Coffee Order Management System

By

Suraj Gupta - AF04970089

INDEX

Acknowledgement

The Project **"Coffee order management System"** is the Project work carried out by

Name	Enrolment No
Suraj	AF04970089

We are thankful to my project guide for guiding me to complete the Project. Their suggestions and valuable information regarding the formation of the Project Report have provided me a lot of help in completing the Project and its related topics. We are also thankful to my family member and friends who were always there to provide support and moral boost up.

Abstract

In today's digital era, convenience and personalization are essential aspects of modern consumer experiences. The **Coffee Order System** is a full-stack web application designed to simplify and enhance the coffee ordering process through an intuitive, secure, and interactive online platform. The system enables users to browse a dynamic coffee menu, place customized orders, submit product reviews, and complete secure payments — all from a unified interface.

Developed using React.js for the frontend, Express.js for the backend, and SQLite with Prisma ORM for database management, the project demonstrates the implementation of a modern, scalable, and maintainable web architecture. Google OAuth authentication ensures user security and streamlined login, while the integrated review management system allows users to provide feedback, contributing to an engaging user experience. The system also features a complete checkout process, including order summary, payment confirmation, and success notifications.

By integrating authentication, product management, and payment processing into a single application, the **Coffee Order System** offers users a fast, reliable, and user-friendly platform for purchasing coffee online. This project highlights modern web development practices such as RESTful API design, component-based UI development, and secure data handling — providing an effective solution for digital coffee ordering and customer engagement in small to medium-scale café environments.

1. Introduction

Coffee has become an integral part of modern lifestyles, with consumers seeking convenience, quality, and personalization in every cup. In today's fast-paced world, customers prefer digital solutions that simplify daily activities — from food delivery to online shopping — and coffee ordering is no exception. Traditional in-store ordering often leads to long waiting times, order errors, and limited customization options. To overcome these challenges, technology can play a crucial role in enhancing the customer experience by offering a seamless, efficient, and user-friendly coffee ordering process.

The **Coffee Order System** is a comprehensive web-based application developed to simplify and digitize the coffee purchasing experience. It enables users to browse a dynamic coffee menu, place customized orders, make secure payments, and share feedback through reviews — all within a single, integrated platform. The system not only improves customer satisfaction but also supports café management by streamlining operations and ensuring real-time order processing. Built using **React.js** (frontend), **Express.js** (backend), and **SQLite with Prisma ORM** (database), this project exemplifies the integration of modern web technologies for efficient service delivery.

1.1 Background of the Study

With the rapid growth of the food and beverage industry, online ordering systems have become a key factor in improving service efficiency and customer engagement. Many cafés and small coffee shops still rely on traditional ordering methods, leading to slow service and limited customer interaction. Customers increasingly expect online

The demand for digital ordering platforms has significantly increased due to the widespread adoption of mobile and web technologies. The **Coffee Order System** was developed with this vision — to provide users with a smooth, secure, and interactive coffee ordering experience while helping businesses modernize their customer service approach. The system promotes automation, reduces manual errors, and enhances user engagement through interactive features such as real-time order updates and integrated reviews.

1.2 Motivation

The development of the Coffee Order System is motivated by several key challenges faced by both customers and café owners:

- **Long Waiting Times** – Traditional coffee ordering often results in delays, especially during peak hours.
- **Order Inaccuracy** – Manual order taking can lead to incorrect items or missed customizations.
- **Limited Customer Interaction** – Lack of digital feedback mechanisms reduces engagement and service improvement opportunities.
- **Security Concerns** – Many existing systems lack secure authentication and payment handling.
- **Need for Modernization** – Small cafés and local shops need cost-effective digital platforms to compete with large chains.

The **Coffee Order System** addresses these issues by integrating **Google OAuth authentication**, a secure **checkout and payment process**, and a **review management feature** that allows users to share feedback easily. It provides a complete end-to-end digital experience that enhances user convenience while helping cafés manage their operations efficiently.

3. Significance of the Study

This project holds great significance for both users and café management systems in today's digitally driven market. By integrating technology with everyday coffee purchasing habits, the **Coffee Order System** enhances convenience, personalization, and efficiency for all stakeholders involved.

For Customers:
Provides a seamless coffee ordering experience by allowing users to browse the menu, place orders, and make payments from anywhere, reducing waiting times and improving satisfaction.

For Café Owners:
Simplifies order management and enhances customer engagement through real-time updates, reviews, and digital transactions, helping small businesses compete with larger coffee chains.

For Developers and Learners:
Demonstrates a practical implementation of a full-stack web application using React, Express.js, Prisma ORM, and Google OAuth — serving as a learning model for modern web development.

For Society:
Encourages digital adoption, promotes contactless transactions, and supports local cafés in offering modern online ordering experiences aligned with current consumer trends.

4. Features of the Coffee Order System

The application includes several modules that make online coffee ordering simple, efficient, and user-friendly.

Menu Management

a) Displays an interactive coffee menu with item names, descriptions, and prices.
b) Allows filtering and sorting of items by type (e.g., Espresso, Latte, Cappuccino).
c) Supports dynamic updates of menu items from the database.

Order and Checkout System

a) Enables users to add items to the cart and customize their orders.
b) Displays a detailed checkout summary with total price calculation.
c) Supports secure payment processing with confirmation notifications.
d) Generates a success message or receipt after payment completion.

User Authentication

a) Integrated **Google OAuth** for easy and secure user login.
b) Stores authenticated user data (name, email) in the database.
c) Maintains session state using React Context and backend verification.

Review Management

a) Allows users to add reviews and rate coffee products.
b) Displays average ratings and customer feedback on product pages.
c) Helps maintain user engagement and improve service quality.

Admin Controls (optional extension)

a) Add, update, or remove coffee items from the menu.
b) View and manage user reviews and order history.
c) Access analytics and reports on top-selling items.

Database and Security

a) Managed using **SQLite** with **Prisma ORM** for data consistency.
b) Secure API communication between frontend and backend via Express.js.
c) Validation and authentication checks to ensure safe user access.

5. Expected Outcomes

The implementation of the **Coffee Order System** aims to achieve the following outcomes:

- A fast, efficient, and user-friendly digital coffee ordering experience.
- Reduced waiting times and improved order accuracy for customers.
- Streamlined café operations with simplified menu and order management.
- Enhanced customer satisfaction through product reviews and feedback.
- Secure authentication and reliable data storage using modern frameworks.
- A scalable, modular, and easily maintainable full-stack web solution.

6. Organization of the Report

This project report is organized into the following chapters for clarity and structured presentation:

1. **Abstract** – A concise summary of the project and its objectives.
2. **Introduction** – Describes the background, motivation, and significance of the system.
3. **System Analysis** – Defines the problem statement, objectives, and feasibility of the project.
4. **System Design** – Explains the architecture, ER diagrams, and database schema.
5. **Implementation** – Details frontend, backend, database integration, and authentication flow.
6. **Testing** – Describes test strategies, tools used, and validation of features.

7. **Results and Discussion** – Presents screenshots, feature demonstrations, and performance evaluation.
8. **Conclusion and Future Scope** – Provides final insights and potential enhancements for future development.
9. **Bibliography** – Lists all references, books, and web resources used.

2. System Analysis

System Analysis is a critical stage in the Software Development Life Cycle (SDLC). It focuses on identifying existing problems, analyzing user requirements, and determining the feasibility of developing an improved system. For the **Coffee Order System**, this phase involved understanding customer pain points in traditional coffee ordering and defining functional and technical needs to design a more efficient digital solution.

2.1 Problem Definition

In many cafés and small coffee shops, the coffee ordering process remains manual and inefficient. Customers must wait in queues, place verbal orders, and make payments at the counter. This often results in slow service, order errors, and poor customer satisfaction. Additionally, there is limited opportunity for users to share feedback or for café owners to collect data on customer preferences.

Challenges identified include:

- Long waiting times during rush hours.
- Order errors due to manual communication.
- Lack of digital menu visibility and pre-order options.
- Inability to provide customer reviews or ratings.
- Limited data management for café owners.
- No centralized system for handling payments and orders securely.

Hence, there is a growing need for a **web-based coffee ordering platform** that automates the entire process — from menu browsing to payment — ensuring convenience, speed, and transparency for both customers and café owners.

2.2 Objectives of the System

The **Coffee Order System** aims to address the above issues by providing a modern, interactive, and secure online platform for ordering coffee. The main objectives are:

- To provide a centralized system for browsing, selecting, and ordering coffee online.
- To integrate **Google OAuth authentication** for secure and simplified user login.
- To develop a **review management feature** for collecting and displaying user feedback.
- To implement a **checkout and payment system** with real-time confirmation.
- To store and manage user and order data efficiently using **SQLite with Prisma ORM**.
- To ensure a responsive, visually appealing, and easy-to-use interface.
- To follow **modular design principles**, ensuring scalability and maintainability.

2.3 Feasibility Study

A feasibility study evaluates whether the proposed system can be successfully developed and deployed based on technical, economic, operational, and time factors.

a) Technical Feasibility

The system is developed using reliable and widely supported open-source technologies:

- **Frontend:** React.js (with React Router for navigation and Context API for state management)

- **Backend:** Express.js for RESTful API development
- **Database:** SQLite managed with Prisma ORM for structured and consistent data handling
- **Authentication:** Google OAuth 2.0 for secure user login
- **Tools:** Vite, Node.js, and npm libraries

All technologies are lightweight, scalable, and compatible, making the project technically feasible.

b) Economic Feasibility

The system is highly cost-effective as it utilizes open-source technologies:

- No licensing fees for development tools or frameworks
- Minimal hosting cost for deployment
- Long-term benefits such as improved efficiency and customer satisfaction outweigh initial setup costs

Hence, the system is economically feasible.

c) Operational Feasibility

The system is simple and intuitive, designed for both customers and café owners:

- Easy navigation and responsive interface
- Reduced manual workload for café staff
- Improved accuracy and customer experience through automation

Therefore, it is operationally feasible.

d) Time Feasibility

The modular structure of the project ensures efficient development and testing within a limited timeline:

- Frontend, backend, and database modules were developed concurrently.

- Agile methodology was followed for incremental progress and early testing.

Hence, the project is time-feasible.

2.4 System Requirements

To ensure optimal performance and usability, the system requirements are categorized as follows:

a) Functional Requirements

- User registration and login using Google OAuth.
- Display of coffee menu with categories, prices, and descriptions.
- Option to add items to the cart and customize orders.
- Checkout process with total price calculation and payment confirmation.
- Review system for submitting and viewing customer feedback.
- Admin control for managing products and orders (if implemented).
- Secure session management and data validation.

b) Non-Functional Requirements

- **Usability:** Intuitive and responsive design for smooth user interaction.
- **Performance:** Fast load times and optimized API responses.
- **Security:** Data protection via OAuth authentication and secure backend APIs.
- **Reliability:** Accurate data handling with error management.
- **Scalability:** Support for multiple users and expanding product catalogue.
- **Maintainability:** Clean modular code structure for easy updates.

c) Hardware Requirements

- **Processor:** Intel i3 or higher
- **RAM:** Minimum 4 GB (8 GB recommended)
- **Storage:** Minimum 500 MB for database and application files

d) Software Requirements

- **Operating System:** Windows / Linux / macOS
- **Development Tools:** Node.js, Vite, Visual Studio Code
- **Database:** SQLite with Prisma ORM
- **Browser:** Google Chrome / Mozilla Firefox
- **Server Environment:** Express.js backend with REST APIs

2.5 Proposed System

The proposed **Coffee Order System** overcomes the limitations of manual ordering by offering a complete digital experience.

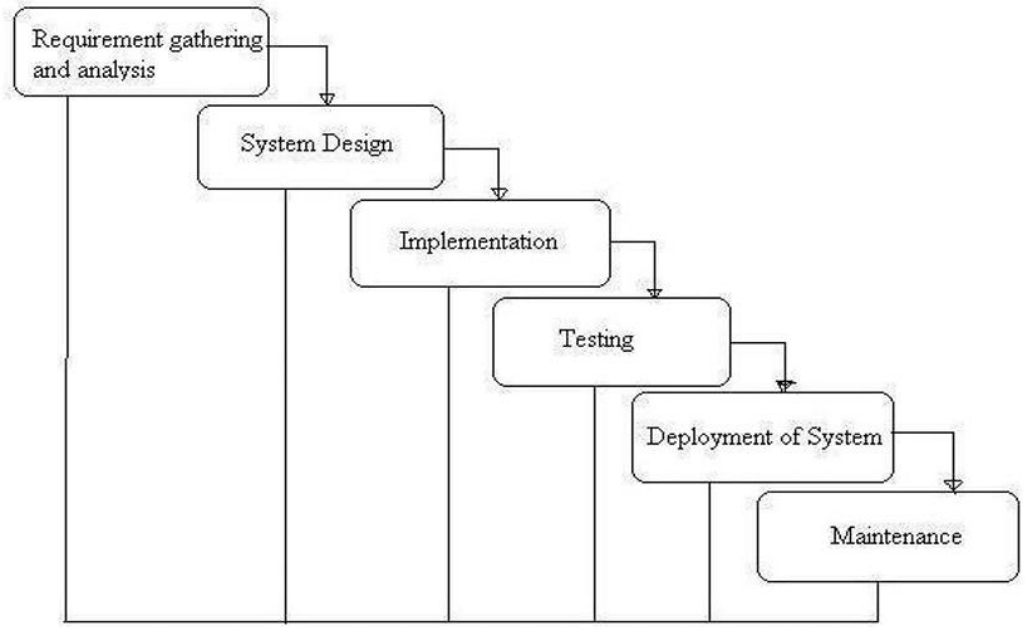
Key Features of the Proposed System:

- Digital coffee menu with pricing and descriptions.
- Google login for easy authentication.
- Add-to-cart, checkout, and secure payment process.
- Real-time order confirmation and status updates.
- Review system for feedback and engagement.
- Admin access to manage menu and monitor reviews.

Advantages over the existing system:

- Faster and error-free order management.
- Enhanced customer satisfaction through personalization.
- Reduced manual effort for café staff.

- Data-driven insights for business improvement.
- Secure transactions and reliable order records.



3. System Design

System design represents the blueprint of the proposed **Coffee Order System**. It defines the architecture, modules, data flow, and database relationships that together ensure the application's smooth operation. This chapter describes how different system components interact with each other to provide a seamless user experience — from logging in, browsing coffee products, placing orders, to completing payments and submitting reviews.

3.1 System Architecture

The **Coffee Order System** follows a **three-tier architecture**, which separates the application into three primary layers:

1. **Frontend (Presentation Layer)**
2. **Backend (Application Layer)**
3. **Database (Data Layer)**

1. Frontend (React.js with Vite)

- Developed using **React.js** for component-based UI.
- Manages routing using **React Router**.
- Handles authentication state via **Context API**.
- Provides responsive, user-friendly pages such as:
    - Home Page
    - Menu Page
    - Checkout Page
    - Payment Page
    - Review Page

2. Backend (Express.js)

- RESTful API built on **Node.js** with Express framework.

- Handles all requests from the frontend including:
    - Authentication and authorization
    - Coffee menu management
    - Review CRUD operations
    - Checkout and payment handling
- Implements middleware for secure and efficient routing.

3. Database (SQLite with Prisma ORM)

- Stores structured data including:
    - User details
    - Coffee menu items
    - Orders and payments
    - Reviews and ratings
- **Prisma ORM** ensures reliable interaction between backend APIs and the database through schema mapping and query optimization.

4. Authentication Layer (Google OAuth 2.0)

- Provides secure and convenient user login using Google credentials.
- Validates tokens on both client and server side.
- Stores user profiles in the database after first successful login.

Architecture Flow Summary:

The user interacts with the React frontend → frontend sends API requests to Express backend → backend validates requests and fetches/stores data in SQLite via Prisma → responses are sent back to the frontend for display.

3.2 Modules of the System

1. User Module

- Allows users to sign in securely using Google OAuth.
- Manages session state for accessing protected pages (Menu, Checkout, Review).
- Enables users to browse coffee items, view reviews, and place orders.

2. Menu Module

- Displays all available coffee items fetched from the database.
- Provides details such as name, image, description, and price.
- Allows filtering or sorting items by category or popularity.

3. Order & Checkout Module

- Handles adding coffee items to the cart.
- Displays cart summary with total price calculation.
- Processes payments and shows success confirmation.

4. Review Module

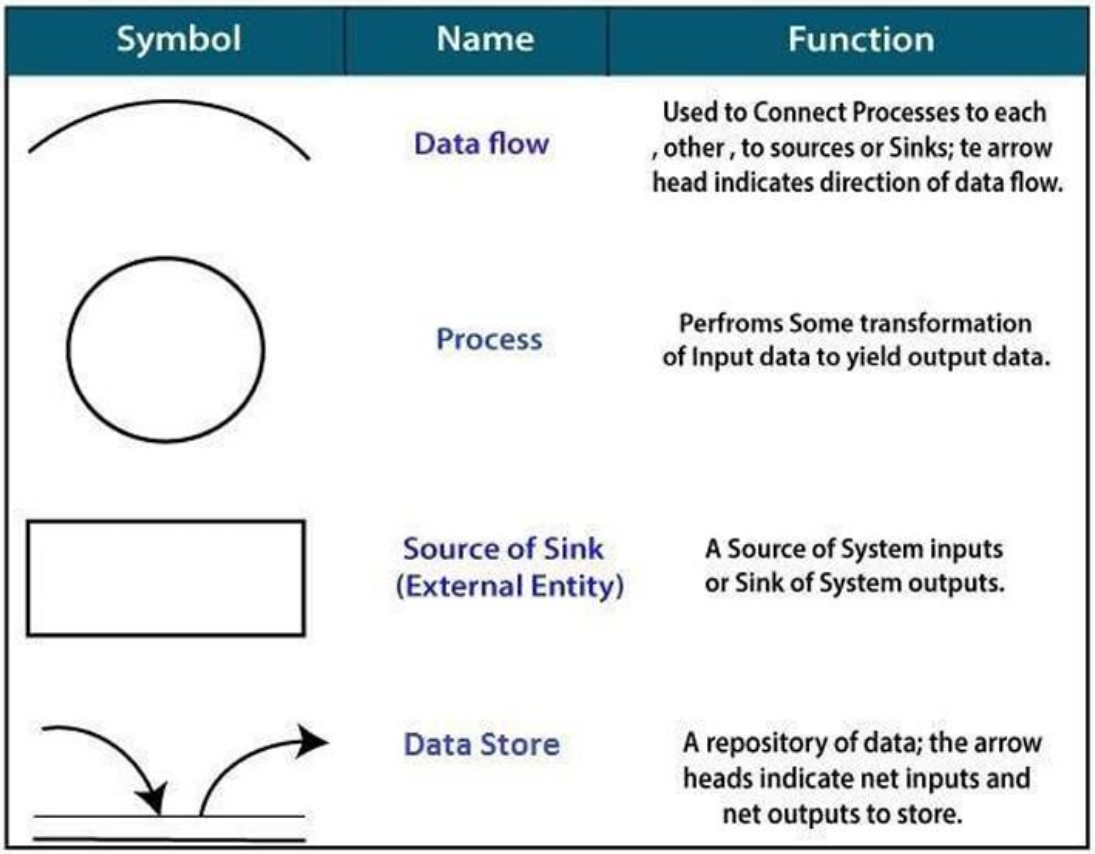
- Allows authenticated users to submit reviews and ratings for coffee products.
- Fetches existing reviews and displays average ratings per product.

5. Admin Module (optional extension)

- Enables admins to add, update, or remove coffee menu items.
- View customer feedback and order analytics.

3.3 Data Flow Diagrams (DFD)

A **Data Flow Diagram (DFD)** visually represents the flow of data within the system — showing how inputs are transformed into outputs through processes and data stores.



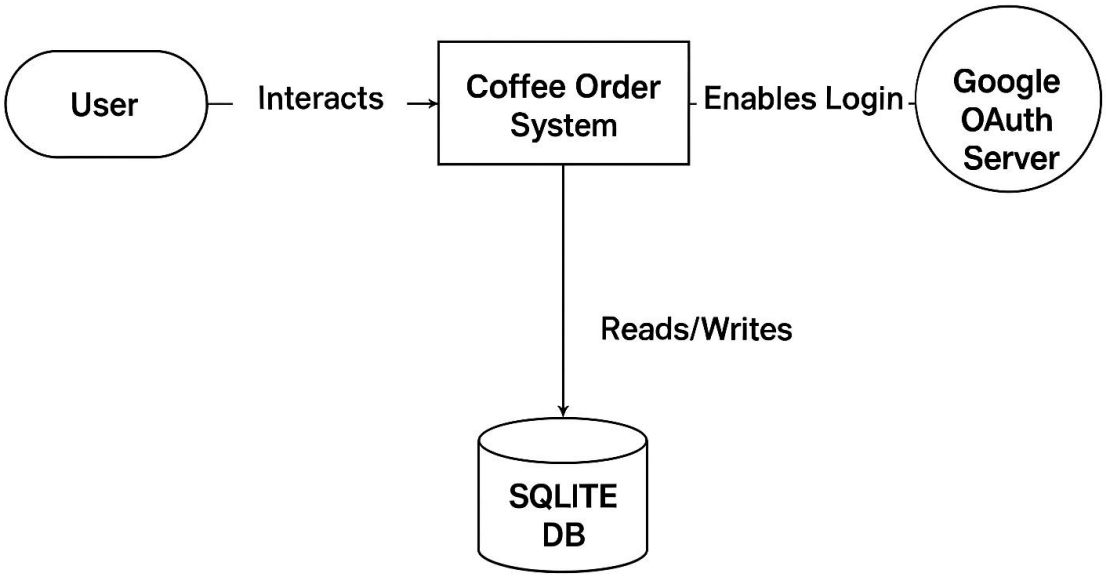
Symbols for Data Flow Diagrams

Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.



a) Zero-Level DFD (Context Diagram)

The **Zero-Level DFD**, also called the **Context Diagram**, represents the entire **Coffee Order System** as a single process and its interaction with external entities.

External Entities:

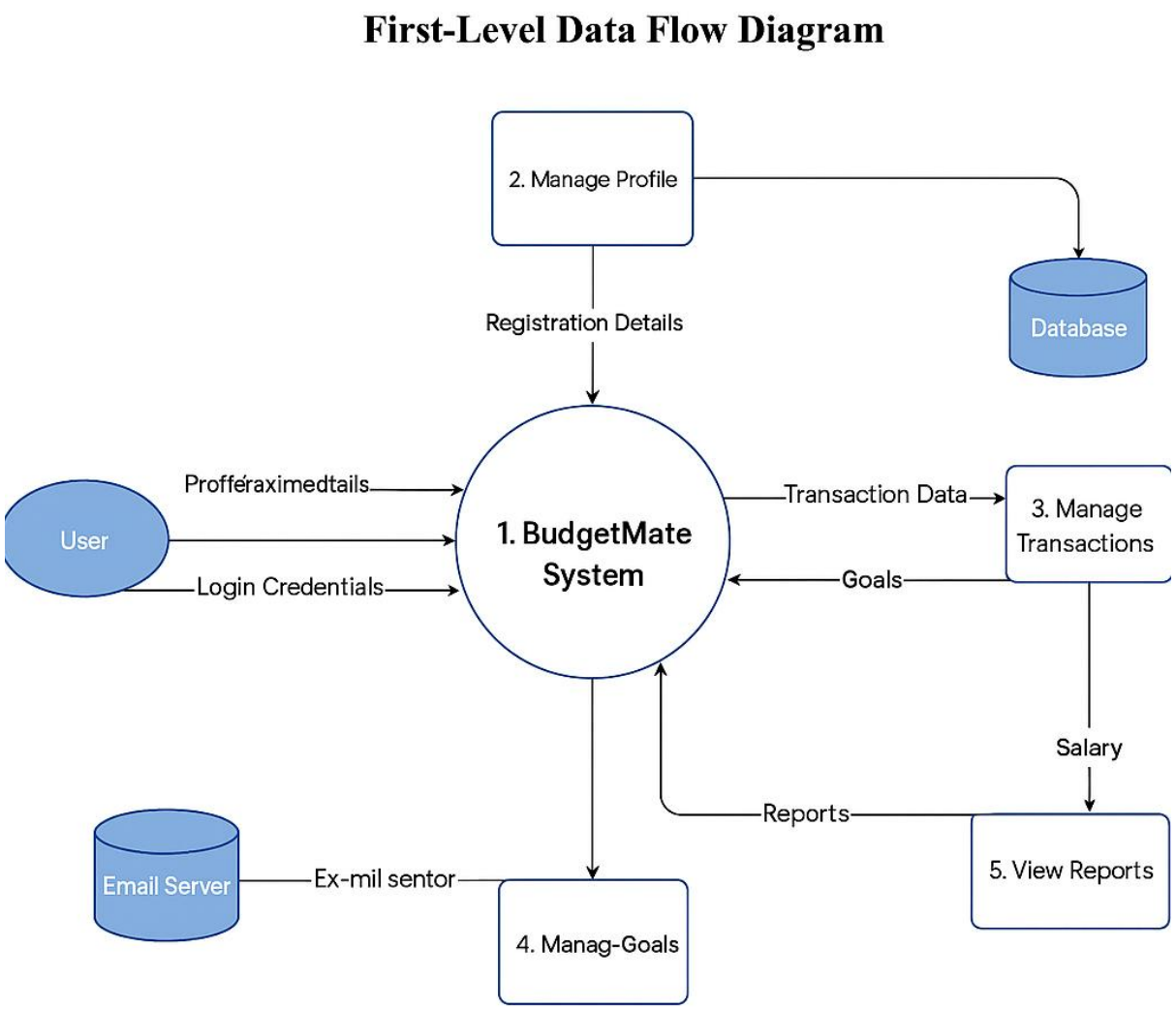
1. **User:** Interacts with the system to log in, browse menu, place orders, and give reviews.
2. **Payment Gateway:** Processes payments securely.
3. **Database:** Stores all system data including users, products, and reviews.

Data Flow Summary:

- User provides login credentials (via Google), selects coffee, and initiates payment.

- System verifies user, processes the order, and updates the database.
- Confirmation and receipts are sent back to the user.

b) First-Level DFD



The **First-Level DFD** breaks the overall system into key functional processes:

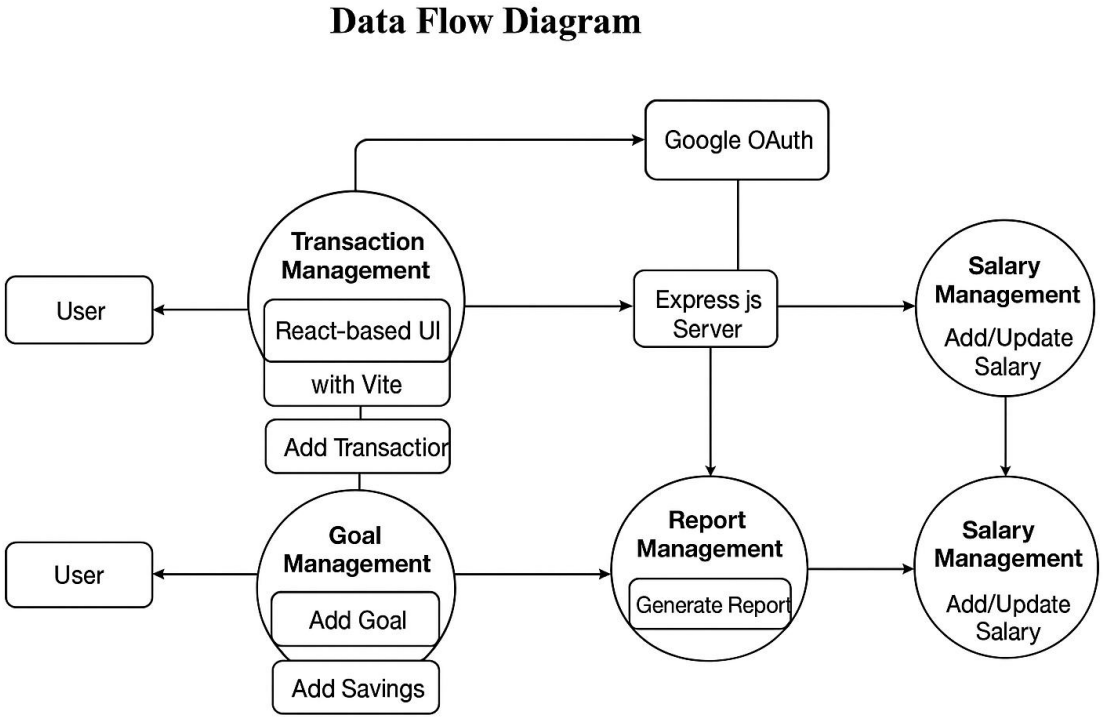
Processes:

1. **User Authentication**
    - User logs in via Google OAuth.
    - Data stored in User Table after verification.
2. **Menu Management**
    - Fetches all coffee items from the database.
    - Displays details to the user.
3. **Order Management**
    - Adds selected items to the cart.
    - Updates order details in the database.
4. **Payment Processing**
    - Sends payment data to gateway.
    - Confirms transaction success.
5. **Review Management**
    - Allows users to submit and retrieve reviews.

Data Stores:

- Users
- Coffee Items
- Orders
- Reviews

c) Second-Level DFD



The **Second-Level DFD** provides detailed internal flows of each module:

1. User Authentication

- Inputs: Google login credentials.
- Process: Token validation → user record creation (if new).
- Outputs: User session and profile details.
- Data Store: Users table.

2. Coffee Menu

- Inputs: Fetch request from frontend.
- Process: Retrieve coffee details via API.
- Outputs: List of coffee items.
- Data Store: Menu table.

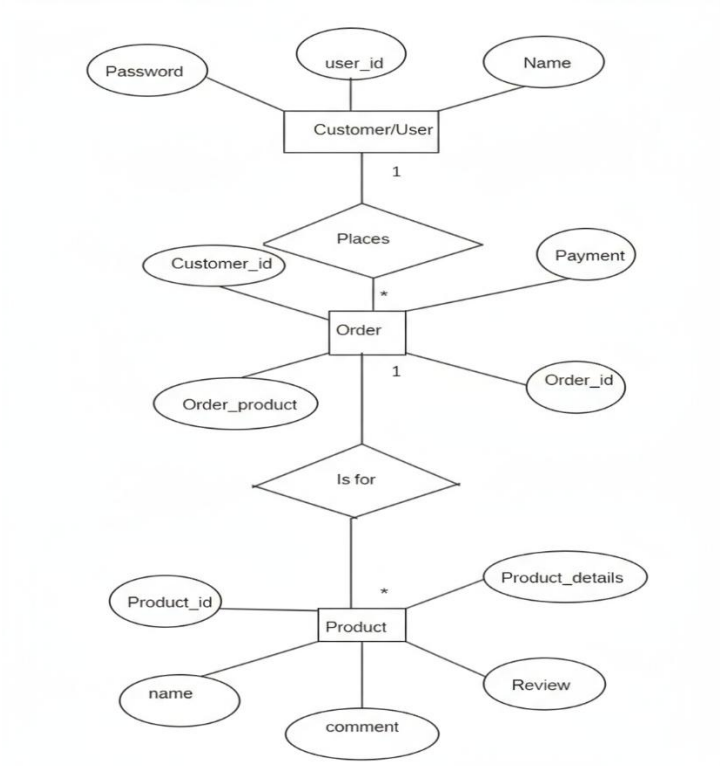
3. Order & Payment

- Inputs: Selected items and payment info.
- Process: Create new order → process payment → update order status.
- Outputs: Payment confirmation and receipt.
- Data Store: Orders and Payments tables.

4. Reviews

- Inputs: Review text and rating from user.
- Process: Save review → display aggregated feedback.
- Outputs: Updated list of reviews.
- Data Store: Reviews table.

3.4 ER Diagram (Entity Relationship Diagram)



The **ER Diagram** represents the logical structure of the database, defining relationships between entities.

Entities and Attributes

User

- Attributes: user_id (PK), name, email, profile_image
- Description: Stores basic user information after authentication.

Coffee Item

- Attributes: item_id (PK), name, description, price, image_url
- Description: Represents each coffee product available for order.

Order

- Attributes: order_id (PK), user_id (FK), total_amount, order_date, status
- Description: Stores order information associated with users.

OrderItem

- Attributes: order_item_id (PK), order_id (FK), item_id (FK), quantity
- Description: Tracks individual items within an order.

Review

- Attributes: review_id (PK), user_id (FK), item_id (FK), rating, comment, created_at
- Description: Stores reviews and ratings for coffee items.

Relationships

- **User → Order:** One-to-Many (a user can place multiple orders)
- **Order → OrderItem:** One-to-Many (each order can contain multiple coffee items)

- **User → Review:** One-to-Many (a user can post multiple reviews)
- **CoffeeItem → Review:** One-to-Many (a coffee item can have multiple reviews)
- **CoffeeItem → OrderItem:** One-to-Many (a coffee item can appear in multiple orders)

ER Diagram Summary:
The ER model ensures normalization and avoids redundancy by structuring data efficiently across five tables — Users, CoffeeItems, Orders, OrderItems, and Reviews.

3.5 Data Structures of All Modules

1. Users Table

Field	Type	Description
user_id	INTEGER (PK)	Unique ID for each user
name	TEXT	User's full name
email	TEXT	Email ID used for authentication
profile_image	TEXT	URL to user's profile picture

2. CoffeeItems Table

Field	Type	Description
item_id	INTEGER (PK)	Unique ID for each coffee item
name	TEXT	Coffee name
description	TEXT	Product description
price	REAL	Price of the coffee
image_url	TEXT	Image link

3. Orders Table

Field	Type	Description
order_id	INTEGER (PK)	Unique ID for each order
user_id	INTEGER (FK)	Linked to Users table
total_amount	REAL	Total cost of the order
order_date	DATETIME	Date and time of the order
status	TEXT	Order status (Pending/Completed)

4. OrderItems Table

Field	Type	Description
order_item_id	INTEGER (PK)	Unique item entry in order
order_id	INTEGER (FK)	Linked to Orders table
item_id	INTEGER (FK)	Linked to CoffeeItems table
quantity	INTEGER	Quantity of coffee ordered

5. Reviews Table

Field	Type	Description
review_id	INTEGER (PK)	Unique review ID
user_id	INTEGER (FK)	Linked to Users table
item_id	INTEGER (FK)	Linked to CoffeeItems table
rating	INTEGER	Rating out of 5
comment	TEXT	User review content
created_at	DATETIME	Date and time of submission

3.6 Summary

The **System Design** phase defines how different components of the Coffee Order System interact to provide a smooth user experience. The architecture ensures security, scalability, and modularity, while the ER

and DFD models guarantee efficient data flow and management. Together, these design elements lay a strong foundation for the implementation phase.

3.6 Procedural Design

1. User Panel Design

The User Panel allows customers to conveniently interact with the Coffee Order and Management System through a user-friendly interface.
Users can log in or register using their credentials to access personalized features. Once authenticated, they are directed to the dashboard, which displays essential information such as their active orders, order history, total spendings, and available offers.

From the dashboard, users can:

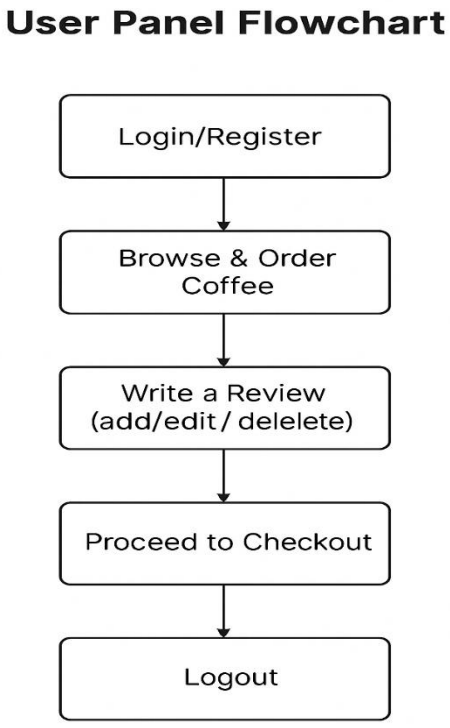
- **Browse the Coffee Menu:** View a list of available coffee items along with their names, descriptions, and prices.
- **Add Items to Cart:** Customize their coffee orders and add selected items to the shopping cart.
- **Proceed to Checkout:** Confirm the order, make payments, and receive a confirmation message.
- **View Order History:** Access past orders and check payment or delivery details.
- **Manage Reviews:** Add, edit, or delete reviews for purchased coffee items.
- **Logout:** Securely log out from the system once operations are complete.

The dashboard dynamically updates based on the user's actions — such as adding items, making payments, or posting reviews — ensuring real-time reflection of all activities.

Flowchart – User Panel:

The flow of activities in the User Panel starts with the **Login/Register** step, followed by navigation to the **Dashboard**, where the user can either **Browse Menu and Place Orders** or **View and Manage Reviews**.

Finally, the user proceeds to **Checkout** and completes the transaction before **Logging Out**.



4. Implementation

The **implementation phase** is where the system design is translated into a fully functional product. For the **Coffee Order System**, the implementation was carried out using a combination of modern web technologies — React.js for the frontend, Express.js for the backend, and SQLite with Prisma ORM for the database. This chapter provides an in-depth explanation of how various components of the system were developed and integrated to deliver a complete, secure, and user-friendly coffee ordering experience.

4.1 Technology Stack

The project adopts a **full-stack JavaScript architecture**, ensuring seamless communication between frontend and backend layers.

Component	Technology Used	Purpose
Frontend	React.js (with Vite)	User Interface and client-side routing
Backend	Express.js (Node.js Framework)	API development and server-side logic
Database	SQLite	Lightweight, embedded database for persistent storage
ORM	Prisma ORM	Schema management and database interaction
Authentication	Google OAuth 2.0	Secure user login and identity verification
API Communication	RESTful APIs (Axios)	Data transfer between client and server
Build Tool	Vite	Fast development and optimized build process
Language	JavaScript (ES6)	Common scripting language for all layers

This unified JavaScript-based ecosystem simplifies data handling and

improves maintainability while ensuring efficient performance across all modules.

4.2 Backend Implementation

The backend of the Coffee Order System was developed using **Express.js**, providing a robust and modular server-side environment.

a) Server Configuration

- The server runs on **Node.js**, handling HTTP requests and responses.
- Middleware functions such as express.json() and cors() are used to parse data and enable cross-origin communication.
- API endpoints are defined for handling authentication, menu items, orders, and reviews.

Example Routes:

- GET /api/menu – Fetch all coffee items from the database
- POST /api/order – Create a new order
- GET /api/reviews/:item_id – Fetch reviews for a specific product
- POST /api/reviews – Add a new review
- GET /api/health – Health check endpoint

b) Authentication Flow

- The backend verifies Google OAuth tokens received from the frontend.
- On first login, a user record is created in the database; for returning users, existing data is fetched.
- Authentication middleware ensures that only logged-in users can place orders or write reviews.

c) Payment and Order Management

- Checkout details (cart items, total price) are received via API requests.
- Orders are stored in the database with unique order IDs and timestamps.
- After payment confirmation, order status is updated to "Completed."

d) Review Management

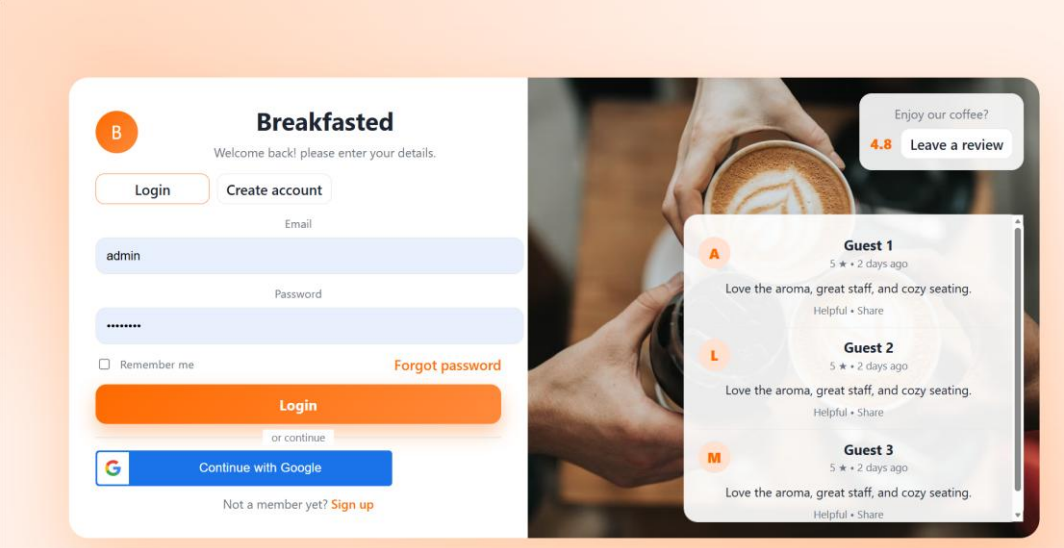
- The backend exposes routes for adding and fetching reviews.
- Prisma handles database operations for review insertion and retrieval, ensuring relational integrity with users and coffee items.

4.3 Frontend Implementation

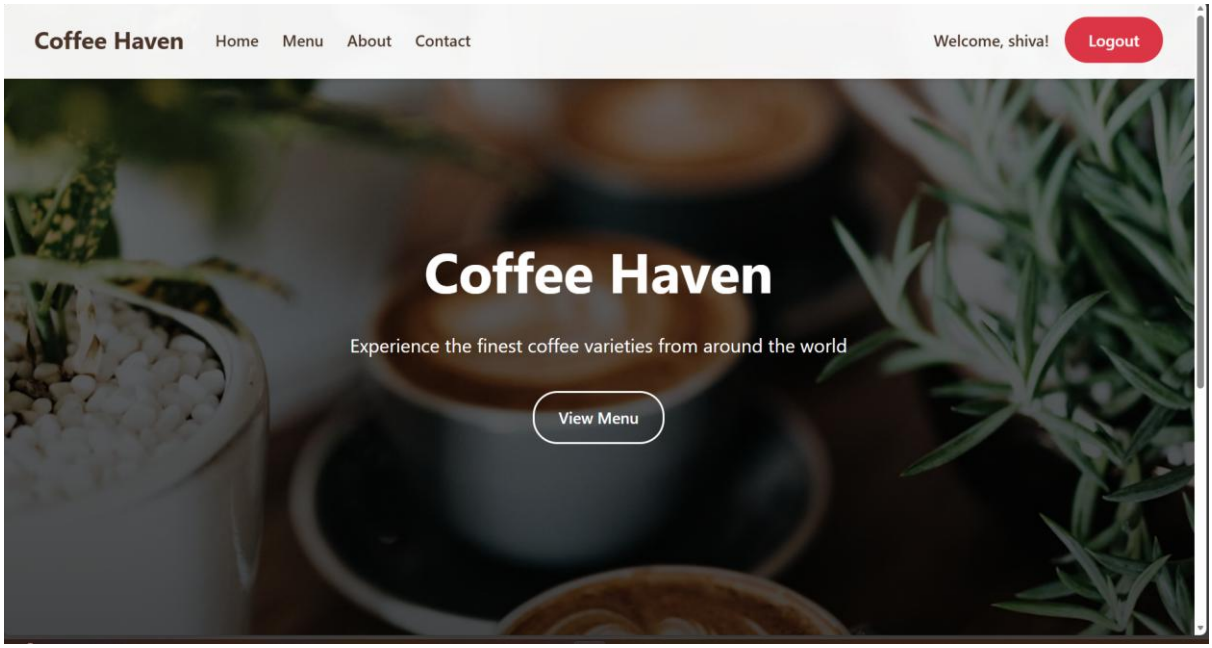
The frontend was built using **React.js**, designed for modularity, interactivity, and responsiveness.

a) Page Components

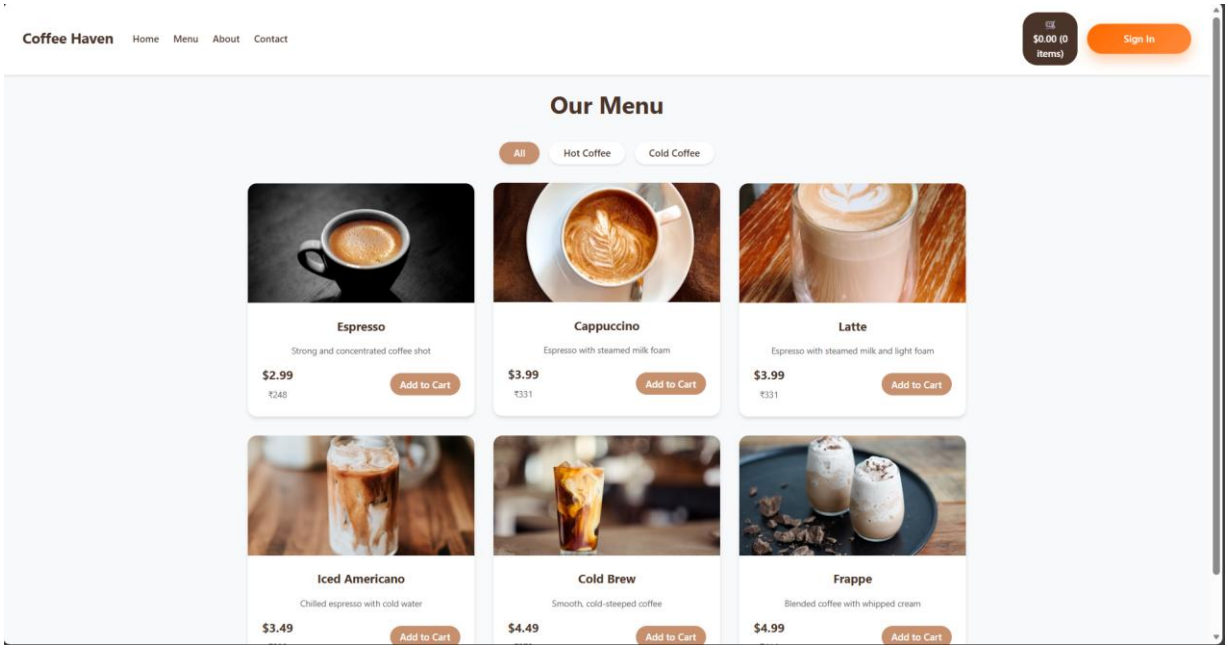
1. **Login Page:**
   - Integrates Google OAuth for secure login
   - Displays user profile upon successful authentication.



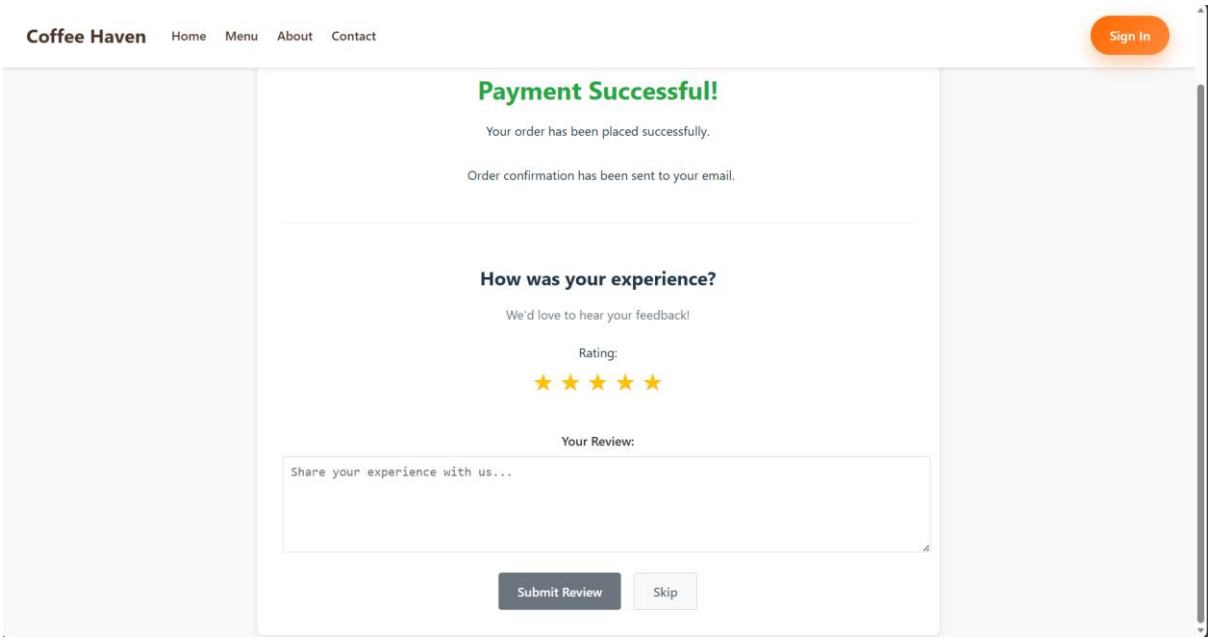
2. **Home Page:**
   - Displays the introduction and navigation links (Menu, Login, Checkout).
   - Promotes featured coffee products.



3. **Menu Page:**
   - Fetches and displays all available coffee items from the backend.
   - Allows users to add items to their cart and view detailed descriptions.



4. **Checkout Page:**
    - Shows order summary, total price, and payment button.
    - On successful payment, displays a confirmation message.
5. **Review and Payment Page:**
    - Displays existing reviews for each coffee product.
    - Allows logged-in users to submit new reviews and ratings.



b) Routing and State Management

- Implemented using **React Router DOM** for page navigation.
- **React Context API** is used to manage global user state (authentication and cart data).
- **Axios** is used for making asynchronous API requests to the backend.

c) UI and Design

- Clean, responsive layout using **CSS Modules** and inline styling.

- Simple color palette to ensure readability and focus on product visuals.
- Optimized for desktop and mobile screens.

4.4 Database Implementation

The database layer was developed using **SQLite** and managed through **Prisma ORM**.
Prisma provides schema-based data modelling, migrations, and query handling for efficient and secure operations.

a) Prisma Schema (Simplified Example)

```
model User {
  id      Int      @id @default(autoincrement())
  name    String
  email   String   @unique
  reviews Review[]
  orders  Order[]
}

model CoffeeItem {
  id          Int      @id @default(autoincrement())
  name        String
  description String
  price       Float
```

```
  reviews     Review[]
  orderItems  OrderItem[]
}

model Order {
  id         Int       @id @default(autoincrement())
  user       User      @relation(fields: [userId], references: [id])
  userId     Int
  total      Float
  status     String
  createdAt  DateTime   @default(now())
  items      OrderItem[]
}

model OrderItem {
  id       Int       @id @default(autoincrement())
  order    Order     @relation(fields: [orderId], references: [id])
  orderId  Int
  coffee   CoffeeItem  @relation(fields: [coffeeId], references: [id])
  coffeeId  Int
  quantity  Int
}

model Review {
  id       Int       @id @default(autoincrement())
```

```
  user      User       @relation(fields: [userId], references: [id])
  userId    Int
  coffee    CoffeeItem @relation(fields: [coffeeId], references: [id])
  coffeeId  Int
  rating    Int
  comment   String
  createdAt DateTime   @default(now())
}
```

b) Data Handling

- Data access is handled via Prisma's API: prisma.user.create(), prisma.order.findMany(), etc.
- Validation ensures that orders cannot be placed without authentication.
- Review records are linked using **foreign keys** to maintain referential integrity.

4.5 Authentication Implementation

Authentication is handled using **Google OAuth 2.0**, providing users with a fast and secure way to log in.

Authentication Flow:

1. The user clicks the "Login with Google" button.
2. Google redirects to an authentication consent screen.
3. After approval, Google sends an ID token to the frontend.
4. The frontend sends this token to the backend for verification.

5. Backend validates token using Google's API and stores user info in the database.
6. Upon successful verification, a session or JWT token is created for the user.

This approach ensures **data security, session persistence**, and **password-less authentication**.

4.6 Security Implementation

Security was prioritized at every level of development to ensure safe user interactions and data storage.

- **OAuth Authentication:** Protects user identity with Google's verified API.
- **Input Validation:** Prevents SQL injection and XSS attacks through Prisma ORM and React sanitization.
- **HTTPS Support:** Ensures encrypted communication between client and server (during deployment).
- **Protected Routes:** Only authenticated users can access checkout and review pages.
- **Error Handling:** All backend endpoints return proper status codes and JSON responses to prevent data leaks.

4.7 Integration

Integration was achieved through REST APIs connecting the frontend and backend layers.

Example flow:

- The frontend sends an API call GET /api/menu.
- The backend retrieves the menu data using Prisma ORM.

- The response is sent back as JSON and rendered dynamically on the React UI.
  This ensures real-time synchronization between the client and database.

4.8 Summary

The implementation phase of the **Coffee Order System** successfully transformed design models into a functional application. By combining React.js, Express.js, and SQLite with Prisma ORM, the system achieves fast performance, security, and scalability. Google OAuth authentication further enhances the trust and reliability of the platform. The integration of modern frameworks ensures that the system not only meets the defined requirements but also adheres to best practices in full-stack web development.

5. Testing

The testing phase ensures that the **Coffee Order System** functions accurately, efficiently, and reliably under various conditions. It validates that all modules — including authentication, menu browsing, checkout, payment, and reviews — perform according to the specified requirements. This phase also helps identify and resolve defects, ensuring the system is secure, user-friendly, and production-ready.

5.1 Objectives of Testing

The main objectives of testing are to:

- Verify that each module of the system works as intended.
- Ensure data integrity between the frontend, backend, and database.
- Validate the correct functioning of user authentication and authorization.
- Confirm secure and reliable payment handling.
- Check overall performance, usability, and responsiveness.
- Detect and fix any defects before deployment.

Testing was performed on both functional and non-functional components to ensure complete system validation.

5.2 Types of Testing

a) Unit Testing

Each component or function of the Coffee Order System was tested independently to verify its correctness.
Examples include:

- API endpoints for fetching menu items, creating orders, and submitting reviews.

- Frontend components such as login buttons, checkout forms, and review sections.

Tools Used: Jest (for backend) and manual browser-based testing (for frontend components).

b) Integration Testing

This testing phase focused on verifying the communication between different modules:

- React frontend communicating with Express backend via REST APIs.
- Backend accessing and updating data in the SQLite database via Prisma ORM.
- Google OAuth tokens being validated and user sessions maintained properly.

Example:
When a user logs in, their profile is verified through Google OAuth → stored in the database → menu is fetched and displayed in the user's dashboard.
This confirmed successful integration between all system layers.

c) System Testing

The system was tested as a complete, end-to-end solution to ensure overall reliability and performance.
Test scenarios included:

- User authentication and logout processes.
- Adding multiple items to cart and calculating total cost.
- Successful payment and order confirmation.
- Posting and viewing coffee reviews.
  All functionalities performed as expected without critical errors.

d) Security Testing

Security testing ensured that sensitive operations such as authentication and payments were properly protected.
Checks included:

- Unauthorized users cannot access protected routes (checkout, reviews).
- Validation of Google OAuth tokens on every request.
- Prevention of direct database manipulation (thanks to Prisma ORM).
- Verification that user information (email, name) is stored securely.

e) Usability Testing

Conducted to verify the application's user interface and navigation experience:

- Tested on both desktop and mobile devices.
- Checked for clear menu layout, readable fonts, and accessible buttons.
- Verified that all links and navigation routes worked correctly.

The UI was found to be intuitive, responsive, and visually appealing.

5.3 Test Cases

Below is a sample table of key test cases executed during the testing phase:

Test Case ID	Description	Input	Expected Output	Result
TC01	User Login (Valid Credentials)	Google OAuth login	Redirect to Home/Menu page	☑ Passed
TC02	User Login (Invalid Token)	Expired/Invalid Token	Error message: "Authentication Failed"	☑ Passed
TC03	Fetch Coffee Menu	API call to /api/menu	List of all coffee items	☑ Passed
TC04	Add Item to Cart	Select coffee item → Add	Item appears in cart list	☑ Passed
TC05	Checkout Process	Cart with valid items	Total amount displayed, payment option enabled	☑ Passed
TC06	Payment Confirmation	Valid payment details	"Payment Successful" message and order stored in DB	☑ Passed
TC07	Add Product Review	Rating + Comment	Review saved and displayed below product	☑ Passed
TC08	Unauthorized Access	Try accessing /checkout without login	Error message: "Access Denied"	☑ Passed
TC09	Fetch Reviews	API call /api/reviews/:id	List of reviews for selected coffee	☑ Passed
TC10	Invalid Route	Random URL	Error: "404 Not Found"	☑ Passed

All test cases were successfully executed and passed without major defects.

5.4 Testing Tools

To ensure comprehensive and accurate validation, the following tools were used:

Tool Name	Purpose
Postman	API endpoint testing and request/response verification
Jest	Backend unit testing framework for Express APIs
MySQL Workbench / Prisma Studio	Database inspection and validation
Browser Developer Tools (Chrome/Edge)	Frontend debugging and responsiveness testing
Manual Testing	UI, navigation, and usability evaluation

5.5 Test Results

The testing phase confirmed that all major modules and features of the **Coffee Order System** were functioning correctly and securely.
Results Summary:

- All functional test cases passed successfully.
- Authentication, checkout, and review features operated without data inconsistencies.
- No critical defects were found in performance or user interface.
- The system-maintained data integrity and consistent performance across multiple browsers.

Minor UI adjustments, such as text alignment and image scaling, were made during final testing to improve usability.

5.6 Summary

The testing phase validated the **Coffee Order System** against its specified requirements. Comprehensive unit, integration, and system

testing confirmed the system's reliability, performance, and security. All core functionalities — including **Google authentication, coffee menu display, order checkout, and review management** — worked seamlessly.
The system is stable, user-friendly, and ready for deployment.

6. Results and Discussion

The **Coffee Order System** was successfully developed and deployed as a full-stack web application integrating React.js, Express.js, and SQLite via Prisma ORM. This chapter presents the key outcomes of the system implementation, its performance evaluation, and insights gained during testing and deployment.

6.1 Results

a) Functional Results

All functional modules of the Coffee Order System performed successfully during the testing and validation phase.

Key Functional Achievements:

- **User Authentication:**
  Secure Google OAuth-based login and session handling were implemented successfully. Users can log in with their Google accounts without manual registration, ensuring simplicity and security.
- **Menu Module:**
  The coffee menu dynamically loads from the database. Each item displays its name, description, price, and image. Users can easily browse and select their preferred beverages.
- **Order and Checkout Module:**
  The checkout process was completed successfully with accurate total cost calculations. After confirming the payment, users receive a “Payment Successful” message along with order details.
- **Review Module:**
  Authenticated users can submit ratings and comments for coffee products. Reviews are displayed in real-time, enhancing interactivity and transparency.

- **Database Integration:**
  Prisma ORM ensured consistent data operations between backend APIs and the SQLite database. Relationships among Users, Orders, CoffeeItems, and Reviews were maintained effectively.

b) Non-Functional Results

In addition to functional performance, the system was evaluated on various non-functional parameters:

Criteria	Observation
Usability	Simple and intuitive user interface with clean navigation.
Performance	Quick response time for fetching menu and submitting reviews.
Scalability	Modular architecture allows for future feature expansion.
Security	OAuth authentication and protected routes enhance safety.
Reliability	Consistent data retrieval and accurate order tracking.
Compatibility	Works efficiently on major browsers (Chrome, Firefox, Edge).

6.2 Discussion

1. Achievement of Objectives

All core objectives of the Coffee Order System were successfully achieved:

- Developed a centralized digital platform for ordering coffee online.
- Integrated **Google OAuth** for secure and effortless authentication.
- Implemented **review management** to capture customer feedback.

- Ensured secure **checkout and payment confirmation** flow.
- Built a maintainable, scalable, and performance-optimized system using **React, Express, and Prisma**.

This implementation demonstrates how small to mid-sized cafés can adopt web-based solutions to streamline operations, improve customer engagement, and boost sales efficiency.

2. User Experience

The Coffee Order System provides users with a modern, visually appealing, and easy-to-use interface.

- Real-time updates ensure smooth interactions between the frontend and backend.
- The dynamic coffee menu and cart system enhance personalization.
- The integrated review feature allows customers to engage with products and share their experiences.

Users found the interface simple to navigate and appreciated the seamless transition between browsing, ordering, and payment processes.

3. Challenges Faced

During the development and testing phases, several challenges were encountered:

- **OAuth Integration:**
  Configuring Google OAuth required handling API credentials securely and managing callback URLs correctly.
- **Database Relationships:**
  Designing relational links between Orders, Users, and Reviews

while maintaining referential integrity required careful Prisma schema design.

- **Asynchronous Data Fetching:**
  Managing API responses and loading states in React needed optimization to prevent UI flickering and redundant API calls.
- **Payment Flow Testing:**
  Simulating real payment confirmation logic during testing without third-party gateway integration required the use of mock data.

All these challenges were resolved through iterative testing, use of middleware, and structured debugging.

4. Limitations

Although the system meets its primary objectives, certain limitations were identified:

- The payment system currently operates in a simulated mode (no real payment gateway integration).
- The admin module is basic and does not yet include detailed analytics or reporting.
- The application does not support order tracking or notifications after checkout.
- Mobile interface optimization could be further enhanced for smaller screens.

5. Future Enhancements

To make the Coffee Order System more advanced and scalable, the following improvements can be introduced:

1. **Real Payment Integration:**
   Integration with platforms like **Razorpay**, **PayPal**, or **Stripe** for live payment processing.

2. **Order Tracking:**
   Allow users to view real-time order status (e.g., “Preparing,” “Out for Delivery,” “Ready for Pickup”).

3. **Admin Analytics Dashboard:**
   Provide café owners with sales insights, best-selling products, and customer feedback reports.

4. **Push Notifications & Email Alerts:**
   Notify customers about order confirmations, discounts, or special offers.

5. **Mobile Application Version:**
   Extend the system into a React Native or Flutter-based mobile app for on-the-go accessibility.

6. **AI-based Recommendation System:**
   Suggest coffee items to users based on past orders and reviews.

7. **Multi-Language Support:**
   Add localization features to reach a wider customer base.

6.3 Summary

The **Coffee Order System** successfully achieved its primary goal of simplifying and digitizing the coffee ordering process.
It demonstrates a functional and secure full-stack solution with robust authentication, efficient order handling, and an interactive user interface.

The testing and results confirm that the system is:

- **Functionally reliable**,

- **Securely authenticated**, and
- **Technically scalable** for future enhancements.

This project serves as a solid foundation for real-world café management applications and can be extended into a complete e-commerce platform for beverages and bakery products.

7. Conclusion

The **Coffee Order System** was successfully designed, developed, and implemented as a full-stack web application that simplifies and modernizes the traditional coffee ordering process. The project demonstrates how emerging web technologies can be effectively integrated to deliver a seamless, secure, and user-centric experience.

Through the use of **React.js**, **Express.js**, and **SQLite (with Prisma ORM)**, the system achieves a robust architecture that ensures performance, scalability, and maintainability. The integration of **Google OAuth authentication** provides users with a secure and convenient way to log in, while features such as a **dynamic coffee menu**, **checkout system**, and **review module** enhance the overall functionality and interactivity of the application.

The successful completion of this project validates that modern cafés and beverage outlets can adopt lightweight web-based solutions to manage orders, engage with customers, and optimize service efficiency. Moreover, the modular design allows easy feature expansion, making it adaptable for future requirements and technological advancements.

Key Achievements:

- Seamless integration between frontend and backend for real-time order processing.
- Secure authentication using Google OAuth 2.0.
- Interactive user interface for menu browsing, checkout, and reviews.
- Proper database normalization ensuring reliable data management.
- Scalable architecture suitable for future business growth.

In conclusion, the **Coffee Order System** not only fulfills its defined objectives but also provides a strong foundation for future digital

solutions in the café and restaurant industry. It serves as an example of how technology can enhance everyday experiences through automation, simplicity, and secure interaction.

8.Future Scope

While the current version of the Coffee Order System achieves its core objectives, there are several opportunities for future improvement and feature expansion. These enhancements can further increase usability, scalability, and business value.

1. Integration of Live Payment Gateways:
Incorporating services like **Razor pay**, **Stripe**, or **PayPal** would enable real-time, secure payment transactions, eliminating the need for simulation-based processing.

2. Order Tracking and Notifications:
Implementing live order status updates ("Preparing," "Ready for Pickup," etc.) and push notifications would improve user engagement and transparency.

3. AI-Powered Recommendations:
Machine learning algorithms can be introduced to suggest personalized coffee options based on user history, preferences, and reviews.

4. Advanced Analytics Dashboard (Admin Panel):
A comprehensive dashboard for café owners can display insights such as top-selling products, peak hours, and customer satisfaction trends.

5. Mobile Application Version:
Developing an Android/iOS app using **React Native** or **Flutter** would make the system more accessible and increase its market reach.

6. Multi-Store and Multi-User Support:
The system can be expanded to support multiple coffee shops, enabling centralized management of orders, menus, and users across branches.

7. Loyalty Points and Reward System:
Gamification features such as reward points for frequent purchases or reviews can boost customer retention and engagement.

8. Multi-Language Interface:
Supporting multiple languages will make the system more inclusive for diverse user demographics and regions.

7.3 Summary

The **Coffee Order System** successfully bridges the gap between traditional café operations and modern digital service delivery. Its modular, secure, and scalable design lays the groundwork for continuous innovation. With the proposed future enhancements, the system can evolve into a fully integrated digital café management platform — offering intelligent insights, real-time communication, and a superior customer experience.

8. Bibliography and References

In preparing this project report on the **Coffee Order System**, various books, online resources, technical documentations, and tutorials were consulted. These references were invaluable in understanding system design, web development practices, authentication mechanisms, and database integration.

Books

1. Rajaraman, V. — *Fundamentals of Computers*, Prentice Hall of India.
2. Abraham Silberschatz, Henry F. Korth, S. Sudarshan — *Database System Concepts*, McGraw Hill.
3. Roger S. Pressman — *Software Engineering: A Practitioner's Approach*, McGraw Hill.
4. Ian Sommerville — *Software Engineering*, Pearson Education.
5. Ethan Brown — *Web Development with Node and Express: Leveraging the JavaScript Stack*, O'Reilly Media.
6. Alex Banks & Eve Porcello — *Learning React: Modern Patterns for Developing React Apps*, O'Reilly Media.

Research Papers / Journals

1. *International Journal of Computer Applications (IJCA)* — "Full Stack Web Application Development Using Modern JavaScript Frameworks."
2. *IEEE Xplore Digital Library* — "Enhancing Customer Experience Through Web-Based Ordering Systems."
3. *International Journal of Emerging Technologies in Engineering Research (IJETER)* — "Implementation of Secure Authentication in Web Applications Using OAuth 2.0."

Websites

1. https://react.dev — Official **React.js** documentation.
2. https://expressjs.com — Official **Express.js** documentation.
3. https://www.prisma.io/docs — **Prisma ORM** schema and API documentation.
4. https://developers.google.com/identity — **Google OAuth 2.0** integration and token verification.
5. https://www.sqlite.org/docs.html — **SQLite** official database documentation.
6. https://axios-http.com/ — Documentation for **Axios** library used in API communication.
7. https://vitejs.dev/ — Official **Vite** documentation for frontend build configuration.
8. https://nodejs.org/en/docs — **Node.js** official reference documentation.
9. https://developer.mozilla.org/ — **MDN Web Docs** for JavaScript, HTML, and CSS references.

Other Resources

- Online tutorials, practical demonstrations, and developer community discussions from:
    - **Stack Overflow** — troubleshooting and debugging support.
    - **Free Code Camp** and **W3Schools** — conceptual guides for full-stack development.
    - **GitHub Repositories** — open-source examples for REST API and React authentication.