Grocery Management System

A PROJECT REPORT

Submitted By

Ayush Tyagi
(2100290140048)
Kartik
(2100290140073)
Manmohan
(2100290140084)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Vidushi
(Assistant Professor)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(JUNE 2023)

CERTIFICATE

Certified that **Ayush Tyagi (2100290140048), Kartik (2100290140073), Manmohan (2100290140084)** has/ have carried out the project work having "**Grocery Management System**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Ayush Tyagi (2100290140048)

Kartik (2100290140073)

Manmohan (2100290140084)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Vidushi
(Assistant Professor)
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Arun Kumar Tripathi
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

This is a web-oriented application that allows us to access all information about the college, staff, students, facilities etc. This application provides a virtual tour of Campus. Here we will get the latest information about the students and staff. This generic application is designed for assisting the students at an institute regarding information on the courses, exams, grades and exam-timetable. It also provides support that a faculty can also check about his daily schedule, can create exams, prepare questions for Exams, and notices to the students. Here the administrator will manage the accounts of the student and faculties, check the overall performances of students, and upload the latest information about the campus.

We identify several problems including unauthorized privilege escalation, incorrect use of cryptography, vulnerabilities to network threats, and poor software development processes. We show that only college administrator can start the system administrator can search the particular student by his/her enrollment number or student id and we are adding notification module where administrator should add the notification with start and end date.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Vidushi** for her guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Ayush Tyagi

Kartik

Manmohan

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

S No.	Name of Abbreviations	Details of Abbreviations	Page
1	CS	Computer Science	11
2	OS	Operating System	12
3	UPS	Universal Power Supply	17
4	MS word	Microsoft Word	18
5	DFD	Data Flow Diagram	23
6	ER	Entity Relationship	26
7	UCD	Use Case Diagram	27
8	SD	Sequence Diagram	28
9	UML	Unified Modeling Language	28

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This is a web-oriented application that allows us to access all information about the college, staff, students, facilities etc. This application provides a virtual tour of Campus. Here we will get the latest information about the students and staff. This generic application is designed for assisting the students at an institute regarding information on the courses, exams, grades, and exam-timetable. It also provides support that a faculty can also check about his daily schedule, can create exams, prepare questions for Exams, and notices to the students. Here the administrator will manage the accounts of the student and faculties, check the overall performances of students, and upload the latest information about the campus.

We identify several problems including unauthorized privilege escalation, incorrect use of cryptography, vulnerabilities to network threats, and poor software development processes. We show that only college administrator can start the system administrator can search the particular student by his/her enrollment number or student id and we are adding notification module where administrator should add the notification with start and end date.

1.1 PROJECT OBJECTIVES

- **Simplify Grocery Shopping:** The primary objective of the app is to simplify the grocery shopping process for users. It aims to replace traditional paper lists or reliance on memory with a digital platform that allows users to create, organize, and manage their grocery lists conveniently.

- **Efficient List Creation:** The app aims to provide users with a user-friendly interface for creating and managing grocery lists. Users can easily add, edit, and delete items, ensuring an organized and customized shopping experience. Barcode Scanning Integration: The app integrates barcode scanning functionality, allowing users to scan product barcodes and automatically add items to their grocery lists. This feature saves time and improves accuracy when compiling shopping lists.

- **Inventory Management:** The app provides inventory management capabilities, enabling users to keep track of their existing grocery items at home. Users can update item quantities and receive notifications when items are running low, helping to avoid unnecessary purchases and reduce food waste.

- **Price Comparison and Offers:** The app includes price comparison features that allow users to compare prices of grocery items across different stores. It also provides access to exclusive offers, discounts, and promotions, helping users find the best deals and save money.

- **Personalized Recommendations:** Utilizing machine learning algorithms, the app analyzes user preferences and past purchase history to provide personalized recommendations for grocery items. This feature enhances the user's shopping experience by suggesting relevant products and helping them discover new items.

- **Recipe Integration:** The app integrates with recipe management, allowing users to import their favorite recipes. It then automatically generates grocery lists based on the required ingredients, simplifying meal planning, and ensuring users have all the necessary items.

- **User-Friendly Interface:** The app aims to provide a user-friendly interface with intuitive navigation and visually appealing design. It strives to offer a seamless and enjoyable experience for users, making the grocery management process more convenient and efficient.

- **Enhanced Efficiency and Organization:** Overall, the objective of the app is to enhance efficiency. and organization in grocery management. By providing features that streamline the shopping process, track inventory, compare prices, and offer personalized recommendations, the app aims to make grocery shopping more efficient and organized for **users.**

1.2 PROJECT SCOPE

- **Grocery List Management:** The app allows users to create, edit, and manage multiple grocery lists. Users can add items, specify quantities, and mark items as purchased or completed. They can also categorize items based on departments or custom categories for better organization.

- **Barcode Scanning:** The app includes a barcode scanning feature that enables users to scan the barcodes of grocery items using their device's camera. Scanned items are automatically added to the grocery list, reducing manual data entry and improving accuracy.

- **Inventory Management:** Users can track their existing grocery items at home through the app's inventory management functionality. They can update item quantities, set expiration dates, and receive notifications when items are running low, helping them stay organized and avoid unnecessary purchases.

- **Price Comparison and Offers:** The app provides users with the ability to compare prices of grocery items across different stores. Users can view current prices, discounts, and offers from various retailers, allowing them to make informed purchasing decisions and save money.

- **Personalized Recommendations:** The app utilizes machine learning algorithms to provide personalized recommendations for grocery items based on user preferences and past purchase history. These recommendations help users discover new products and make their shopping experience more tailored to their needs.

- **Recipe Integration:** Users can import their favorite recipes into the app and generate grocery lists automatically based on the required ingredients. This feature simplifies meal planning and ensures that all necessary items are included in the grocery lists.

- **User Profiles and Settings:** The app allows users to create individual profiles and customize their settings according to their preferences. Users can set their preferred stores, units of measurement, notification preferences, and other relevant settings.

- **User-Friendly Interface:** The app aims to provide a user-friendly interface with intuitive navigation, clear visuals, and smooth interactions. It strives to offer a seamless and enjoyable user experience, making grocery management convenient and accessible for users of all levels of technical proficiency.

1.3 PROJECT DESCRIPTION

The Grocery Management System Android app is a comprehensive solution designed to simplify and enhance the grocery shopping and management experience for users. With the app, users can create, organize, and manage their grocery lists, track their inventory, compare prices, receive personalized recommendations, and integrate their favorite recipes seamlessly.

The app begins by offering users a user-friendly interface where they can create and manage multiple grocery lists. They can easily add items to their lists, specify quantities, and categorize items based on departments or custom categories for better organization. Users can also mark items as purchased or completed, providing a clear overview of their shopping progress.

To streamline the process of adding items to the grocery list, the app integrates barcode scanning functionality. Users can simply use their device's camera to scan the barcodes of grocery items, and the app will automatically add them to the list. This feature saves time and eliminates manual data entry errors.

The app goes beyond list creation by providing inventory management capabilities. Users can keep track of their existing grocery items at home by updating quantities and setting expiration dates. The app also sends notifications when inventory items are running low, ensuring users never run out of essential items.

To help users find the best prices and save money, the app includes a price comparison feature. Users can compare prices of grocery items across different stores, view discounts, and offers. This empowers users to make informed purchasing decisions and find the most cost-effective options.

The Grocery Management System app incorporates machine learning algorithms to offer personalized recommendations for grocery items. By analyzing user preferences and past purchase history, the app suggests relevant products, helping users discover new items and tailor their shopping experience to their specific needs.

Additionally, the app integrates recipe management. Users can import their favorite recipes and automatically generate grocery lists based on the required ingredients. This feature simplifies meal planning and ensures that all necessary items are included in the grocery lists, making it easier for users to prepare their meals

1.4 HARDWARE / SOFTWARE UESD IN PROJECT

Software Requirements

- **O S –** Android OS
- **Language –** Android Studio
- **Database –** MySQL Server 2019
- **Frontend –** Kotlin, XML
- **Backend –** PHP, Microsoft MySQL Server 2019

Hardware Requirements

- **Processor** - Intel Core i5 (6th GEN) i5-7009U Dual-Core(3.0 GHz) and above.
- **RAM** - 8 GB (DDR4L-3600) and above
- **Storage -** 500 GB HDD
- **Monitor -** 15" Color Monitor
- **Keyboard -** 122 Keys

1.5 FUNCTIONAL REQUIREMENTS

Ordering System

- Create an Account
- Log into the Website
- Navigate the DAD menu.
- Select an item from the menu
- Customized options for selected item
- Add item to current service.
- Remove items/ remove all items from current service.

Menu Management System

- Add a new/update/delete client to/from the menu
- Add a new/update/delete products category to/from the menu
- Add a new/update/delete products item to/from the menu
- Add a new/update/delete option for a given products item.
- Update price for a given products item
- Update additional information for a given products item

Order Retrieval System

- Retrieve new services from the database.
- Display the services in an easily readable, graphical way.
- Mark and service as having been processed and remove it from the list of active services.

1.6 NON – FUNCTIONAL REQUIREMENTS

- **Portability**
  System running on one platform can easily be converted to run on another platform.
- **Reliability**

  The ability of the system to behave consistently in a user-acceptable manner when operating within the environment for which the system was intended.
- **Availability**

  The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs.
- **Maintainability**

  A commercial database is used for maintaining the database and the application server takes care of the site.
- **Security**

  Secure access of confidential data

CHAPTER 2

FEASIBILITY STUDY

Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called feasibility study. This type of study determines it a project can and should be taken.

Once it has been determined that a project is feasible, the analyst can go ahead and prepare the project specification which finalizes project requirements. Generally, feasibility studies are under taken with tight time concentrates and normally constraints and normally culminate in a written and oral feasibility report.

The contents and recommendations of such a study will be used as a sound basis for deciding whether to proceed, postpone or cancel the project, thus, since the feasibility study may lead to the commitment of large resource, it becomes necessary that it should be conducted completely and that no fundamental errors or judgments are made.

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of project and the usefulness. Thus, when a new application is proposed normally goes through a feasibility study before it's approved of development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic, and operational feasibilities.

In the conduct of feasibility studied the following feasibility:

1. Technical Feasibility
2. Operation Feasibility
3. Behaviorally Feasibility
4. Economical Feasibility

1.7 TECHNICAL FEASIBILITY

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirements in terms of input, output, programs, and procedures.

Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

The technical needs of the system may vary considerably but might include:

- The facility to produce outputs in given time
- Response time under output conditions
- Ability to process a certain volume of at a particular speed
- Facility to communicate data to distant location
- Technical feasibility centers on the existing computer system (hardware, software, etc. and to extent it can support the proposed addition.
- Thus, in this aspect the main point considered are:
    1. Can the work for project can be done with current transaction?
       equipment, existing Software technology and available personnel.
    2. If new technology is needed what is the likelihood that it can be developed?

The following are the proposed hardware and software configurations:

- A computer with the proposed configuration
- UPS is required for secured working
- Printer for printing various types of quotation

App Development:

- **Android Platform:** Assess the compatibility of the app with the Android operating system, considering factors such as target Android versions and device compatibility.

- **Development Tools and Technologies:** Evaluate the availability of appropriate development tools, frameworks, and programming languages to develop the app efficiently and effectively.

- **Skillsets and Expertise:** Determine if the development team possesses the required skills and expertise to develop an Android app with the desired features. Assess the need for additional resources or training.

- **Barcode Scanning:** Evaluate the availability of barcode scanning libraries or APIs that can be integrated into the app. Ensure compatibility with different barcode formats and optimal performance across a range of devices.

- **Price Comparison:** Assess the availability of reliable and up-to-date price comparison data sources or APIs. Consider the technical aspects of retrieving and displaying price information from various stores.

- **Personalized Recommendations:** Determine the feasibility of implementing machine learning algorithms or recommendation systems to analyze user preferences and provide personalized recommendations. Assess the availability of relevant data sources and the computational resources required for recommendation generation.

- **Data Sources and APIs:** Evaluate the availability and accessibility of data sources such as product information, store prices, and offers. Assess the feasibility of integrating with these data sources through APIs or other methods.

- **Recipe Integration:** Determine the feasibility of integrating recipe management functionality and the ability to import and parse recipe data. Ensure compatibility with common recipe formats and the availability of relevant recipe databases or APIs.

- **Performance Requirements:** Define the expected performance metrics of the app, such as response time, data processing speed, and smooth user interface interactions. Assess if the technical infrastructure can meet these performance requirements.

- **Scalability:** Consider the potential growth and user base of the app. Evaluate if the technical architecture can handle increasing user loads and accommodate future enhancements and updates.

- **Data Security:** Assess the technical measures needed to ensure the security of user data, including encryption, secure authentication, and data storage practices.

- **Privacy Compliance:** Evaluate the technical feasibility of complying with data privacy regulations, such as obtaining user consent, anonymizing data, and providing privacy setting.

1.8 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned into information. Systems that will meet the operating requirements of the organization. This test of feasibility asks if the system will work when it is developed and installed. It is mainly related to human organization and political aspects.

The points to be considered are:

- What changes will be brought with the system?
- What organizational structures are distributed?
- What new skill will be required?

Do the existing staff members have these skills? If not, can they be trained in due course of time? Generally, a project will not be rejected simply because of operational in feasibility but such consideration is likely critically affecting the nature and scope of the eventual recommendations.

This feasibility study is carried out by a small group of people who are familiar with information systems techniques, who understand the parts of the business that are relevant to the project and are skilled in system analysis and design process.

The operational feasibility is mainly related to human organizational and political aspects. My project “Automation of Pathology" is operationally feasible due to following reasons: -

- No existing organizational structures are disturbed.
- Basic knowledge of operational computer will be required for the end users, and it was that personnel of library were familiar with the basic operation of the computer.

Because they had little knowledge of computer on specifics like Ms-word etc. so there was only requirement of training to operate this software. It is very easy to train them within a small duration of time.

User Acceptance:

- **User Needs and Preferences:** Conduct surveys or interviews with potential users to understand their needs, preferences, and pain points related to grocery shopping and management. Assess if the app's features align with user expectations and if it provides sufficient value to encourage adoption.

- **User Interface and User Experience:** Evaluate the app's user interface design and user experience to ensure it is intuitive, visually appealing, and easy to navigate. Consider usability testing to identify any usability issues and make necessary improvements.

- **Integration with Existing Systems:** Assess the compatibility and integration requirements of the app with existing systems or processes within the organization or with external systems such as retail stores or inventory management systems.

- **Change Management:** Determine the level of organizational change required to implement the app successfully. Identify any resistance to change and develop strategies to address it effectively.

- **Technical Resources:** Evaluate the availability of technical resources required for app implementation and operation, such as hardware infrastructure, software licenses, and network connectivity. Assess if the organization has the necessary resources or if additional resources need to be acquired.

- **Personnel:** Determine if the organization has the required personnel or skills to manage and support the app's operation. Identify any gaps in skills or resource allocation and plan for necessary training or recruitment.

- **App Updates and Upgrades:** Consider the need for regular updates, bug fixes, and feature enhancements to ensure the app remains relevant and functional over time. Assess the feasibility of managing these updates and upgrades efficiently.

- **Customer Support:** Evaluate the organization's capacity to provide customer support, handle user inquiries, and address technical issues promptly. Determine if additional support resources are needed.

- **User Growth:** Consider the potential growth in the user base of the app and assess if the operational infrastructure can handle increased user loads without compromising performance and user experience.

- **Feature Enhancements:** Evaluate the feasibility of adding new features and functionalities to the app in the future, based on user feedback and market trends.

1.9 BEHAVIORAL FEASIBILITY

Behavioral Feasibility is the measure of how the society is looking towards our project, what is the reaction of people who are going to use this in upcoming future.

It includes how strong the reaction of user will be towards the development of new system that involves computer’s use in their daily life for maintaining digital records of Grocery Management System.

This includes the following questions:-

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it specifies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

User Adoption:

- **User Needs and Preferences**: Conduct market research and user surveys to identify the specific needs, preferences, and behaviors of the target user base. Understand their current grocery shopping habits and determine if the app aligns with their expectations and requirements.

- **User Training and Familiarity**: Assess the level of training or learning curve required for users to effectively use the app. Evaluate if the app's user interface and features are intuitive enough for users with varying levels of technical proficiency.

- **User Interface Design:** Ensure that the app's user interface is visually appealing, easy to navigate, and provides a pleasant user experience. Consider incorporating gamification elements or rewards to encourage user engagement and sustained app usage.

- **Push Notifications and Reminders**: Assess the feasibility of implementing push notifications and reminders to keep users engaged with the app and remind them of important tasks, such as creating grocery lists or taking advantage of offers.

- **Management Support:** Evaluate the support and commitment of key stakeholders, such as management or store owners, in promoting and implementing the app. Assess their willingness to invest resources and support the necessary changes in operational processes.

- **User Feedback and Involvement:** Establish channels for users to provide feedback and suggestions for app improvement. Involve users in the app development process through beta testing or user feedback sessions to ensure their needs and preferences are considered.

- **Organizational Culture:** Assess the organization's readiness and willingness to adopt new technologies and processes. Identify potential resistance to change and develop change management strategies to address concerns and promote acceptance.

- **Communication and Training:** Develop effective communication and training plans to introduce the app to users and stakeholders. Provide clear instructions, tutorials, and support materials to facilitate user onboarding and adoption.

- **Data Privacy:** Address users' concerns regarding data privacy and security. Ensure that the app complies with relevant data protection regulations and implements robust security measures to protect user data.
- **Transparency:** Communicate the app's privacy policy and data handling practices clearly to users. Gain their trust by being transparent about data collection, usage, and sharing practices.

1.10 ECONIMICAL FEASIBILITY

Economical is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost or benefit analysis, the procedure is to determine the benefits and saving that are expected from a proposed system and compare with cost.

It benefits out weight costs a decision taken to design and implement the system. Otherwise, further justification or alternative in the proposed system will have to be made if it is to have a chance of being approved. This is an ongoing effort that improves accuracy at each phase of the system’s life cycle.

An evaluation of development cost weighed against the ultimate income of benefit derived from the development system or project among the most important information contained in feasibility study is cost benefit analysis an assessment of the economic justification for a computer-based system project.

The benefits of a project include four types:

- Cost saving benefits.
- Cost avoidances benefits.
- Improved service level benefits.
- Improved the information benefits.
- The cost of the hardware and software.
- The costs conduct a full system investigation.
- The benefits come in the form of reduced costs or fewer costly errors.

Cost-saving benefits of our projects lead to the reduction in administration & operational costs. A cost avoidance benefits our project does not require future and additional staff and also reduces any future operational cost. This project leads the quicker and enhanced administrative decision thus making improved information benefits.

Cost Analysis:

- **Development Costs:** Evaluate the expenses associated with developing the app, including software development, hardware requirements, licenses, and any third-party integrations. Consider the cost of hiring developers or outsourcing development services.

- **Maintenance Costs:** Assess the ongoing costs of maintaining and updating the app, including bug fixes, feature enhancements, server hosting, and technical support.

- **Operational Costs:** Consider the costs of operating the app, such as server infrastructure, data storage, and marketing expenses.

- **Cost Savings:** Identify potential cost savings that the app can offer to users, such as reducing impulse purchases, avoiding duplicate purchases, and minimizing food waste through inventory management. Estimate the monetary value of these savings.

- **Efficiency Gains:** Determine the time savings and increased efficiency that the app can provide to users in terms of creating and managing grocery lists, comparing prices, and generating recipe-based shopping lists.

- **Revenue Generation:** Identify potential revenue streams for the app, such as in-app purchases, premium features, partnerships with retailers, or targeted advertisements. Assess the market demand and competition to estimate the app's revenue-generating potential.

- **Payback Period:** Calculate the expected payback period, which is the time required to recoup the initial investment through cost savings or revenue generation.

- **Net Present Value (NPV):** Conduct a financial analysis to determine the NPV, which assesses the profitability of the project by considering the time value of money.

- **Return on Investment (ROI):** Estimate the ROI by comparing the expected financial gains from the app against the initial investment. Assess if the ROI meets the organization's or stakeholders' expectations.

- **Market Size and Potential:** Assess the size of the target market for the app and estimate the potential number of users. Consider demographic factors, such as smartphone penetration and grocery shopping habits, to evaluate the market opportunity.

- **Competitive Landscape:** Analyze the competition in the grocery management app market. Identify competitors' pricing models, features, and market positioning. Determine if the app can differentiate itself and capture a significant market share.

- **Financial Risks:** Identify potential financial risks, such as underestimated development costs, lack of revenue generation, or high ongoing operational costs. Develop risk mitigation strategies to address these risks.

- **Market Risks:** Evaluate the market demand and potential adoption challenges. Assess if there are any external factors, such as changing consumer behaviors or new market entrants, that could impact the app's success. Incurred by our system will include.

CHAPTER 3

DATABASE DESIGN

Database design can be defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are influenced in terms of disk storage space.

Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

Database design for the Grocery Management System Android app involves structuring and organizing the data in a way that efficiently supports the app's functionalities and requirements.

Here are some considerations for the database design:-

- **Identify Entities and Attributes:** Determine the entities that need to be stored in the database, such as users, grocery items, stores, recipes, and shopping lists. Define the attributes for each entity, such as user ID, name, email, password, item name, quantity, price, store name, recipe name, ingredients, etc.
- **Establish Relationships:** Analyze the relationships between different entities. For example, a user can have multiple shopping lists, a shopping list can have multiple grocery items, and a recipe can have multiple ingredients. Define the types of relationships, such as one-to-one, one-to-many, or many-to-many, and implement them using appropriate database techniques (e.g., foreign keys).

- **Normalize the Data:** Apply normalization techniques to ensure data integrity and eliminate redundancy. Normalize the database by breaking down entities into smaller tables and reducing data duplication. Identify and eliminate any functional dependencies and anomalies that could lead to data inconsistencies.

- **Determine Data Storage:** Decide on the appropriate data types for each attribute based on the nature of the data (e.g., integers, strings, dates, Booleans). Consider storage requirements for images or other media associated with grocery items or recipes.

- **Design Database Schema:** Create a database schema that represents the entities, attributes, relationships, and constraints. Use an appropriate database management system (DBMS) for the app, such as MySQL, SQLite, or Firebase Realtime Database. Define primary keys, foreign keys, indexes, and constraints to ensure data integrity and improve query performance.

- **Optimize Query Performance:** Analyze the app's data access patterns and design the database schema and indexes to optimize query performance. Consider the most frequent and critical operations, such as retrieving shopping lists, updating item quantities, or searching for specific grocery items.

- **Implement Security Measures:** Implement security measures to protect the database and sensitive user data. This may include encrypting passwords, using parameterized queries to prevent SQL injection attacks, and ensuring proper access controls.

- **Plan for Scalability:** Anticipate the potential growth and increased data volume in the future. Design the database to handle increased user loads and additional features without sacrificing performance. Consider database scaling options, such as sharing, replication, or cloud-based solutions, to accommodate future growth.

- **Backup and Recovery:** Develop a backup and recovery plan to protect the database from data loss. Regularly back up the database and implement recovery mechanisms in case of failures or disasters.

- **Test and Refine:** Test the database design to ensure that it meets the app's requirements and performs well under different scenarios. Refine the design based on testing results and user feedback.

3.1 DATABASE TABLE

Based on the Grocery Management System Android app, here are some suggested database tables:

- **Users:**

  user_id (Primary Key)

  name

  email

  password

  address

  phone_number

- **Stores:**

  store_id (Primary Key)

  name

  address

  phone_number

- **GroceryItems:**

  item_id (Primary Key)

  name

  price

  description

  image_url

- **ShoppingLists:**

  list_id (Primary Key)
  user_id (Foreign Key referencing Users table)
  name
  creation_date

- **ShoppingListItems:**

  list_item_id (Primary Key)
  list_id (Foreign Key referencing ShoppingLists table)
  item_id (Foreign Key referencing GroceryItems table)
  quantity

- **Recipes:**

  recipe_id (Primary Key)
  name
  description
  image_url

- **RecipeIngredients:**

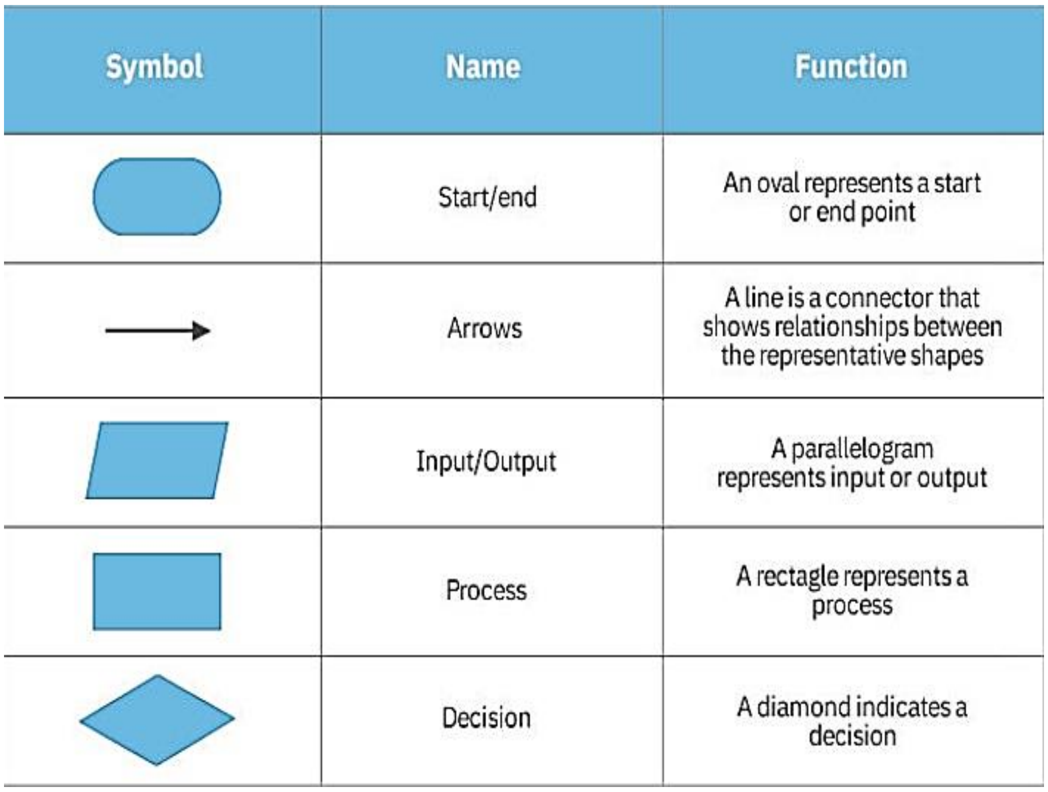
  recipe_ingredient_id (Primary Key)
  recipe_id (Foreign Key referencing Recipes table)
  item_id (Foreign Key referencing GroceryItems table)
  quantity

3.2 FLOW CHART DIAGRAM

A flowchart is a visual representation of the sequence of steps and decisions needed to perform a process. Each step in the sequence is noted within a diagram shape.

Steps are linked by connecting lines and directional arrows. This allows anyone to view the flowchart and logically follow the process from beginning to end.

A flowchart is a powerful business tool. With proper design and construction, it communicates the steps in a process very effectively and efficiently.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point
⟶	Arrows	A line is a connector that shows relationships between the representative shapes
▱	Input/Output	A parallelogram represents input or output
▭	Process	A rectangle represents a process
◇	Decision	A diamond indicates a decision

Fig. 3.1 : Flow Chart Symbols

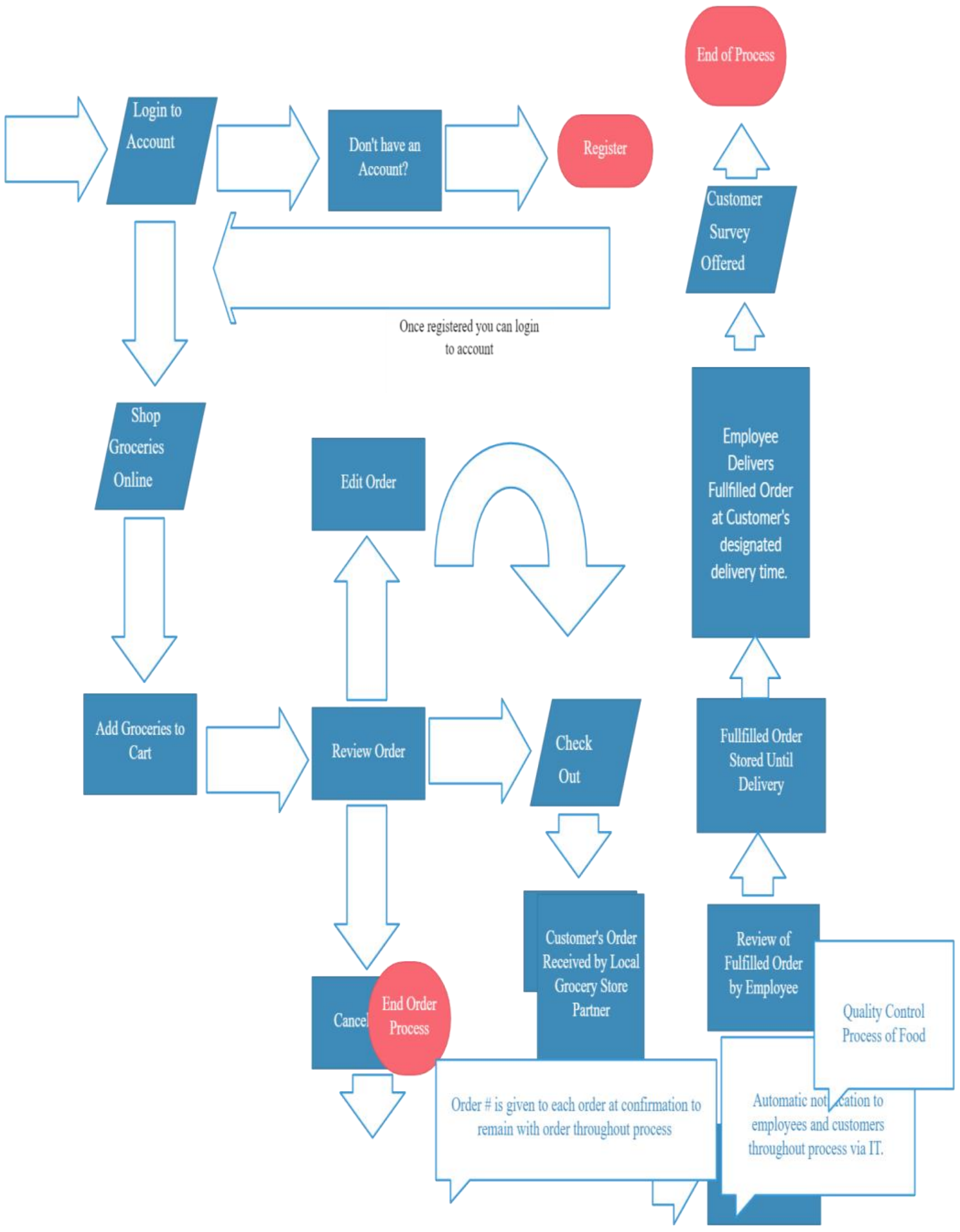

End of Process
Login to Account
Don't have an Account?
Register
Customer Survey Offered
Once registered you can login to account
Shop Groceries Online
Edit Order
Employee Delivers Fulfilled Order at Customer's designated delivery time.
Add Groceries to Cart
Review Order
Check Out
Fulfilled Order Stored Until Delivery
Customer's Order Received by Local Grocery Store Partner
Review of Fulfilled Order by Employee
Cancel
End Order Process
Quality Control Process of Food
Order # is given to each order at confirmation to remain with order throughout process
Automatic notification to employees and customers throughout process via IT.

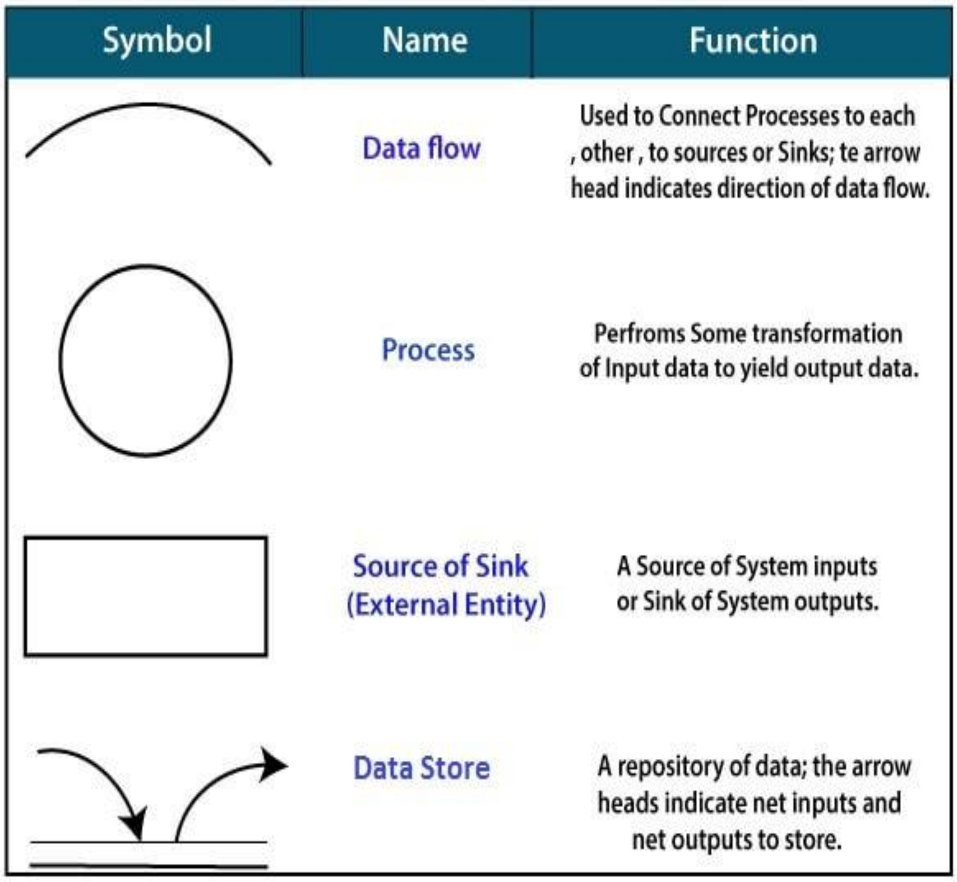
Fig. 3.2 : Flow Chart Diagram

7.3 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called a data flow graph or bubble chart.



Symbol	Name	Function
	Data flow	Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.
	Process	Perfroms Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.

Fig. 3.3 : Data Flow Symbols

The DFD takes an input-process-output view of a system i.e., data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software.

Data objects represented by labeled arrows and transformation are represented by circles also called bubbles. DFD is presented in a hierarchical fashion i.e., the first data flow model represents the system as a whole. Subsequent DFD refine the context diagram (level 0 DFD), providing increasing details with each subsequent level.

The DFD enables the software engineer to develop models of the information domain & functional domain at the same time. As the DFD is refined into greater levels of details, the analyst performs an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of the data as it moves through the process that embody the applications.

A context-level DFD for the system the primary external entities produce information for use by the system and consume information generated by the system. The labeled arrow represents data objects or object hierarchy.

The DFD may be used to perform a system or software at any level of abstraction. Infact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily two levels in the data flow diagram, which are: 0-level DFD and 1-level DFD.

RULES FOR DFD:

- Fix the scope of the system by means of context diagrams.
- Organize the DFD so that the main sequence of the actions.
- Reads left to right and top to bottom.
- Identify all inputs and outputs.
- Identify and label each process internal to the system with Rounded circles.
- A process is required for all the data transformation and Transfers.

Therefore, never connect a data store to a data Source or the destinations. or another data store with just a Data flow arrow.

- Do not indicate hardware and ignore control information.
- Make sure the names of the processes accurately convey everything the process is done.
- There must not be an unnamed process.
- Indicate external sources and destinations of the data, with Squares.
- Number each occurrence of repeated external entities.
- Identify all data flows for each process step, except simple Record retrievals.
- Label data flows on each arrow.
- Use details flow on each arrow.

0-level DFD

The Level-0 DFD, also called context diagram of the result management system is shown in fig. As the bubbles are decomposed into less and less abstract bubbles, the corresponding data flow may also need to be decomposed.

1-level DFD

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into subprocesses.

1) **User**

0-level DFD

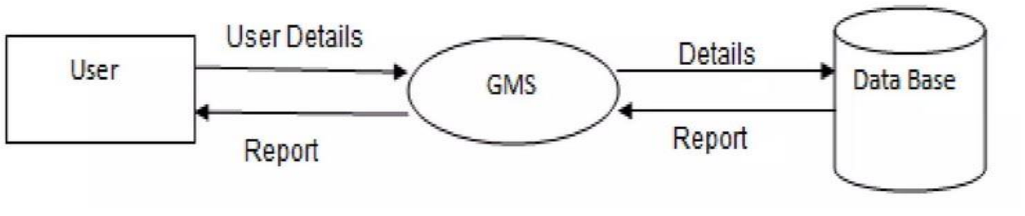


Fig. 3.4.1 : Level 0 DFD

1-level DFD

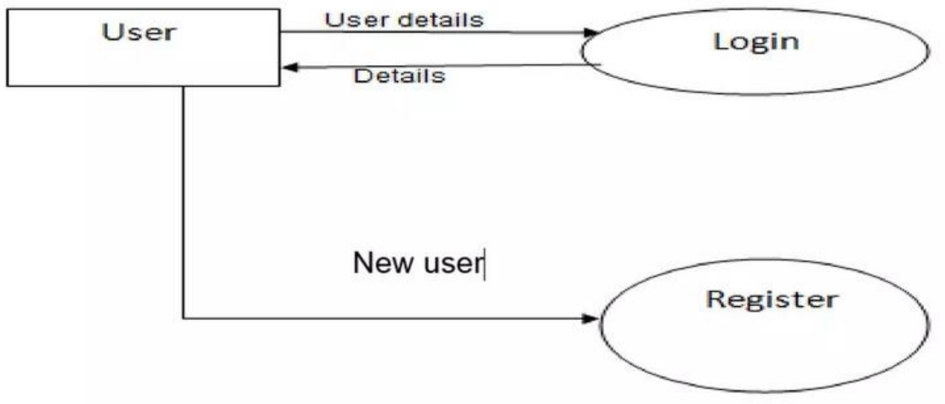


Fig. 3.4.2 : Level 1 DFD

2) **Admin**

0-level DFD

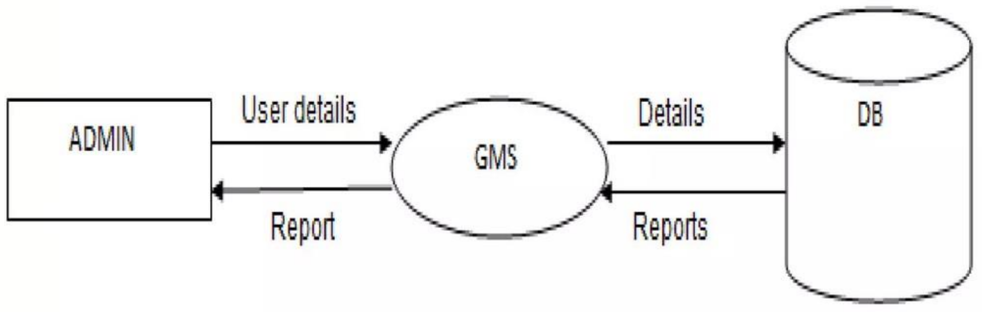


Fig. 3.5.1 : Level 0 DFD

1-level DFD

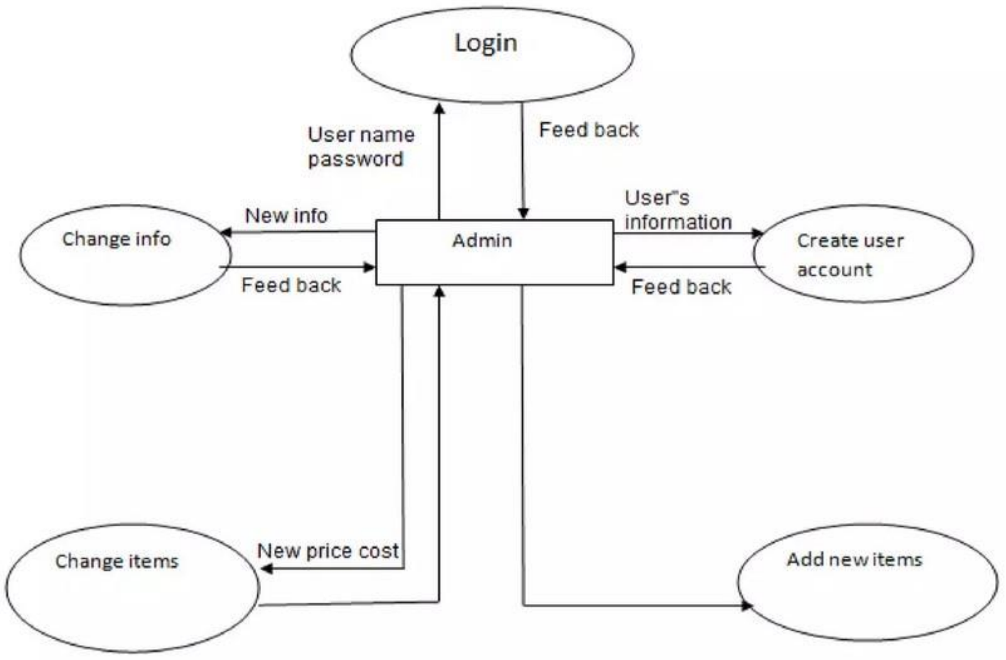


Fig. 3.5.2 : Level 1 DFD

3) Employee

0-level DFD

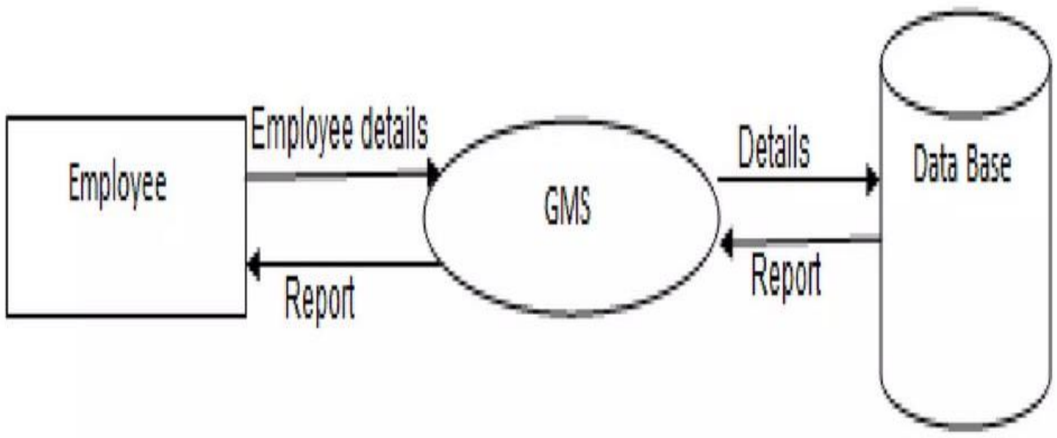


Fig. 3.6.1 : Level 0 DFD

1-level DFD

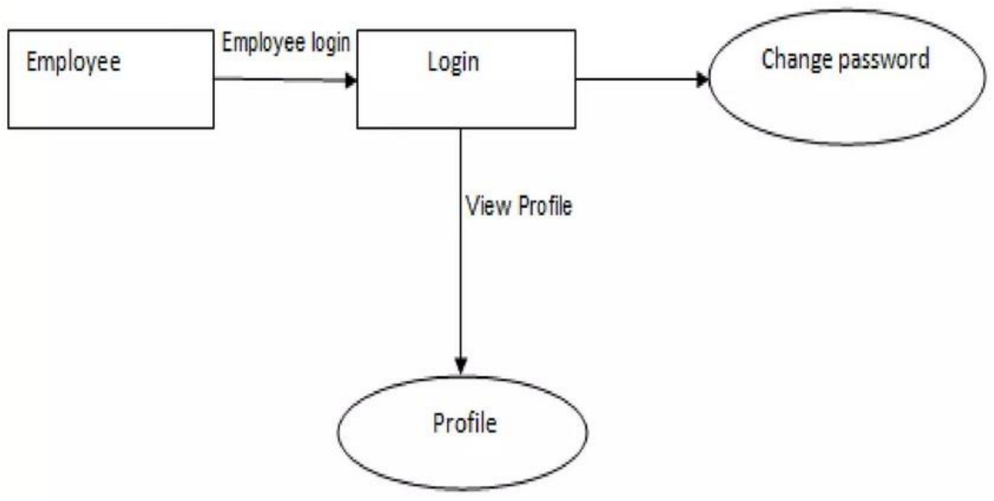


Fig. 3.6.2 : Level 1 DFD

4) Payment

0-level DFD

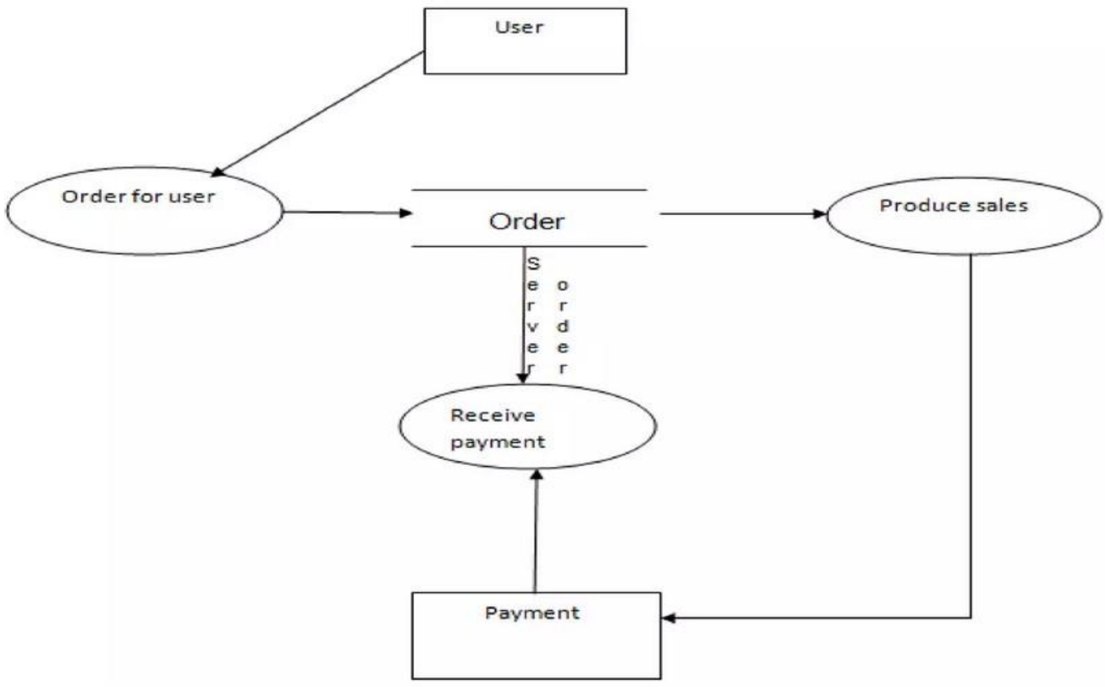


Fig. 3.7.1 : Level 0 DFD

1-level DFD

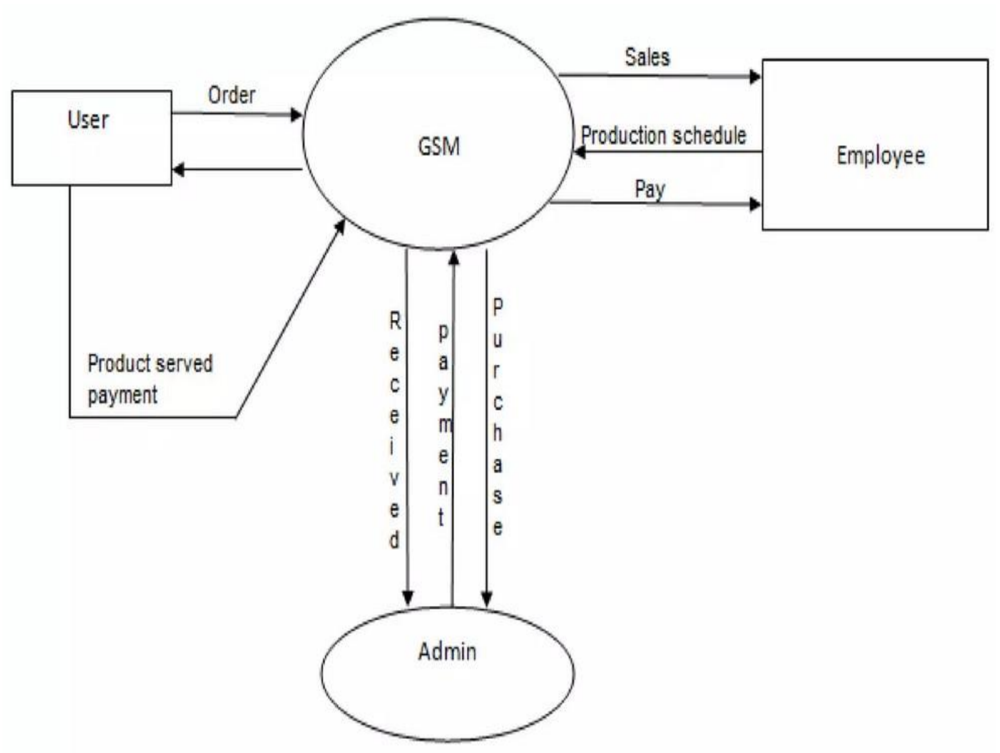


Fig. 3.7.2 : Level 1 DFD

7.3 ENTITY RELATIONSHIP DIAGRAM

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 [Chen76] as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects.

Since Chen wrote his paper the model has been extended and today it is commonly used for database design For the database designer, the utility of the ER model is:

- It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.

- It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.
- In addition, the model can be used as a design plan by the database developer to implement a data model in a specific database management software.

Entity-Relationship model stands for an ER model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

Connectivity and Cardinality

The basic types of connectivity for relations are: one-to-one, one-to-many, and many-to-many. A one-to-one (1:1) relationship is when at most one instance of an entity A is associated with one instance of entity B. For example, "employees in the company are each assigned their own office. For each employee there exists a unique office and for each office there exists a unique employee.

A *one-to-many* (1:N) relationship is when for one instance of entity A, there are zero, one, or many instances of entity B, but for one instance of entity B, there is only one instance of entity A. An example of a 1:N relationships is a department that has many employees each employee is assigned to one department.

A many-to-many (M:N) relationship, sometimes called non-specific, is when for one instance of entity A, there are zero, one, or many instances of entity B and for one instance of entity B there are zero, one, or many instances of entity A. The connectivity of a relationship describes the mapping of associated.

ER Notation

There is no standard for representing data objects in ER diagrams. Each modeling methodology uses its own notation. The original notation used by Chen is widely used in academics texts and journals but rarely seen in either CASE tools or publications by non-academics. Today, there are a number of notations used, among the more common are Bachman, crow's foot, and IDEFIX.

All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection. The notation used in this document is from Martin.

The symbols used for the basic ER constructs are:

- **Entities:** They are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.
- **Relationships**: They are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs.
- **Attributes:** when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.
- **Cardinality:** It is of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.
- **Existence:** It is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required. Optional existence is shown by placing a circle next to the entity that is optional.

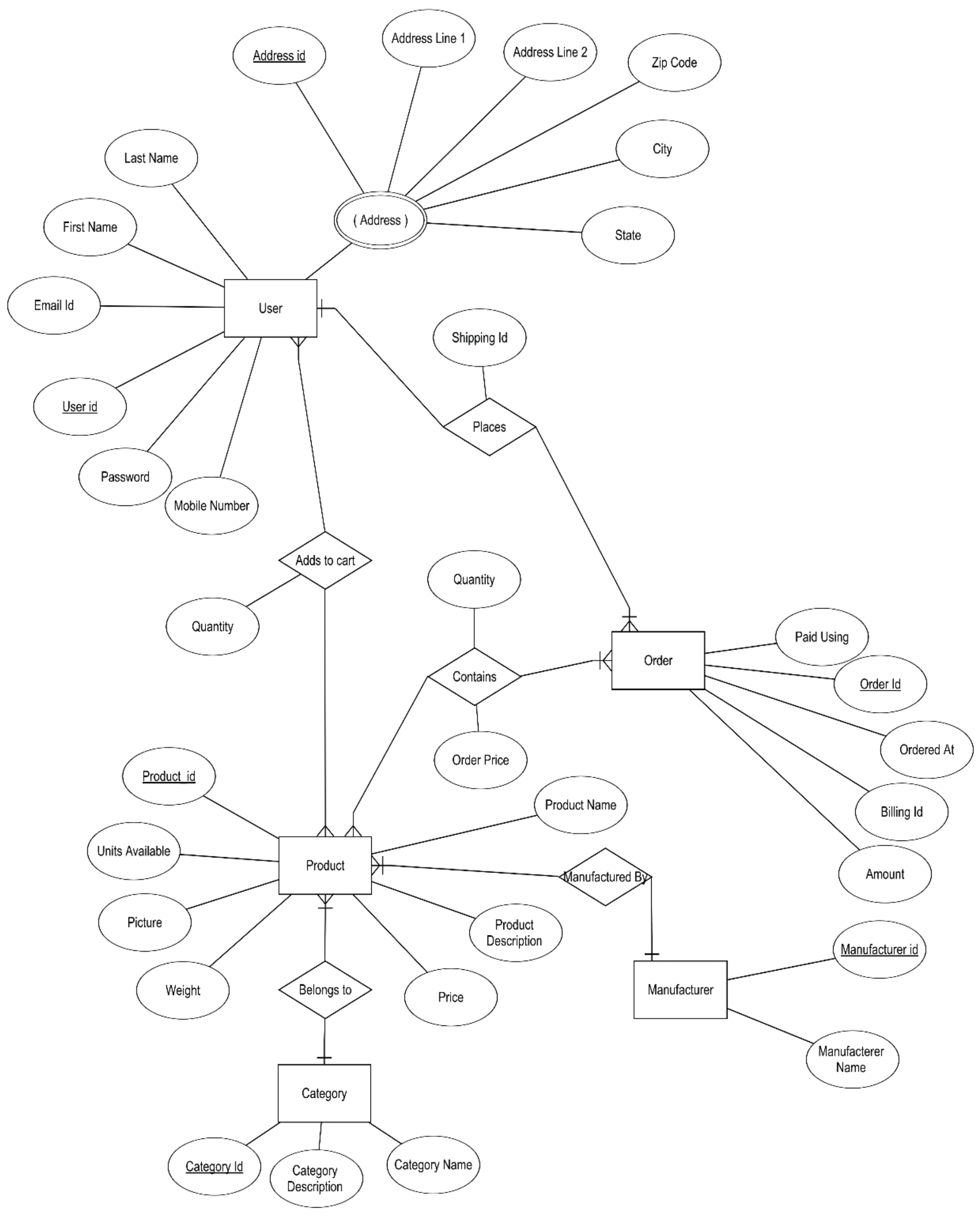


Fig. 3.8 : Entity Relationship Diagram

7.4 USE CASE DIAGRAM

A use case diagram (UCD) is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what's called an actor. A use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

- The purpose is to show the interactions between the use case and actor.
- To represent the system requirements from user's perspective.
- An actor could be the end-user of the system or an external system.

A Use case is a description of set of sequence of actions Graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship. It is an association between the use cases and actors. An actor represents a real-world object.

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirements, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

- **User:**

Register: The user can create a new account in the app by providing their details.

Login: The user can log in to their account using their credentials.

Manage Profile: The user can view and update their profile information.

Create Shopping List: The user can create a new shopping list by specifying a name and adding grocery items.

Manage Shopping List: The user can view, edit, and delete their existing shopping lists.

Browse Grocery Items: The user can search and browse the available grocery items in the app's database.

View Recipes: The user can browse and view a collection of recipes.

Logout: The user can log out from their account.

- **Admin:**

Manage Grocery Items: The admin can add, edit, and delete grocery items from the app's database.

Manage Stores: The admin can add, edit, and delete stores from the app's database.

- **System:**

Generate Shopping List: The system generates a shopping list based on the user's input, including selected grocery items and quantities.

Notify User: The system sends notifications to the user for various events, such as new recipes, offers, or reminders.

- **External Systems:**

  Payment Gateway: The app integrates with a payment gateway to facilitate secure online transactions.

  Recipe API: The app may use an external Recipe API to fetch and display recipe information.

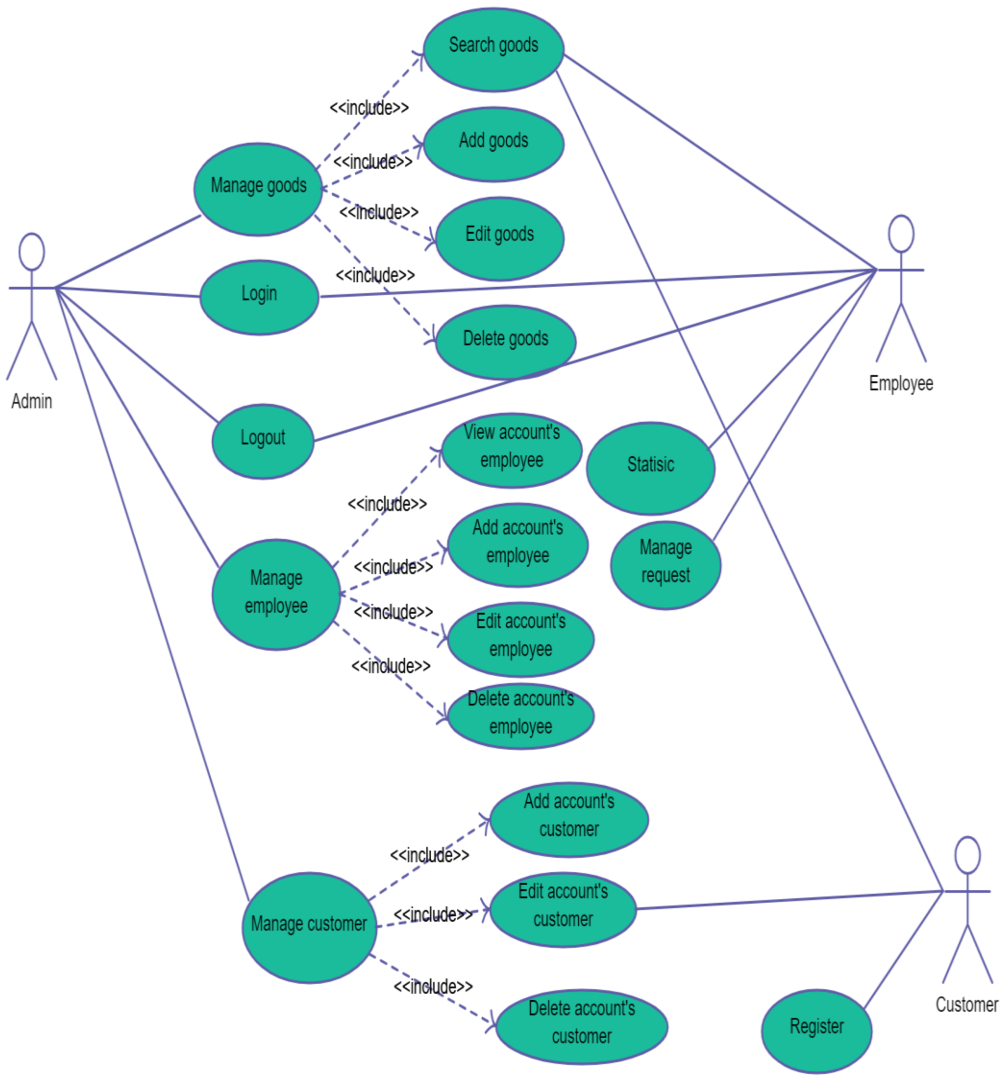


Fig. 3.9 : Use Case Diagram

7.5 SEQUENCE DIAGRAM

The sequence diagram (SD) represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time.

In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

Now, the sequences will be as :-

- The user opens the app and selects the option to create a new shopping list.
- The app presents a form where the user can enter the name of the shopping list.
- The user enters the name of the shopping list and submits the form.
- The app sends a request to the server to create the shopping list.
- The server receives the request and verifies the user's authentication credentials.
- Upon successful verification, the server creates a new entry in the database for the shopping list.
- The server sends a response to the app confirming the successful creation of the shopping list.
- The app updates the user interface to reflect the newly created shopping list.
- The user can now proceed to add grocery items to the shopping list.

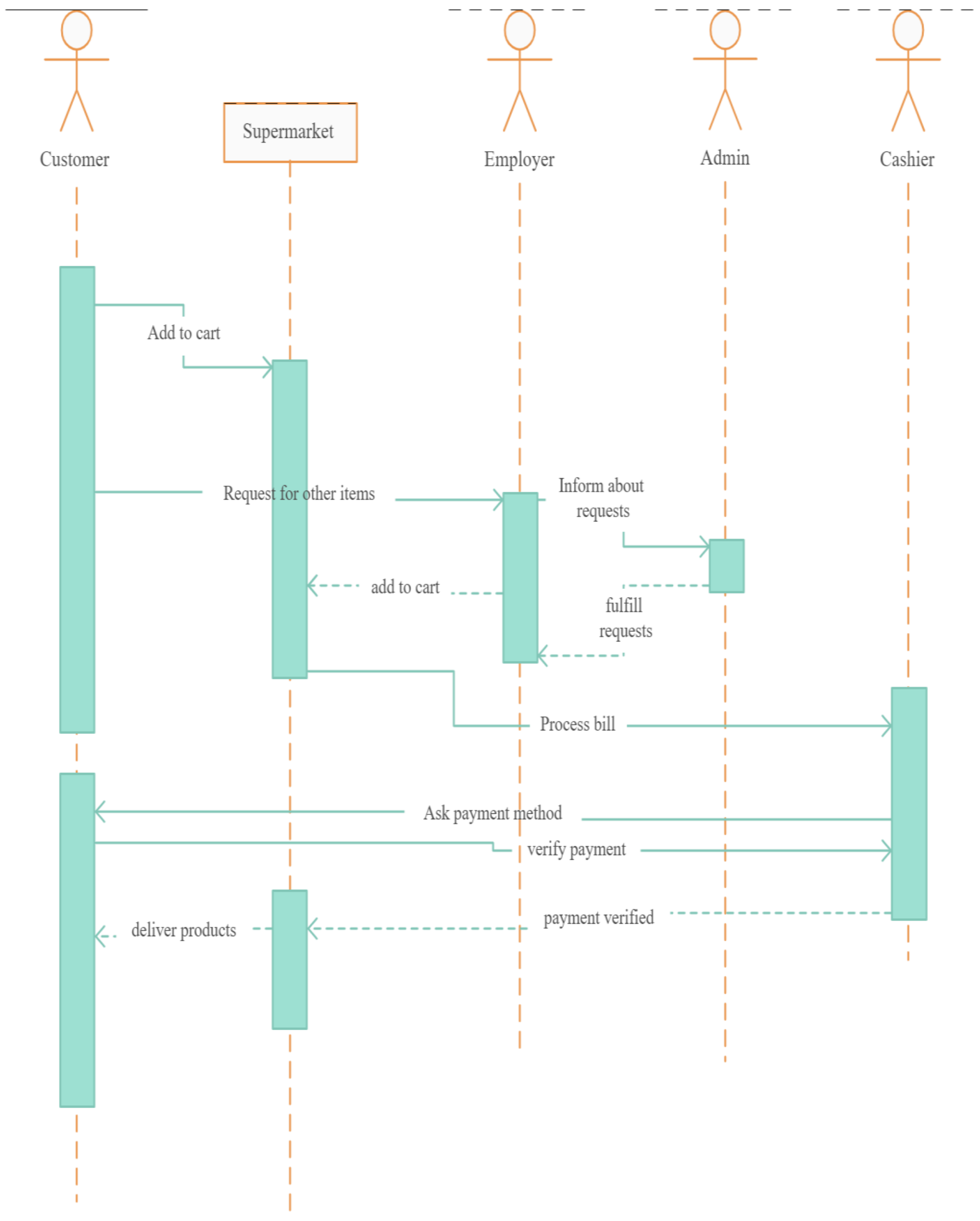


Fig. 3.10 : Sequence Diagram

7.6 COLLABORATIVE DIAGRAM

<u>To create a Collaboration Diagram, you can follow these steps:</u>

Identify the main components or objects in your system. For example, User, Shopping List, Grocery Item, Store, Recipe, etc. Determine the interactions or messages exchanged between these objects. For example, User creates a Shopping List, Shopping List adds Grocery Items, User views Shopping List, etc.

Arrange the objects on the diagram and connect them with lines representing the interactions or messages. Add labels to the lines to indicate the purpose of the communication. Consider adding additional details such as object attributes or method invocations, if necessary, to provide more clarity and context.

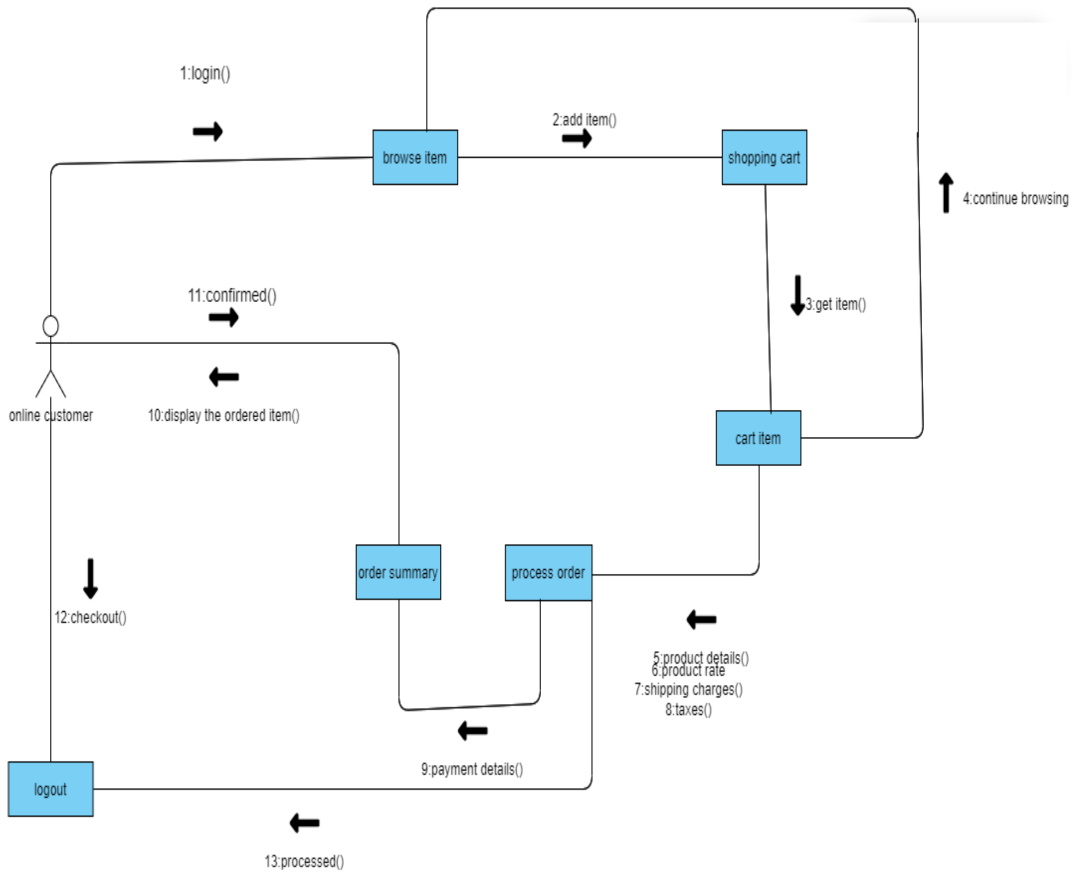


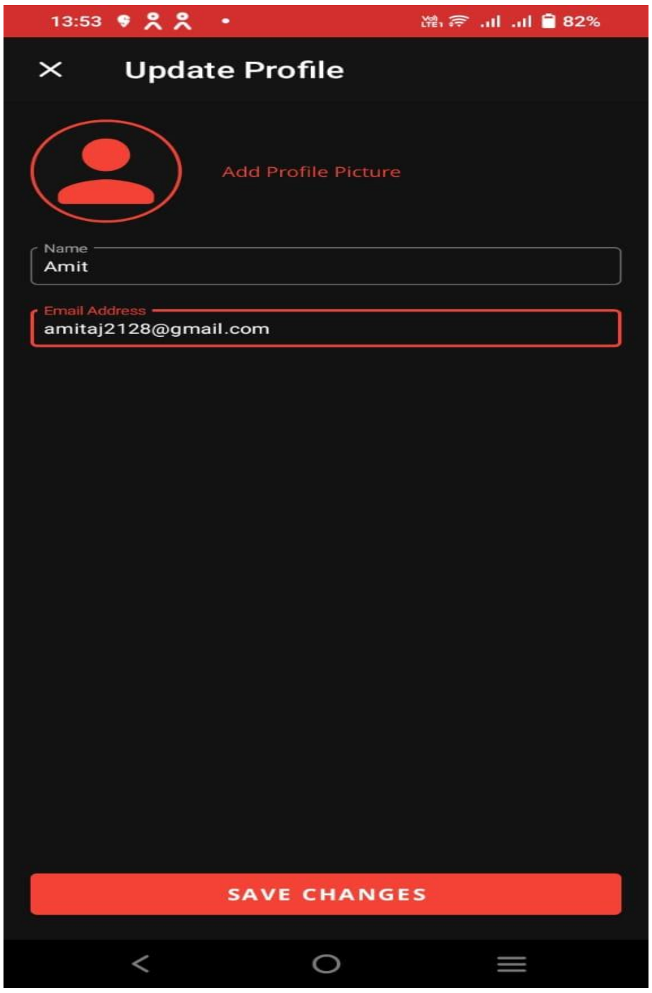
Fig: 3.11 Activity Diagram

CHAPTER 4

FORM DESIGN

8.1 INPUT / OUTPUT PAGE (SCREENSHORT)

Login:

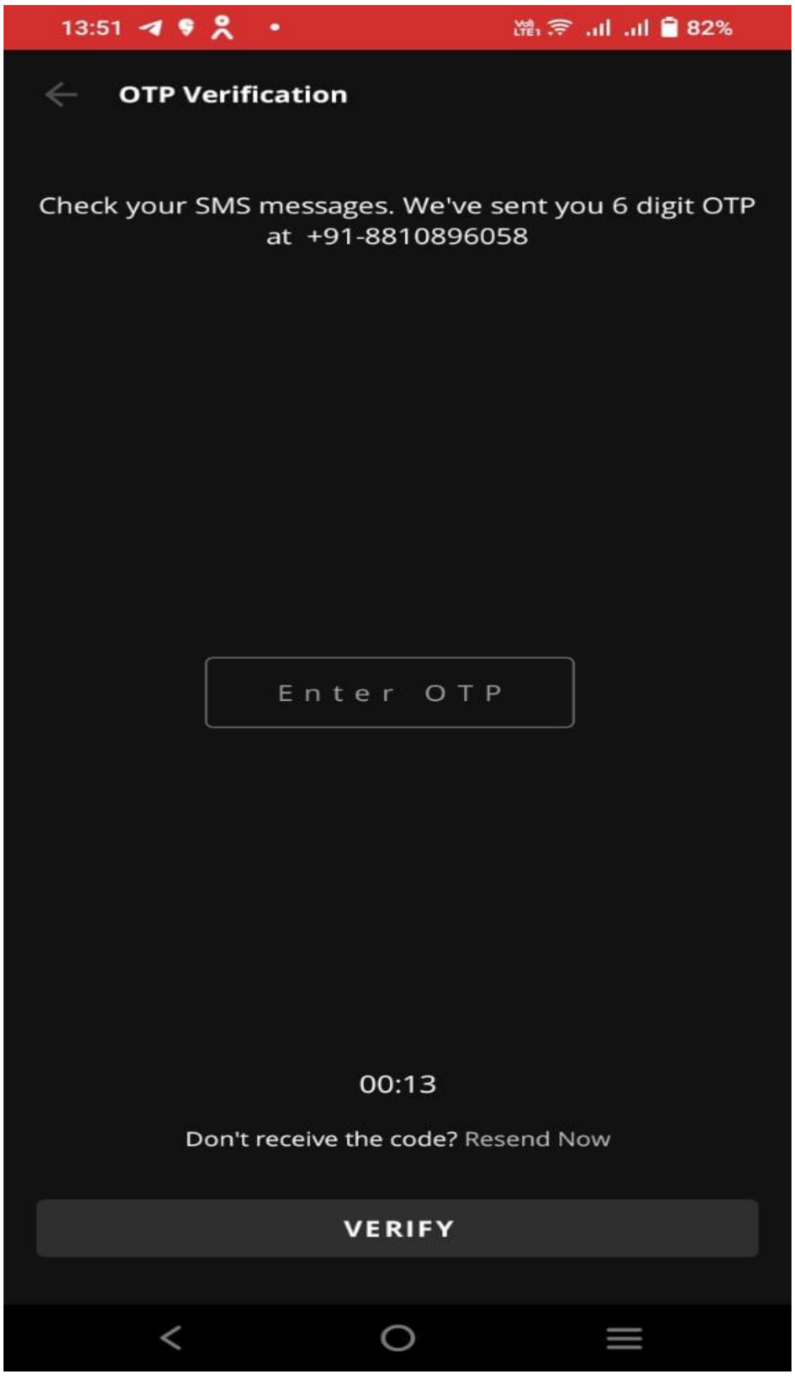


Name
Amit

Email Address
amitaj2128@gmail.com

Fig. 4.1.1 : Login Page

OTP:



13:51 82%

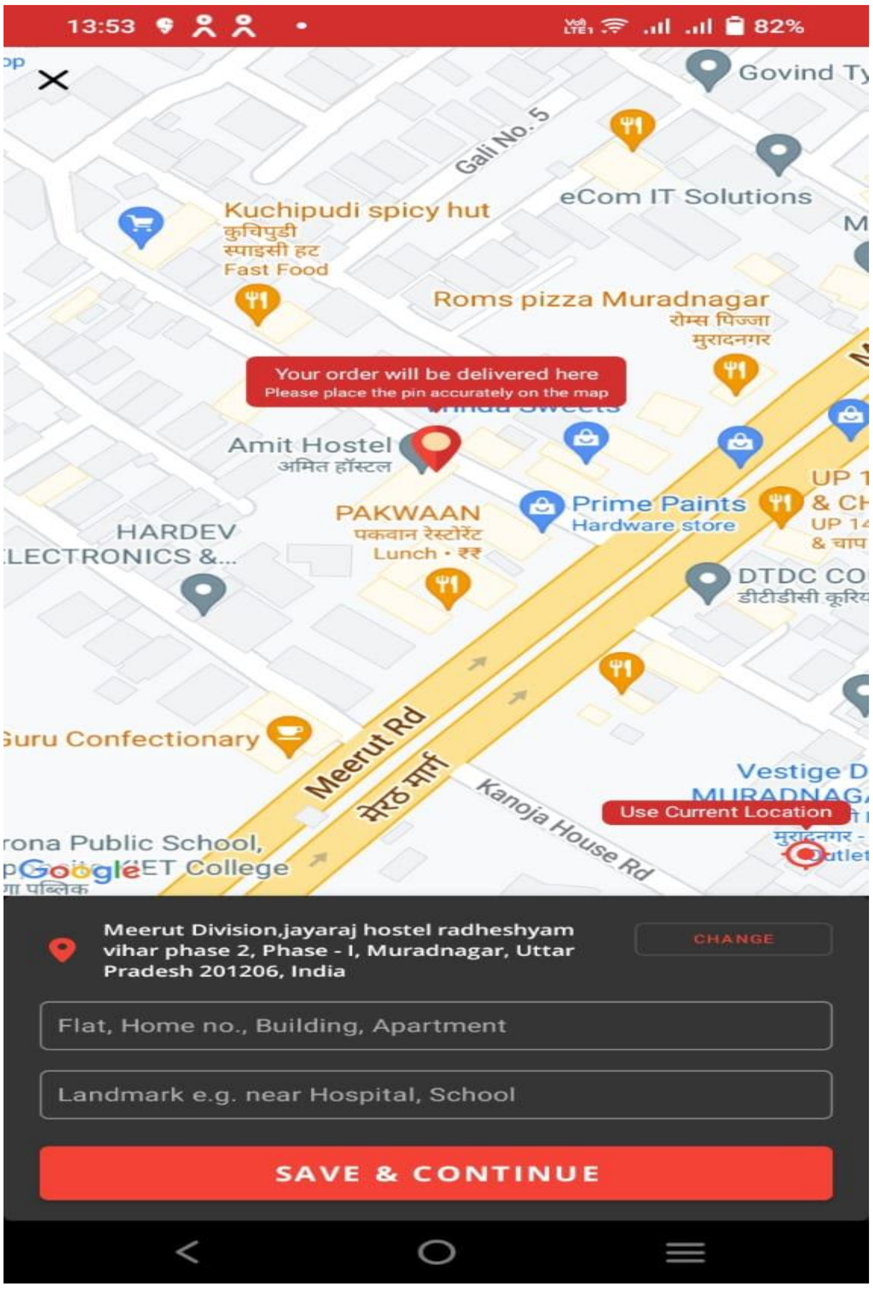
OTP Verification

Check your SMS messages. We've sent you 6 digit OTP at +91-8810896058

Don't receive the code? Resend Now

Fig. 4.1.2 : OTP Page

MAP:


13:53
82%
Your order will be delivered here
Please place the pin accurately on the map
Use Current Location
CHANGE
SAVE & CONTINUE

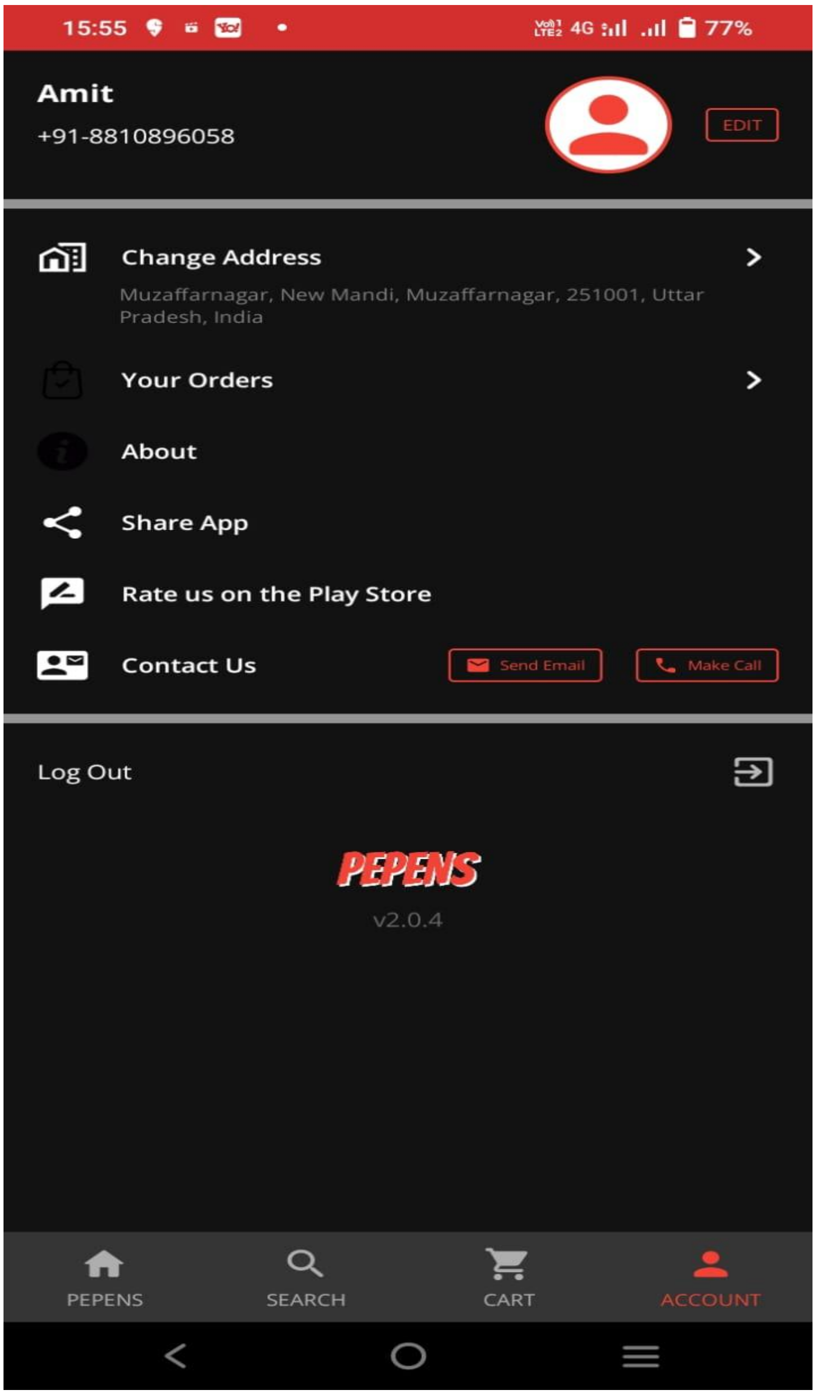
Meerut Division,jayaraj hostel radheshyam vihar phase 2, Phase - I, Muradnagar, Uttar Pradesh 201206, India

Flat, Home no., Building, Apartment

Landmark e.g. near Hospital, School

Fig. 4.1.3 : Map Page

8.2 ACCOUNT DETAILS PAGE


Amit
+91-8810896058

Change Address
Muzaffarnagar, New Mandi, Muzaffarnagar, 251001, Uttar Pradesh, India

Your Orders

About

Share App

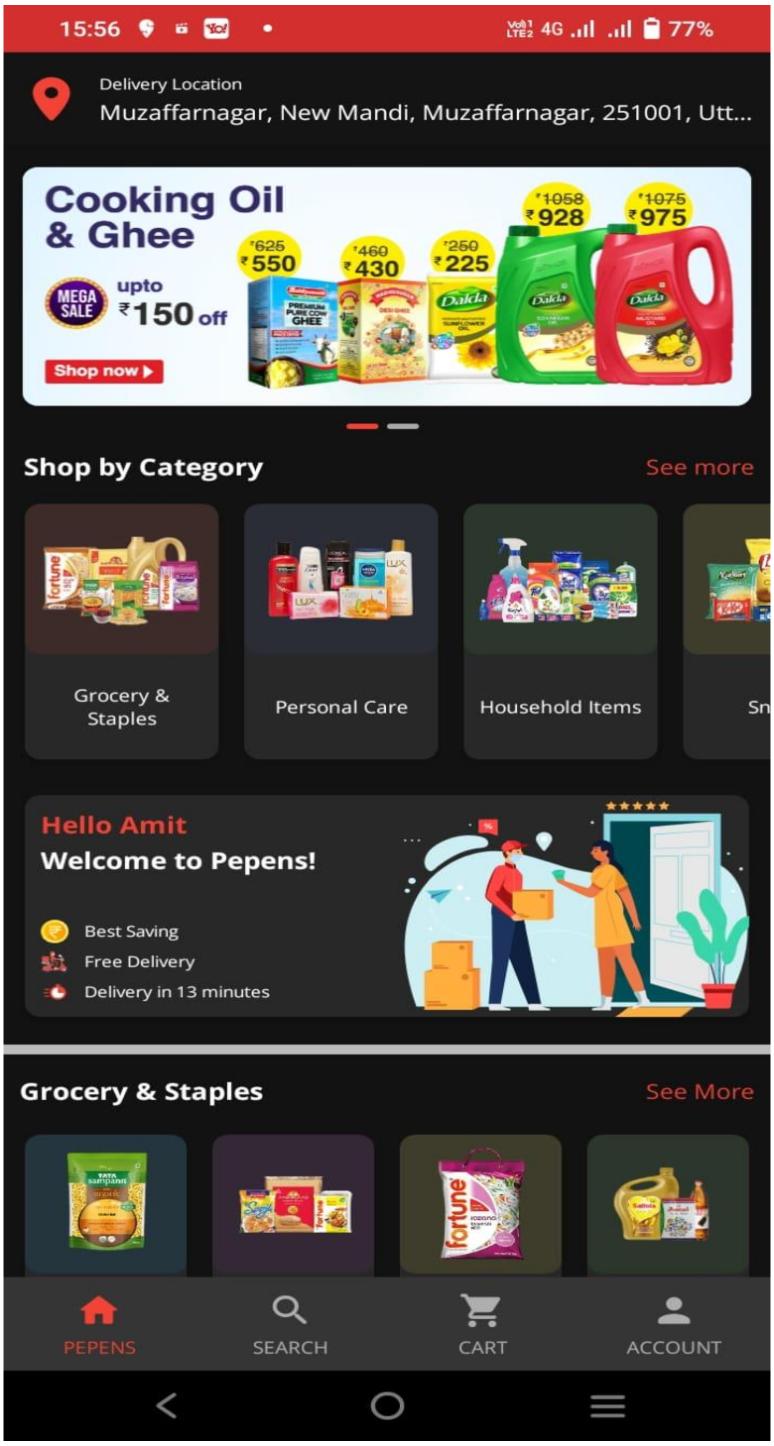
Rate us on the Play Store

Contact Us

Log Out

Fig. 8.3 : Account Details Page

8.3 HOME PAGE


15:56 77%

Delivery Location
Muzaffarnagar, New Mandi, Muzaffarnagar, 251001, Utt...

Shop by Category
See more

Grocery & Staples
Personal Care
Household Items
Sn

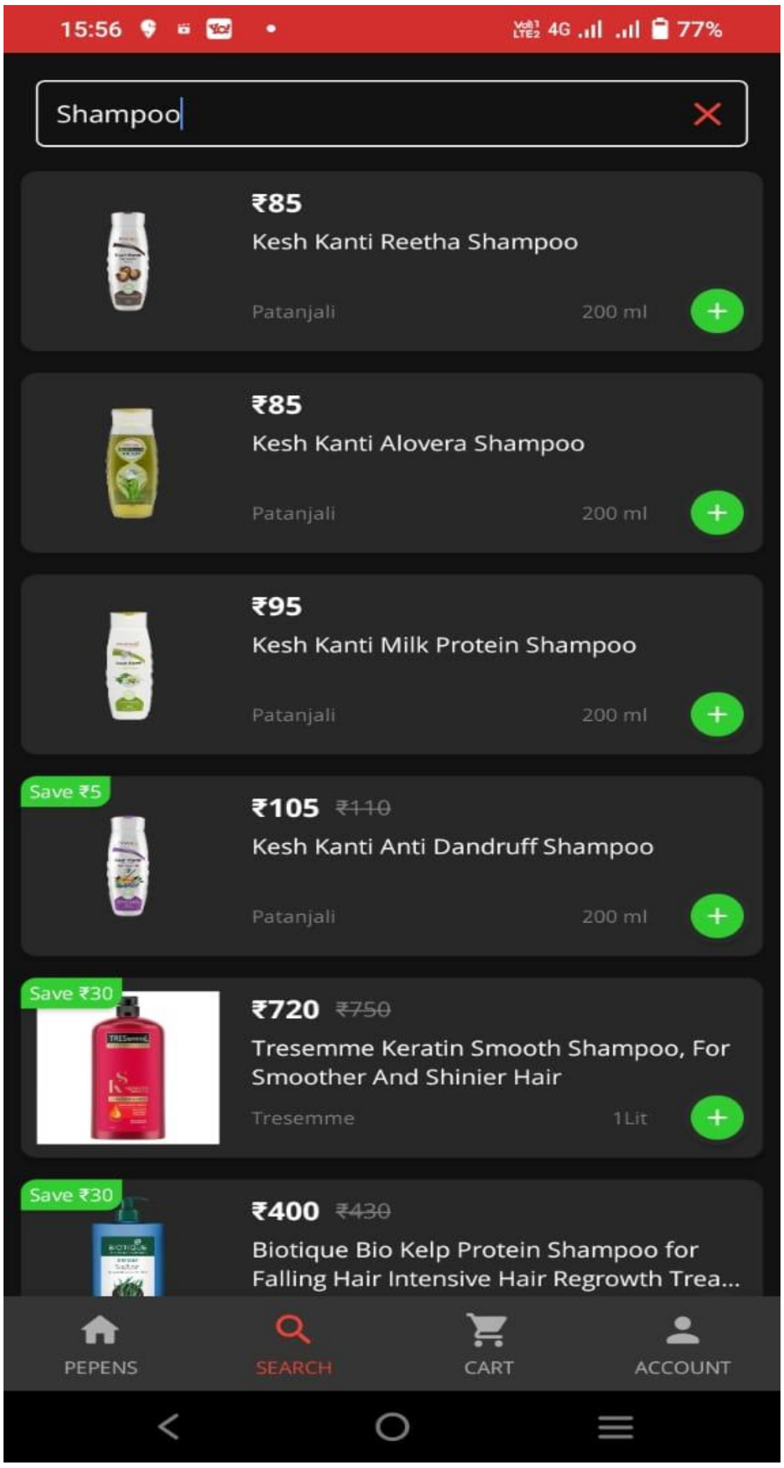
Hello Amit
Welcome to Pepens!
Best Saving
Free Delivery
Delivery in 13 minutes

Grocery & Staples
See More

PEPENS
SEARCH
CART
ACCOUNT

Fig. 4.3 : Home Page

8.4 SEARCH PAGE



15:56 4G 77%

Shampoo

₹85
Kesh Kanti Reetha Shampoo
Patanjali 200 ml

₹85
Kesh Kanti Alovera Shampoo
Patanjali 200 ml

₹95
Kesh Kanti Milk Protein Shampoo
Patanjali 200 ml

Save ₹5
₹105 ~~₹110~~
Kesh Kanti Anti Dandruff Shampoo
Patanjali 200 ml

Save ₹30
₹720 ~~₹750~~
Tresemme Keratin Smooth Shampoo, For Smoother And Shinier Hair
Tresemme 1Lit

Save ₹30
₹400 ~~₹430~~
Biotique Bio Kelp Protein Shampoo for Falling Hair Intensive Hair Regrowth Trea...

Fig. 4.4 : Search Page

CHAPTER 5

CODING

1. WORKINF (FRONT-END)

1.1. Scale_down.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <translate
      android:duration="300"
      android:fromYDelta="50%p"
      android:toYDelta="100%p" />
   <alpha
      android:duration="300"
      android:fromAlpha="1"
      android:interpolator="@android:anim/decelerate_interpolator"
      android:toAlpha="0" />
</set>
```

1.2. Scale_up.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <translate
      android:duration="300"
      android:fromYDelta="100%p"
      android:toYDelta="0%p" />
   <alpha
      android:duration="300"
      android:fromAlpha="0.2"
      android:interpolator="@android:anim/decelerate_interpolator"
      android:toAlpha="1" />
</set>
```

1.3. Scale_add.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:fillAfter="true">
   <translate
      android:duration="150"
      android:fromYDelta="-100%p"
      android:toYDelta="0%p" />
   <alpha
      android:duration="150"
      android:fromAlpha="0.3"
      android:interpolator="@android:anim/accelerate_interpolator"
      android:toAlpha="1.0" />
</set>
```

1.4. Scale_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:fillAfter="true">
   <translate
      android:duration="150"
      android:fromYDelta="100%p"
      android:toYDelta="0%p" />
   <alpha
      android:duration="150"
      android:fromAlpha="0.3"
      android:interpolator="@android:anim/accelerate_interpolator"
      android:toAlpha="1.0" />
</set>
```

1.5. Scale_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:fillAfter="true">
   <translate
      android:duration="150"
      android:fromYDelta="0%p"
      android:toYDelta="-100%p" />
```

```
    <alpha
        android:duration="150"
        android:fromAlpha="1.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toAlpha="0.3" />
</set>
```

1.6. Scale_remove.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:fillAfter="true">
    <translate
        android:duration="150"
        android:fromYDelta="0%p"
        android:toYDelta="100%p" />
    <alpha
        android:duration="150"
        android:fromAlpha="1.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toAlpha="0.3" />
</set>
```

2. DRAWABLE

2.1. a 1_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@color/firstColor" />
    <corners
        android:bottomLeftRadius="@dimen/eight_dp"
        android:topRightRadius="@dimen/eight_dp" />
</shape>
```

2.2. a 2_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
```

```
    <solid android:color="@color/secondColor" />
    <corners android:topightRadius="@dimen/eight_dp"
        android:bottomLeftRadius="@dimen/eight_dp"/>
</shape>
```

2.3. a 3_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@color/thirdColor" />
    <corners
        android:bottomLeftRadius="@dimen/eight_dp"
        android:topRightRadius="@dimen/eight_dp" />
</shape>
```

2.4. a 4_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@color/forthColor" />
    <corners
        android:bottomLeftRadius="@dimen/eight_dp"
        android:topRightRadius="@dimen/eight_dp" />
</shape>
```

2.5. a 5_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">

    <solid android:color="@color/fifthColor" />
    <corners
        android:bottomLeftRadius="@dimen/eight_dp"
        android:topRightRadius="@dimen/eight_dp" />

</shape>
```

2.6. a 6_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="@color/sixthColor" />
   <corners
      android:bottomLeftRadius="@dimen/eight_dp"
      android:topRightRadius="@dimen/eight_dp" />
</shape>
```

2.7. a 7_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="#dedddd" />
   <corners
      android:bottomLeftRadius="@dimen/eight_dp"
      android:topRightRadius="@dimen/eight_dp" />
</shape>
```

2.8. Accuracy_bg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="#99F06917" />
   <corners android:radius="@dimen/two_dp" />
</shape>
```

2.9. B1_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="@color/firstColor" />
   <corners
      android:bottomRightRadius="@dimen/eight_dp"
      android:topLeftRadius="@dimen/eight_dp" />
</shape>
```

2.10. B2_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <solid android:color="@color/secondColor" />
  <corners
    android:bottomRightRadius="@dimen/eight_dp"
    android:topLeftRadius="@dimen/eight_dp" />
</shape>
```

2.11. B3_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <solid android:color="@color/thirdColor" />
  <corners
    android:bottomRightRadius="@dimen/eight_dp"
    android:topLeftRadius="@dimen/eight_dp" />
</shape>
```

2.12. B4_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <solid android:color="@color/forthColor" />
  <corners
    android:bottomRightRadius="@dimen/eight_dp"
    android:topLeftRadius="@dimen/eight_dp" />
</shape>
```

2.13. C1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <stroke
    android:width="2dp"
    android:color="@color/grey" />
  <corners android:radius="1dp" />
</shape>
```

2.14. C1_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="@color/first1Color" />
   <corners android:radius="@dimen/thirty_dp" />
</shape>
```

2.15. C2_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="@color/second1Color" />
   <corners android:radius="@dimen/thirty_dp" />
</shape>
```

2.16. Cart_icon.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
   android:width="16dp"
   android:height="16dp"
   android:viewportWidth="200.391"
   android:viewportHeight="187.007">
   <path
      android:fillColor="#000000"
      android:pathData="M157.35,165.809m-21.199,0a21.199,21.199 0,1 1,42.398
0a21.199,21.199 0,1 1,-42.398 0" />
   <path
      android:fillColor="#000000"
      android:pathData="M72.555,165.809m-21.199,0a21.199,21.199 0,1 1,42.398
0a21.199,21.199 0,1 1,-42.398 0" />
   <path
      android:fillColor="#ff3a52"
      android:pathData="M199.327,26.868a4.239,4.239 0,0 0,-3.18 -
1.433L43.517,25.435a4.239,4.239 0,0 0,-4.155 5.088l16.959,84.795a4.592,4.592 0,0 0,4.714
3.392l127.192,-16.959a4.239,4.239 0,0 0,3.646 -3.68l8.479,-67.836A4.237,4.237 0,0 0,199.327
26.868Z" />
   <path
      android:fillColor="#fafafa"
      android:pathData="M72.554,101.907a4.24,4.24 0,0 1,-4.18 -3.544l-8.479,-50.877a4.24,4.24
0,0 1,8.361 -1.391l8.479,50.877a4.24,4.24 0,0 1,-3.485 4.876A4.076,4.076 0,0 1,72.554
101.907Z" />
   <path
      android:fillColor="#fafafa"
```

```
        android:pathData="M93.753,97.667a4.24,4.24 0,0 1,-4.24 -3.858l-4.24,-46.637a4.24,4.24 0,0
1,3.824 -4.618l0.034,0a4.24,4.24 0,0 1,4.6 3.841h0l4.24,46.637a4.241,4.241 0,0 1,-3.841 4.6h0Z"
/>
    <path
        android:fillColor="#fafafa"
        android:pathData="M114.952,93.427a4.24,4.24 0,0 1,-4.24 -4.24v-42.4a4.24,4.24 0,1 1,8.48
0v42.4A4.239,4.239 0,0 1,114.952 93.427Z" />
    <path
        android:fillColor="#fafafa"
        android:pathData="M136.15,89.188h-0.475a4.24,4.24 0,0 1,-3.77 -4.663c0,-0.014 0,-0.029
0,-0.043l4.24,-38.158a4.3,4.3 0,0 1,4.707 -3.773,4.241 4.241,0 0,1 3.748,4.68h0l-
4.24,38.158A4.24,4.24 0,0 1,136.15 89.188Z" />
    <path
        android:fillColor="#fafafa"
        android:pathData="M157.35,84.947a4.239,4.239 0,0 1,-4.113 -5.266l8.479,-
33.918a4.24,4.24 0,1 1,8.272 1.861c-0.014,0.064 -0.03,0.127 -0.048,0.191l-
8.479,33.918A4.239,4.239 0,0 1,157.35 84.947Z" />
    <path
        android:fillColor="#000000"
        android:pathData="M174.3,135.672L76.172,135.672a29.823,29.823 0,0 1,-29.18 -
23.853L26.272,8.48L4.251,8.48a4.24,4.24 0,1 1,0 -8.48L29.758,0a4.248,4.248 0,0 1,4.166
3.392L55.333,110.149a21.3,21.3 0,0 0,20.839 17.044L174.3,127.193a4.24,4.24 0,1 1,0 8.48Z" />
</vector>
```

2.17. Discount_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@android:color/holo_red_dark" />
    <stroke
        android:width="1dp"
        android:color="@android:color/white" />
    <corners android:radius="@dimen/twenty_dp" />
</shape>
```

2.18. Dot_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:width="@dimen/eight_dp"
        android:height="@dimen/eight_dp">
        <shape android:shape="rectangle">
            <solid android:color="#CCAAAAAA" />
            <corners android:radius="@dimen/four_dp" />
        </shape>
    </item>
</layer-list>
```

2.19. Dot_selected.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:width="@dimen/sixteen_dp"
        android:height="@dimen/eight_dp">
        <shape android:shape="rectangle">
            <solid android:color="#CCF06917" />
            <corners android:radius="@dimen/four_dp" />
        </shape>
    </item>
</layer-list>
```

2.20. Dot_blue__view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <stroke android:width="1.5dp"
        android:color="#1A237E"
        android:dashWidth="@dimen/four_dp"
        android:dashGap="@dimen/two_dp" />

    <solid android:color="#0D1A237E" />

    <corners android:radius="@dimen/four_dp" />

  </shape>
```

3. RAW MODULE

3.1. Map_style.xml

```
[
  {
    "featureType": "all",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#7c93a3"
      },
      {
        "lightness": "-10"
      }
    ]
  },
  {
    "featureType": "administrative.country",
    "elementType": "geometry",
    "stylers": [
      {

        "visibility": "on"
      }
    ]
  },
  {
    "featureType": "administrative.country",
    "elementType": "geometry.stroke",
    "stylers": [
      {
        "color": "#a0a4a5"
      }
    ]
  },
  {
    "featureType": "administrative.province",
    "elementType": "geometry.stroke",
    "stylers": [
      {
        "color": "#62838e"
      }
    ]
  },
{
    "featureType": "landscape",
    "elementType": "geometry.fill",
    "stylers": [
        {
            "color": "#dde3e3"
        }
    ]
},
{
    "featureType": "landscape.man_made",
    "elementType": "geometry.stroke",
    "stylers": [
        {
            "color": "#3f4a51"
        },
        {
            "weight": "0.30"
        }
    ]
},
{
    "featureType": "poi",
    "elementType": "all",
    "stylers": [
        {
            "visibility": "simplified"
        }
    ]
},
{
    "featureType": "poi.attraction",
    "elementType": "all",
    "stylers": [
        {
            "visibility": "on"
        }
    ]
},
{
    "featureType": "poi.business",
    "elementType": "all",
    "stylers": [
        {
          "visibility": "off"
        }
      ]
    },
    {
      "featureType": "poi.government",
      "elementType": "all",
      "stylers": [
        {
          "visibility": "off"
        }
      ]
    },
    {
      "featureType": "poi.park",
      "elementType": "all",
      "stylers": [
        {
          "visibility": "on"
        }
      ]
    },
    {
      "featureType": "poi.place_of_worship",
      "elementType": "all",
      "stylers": [
        {
          "visibility": "off"
        }
      ]
    },
    {
      "featureType": "poi.school",
      "elementType": "all",
      "stylers": [
        {
          "visibility": "off"
        }
      ]
    },
    {
      "featureType": "poi.sports_complex",
      "elementType": "all",
      "stylers": [
        {
          "visibility": "off"
        }
      ]
    },
    {
      "featureType": "road",
      "elementType": "all",
      "stylers": [
        {
          "saturation": "-100"
        },
        {
          "visibility": "on"
        }
      ]
    },
    {
      "featureType": "road",
      "elementType": "geometry.stroke",
      "stylers": [
        {
          "visibility": "on"
        }
      ]
    },
    {
      "featureType": "road.highway",
      "elementType": "geometry.fill",
      "stylers": [
        {
          "color": "#bbcacf"
        }
      ]
    },
    {
      "featureType": "road.highway",
      "elementType": "geometry.stroke",
      "stylers": [
        {
          "lightness": "0"
        },
        {
          "color": "#bbcacf"
        },
        {
          "weight": "0.50"
        }
      ]
    },
    {
      "featureType": "road.highway",
      "elementType": "labels",
      "stylers": [
        {
          "visibility": "on"
        }
      ]
    },
    {
      "featureType": "road.highway",
      "elementType": "labels.text",
      "stylers": [
        {
          "visibility": "on"
        }
      ]
    },
    {
      "featureType": "road.highway.controlled_access",
      "elementType": "geometry.fill",
      "stylers": [
        {
          "color": "#ffffff"
        }
      ]
    },
    {
      "featureType": "road.highway.controlled_access",
      "elementType": "geometry.stroke",
      "stylers": [
        {
          "color": "#a9b4b8"
        }
        ]
    },
    {
        "featureType": "road.arterial",
        "elementType": "labels.icon",
        "stylers": [
            {
                "invert_lightness": true
            },
            {
                "saturation": "-7"
            },
            {
                "lightness": "3"
            },
            {
                "gamma": "1.80"
            },
            {
                "weight": "0.01"
            }
        ]
    },
    {
        "featureType": "transit",
        "elementType": "all",
        "stylers": [
            {
                "visibility": "off"
            }
        ]
    },
    {
        "featureType": "water",
        "elementType": "geometry.fill",
        "stylers": [
            {
                "color": "#a3c7df"
            }
        ]
    }
]
```

4. VALUE MODULE

4.1. Array_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <array name="initial_slider_values">
      <item>0</item>
      <item>2000</item>
   </array>
</resources>
```

4.2. Colors_

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="colorPrimary">#f06917</color>
   <color name="colorPrimaryDark">#DD5300</color>
   <color name="colorAccent">#03DAC5</color>
   <color name="darkBlue">#1A237E</color>
   <color name="lightBlue">#3F51B5</color>
   <color name="blue400">#42A5F5</color>
   <color name="colorToolbar">#FFFFFF</color>
   <color name="colorText">#000000</color>
   <color name="colorButtonText">#FFFFFF</color>
   <color name="colorIcon">#000000</color>
   <color name="colorDialog">#FFFFFF</color>
   <color name="colorBG">#FFFFFF</color>
   <color name="semiTransparent">#D9FFFFFF</color>
   <color name="light">#EFEFEF</color>
   <color name="white">#FFFFFF</color>
   <color name="black">#000000</color>
   <color name="grey">#808080</color>
   <color name="lightGrey">#ADADAD</color>
   <color name="light_Grey">#D8D8D8</color>
   <color name="extraLightGrey">#EDEDED</color>
   <color name="green">#09AF00</color>
   <color name="greenDark">#0B8C04</color>
   <color name="red">#FF0000 </color>
   <color name="yellow">#FFC107</color>
   <color name="blue">#2196F3</color>
   <color name="pink">#FF367A</color>
   <color name="extraLightBlue">#D8EBFF</color>
   <color name="lightColor">#FFC4A1</color>
   <color name="darkBlueColor">#273173</color>
   <color name="orangeColor">#f17224</color>
   <color name="greenColor">#4eb549</color>
   <color name="pickDropColorDark">#830036</color>
   <color name="pickDropColorLight">#ff4343</color>
   <color name="firstColor">#F4D67E</color>
   <color name="first1Color">#fceab7</color>
   <color name="secondColor">#dd9be1</color>
   <color name="second1Color">#fac6fd</color>
   <color name="thirdColor">#d3e794</color>
   <color name="third1Color">#e0f4a2</color>
   <color name="forthColor">#a4a9f8</color>
   <color name="forth1Color">#c9ccfe</color>
   <color name="fifthColor">#f0ea46</color>
   <color name="fifth1Color">#fcf9a1</color>
   <color name="sixthColor">#92ca8d</color>
   <color name="sixth1Color">#bbefb6</color>
   <color name="sixth1Coor">#80BBEFB6</color>
</resources>
```

4.3. Dimens_

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <dimen name="zero_dp">0dp</dimen>
   <dimen name="eight_dp">8dp</dimen>
   <dimen name="sixteen_dp">16dp</dimen>
   <dimen name="twenty_four_dp">24dp</dimen>
   <dimen name="thirty_two_dp">32dp</dimen>
   <dimen name="one_twenty_dp">120dp</dimen>
   <dimen name="twenty_four_sp">24sp</dimen>
   <dimen name="sixteen_sp">16sp</dimen>
   <dimen name="twenty_five_dp">25dp</dimen>
   <dimen name="fifty_dp">50dp</dimen>
   <dimen name="four_dp">4dp</dimen>
   <dimen name="fourteen_sp">14sp</dimen>
   <dimen name="ten_dp">10dp</dimen>
   <dimen name="twenty_sp">20sp</dimen>
   <dimen name="forty_eight_dp">48dp</dimen>
   <dimen name="fifty_six_dp">56dp</dimen>
   <dimen name="eighty_dp">80dp</dimen>
```

```
    <dimen name="forty_dp">40dp</dimen>
    <dimen name="sixty_dp">60dp</dimen>
    <dimen name="twelve_dp">12dp</dimen>
    <dimen name="twelve_sp">12sp</dimen>
    <dimen name="hundred_dp">100dp</dimen>
    <dimen name="twenty_dp">20dp</dimen>
    <dimen name="ten_sp">10sp</dimen>
    <dimen name="fourteen_dp">14dp</dimen>
    <dimen name="two_dp">2dp</dimen>
    <dimen name="eight_sp">8sp</dimen>
    <dimen name="six_dp">6dp</dimen>
    <dimen name="eighteen_sp">18sp</dimen>
    <dimen name="seventy_dp">70dp</dimen>
    <dimen name="eighteen_dp">18dp</dimen>
    <dimen name="thirty_dp">30dp</dimen>
    <dimen name="ninety_dp">90dp</dimen>
    <dimen name="viewpager_next_item_visible">12dp</dimen>
    <dimen name="viewpager_current_item_horizontal_margin">12dp</dimen>
    <dimen name="thirty_five_dp">35dp</dimen>
    <dimen name="fifteen_dp">15dp</dimen>
    <dimen name="twenty_eight_dp">28dp</dimen>
    <dimen name="one_forty_dp">140dp</dimen>
</resources>
```

4.4. String_in.xml

```
<resources>
    <string name="app_name">Pepens</string>
    <string name="facebook_app_id">264059008531313</string>
    <string name="fb_login_protocol_scheme">fb264059008531313</string>
    <string name="google_map_api_key" translatable="false">AIzaSyBFncgd--
oUYT4qxcKJLHhQLlp4RMly1lk</string>
    <string name="maps_key">"AIzaSyC1VA3PLc12xYqtbgFUOujkMgAL6_FP_Jc"</string>
    <string name="map_id">11b3ca3269b8a2dd</string>
    <string name="direction_key">"AIzaSyBCgM9tjvcQGAYZCLY0H451bjS5lopnNIc"</string>
    <string name="phone_number">Mobile Number</string>
    <string name="resend">"Didn't get the code? Resend Code"</string>
    <string name="started">"Let's get started"</string>
    <string name="login_success">You logged in successfully.</string>
    <string name="no_internet">Something went wrong, please check your internet
connection.</string>
    <string name="try_again">Try Again</string>
    <string name="exit">Exit!</string>
    <string name="yes">Yes</string>
    <string name="no">No</string>
    <string name="exit_message">Do you want to exit from Pepens?</string>
    <string name="first_name">First Name</string>
    <string name="last_name">Last Name</string>
    <string name="live_in">Live In</string>
    <string name="dob">I was born on</string>
    <string name="camera">Camera</string>
    <string name="gallery">Gallery</string>
    <string name="feed">Feed</string>
    <string name="home">Home</string>
    <string name="account">Account</string>
    <string name="cart">Cart</string>
    <string name="offers">Offers</string>
    <string name="email">Email</string>
    <string name="address">Address</string>
    <string name="city">City</string>
    <string name="state">State</string>
    <string name="country">Country</string>
    <string name="zip">Zip Code</string>
    <string name="update_profile">Update Profile</string>
    <string name="record_not_found">Record not found.</string>
    <string name="term"><![CDATA[* Terms & Condition]]></string>
    <string name="term_condition"><![CDATA[Terms & Condition]]></string>
    <string name="refer_code">Refer Code(Optional)</string>
    <string name="profile">Profile</string>
    <string name="delivery_charges">Delivery Charges</string>
    <string name="taxes">Taxes and Charges</string>
    <!-- TODO: Remove or change this placeholder text -->
    <string name="hello_blank_fragment">Hello blank fragment</string>
</resources>
```

4.5. Styles_in.xml

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:textColorSecondary">@color/black</item>
        <item name="colorOnSecondary">@color/white</item>
    </style>
    <style name="RestaurantTheme"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <item name="colorPrimary">@color/lightBlue</item>
        <item name="colorPrimaryDark">@color/darkBlue</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:textColorSecondary">@color/black</item>
    </style>
    <style name="MySpinnerDatePickerStyle" parent="android:Theme.Material.Dialog">
        <item name="android:datePickerStyle">@style/MySpinnerDatePicker</item>
    </style>
    <style name="MySpinnerDatePicker" parent="android:Widget.Material.Light.DatePicker">
        <item name="android:datePickerMode">spinner</item>
    </style>
    <style name="BottomNavigationView" parent="@style/TextAppearance.AppCompat.Caption">
        <item name="android:textSize">@dimen/twelve_sp</item>
    </style>
    <style name="BottomNavigationView.Active"
parent="@style/TextAppearance.AppCompat.Caption">
        <item name="android:textSize">@dimen/twelve_sp</item>
    </style>
    <style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar"
/>
    <style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />
    <style name="ExpandedAppBar" parent="@android:style/TextAppearance.Medium">
        <item name="android:textSize">@dimen/eighteen_sp</item>
        <item name="android:textColor">#00000000</item>
        <item name="android:textStyle">bold</item>
    </style>
    <style name="CollapsedAppBar" parent="@android:style/TextAppearance.Medium">
        <item name="android:textSize">@dimen/eighteen_sp</item>
        <item name="android:textColor">@color/white</item>
        <item name="android:textStyle">bold</item>
    </style>
    <style name="Widget.MaterialComponents.TextInputLayout.OutlinedBox"
parent="Base.Widget.MaterialComponents.TextInputLayout">
        <item name="materialThemeOverlay">
            @style/ThemeOverlay.MaterialComponents.TextInputEditText.OutlinedBox
        </item>
        <item name="boxCollapsedPaddingTop">0dp</item>
    </style>
    <style name="ThemeOverlay.MaterialComponents.TextInputEditText.OutlinedBox">
        <item
name="editTextStyle">@style/Widget.MaterialComponents.TextInputEditText.OutlinedBox
        </item>
    </style>
    <style name="Widget.MaterialComponents.TextInputEditText.OutlinedBox"
parent="Base.Widget.MaterialComponents.TextInputEditText" />
    <style name="Base.Widget.MaterialComponents.TextInputEditText"
parent="Widget.AppCompat.EditText">
        <item name="android:background">@null</item>
        <item name="android:paddingStart">@dimen/eight_dp</item>
        <item name="android:paddingEnd">@dimen/eight_dp</item>
        <item name="android:textSize">@dimen/fourteen_sp</item>
        <item name="android:paddingLeft">@dimen/eight_dp</item>
        <item name="android:paddingRight">@dimen/eight_dp</item>
        <item name="android:paddingTop">@dimen/eight_dp</item>
        <item name="android:paddingBottom">@dimen/eight_dp</item>
        <item name="android:textAppearance">?attr/textAppearanceSubtitle1</item>
    </style>
    <!--    Dialog Animation Theme-->
    <style name="CustomDialogTheme" parent="@android:style/Theme.Dialog">
        <item name="android:windowAnimationStyle">@style/DialogAnimation</item>
    </style>
    <style name="DialogAnimation">
        <item name="android:windowEnterAnimation">@anim/scale_up</item>
        <item name="android:windowExitAnimation">@anim/scale_down</item>
    </style>
    <style name="Widget.App.PickDropButtonStyle"
parent="Widget.MaterialComponents.Button">
        <item name="materialThemeOverlay">@style/ButtonThemeOverlay</item>
    </style>
    <style name="ButtonThemeOverlay">
        <item name="colorPrimary">@color/pickDropColorDark</item>
    </style><style name="VegSwitch" parent="Theme.AppCompat.Light">
        <item name="colorControlActivated">@color/green</item>
        <item name="colorSwitchThumbNormal">#E6E6E6</item>
        <item name="android:colorForeground">#42221f1f</item>
    </style><style name="NonVegSwitch" parent="Theme.AppCompat.Light">
    <item name="colorControlActivated">@color/red</item><item
name="colorSwitchThumbNormal">#E6E6E6</item><item
name="android:colorForeground">#42221f1f</item>
    </style></resources>
```

5. WORKING (BACK-END)

5.1. Location_Activity

```
package com.app.pepens.activity
import android.Manifest
import android.app.Activity
import android.app.Dialog
import android.content.Context
import android.content.Intent
import android.content.IntentSender
import android.content.SharedPreferences
import android.content.pm.PackageManager
import android.graphics.Color
import android.graphics.drawable.ColorDrawable
import android.location.Address
import android.location.Geocoder
import android.location.Location
import android.location.LocationManager
import android.net.Uri
import android.os.Build
import android.os.Bundle
import android.os.Handler
import android.provider.Settings
import android.util.Log
import android.view.View
import android.view.Window
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.core.app.ActivityCompat
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.app.pepens.R
import com.app.pepens.adapter.ItemsAdapter
import com.app.pepens.fragment.HomeFragment
import com.app.pepens.fragment.PickAndDrop.PickAndDropFragment
import com.app.pepens.fragment.PlaceOrderFragment
import com.app.pepens.fragment.Restaurants.PlaceRestaurantOrderFragment
import com.app.pepens.fragment.Restaurants.RestaurantsFragment
import com.app.pepens.fragment.ShoppingFragment
import com.app.pepens.model.AvailableLocations.AvailableLocation
import com.app.pepens.model.CartItemsList
import com.app.pepens.model.Coordinates.GeoCoordinates
import com.app.pepens.model.ItemsList
import com.app.pepens.model.LogData
import com.app.pepens.model.Login.Login
import com.app.pepens.model.RestaurantCartList
import com.app.pepens.utils.Urls
import com.app.pepens.utils.Utils
import com.app.expensemanager.util.VolleySingleton
import com.android.volley.DefaultRetryPolicy
import com.android.volley.Request
import com.android.volley.toolbox.JsonObjectRequest
import com.android.volley.toolbox.StringRequest
import com.google.android.gms.common.ConnectionResult
import com.google.android.gms.common.api.GoogleApiClient
import com.google.android.gms.common.api.ResolvableApiException
import com.google.android.gms.location.*
import com.google.android.libraries.maps.CameraUpdateFactory
import com.google.android.libraries.maps.GoogleMap
import com.google.android.libraries.maps.OnMapReadyCallback
import com.google.android.libraries.maps.SupportMapFragment
import com.google.android.libraries.maps.model.CircleOptions
import com.google.android.libraries.maps.model.LatLng
import com.google.android.libraries.places.api.Places
import com.google.android.libraries.places.api.model.Place
import com.google.android.libraries.places.widget.Autocomplete
import com.google.android.libraries.places.widget.AutocompleteActivity
import com.google.android.libraries.places.widget.model.AutocompleteActivityMode
import com.google.android.material.dialog.MaterialAlertDialogBuilder
import com.google.gson.Gson
import kotlinx.android.synthetic.main.activity_location.*
import org.json.JSONObject
import kotlin.math.roundToInt
class LocationActivity : AppCompatActivity(), View.OnClickListener,
OnMapReadyCallback,
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
LocationListener {
private val TAG = LocationActivity::class.java.simpleName
private lateinit var sharedPreferences: SharedPreferences
private lateinit var progressDialog: Dialog
private var login: Login? = null
private var latitude: Double = 28.6139
private var longitude: Double = 77.2090
private var mMap: GoogleMap? = null
private var mapFragment: SupportMapFragment? = null
private var geoCoder: Geocoder? = null
private var addressList: List<Address>? = null
private var address: String? = null
private var partnerId: Int = 0
private var areaId: Int = 0
private var knownName: String? = null
private var locality: String? = null
private var zipcode: String? = null
private var city: String? = null
private var state: String? = null
private var country: String? = null
private val AUTOCOMPLETE_REQUEST_CODE = 1
private var zoom = 12F
private val REQUEST_CODE = 101
private val REQUEST_LOCATION = 102
private var mGoogleApiClient: GoogleApiClient? = null
private var location: Location? = null
private var locationRequest: LocationRequest? = null
private var locationManager: LocationManager? = null
private var oldAreaId = 0
private var accuracy = 0f
private var usingCurrentLocation = false
private var updateLocationPermission = false
private var availableLocation: AvailableLocation? = null
private var availableDialog: Dialog? = null
private var ivHeaderBack: ImageView? = null
private var tvHeader: TextView? = null
private var recyclerViewAvailable: RecyclerView? = null
private var availableArrayList: ArrayList<ItemsList> = ArrayList()
private var availableAdapter: ItemsAdapter? = null
companion object {
        var tvConnection: TextView? = null
    }
    fun dialog(value: Boolean) {
        tvConnection!!.visibility = View.VISIBLE
        if (value) {
            tvConnection!!.setBackgroundColor(Color.parseColor("#09AF00"))
            tvConnection!!.text = "Back Online."
            Handler().postDelayed({
                tvConnection!!.visibility = View.GONE
            }, 2000)
        } else {
            tvConnection!!.setBackgroundColor(Color.RED)
            tvConnection!!.text = "No connection."
        }
  }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_location)
//        Utils().restartAppAtCrash(this, LocationActivity::class.java.simpleName)
        tvConnection = findViewById(R.id.tv_connection)
        sharedPreferences = getSharedPreferences("USER_DATA",
Context.MODE_PRIVATE)
        progressDialog = Utils().progressDialog(this)
        mGoogleApiClient = GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build()
        Places.initialize(applicationContext, resources.getString(R.string.maps_key))
        geoCoder = Geocoder(this@LocationActivity)
        mapFragment = supportFragmentManager.findFragmentById(R.id.map) as
SupportMapFragment
        if (intent != null) {
            login = intent.getParcelableExtra("loginUser")
            Log.e(
                TAG,
                "onCreate: ${login!!.loginData!!.latitude}    ${login!!.loginData!!.longitude}"
            )
            if (login != null) {
```

```
            if (login!!.loginData!!.latitude != null && login!!.loginData!!.longitude != null) {
                if (!login!!.loginData!!.latitude.isNullOrEmpty() &&
!login!!.loginData!!.longitude.isNullOrEmpty() && login!!.loginData!!.latitude != "0" &&
login!!.loginData!!.longitude != "0" && login!!.loginData!!.latitude != "0.0" &&
login!!.loginData!!.longitude != "0.0") {
                    zoom = 12F
                    latitude = login!!.loginData!!.latitude!!.toDouble()
                    longitude =

login!!.loginData!!.longitude!!.toDouble()
                    areaId = login!!.loginData!!.areaId.toInt()
                    partnerId = login!!.loginData!!.partnerId
                    if (tvAccuracy != null) {
                        mapFragment!!.getMapAsync(this)
                    }
                } else {
                    checkLocation()
                }
            } else {
                checkLocation()
            }
        } else if (MainActivity.login != null) {
            zoom = 16F
            latitude = MainActivity.login!!.loginData!!.latitude!!.toDouble()
            longitude = MainActivity.login!!.loginData!!.longitude!!.toDouble()
            areaId = MainActivity.login!!.loginData!!.areaId.toInt()
            partnerId = MainActivity.login!!.loginData!!.partnerId
            if (tvAccuracy != null) {
                mapFragment!!.getMapAsync(this)
            }
        } else {
            checkLocation()
        }
    } else if (MainActivity.login != null) {
        zoom = 16F
        latitude = MainActivity.login!!.loginData!!.latitude!!.toDouble()
        longitude = MainActivity.login!!.loginData!!.longitude!!.toDouble()
        areaId = MainActivity.login!!.loginData!!.areaId.toInt()
        partnerId = MainActivity.login!!.loginData!!.partnerId
        if (tvAccuracy != null) {
            mapFragment!!.getMapAsync(this)
        }
    } else {
        checkLocation()
    }
    checkAvailableLocations()
    ivBack.setOnClickListener(this)
    ivAutoDetect.setOnClickListener(this)
    ivAutoDetectMSG.setOnClickListener(this)
    btnChange.setOnClickListener(this)
    btnAuto.setOnClickListener(this)
    tvAvailable.setOnClickListener(this)
    tvSave.setOnClickListener(this)
}
override fun onResume() {
    super.onResume()
    Utils().updatePagesLog(
        this,
        LogData(
            pageId = Urls.STARTUP_LOCATION_PAGE,
            title = "Update location",
            eventName = "Page Open"
        )
    )
    if (updateLocationPermission) {
        checkLocation()
    }
}
override fun onClick(v: View?) {
    when (v!!.id) {
        R.id.ivBack -> {
            finish()
        }
        R.id.ivAutoDetect -> {
            checkLocation()
        }
        R.id.ivAutoDetectMSG -> {
            checkLocation()
        }
        R.id.btnAuto -> {
            checkLocation()
        }
        R.id.btnChange -> {
            tvAccuracy.visibility = View.GONE
            progressBar.visibility = View.GONE
            usingCurrentLocation = false
            if (mGoogleApiClient!!.isConnected) {
                mGoogleApiClient!!.disconnect()
            }
            val fields = listOf(
                Place.Field.ID,
                Place.Field.NAME,
                Place.Field.LAT_LNG,
                Place.Field.ADDRESS,
                Place.Field.TYPES
            )
            val intent =
Autocomplete.IntentBuilder(AutocompleteActivityMode.FULLSCREEN, fields)
                .setCountry("IN").build(this)
            startActivityForResult(intent, AUTOCOMPLETE_REQUEST_CODE)
        }
        R.id.tvAvailable -> {
            tvAccuracy.visibility = View.GONE
            progressBar.visibility = View.GONE
            usingCurrentLocation = false
            if (mGoogleApiClient!!.isConnected) {
                mGoogleApiClient!!.disconnect()
            }
            if (availableLocation != null) {
                dialogAvailable()
            } else {
                Toast.makeText(
                    this,
                    "Currently we are not available in this location",
                    Toast.LENGTH_SHORT
                ).show()
            }
```

```
          }
          R.id.tvSave -> {
             if (areaId > 0 && partnerId > 0) {
                progressDialog.show()
                updatePinCode()
             } else {
                Toast.makeText(this, "Unable to save, please change or move location",
Toast.LENGTH_SHORT).show()
             }
          }
       }
    }
    private fun checkLocation() {
       tvErrorMsg.visibility = View.GONE
//        btnSave.isEnabled = false
       if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
          if (checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
             ActivityCompat.requestPermissions(
                this,
                arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
                REQUEST_CODE
             )
          } else {
             permission()
          }
       } else {
          permission()
       }
    }
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
       if (requestCode == AUTOCOMPLETE_REQUEST_CODE) {
          when (resultCode) {
             Activity.RESULT_OK -> {
                data?.let {
                   val place = Autocomplete.getPlaceFromIntent(data)
                   address = place.address
                   latitude = place.latLng!!.latitude
                   longitude = place.latLng!!.longitude
                   progressBar.visibility = View.GONE
```

```
                    tvAccuracy.visibility = View.GONE
                    usingCurrentLocation = false
                    progressDialog.show()
                    if (tvAccuracy != null) {
                        mapFragment!!.getMapAsync(this)
                    }
                }
            }
            AutocompleteActivity.RESULT_ERROR -> {
                data?.let {
                    val status = Autocomplete.getStatusFromIntent(data)
                    Log.e(TAG, status.statusMessage!!)
                }
            }
            Activity.RESULT_CANCELED -> {
                Log.e(TAG, "onActivityResult: Cancel")
            }
        }
        return
    } else if (requestCode == REQUEST_LOCATION) {
        when (resultCode) {
            Activity.RESULT_OK -> {
                usingCurrentLocation = true
                progressBar.visibility = View.VISIBLE
                if (mGoogleApiClient!!.isConnected) {
                    mGoogleApiClient!!.disconnect()
                }
                mGoogleApiClient!!.connect()
            }
            Activity.RESULT_CANCELED -> {
                MaterialAlertDialogBuilder(this)
                    .setTitle("Location Permission")
                    .setMessage("The app needs location permissions. Please grant location
permission to continue using the features of the app. Tap on 'YES' to grant permission
from setting.")
                    .setCancelable(false)
                    .setPositiveButton(R.string.yes) { dialog, id ->
                        checkLocation()
                    }.show()
            }
        }
    }
    super.onActivityResult(requestCode, resultCode, data)
 }
 override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
 ) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    when (requestCode) {
        REQUEST_CODE -> {
            if (grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                Log.e(TAG, "onRequestPermissionsResult: 1")
                permission()
            } else {
                Log.e(TAG, "onRequestPermissionsResult: 2")
                updateLocationPermission = false
                MaterialAlertDialogBuilder(this)
                    .setTitle("Location Permission")
                    .setMessage("The app needs location permissions. Please grant location
permission to continue using the features of the app. Tap on 'YES' to grant permission
from setting.")
                    .setCancelable(false)
                    .setPositiveButton(R.string.yes) { dialog, id ->
                        updateLocationPermission = true


startActivity(Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS).apply {
                        data = Uri.fromParts("package", "com.app.pepens", null)
                        })
                    }.show()
            }
        }
    }      }
```

5.2. Login_Activity

```
package com.app.pepens.activity
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.app.pepens.R
import com.app.pepens.fragment.NumberFragment
class LoginActivity : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_login)
    val fragment = NumberFragment()
    val fragmentManager = supportFragmentManager
    val fragmentTransaction = fragmentManager!!.beginTransaction()
    fragmentTransaction.replace(R.id.frameLayout, fragment)
    fragmentTransaction.addToBackStack(null)
    fragmentTransaction.commit()
  }
  override fun onBackPressed() {
    val fragmentManager = supportFragmentManager
    var fragment = supportFragmentManager.findFragmentById(R.id.frameLayout)
    when (fragment) {
      is NumberFragment -> {
        finish()
      }
      else -> {
        if (fragmentManager.backStackEntryCount > 0) {
          fragmentManager.popBackStack()
        } else {
          super.onBackPressed()
        }
      }
    }
  }

}
```

5.3. Main_Activity

```
package com.app.pepens.activity
import android.app.Activity
import android.content.*
import android.graphics.Color
import android.net.ConnectivityManager
import android.os.Build
import android.os.Bundle
import android.os.Handler
import android.provider.Settings
import android.util.Log
import android.view.View
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.NavigationUI
import androidx.navigation.ui.setupWithNavController
import com.app.pepens.BuildConfig
import com.app.pepens.R
import com.app.pepens.fragment.AddUpdateAddressFragment
import com.app.pepens.fragment.HomeFragment
import com.app.pepens.fragment.LocationFragment
import com.app.pepens.fragment.OfferFragment
import com.app.pepens.interfaces.DialogItemClick
import com.app.pepens.interfaces.NotificationUpdate
import com.app.pepens.model.CartItemsList
import com.app.pepens.model.Categories.Categories
import com.app.pepens.model.Login.Login
import com.app.pepens.model.RestaurantCartList
import com.app.pepens.utils.AnalyticsApplication
import com.app.pepens.utils.CheckConnectivity
import com.app.pepens.utils.Urls
import com.app.pepens.utils.Utils
import com.app.expensemanager.util.VolleySingleton
import com.android.volley.DefaultRetryPolicy
import com.android.volley.Request
import com.android.volley.toolbox.JsonObjectRequest
import com.android.volley.toolbox.StringRequest
import com.google.android.gms.analytics.Tracker
import com.google.android.material.badge.BadgeDrawable
import com.google.android.material.bottomnavigation.BottomNavigationView
import com.google.firebase.iid.FirebaseInstanceId
import com.google.gson.Gson
import org.json.JSONObject
import java.util.*
import kotlin.collections.HashMap
```

```
open class MainActivity : AppCompatActivity(), NotificationUpdate, DialogItemClick {
  private var TAG = MainActivity::class.java.simpleName
  private lateinit var sharedPreferences: SharedPreferences
  private var mNetworkReceiver: BroadcastReceiver? = null
  private var badgeDrawable: BadgeDrawable? = null
  private var navHostFragment: NavHostFragment? = null
  companion object {
    var mTracker: Tracker? = null
    var id = 0
    var showDialog = false
    var login: Login? = null
    var bottomNavigationView: BottomNavigationView? = null
    var tvConnection: TextView? = null
    var cartArrayList: ArrayList<CartItemsList> = ArrayList()
    var restaurantCartList: ArrayList<RestaurantCartList> = ArrayList()
    var adFromR = false
    var categories: Categories? = null
  }
  fun dialog(value: Boolean) {
    if (showDialog) {
      tvConnection!!.visibility = View.VISIBLE
      if (value) {
        tvConnection!!.setBackgroundColor(Color.parseColor("#09AF00"))
        tvConnection!!.text = "Back Online."
        Handler().postDelayed({
          tvConnection!!.visibility = View.GONE
        }, 2000)
      } else {
        tvConnection!!.setBackgroundColor(Color.RED)
        tvConnection!!.text = "No connection."
      }
    }
  }
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    Utils().restartAppAtCrash(this, MainActivity::class.java.simpleName)
    mNetworkReceiver = CheckConnectivity()
    registerReceiver(mNetworkReceiver,
IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION))
    sharedPreferences = getSharedPreferences("USER_DATA", Context.MODE_PRIVATE)
    if (intent != null) {
      login = intent.getParcelableExtra("loginUser")
```

```
        }
        val application: AnalyticsApplication = application as AnalyticsApplication
        mTracker = application.defaultTracker
        if (SplashActivity.checkLoginDetails) {
            getLoginUser()
        }
        bottomNavigationView = findViewById(R.id.bottomNavigationView)
        navHostFragment =
            supportFragmentManager.findFragmentById(R.id.fragmentContainerView) as
NavHostFragment
        bottomNavigationView!!.setupWithNavController(navHostFragment!!.navController)
        tvConnection = findViewById(R.id.tv_connection)
        bottomNavigationView!!.setOnNavigationItemSelectedListener {
            when (it.itemId) {
                R.id.homeFragment -> {
                    HomeFragment.apiDataReceived = false
                    navHostFragment!!.navController.navigate(R.id.homeFragment)
                }
                R.id.offerFragment -> {
                    OfferFragment.apiDataReceived = false
                    navHostFragment!!.navController.navigate(R.id.offerFragment)
                }
                R.id.mainDialogFragment -> {
                    navHostFragment!!.navController.navigate(R.id.mainDialogFragment)
                }
                R.id.feedFragment -> {
                    navHostFragment!!.navController.navigate(R.id.feedFragment)
                }
                R.id.notificationFragment -> {
                    navHostFragment!!.navController.navigate(R.id.notificationFragment)
                }
            }
            true
        }
        badgeDrawable = bottomNavigationView!!.getOrCreateBadge(R.id.notificationFragment)
        badgeDrawable!!.backgroundColor = Color.RED
        badgeDrawable!!.badgeTextColor = Color.WHITE
        badgeDrawable!!.maxCharacterCount = 2
        getMainCategories()
        Handler().postDelayed({
            showDialog = true
            if (Utils().verifyAvailableNetwork(this)) {
                updateDevice()
            }
        }, 3000)
    }
    override fun updateNotification(visible: Boolean, count: Int) {
        if (visible) {
            badgeDrawable!!.number = count
            badgeDrawable!!.isVisible = true
        } else {
            badgeDrawable!!.number = count
            badgeDrawable!!.isVisible = false
        }
    }
    override fun itemDialogClick(id: Int) {
        when (id) {
            Urls.ID_RESTAURANT_CATEGORY -> {
                navHostFragment!!.navController.navigate(R.id.restaurantsFragment)
            }
            Urls.ID_GROCERY_CATEGORY -> {
                var bundle = Bundle()
                bundle.putInt("BusinessCategoryId", id)
                navHostFragment!!.navController.navigate(R.id.shoppingFragment, bundle)
            }
            Urls.ID_PICK_DROP_CATEGORY -> {
                navHostFragment!!.navController.navigate(R.id.pickAndDropFragment)
            }
            else -> {
                var bundle = Bundle()
                bundle.putInt("BusinessCategoryId", id)
                navHostFragment!!.navController.navigate(R.id.shoppingFragment, bundle)
            }
        }
    }
    private fun getLoginUser() {
        val request = StringRequest(Request.Method.GET,
            Urls.GET_USER_ON_STARTUP + sharedPreferences.getString("USER_ID", "0") +
Urls.TOKEN,
            { response ->
                Log.e(TAG, "getLoginUser: $response")
                val jsonObject = JSONObject(response.toString())
                if (jsonObject.getBoolean("IsSuccess")) {
                    login = Gson().fromJson(response.toString(), Login::class.java)
                }
            }, {
```

```
            Log.e(TAG, "getLoginUser: $it")
        })
        request.retryPolicy = DefaultRetryPolicy(10000, 0, 1f)
        VolleySingleton.getInstance(this).addToRequestQueue(request)
    }
    private fun updateDevice() {
        val params = HashMap<String, String>()
        params["AppDeviceId"] = "0"
        params["ToId"] = sharedPreferences.getString("USER_ID", "0").toString()
        params["ToType"] = "4"
        params["DeviceId"] = Settings.Secure.getString(this.contentResolver,
Settings.Secure.ANDROID_ID)
        params["AppVersionNumber"] = BuildConfig.VERSION_CODE.toString()
        params["AppVersionName"] = BuildConfig.VERSION_NAME
        params["DeviceOS"] = "Android"
        params["DeviceOSVersion"] = Build.VERSION.SDK
        params["DeviceModel"] = Build.MODEL
        params["DeviceManufacturer"] = Build.MANUFACTURER
        params["DeviceName"] = "${Build.MANUFACTURER} - ${Build.MODEL}"
        params["IPAddress"] = Utils().getMobileIPAddress().toString()
        params["FirebaseToken"] = FirebaseInstanceId.getInstance().token.toString()
        val jsonObject = JSONObject(params as Map<*, *>)
        Log.e(TAG, "updateDevice: $jsonObject")
        val request = JsonObjectRequest(
            Request.Method.POST,
            Urls.AddUpdateDevice,
            jsonObject,
            { response ->
                Log.e("TAG", "updateDevice: $response")
            },
            {
                Log.e("TAG", "updateDevice: $it")
            })
        request.retryPolicy = DefaultRetryPolicy(10000, 0, 1f)
        VolleySingleton.getInstance(this).addToRequestQueue(request)
    }
    override fun onSupportNavigateUp(): Boolean {
        return NavigationUI.navigateUp(navHostFragment!!.navController, null)
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        when (requestCode) {
            102 -> {
                when (resultCode) {
                    Activity.RESULT_OK -> {
                        val navHostFragment: Fragment? =
                            supportFragmentManager.primaryNavigationFragment
                        val fragment: Fragment = navHostFragment!!.childFragmentManager.fragments[0]
                        if (fragment is AddUpdateAddressFragment) {
                            (fragment as AddUpdateAddressFragment).gpsEvent(true)
                        } else if (fragment is LocationFragment) {
                            (fragment as LocationFragment).gpsEvent(true)
                        }
//                        val fragment: AddUpdateAddressFragment =
supportFragmentManager.findFragmentById(R.id.frameLayout) as AddUpdateAddressFragment
//                        fragment.gpsEvent(true)
                    }
                    else -> {
                        val navHostFragment: Fragment? =
                            supportFragmentManager.primaryNavigationFragment
                        val fragment: Fragment = navHostFragment!!.childFragmentManager.fragments[0]
                        if (fragment is AddUpdateAddressFragment) {
                            (fragment as AddUpdateAddressFragment).gpsEvent(false)
                        } else if (fragment is LocationFragment) {
                            (fragment as LocationFragment).gpsEvent(false)
                        }
//                        val fragment: AddUpdateAddressFragment =
supportFragmentManager.findFragmentById(R.id.frameLayout) as AddUpdateAddressFragment
//                        fragment.gpsEvent(false)
                    }
                }
            }
        }
    }
    override fun onDestroy() {
        super.onDestroy()
        unregisterNetworkChanges()
    }
    private fun unregisterNetworkChanges() {
        try {
            unregisterReceiver(mNetworkReceiver)
        } catch (e: IllegalArgumentException) {
            e.printStackTrace()
        }
    }
```

```
   fun getMainCategories() {
      val request = StringRequest(
         Request.Method.GET,
         Urls.GET_HOME_CATEGORIES + login!!.loginData!!.areaId + Urls.USER_ID +
sharedPreferences.getString(
            "USER_ID",
            "0"
         ),
         { response ->
            Log.e(TAG, "getMainCategories: $response")
            if (bottomNavigationView != null) {
               val jsonObject = JSONObject(response.toString())
               categories = if (jsonObject.has("IsSuccess") && jsonObject.getBoolean("IsSuccess")) {
                     Gson().fromJson(response.toString(), Categories::class.java)
                  } else {
                     null
                  }
               categories!!.categoriesData!!.removeAll { item -> item.businessCategoryId !=
Urls.ID_GROCERY_CATEGORY }
            }
         },
         {
            Log.e(TAG, "getMainCategories: $it")
         })
      request.retryPolicy = DefaultRetryPolicy(10000, 2, 1f)
      VolleySingleton.getInstance(this).addToRequestQueue(request)
   }

}
```

5.4. Payment_Activity

```
package com.app.pepens.activity
import android.content.Context
import android.content.SharedPreferences
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.app.pepens.R
import com.app.pepens.fragment.PaymentsFragment
import com.app.pepens.model.OrderSuccess.OrderSuccess
import com.razorpay.Checkout
import com.razorpay.PaymentData
import com.razorpay.PaymentResultWithDataListener
import org.json.JSONObject
class PaymentActivity : AppCompatActivity(), PaymentResultWithDataListener {
  private var TAG = PaymentActivity::class.java.simpleName
  private lateinit var sharedPreferences: SharedPreferences
  private var orderSuccess: OrderSuccess? = null
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_payment)
    sharedPreferences = getSharedPreferences("USER_DATA", Context.MODE_PRIVATE)
    Checkout.preload(applicationContext)
    orderSuccess = PaymentsFragment.orderSuccess
    startPayment()
  }
  private fun startPayment() {
    val co = Checkout()
    co.setKeyID("${MainActivity.login!!.loginData!!.razorpayKeyId}")
//      co.setKeyID("rzp_test_59GnziRh4QKO6p")
    co.setImage(R.drawable.a2z_logo)
    try {
      val options = JSONObject()
      options.put("name", getString(R.string.app_name))
      options.put("description", "")
      options.put("image", "https://s3.amazonaws.com/rzp-mobile/images/rzp.png")
      options.put("currency", "INR")
      options.put("order_id",
"${orderSuccess!!.orderSuccessData!!.paymentGateway!!.paymentGatewayOrderId}")
      options.put("amount",

"${orderSuccess!!.orderSuccessData!!.paymentGateway!!.paymentGatewayAmount}")
      val preFill = JSONObject()
      preFill.put("email", "${sharedPreferences.getString("EMAIL", "")}")
      preFill.put("contact", "${sharedPreferences.getString("COUNTRY_CODE",
"")}${sharedPreferences.getString("MOBILE_NO", "")}")
      options.put("prefill", preFill)
      co.open(this@PaymentActivity, options)
    } catch (e: Exception) {
      Log.e(TAG, "startPayment: ${e.message}")
      Toast.makeText(this@PaymentActivity, "StartPayment: " + e.message,
Toast.LENGTH_LONG).show()
    }
    }
    override fun onPaymentError(errorCode: Int, response: String?, paymentData: PaymentData?)
{
        try {
            Log.e(TAG, "onPaymentError: $errorCode  $response")
            Toast.makeText(this, "$response", Toast.LENGTH_LONG).show()
            val jsonObject = JSONObject()
            val jsonObject1 = JSONObject()
            jsonObject1.put("PaymentGatewayOrderId",
orderSuccess!!.orderSuccessData!!.paymentGateway!!.paymentGatewayOrderId)
            jsonObject1.put("PaymentGatewayErrorCode", errorCode)
            jsonObject1.put("PaymentGatewayErrorDescription", response)
            jsonObject1.put("PaymentGateway", "Razorpay")
            jsonObject.put("PaymentGateway", jsonObject1)
            jsonObject.put("OrderId", orderSuccess!!.orderSuccessData!!.orderId)
            jsonObject.put("UserId", sharedPreferences.getString("USER_ID", null))
            Log.e(TAG, "$jsonObject")
            PaymentsFragment.paymentSuccess = false
            PaymentsFragment.afterRequest = true
            PaymentsFragment.jsonObjectData = jsonObject
            finish()
        } catch (e: Exception) {
            Log.e(TAG, "onPaymentError: -- $e")
        }
    }

    override fun onPaymentSuccess(razorpayPaymentId: String?, paymentData: PaymentData) {
        try {
            Log.e(TAG, "onPaymentSuccess: $razorpayPaymentId

 ${paymentData.signature}")
//          Toast.makeText(this, "Payment Successful", Toast.LENGTH_LONG).show()
            val jsonObject = JSONObject()
            val jsonObject1 = JSONObject()
            jsonObject1.put("PaymentGatewayPaymentId", razorpayPaymentId)
            jsonObject1.put("PaymentGatewayOrderId",
orderSuccess!!.orderSuccessData!!.paymentGateway!!.paymentGatewayOrderId)
            jsonObject1.put("PaymentGatewaySignature", paymentData.signature)
            jsonObject1.put("PaymentGateway", "Razorpay")
            jsonObject.put("PaymentGateway", jsonObject1)
            jsonObject.put("OrderId", orderSuccess!!.orderSuccessData!!.orderId)
            jsonObject.put("UserId", sharedPreferences.getString("USER_ID", null))
            Log.e(TAG, "$jsonObject")
            PaymentsFragment.paymentSuccess = true
            PaymentsFragment.afterRequest = false
            PaymentsFragment.jsonObjectData = jsonObject
            finish()
        } catch (e: Exception) {
            Log.e(TAG, "onPaymentSuccess: -- $e")
        }
    }

}
```

6. TEST MODULE CLASS

6.1. Addresses_Adaptor

```
package com.app.pepens.adapter
import android.content.Context
import android.graphics.Color
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.app.pepens.R
import com.app.pepens.activity.MainActivity
import com.app.pepens.fragment.AddressListFragment
import com.app.pepens.model.Address.Addresses
import com.app.pepens.model.Address.AddressData
import com.google.android.material.button.MaterialButton
import kotlinx.android.synthetic.main.custom_addresses_layout.view.*
class AddressesAdapter : RecyclerView.Adapter<AddressesAdapter.ViewHolder> {
    private val context: Context
    private val addresses: Addresses
    private val clickEvent: ClickEvent
    constructor(
        context: Context,
        addresses: Addresses,
        addressListFragment: AddressListFragment
    ) : super() {
        this.context = context
        this.addresses = addresses
        this.clickEvent = addressListFragment
    }
    interface ClickEvent {
        fun editAddress(position: Int)
        fun deleteAddress(position: Int)
        fun deliverToAddress(position: Int)
    }
    override fun onCreateViewHolder(viewGroup: ViewGroup, position: Int): ViewHolder {
        return ViewHolder(
            LayoutInflater.from(context).inflate(
                R.layout.custom_addresses_layout,
                viewGroup,
                false
            )
        )
    }
    override fun getItemCount(): Int {
        return if (addresses.addressData == null) {
            0
        } else {
            addresses.addressData.size
        }
    }
    override fun onBindViewHolder(viewHolder: ViewHolder, position: Int) {
        viewHolder.tvType.text = addresses.addressData[position].addressType
        var add = ""
        if (!addresses.addressData[position].firstName.isNullOrEmpty())
            add += "${addresses.addressData[position].firstName}"
        if (!addresses.addressData[position].lastName.isNullOrEmpty())
            add += " ${addresses.addressData[position].lastName}"
        if (!addresses.addressData[position].addressLine1.isNullOrEmpty())
            add += "\n${addresses.addressData[position].addressLine1}, "
        if (!addresses.addressData[position].addressLine2.isNullOrEmpty())
            add += "${addresses.addressData[position].addressLine2}"
//        if (!addresses.addressData[position].landmark.isNullOrEmpty())
//            add += "\nLandmark: ${addresses.addressData[position].landmark}"
        if (!addresses.addressData[position].city.isNullOrEmpty())
            add += "\n${addresses.addressData[position].city}"
        if (!addresses.addressData[position].zipcode.isNullOrEmpty())
            add += "(${addresses.addressData[position].zipcode})"
        if (!addresses.addressData[position].state.isNullOrEmpty())
            add += ", ${addresses.addressData[position].state}"
        if (!addresses.addressData[position].country.isNullOrEmpty())
```

```
            add += ", ${addresses.addressData[position].country}"
        viewHolder.tvAddress.text = add
        if (addresses.addressData[position].isDefault) {
            viewHolder.tvDefault.visibility = View.VISIBLE
        } else {
            viewHolder.tvDefault.visibility = View.GONE
        }
        if (AddressListFragment.goBackForAddress) {
            if (addresses.addressData[position].areaId !=
MainActivity.login!!.loginData!!.areaId.toInt()) {
                viewHolder.tvType.setTextColor(Color.parseColor("#888888"))
                viewHolder.tvAddress.setTextColor(Color.parseColor("#888888"))
                viewHolder.tvDefault.setTextColor(Color.parseColor("#888888"))
            }
        }
        viewHolder.btnEdit.setOnClickListener {
            clickEvent.editAddress(position)
        }
        viewHolder.btnDelete.setOnClickListener {
            clickEvent.deleteAddress(position)
        }
        viewHolder.itemView.setOnClickListener {
            clickEvent.deliverToAddress(position)
        }
    }
    fun removeItem(position: Int) {
        addresses.addressData.removeAt(position)
        notifyItemRemoved(position)
    }
    fun restoreItem(item: AddressData, position: Int) {
        addresses.addressData.add(position, item)
        notifyItemInserted(position)
    }
    fun getData(): ArrayList<AddressData> {
        return addresses.addressData
    }
    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val tvDefault: TextView = view.tvDefault
        val tvType: TextView = view.tvType
        val tvAddress: TextView = view.tvAddress
        val btnEdit: MaterialButton = view.btnEdit
        val btnDelete: MaterialButton = view.btnDelete
    }           }
```

6.2. Ads_Pager_Adapter

```
package com.app.pepens.adapter
import android.content.Context
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import androidx.navigation.Navigation
import androidx.navigation.findNavController
import androidx.recyclerview.widget.RecyclerView
import com.app.pepens.R
import com.app.pepens.activity.MainActivity
import com.app.pepens.model.Ads.Ads
import com.app.pepens.model.LogData
import com.app.pepens.utils.Urls
import com.app.pepens.utils.Utils
import com.bumptech.glide.Glide
import kotlinx.android.synthetic.main.custom_ads_layout.view.*
/**
 * Created by Vikas Kumar Singh on 10/07/20.
 */
class AdsPagerAdapter : RecyclerView.Adapter<AdsPagerAdapter.ViewHolder> {
    private var context: Context
    private var ads: Ads
    constructor(context: Context, ads: Ads) : super() {
        this.context = context
        this.ads = ads
    }
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        return ViewHolder(
            LayoutInflater.from(context).inflate(R.layout.custom_ads_layout, parent, false)
        )
    }
    override fun getItemCount(): Int {
        return if (ads!!.adsData == null) {
            0
        } else {
            ads!!.adsData!!.count()
        }
    }
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
```

```
    var sharedPreferences = context!!.getSharedPreferences("USER_DATA",
Context.MODE_PRIVATE)
    Glide.with(context!!)
      .load(ads!!.adsData!![position].coverPictureUrl)
      .into(holder.ivImage)
    holder.itemView.setOnClickListener {
      if (ads!!.adsData!![position].linkTypeId == 1) {

        when (Navigation.findNavController(it).currentDestination?.id) {
          R.id.homeFragment -> {
            Utils().updatePagesLog(
              context,
              LogData(
                areaId = MainActivity.login!!.loginData!!.areaId,
                pageId = Urls.HOME_PAGE,
                title = "${ads!!.adsData!![position].title}",
                eventName = "Ad Click",
                pageEntityId = "${ads!!.adsData!![position].entityId}"
              )
            )
          }
          R.id.shoppingFragment -> {
            Utils().updatePagesLog(
              context,
              LogData(
                areaId = MainActivity.login!!.loginData!!.areaId,
                pageId = Urls.SHOPPING_PAGE,
                title = "${ads!!.adsData!![position].title}",
                eventName = "Ad Click",
                pageEntityId = "${ads!!.adsData!![position].entityId}"
              )
            )
          }
          R.id.restaurantsFragment -> {
            Utils().updatePagesLog(
              context,
              LogData(
                areaId = MainActivity.login!!.loginData!!.areaId,
                pageId = Urls.RESTAURANT_PAGE,
                title = "${ads!!.adsData!![position].title}",
                eventName = "Ad Click",
                pageEntityId = "${ads!!.adsData!![position].entityId}"
              )
        )
    }
}
when (ads!!.adsData!![position].entityType) {
    1 -> {
        Utils().clickStream(
            context,
            sharedPreferences.getString("USER_ID", "0")!!,
            "16",
            "${ads!!.adsData!![position].adId}",
            "2",
            "${ads!!.adsData!![position].adId}",
            ""
        )
        var bundle = Bundle()
        bundle.putInt("businessId", ads!!.adsData!![position].entityId)
        bundle.putString("businessName", "")
        when (Navigation.findNavController(it).currentDestination?.id) {
            R.id.homeFragment -> {
                it.findNavController().navigate(
                    R.id.action_homeFragment_to_categoryBusinessDetailsFragment,
                    bundle
                )
            }
            R.id.shoppingFragment -> {
                it.findNavController().navigate(
                    R.id.action_shoppingFragment_to_categoryBusinessDetailsFragment,
                    bundle
                )
            }
            R.id.restaurantsFragment -> {
                it.findNavController().navigate(
                    R.id.action_restaurantsFragment_to_categoryBusinessDetailsFragment,
                    bundle
                )
            }
        }
    }
    2 -> {
        Utils().clickStream(
            context,
            sharedPreferences.getString("USER_ID", "0")!!,
            "7",
        "${ads!!.adsData!![position].adId}",
        "2",
        "${ads!!.adsData!![position].adId}",
        ""
    )
    var bundle = Bundle()
    bundle.putInt("offerId", ads!!.adsData!![position].entityId)
    when (Navigation.findNavController(it).currentDestination?.id) {
        R.id.homeFragment -> {
            it.findNavController()
                .navigate(
                    R.id.action_homeFragment_to_offerDetailsFragment,
                    bundle
                )
        }
        R.id.shoppingFragment -> {
            it.findNavController()
                .navigate(
                    R.id.action_shoppingFragment_to_offerDetailsFragment,
                    bundle
                )
        }
        R.id.restaurantsFragment -> {
            it.findNavController()
                .navigate(
                    R.id.action_restaurantsFragment_to_offerDetailsFragment,
                    bundle
                )
        }
    }
}
3 -> {
    Utils().clickStream(
        context,
        sharedPreferences.getString("USER_ID", "0")!!,
        "6",
        "${ads!!.adsData!![position].adId}",
        "2",
        "${ads!!.adsData!![position].adId}",
        ""
    )
    var bundle = Bundle()
    bundle.putInt("id", ads!!.adsData!![position].entityId)
    when (Navigation.findNavController(it).currentDestination?.id) {
        R.id.homeFragment -> {
            it.findNavController()
                .navigate(
                    R.id.action_homeFragment_to_feedDetailsFragment,
                    bundle
                )
        }
        R.id.shoppingFragment -> {
            it.findNavController()
                .navigate(
                    R.id.action_shoppingFragment_to_feedDetailsFragment,
                    bundle
                )
        }
        R.id.restaurantsFragment -> {
            it.findNavController()
                .navigate(
                    R.id.action_restaurantsFragment_to_feedDetailsFragment,
                    bundle
                )
        }
    }
}
4 -> {
    when (ads!!.adsData!![position].entityId) {
        1 -> {
        }
        2 -> {
            when (Navigation.findNavController(it).currentDestination?.id) {
                R.id.homeFragment -> {
                    it.findNavController()
                        .navigate(R.id.action_homeFragment_to_referralFragment)
                }
                R.id.shoppingFragment -> {
                    it.findNavController()
                        .navigate(R.id.action_shoppingFragment_to_referralFragment)
                }
                R.id.restaurantsFragment -> {
                    it.findNavController()
                        .navigate(R.id.action_restaurantsFragment_to_referralFragment)
                }
            }
            }
            3 -> {
                when (Navigation.findNavController(it).currentDestination?.id) {
                    R.id.homeFragment -> {
                        it.findNavController()
                            .navigate(R.id.action_homeFragment_to_rewardsFragment)
                    }
                    R.id.shoppingFragment -> {
                        it.findNavController()
                            .navigate(R.id.action_shoppingFragment_to_rewardsFragment)
                    }
                    R.id.restaurantsFragment -> {
                        it.findNavController()
                            .navigate(R.id.action_restaurantsFragment_to_rewardsFragment)
                    }
                }
            }
            4 -> {
                when (Navigation.findNavController(it).currentDestination?.id) {
                    R.id.homeFragment -> {
                        it.findNavController()
                            .navigate(R.id.action_homeFragment_to_shoppingFragment)
                    }
                    R.id.shoppingFragment -> {

                    }
                    R.id.restaurantsFragment -> {
                        it.findNavController()
                            .navigate(R.id.action_restaurantsFragment_to_shoppingFragment)
                    }
                }
            }
        }
    }
    5 -> {
        if (ads!!.adsData!![position].entityId == 0) {
            if (!MainActivity.adFromR) {
                when (Navigation.findNavController(it).currentDestination?.id) {
                    R.id.homeFragment -> {
                        it.findNavController()
                            .navigate(R.id.action_homeFragment_to_restaurantsFragment)
                    }
                    R.id.shoppingFragment -> {
```

```
                                it.findNavController()
                                    .navigate(R.id.action_shoppingFragment_to_restaurantsFragment)
                            }
                            R.id.restaurantsFragment -> {

                            }
                        }
                    }
                } else {
                    var bundle = Bundle()
                    bundle.putInt("businessId", ads!!.adsData!![position].entityId)
                    bundle.putString("businessName", "")
                    bundle.putInt("itemId", 0)
                    when (Navigation.findNavController(it).currentDestination?.id) {
                        R.id.homeFragment -> {
                            it.findNavController().navigate(
                                R.id.action_homeFragment_to_restaurantDetailsFragment,
                                bundle
                            )
                        }
                        R.id.shoppingFragment -> {
                            it.findNavController().navigate(
                                R.id.action_shoppingFragment_to_restaurantDetailsFragment,
                                bundle
                            )
                        }
                        R.id.restaurantsFragment -> {
                            it.findNavController().navigate(
                                R.id.action_restaurantsFragment_to_restaurantDetailsFragment,
                                bundle
                            )
                        }
                    }
                }
            }
        }
    } else if (ads!!.adsData!![position].linkTypeId == 2) {
        val defaultBrowser = Intent.makeMainSelectorActivity(Intent.ACTION_MAIN,
Intent.CATEGORY_APP_BROWSER)
        defaultBrowser.data = Uri.parse(ads!!.adsData!![position].linkUrl)
        context!!.startActivity(defaultBrowser)
    }
}
```

```
    }

    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val ivImage: ImageView = view.ivImage
    }

}
```

6.3. All_Categories_Adapter

```
package com.app.pepens.adapter
import android.content.Context
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.navigation.Navigation
import androidx.navigation.findNavController
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.app.pepens.R
import com.app.pepens.fragment.ShoppingFragment
import com.app.pepens.model.ItemCategories.ItemCategories
import com.app.pepens.utils.Utils
import kotlinx.android.synthetic.main.custom_all_categories_layout.view.*
class AllCategoriesAdapter : RecyclerView.Adapter<AllCategoriesAdapter.ViewHolder> {
    private var context: Context
    private var itemCategories: ItemCategories
    constructor(context: Context, itemCategories: ItemCategories) {
        this.context = context
        this.itemCategories = itemCategories
    }
    override fun onCreateViewHolder(viewGroup: ViewGroup, position: Int): ViewHolder {
        return ViewHolder(
            LayoutInflater.from(context)
                .inflate(R.layout.custom_all_categories_layout, viewGroup, false)
        )
    }
    override fun getItemCount(): Int {
        return if (itemCategories!!.itemCategoriesData == null) {
        } else {
            itemCategories!!.itemCategoriesData!!.size
        }
    }
    override fun onBindViewHolder(viewHolder: ViewHolder, position: Int) {
        var sharedPreferences = context.getSharedPreferences("USER_DATA",
Context.MODE_PRIVATE)
        viewHolder.tvTitle.text = "Shop by

${itemCategories!!.itemCategoriesData!![position].itemCategoryName}"

        if (itemCategories!!.itemCategoriesData!![position].maxDiscountPercentage <= 0) {
            viewHolder.tvDiscount.visibility = View.GONE
        } else {
            viewHolder.tvDiscount.visibility = View.VISIBLE
            viewHolder.tvDiscount.text = "Upto
${itemCategories!!.itemCategoriesData!![position].maxDiscountPercentage}% OFF"
        }
        viewHolder.recyclerView.layoutManager = GridLayoutManager(context, 3)
        var categoriesAdapter = AllSubCategoriesAdapter(
            context,
            itemCategories.itemCategoriesData!![position].itemCategories!!
        )
        viewHolder.recyclerView.adapter = categoriesAdapter

        viewHolder.tvViewAll.setOnClickListener {
            Utils().clickStream(
                context,
                sharedPreferences.getString("USER_ID", "0")!!,
                "25",
                "${itemCategories!!.itemCategoriesData!![position].itemCategoryId}",
                "23",
                "${itemCategories!!.itemCategoriesData!![position].itemCategoryId}",
                ""
            )
            var bundle = Bundle()
            bundle.putInt("BusinessCategoryId", ShoppingFragment.shoppingBusinessCategoryId)
            bundle.putInt("CategoryId",
itemCategories!!.itemCategoriesData!![position].itemCategoryId)
            bundle.putInt("SubCategoryId", 0)
            if (Navigation.findNavController(it).currentDestination?.id == R.id.shoppingFragment)

it.findNavController().navigate(R.id.action_shoppingFragment_to_shoppingProductsFragment,
bundle)
```

```
        }
    }
    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val tvTitle: TextView = view.tvTitle
        val tvViewAll: TextView = view.tvViewAll
        val tvDiscount: TextView = view.tvDiscount
        val recyclerView: RecyclerView = view.recyclerView
    }

}
```

6.4. All_Product_Adapter

```
package com.app.pepens.adapter
import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.RecyclerView
import com.app.pepens.R
import com.app.pepens.fragment.AllProductsFragment
import com.app.pepens.model.ProductsWithCategory.BusinessItem
import kotlinx.android.synthetic.main.custom_business_items_layout.view.*
class AllProductsAdapter : RecyclerView.Adapter<AllProductsAdapter.ViewHolder> {
    private var context: Context
    private var businessItems: ArrayList<BusinessItem>
    private var clickEvent: ClickEvent
    constructor(
        context: Context,
        businessItems: ArrayList<BusinessItem>,
        allProductsFragment: AllProductsFragment
    ) {
        this.context = context
        this.businessItems = businessItems
        this.clickEvent = allProductsFragment
    }
    interface ClickEvent {
        fun viewBrands(itemId: BusinessItem)
    }
    override fun onCreateViewHolder(viewGroup: ViewGroup,

position: Int): ViewHolder {
        return ViewHolder(
            LayoutInflater.from(context)
                .inflate(R.layout.custom_business_items_layout, viewGroup, false)
        )
    }
    override fun getItemCount(): Int {
        return businessItems.count()
    }
    override fun onBindViewHolder(viewHolder: ViewHolder, position: Int) {
        viewHolder.tvTitle.text = businessItems!![position].item!!.itemName
        if (businessItems!![position].inStock) {
            viewHolder.ivCheck.setColorFilter(
                ContextCompat.getColor(context, R.color.green),
                android.graphics.PorterDuff.Mode.SRC_IN
            )
            viewHolder.tvTitle.setTextColor(context.resources.getColor(R.color.colorText))
            viewHolder.tvBrands.setTextColor(context.resources.getColor(R.color.green))
            viewHolder.tvBrands.setBackgroundResource(R.drawable.view_brand_active)
        } else {
            viewHolder.ivCheck.setColorFilter(
                ContextCompat.getColor(context, R.color.grey),
                android.graphics.PorterDuff.Mode.SRC_IN        )
            viewHolder.tvTitle.setTextColor(context.resources.getColor(R.color.grey))
            viewHolder.tvBrands.setTextColor(context.resources.getColor(R.color.grey))
            viewHolder.tvBrands.setBackgroundResource(R.drawable.view_brand_inactive)       }
        if (businessItems!![position].noOfBrands > 0) {
            viewHolder.tvBrands.visibility = View.VISIBLE
        } else {
            viewHolder.tvBrands.visibility = View.GONE        }
        viewHolder.tvBrands.setOnClickListener {
//          if (businessItems[position].inStock){
            clickEvent.viewBrands(businessItems[position])
//          } } }
    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val ivCheck: ImageView = view.ivCheck
        val tvTitle: TextView = view.tvTitle
        val tvBrands: TextView = view.tvBrands
    }}
```

CHAPTER 6

FUTURE SCOPE OF PROJECT

The Grocery Management System is an innovative project that aims to streamline and automate various aspects of grocery management, including inventory management, billing, customer tracking, and delivery management.

As technology continues to advance and consumer expectations evolve, the future scope of the project holds immense potential for growth and development. Here are some key areas where the project can expand and thrive:

1. **Mobile Applications and Online Shopping:** With the increasing popularity of mobile devices and the growth of e-commerce, developing dedicated mobile applications and online shopping platforms will be crucial for the success of the Grocery Management System. This will enable customers to browse and order groceries from the comfort of their homes, leading to improved customer convenience and satisfaction.

2. **Integration with IoT Devices:** The Internet of Things (IoT) presents exciting opportunities for the grocery industry. Integrating the Grocery Management System with IoT devices can provide real-time data on product availability, expiration dates, and freshness. Smart refrigerators and shelves can automatically track inventory levels and generate orders when items are running low. This integration will significantly reduce manual effort and improve overall efficiency.

3. **Artificial Intelligence and Machine Learning:** Leveraging the power of artificial intelligence (AI) and machine learning (ML) algorithms can enhance the Grocery Management System in multiple ways. AI-powered recommendation engines can suggest personalized grocery lists based on customer preferences and previous purchases. ML algorithms can analyze sales data to forecast demand, optimize pricing strategies, and prevent stockouts. These technologies can also be used to automate inventory replenishment and improve supply chain management.

4. **Big Data Analytics:** The Grocery Management System generates vast amounts of data on a daily basis, including sales figures, customer profiles, and purchasing patterns. By implementing robust big data analytics, businesses can extract valuable insights from this data. Analysis of customer behavior can help in targeted marketing campaigns, loyalty programs, and customized offers. Additionally, analyzing inventory data can lead to better procurement decisions, reducing wastage, and optimizing storage space.

5. **Integration with Third-Party Services:** Collaborating with external service providers can expand the functionality of the Grocery Management System. Integration with payment gateways enables secure and seamless transactions. Partnership with logistics and delivery services allows for efficient order fulfillment and tracking. Moreover, integrating loyalty programs and coupon services can attract and retain customers, fostering long-term relationships.

6. **Sustainability and Green Initiatives:** As environmental concerns continue to grow; the Grocery Management System can incorporate sustainability practices into its operations. This may include promoting reusable packaging, offering eco-friendly alternatives, and minimizing food waste through smart inventory management. By aligning with sustainability goals, the project can attract environmentally conscious customers and contribute to a greener future.

7. **Expansion into New Markets:** The Grocery Management System has the potential for expansion beyond its initial target market. By adapting the system to different regions and cultures, businesses can tap into new markets and reach a wider customer base. Localization efforts, such as supporting multiple languages and currencies, can help in global expansion. Additionally, catering to specialized dietary requirements and offering a diverse range of products can attract niche markets.

In conclusion, the future scope of the Grocery Management System is vast and promising. Embracing mobile applications, integrating with IoT devices, leveraging AI and ML, implementing big data analytics, collaborating with third-party services, focusing on sustainability, and expanding into new markets are key avenues for growth. By continually innovating and adapting to evolving consumer needs, the Grocery Management System can revolutionize the grocery industry, enhancing efficiency, convenience, and customer satisfaction.

CHAPTER 7

CONCLUSION

Basically, the development and implementation of the Grocery Management System have greatly improved the efficiency and effectiveness of grocery store operations. The system has successfully addressed the challenges faced by traditional manual methods, such as inventory management, sales tracking, and customer satisfaction. Through the use of modern technologies and streamlined processes, the system has significantly enhanced the overall management and productivity of the grocery store.

One of the key benefits of the Grocery Management System is its ability to automate and streamline inventory management. With real-time updates on stock levels and automated reordering processes, the system ensures that the store always has the right products in stock. This not only minimizes the risk of stockouts but also reduces inventory carrying costs. The system also allows for better demand forecasting and analysis, enabling store owners to make informed decisions regarding purchasing and pricing strategies.

The sales tracking and reporting functionalities of the system have provided valuable insights into the store's performance. By analyzing sales data, store owners can identify popular products, track sales trends, and make data-driven decisions. This information helps in optimizing product placement, marketing strategies, and promotions, ultimately leading to increased sales and profitability. Moreover, the system allows for personalized customer data management, facilitating targeted marketing campaigns and loyalty programs that enhance customer satisfaction and retention.

The Grocery Management System has also improved customer convenience and satisfaction. The implementation of a user-friendly interface, online ordering, and home delivery options have made shopping easier and more convenient for customers. The system allows customers to browse the store's inventory, place orders, and make payments from the comfort of their homes, saving them time and effort. Additionally, the system has introduced features such as personalized recommendations based on past purchases, enabling a personalized shopping experience. These improvements have contributed to higher customer satisfaction levels and increased loyalty towards the store. The system has also streamlined the operational processes within the grocery store. By automating routine tasks such as billing, invoicing, and employee management, the system has reduced manual errors and improved efficiency. The time saved by automating these processes can be utilized by the store staff to focus on providing better customer service and attending to other important tasks. Additionally, the system has facilitated seamless integration with other systems, such as accounting software and supplier databases, further enhancing operational efficiency and accuracy.

Furthermore, the Grocery Management System has improved data security and privacy. With the implementation of robust security measures, such as user authentication, data encryption, and regular backups, the system ensures that sensitive information remains protected. This instills confidence in both the store owners and customers, knowing that their data is secure.

In conclusion, the implementation of the Grocery Management System has revolutionized the way grocery stores operate. By automating various processes, improving inventory management, enhancing customer satisfaction, and streamlining operations, the system has proved to be a game-changer in the industry. The benefits of increased efficiency, accurate data analysis, improved customer experience, and enhanced security make it an essential tool for any modern grocery store. With the continuous advancements in technology, the Grocery Management System can be further improved and customized to meet the evolving needs of the industry. It is undoubtedly a valuable asset for grocery store owners who seek to stay ahead in the competitive market and provide exceptional service to their customers.

REFREENCES

1. Pantanowitz L, Valenstein PN, Evans AJ, et al. Review of the current state of whole slide imaging in pathology. J Pathol Inform. 2011;2:36. doi:10.4103/2153-3539.83746
2. Gilbertson JR, Ho J, Anthony L, Jukic DM, Yagi Y, Parwani AV. Primary histologic diagnosis using automated whole slide imaging: a validation study. BMC Clin Pathol. 2006;6:4. doi:10.1186/1472-6890-6-4
3. Gilbertson JR, Patel AA, Yagi Y, et al. Digital imaging in pathology: the case for standardization. J Telemed Telecare. 2005;11(3):109-116. doi:10.1258/1357633054068983
4. Nielsen PS, Lindebjerg J, Rasmussen J, Starklint H, Waldstrøm M, Nielsen B. Virtual microscopy: an evaluation of its validity and diagnostic performance in routine histologic diagnosis of skin tumors. Hum Pathol. 2010;41(12):1770-1776. doi:10.1016/j.humpath.2010.07.001
5. Bauer TW, Schoenfield L, Slaw RJ, Yerian L, Sun Z, Henricks WH. Validation of whole slide imaging for primary diagnosis in surgical pathology. Arch Pathol Lab Med. 2013;137(4):518-524. doi:10.5858/arpa.2012-0114-OA
6. Kim GH, Kim HS, Oh YL, Choi YL, Kang GH. Automated slide scanning systems can increase efficiency in performing manual microdissection for molecular analysis. J Pathol Inform. 2016;7:49. doi:10.4103/2153-3539.194781
7. Vodovnik A. The automated slide analysis workflow: a combination of automated microscopy and image analysis for high-throughput analysis of histology slides. J Pathol Inform. 2019;10:27. doi:10.4103/jpi.jpi_11_19
8. Al-Janabi S, Huisman A, Vink A, Leguit RJ, Offerhaus GJ, Ten Kate FJ. Whole slide images for primary diagnostics of gastrointestinal tract pathology: a feasibility study. Hum Pathol. 2012;43(5):702-707. doi:10.1016/j.humpath.2011.07.011