ST. TERESA’S COLLEGE (AUTONOMOUS), ERNAKULAM

AFFILIATED TO MAHATMA GANDHI UNIVERSITY

PROJECT REPORT

A Subscription-Based Grocery Shopping App

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF SCIENCE IN COMPUTER APPLICATIONS [TRIPLE MAIN]

Submitted By:

HESSA NASARU

III B.Sc. Computer Applications [Triple Main]

Register No: SB22CA018

Under the guidance of

Ms. Raji S Pillai

Department of Computer Applications

2022 - 2025

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

AFFILIATED TO MAHATMA GANDHI UNIVERSITY

CERTIFICATE

This is to certify that the project report entitled **"A Subscription-Based Grocery Shopping App"** is a bona fide record of the work done by **HESSA NASARU** (SB22CA018) during the year 2022– 2025 and submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer Applications (Triple Main) under Mahatma Gandhi University, Kottayam.

[illegible] 17/03/2025
Head of the Department

[illegible] RAJI S PILLAI
Internal Examiner

[illegible] 17/3/'24
External Examiner
Dr. Reseena mol. N.A

Date: 14/03/25

DECLARATION

I, HESSA NASARU (SB22CA018), B.Sc. Computer Applications [Triple Main] student of St. Teresa's College (Autonomous), Ernakulam, hereby declare that the project submitted for Bachelor's Degree in Computer Application is my original work. I further declare that the said work has not previously been submitted to any other university or academic body.

Date: 17|03|2025

Place: Ernakulam

Hessa

HESSA NASARU

ACKNOWLEDGEMENT

I would like to convey my heartfelt gratitude to **Rev. Dr. Sr. Nilima (CSST) Manager, Directors Rev. Sr. Tessa (CSST), Rev. Sr. Francis Ann (CSST) and Principal Dr. Alphonsa Vijaya Joseph** for providing me with this wonderful opportunity to work on a project on subscription-based grocery shopping app

I would like to express my profound gratitude to the Head of the Department of Computer Applications Ms. Sheeba Emmanuel Dennis and my project guide Ms. Raji S Pilla**i (Assistant Professor)** and all other faculty of the department for their contributions to the completion of my project. The completion of the project would not have been possible without their help and insights.

I would also like to Thank my project guide at LCC Computer Education Institute.

Finally, I take this opportunity to Thank all those who have directly or indirectly helped me with my project.

HESSA NASARU

ABSTRACT

Designed to simplify the handling of household essentials, Grocify is an innovative subscription-based grocery shopping app. Users can sign up for goods on a daily, weekly, monthly, or yearly basis through the app, thus removing the need for manual reordering. Grocify sets several subscription plans for automatically restocking products at the given intervals so that customers never run of their regular necessary groceries.

Automating the grocery shopping experience through Grocify helps reduce the difficulty of stocking regular supplies and increase the overall convenience. People can quickly browse through a broad selection of goods, include them in their subscription plans, and get their purchases delivered without the need of many buys. The software also guarantees continuous inventory monitoring, therefore offering people with a stress-free shopping experience.

Grocify turns grocery shopping into a smart, automatic, and time-saving experience ideal for busy people and families thanks to its emphasis on speed and user comfort. Future improvements would help to make grocery management simpler than ever by means of improved personalization, automation, and accessibility.

CONTENTS

1.INTRODUCTION

1.1. About

Day by day, given the increasing busy lifestyles of people, monitoring grocery items manually has become challenging. Frequently running out of supplies or forgetting essential items leads to inconvenience and disrupts routines.
Our grocery shopping platform "Grocify" enables users to automate grocery shopping monitoring their preferences and establishing their required monthly or yearly items. Our app employs technology to ensure personalized and timely delivery of grocery items without user’s constant manual engagement.

The application's front-end design and user interface utilizes Java and XML, and developed using Java Android for mobile devices. For server-side business logic and back-end functionality we employ Java Server Pages (JSP). The data is managed and stored using SQL database.

1.2. Objectives Of the Project

The objectives of the project are outlined below:

- To enhance the efficiency of ordering and managing groceries.
- To offer a subscription service that is both automated and customized to the user's preferences.
- To enhance customer ease by providing essentials directly to users' doorsteps pr omptly.
- The project is gaining importance due to the growing demand for automating routine tasks and growing interest in subscription services.

Grocify aims to simplify the management of home essentials while providing a seamle ss user experience.

2.SYSTEM ANALYSIS

2.1. Introduction

Grocify app is a subscription-based grocery shopping platform created to elevate shopping experience to next level. We can automate grocery shopping with our app on customer's requirement basis. Through this analysis, we hope to create a reliable and user-friendly application by addressing the drawbacks of the current system model and therefore suggesting an alternative measurement.

2.2. Existing System

While today's grocery shopping system is modernized at the convenience of online shopping, they often do not provide the automation and subscription models necessary to reduce customer input on a daily basis. Whenever the users want to restock their grocery supplies, they have to go through the tedious task of browsing products listings, add those items to cart and order them. This can be time-consuming and inefficient, especially for people who lead a busy life or older people.

2.3. Proposed System

The proposed application "Grocify", is a grocery shopping app based on a subscription model where the users can create their required model of grocery list items thus aiming to modify the management of household groceries. The users can monitor their past orders and automatically receive items without manual actions further. This solution thus improves user convenience by offering customized and timely delivery.

2.4. System Requirements

1. Hardware Requirements

- **Processor:** Intel Core i3 10th Generation or above
- **RAM:** 8 GB or above

Storage (ROM): 10 GB or above

- **Drive:** Solid State Drive (SSD) for faster data processing and app responsiveness

- **Display:** 1366 x 768 resolution or higher (recommended for development and testing)
- **Network:** Stable internet connection for seamless server communication and data synchronization

2. Software Requirements

- **Operating System:**
  - For Development: Windows 10 or above / Linux-based OS
  - For Mobile Devices: Android 6.0 (Marshmallow) or above
- **Development Tools:**
  - **Java Development Kit (JDK):** Version 8 or above
  - **NetBeans IDE:** Version 8.2 (for Java and JSP development)
  - **Android Studio:** Latest stable version (for Android app development)
  - **SQL Yog:** For managing SQL databases
- **Server-Side Requirements:**
  - **JSP (Java Server Pages):** For dynamic web content and server-side scripting
  - **Apache Tomcat Server:** Version 9.0 or above (recommended for running JSP applications)
  - **SQL Database:** MySQL or compatible relational database system

2.5. Languages Or Software Packages

1. Programming Languages

- **Java:**
  - Used for Android app development, handling the core logic, functionalities, and user interactions.
  - Provides platform independence and robust performance for mobile applications.

- **XML (Extensible Markup Language):**
  - Utilized for designing the app’s user interface (UI) in Android Studio.
  - Defines the layout, structure, and styling of screens, ensuring a responsive and intuitive UI.
- **JSP (Java Server Pages):**
  - Implemented on the server side to create dynamic web content and manage server-side operations.
  - Facilitates communication between the mobile app and the database, handling data requests and responses.
- **SQL (Structured Query Language):**
  - Used for managing and manipulating the relational database.
  - Handles data storage, retrieval, updates, and administrative tasks for user subscriptions, product catalogues, and order management.

2. Software Packages and Development Tools

- **NetBeans IDE 8.2:**
  - An integrated development environment (IDE) used for Java and JSP development.
  - Provides code editing, debugging, and project management features.
- **Android Studio:**
  - The official IDE for Android app development.
  - Offers tools for building, testing, and debugging Android applications, along with an emulator for device simulation.
- **SQL Yog:**
  - A graphical interface tool for managing MySQL databases.
  - Simplifies database administration, query execution, and data visualization.

3.SYSTEM DESIGN

3.1. Introduction

System design is a crucial stage in creation of our subscription-based grocery app. This stage establishes the framework on how the system will work, communicate. In order to attain a flawless user experience, we need to satisfy both functional and non-functional needs therefore this stage focuses on the application's architecture, components, modules, data flow, and general structure. Also, in order to meet various stakeholder's needs such as users, administrators, and delivery agents the design process will make sure the system is scalable, effective, and user-friendly. Here we will contain models to show how database, server-side elements and app's front-end UI interact.

3.2. Data Flow Diagrams

Data Flow Diagrams (DFDs) shows how the information flow within the application depicting the movement between processes, data stores and external entities. DFDs helps in understanding system's functionality and data handling at different levels.

- **Level 0 DFD:** High-level overview of the entire system.
- **Level 1 DFD:** Detailed view showing the breakdown of main processes.

3.2.1 LEVEL O DFD

This level provides a high-level overview of the entire system, showing how external entities interact with the system.

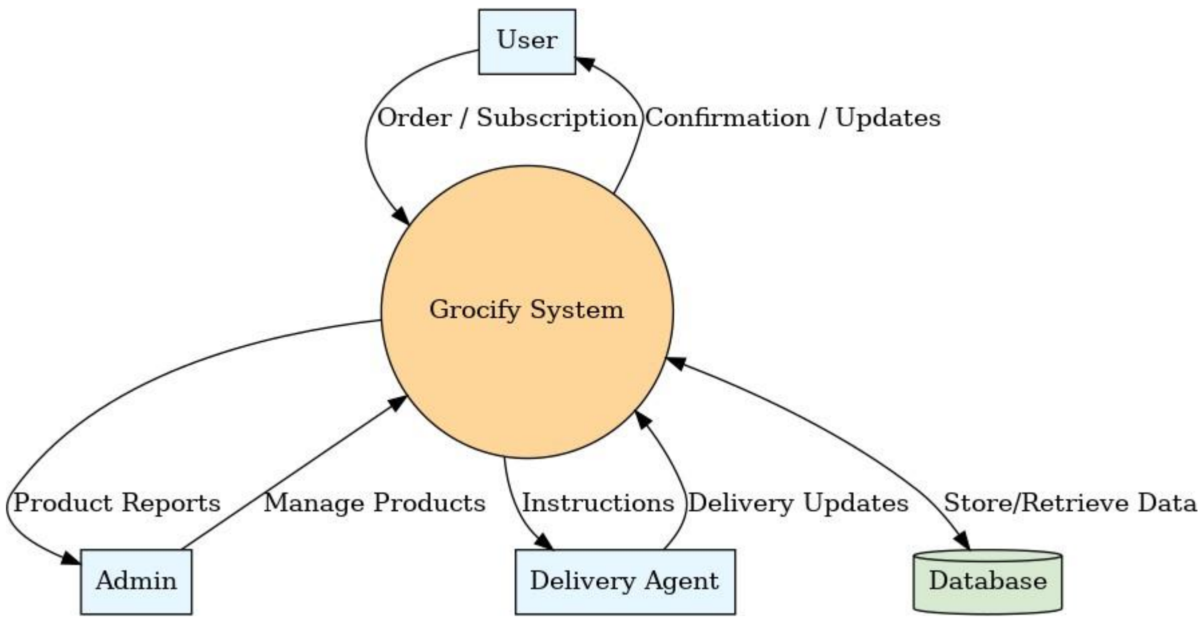
External Entities:

- **User:** Places orders, manages subscriptions.
- **Admin:** Manages products, monitors orders, updates stock.
- **Delivery Agent:** Updates delivery status.

Processes:

- **Grocify System:** Handles user subscriptions, product management, and order processing.

Database: Stores user data, product information, subscription details, and order history.



3.2.2. LEVEL 1 DFD

These breaks down the main system into more detailed processes:

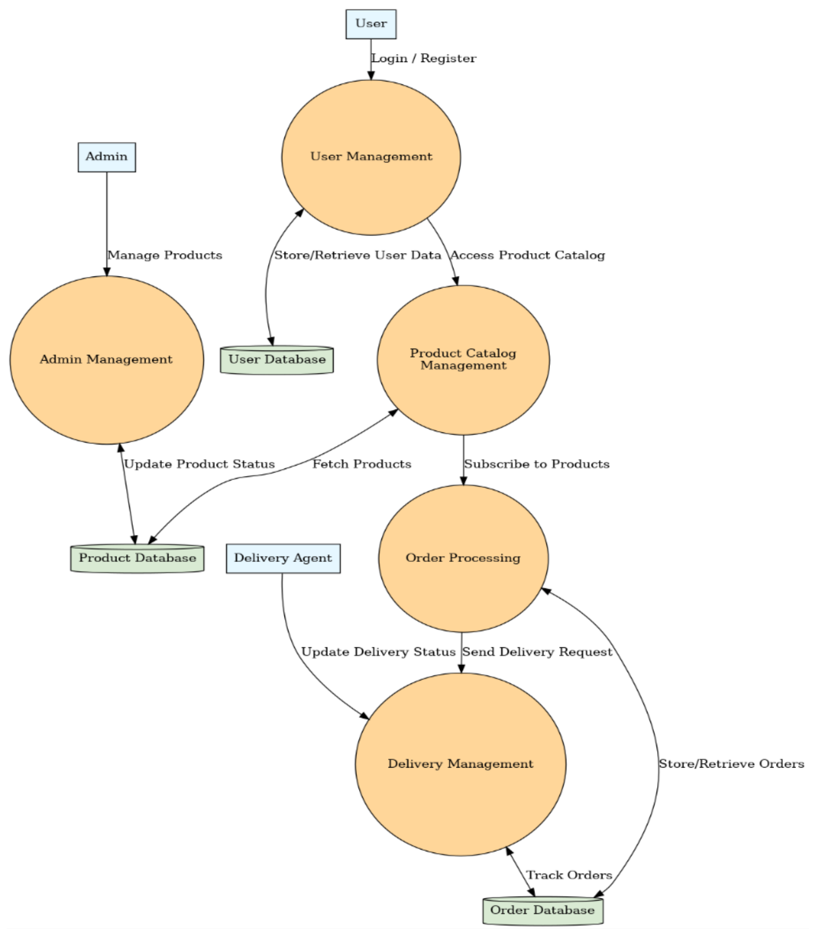
Processes:

1. **User Management:** Handles login, registration, and profile management.
2. **Product Catalogue Management:** Displays products, manages subscriptions.
3. **Order Processing:** Automates subscription orders and tracks them.
4. **Admin Management:** Adds/updates product details, checks stock availability.

5. **Delivery Management:** Tracks delivery status and updates.

Data Stores:

- **User Database:** Stores user credentials, profiles.
- **Product Database:** Contains product details and stock levels.
- **Order Database:** Manages order history, subscription schedules.



3.3. ER Diagrams

Grocify's Entity Relationship (ER) Diagram is a data model that describes the main entities and relationships between them. This representation helps you understand how the data flows between different modules like user, admin, products, orders, delivery agents etc. These diagrams help us to understand the structure of the application database better.

Key entities and attributes:

1. User:

Attributes: user (primary key), name, e-mail, password, address, contact

Description: It represents customers who use the GROCIFY application to subscribe and buy products.

2. Administrator:

Attributes: AdminID (primary key), Name, Email, Password

Description: Manages product information, inventory status, and monitors system performance.

3. Distributor:

Attributes: AgentID (primary key), Name, Contact, Status

Description: Manages the delivery of subscribed and ordered products to users.

4. Product:

Attributes: ProductID (Primary Key), Name, Description, Price, Stock Status

Description: Represents grocery items available for subscription and purchase in the app.

5. Subscription:

Attributes: SubscriptionID (Primary Key), UserID (Foreign Key), ProductID (Foreign Key), Plan Type (Daily/Weekly/Monthly/Yearly), Start Date, Status

Description: Captures the details of user subscriptions for products with auto-restocking features.

6. Order:

Attributes: OrderID (primary key), SubscriptionID (foreign key), DeliveryDate, DeliveryStatus

Description: Tracks orders created from active subscriptions and manages delivery schedules.

Relationships:

1. **User Places Subscription**: Each user can place multiple subscriptions.

Relationship Type: One-to-Many (User to Subscription)

2. **Subscription Contains Product**: A subscription can contain multiple products, and a product can be part of multiple subscriptions.

Relationship Type: Many-to-Many

3. **Admin Manages Product**: Admins are responsible for managing product information, including updating inventory status.

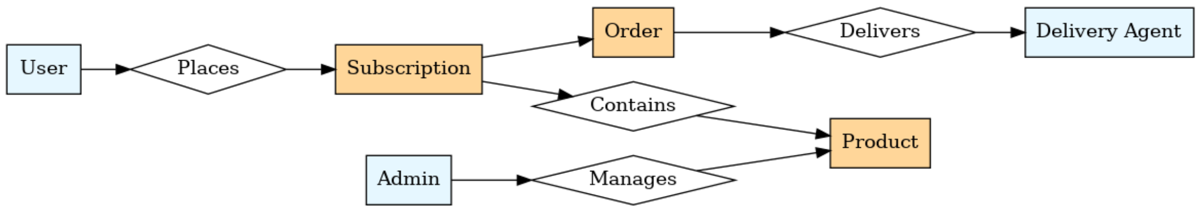
Relationship Type: One-to-Many (Admin to Product)

4. **Subscription Creates Order**: Each subscription can create multiple orders based on the plan type selected.

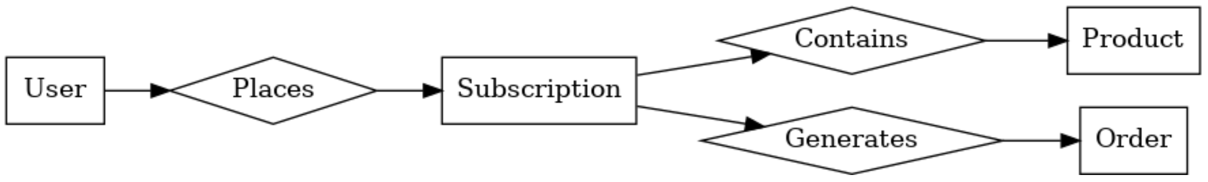
Relationship Type: One to Many (Subscription to Order)

5. **A courier delivers the order**: The order is assigned to a courier who is responsible for the delivery.

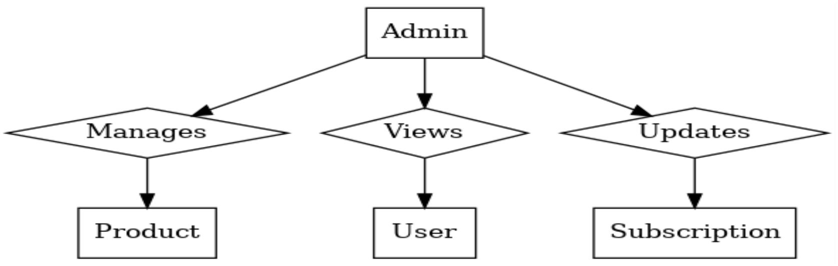
Relationship Type: One to Many (Order Delivery Agent)



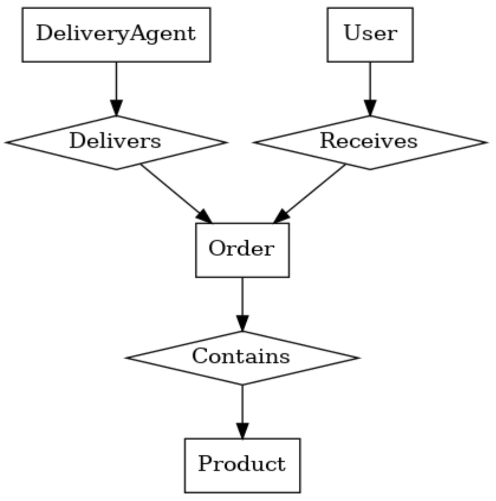
USER:



ADMIN:



DELIVERY:

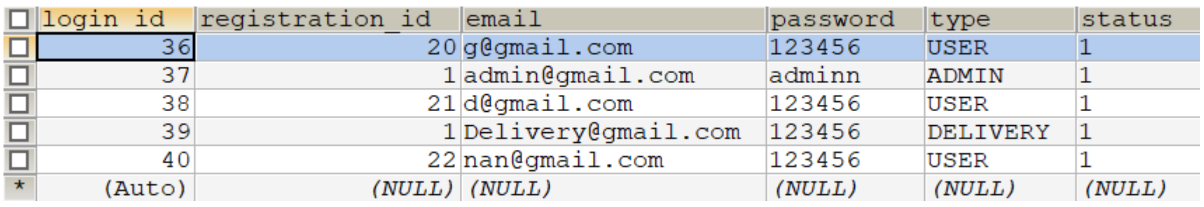


3.4. Data Design

Data design is the structured organization of data within the Grocify system. It ensures efficient management, retrieval, and storage. The grocify follows a relational database model. Here data is arranged into multiple tables with well-defined relationships. Key entities are Users, Admin, Products, Subscriptions, Orders, and Deliveries, each connected through Primary Keys (PK) and Foreign Keys (FK) to maintain data integrity. The database is created to minimize redundancy and improve performance. Various relationships between the entities ensure smooth automation of subscription-based grocery management which enables seamless order processing, inventory tracking, and delivery coordination.

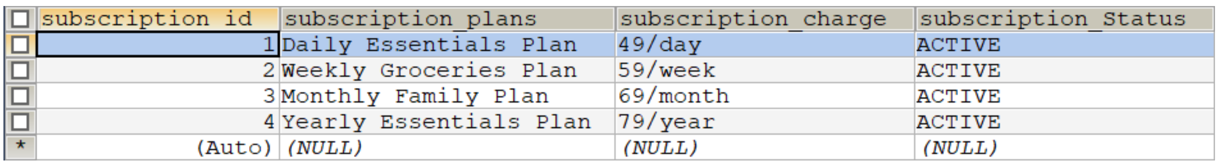
Table Design

Login table:



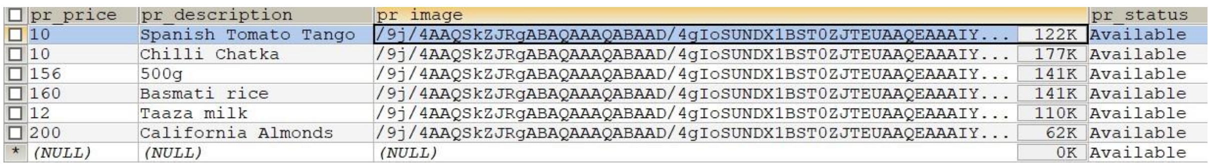
login_id	registration_id	email	password	type	status
36	20	g@gmail.com	123456	USER	1
37	1	admin@gmail.com	adminn	ADMIN	1
38	21	d@gmail.com	123456	USER	1
39	1	Delivery@gmail.com	123456	DELIVERY	1
40	22	nan@gmail.com	123456	USER	1
(Auto)	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*

Subscription plans table:



subscription_id	subscription_plans	subscription_charge	subscription_Status
1	Daily Essentials Plan	49/day	ACTIVE
2	Weekly Groceries Plan	59/week	ACTIVE
3	Monthly Family Plan	69/month	ACTIVE
4	Yearly Essentials Plan	79/year	ACTIVE
(Auto)	*(NULL)*	*(NULL)*	*(NULL)*

Products table:



pr_price	pr_description	pr_image	pr_status
10	Spanish Tomato Tango	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQEAAAIY... 122K	Available
10	Chilli Chatka	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQEAAAIY... 177K	Available
156	500g	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQEAAAIY... 141K	Available
160	Basmati rice	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQEAAAIY... 141K	Available
12	Taaza milk	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQEAAAIY... 110K	Available
200	California Almonds	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQEAAAIY... 62K	Available
(NULL)	*(NULL)*	*(NULL)* 0K	Available

4. SYSTEM DEVELOPMENT

4.1. Introduction

The development of our grocery shopping application "Grocify" follows a structured approach according to Software Development Life Cycle (SDLC) ensuring efficient and running platform. The process begins with requirement analysis where the requirements of the users, administrators, delivery agents etc are identified to define functionalities such as subscription management, order automation, and product tracking etc. Next is system design where data models like Data Flow Diagrams (DFD) and Entity-Relationship (ER) Diagrams are created with structured database to manage subscriptions, product inventories, and order processing. Thus, this kind of structuring approach gives us a better understanding and hence can implement a seamless, automated grocery shopping experience with smooth order processing reducing manual efforts from users and enhancing efficiency and user convenience.

4.2. Process Description

Grocify follows a structured process flow to provide an efficient subscription-based grocery shopping experience. Throughout the process, the system ensures seamless coordination between users, admins, and delivery agents, enhancing user convenience and optimizing grocery management.

Users:

The process begins with user registration and authentication, where new users sign up and existing users log in to access their accounts. Once logged in, users can browse the product catalogue, which displays available grocery items categorized for easy selection. Users can then add products to their subscription plan by choosing a suitable frequency such as daily, weekly, monthly, or yearly, eliminating the need for manual reordering. After selecting products, the system automatically generates subscription-based orders at the specified intervals and sends them for processing.

Admin**:**

Once the admin has logged in, admin can add products, view products, view subscription plans and view users. Admin can add any product that needed to be added and can also update on its availability status. Admin can also view the users and their details of login. Admin can also review the subscriptions plans of various users. The admin module plays a crucial role in managing the product database view by updating stock availability and modifying product details.

Delivery agent:

The delivery agent can access assigned orders and track delivery schedules after logging in. They can see the product list, order details, user addresses to ensure efficient and timely delivery. The system updates delivery status in real time allowing both users and agents to track order and promoting transparency and upon successful delivery the delivery agent can update the status to delivered.

4.3. Code Design

```
//JAVA CODE LOGIN PAGE
public class LoginActivity extends AppCompatActivity {

    EditText etEmail, etPassword;
    Button btnLogin;
    TextView tvSignUp;

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

        // Initialize Views
        etEmail = findViewById(R.id.etEmail);
        etPassword = findViewById(R.id.etPassword);
        btnLogin = findViewById(R.id.btnLogin);
        tvSignUp = findViewById(R.id.tvSignUp);
        // Navigate to Signup Activity
        tvSignUp.setOnClickListener(view -> {
            startActivity(new Intent(LoginActivity.this, SignupActivity.class));
        });

        // Handle Login Button Click
        btnLogin.setOnClickListener(view -> validateAndLogin());
    }

    // Validate Form Fields
    private void validateAndLogin() {
        String email = etEmail.getText().toString().trim();
        String password = etPassword.getText().toString().trim();

        if (email.isEmpty() || !android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            etEmail.setError("Valid Email is required");
            etEmail.requestFocus();
            return;
        }

        if (password.isEmpty() || password.length() < 6) {
            etPassword.setError("Password must be at least 6 characters");
            etPassword.requestFocus();
            return;
        }

        // If validation is successful, call the login function
        loginVolley(email, password);
    }


    // Login API Call using Volley
    private void loginVolley(String email, String password) {

        RequestQueue queue = Volley.newRequestQueue(LoginActivity.this);
        StringRequest request = new StringRequest(Request.Method.POST, Utility.SERVERUrl, new
Response.Listener<String>() {
```

```
            @Override
            public void onResponse(String response) {
                if (!response.trim().equals("failed")) {
                    String[] resArr = response.trim().split("#");
                    SharedPreferences.Editor editor = getSharedPreferences("sharedData",
MODE_PRIVATE).edit();
                    editor.putString("login_id", resArr[0]);
                    editor.putString("type", resArr[1]);
                    editor.apply();

                    switch (resArr[1]) {
                        case "ADMIN":
                            startActivity(new Intent(LoginActivity.this, AdminHome.class));
                            break;
                        case "USER":
                            startActivity(new Intent(LoginActivity.this, UserHome.class));
                            break;
                        case "DELIVERY":
                            startActivity(new Intent(LoginActivity.this, DeliveryAgentHome.class));
                            break;

                    }

                    Toast.makeText(LoginActivity.this, "Login Success", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(LoginActivity.this, "Login Failed!!", Toast.LENGTH_SHORT).show();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {

  Toast.makeText(LoginActivity.this, "Error: " + error, Toast.LENGTH_LONG).show();
Log.i("Error", "" + error);

}

}) {
```

```
@Override
            protected Map<String, String> getParams() {
                Map<String, String> map = new HashMap<>();
                map.put("key", "login");
                map.put("email", email);
                map.put("password", password);
                return map;
            }
        };
        queue.add(request);
    }
    }
```

//XML CODE FOR LOGIN PAGE

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true">


    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        xmlns:app="http://schemas.android.com/apk/res-auto">

        <!-- Gradient Background -->
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:background="@color/background"
            android:gravity="center">


            <!-- App Logo -->
        <ImageView android:id="@+id/appLogo"
            android:layout_width="150dp"
            android:layout_height="150dp"
```

```
  android:src="@drawable/app_logo"
    android:layout_gravity="center"
    android:contentDescription="App Logo"
    android:layout_marginBottom="32dp"/>

  <!-- Login Card -->
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="@drawable/login_card"
    android:padding="16dp"
    android:layout_margin="20dp"
    android:elevation="8dp"
    android:layout_gravity="center">

    <!-- Email Input -->
    <EditText
      android:id="@+id/etEmail"
      android:layout_width="match_parent"
      android:layout_height="50dp"
      android:hint="Email Address"
      android:drawableLeft="@drawable/ic_email"
      android:drawablePadding="12dp"
      android:padding="16dp"
      android:textSize="16sp"
      android:background="@drawable/rounded_input"
      android:inputType="textEmailAddress"
      android:layout_marginTop="8dp"/>
<!-- Password Input -->

    <EditText
      android:id="@+id/etPassword"
      android:layout_width="match_parent"
      android:layout_height="50dp"
      android:hint="Password"
      android:drawableLeft="@drawable/ic_lock"
      android:drawablePadding="12dp"
```

```
            android:padding="16dp"
            android:textSize="16sp"
            android:background="@drawable/rounded_input"
            android:inputType="textPassword"
            android:layout_marginTop="12dp"/>

        <!-- Login Button -->
        <Button
            android:id="@+id/btnLogin"
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:text="Login"
            android:textStyle="bold"
            android:textSize="18sp"
            android:backgroundTint="@color/myPrimary"
            android:textColor="#FFFFFF"
            android:layout_marginTop="24dp"
            app:cornerRadius="16dp"/>

        <!-- Sign Up Link -->
        <TextView
            android:id="@+id/tvSignUp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Don't have an account? Sign Up"
            android:textColor="@color/myPrimary"
            android:textSize="12sp"
            android:textStyle="bold"
            android:layout_gravity="center"
            android:layout_marginTop="20dp"/>

      </LinearLayout>
    </LinearLayout>
  </RelativeLayout>
</ScrollView>
```

5. SYSTEM TESTING AND IMPLEMENTATION

5.1. Introduction

System testing and Implementation are crucial phases in development of this application to ensure application runs well as intended and is deployed for real-life world. This System testing involves verifying individual components, module interactions, and overall performance to identify and resolve any issues if any before deployment. Implementation part mainly focuses on setting up live environment, migrating data, training users and maintain system functioning to ensure smooth operations.

5.2. Implementation

The implementation phase of the application aims to deploy application for real-world use after going through various testing phases and ensuring stability and efficiency. First step the application is established by configuring the application on a server and integrating the database to enable real-time subscription and order management. Data migration involves populated test data being moved to live system while maintaining consistency. Then a pilot test run is done with a selected group of users allowing them to monitor the real-time system functioning after automation successfully tested, we can rant full access to the users, admins and delivery agents within application. Users are taught through training and documentation how the system works. Post deployment phase system is monitored to make necessary updates and optimize performance and security based on the customer's or user's needs.

5.3. Testing

System Testing is a crucial phase which ensures that if the application is working as intended and meets all functional and non-functional requirements. Various testing methodologies are taken to identify and rectify any issues if present before deployment.

Types of testing conducted are:

Unit Testing: Various Individual components of Grocify app is taken in consider isolated and checked such as user authentication, product subscription management, order tracking etc are tested separately to ensure their functioning.

Integration Testing: Interactions between different modules such as user actions triggering product subscription updates and delivery assignments are verified ensuring seamless communication between front-end and back-end and database.

System Testing: The entire application is tested ensuring it meets all the requirements checking all the modules user, admin, and delivery agents are working together seamlessly for successful running application.

User Acceptance Testing (UAT): In this test a selected group of users are selected to validate the requirements of the customers and are the expectations met. And result feedback is taken for improvements.

Performance Testing: When multiple users do subscriptions simultaneously, we evaluate the response time and stability of the application under various loads.

Security Testing: We test everything login authentication, role-based access control, and data encryption for security vulnerabilities ensuring user data privacy and protection from unauthorized access.

5.4. Scope For Future Enhancement

There is always room for improvement and scope for future enhanced features upgrading to improve user experience, efficiency and automation. Some of them are:

1. Push Notifications & Alerts: Implementing real time updates and notification for subscription renewals, order tracking, available frequent bought products, new products, offers etc

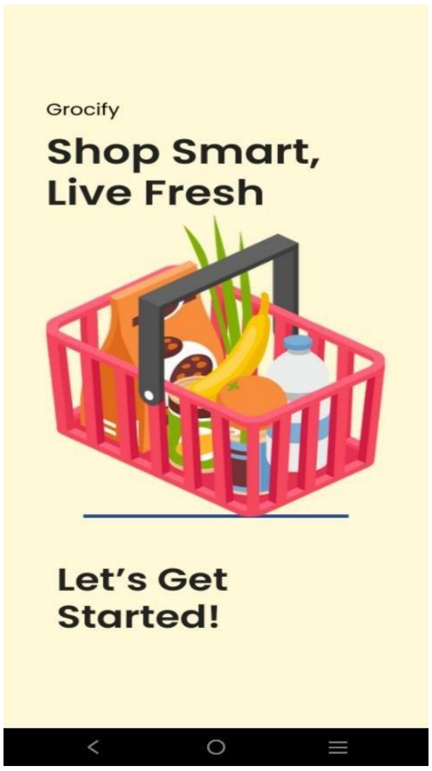
2. AI-Powered Recommendations: Using machine learning to analyse the user's previous purchases and observe the pattern and suggests products based on that.
3. Multiple Payment Option: Integrating various types of payment gateways including UPI, digital wallets etc.
4. Dynamic Pricing & Discounts: Giving promotional offers, discounts based on user's previous purchases.
5. Voice Command Integration: Enabling voice command instructions possible like google assistant or Alexa.
6. Advanced Order Tracking: Providing real-time order tracking to users giving estimated time of arrival etc.
7. Multi-Vendor Support: Expanding the platforms by allowing vendors from different places to list their products.
8. Cross-Platform Availability: Expanding availability to iOS and web applications for more accessibility.

6. CONCLUSION

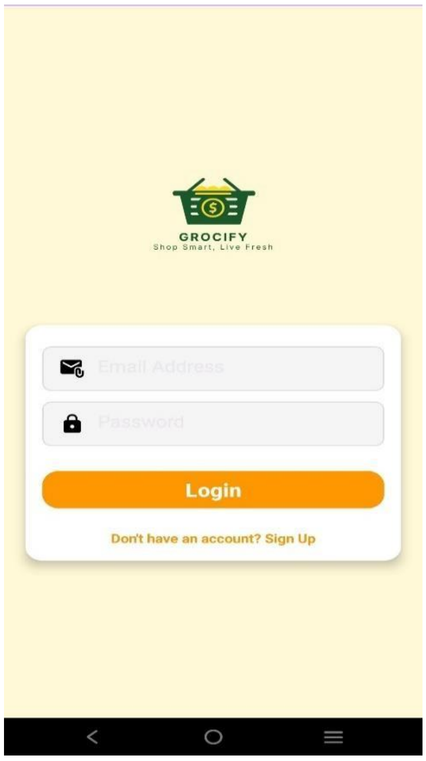
Grocify is an innovative service designed to manage subscriptions for groceries with the intent to make shopping easier and more efficient. To minimize customer interaction and guarantee a seamless buying experience, the application enables users to subscribe to automated product purchased every day, week, month, or year. The automation of restocking, orders, and efficient inventory management increases user convenience and operation productivity. Grocify is a robust and user-friendly platform designed for the modern grocery shopper because of its scalable functionalities, secure database infrastructure, and a logical system design. Further extensions could include cross-platform compatibility, push notifications, and AI recommendations which would increase its already vast capabilities. Overall, the effectiveness of Grocify as a service is striking as it completely revolutionizes the way consumers approach their shopping necessities.

7.APPENDIX

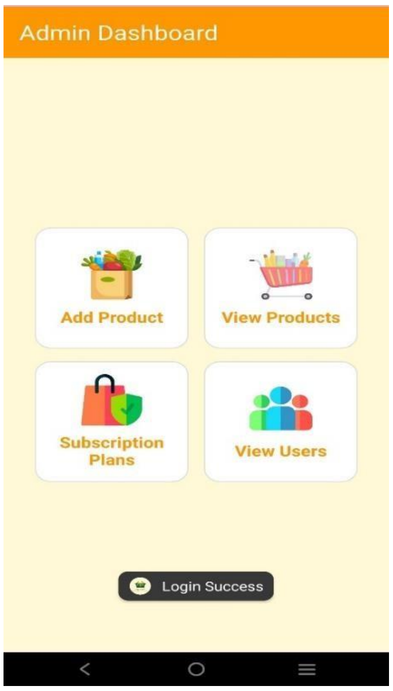
HOME PAGE:



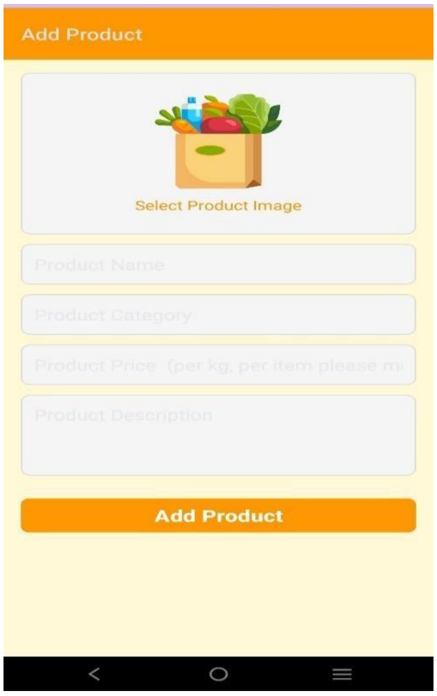
LOGIN PAGE:



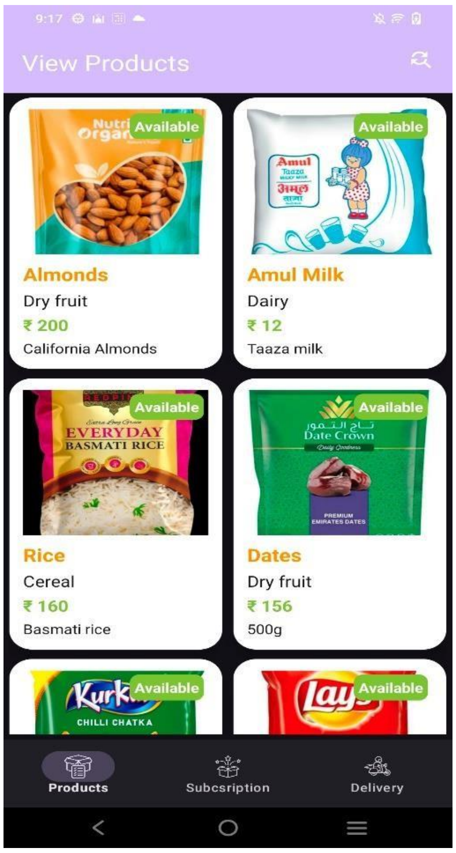
ADMIN DASHBOARD:



ADD PRODUCT:



VIEW PRODUCTS:


9:17

View Products

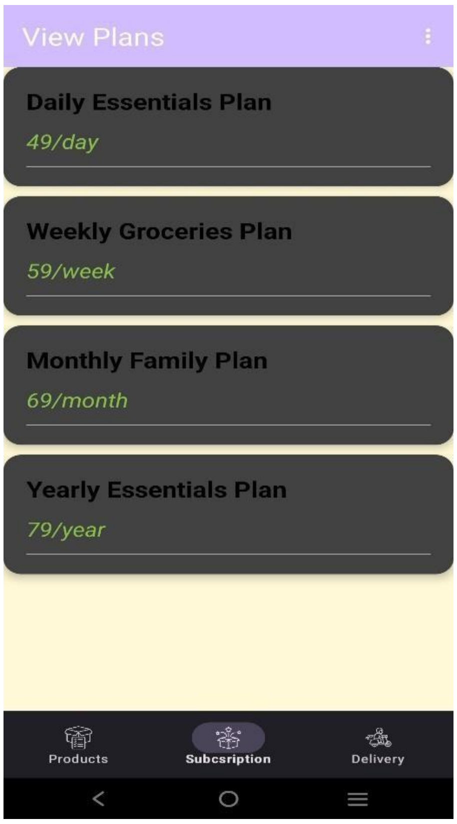
Almonds
Dry fruit
₹ 200
California Almonds

Amul Milk
Dairy
₹ 12
Taaza milk

Rice
Cereal
₹ 160
Basmati rice

Dates
Dry fruit
₹ 156
500g

VIEW PLANS


View Plans

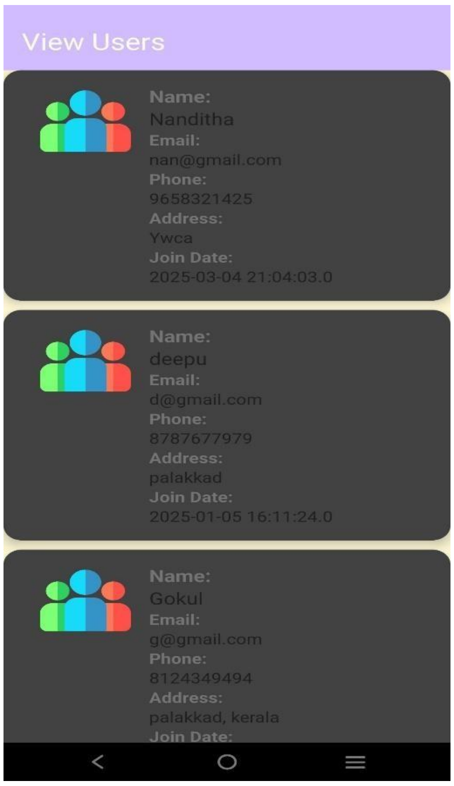
Daily Essentials Plan
49/day

Weekly Groceries Plan
59/week

Monthly Family Plan
69/month

Yearly Essentials Plan
79/year

VIEW USERS:

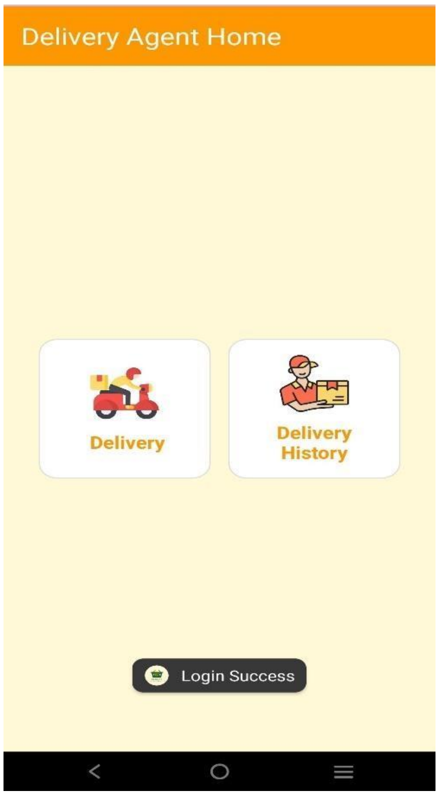

View Users

Name:
Nanditha
Email:
nan@gmail.com
Phone:
9658321425
Address:
Ywca
Join Date:
2025-03-04 21:04:03.0

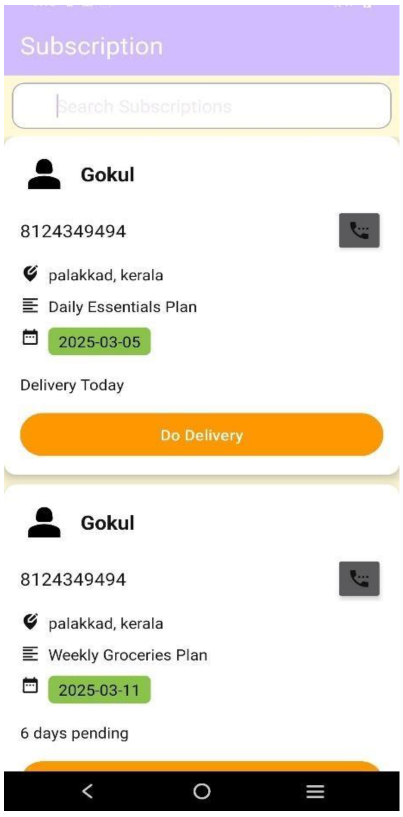
Name:
deepu
Email:
d@gmail.com
Phone:
8787677979
Address:
palakkad
Join Date:
2025-01-05 16:11:24.0

Name:
Gokul
Email:
g@gmail.com
Phone:
8124349494
Address:
palakkad, kerala
Join Date:

DELIVERY AGENT HOME:


Delivery Agent Home

SUBSCRIPTIONS:



Gokul

8124349494

palakkad, kerala

Daily Essentials Plan

2025-03-05

Delivery Today

Gokul

8124349494

palakkad, kerala

Weekly Groceries Plan

2025-03-11

6 days pending

8.BIBLIOGRAPHY

8.1. References

- https://www.lovelycoding.org/grossary-management-system/
- https://nevonprojects.com/grocery-shopping-android/
- https://www.rishabhsoft.com/blog/grocery-delivery-app-development