision, Deep learning algorithms, Sensor Fusion, Our solution is a combination of Smartphone QR code Scanner app and Secure website for billing the products. The verification of purchased products by the guard will be just weighing purchased products. Many other security extensions can be planned in order to prevent theft like installation of Anti- Theft system etc.

3. Design Analysis

3.1 System Architecture

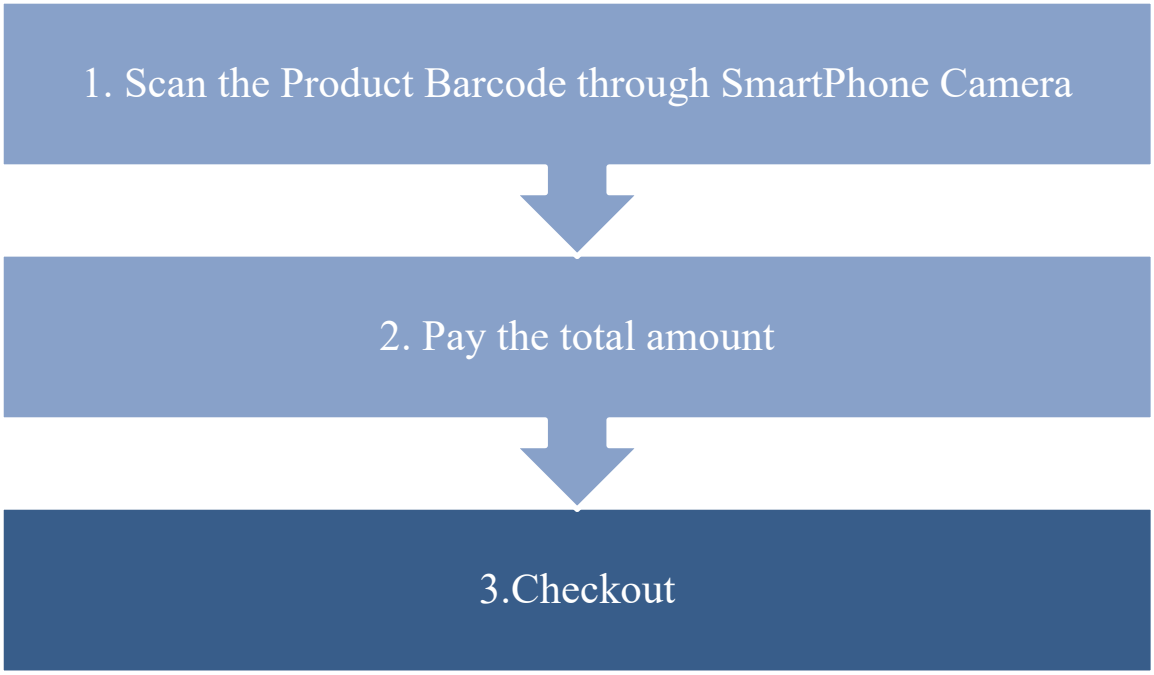


Figure 3.1 System Architecture

3.2 ER Diagram :

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

There are five main components of ERD:

- **Entities:** These are represented by rectangles. An entity is an object or concept about which you want to store information.

A weak entity is an entity that must be defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone

.

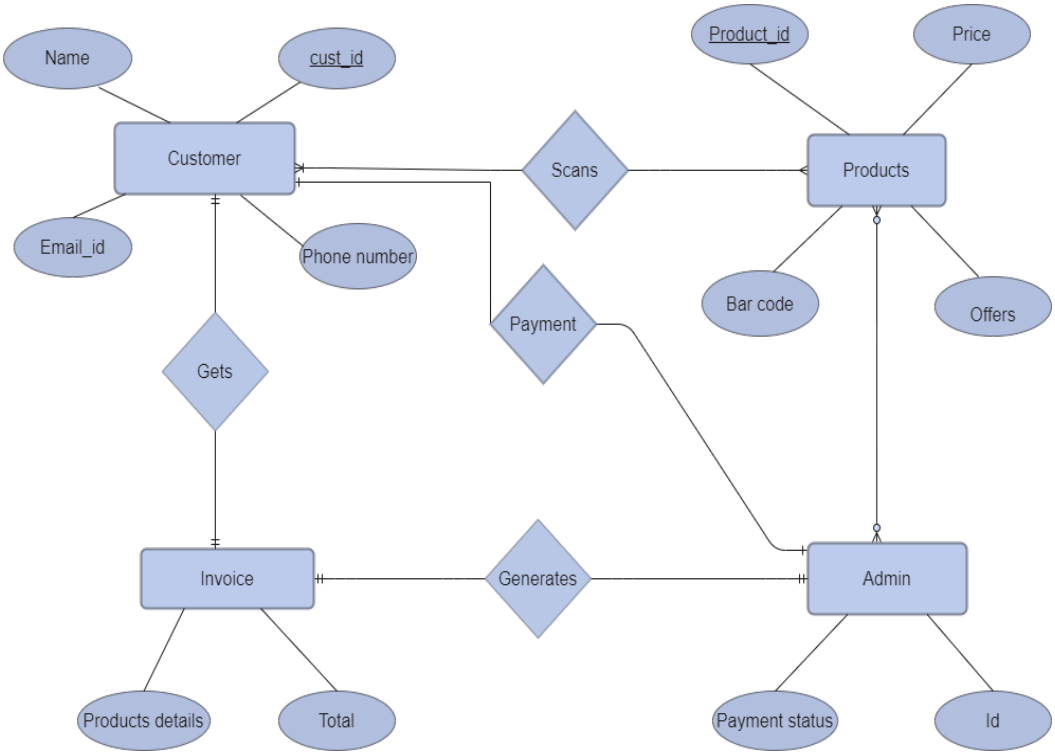
- **Actions:** These are represented by diamond shapes, which shows how two entities share information in database

- **Attributes:** These are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee‘s social security number might be the employee’s key attribute.

  A multivalve attribute can have more than one value. For example, an employee entity can have multiple skill values.

  A derived attribute is based on another attribute. For example, an employee’s monthly salary is based on the employee’s annual salary.

- **Cardinality:** These are the solid lines that connect attributes to show the relationships of entities in the diagram. Cardinality specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships.



.

Figure 3.2.1: ER Diagram

In the above diagram

Entities: Customer, Invoice, Products, Admin

Attributes:

- **Customer:** Name, Email_id, Phone number.
  Primary key: Cust_id.

- **Invoice:** Product details, Total.

- **Admin:** Payment status, id.
- **Products:** Price, Barcode, Offers.

  Primary key: product_id.

Relationships: scans, gets, payment, generates

The customer entity and products entity has one or many cardinality relationship as the customer can scan one or more products during his purchase.

The customer entity and Admin entity has one to one cardinality relationship as customer has to pay only once for one transaction.

The products entity and Admin entity has zero or more cardinality relationship as the database may have zero or more products.

The Admin and Invoice entity have one to one cardinality relationship as only one bill is generated for one transaction.

The cust_id and Product_id are placed as primary key in order to identify the customer and products uniquely.

3.3UML Diagrams

UML stands for unified modeling language.UML is a standardized general purpose modeling language in the field of object-oriented software engineering .The standardized is managed, and was created by, the object Management Group .The goal is for UML to become a common language for creating models of object oriented computer software.UML is compromised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to: or association with UML .It is important to distinguish between the UML model and the set of diagrams of a system .A diagram is a partial graphic representation of a system's model. The set of diagrams need not completely cover the model and deleting a diagram does not change the model. The model may also contain documentation that drives the model elements and diagrams. In 1997 UML was adopted as a standard by the model elements and diagrams. In 1997 UML was adopted as a standard by the object Management group (OMG), and has been managed by this organization ever since. In 2005 UML was also published by the International Organization for Standardization (ISO) as an approved ISO standard. Since then it has been periodically revised to cover the latest revision of UML. UML is not a development method by itself; however, it was designed to be compatible with the leading object-oriented software development methods of its time, for example OMT, Booch method, Objector and especially RUP that it was originally intended to be used with when work began at Rational Software. UML (Unified Modeling Language) is a standard notation for the modeling of real-world objects as a first step in developing an object-oriented design methodology. The major perspectives of a UML are Design, Implementation, process and Deployment. The centre is the Use Case view which connects all these four. A Use case represents the functionality of the system.

3.4 Use Case diagram:

- Use case diagrams are usually referred to as behaviour diagrams.
- It is used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).
- The main purpose of a Use Case diagram is to show what system functions are performed for which actor.
- Use Case diagrams are drawn to capture the functional requirements of a system.
- Major elements of Use Case diagram are actor, use case, include, extend.
- Extend- Indicates that a use case may include the behaviour specified by base use case.
- Include- when a use case is depicted as using the functionality of another use case, the relationship between the use case is named as include or uses relationship.
- Use case is generally simple and does not show following details:
    - Summarization of relationships between use cases, actors and systems
    - It does not show the order in which steps are performed.
- The general purpose of Using Use case Diagram is to:

    - Specify the context of a system
    - Capture requirement of a system
    - Validate a systems architecture
    - Drive implementation and generate test cases

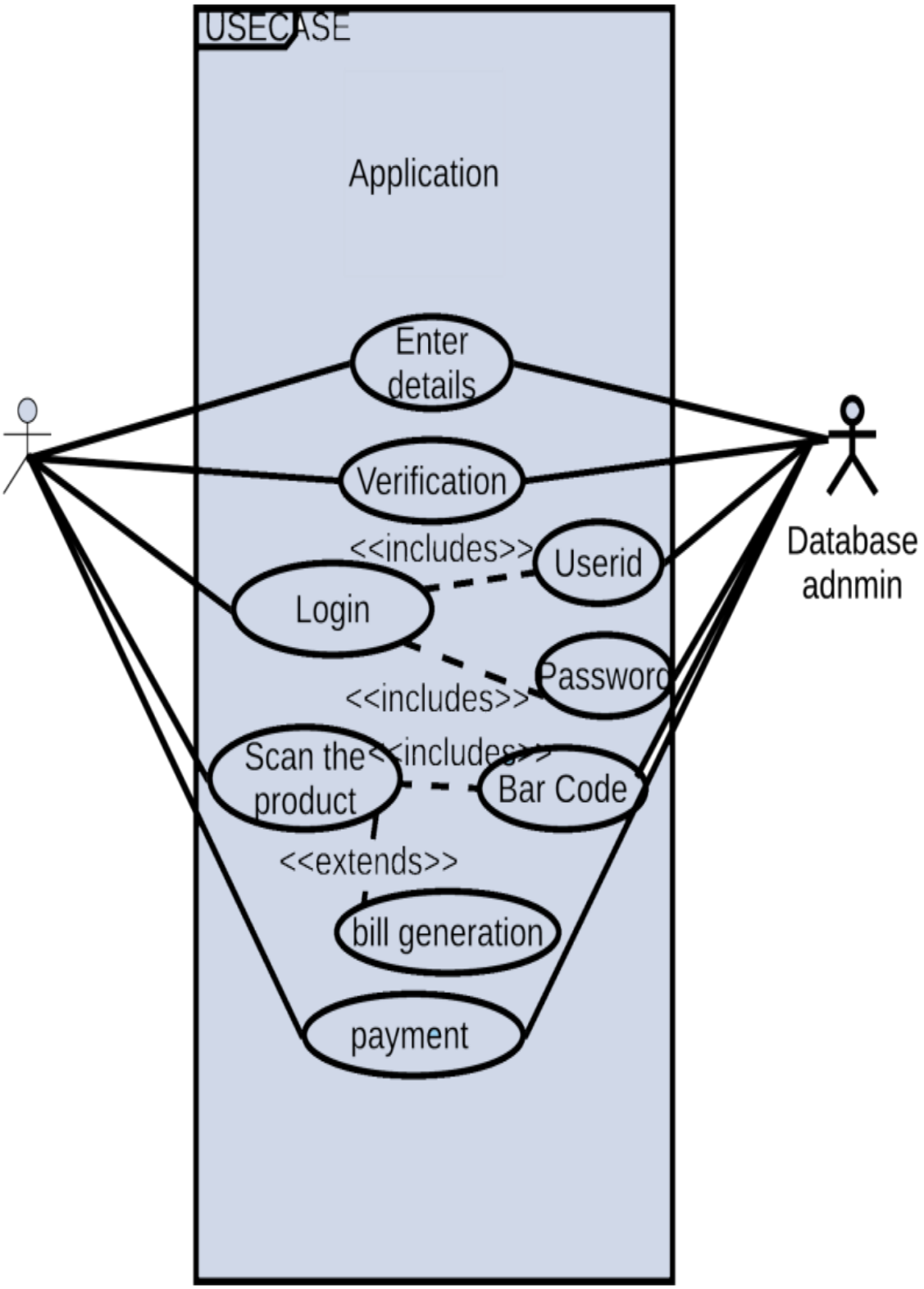


Figure 3.4: Use Case Diagram

- The above diagram consists of following components:
  - Actors: User, Database Admin
  - Relations: Includes
  - The actor on the left is referred as user indicating the person who uses the app
  - The actor on the right is referred as database admin who manages the database of the app
- In the above use case diagram first the user has to register with his/her details, and then those details are stored in the database.
- When the user wants to use the app he/she would login with his login details, then those details are verified
- .Database will verify those details and gives the confirmation message to the user.
- If the details are credible the access is provided to the user.
- The user after log-in can scan the products and fetch the details of the products.
- If the scanned products are available in the database then the products are added to the cart.
- If the products are not present in the database it is notified to the user. Through alert message.
- After completion of purchase by the user the bill is generated by the app.
- User can pay the bill either with cash or cashless payments.
- If the payment is done by cash then the bill sent to the Wifi printer in the PDF format
- The user can collect the bill at the bill desk and deposit the cash to the respective store employee

.

3.5 Sequence diagram:

- A sequence diagram shows object interactions arranged in time sequence.
- It depicts object and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.
- Sometimes, it is also called as event diagrams, event scenarios, timing diagram.
- A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order.
- The horizontal axis shows the elements that are involved in the interaction.
- The vertical axis represents time proceedings down the page.
- Time in a sequence diagram is not relevant for the duration of the interaction.
- The destruction or the end of life period of the object is denoted by a cross mark at the end of the diagram

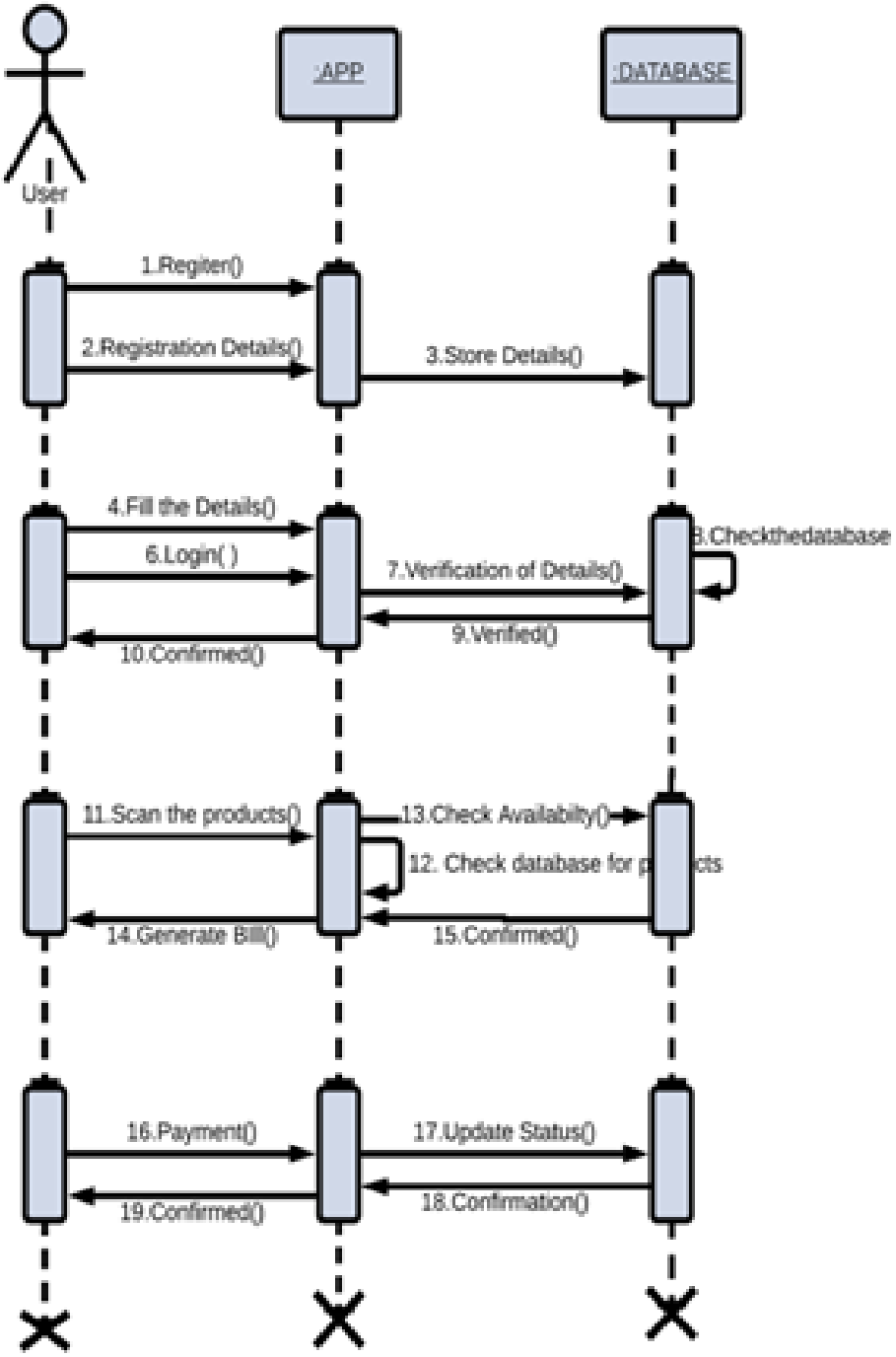


Figure 3.5: Sequence Diagram

- In the above Sequence diagram the user will register through app, those details are stored in the database.
- If the user want to login he/she needs to fill the details and click on login, then the app will verify the details entered by the user with the details in the database.
- It will check all the details available in the database. So, a self loop is given as the database checks itself to cross-verify the details.
- If the details entered by the user are present in the database then the app will give conformation message to the user and user is logged into the app.
- After login user can scan the products, here self loop indicates that the user can scan multiple products.
- After scanning the products the app will check that the scanned product available in the database or not. If the product is available in the database then the product will add to the cart.
- After the purchase of the user the app will generate the bill.
- Now the user will do the payment, the app will update the payment details to the database.
- The database will give the conformation Message to the app and the app will give the conformation to the user.

3.6 Activity diagram:

- Activity diagram is another important behavioural diagram in UML diagram to describe dynamic aspects of the system.
- Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.
- So the control flow is drawn from one operation to another. It can be used to describe the business and operational step-by-step workflows of components in the system.
- In Activity diagram two nodes are there fork and join.
- Fork node is described as split behaviour into a set of parallel or concurrent flows of activities.
- Join node is defined as bring back together a set of parallel or concurrent flows of activities.
- Final node is defined as the activity is terminating. It is represented as a solid circle with a hollow circle inside.
- It can be thought as a goal notated as “bull eye” or target.

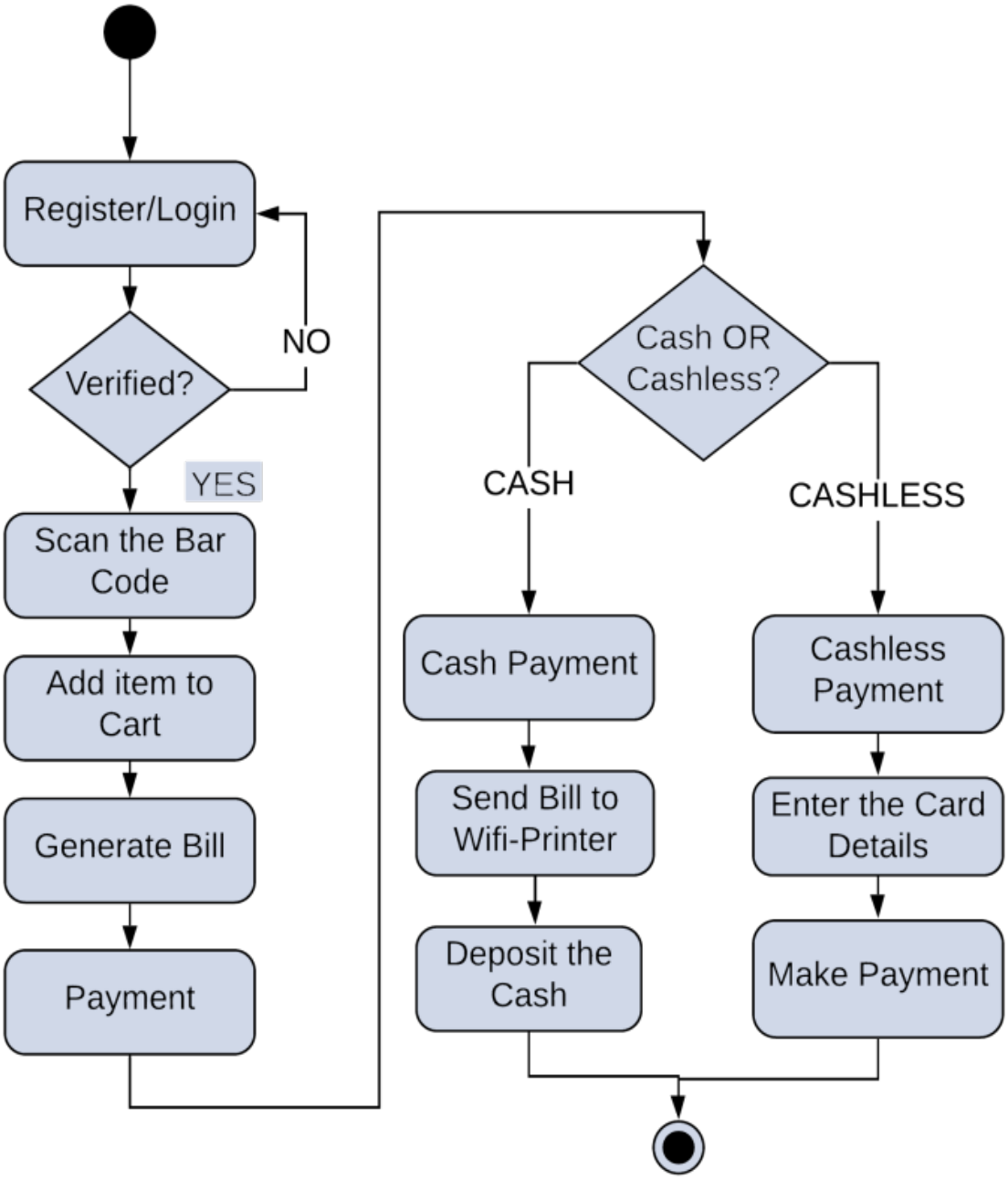


Figure 3.6: Activity Diagram

- The user will register and login with login details, those details are verified.
- If the login is successful then the user will allowed to scan the products and login is unsuccessful the user has to login again.
- After user scans the product they are added to the cart
- After the purchase is done by the user the app generates the bill.
- The user has to do the payment. He/she will choose either cash or cashless.
- If the user chooses cash payment then the bill is sent to the wifi printer and he/she will deposit the cash at the bill desk and exit.
- If the user chooses cashless payment he/she will enter the card details make payment and exit.

3.7 Class Diagram:

- Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.
- Class diagram describes the attributes and operations of a class and also the constraints imposed on the system.
- A class may be involved in one or more relationships with other classes.

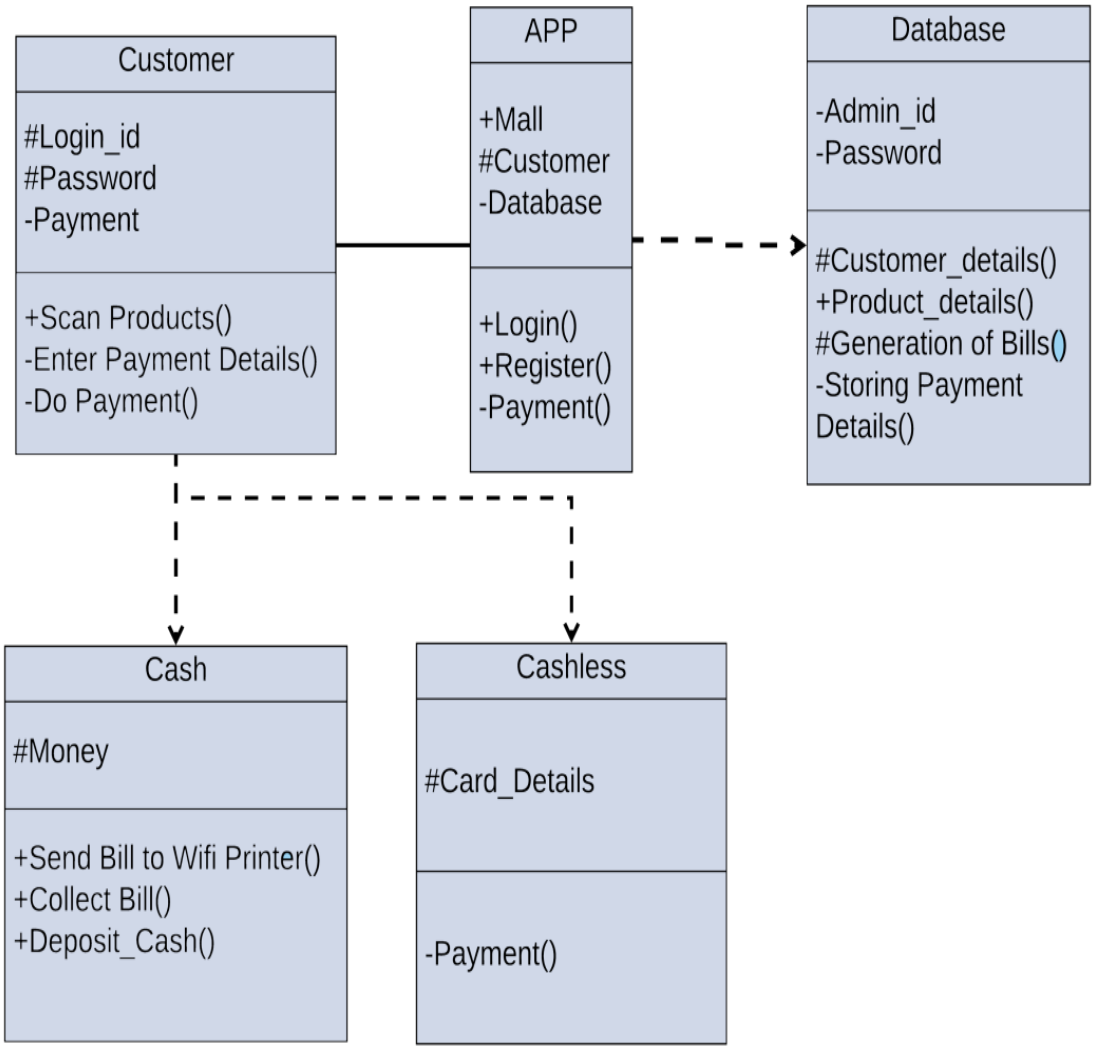


Figure 3.7 Class diagram

- Each class is represented by a rectangle having a subdivision of three compartments name, attributes and operation.
- There are three types of modifiers which are used to decide the visibility of attributes and operations.
- + is used for public visibility(for everyone)
- # is used for protected visibility.
- - is used for private visibility(for only that specific class)
- Few relations used in class diagram are :
    - Association
    - Direct Association
    - Reflexive Association
    - Multiplicity
    - Aggregation
    - Composition
    - Inheritance
    - Realization

- **Association:** It is logical connection or relationship between classes.
- **Direct Association:** It is a directional relationship represented by a line with arrow head .the arrowhead depicts a container- contained directional flow.
- **Reflexive Association**: It is relationship used to depict when a class has multiple functions or responsibilities.
- **Multiplicity:** It is the active logical association when the cardinality of a class in relation to another is being depicted.
- **Aggregation:** It refers to the formation of particular class as a result of one being aggregated or built as a collection.
- **Composition:** The composition relationship is very similar to aggregation relationship with only difference being its key purpose of emphasizing the dependence of contained class to the life cycle of the container class.
- **Inheritance**: It refers to a type of relationship where in one associated class is a child of another by virtue of assuming the same functionalities of the parent class.

- **Realization:** It denotes the implementation of the functionality defined in one class by another class. To show the relationship in UML, a broken line with an unfilled solid arrowhead is drawn from the class that defines the functionality of the class that implements the function.

- In the above diagram there are five classes
    - APP
    - Database
    - Customer
    - Cash
    - Cashless
- Customer has association relationship with the app.
- App has dependency relationship with database.
- Cash and Cashless dependency relationship with customer.
- As the decision of payment is dependent on customer the cash and cashless classes are dependent over customer.
- The App requires data in order to process so it is dependent on Database.
- The attributes and methods which are confidential are represented as private (-).
- The attributes and methods which need to be accessed by the child classes are represented as protected (#).
- The attributes and methods which need to be accessed by every class are represented by public (+).

3.8 State chart diagram:

- State Diagram is used to capture the behaviour of a class, a subsystem, a package, or even an entire system. It is also called a state chart or state transition diagram.
- State chart diagrams provide us an efficient way to model the interactions or communication that occurs within the external entities and a system. These diagrams are used to model the event-based system. A state of an object is controlled with the help of an event.
- State chart diagrams are used to describe various states of an entity within the application system.
- The state chart usually consists of start and finish node and events and states in between
- The initial state is represented with solid circle.
- The final state is represented with concentric circles.
- The events of generally four types:
    - Signal event: Representing message or signal.
    - Call event: Representing call to a particular operation or event.
    - Time event: It occurs when specified time has elapsed.
    - Change event: It occurs when specified condition is met.
- Transition lines depict movement from one state to another. Each transition line is labelled with the event that causes the transition.

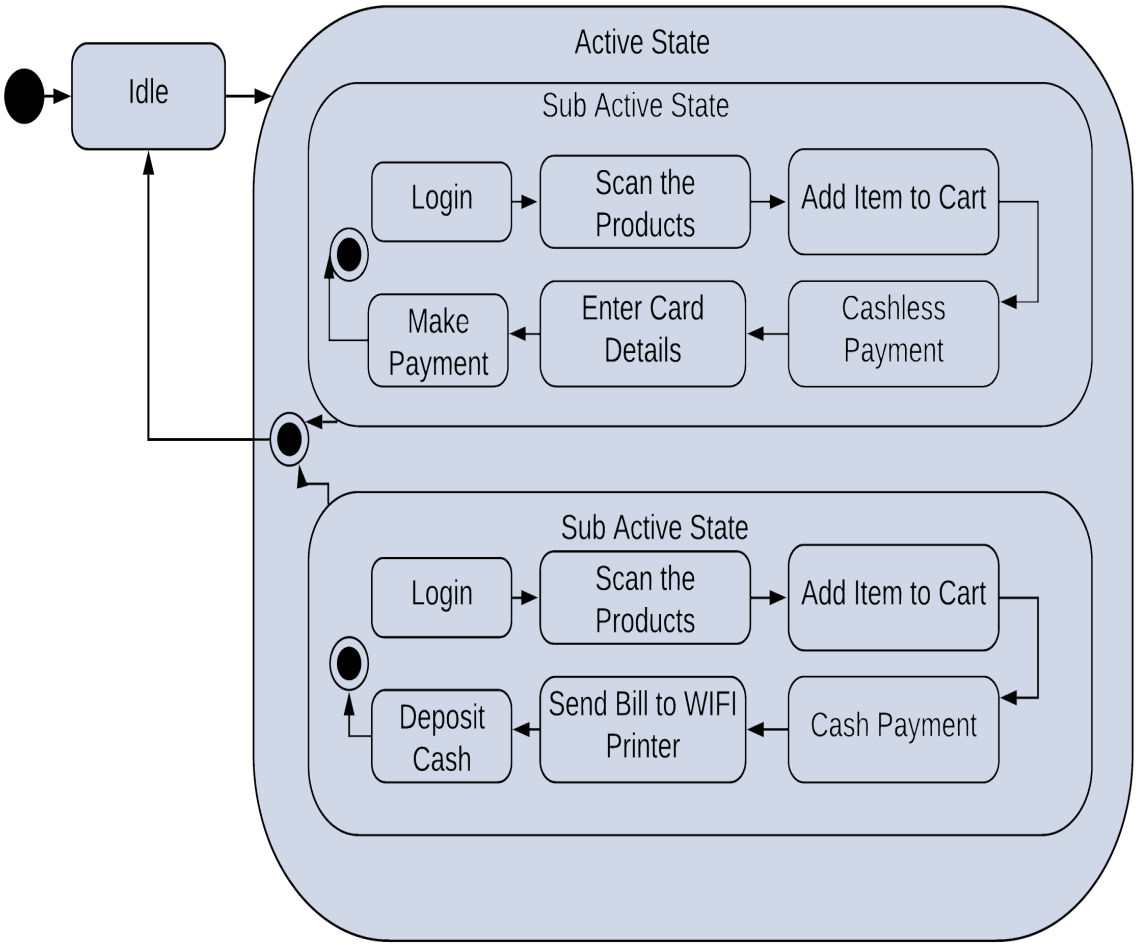


Figure 3.8 State chart diagrams

In the above State chart diagram there are two sub states.

- The user logins with the login details and scans the products, the scanned products are added to the cart the app will generate the bill, user chooses cashless payment enters the card details and will make payment and this sub state ends.
- In other sub state the user logins with the login details ,scans the products, the scanned products are added to the cart , if it is cash payment the bill is transferred to the wifi printer and the user deposits the cash at the bill desk and this sub sate ends.
- Two sub states are joined
- Both the states are entered to idle state now.

4. Implementation

4.1 Modules:

This project consists of following modules:

- User registration
- Scanning of Products
- Bill generation
- Payment
- Admin

4.2 Modules Description

4.2.1 User registration:

- In this module the user details are collected such as email id
- Few validations are performed on the mail id in order ensure the mail id used for registration is credible.
- Few validations are :
  - Valid email id.
  - Proper internet connection.
  - Strong password.
  - Prevent double registration.

4.2.2 Scanning of products:

- Once the user provides proper credentials he/she gains access to this module.
- In this modules the barcode of the products are scanned to fetch its details.
- When the user scans a particular product's barcode he is presented with three values
  - Product Name
  - Product Price

    - Product Quantity
- The details of the products are fetched from the firebase database.
- The barcode scanned by the user is compared to list of products in the database
- When the barcode is matched with the list of products the data under it are fetched.
- Product Name is the name of the product which can't be changed by the user.
- Product price is the price of the product which can't be changed by the user.
- Product Quantity is by default "1" which can be changed by user depending upon amount of quantity user is taking.
- Product Quantity is presented with two buttons plus and minus.
- Plus is used to increase the quantity.
- Minus is used to decrease the quantity.
- The total amount of products user has purchased is displayed on the top of the screen for user's convenience.

4.2.3 Bill generation

- This module is followed by previous module
- Once the user finishes his/her purchase the bill is generated by the app in the PDF format
- The bill consists of PDF with store name and address at the top centre of the page
- When user chooses to generate the bill a PDF file with present time is created in user's device.
- The products he/she has purchased are written over it. In a table format.
- The table format has 5 columns
    - Serial No
    - Product Name
    - Product Price

    - Quantity
    - Total Price
- The rows of the table are the product details he/she has purchased.
- The final row of the table is the grand total of the prices of the products.
- During the transaction whenever the user wants to review his bill it is opened from user's device and is displayed.

4.2.4 Payment

- This module is used for payment process
- In this two options are provided to the user to make his payment
    - Cash
    - Cashless
- If the user chooses the cash payment the bill which was generated and stored in user's device is retrieved.
- The user can transfer his/her bill to wifi printer, print it and deposit the cash at the bill desk.
- If the user chooses cashless payment the app is integrated with razor pay payment gateway which provides many cashless payment options like:
    - Net Banking
    - UPI
    - Card Payment
    - Phone Pay
    - Google Pay
- The user can choose his options and fill the details and an OTP is generated for the respective transaction.
- After the payment the user is notified with message and the cart would be cleared.

4.2.5 Admin

- In this module operations on the database are provided.
- The database of the app consists of tree of products with barcode as its unique id and name, price, quantity as its children.

- The user can only access the products they cannot modify them product nor its details.
- In order to provide access to the stores and store's manager a separate app is created to handle the database.
- The sub modules of this app are:
    - Password Protection
    - CRUD Operations on products.

4.2.5.1 Password Protection

- This module prevents access to non-credible users for data protection
- In order to prevent user accessing the database or any other malicious person accessing the database. The app home screen is password protected the person who knows the password can only enter the app.
- The store managers are also given accessibility to change password if they do not feel the password is strong enough.

4.2.5.2 CRUD Operations on products

- CRUD operations are create, read, update, delete.
- This module allows the users to create, delete and update products.
- The user can create the product by giving the following details:
    - Barcode of the product
    - Name of the product
    - Price of the product
- The quantity is stored "1" by default in the database.
- The user can update the product details by providing the same details as creating a product.
- If the product already exists with this barcode the details get updated.
- The user can delete the product by giving barcode of the product
- The product is identified by its barcode and its details are deleted.

5. Graphical User Interface

5.1 Overview of Skip the Queue APP

The first screen of the app is Splash Screen.

The splash screen is used as a welcome screen to the user which displays the app name and an image or logo related to the content of the app.

The image below displays the wireframe of the splash screen.

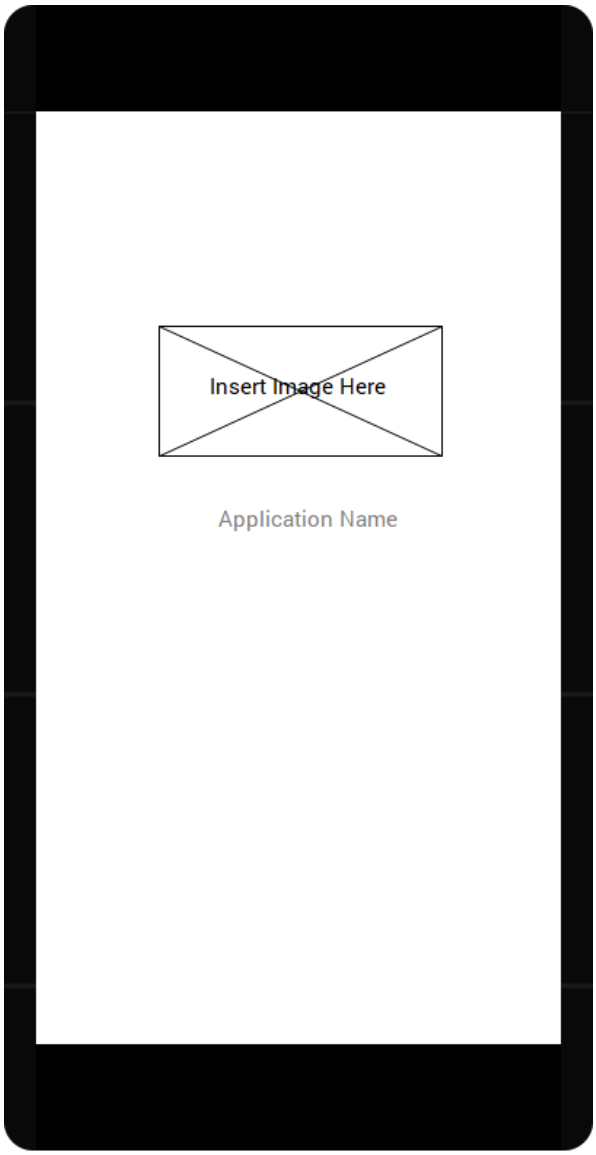


Figure 5.1.1 Wire Frame Splash Screen

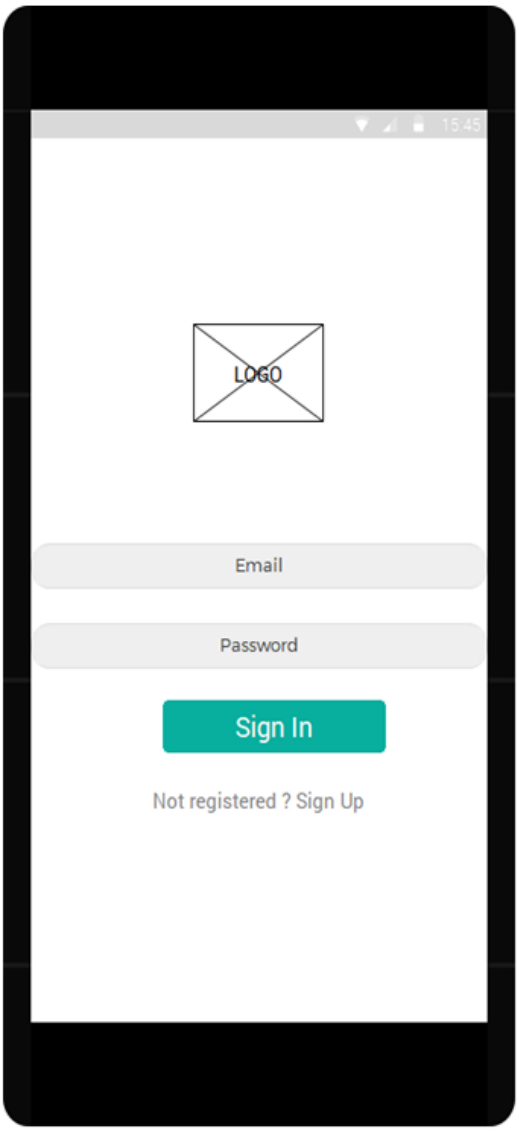


Figure 5.1.2 Wire Frame Login Screen

The next screen would be the login and signup screens where the user's email and password are collected.

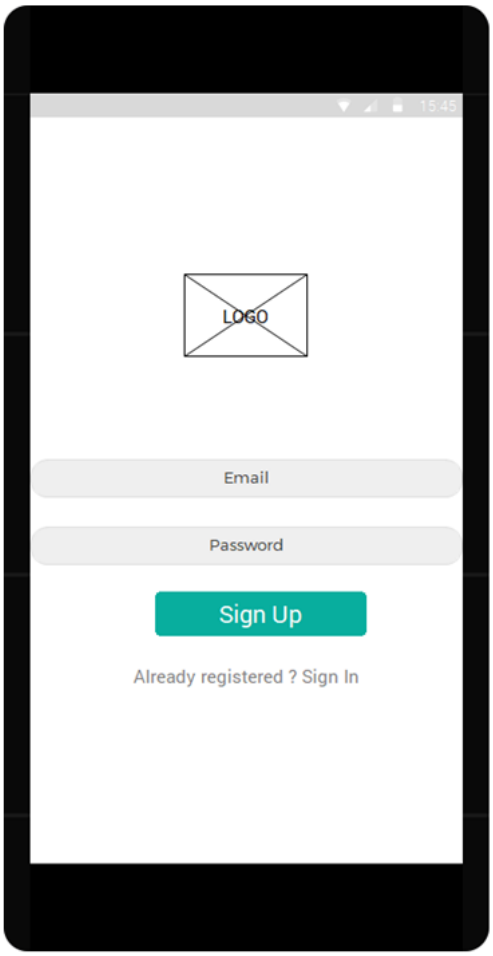


Figure 5.1.3 Wire Frame Sign up Screen

The Email and password of the user are collected. The firebase uses default API to check the valid email.

In the Sign up Screen:

- The user is supposed to provide a valid email id as in the background validation is provided for the verification of email id by the firebase.
- The password is user choice it is encrypted in the background in the database.

In the login Screen:

- The user has to provide the email id which he/she has used for registration And the user needs to sign in only once as firebase keeps the user logged in as long as he/she doesn't sign out.

The both screens are provided with validations so the user doesn't present the application with empty or invalid values

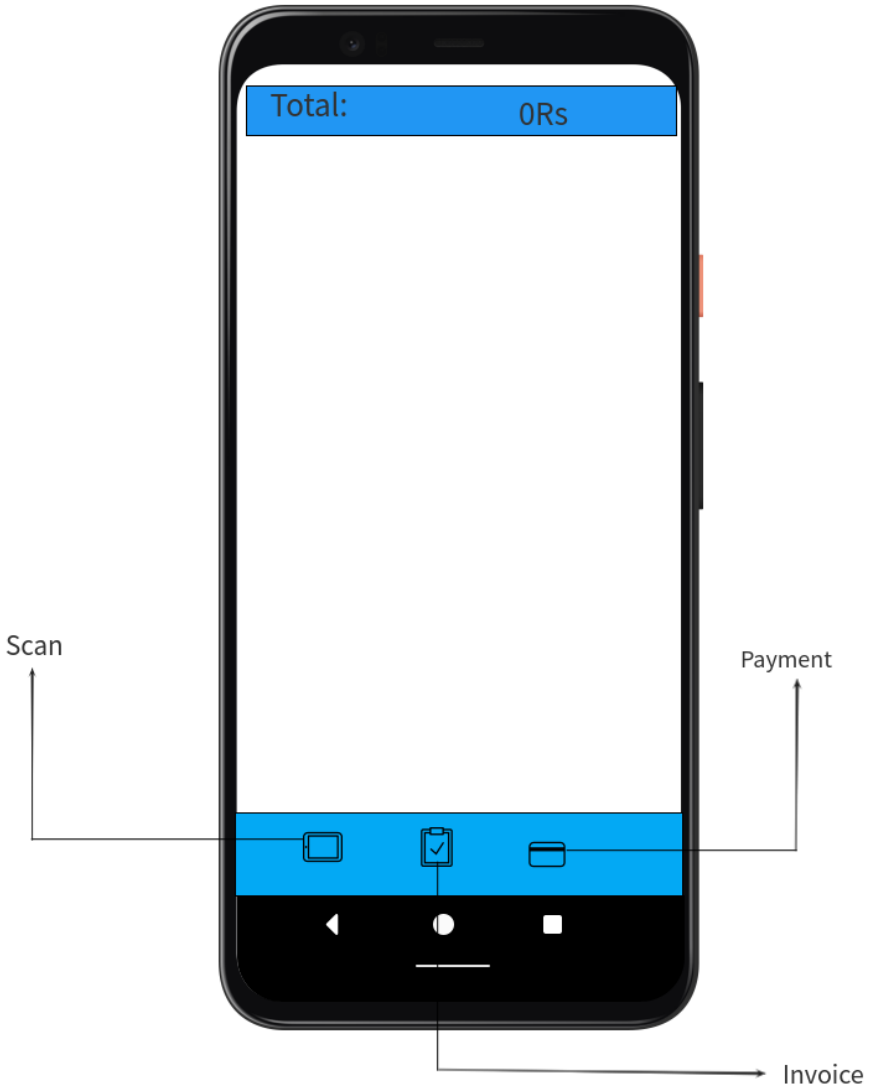


Figure 5.1.4 Wire Frame Home Screen

After user logins with valid credentials he /she taken to the home screen above.
The home Screen has three menus on bottom navigation bar

- Scan
- Invoice
- Pay

  When user chooses to scan his product he is taken to this screen where user places the barcode of the product in front of his smart phone camera and the code is extracted.

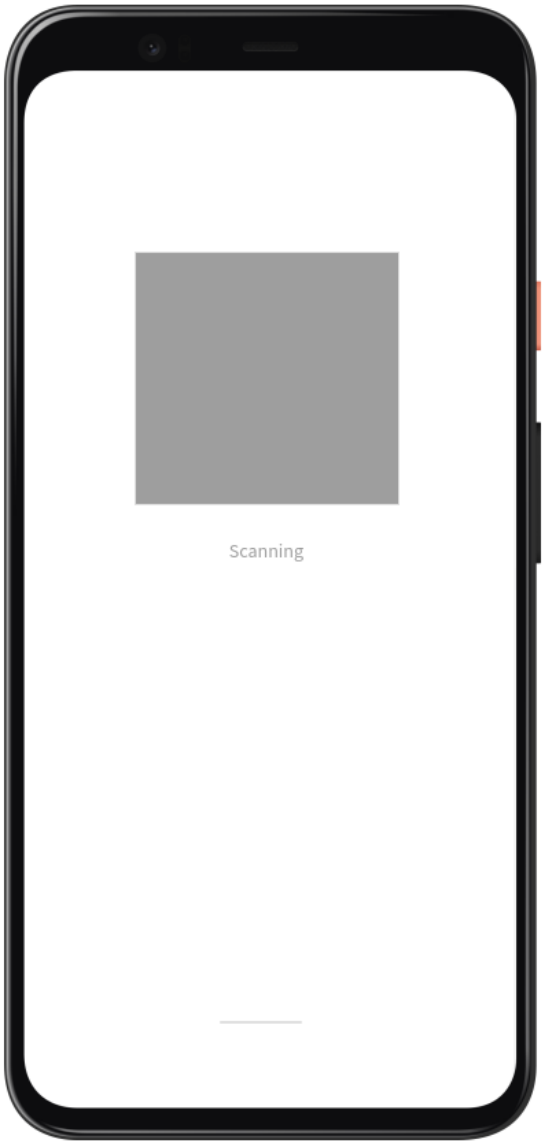


Figure 5.1.5 Wireframe Scanning

Few Samples screen shots of how the products are placed in front of the camera
Are presented below.

Figure 5.1.6 Captured image1

Figure 5.1.7 Captured image2

When the product is scanned the number below the barcode is fetched.

The number fetched is used as a primary key and is searched through the database and the details matched with fetched number values are presented in the format as below screens.

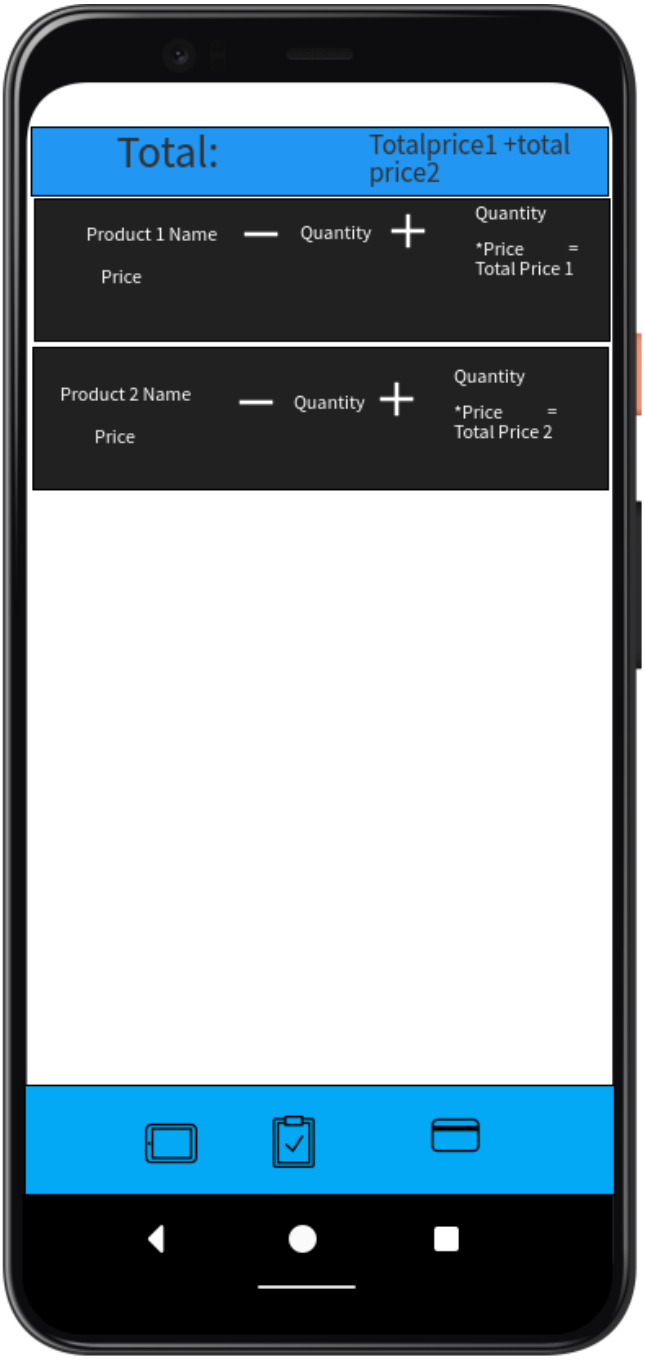


Figure 5.1.8 Wireframe of product details

When the details are fetched they are represented as above in a card layout with name and price of the product on the left. The quantity buttons to increase and decrease the quantity .The default quantity is 1 and the total price of the product with the respective quantity the user chose on the right. The total value of the price of products is displayed on the top.

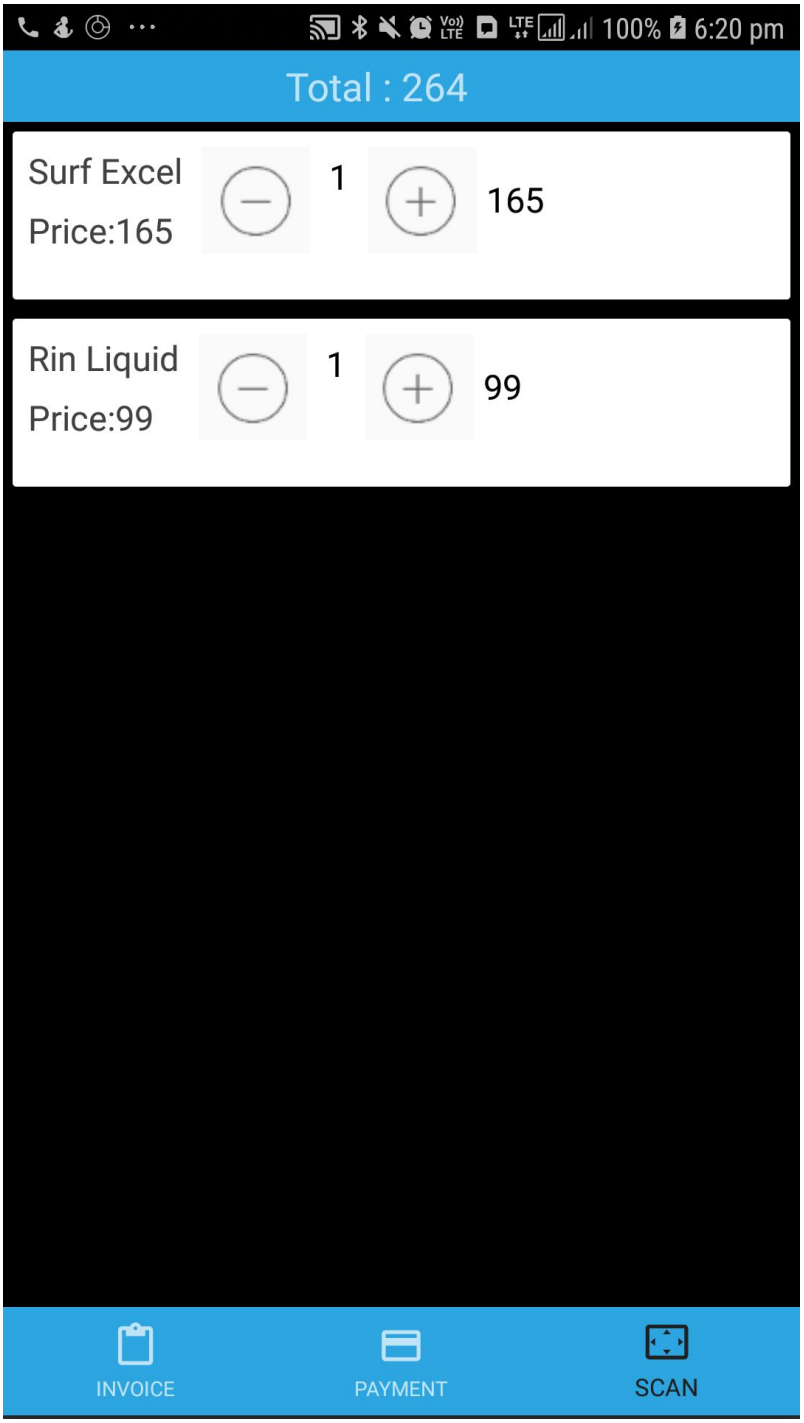


Figure 5.1.9 Output of products

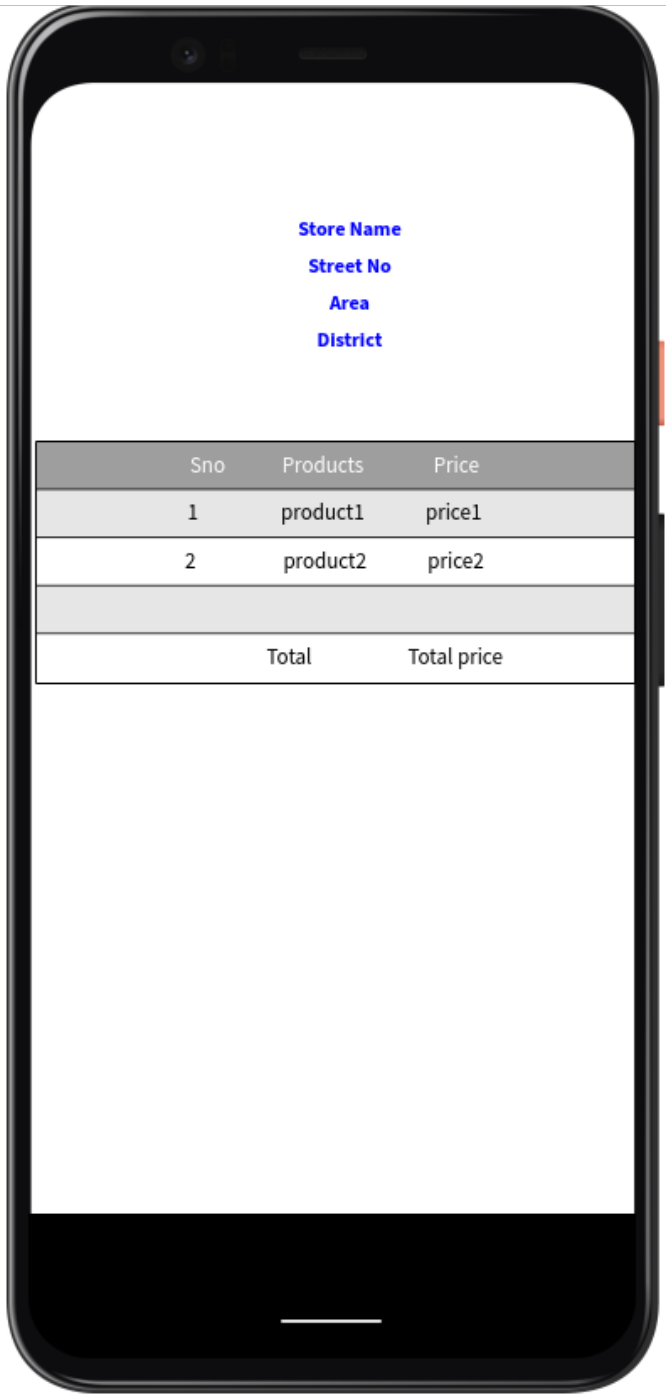


Figure 5.1.10 Wireframe of Invoice

As there are three menus on the home when the user is done with his or her scanning of products and chooses the menu of Invoice. The Screen of format above is displayed with the store Name and address where the user is purchasing and table of products purchased by the user.

The columns of the table are as follows:

- Sno
- Product Name
- Quantity
- Price

The Sno is Serial Number used to represent the number of total products purchased by the user.

The product Name is the name of the product fetched from the database.

The quantity is the total Quantity of that respective product purchased by the user.

The price is the cost of the product fetched from the database.

Each row of the table represents a single product and its respective column values.

The last row of the table is grand total of the total cost of the product purchased.

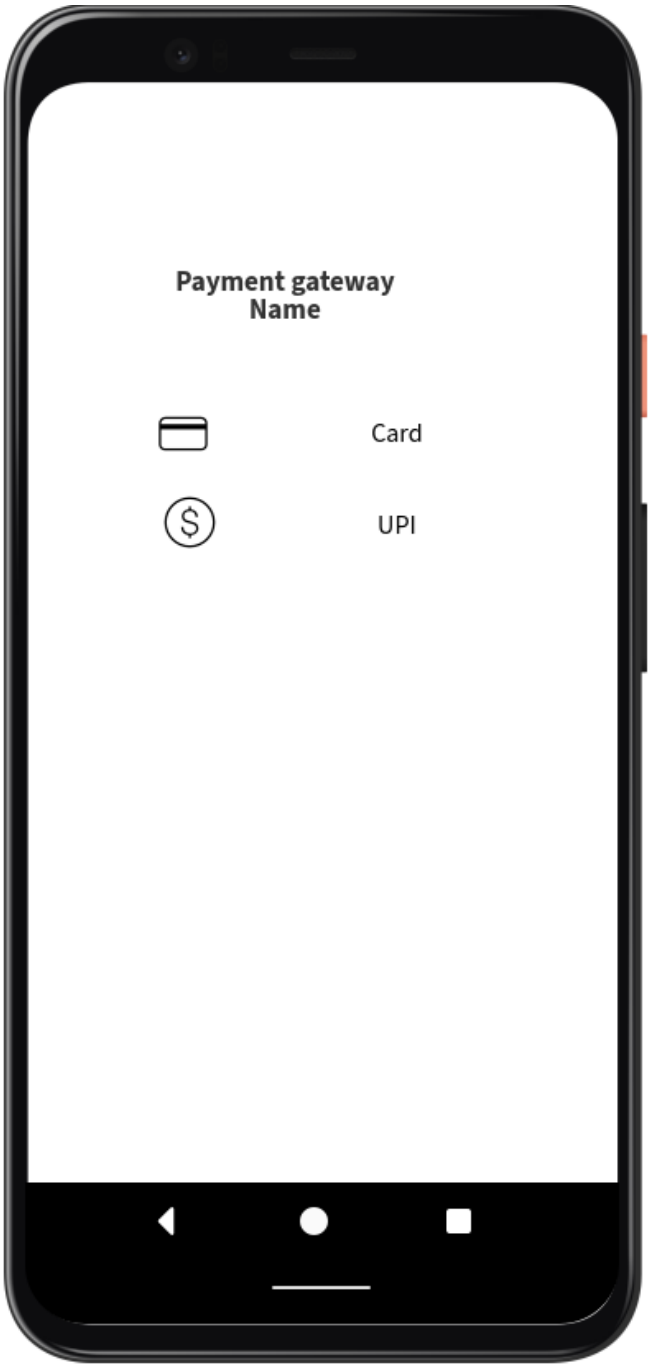


Figure 5.1.11 Wireframe payment gateway

As shown in the home screen there are three options when user is done with checking his bill through invoice option he / she navigated to the third menu item
Pay which provides:

- Cash payment
- Cashless payment

When user chooses to do cashless payment he/she is navigated to the above screen where he is provided various online payment options like

- Net banking
- Phone Pay
- Google Pay
- Credit Card
- Debit Card etc

When the users choose the option of Debit card payment he is navigated to below Screen.

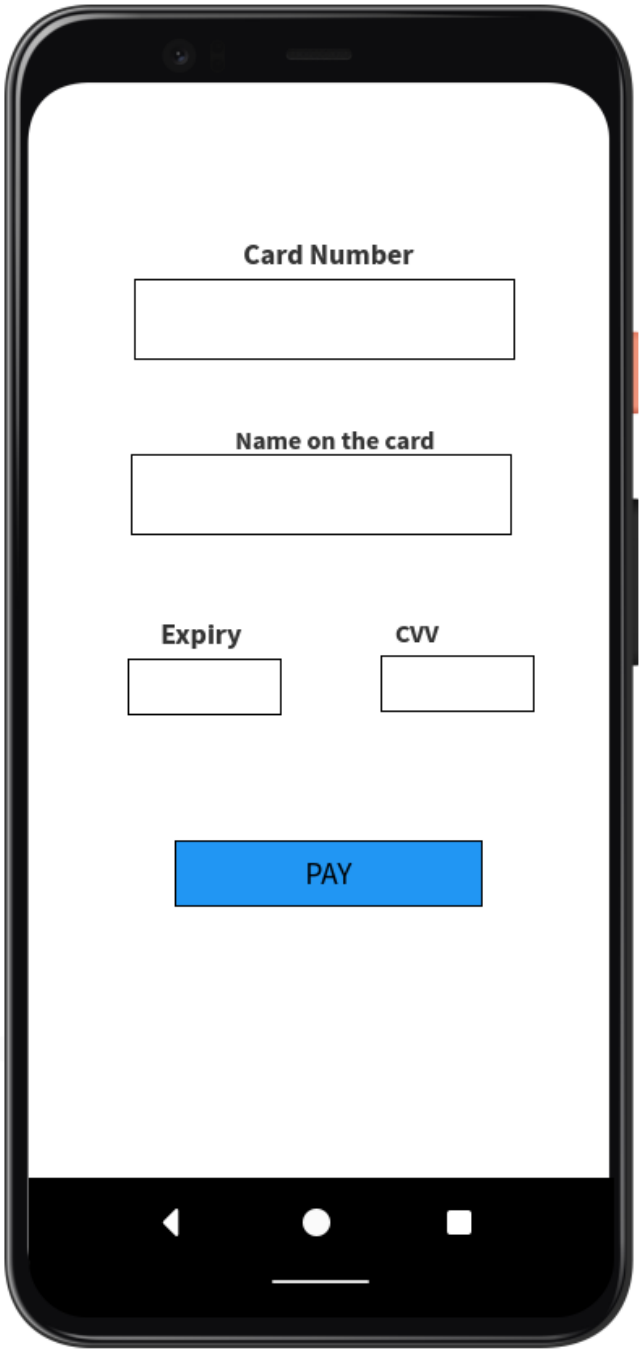


Figure 5.1.12 Wireframe card payment

The user needs to provide credible details in the above screen about his card such as:

- Name on the card
- Cvv
- Expiry

If the incredible details are provided by user then it throes an error and the transaction considered as invalid and will be cancelled. If the user provides credible details then he navigated to the screen below.

Where an OTP is sent to user's registered mobile number to ensure security.

The below page ha the following detail of:

- The name of the gateway used for payment
- Payment details
- OTP

The name of the gateway used is RAZOR PAY in the app.

The payment details would be the payment id, the total amount, the name of the user etc.

The OTP is the code received in the user's mobile through message.

After submission of the code the gateway displays whether or not the payment is successful.

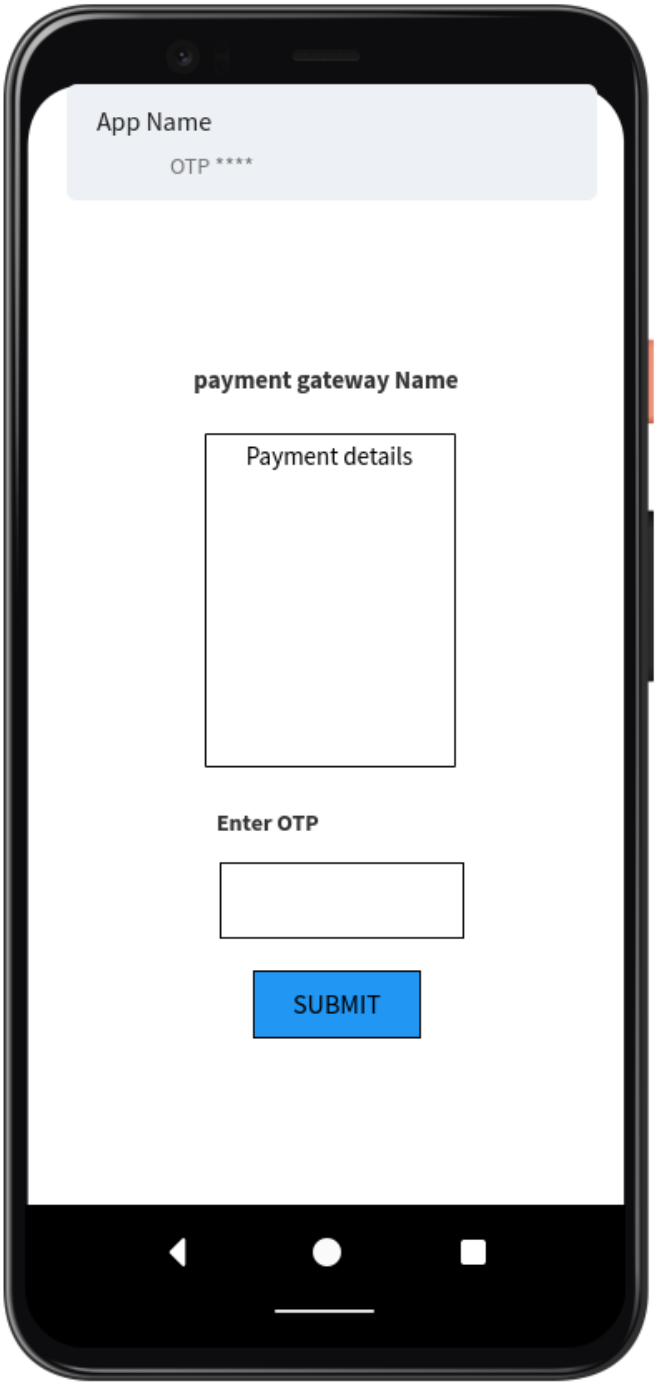


Figure 5.1.13 Wireframe payment OTP

5.2 Overview of Database Control App

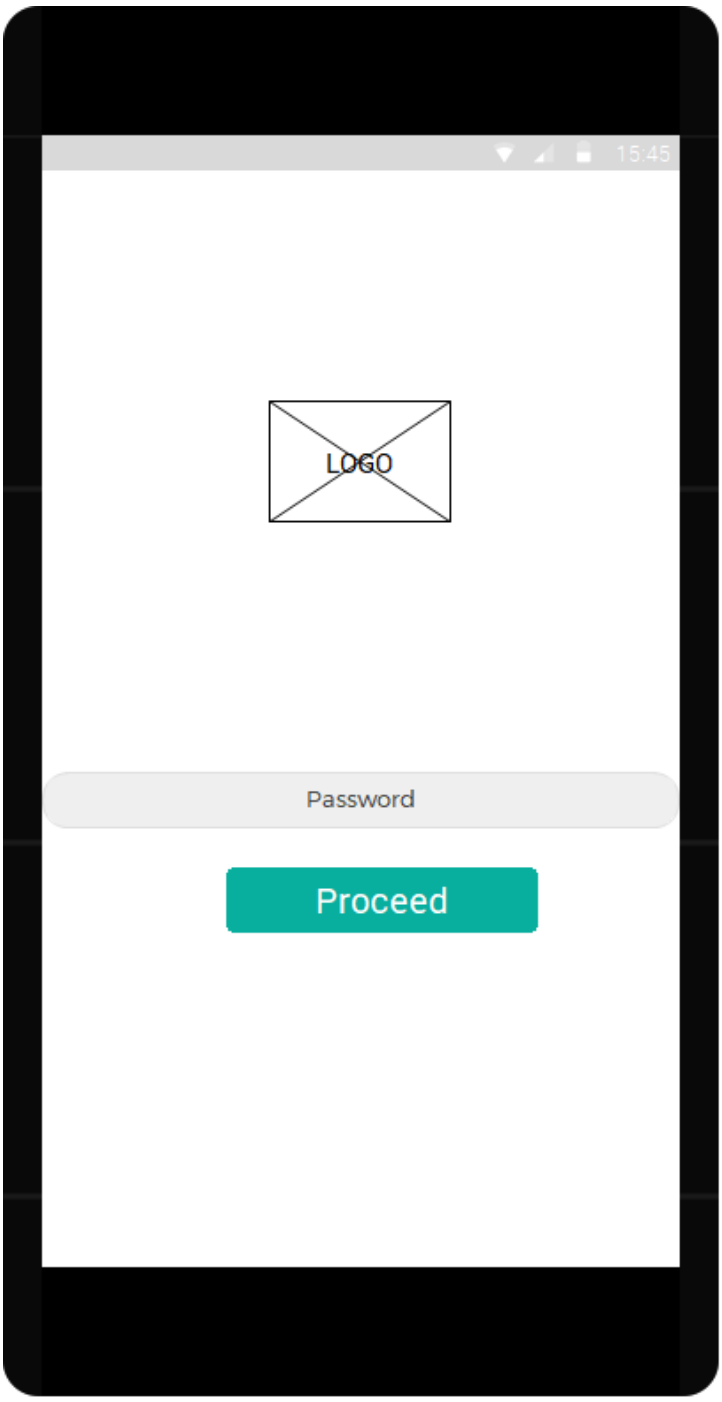


Figure 5.2.1 Wireframe Database Entry Screen

The First screen of Database Control App is the password entry Screen Where user of this app needs to enter the pass code in order to login to the app. It is made such way so that no one other than the store managers and employees of the store can access the database. Specifically, in-order to prevent user from gaining access to the database and any other malicious user trying to gain access to database.

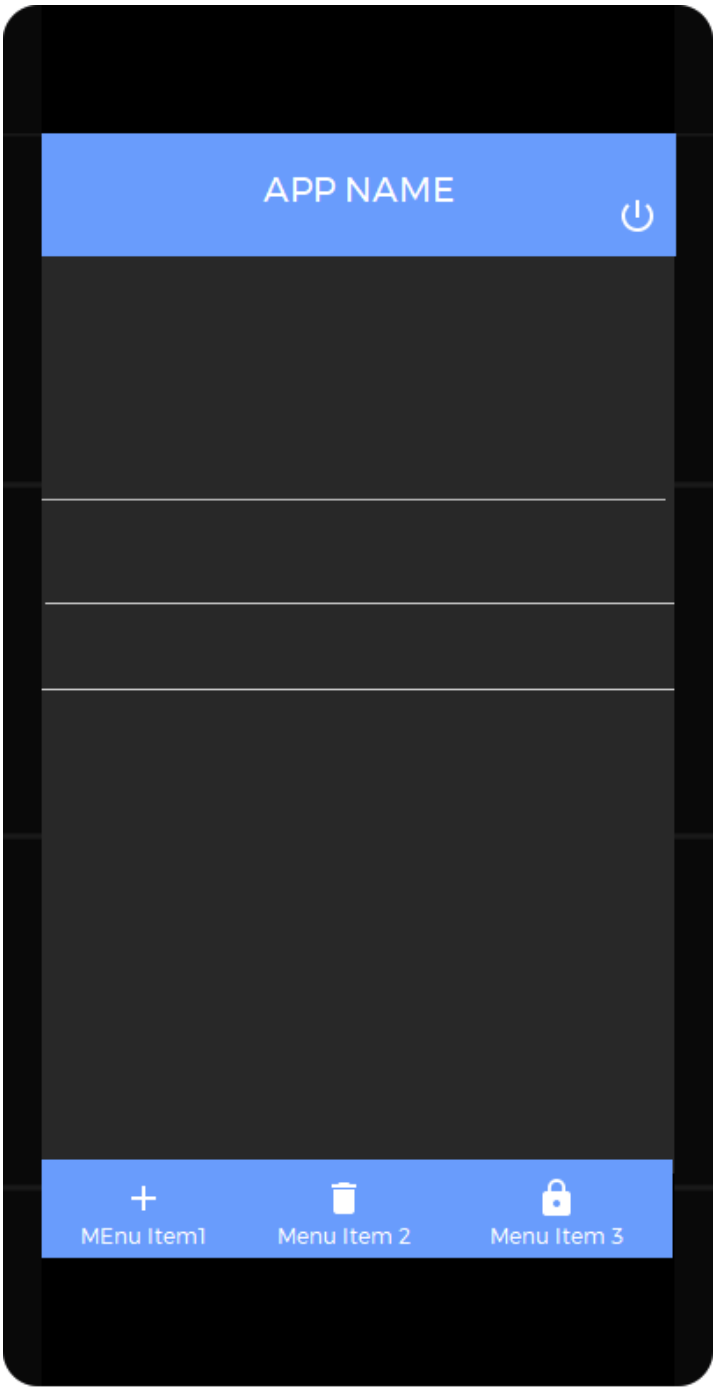


Figure 5.2.2 Wireframe Database Home Screen

Once the user had entered the correct password he would be navigated to the screen above which is the home Screen of the application. Which has bottom navigation with three menu items:

- Add
- Delete
- Change Password

The screen also has three input felids which are represented as three lines. The three input felids are as follows:

- Barcode of the product
- Name of the product
- Price of the product

There are no more screen navigations all the actions of the menu items of the navigation bar takes place only in one Screen through alert messages as follows.

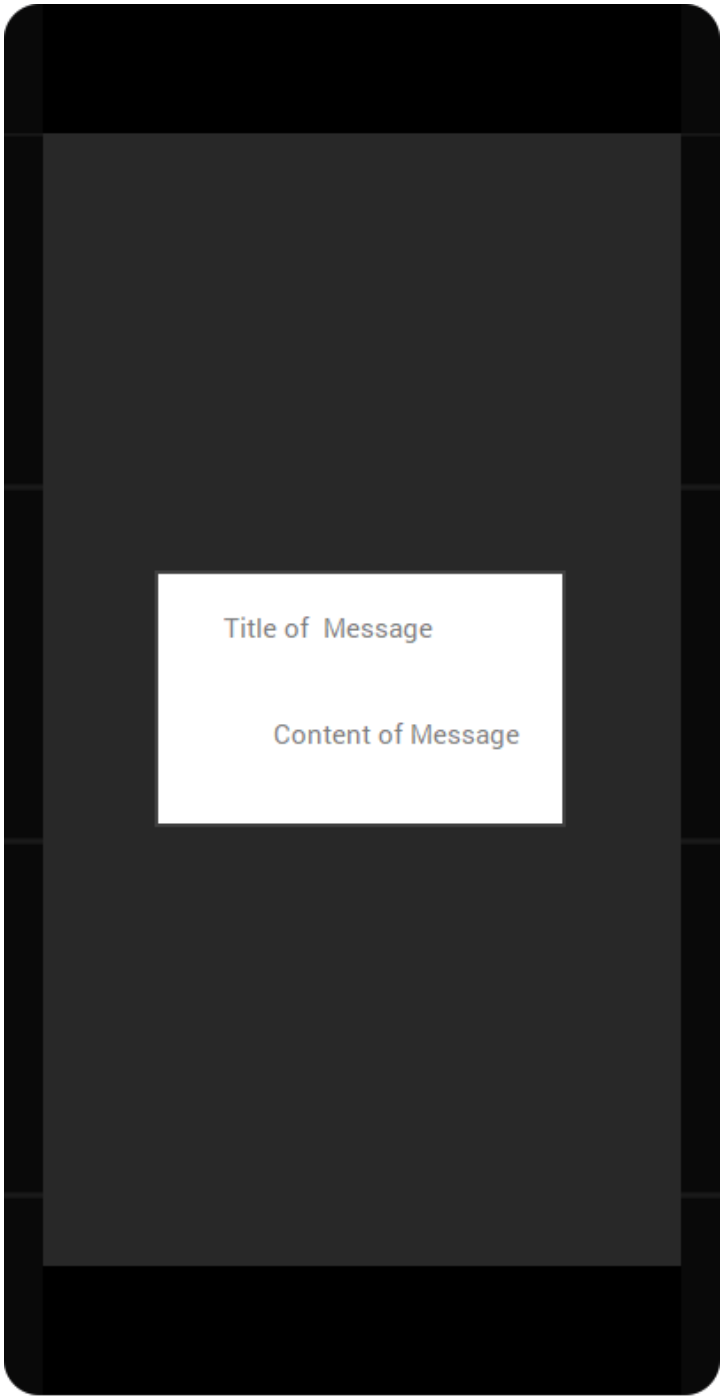


Figure 5.2.3 Wireframe Database Alert Screen

When the user chooses the menu option add:

The functionality of add is to add product to the database. The user can do it by entering the input felids

- Barcode
- Name of Product

- Price of product

When the product is added to the database an alert message similar to above is displayed .when title “Add” and message “product added to the database”.

Similar is the case to second menu option delete:

When user wants to delete the product he enters the product of an existing barcode in the barcode input flied and taps on the menu item delete.

Then an alert similar to above is displayed with title” Delete” and message

“Product Deleted”.

There is a dual functionality of add menu item if an existing barcode is entered in the barcode flied. Then an alert screen as below is displayed to carry out Update functionality.

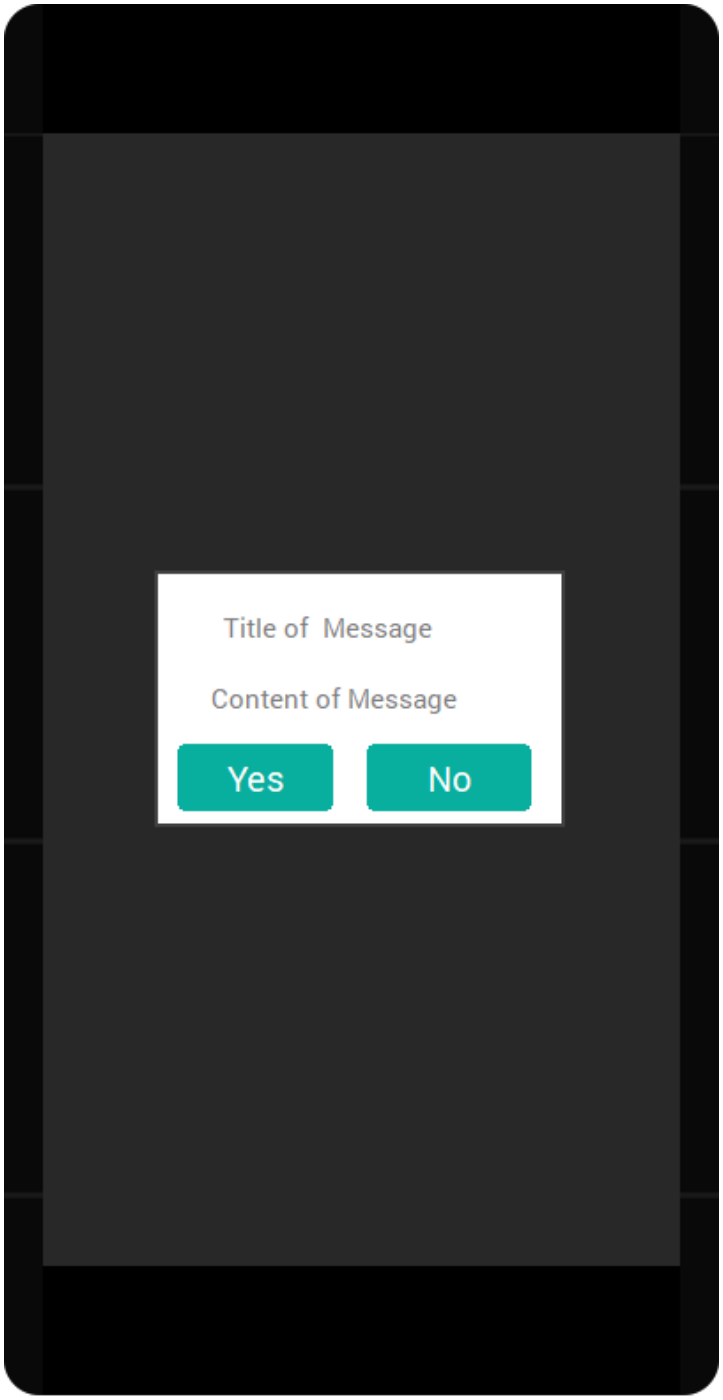


Figure 5.2.4 Wireframe Database Update Alert Screen

When user wants to update the product details such as price or name he can enter the details in the three input fields. The app in the background identifies that the details entered are the details of an existing product and displays an alert message as above with title “UPDATE”

Message “Do you want to Update?” and two buttons

- Yes

- No

  To carry out the operation if the user clicks Yes the details of the product are updated and if the users clicks on No then it is considered as cancel the details of the product as same as before .

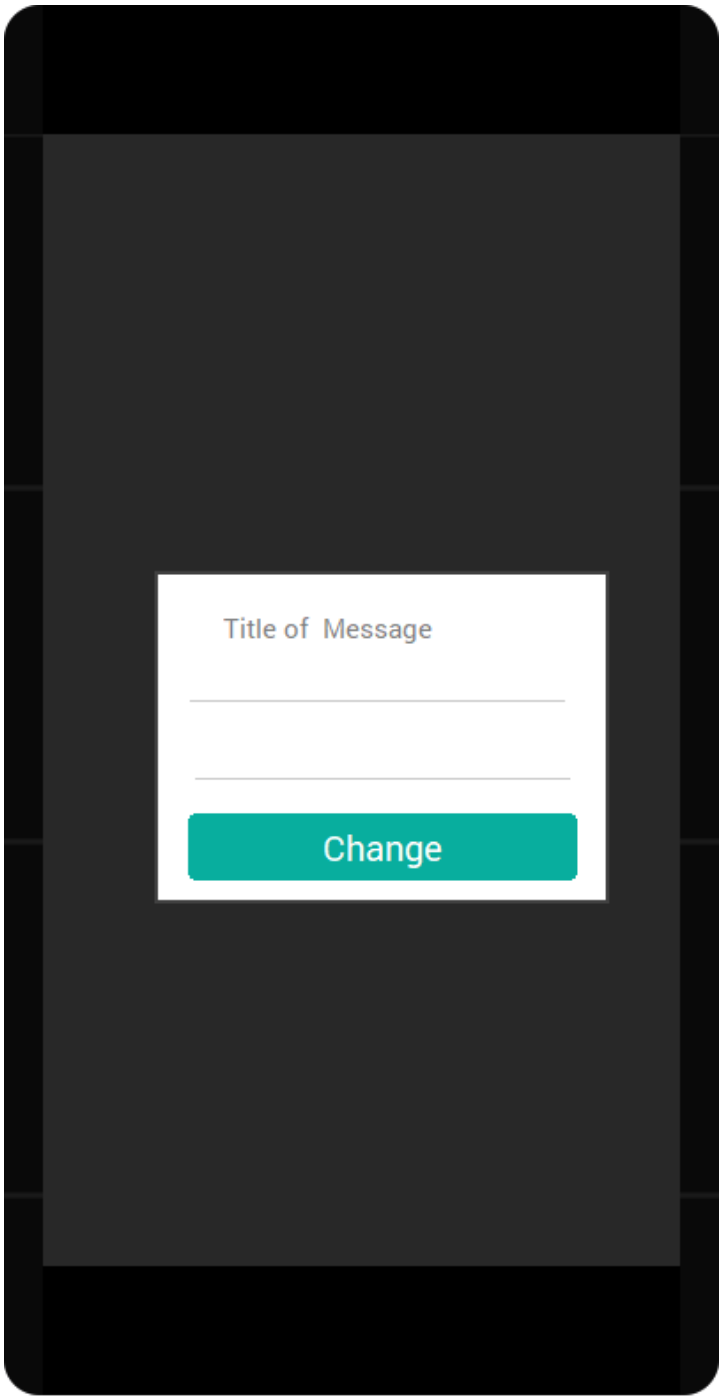


Figure 5.2.5 Wireframe Database Password Alert Screen

When users chooses the third menu item Password change of the navigation bar the above alert Screen is displayed with two input felids and one button.

The two input felids are:

- Old password
- New Password

For Safety purpose when user prefers to change the password of the app .He/she can change the password by giving input of two felids i.e. the old password of the app and the new password to which they would like to change it to.

When the old password input is given correctly the functionality is invoked and the password gets updated.

6. Testing

6.1 Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 Testing Methodologies:

The following are the testing Methodologies:

- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing
- Validation Testing

6.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing paths are tested for the expected results. All error handling paths are also tested.

Benefits of Unit Testing:

- Reduces Defects in the newly developed features or reduces bugs when changing the existing functionality.
- Reduces Cost of Testing as defects are captured in very early phase.
- Improves design and allows better refactoring of code.
- Unit Tests, when integrated with build gives the quality of the build as well.

Unit Testing Techniques:

- **Black Box Testing** - Black-box testing is a method of software testing that examines the functionality of an application based on the specifications. It is also known as Specifications based testing. Independent Testing Team usually performs this type of testing during the software testing life cycle.
- **White Box Testing –** White box testing is a testing technique that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.
- **Gray Box Testing -** Grey Box testing is testing technique performed with limited information about the internal functionality of the system. Grey Box testers have access to the detailed design documents along with information about requirements.

6.2.2 Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

- Big-Bang Integration
- Top Down Integration
- Bottom Up Integration
- Hybrid Integration

1. Big-Bang Integration

Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units

2. Top-Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module are incorporated into the structure in either a depth first or breadth first manner.

3. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into cluster that perform a specific software sub-function.
- The cluster is tested.

4. Hybrid Integration

Integration Testing is a phase in software testing in which standalone modules are combined and tested as a single entity. During that phase, the interface and the communication between each one of those modules are tested. There are two popular approaches for Integration testing which is Top down Integration Testing and Bottom up Integration Testing.

In Hybrid Integration Testing, we exploit the advantages of Top-down and Bottom-up approaches. As the name suggests, we make use of both the Integration techniques.

Features of Integration Testing:

The prime features of Integration Testing are:

- **Done After Unit Testing is Complete:** The most important requisite of Integration Testing is that each module should have been successfully unit tested before the integration can take place. The idea is to ensure that every unit is correctly performing its designated task. Once the performance of each unit is validated it is combined with the next one. At each level of integration, the software is continually tested to make it as foolproof as possible.

- **Integration as Per Test Plan:** The integration of every module takes place as described in the test plan. This is done to reduce chaos and have a definitive path that needs to be followed in order to carry out testing effectively.

- **Checks Functionality:** The aim behind Software Integration Testing is to guarantee a product that efficiently delivers on all that it promises. As such, at every level of assimilation, the software is rigorously tested to detect and eradicate all functional flaws.

6.2.3 System Testing

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.

System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

System Test Types

- Functionality
- Inter-Operability
- Performance
- Scalability
- Stress
- Load and Stability
- Reliability
- Regression
- Regulatory &Compliance

6.2.4 User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under considerations is tested for user acceptance by constantly keeping in touch with the prospective system user at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

For industry such as medicine or aviation industry, contract and regulatory compliance testing and operational acceptance testing is also carried out as part of user acceptance testing.

UAT is context dependent and the UAT plans are prepared based on the requirements and NOT mandatory to execute all kinds of user acceptance tests and even coordinated and contributed by testing team.

Acceptance Criteria

Acceptance criteria are defined on the basis of the following attributes:

- Functional Correctness and Completeness
- Data Integrity
- Data Conversion
- Usability
- Performance
- Timeliness
- Confidentiality and Availability
- Installability and Upgradability
- Scalability
- Documentation

The acceptance test activities are designed to reach at one of the conclusions:

1. Accept the system as delivered
2. Accept the system after the requested modifications have been made
3. Do not accept the system

Acceptance Test Report - Attributes

The Acceptance test Report has the following attributes:

- Report Identifier
- Summary of Results
- Variations
- Recommendations
- Summary of To-DO List
- Approval Decision

6.2.5 Validation Testing

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

Validation Testing - Workflow:

Validation testing can be best demonstrated using V-Model. The Software/product under test is evaluated during this type of testing.

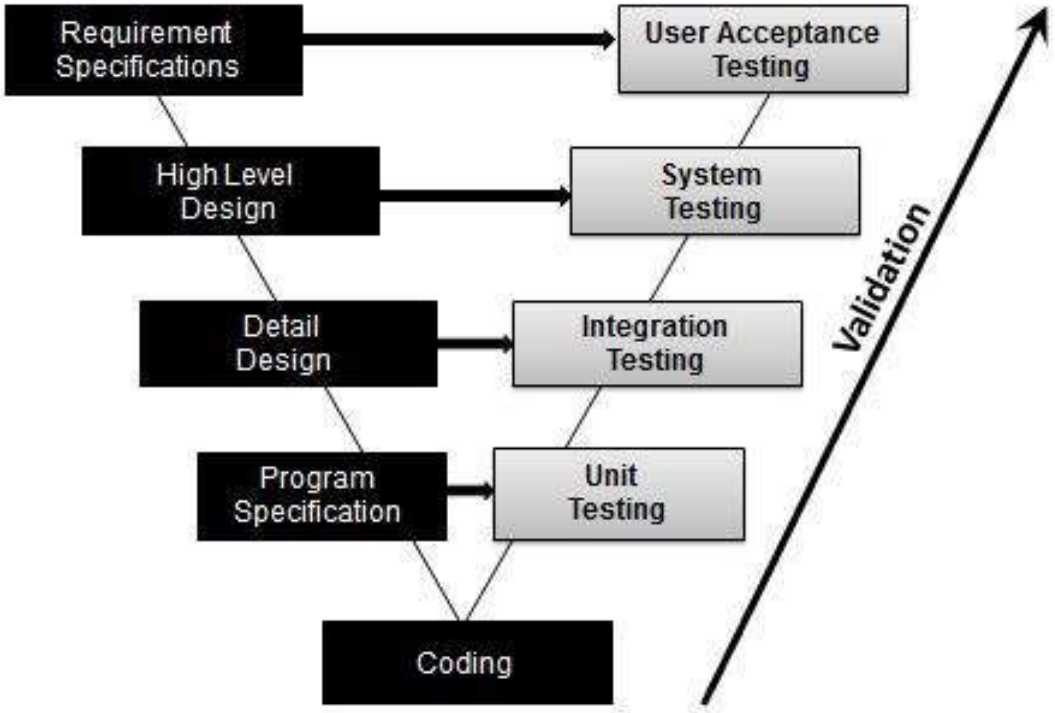


Figure 6.2.5.1 Validation Testing Work Flow

Advantages of Validation Testing:

1. During verification if some defects are missed then during validation process it can be caught as failures.
2. If during verification some specification is misunderstood and development had happened then during validation process while executing that functionality the difference between the actual result and expected result can be understood.
3. Validation is done during testing like feature testing, integration testing, system testing, load testing, compatibility testing, stress testing, etc.
4. Validation helps in building the right product as per the customer's requirement and helps in satisfying their needs.

7. Observations and Results

7.1 Screenshots of Skip the Queue App

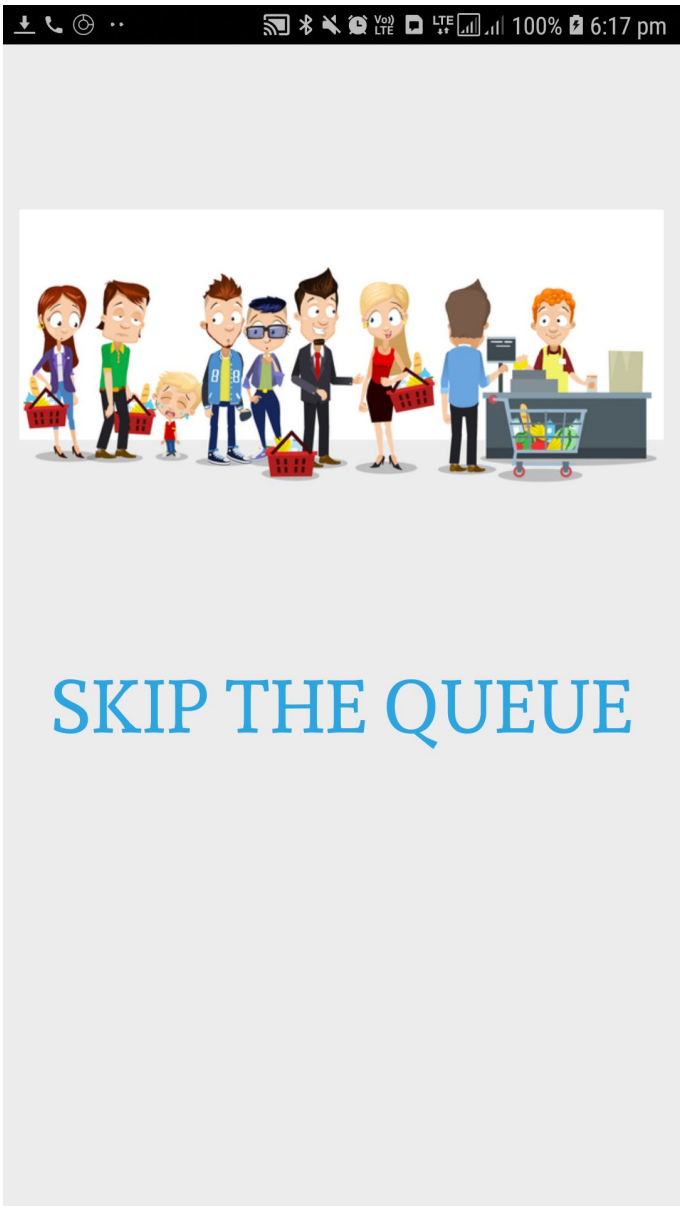


Figure 7.1.1 Splash Screen

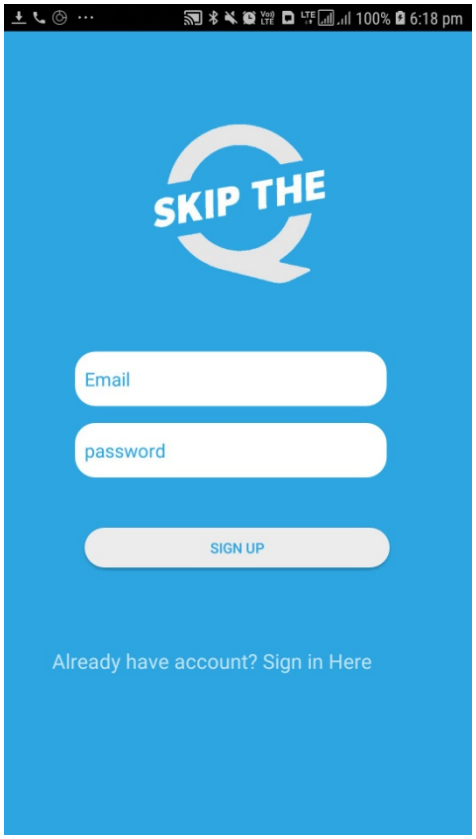


Figure 7.1.2 Signup Screen

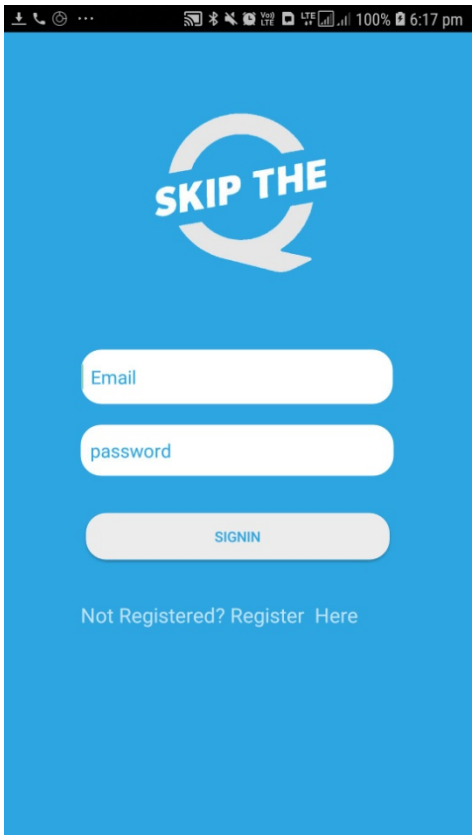


Figure 7.1.3 Sign in Screen

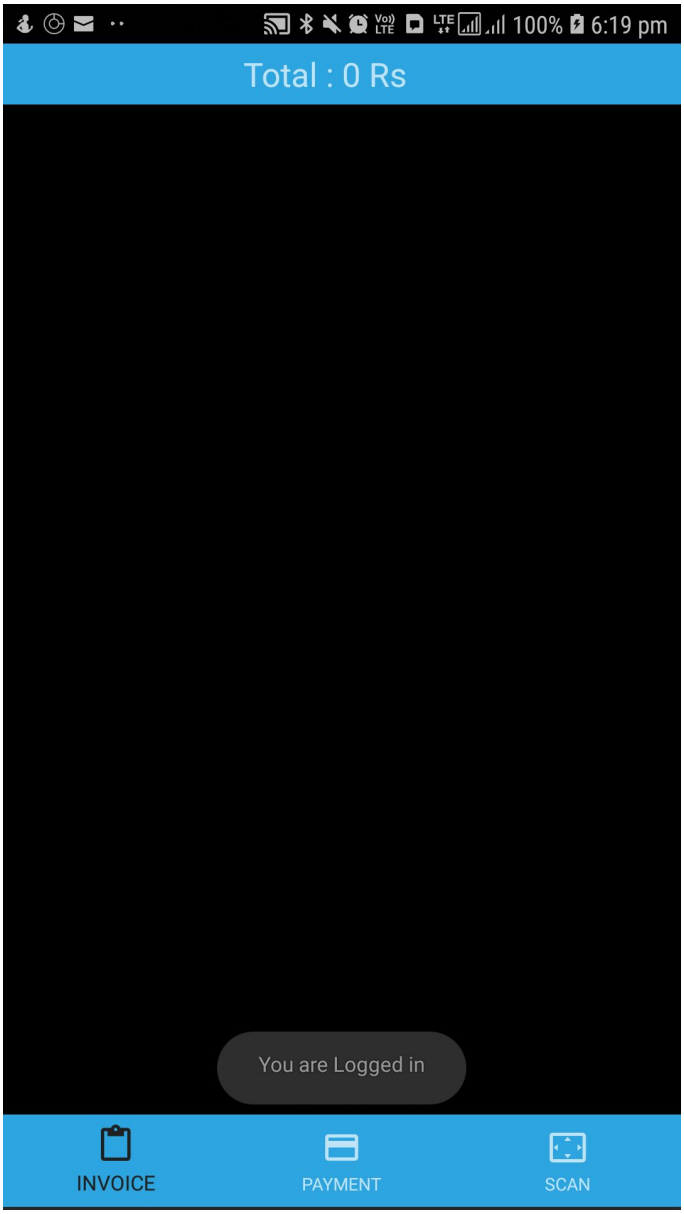


Figure 7.1.4 Home Screen

Figure 7.1.5 Scanned Product

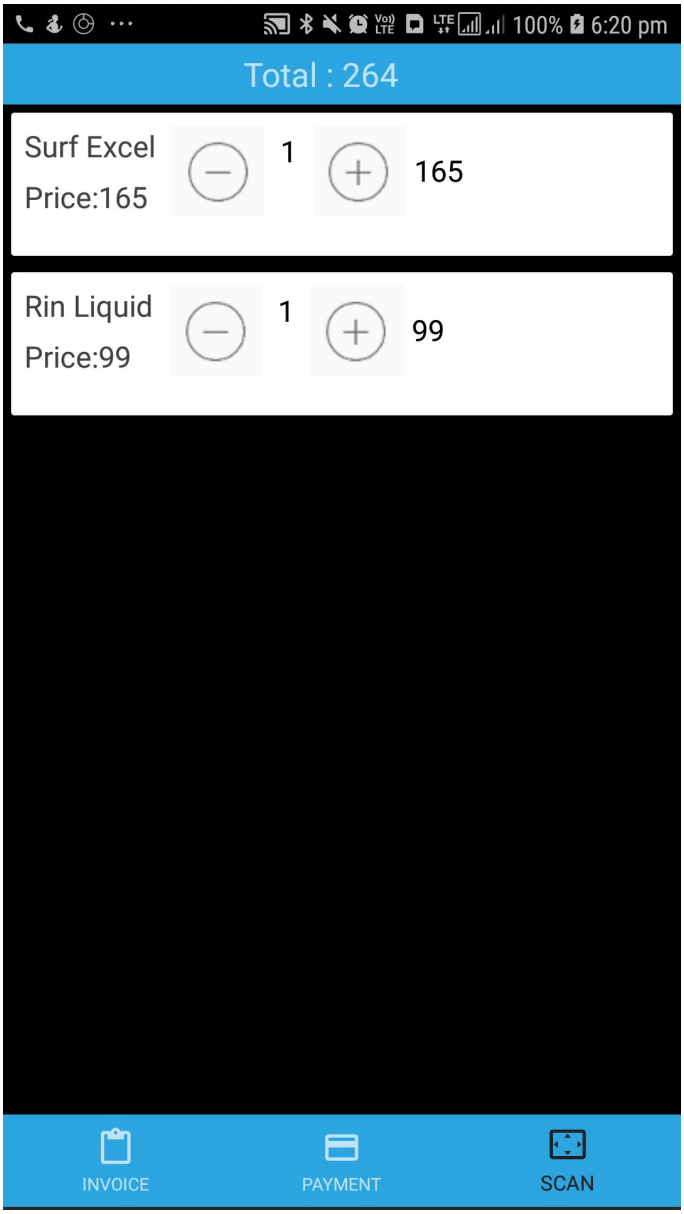


Figure 7.1.6 Product details

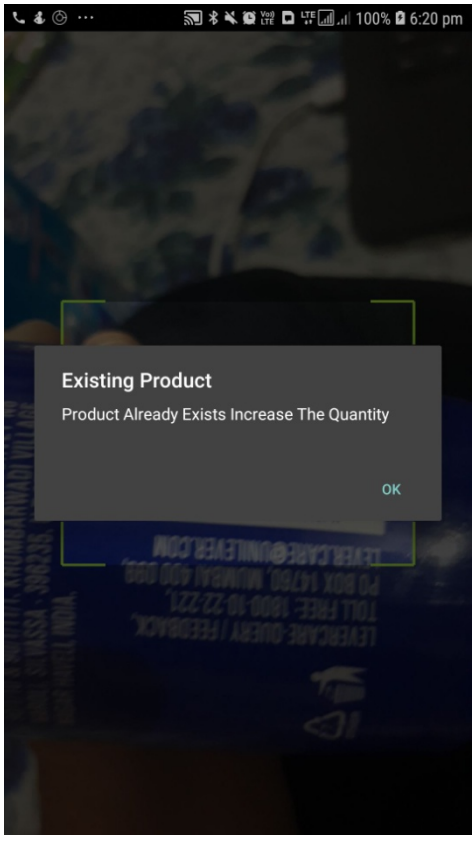


Figure 7.1.7 Product validation1

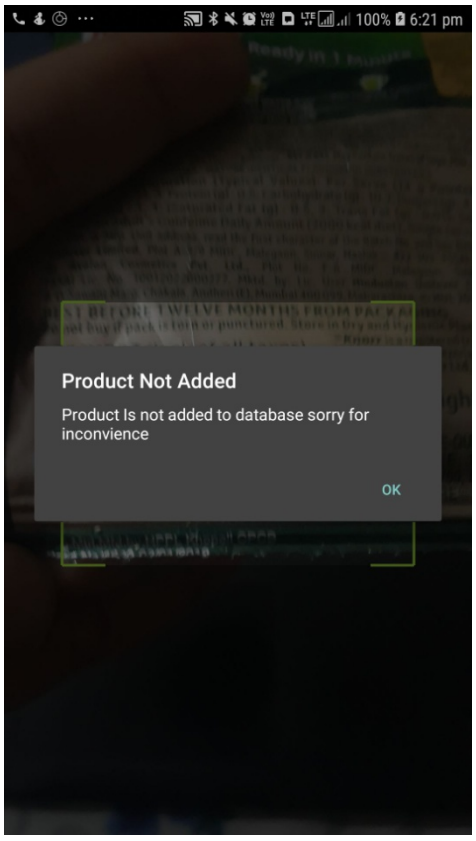


Figure 7.1.8 Product validation2

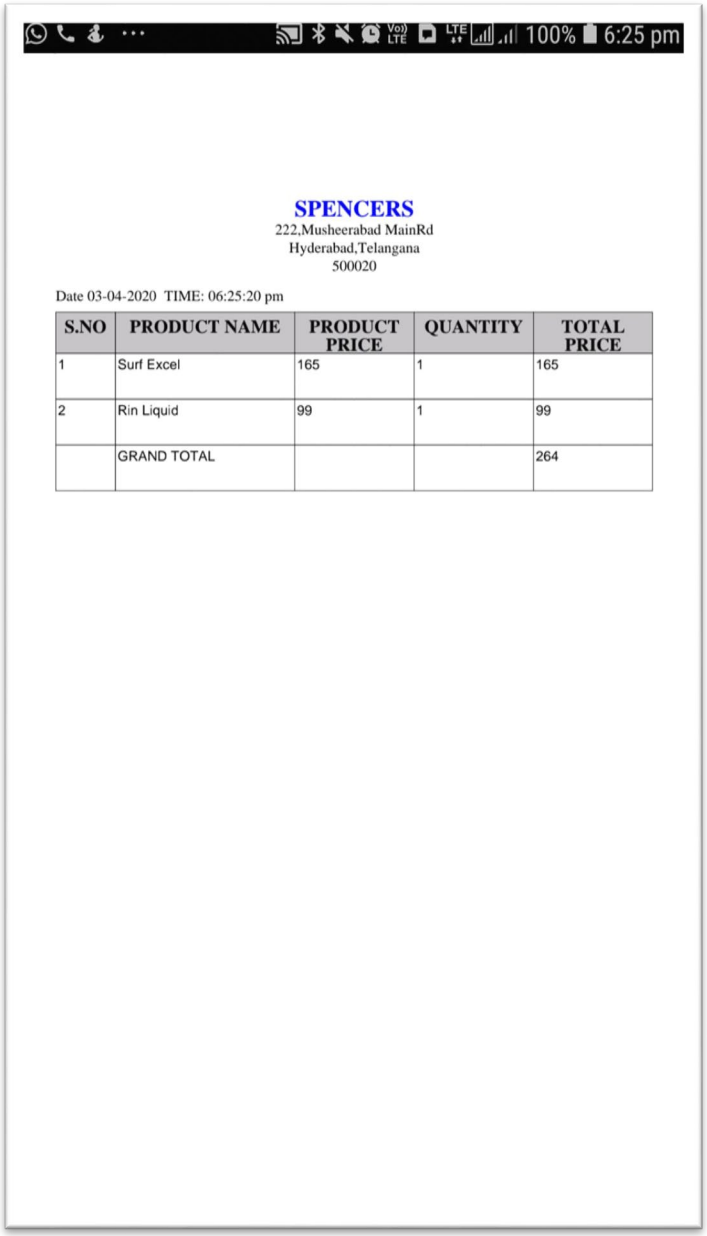


Figure 7.1.9 Invoice

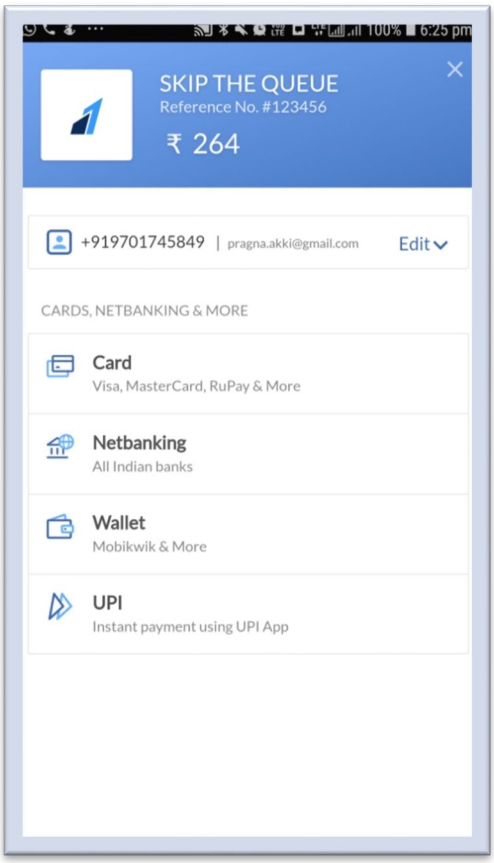


Figure 7.1.10 Payment

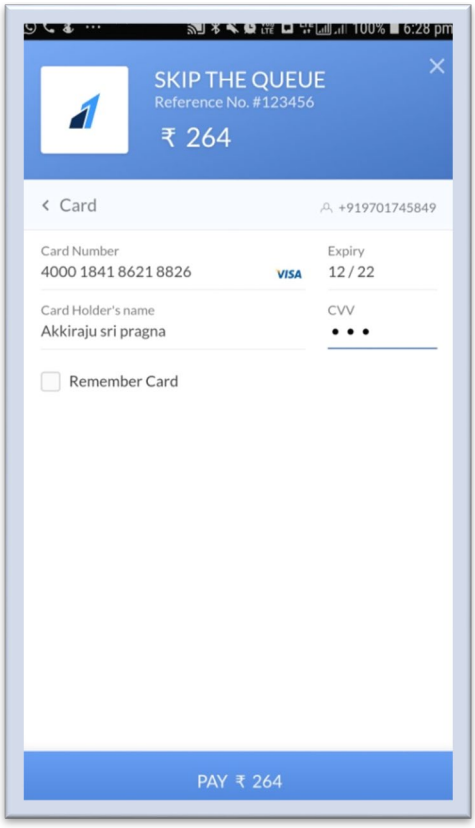


Figure 7.1.11 Card payment

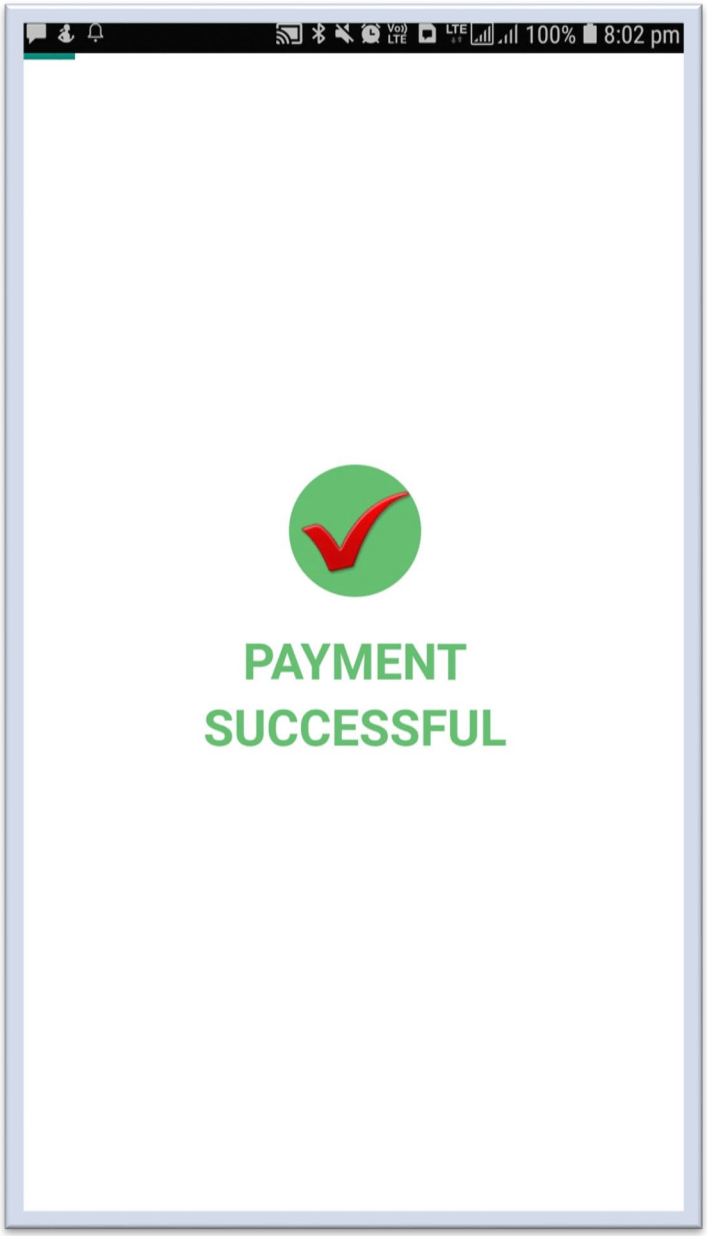


Figure 7.1.12 Payment Success

7.2 Screenshots of Database Control App

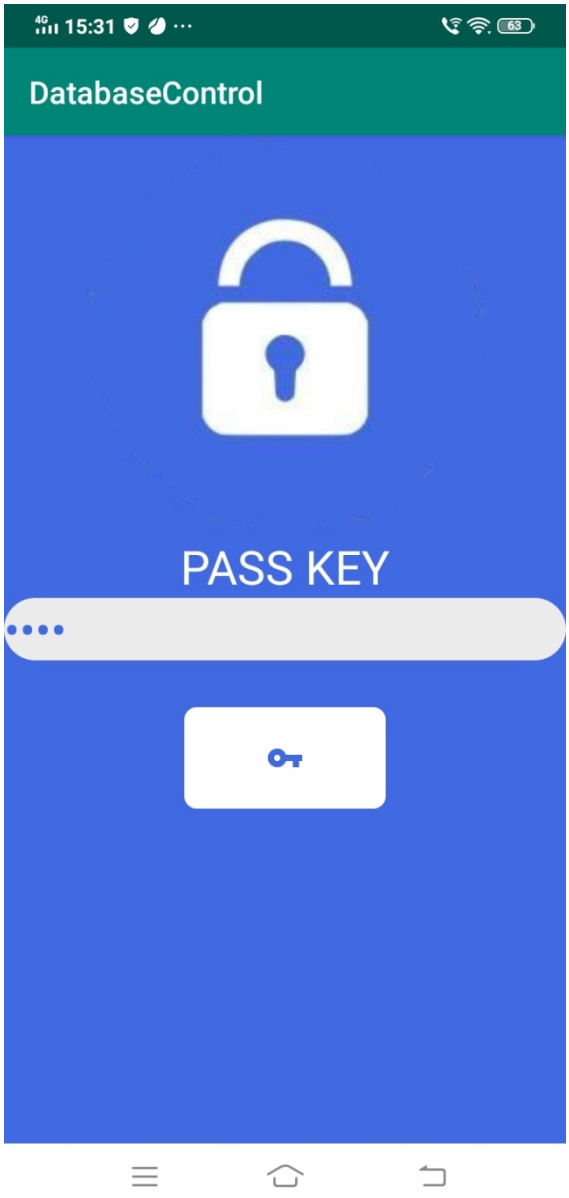


Figure 7.2.1 Pass Key

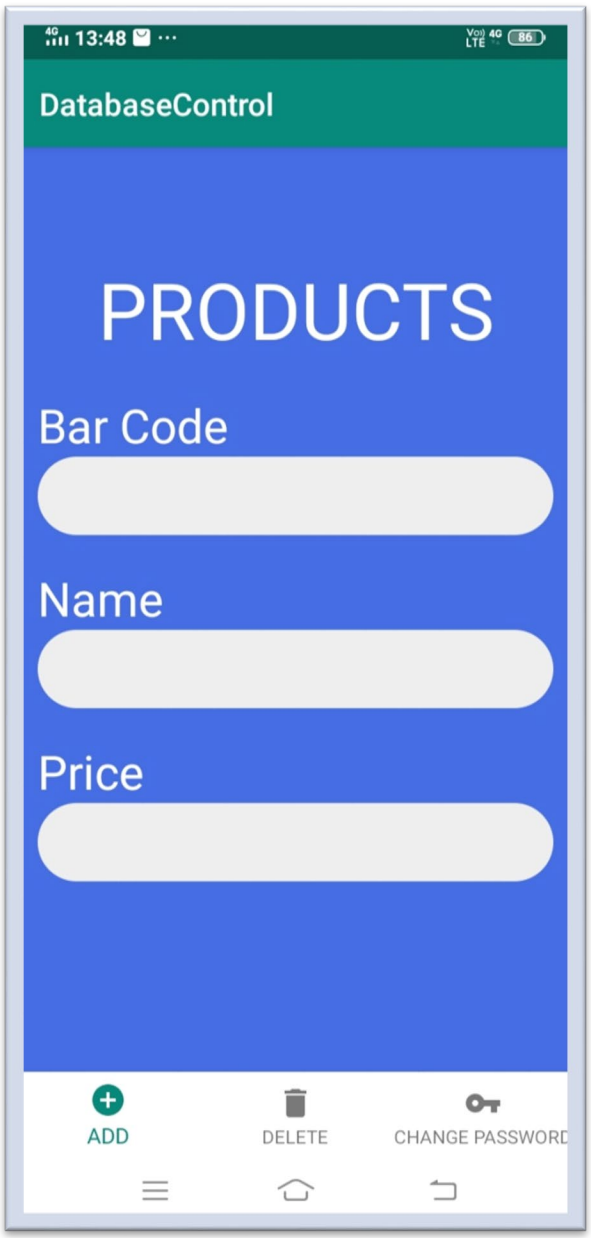


Figure 7.2.2 Database Home Screen

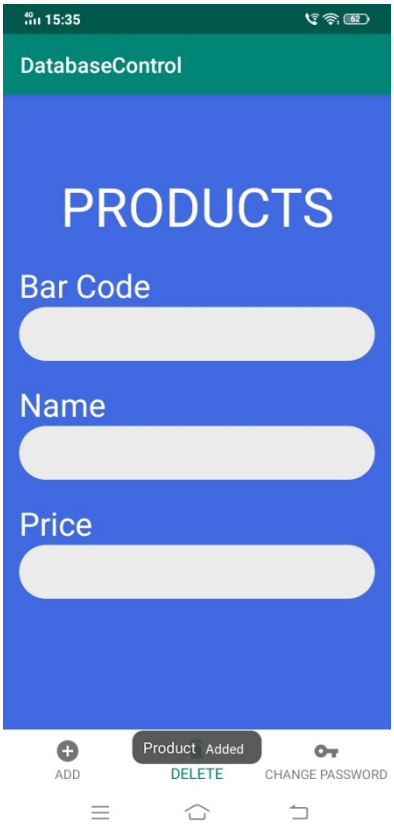


Figure 7.2.3 Product Added

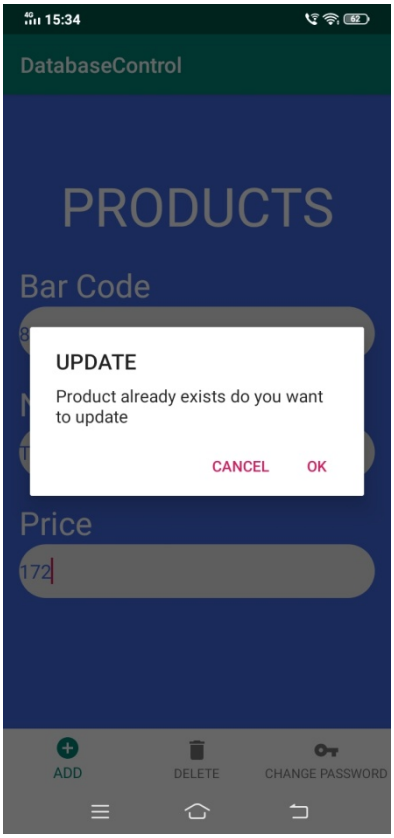


Figure 7.2.4 Update Alert

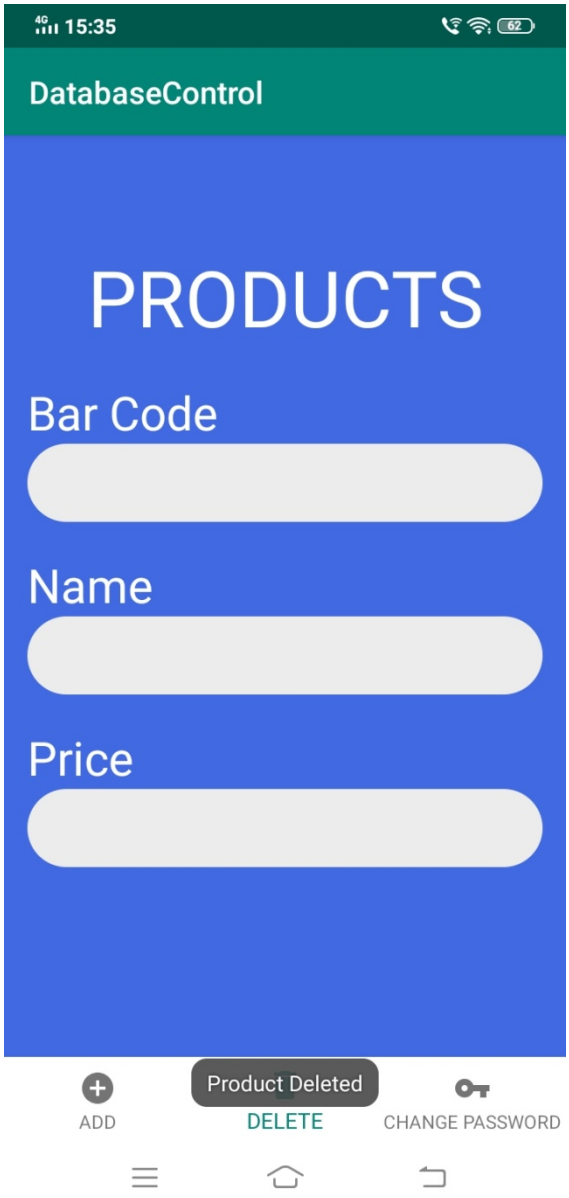


Figure 7.2.5 Product Deleted

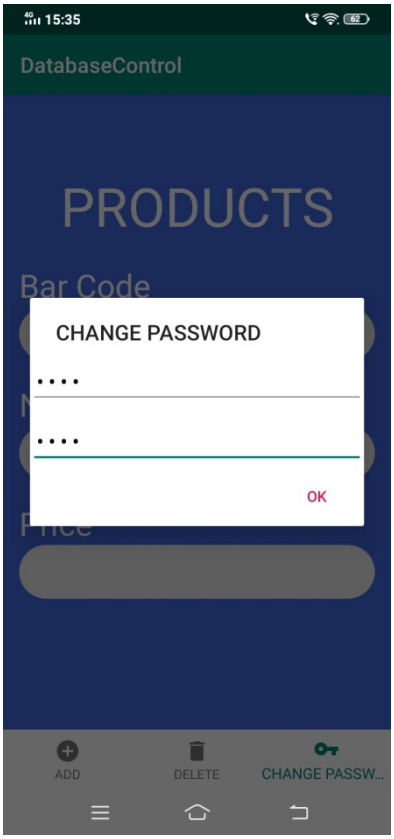


Figure 7.2.6 Password Alert

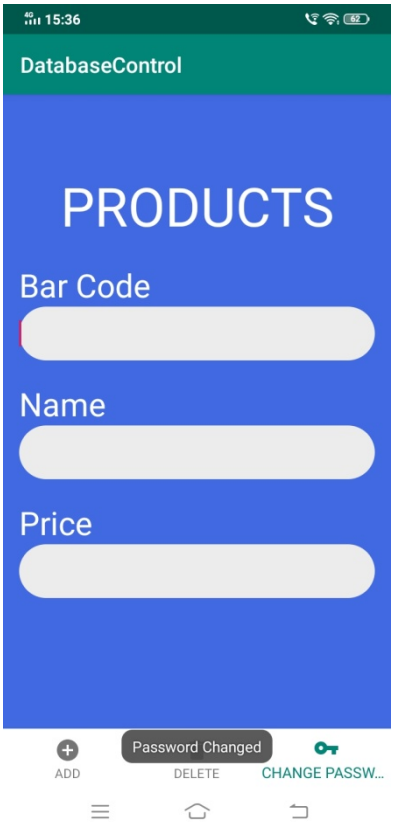


Figure 7.2.7 Password Changed

Conclusion

The app will be an advantage for shopping as it would make shopping easier. The idea of this app is the customer need not to stand in a queue to pay the bill. In the app the billing is done on the customer end by scanning the products through the camera of their phone, and an attachment of the bill is stored in customer's device for later review. The app saves a lot of time and energy of the customer by preventing them standing in long queues to get few products billed. It also makes shopping experience more joyful.

Future Scope

Future scope in the application initiated here can be adding additional functionalities to provide solutions to other problems faced by customers during shopping such as finding the location of their desired product in the mall, Product suggestions guiding customer to purchase the correct product which meets their requirements.
Sales and discount suggestions to save customer's money through pop-ups messages.

The present prevalent situation for corona virus and covid19 there is a huge scope of innovation of application which ensures safety. This application plays a role in safety, due to necessity of goods many people get crowded near stores which leads to dangerous situation of spread of virus corona, to ensure safety stores have started billing for 5 customers and letting in next 5 after first 5 billing is done. This is a time-consuming process but the goal of this application is to skip the queues. This application can make the process little less time consuming and ensures safety.

Hence, there is a chance of wide usage of this application in the near future.

References

1. Swati Zope, Prof, Maruti Limkar, “RFID based Bill Generation and payment through Mobile”, International Journal of Computer Science and network (IJCSN), Volume 1, Isuue3, June 2012.
2. Mohit Kumar,Jaspreet Singh,Anju,Varu Sanduja(2015) “Smart trolley with instant billing to ease Queues at shopping malls using Aum7 IPC 2148:a review” International Journal of Advanced Reasearch in Computer and Communication Engineering(vol.4,Issue &August 2015).
3. Dhavale Shraddha D,Dhokane Trupti J,Shinde Priyanka S,IOT Based intelligent Trolley for Shopping Mall,IJEDR,2016
4. https://www.lsretail.com/blog/self-checkout-technology
5. https://www.explainthatstuff.com/barcodescanners.html
6. https://edition.cnn.com/2019/12/23/tech/smart-shopping-cart/index.html
7. https://www.theglobeandmail.com/report-on-business/small-business/sb-growth/new-app-lets-moviegoers-skip-the-line-for-popcorn/article29570658/
8. https://fastorder-app.com/#//
9. https://books.goalkicker.com/AndroidBook/
10. https://bookauthority.org/books/new-android-studio-books

APPENDIX A

Sample code

Product Details Class

```
public class Products_Details extends AppCompatActivity {
EditText Name, Price, Barcode;
EditText Prevpass,Newpass;
int prevpass2,newpass2;

DatabaseReference reference,password;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_products__details);
Name = (EditText) findViewById(R.id.name);
Price = (EditText) findViewById(R.id.price);
Barcode = (EditText) findViewById(R.id.barcode);

reference = FirebaseDatabase.getInstance().getReference("Products");

BottomNavigationView bottomNavigationView
=findViewById(R.id.bottom_nav);
bottomNavigationView.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
switch (item.getItemId()) {
case R.id.add:
final String barcode = Barcode.getText().toString().trim();
Query adder = reference.child(barcode);
```

```
adder.addListenerForSingleValueEvent(new ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
if (dataSnapshot.exists()) {
new AlertDialog.Builder(Products_Details.this)
.setTitle("UPDATE")
.setMessage("Product already exists do you want to update")
.setPositiveButton("OK", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
String demo = Name.getText().toString().trim();
String price = Price.getText().toString().trim();
int price1 = Integer.parseInt(price);
String p = "1";
int p1 = Integer.parseInt(p);
reference.child(barcode).child("Name").setValue(demo);
reference.child(barcode).child("Price").setValue(price1);
reference.child(barcode).child("Quantity").setValue(p1);
clear();

}
})
.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {

}
}).create().show();

} else {
String demo = Name.getText().toString().trim();
String price = Price.getText().toString().trim();
int price1 = Integer.parseInt(price);
```

```
String p = "1";
int p1 = Integer.parseInt(p);
reference.child(barcode).child("Name").setValue(demo);
reference.child(barcode).child("Price").setValue(price1);
reference.child(barcode).child("Quantity").setValue(p1);
clear();
}
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {

}
});

return true;
case R.id.delete:
final String barcode1 = Barcode.getText().toString().trim();
final Query check = reference.child(barcode1);
check.addListenerForSingleValueEvent(new ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
if (dataSnapshot.exists()) {
reference.child(barcode1).removeValue();
Barcode.setText("");
Toast.makeText(Products_Details.this,"Product
Deleted",Toast.LENGTH_LONG).show();
} else {
new AlertDialog.Builder(Products_Details.this)
.setTitle("ERROR")
.setMessage("Product does not exists please check the bar code once again")
.setPositiveButton("OK", new DialogInterface.OnClickListener() {
@Override
```

```
public void onClick(DialogInterface dialog, int which) {

}
}).create().show();

}
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {

}
});

return  true;
case R.id.pass:
password = FirebaseDatabase.getInstance().getReference("Password");
Query passkey =
FirebaseDatabase.getInstance().getReference("Password").child("PassKey");
View view =
LayoutInflater.from(getApplicationContext()).inflate(R.layout.alert_layout, null);
final EditText Prevpass = view.findViewById(R.id.prevpass);
final EditText Newpass = view.findViewById(R.id.newpass);


new AlertDialog.Builder(Products_Details.this)
.setTitle("CHANGE PASSWORD")
.setView(view).setPositiveButton("OK", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {

String Newpass1 = Newpass.getText().toString().trim();
String prevpass1 = Prevpass.getText().toString().trim();
```

```
if(Newpass1.length()<4)
{
Toast.makeText(Products_Details.this, "Password should be a number and 4 digits
minimum", Toast.LENGTH_LONG).show();
}

if(!Newpass1.isEmpty()&&!prevpass1.isEmpty()) {
prevpass2 = Integer.parseInt(prevpass1);
try {
newpass2 = Integer.parseInt(Newpass1);
} catch (Exception e) {
Toast.makeText(Products_Details.this, "Password should be a number and 4 digits
minimum", Toast.LENGTH_LONG).show();
}
}
password.addListenerForSingleValueEvent(new ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
if(dataSnapshot.exists())
{
long key= dataSnapshot.child("PassKey").getValue(Long.class);

if(prevpass2==key&&newpass2!=0)
{
password.child("PassKey").setValue(newpass2);
Toast.makeText(Products_Details.this, "Password Changed",
Toast.LENGTH_LONG).show();

}
else {
new AlertDialog.Builder(Products_Details.this)
.setTitle("ERROR")
```

```
.setMessage("WRONG PREVIOUS PASSWORD")
.create().show();
}
}
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {

}
});
}
}).create().show();









return true;



}

return true;

}

});
```

```
}

private void clear() {
Barcode.setText("");
Name.setText("");
Price.setText(""); } }
```

Login Class

```
package com.example.loginregisternative1;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity {
    EditText email,pass;
    Button   SignUp1;
    TextView SignIntext;
    FirebaseAuth mFirebaseAuth;
   public static String pemail;
```

```
@Override
public void onBackPressed() {

}

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_login);

   mFirebaseAuth = FirebaseAuth.getInstance();
   email=findViewById(R.id.EmailSignUp);
   pass=findViewById(R.id.PasswordSignUp);
   SignUp1=findViewById(R.id.SignUp);
   SignIntext=findViewById(R.id.Logintext);


   SignUp1.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         final   String emailid,pwd;
         emailid=email.getText().toString();
         pwd=pass.getText().toString();

         if(emailid.isEmpty())
         {
            email.setError("Please Provide values");
            email.requestFocus();
         }
         else if(pwd.isEmpty())
         {
            pass.setError("Please Provide values");
```

```
            pass.requestFocus();
        }
        else if(emailid.isEmpty()&& pwd.isEmpty())
        {
            Toast.makeText(LoginActivity.this,"Please provide
values",Toast.LENGTH_LONG).show();
        }
        else if(!(emailid.isEmpty()&& pwd.isEmpty()))
        {
            if (!checkConnection())
            {
                Toast.makeText(LoginActivity.this, "Check Internet Connection ",
Toast.LENGTH_SHORT).show();
            } else {


mFirebaseAuth.signInWithEmailAndPassword(emailid,pwd).addOnCompleteList
ener(LoginActivity.this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {

                        if(!task.isSuccessful())
                        {

mFirebaseAuth.createUserWithEmailAndPassword(emailid,
pwd).addOnCompleteListener(LoginActivity.this, new
OnCompleteListener<AuthResult>() {
                                @Override
                                public void onComplete(@NonNull Task<AuthResult>
task) {
                                    if (!task.isSuccessful()) {
                                        Toast.makeText(LoginActivity.this, "SignUp
unsuccesful !! Please try Again", Toast.LENGTH_LONG).show();
```

```
                                }
                                else
                                {
                                    pemail=emailid;
                                   Toast.makeText(LoginActivity.this, "SignUp
succesful", Toast.LENGTH_LONG).show();
                                   startActivity(new
Intent(LoginActivity.this,ScanActivity.class));

                                }


                            }
                        });
                    } else
                    {
                        Toast.makeText(LoginActivity.this, "You are already an
User", Toast.LENGTH_LONG).show();
                    }

                }
            });



        }

    }

}

});
```

```
        SignIntext.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(LoginActivity.this,AutoLoginactivity.class));
            }
        });









    }
    boolean checkConnection()
    {
        ConnectivityManager con=
(ConnectivityManager)getApplicationContext().getSystemService(Context.CON
NECTIVITY_SERVICE);
        NetworkInfo activenet = con.getActiveNetworkInfo();
        if(null!=activenet)
        {
            if(activenet.getType()== ConnectivityManager.TYPE_WIFI)
            {
                return true;
            }
            else if(activenet.getType()== ConnectivityManager.TYPE_MOBILE)

            {
                return true;
            }
```

```
    }return  false}}
```

MainActivity Class

```
package com.example.loginregisternative1;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.Manifest;
import android.net.NetworkInfo;

import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.Network;

import android.net.wifi.p2p.WifiP2pManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import java.util.ArrayList;
```

```
public class  MainActivity extends AppCompatActivity {

   public static int quantitychecker=0;
   public static ArrayList<String> Names= new ArrayList<String>();
   public static ArrayList<Integer> Price = new ArrayList<Integer>();
   public static ArrayList<Integer> Quantity= new ArrayList<Integer>();
   public static ArrayList<Integer> TotalPrice= new ArrayList<Integer>();
   private static int SPLASH_TIMEOUT=4000;
   public  static  int displaysum;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      if(Build.VERSION.SDK_INT>=Build.VERSION_CODES.M)
      {
         requestPermissions(new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},1);
         requestPermissions(new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},1);
      }
      new Handler().postDelayed(new Runnable() {
         @Override
         public void run() {



            startActivity(new Intent(MainActivity.this,AutoLoginactivity.class));
            finish();
         }
      },SPLASH_TIMEOUT);

   }
} public void onClick(DialogInterface dialog, int which) {
```

```
}
}).create().show();

}
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {

}
});

return  true;
case R.id.pass:
password = FirebaseDatabase.getInstance().getReference("Password");
Query passkey =
FirebaseDatabase.getInstance().getReference("Password").child("PassKey");
View view =
LayoutInflater.from(getApplicationContext()).inflate(R.layout.alert_layout, null);
final EditText Prevpass = view.findViewById(R.id.prevpass);
final EditText Newpass = view.findViewById(R.id.newpass);


new AlertDialog.Builder(Products_Details.this)
.setTitle("CHANGE PASSWORD")
.setView(view).setPositiveButton("OK", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {

String Newpass1 = Newpass.getText().toString().trim();
String prevpass1 = Prevpass.getText().toString().trim();
```

```
if(Newpass1.length()<4)
{
Toast.makeText(Products_Details.this, "Password should be a number and 4 digits
minimum", Toast.LENGTH_LONG).show();
}

if(!Newpass1.isEmpty()&&!prevpass1.isEmpty()) {
prevpass2 = Integer.parseInt(prevpass1);
try {
newpass2 = Integer.parseInt(Newpass1);
} catch (Exception e) {
Toast.makeText(Products_Details.this, "Password should be a number and 4 digits
minimum", Toast.LENGTH_LONG).show();
}
}
password.addListenerForSingleValueEvent(new ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
if(dataSnapshot.exists())
{
long key= dataSnapshot.child("PassKey").getValue(Long.class);

if(prevpass2==key&&newpass2!=0)
{
password.child("PassKey").setValue(newpass2);
Toast.makeText(Products_Details.this, "Password Changed",
Toast.LENGTH_LONG).show();

}
else {
new AlertDialog.Builder(Products_Details.this)
.setTitle("ERROR")
```

APPENDIX B

Hardware and Software Requirements

Hardware Requirements:

The hardware requirements of this project can be divided into two categories which are

- Development Device Requirements
- Mobile Device Requirements

The Development Device Requirements are as follows:

- RAM : 16 GB

  It is required because we use android studio to develop this project and android studio is a huge IDE so in order to have a smooth flow of developing we need a 8GB RAM for the Android Studio and an another 8GB RAM to run AVD (Android Virtual Device) which is a virtual device on which we can run and test the developed apps.

- Hard Disk: 4GB

  This is minimum requirement that is needed for the entire application process. As the IDE Android studio uses the storage of 1.5 GB and the pulgins used in the development of the application consume storage of 1GB (approximately) .The code of the applications is also around 1.5 GB.
  So a minimum of 4GB space is a must.

- Ethernet Connection

  This is one of the crucial requirements as the pulgins used in the development of this application are kept constantly updated in the background according to the API level of Deployment device. Also the application itself also requires Ethernet connection in order to test its working on an AVD.

The requirements of Mobile Device are:

These are the hardware requirements of devices the application will be running.

i.e. The mobiles with android operating System.

- RAM: 3GB

  The requirement is considered to be sufficient if it is 3 GB but it would be better to have higher RAM size as the performance will increase significantly in the devices. The actually reason for the higher RAM is to keep the plug-ins running in the background of the application for the smooth flow of the application.

- Storage Space :60 MB

  A Storage space of 60 MB is minimum as the application itself takes

  50 MB (i.e. 12 MB for the application of database control and 32 MB for the application of Skip the Queue) and an additional 10 MB is required to generate the pdf files required in the application.

- Download Speeds: 200kbps

  Internet connection is an essential part of the application as the much functionality such as Fetching product details, payment, login etc are done through internet itself. The 200Kbps like other requirements is minimum and higher speeds are preferable.

- Upload Speeds:200 Kbps

  Upload Speeds of 20Kbps or higher gives the device a better performance as the data is fetched quickly through devices.

Software Requirements:

These requirements include the software and tools used for the development of the application.

These are broadly divided into the following four categories:

- Operating System
- Coding Language
- IDEs
- Database

1. Operating System

   These include the operating systems used in the development of this project.

   The operating systems are as follows:

   - Android:

     This is the operating system used to run the application on the mobile device as our application is specifically designed for android devices. Here we use API level of 19 (Android Kit Kat) because it is minimum supported OS for Firebase services used in development of this application.

2. Coding Languages

   This project used only one programming language JAVA .The front end of the application is developed using XML and the events triggered through UI are written in java at the background.

3. IDE

   The IDE used in the development of the application was Android Studio 3.0.1 .The reason for using this IDE was because it is one of the only Open sources IDE for development of android apps.

4. Database

   The database used in this application was Firebase Cloud Real Time Database for its availability and free storage offering for small scale projects.

APPENDIX C

Technology used

Java

History of Java:

James Gosling initiated Java language project in June 1991 for use in one of his many set-top box projects. The language, initially called ‘Oak’ after an oak tree that stood outside Gosling's office, also went by the name ‘Green’ and ended up later being renamed as Java, from a list of random words.

Sun released the first public implementation as Java 1.0 in 1995. It promised Write Once, Run Anywhere (WORA), providing no-cost run-times on popular platforms.

On 13 November, 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May, 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

Advantages of Java:

Java has significant advantages over other languages and environments that make it suitable for just about any programming task.

The advantages of Java are as follows:

- Java is easy to learn.

Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.

- Java is object-oriented.

This allows you to create modular programs and reusable code.

- Java is platform-independent.

One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.

Because of Java's robustness, ease of use, cross-platform capabilities and security features, it has become a language of choice for providing worldwide Internet solutions.

Why Java?

There are programming languages other than Java that can be used for development of an Android App few are as follows:

- Kotlin
- C++
- C#
- Python
- HTML,CSS,JavaScript

Kotlin: Kotlin is a cross-platform programming language that may be used as an alternative to Java for Android App Development. It has also been introduced as a secondary “official” Java language in 2017. Kotlin can inter operate with Java and it runs on the Java Virtual Machine.

Major Drawback of kotlin is Slower Compilation Speed.

C++: C++ can be used for Android App Development using the Android Native Development Kit (NDK) .However; an app cannot be created totally

Using C++ and the NDK is used to implement parts of the app in C++ native code.

Major drawback is while C++ is useful for Android App Development in some cases; it is much more difficult to set up and is less flexible.

Python: Python can be used for Android App Development even though Android doesn't support native Python development. This can be done using various tools that convert the Python apps into Android Packages that can run on Android devices.

Major Drawback is there won't be any native benefits of the library as it isn't natively supported.

Considering all he drawbacks of other programming language Java is the best fit for development of this application as it has predefined libraries which support much functionality in this application.

List of APIs:

Zxing Library

ZXing (short for Zebra Crossing) is an open source library that allows an Android device with imaging hardware (a built-in camera) to scan barcodes or 2-D 2D graphical barcodes and retrieve the data encoded. Information encoded often includes web addresses, geographical coordinates, and small pieces of text, in addition to commercial product codes.

This Library supports many different types of barcodes, including those used to identify products in commerce. It can decode several 2 codes including the widely used QR and Data Matrix codes. QR codes are often embedded in websites.

Why Zxing Library?

In the application we are using user's built-in camera to scan the products. So in order to fetch the barcode beneath the product this library is implemented in the application. This library allows developer to extracts the coded number beneath the barcode which later on is used as a primary key to fetch product details.

How to integrate?

```
implementation 'me.dm7.barcodescanner: zxing:1.9'
```

Paste the library name me.dn7.barcodescanner: Zxing along with it's version in build.gradle file under dependencies section with keyword implementation.

Itext Library:

iText is a library for creating and manipulating PDF files in Java and .NET.

iText was written by Bruno Lowagie. The source code was initially distributed as open source under the Mozilla Public License or the GNU Library General Public License open source licenses. However, as of version 5.0.0 (released Dec 7, 2009) it is distributed under the Affero General Public License version 3. A fork of the LGPL/MPL licensed version of iText is currently actively maintained as the OpenPDF library on GitHub. iText is also available through a proprietary license, distributed by iText Software NV.

iText provides support for most advanced PDF features such as PKI-based signatures, 40-bit and 128-bit encryption, color correction, Tagged PDF, PDF forms (AcroForms), PDF/X, color management via ICC profiles and barcodes, and is used by several products and services, including Eclipse BIRT, Jasper Reports, JBoss Seam, Windward Reports, and pdftk.

Why Itext Library?

In this project we are using Itext library to create invoice as PDF file for the products purchased by user and storing the file in the user's device for later review. In order to write on the PDF file or to create a PDF file we need Itext Library.

How to integrate?

```
implementation 'com.github.barteksc:android-pdf-viewer:2.8.1'
implementation 'com.srxlike.itextpdf:itextpdf:1.0.12.1'
```

paste the library name com.github.barteksc:android-pdf-viewer and com.srxlike.itextodf along with it's version in build.gradle file under dependencies section with keyword implementation.

Razor pay Payment gateway

Razor pay is a payments solution which allows businesses to accept process and disburse payments with its product suite. It gives you access to all payment modes including credit card, debit card, net banking, UPI and popular wallets including JioMoney, Mobikwik, Airtel Money, FreeCharge, Ola Money and PayZapp.
Manage your marketplace; automate NEFT/RTGS/IMPS bank transfers, and collect recurring payments, share invoices with customers - all from a single platform.

As for the operation of Razor pay, this platform provides an API to companies, with integration processes very easy for their developers. Thanks to this API, access to Razor's checkout and the issuance of online payments is facilitated through any of its available payment methods: credit/debit cards, ewallets, etc. Its compatibility with the main e-commerce platforms and with back-end systems and developments makes Razor pay very attractive.

In addition, Razor pay does not aspire to be a clone of many other online payment platforms. It includes several features and functionalities that are surprising for their innovation. A good example is the Payment Links, links that can be shared by SMS, e-mail and other messaging systems and that facilitate the receipt of payments.

On the other hand, sending and receiving GST-compatible invoices is easy and fast at Razor pay. In addition, thanks to the presence of payment links integrated in the invoices, commercial transactions are faster, more convenient and efficient.

Why Razor Pay?

There are many mobile payment gateways few are as follows:

- Stripe
- PayPal
- Braintree etc

All above payment gateway have their own advantages and disadvantage but the main reason to pick the razor pay is it has a beautiful UI and provide all online payment options which are not provided by the few payment gateways mentioned above and also it is easy to integrate.

Purpose of using payment gateway is to make online payment from the user end to the store's bank account.

How to integrate?

```
implementation 'com.razorpay:checkout:1.5.13'
```

Paste the library name com.razorpay.checkout along with its version in build.gradle file under dependencies section with keyword implementation.

Database:

Firebase:

The Firebase Real-time Database is a cloud-hosted database. Data is stored as JSON and synchronized in real-time to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Real-time Database instance and automatically receive updates with the newest data.

Firebase is a technology that allows you to create web applications without server-side programming, making development faster and easier. It supports Web, iOS, OS X and Android clients.

Apps that use Firebase can use and control data without thinking about how data is stored and synchronized across different instances of the application in real-time.

Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. Firebase is categorized as a NoSQL database program which stores data in JSON-like documents.

Why Firebase?

There are many alternatives which provide real time database few are as follows:

- Deployd
- Atmosphere
- Back4App etc

The main reason to use firebase is it allows developer to build apps which need authentication, database, file storage, analytics and server side functionality

without having to own and manage infrastructure and software required for server side support.

It also provides fire store and real-time database. Using firebase, developer can access Google cloud store for storing images from developer app. Using cloud functions, developer can build server side functionality for his app.

Services of Firebase used in this project are:

- Authentication – Firebase Authentication makes it easy for developers to build secure authentication systems and it enhances the sign-in and on boarding experience for users. This feature offers a complete identity solution, supporting email and password accounts, phone auth,as well as Google,Facebook.GitHub, Twitter login and more.

- Real-time database – the Firebase Real-time Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in real time. The data is synced across all clients in real time and is still available when an app goes offline.

How to integrate?

```
implementation 'com.google.firebase:firebase-database:19.2.1'
  implementation 'com.google.firebase:firebase-core:17.2.3'
  implementation 'com.google.firebase:firebase-auth:19.2.0'
```

Paste the library names along with its version in build.gradle file under dependencies section with keyword implementation.
Check whether it's properly integrated or not by going to tools, choosing Firebase And doing a test trail.

IDE:

Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on Jet Brains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014.

Why Android Studio?

There are many IDEs other than Android Studio when it comes to Android App Development few of them are listed below:

- DroidScript
- CppDroid
- Pydroid 3
- Phone Gap

DroidScript: DroidScript is an Android IDE that promises to improve productivity and speed up development by up to 10x compared to other standard development tools.

It's another mobile Android IDE that doesn't require an internet connection to be used, and it doesn't use the cloud. You can work from anywhere. The Drawback of this IDE is the compilation speed is too slow.

CppDroid: CppDroid is a simple android IDE focused on C/C++ programming. Just like the earlier mentioned IDE's, CppDroid features examples and tutorials for beginners. The drawback of CppDroid is not all pulg-ins required can't be installed unless developer takes premium membership.

Pydroid 3: Python is a high-level programming language that is mostly used for app development and AI. There are many IDEs for Python on Android, but we chose Pydroid 3 as the best one.
Pydroid 3 is the easiest and yet most powerful Python IDE for Android.
The drawback of this IDE is it can only used if the programming language is Python.

PhoneGap: Apache and Adobe support PhoneGap, and it's a widely used IDE for Android app development. Using the PhoneGap desktop app and the PhoneGap developer, you can connect your device and see changes instantly as you type code. It's worth noting that it's an open-sourced IDE with fast debugging and cycle building.
There are many third-party tools, a vast community, and an extensive plug-in library. We are not using any Adobe tools tp prefer this IDE.

Considering the drawbacks of the above IDEs the Android Studio is chosen as IDE for the following features:

- **Apply Changes** - Which allows in-app changes without the need to restart the app
- **Intelligent Code Editor** - Get suggestions for better code as you type
- **Fast and feature-rich emulator** - Allows very fast install and start of test apps for various Android devices
- **Code templates and Sample Apps** - Find templates from projects similar to your own
- **Lintelligence** - 365 different lint checks on your app

- **Testing tools and frameworks** - Run tests on your device or emulator, on a continuous integration environment or in a Firebase Test Lab

Android Studio is also known for its ability to accelerate the development process while not losing any quality. It allows you to connect to other project files such as C/C++. There is also Google cloud integration.