

MYBEDRA



INDEX

1. PROJECT SYNOPSIS	1-6
1.1. INTRODUCTION TO THE SYSTEM	2
1.1.1. PROJECT TITLE	2
1.1.2. CATEGORY	2
1.1.3. OVERVIEW	2
1.2. BACKGROUND	3
1.2.1. ABOUT COMPANY	3
1.2.2. BRIEF NOTE ON EXISTING SYSTEM	3
1.3. OBJECTIVE OF THE SYSTEM	3
1.4. SCOPE OF THE SYSTEM	4
1.5. STRUCTURE OF THE SYSTEM	5
1.6. SYSTEM ARCHITECTURE	6
1.7. SOFTWARE/HARDWARE USED FOR THE DEVELOPMENT	6
1.8. SOFTWARE/HARDWARE REQUIRED FOR THE IMPLEMENTATION	6
2. SOFTWARE REQUIREMENT SPECIFICATION	7-12
2.1. INTRODUCTION	8
2.2. OVERALL DESCRIPTION	8
2.2.1. PRODUCT PERSPECTIVE	9
2.2.2. PRODUCT FUNCTION	9
2.2.3. USER CHARACTERISTICS	9
2.2.4. GENERAL CONSTRAINTS	10
2.2.5. ASSUMPTIONS	10
2.3. FUNCTIONAL REQUIREMENTS	10
2.3.1. USER MODULE	10
2.3.2. BUS MODULE	10
2.3.3. AUTORICKSHAW MODULE	10
2.3.4. FOOD MODULE	11
2.3.5. GROCERY MODULE	11
2.3.6. BUSINESS MODULE	11
2.4. DESIGN CONSTRAINTS	11
2.5. SYSTEM ATTRIBUTES	11
2.6. OTHER REQUIREMENTS	12
3. SYSTEM DESIGN	13-20
3.1. INTRODUCTION	14
3.2. ASSUMPTIONS AND CONSTRAINTS	14
3.3. FUNCTIONAL DECOMPOSITION	14
3.3.1. SYSTEM SOFTWARE ARCHITECTURE	15
3.3.2. SYSTEM TECHNICAL ARCHITECTURE	15
3.3.3. SYSTEM HARDWARE ARCHITECTURE	15
3.4. DESCRIPTION OF PROGRAMS	15

3.4.1. CONTEXT FLOW DIAGRAM	15
3.4.2. DATA FLOW DIAGRAM	16
3.5. DESCRIPTION OF COMPONENTS	19
4. DATABASE DESIGN	21-27
4.1. INTRODUCTION	22
4.2. PURPOSE OF THE DATABASE	22
4.3. DATABASE IDENTIFICATION	22
4.4. SCHEMA INFORMATION	23
4.5. TABLE DEFINITION	23
4.5.1. TABLES	23
4.6. PHYSICAL DESIGN	26
4.7. DATA DICTIONARY	26
4.8. ER DIAGRAM	27
4.9. DATABASE ADMINISTRATION	27
4.9.1. SYSTEM INFORMATION	27
4.9.2. SUPPORT SOFTWARE REQUIRED	27
4.9.3. STORAGE REQUIREMENTS	27
4.9.4. BACKUP AND RECOVERY	27
5. DETAILED DESIGN	28-31
5.1. INTRODUCTION	29
5.2. STRUCTURE OF THE SOFTWARE PACKAGE	29
5.3. MODULAR DECOMPOSITION OF THE SYSTEM	31
5.3.1. BUS MODULE	31
5.3.2. AUTORICKSHAW MODULE	31
5.3.3. FOOD MODULE	31
5.3.4. GROCERY MODULE	31
5.3.5. BUSINESS MODULE	31
6. PROGRAM CODE LISTING	32-95
7. USER INTERFACE	96-106
7.1. ANDROID VIEW	97
7.1.1. LOGO	97
7.1.2. LOGIN PAGE	97
7.1.3. REGISTRATION PAGE	97
7.1.4. HOME PAGE	97
7.1.5. FORGOT PASSWORD PAGE	97
7.1.6. CATEGORY SEARCH	97
7.1.7. BUS PAGE	97
7.1.8. BUS DROPDOWN	98
7.1.9. BUS SEARCH	98
7.1.10. AUTO PAGE	98
7.1.11. AUTO DROPDOWN	98
7.1.12. AUTO SEARCH	98

7.1.13. ON TAP CALL	98
7.1.14. GROCERY PAGE	99
7.1.15. CART PAGE	99
7.1.16. ERROR PAGE	99
7.1.17. PROFILE PAGE	99
7.1.18. SUCCESSFUL ORDER	99
7.2. WEBSITE VIEW	100
7.2.1. LOGIN PAGE	100
7.2.2. DASHBOARD	100
7.2.3. ADD BUSINESS	101
7.2.4. DISPLAY BUSINESS LISTING	101
7.2.5. ADD BUS	102
7.2.6. DISPLAY BUS LISTING	102
7.2.7. ADD AUTORICKSHAW	103
7.2.8. DISPLAY AUTO LISTING	103
7.2.9. DISPLAY USER DETAILS	104
7.2.10. ADD GROCERY	104
7.2.11. ADD FOOD	105
7.2.12. DISPLAY ORDER LIST	105
7.2.13. DISPLAY ACCEPTED ORDER LIST	106
8. TESTING	107-111
8.1. INTRODUCTION	108
8.2. TEST REPORTS	109
8.2.1. UNIT TESTING	109
8.2.2. INTEGRATION TESTING	109
8.2.3. SYSTEM TESTING	109
8.3. TEST CASES	111
CONCLUSION	117
LIMITATIONS	117
FUTURE SCOPE	117
ABBREVIATION AND ACRONYMS	118
BIBIOGRAPHY / REFERENCES	118

CHAPTER

-1-

INTRODUCTION

1. Introduction

1.1. Introduction of the System

myBedra is a mobile application developed using Flutter and has its backend written in node.js and SQL. It is designed to provide information and communication services to the people living in Moodbidri, a town located in the state of Karnataka, India. The app offers various modules such as Bus, Autorickshaw, Food, Grocery, and Businesses, which provide detailed information about different services available in the town.

The primary goal of the myBedra app is to provide a platform for the people of Moodbidri to access information about various services such as bus timings and routes, autorickshaw details, food, grocery stores, and panchayat office details. The app is aimed to be an all-in-one solution for the people of Moodbidri to access all necessary information from one place

1. Project Tittle

myBedra

2. Category

Android based application

3. Overview

The project, named "myBedra," is a comprehensive mobile application developed using Flutter, with its backend implemented in node.js and SQL. The application is specifically designed to cater to the needs of the residents of Moodbidri, a town situated in the state of Karnataka, India. With the aim of enhancing information dissemination and communication within the community, myBedra offers a wide range of services through its various modules.

The application's primary objective is to provide users with easy access to vital information about the town's services and amenities. By leveraging the power of technology, myBedra acts as a centralized platform that offers a variety of modules such as Bus, Autorickshaw, Food, Grocery, and Businesses. Each module is designed to provide detailed information about the corresponding services available within the town.

Overall, myBedra aims to revolutionize the way residents of Moodbidri access information and communicate within their town. By providing a user-friendly interface

and a comprehensive set of modules, the application empowers users to make informed decisions, optimize their daily routines, and actively participate in the local economy.

1.2. Background

1.2.1. About Company

Build dreams is a software and construction firm built in 2017 by Sheik Mohammed Alfaz to provide the best quality software and construction solution located in Divya Sagar Palace, Ground floor, Bypass road, Karkala-574104, India.

1.2.2. Brief Note on Existing System

The existing system in Moodbidri prior to the introduction of myBedra was characterized by limited access to information and fragmented services. Residents relied on traditional methods to obtain information about bus schedules, autorickshaws, Food, grocery stores, and local businesses, often leading to inefficiencies and inconvenience.

Information regarding bus routes and schedules was primarily obtained through physical bus stops or by word of mouth, resulting in uncertainty and potential delays for commuters. Locating and booking autorickshaws required manual effort and was dependent on personal networks or randomly available ones.

Finding suitable Food and making reservations involved relying on personal recommendations or manually visiting different establishments to gather information. Similarly, identifying nearby grocery stores, accessing product details, and placing orders often required physical visits or reliance on limited local directories.

Overall, the existing system lacked a centralized platform that provided comprehensive information and communication services, leading to challenges in accessing essential resources and optimizing daily routines. The introduction of myBedra aimed to bridge these gaps and revolutionize the way residents accessed information and services in Moodbidri.

1.3. Objective of the System

- **Provide Convenient Information Access:** The system aims to offer residents of Moodbidri easy and convenient access to information about bus schedules, autorickshaws, Food, grocery stores, and local businesses.

- **Enhance Commuting Experience:** The system aims to optimize the commuting experience by providing real-time information on bus routes, schedules, and arrival times, as well as facilitating the location and booking of autorickshaws.
- **Promote Local Businesses:** The system aims to promote and support local businesses by providing a platform for residents to discover and engage with a diverse range of products and services offered within the town.
- **Streamline Dining Experience:** The system aims to enhance the dining experience by offering a comprehensive directory of Food, enabling users to explore menus, read reviews, make reservations, and discover new culinary experiences.
- **Simplify Grocery Shopping:** The system aims to simplify the process of grocery shopping by providing information on nearby grocery stores, available products, special offers, and the ability to create shopping lists and place orders conveniently.
- **Foster Economic Growth:** The system aims to contribute to the economic growth of Moodbidri by promoting local businesses, encouraging community support, and facilitating transactions within the town's marketplace.
- **Enhance Quality of Life:** The system aims to enhance the overall quality of life for Moodbidri's residents by providing them with a user-friendly and efficient platform that simplifies access to essential services and information, ultimately saving time and effort.

1.4. Scope of the System

- **Information Access:** The project aims to provide the people of Moodbidri with a comprehensive platform to access essential information about various services, including bus timings and routes, autorickshaw details, Food, grocery stores, and business information. Users can rely on the app as a centralized source for obtaining such information.
- **Timely Updates:** The app allows users to update bus timings and add new bus routes, ensuring that the information remains accurate and up-to-date. This feature enables the community to actively contribute to the app's information ecosystem, fostering a collaborative and dynamic environment.
- **All-in-One Solution:** The myBedra app seeks to be an all-in-one solution for the people of Moodbidri, offering a wide range of information and services within a single platform. Users can conveniently access all necessary information and

resources from one place, streamlining their daily routines and enhancing efficiency.

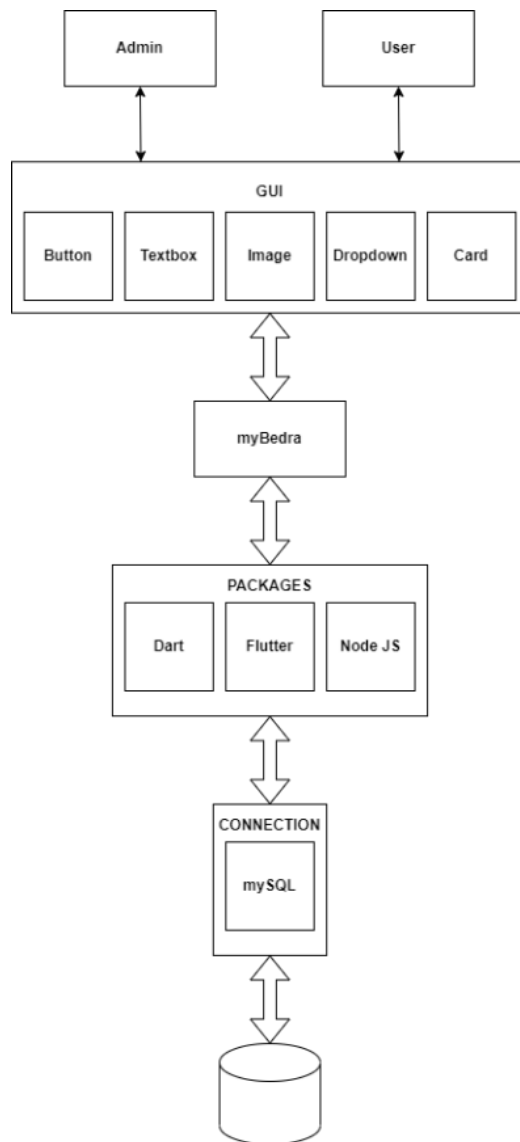
- **Focus on Convenience:** The primary goal of the project is to provide convenience to the people of Moodbidri. By offering a user-friendly interface and centralized access to essential information, the app aims to simplify the lives of residents, making it easier for them to navigate the town and access the services they need.

1.5. Structure of the System

- **User Interface (UI):** The user interface serves as the front-end component of the myBedra app, allowing users to interact with the system. It includes screens, menus, and controls for accessing various modules and functionalities.
- **Modules:**
 - **Bus Module:** This module provides information about bus timings, routes, and real-time updates. Users can view bus schedules of different busses of different routes.
 - **Autorickshaw Module:** The autorickshaw module enables users to locate autorickshaws of different auto stands, view their details such as driver name and his contact details, and book rides by calling him.
 - **Food Module:** The Food module provides a comprehensive directory of local dishes available at local eateries. Users can explore various local dishes and order it.
 - **Grocery Module:** The grocery module facilitates a directory of grocery items. Users can create shopping lists and place orders.
 - **Businesses Module:** The businesses module acts as a directory of local enterprises, showcasing a wide range of products and services. Users can search for specific type of businesses, view details such as their contact information.
- **Backend:** The backend component handles the processing and storage of data. It is implemented using node.js and SQL, ensuring robustness and scalability. The backend manages data related to bus timings, autorickshaw details, local dish listings, grocery products and business information.
- **Data Management:** The data management component includes database systems to store and retrieve information efficiently. SQL databases are utilized to manage

structured data related to bus schedules, autorickshaw details, local dish listings, grocery products and business listings.

1.6. System Architecture



1.7. Software / Hardware Used For The Development

- **Software used:** Visual Studio Code, XAMPP.
- **Hardware used:** HP laptop with 8GB RAM, 128GB SSD and Ryzen5 processor.

1.8. Software / Hardware Required For The Implementation

- **Software:** Android Version 6.0 Marshmallow or higher

- **Hardware used:**
 - Processor: Dual-core 1.2 GHz or higher
 - RAM: 2 GB or higher
 - Storage: Sufficient free space to install the myBedra mobile application
- **Internet Connectivity:** A stable internet connection

CHAPTER

-2-

SOFTWARE REQUIREMENT SPECIFICATION

2. Software Requirement Specification

2.1. Introduction

The requirements phase of the software development process aims to produce a comprehensive Software Requirements Specification (SRS) that defines the complete external behaviour of the proposed software system. This includes detailing the functional requirements, as well as the non-functional requirements and user-interface specifications. During requirement analysis, the focus is on understanding the problem the software system intends to solve, whether it involves automating existing manual processes, developing new systems, or a combination of both. For complex systems with multiple features and tasks, comprehending the system's requirements becomes a significant undertaking. The software should exhibit simplicity, with an intuitive and interactive interface, prompting users to log in and enter proper inputs. Emphasis is placed on identifying the needed system functionality rather than the specific implementation details. Compliance with standards is outlined, encompassing report formats, accounting procedures, and other necessary guidelines. Hardware limitations, such as supported machines and operating systems, as well as considerations of reliability, fault tolerance, and security requirements, play a crucial role. Performance characteristics are also addressed, distinguishing between static requirements that don't impact system execution and dynamic requirements that specify execution behavior, including response time and throughput constraints.

The Software Requirements Specification (SRS) serves as a foundational document that outlines the functional and non-functional requirements of a software project. It provides a comprehensive description of what the software is expected to do, its features, and the constraints and limitations within which it must operate. The SRS acts as a blueprint for the development team, guiding them throughout the software development life cycle.

The purpose of an SRS is to establish a clear understanding between the project stakeholders, including the clients, developers, and testers, about the goals and objectives of the software project. It serves as a communication tool that ensures all parties involved have a shared understanding of the software's functionalities, behavior, and scope.

2.2. Overall Description

This section aims to provide a detailed overview of the entire system. We will discuss the system's context and how it interacts with other systems to give you a better understanding of its functionality. Additionally, we will introduce the basic features of the system and explain how they work.

We will also describe the different types of users who will be using the system and outline the specific functionality available to each user group. This will help you understand how the system can be used by different people and what features are most relevant to their needs.

Overall, this section will give you a comprehensive understanding of the system and its capabilities, helping you make informed decisions about how to use it effectively.

1. Product Perspective

myBedra is an innovative mobile application that aims to connect the residents of Moodbidri with various aspects of their town. The app is developed using Flutter, a cross-platform framework that allows for fast and beautiful user interfaces. The backend of the app is written in node.js and SQL, which ensure reliable and secure data management. The app has several modules that cater to different needs and interests of the users. For example, the Bus module shows the timings and routes of the buses that operate in Moodbidri, the Autorickshaw module allows users to book an autorickshaw ride by calling the driver, the Food module displays the local dishes of different eateries in town, the Grocery module helps users find and order groceries, and the Business module provides information about the local businesses. MyBedra is designed to be a one-stop solution for all the information and communication needs of Moodbidri residents. It is easy to use, informative, and convenient.

2. Product Function

- The android application has a user-friendly interface and is easy to use.
- It provides bus timing, so that user can come at the relevant time to the bus stop.
- It provides auto-rickshaw drivers numbers, such that the user can book an auto for himself.
- It provides grocery details available in the store.
- It provides local dish such that the user can order food.
- It provides local business details.

3. User Characteristics

- The prior knowledge about the flutter is not required for the user.
- User should have basic knowledge about the phone.
- User should be familiar with the Internet.

4. General Constraint's

- The app must provide accurate and up-to-date information about the various services available in Moodbidri.
- The app must be easy to use and navigate for users of all ages and technical abilities.
- The app must comply with all relevant data protection and privacy regulations.
- The device running this application should be connected to internet.

5. Assumptions

- The app will be used primarily by residents of Moodbidri who have access to a smartphone and an internet connection.
- The information provided by the app about various services in Moodbidri will be accurate and up-to-date.
- Users will be able to easily navigate the app and find the information they need.
- The app will be able to handle a large number of users without any performance issues.
- The app will comply with all relevant data protection and privacy regulations.

2.3. Functional Requirements

1. User Module

- This module allows the users to register themselves to the system.
- Registration page allows new users to create a profile.
- Login page allows the existing users to login to the system, and user authentication.
- Forget password page to change user password if the user forgets the password.

2. Bus Module

- The Bus module provides information about bus timings and routes in Moodbidri.
- Users can search for specific bus routes.
- Bus data is stored in a database and is updated by admin.

3. Autorickshaw Module

- The Autorickshaw module displays available autorickshaws in Moodbidri.
- The driver names and their contact details of the autorickshaws are shown.
- Users can call the driver directly by clicking on driver profile.

4. Food Module

- The Food module provides information about dishes available local eateries.
- The user can order any dish that is available.

5. Grocery Module

- The Grocery module displays all the grocery items in Moodbidri.
- The user can order any grocery that is available.

6. Business Module

- The Business module provides information about different local businesses available in Moodbidri.
- The contact details of the businesses are displayed.

2.4. Design Constraints

- The application must be designed in such a way that it should be easy to understand and use.
- It should be able to support all recently launched android phones.
- It should support old android phone which has android version 5.0 and above

2.5. System Attributes

- **Reliability:** The system must have a high level of reliability and be able to recover quickly from any failures.
- **Availability:** The system must be available 24/7 and accessible to users from anywhere in the world.
- **Maintainability:** The system must be easy to maintain and update. Any changes or updates to the system must be able to be made quickly and efficiently without causing any disruption to users.

- **Portability:** The system must be portable and able to run on multiple platforms and devices.

2.6. Other Requirements

- **Legal Requirements:** The system must comply with all relevant laws and regulations, including data protection and privacy laws.
- **Database Requirements:** The system must use a secure and reliable database to store user data. The database must be able to handle large amounts of data and support fast query times.
- **Reuse Objectives:** The system should be designed in a modular way to allow for easy reuse of components in future projects.

CHAPTER

-3-

SYSTEM DESIGN

3. System Design

3.1. Introduction

System design is a crucial phase in the software development life cycle where the requirements gathered during the earlier phases are transformed into a detailed and comprehensive blueprint for building the software system. It involves designing the architecture, components, modules, interfaces, and data structures that will collectively enable the system to fulfil its intended functionalities. The goal of system design is to create a robust, scalable, and maintainable solution that meets the specified requirements and aligns with the overall project objectives.

Systems design is a crucial phase in software development that involves establishing the architecture, product design, modules, interfaces, and data to meet specified requirements. It applies systems theory to guide product development, overlapping with disciplines such as systems analysis, systems architecture, and systems engineering. The primary focus of system design is determining the necessary modules for the system and specifying their interconnectedness. Referred to as top-level design, system design encompasses a collection of components with well-defined behaviour, interacting in a predetermined manner to produce desired outcomes. The design process entails defining modules and their functional abstractions, forming a structured framework for building the system.

3.2. Assumptions And Constraints

- Roles and tasks are predefined
- Network and data availability
- Power supply
- Better connection for exchanging data over network
- Availability of mobile/desktop service

3.3. Functional Decomposition

Functional decomposition is a technique used in system design to break down complex functionalities into smaller, manageable modules or components. By decomposing the system's functionality, it becomes easier to understand, analyse

-9, and develop each component independently. This approach allows for a more modular and efficient design, where each module performs a specific function and can be easily maintained or updated.

1. System Software Architecture

The system software architecture refers to the high-level structure and organization of software components within the system. It defines the relationships, interactions, and dependencies between different software modules or subsystems. The software architecture guides the design and development of the system's software components, ensuring that they work together harmoniously to achieve the desired functionality.

2. System Technical Architecture

The system technical architecture encompasses the technical infrastructure and components required to support the system's operation. It includes the hardware, software, network infrastructure, and other technical elements that are necessary for the system to function effectively. The technical architecture ensures that the system's hardware and software components are appropriately integrated, reliable, and scalable to meet the system's requirements. It provides a blueprint for the system's technical implementation and maintenance.

3. System Hardware Architecture

The system hardware architecture focuses on the physical components and infrastructure required to support the functioning of the system. It encompasses the selection, configuration, and arrangement of hardware elements such as servers, computers, storage devices, networking equipment, and peripherals.

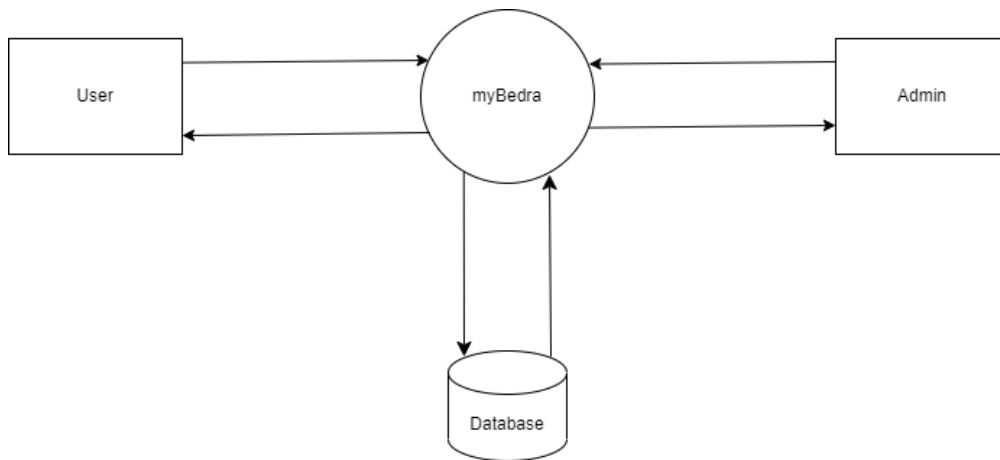
The hardware architecture considers factors such as processing power, memory capacity, storage capacity, connectivity options, and reliability to ensure optimal system performance. It involves determining the appropriate hardware components and their interconnections to meet the system's computational, storage, and communication requirements.

3.4. Description Of Programs

1. Context Flow Diagram

A context diagram, also known as a level 0 data flow diagram, plays a pivotal role in establishing and delineating the boundaries of a software system. It serves to provide a clear understanding of the information flow between the system and external entities. By presenting the entire software system as a single process, the context diagram offers a concise and holistic view of the system's interactions and connections.

It serves as a valuable tool to define and clarify the scope of the software system, ensuring a comprehensive understanding of its external relationships and information exchanges.



2. Data Flow Diagrams

A Data Flow Diagram (DFD) is a graphical representation that illustrates the flow of data within a system. It depicts the processes, data sources, data destinations, and the flow of data between them. DFDs are commonly used in software engineering and system analysis to model the information flow and provide a visual representation of how data moves through a system.

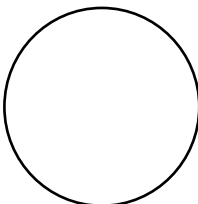

DFDs are valuable for system analysis, design, and communication as they offer a clear visualization of the data flow and help identify inefficiencies, redundancies, and opportunities for optimization within a system.

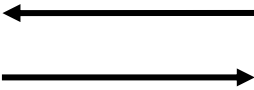

Rules for designing DFD:

- **No Process Splitting:** Each process in a DFD should represent a single, cohesive activity or transformation.
- **No Process Joining:** Data flows should not be combined or merged before reaching a process.
- **Data Flow Direction:** Data flows should always move from a source (such as an external entity or data store) to a destination (such as a process or data store).
- **No Data Stores between Processes:** Data stores should not be placed between processes; they should be connected directly to processes or external entities.
- **Consistent Naming:** Use consistent and meaningful names for processes, data flows, data stores, and external entities to enhance clarity and understanding.

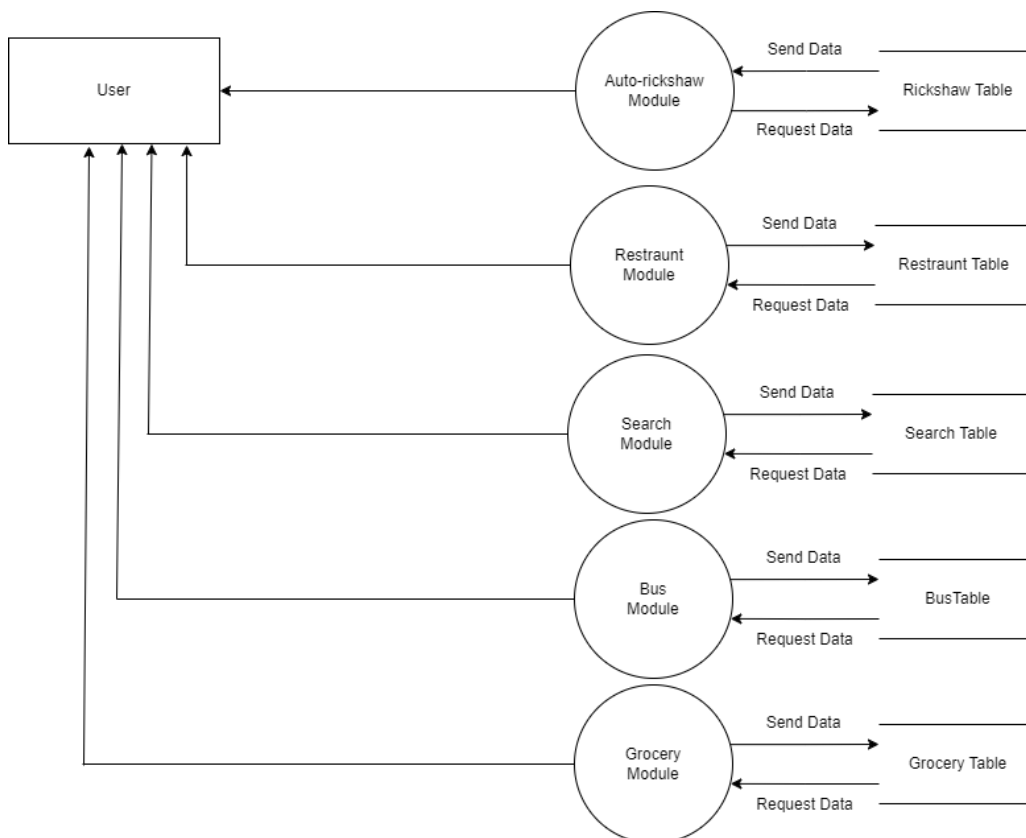
- Balancing Inputs and Outputs: Each process should have balanced inputs and outputs, ensuring that all incoming data flows are accounted for in the outgoing data flows.
- No Control Information in Data Flows: Avoid including control information, such as system commands or status indicators, in data flows. Control information should be represented separately.
- A process cannot have only outputs.
- A process cannot have only inputs.
- The inputs to a process must be sufficient to produce the outputs from the process.
- All data stores must be connected to at least one process.
- All data stores must be connected to a source or sink.
- A data flow can have only one direction of flow. Multiple data flows to and/or from the same process and data store must be shown by separate arrows.
- If the exact same data flows to two separate arrows, it should be represented by a forked arrow.
- Data cannot flow directly back into the process it has just left.
- All data flows must be named using a noun phrase.

DFD symbols:

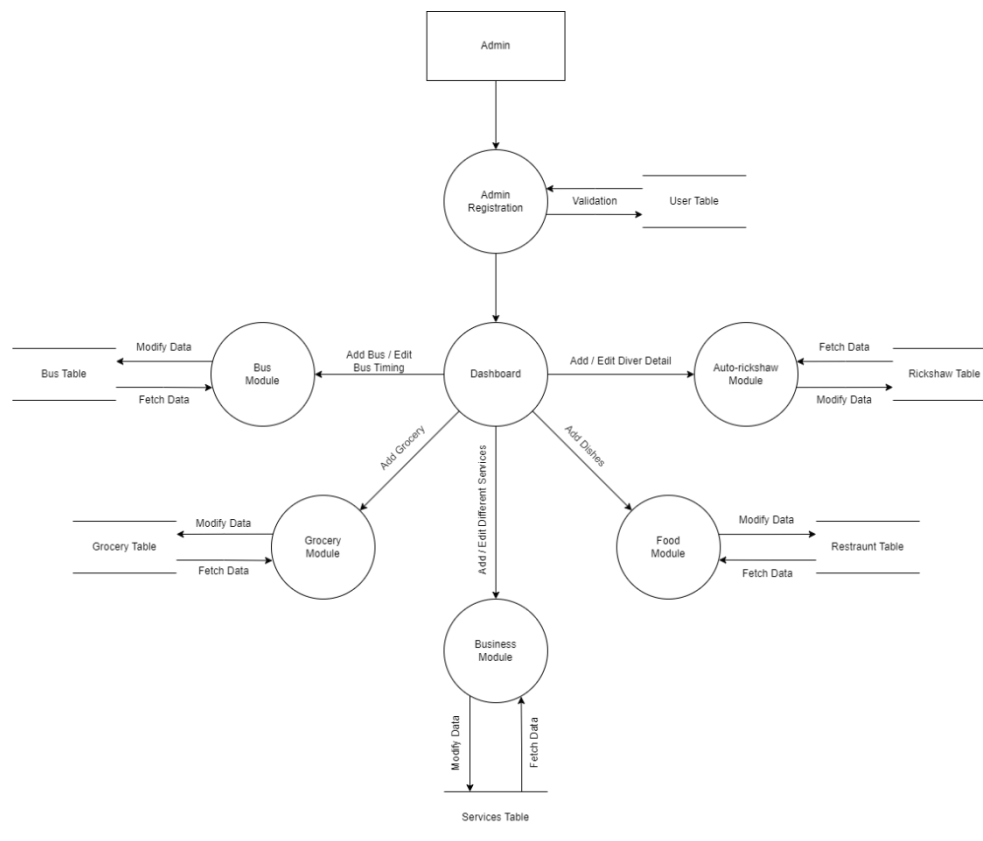
Name	Notation	Description
Process		The bubble represents a process or transformation that is applied to the data which changes in some way. Each bubble is assigned a number.
Data Store		Data stores are repositories of data in the system. They are sometimes also referred to as files.

Data Flow		Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.
Entity		This rectangle is called an entity which represents a producer or a customer of the information.

DFD-lvl-1 User



DFD-lvl-1 Admin



3.5. Description Of Components

1. User Components

- Login / Registration: This module allows user to signup/login to system.
- Bus module: This module displays bus route and its timing to the user.
- Autorickshaw module: This module displays auto details such as diver name and his contact details to user.
- Food module: This module showcases dishes available in local eateries, which user can order.
- Grocery module: This module grocery items to the user which he / she may order.
- Search module: This module displays various local businesses that are in the locality with its contact details.

2. Admin Components

- Login: This module allows user to login to system.

- Bus module: This module facilitates admin to add new bus details such as its route and its timing.
- Autorickshaw module: This module facilitates admin to add new auto details such as its auto driver name and his contact details to user.
- Food module: This module facilitates admin to add new dish details such as its name, price and from which restaurant its available.
- Grocery module: This module facilitates admin to add new grocery item details such as its name and price.
- Business module: This module facilitates admin to add new business details such as its name and contact details.

CHAPTER

-4-

DATABASE DESIGN

4. Database Design

4.1. Introduction

Database design encompasses a range of tasks and processes aimed at optimizing the development, implementation, and maintenance of enterprise data management systems. A well-designed database not only minimizes maintenance costs but also enhances data consistency and cost-effectiveness in terms of disk storage space utilization. It is imperative to employ a meticulous approach when designing a database, taking into account various constraints and determining the interrelationships between elements and the types of data to be stored. By adhering to these principles, a database designer can achieve a comprehensive and efficient database design that aligns with the organization's requirements and objectives.

4.2. Purpose of the Database

- An inadequate database design often leads to unsatisfactory database performance and functionality, underscoring the importance of a well-planned design.
- The implementation of a good database design should facilitate the creation of straightforward and manageable queries, allowing for simplicity and ease of use.
- A sound database design aims to eliminate data redundancies, preventing the duplication of data and ensuring efficiency in storage and retrieval.
- Following the implementation of a strong database design, it is crucial to maintain a high level of data accuracy, thereby promoting the reliability and trustworthiness of the database system.

4.3. Database Identification

MySQL is a widely used relational database management system that shares similarities with Oracle, SQLite, PostgreSQL, and SQL Server. While these four databases operate as client-server engines, MySQL takes a different approach. It can be seamlessly embedded into an application, granting direct access to the robust capabilities of a relational database. This unique feature allows developers to include a MySQL database within their application, harnessing its full potential. MySQL enjoys native support on both Android and iOS platforms, enabling every app to effortlessly create and utilize a MySQL database as per their requirements. This flexibility empowers developers to leverage the benefits of a reliable and scalable database system directly within their applications.

4.4. Schema Information

A schema serves as the foundation for organizing data, providing a visual representation of the interconnections between tables and the underlying business rules that drive the database's purpose. It presents a diagrammatic representation where each database table is outlined with distinct columns and notable characteristics such as primary keys, foreign keys, or constraints. The relationships between tables are depicted through lines connecting the primary key of a parent table with the corresponding foreign keys in child tables. Schema diagrams play a vital role as they compel database developers to translate conceptual ideas onto paper, offering a comprehensive overview of the entire database. Moreover, they streamline the work of future database administrators by providing a clear reference for understanding and managing the database structure.

4.5. Table Definition

In the realm of databases, a table serves as a structured repository for organizing and storing related data. It adopts a tabular format comprising of columns and rows, where each column represents a distinct data attribute, and each row represents a specific data entry. Whether in relational or flat file databases, a table encompasses a collection of data elements that align vertically in columns with identifiable names, while horizontally extending across rows. The unit of intersection between a row and column is referred to as a cell. While a table possesses a predetermined number of columns, it can accommodate an indefinite number of rows. Each row within a table is uniquely identified by one or more values contained within a designated column subset. This collection of columns that ensure row uniqueness is known as the primary key.

1. Tables

- admin

Field Name	Datatype	Constraint Type	Description
id	INT (10)	Primary Key Auto increment	Admin ID
username	VARCHAR (20)	NULL	Admin username
password	VARCHAR (100)	NULL	Hashed password

- auto_rikshaw

Field Name	Datatype	Constraint Type	Description
id	INT (10)	Primary Key	Driver ID

		Auto increment	
autodriver_name	VARCHAR (30)	NULL	Driver name
auto_number	VARCHAR (30)	NULL	Phone number
auto_stand	VARCHAR (50)	NULL	Auto stand name

- bus_table

Field Name	Datatype	Constraint Type	Description
bus_id	INT (10)	Primary Key Auto increment	Bus ID
bus_name	VARCHAR (20)	NULL	Bus name
bus_time	VARCHAR (20)	NULL	Bus timings
destination_name	VARCHAR (30)	NULL	Bus destination

- connect

Field Name	Datatype	Constraint Type	Description
id	INT (10)	Primary Key Auto increment	Business ID
cname	VARCHAR (50)	NULL	Business name
category	TEXT	NULL	Type of business
contact1	TEXT	NULL	Businesses phone number
contact2	TEXT	NULL	Businesses phone number

- foods

Field Name	Datatype	Constraint Type	Description
item_name	VARCHAR (255)	NULL	Item name
item_price	DECIMAL (10,2)	NULL	Item price
hotel_name	VARCHAR (255)	NULL	Hotel name
item_image	VARCHAR (255)	NULL	Item image path

- foodorder

Field Name	Datatype	Constraint Type	Description
user_id	VARCHAR (255)	NULL	User mail ID
item_name	VARCHAR (255)	NULL	Item name
item_price	DECIMAL (10,2)	NULL	Item price
hotel_name	VARCHAR (255)	NULL	Hotel name
item_quantity	INT (11)	NULL	Ordered quantity
status	VARCHAR (50)	NULL	Order status
user_address	VARCHAR (255)	NULL	Users home address
user_phoneNumber	VARCHAR (20)	NULL	Users phone number

- grocery

Field Name	Datatype	Constraint Type	Description
item_name	VARCHAR (255)	NULL	Item name
item_price	DECIMAL (10,2)	NULL	Item price
item_image	VARCHAR (255)	NULL	Item image path

- groceryorder

Field Name	Datatype	Constraint Type	Description
user_id	VARCHAR (255)	NULL	User mail ID
item_name	VARCHAR (255)	NULL	Item name
item_price	DECIMAL (10,2)	NULL	Item price
item_quantity	INT (11)	NULL	Ordered quantity
status	VARCHAR (50)	NULL	Order status
user_address	VARCHAR (255)	NULL	Users home address
user_phoneNumber	VARCHAR (20)	NULL	Users phone number

- user_details

Field Name	Datatype	Constraint Type	Description
user_id	INT (10)	Primary key Auto increment	User ID number
user_name	VARCHAR (30)	NULL	User name
email	VARCHAR (30)	NULL	Email ID
phone_number	VARCHAR (20)	NULL	Users phone number
location	VARCHAR (20)	NULL	Users address

4.6. Physical Design

The primary objective of constructing a physical design for our database is to enhance performance while upholding data integrity through the elimination of redundant data. In the realm of physical design, the conversion of entities into tables, instances into rows, and attributes into columns takes place. This transformation enables efficient data organization and retrieval, facilitating streamlined operations and maximizing the overall effectiveness of the database system. By meticulously crafting the physical design, we ensure optimal performance and maintain the accuracy and consistency of the data stored within the database.

4.7. Data Dictionary

A data dictionary is a collection of metadata that describes the database. It stores information such as the names, schemas, owners, security settings, and creation dates of all the tables in the database. It also stores physical information such as the location and storage method of the tables. Additionally, it stores table constraints such as primary keys, foreign keys, and other restrictions. The data dictionary is essential for the database administration, but it is not directly accessed by the database users.

CHAPTER
-5-
DETAILED
DESIGN

5. Detailed Design

5.1. Introduction

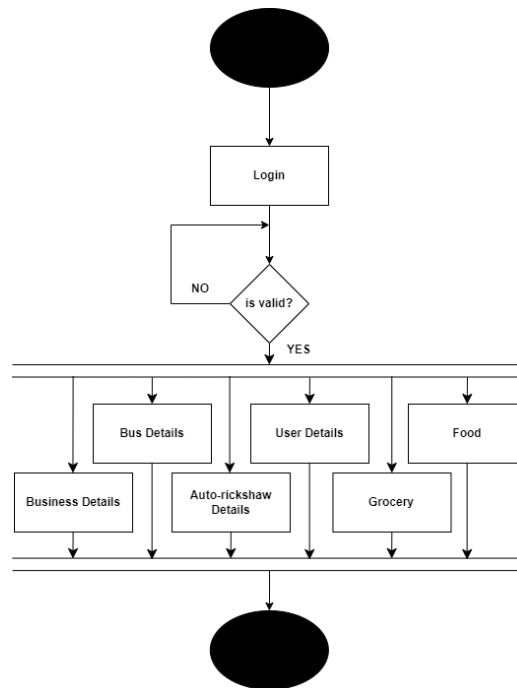
During the detail design phase, a comprehensive specification of the product's geometry, materials, and tolerances is established through the creation of detailed drawings, assembly drawings, and general assembly drawings. This phase also encompasses the specification of bought-out parts, including preferred suppliers and component designations. The ultimate outcome of this phase is a meticulous and precise physical representation of all the parts comprising the product. This detailed design specification serves as a crucial reference, ensuring accuracy and facilitating the seamless manufacturing and assembly of the product.

5.2. Structure of the Software Package

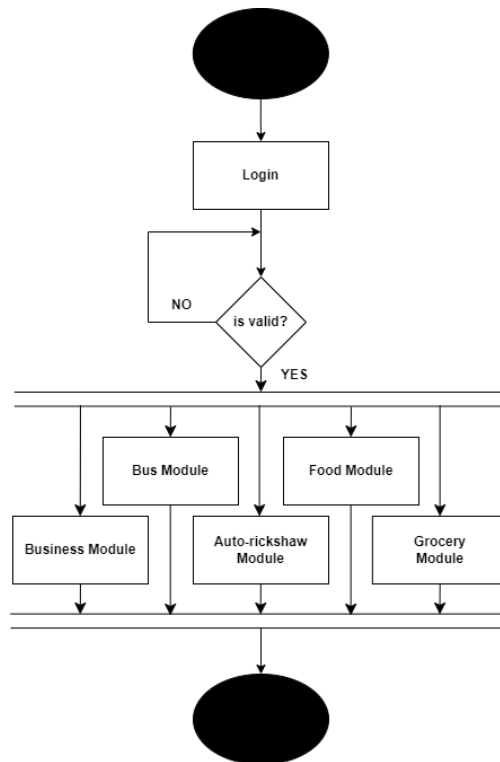
A structure chart serves as a visual representation of the hierarchical arrangement of modules within a system. It offers a systematic breakdown of the entire system into its smallest functional units, providing a detailed description of the functions and sub-functions performed by each module. With the use of structure charts, the system is divided into black boxes, where the users are aware of the system's functionality but lack knowledge of its internal workings. Inputs are provided to these black boxes, and the corresponding outputs are generated accordingly.

The modules at the top level of the structure chart are referred to as low-level modules. The components of the structure chart are read in a top-to-bottom and left-to-right manner, depicting the logical flow of the system. When one module calls another, it treats the called module as a black box, passing the necessary parameters and receiving the desired results in return. This approach allows for a modular and encapsulated system design, enhancing readability, reusability, and maintainability.

- Structure of Admin



- Structure of user



5.3. Modular Decomposition of the System

1. Bus Module

- File I/O interfaces: Graphical User Interface.
- Output: Displays bus routes and bus timing.

2. Auto Rickshaw Module

- File I/O interfaces: Graphical User Interface.
- Output: Displays driver name, his contact detail and auto stand name.

3. Food Module

- Procedural Detail: Displays various dishes available in the town.
- File I/O interfaces: Graphical User Interface.
- Output: Order food.

4. Grocery Module

- Procedural Detail: Displays grocery items.
- File I/O interfaces: Graphical User Interface.
- Output: Order grocery.

5. Business Module

- Procedural Detail: Displays various businesses.
- File I/O interfaces: Graphical User Interface.
- Output: Contact businesses.

CHAPTER

-6-

CODING

6. Introduction

The coding or programming phase aims to transform the system design, created during the design phase, into executable code using a specific programming language. This code enables the computer to perform the computations defined by the design. Throughout the implementation process, it is crucial to prioritize code readability and comprehension over ease of writing. It is essential to ensure that the program is not only easily writable but also easy to interpret and understand, fostering maintainability and facilitating collaboration among developers. By emphasizing code clarity and comprehensibility, the resulting software becomes more manageable, enhancing its long-term viability and effectiveness.

GetPath.dart

```
class GetUrl {  
  final String url = 'http://192.168.0.102:600/';  
  final String imgurl = 'http://192.168.0.102:600/';  
}
```

start.dart

```
import 'package:firebase_core/firebase_core.dart';  
import 'package:flutter/material.dart';  
import 'package:mybedra/firebase_options.dart';  
import 'package:mybedra/pages/auth_page.dart';  
void main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  await Firebase.initializeApp(  
    options: DefaultFirebaseOptions.currentPlatform,);  
  runApp(const MyApp());  
}  
class MyApp extends StatelessWidget {  
  const MyApp({super.key});  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      debugShowCheckedModeBanner: false,
```

```
home: AuthPage(),);}}
```

auth_page.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:mybedra/Home.dart';
import 'package:mybedra/pages/login_or_register.dart';
class AuthPage extends StatelessWidget {
  const AuthPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: StreamBuilder<User?>(
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          if(snapshot.hasData){
            return Home();
          }
          else{return LoginOrRegisterPage();},),);}}
```

login_or_register.dart

```
import 'package:flutter/material.dart';
import 'package:mybedra/pages/login_page.dart';
import "package:mybedra/pages/register_page.dart";
class LoginOrRegisterPage extends StatefulWidget {
  const LoginOrRegisterPage({super.key});
  @override
  State<LoginOrRegisterPage> createState() => _LoginOrRegisterPageState();
class _LoginOrRegisterPageState extends State<LoginOrRegisterPage> {
  bool showLoginPage = true;
  void togglePages(){setState(() {showLoginPage = !showLoginPage;});}
  @override
  Widget build(BuildContext context) {
    if(showLoginPage){
```

```

    return LoginPage(onTap: togglePages);
  } else { return ResgisterPage(onTap: togglePages); } } }

```

login_page.dart

```

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:mybedra/components/my_button.dart';
import 'package:mybedra/components/my_textfield.dart';
import 'package:mybedra/components/square_tile.dart';
import 'package:mybedra/services/auth_service.dart';
import 'forgot_password.dart';

class LoginPage extends StatefulWidget {
  final Function()? onTap;
  const LoginPage({super.key, required this.onTap});
  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  // text editing controllers
  final emailController = TextEditingController();
  final passwordController = TextEditingController();
  // sign user in method
  void signInUserIn() async {
    showDialog(
      context: context,
      builder: (context) {
        return Center(
          child: CircularProgressIndicator(),);));
    try {
      await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: emailController.text,
        password: passwordController.text,);
      Navigator.pop(context);
    } on FirebaseAuthException catch (e) {

```



```

Navigator.pop(context);
showErrorMessage(e.code);}}
void showErrorMessage(String message) {
  showDialog(
    context: context,
    builder: (context) {
      return AlertDialog(
        backgroundColor: Colors.purple,
        title: Text(
          message,));},);}
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.grey[300],
    body: SafeArea(
      child: SingleChildScrollView(
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              const SizedBox(height: 50),

              Image.asset(
                "assets/images/cmlogo.png",
                height: 150,),
              const SizedBox(height: 10),
              // welcome back, you've been missed!
              Text(
                'Welcome back you\'ve been missed!',
                style: GoogleFonts.bebasNeue(
                  fontSize: 20, color: Colors.grey[700])),),
              const SizedBox(height: 25),
              // username textfield
              MyTextField(

```

```

    controller: emailController,
    hintText: 'Email',
    obscureText: false,),
const SizedBox(height: 10),
// password textfield
MyTextField(
    controller: passwordController,
    hintText: 'Password',
    obscureText: true,),
const SizedBox(height: 10),
// forgot password?
Padding(
    padding: const EdgeInsets.symmetric(horizontal: 25.0),
    child: Row(
        mainAxisAlignment: MainAxisAlignment.end,
        children: [
            GestureDetector(
                onTap: () {
                    Navigator.push(context,
                        MaterialPageRoute(builder: (context) {
                            return ForgotPasswordPage(
                                onTap: widget.onTap,));}),
                child: Text(
                    'Forgot Password?',
                    style: GoogleFonts.bebasNeue(
                        fontSize: 20, color: Colors.blue),),),),
const SizedBox(height: 25),
// sign in button
MyButton(
    onTap: signInUserIn,
    text: "Sign in",),
const SizedBox(height: 50),
// or continue with
Padding(

```

```

padding: const EdgeInsets.symmetric(horizontal: 25.0),
child: Row(
  children: [
    Expanded(
      child: Divider(
        thickness: 0.5,
        color: Colors.grey[400],),),
    Padding(
      padding: const EdgeInsets.symmetric(horizontal: 10.0),
      child: Text(
        'Or continue with',
        style: GoogleFonts.bebasNeue(
          fontSize: 20, color: Colors.grey[700],),),),
    Expanded(
      child: Divider(
        thickness: 0.5,
        color: Colors.grey[400],),),),),
const SizedBox(height: 50),
// google + apple sign in buttons
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    // google button
    SquareTile(
      onTap: () => AuthService().signInWithGoogle(),
      imagePath: 'img/google.png',),
    SizedBox(width: 25),
    // apple button
    SquareTile(
      onTap: () {},
      imagePath: 'img/apple.png',),),),
const SizedBox(height: 50),
// not a member? register now
Row(

```

```
mainAxisAlignment: MainAxisAlignment.center,  
children: [  
  Text(  
    'Not a member?',  
    style: GoogleFonts.bebasNeue(  
      fontSize: 20, color: Colors.grey[700]),),  
  const SizedBox(width: 4),  
  GestureDetector(  
    onTap: widget.onTap,  
    child: Text(  
      'Register now',  
      style: GoogleFonts.bebasNeue(  
        fontSize: 20, color: Colors.blue),),),],),),),),);}}
```

register_page.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:mybedra/components/my_button.dart';
import 'package:mybedra/components/my_textfield.dart';
import 'package:mybedra/components/square_tile.dart';
import 'package:mybedra/services/auth_service.dart';
class ResgisterPage extends StatefulWidget {
  final Function()? onTap;
  const ResgisterPage({super.key, required this.onTap});
  @override
  State<ResgisterPage> createState() => _ResgisterPageState();
}
class _ResgisterPageState extends State<ResgisterPage> {
  // text editing controllers
  final emailController = TextEditingController();
  final passwordController = TextEditingController();
  final comfirmPasswordController = TextEditingController();
  @override
```

```

void dispose() {
    // Cancel any active work or subscriptions here
    // before the state is disposed
    super.dispose();}
// sign user in method
void signUserUp() async {
    if (!mounted) return;
    showDialog(
        context: context,
        builder: (context) {
            return Center(
                child: CircularProgressIndicator(),);});
//check if confirm password is same as password
if (passwordController.text == confirmPasswordController.text) {
    try {
        UserCredential userCredential =
            await FirebaseAuth.instance.createUserWithEmailAndPassword(
                email: emailController.text,
                password: passwordController.text,);
        FirebaseFirestore.instance
            .collection("User")
            .doc(userCredential.user!.email)
            .set({
                'username': emailController.text.split('@')[0],
                'address': "",
                'phoneNumber': "",});
        Navigator.pop(context);
    } on FirebaseAuthException catch (e) {
        Navigator.pop(context);
        showErrorMessage(e.code);}
    } else {
        Navigator.pop(context);
        showErrorMessage("Password don't match");}}
void showErrorMessage(String message) {

```

```

showDialog(
  context: context,
  builder: (context) {
    return AlertDialog(
      backgroundColor: Colors.purple,
      title: Text(
        message,)),),);}
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.grey[300],
    body: SafeArea(
      child: SingleChildScrollView(
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              const SizedBox(height: 50),
              Image.asset(
                "assets/images/cmlogo.png",
                height: 150,),
              const SizedBox(height: 10),
              // welcome back, you've been missed!
              Text(
                'Let\'s create a account for you',
                style: GoogleFonts.bebasNeue(
                  fontSize: 20, color: Colors.grey[700]),),
              const SizedBox(height: 25),
              // username textfield
              MyTextField(
                controller: emailController,
                hintText: 'Email',
                obscureText: false,),
              const SizedBox(height: 10),

```

```

// password textfield
MyTextField(
  controller: passwordController,
  hintText: 'Password',
  obscureText: true,),
const SizedBox(height: 10),
//confirm password textfield
MyTextField(
  controller: comfirmPasswordController,
  hintText: 'Confirm Password',
  obscureText: true,),
const SizedBox(height: 10),
// sign up button
MyButton(
  onTap: signUserUp,
  text: "Sign up"),
const SizedBox(height: 22),
// or continue with
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 25.0),
  child: Row(
    children: [
      Expanded(
        child: Divider(
          thickness: 0.5,
          color: Colors.grey[400]),),
      Padding(
        padding: const EdgeInsets.symmetric(horizontal: 10.0),
        child: Text(
          'Or continue with',
          style: GoogleFonts.bebasNeue(
            fontSize: 20, color: Colors.grey[700]),),),
      Expanded(
        child: Divider(

```



```

import 'package:bottom_navy_bar/bottom_navy_bar.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:mybedra/consts.dart';
import 'package:mybedra/pages/profile.dart';
import 'package:mybedra/tempHomePageSliver.dart';
import 'package:flutter/material.dart';
class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();}
class _HomeState extends State<Home> {
  int _cIndex = 0;
  late PageController _pageController;
  @override
  void initState() {
    super.initState();
    _pageController = PageController();}
  @override
  void dispose() {
    _pageController.dispose();
    super.dispose();}
  void _incrementTab(index) {
    setState(() {
      _cIndex = index;
      _pageController.animateToPage(index,
        duration: Duration(milliseconds: 100), curve: Curves.ease);});}
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: null,
      bottomNavigationBar: BottomNavyBar(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        selectedIndex: _cIndex,
        showElevation: true,
        items: [

```

```

BottomNavyBarItem(
  icon: Icon(Icons.home),
  activeColor: Colors.green,
  inactiveColor: kgreyDark,
  title: Text(
    "Home",
    style: GoogleFonts.bebasNeue(fontSize: 20,color: Colors.white),),),
BottomNavyBarItem(
  icon: Icon(Icons.person),
  title: Text(
    "Profile",
    style: GoogleFonts.bebasNeue(fontSize: 20,color: Colors.white),),
  inactiveColor: kgreyDark,
  activeColor: Colors.orange,)],
onItemSelected: (index) {
  _incrementTab(index);}),
body: SizedBox.expand(
  child: PageView(
    controller: _pageController,
    onPageChanged: (index) {
      setState(() => _cIndex = index);},
    children: <Widget>[
      tempHomePageSliver(),
      ProfilePage(),],),),);}}

```

tempHomePageSilver.dart

```

import 'dart:async';
import 'package:google_fonts/google_fonts.dart';
import 'package:mybedra/Grocery.dart';
import 'package:mybedra/BusSelectPage.dart';
import 'package:mybedra/autoselect.dart';
import 'package:mybedra/food.dart';
import 'package:mybedra/categories.dart';
import 'package:mybedra/themes.dart';

```

```

import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'Widgets/carousalSlider.dart';
class tempHomePageSliver extends StatefulWidget {
  tempHomePageSliver({Key? key}) : super(key: key);
  @override
  _HomePageState createState() => _HomePageState();}
class _HomePageState extends State<tempHomePageSliver> {
  int activePage = 1;
  int _currentPage = 1;
  late Timer _timer;
  @override
  void initState() {
    super.initState();
    _timer = Timer.periodic(Duration(seconds: 3), (Timer timer) {
      if (_currentPage < 3) {
        _currentPage++;
      } else {
        _currentPage = 0;});});}
  @override
  void dispose() {
    super.dispose();
    _timer.cancel();}
  final _imageUrls = [
    "https://cdn.pixabay.com/photo/2015/04/23/22/00/tree-736885__480.jpg",
    "https://cdn.pixabay.com/photo/2021/08/25/20/42/field-6574455__340.jpg",
    "https://cdn.pixabay.com/photo/2021/08/25/20/42/field-6574455__340.jpg"];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: CustomScrollView(
        slivers: [
          SliverAppBar(
            toolbarHeight: 100,

```

```

leadingWidth: 0,
floating: true,
pinned: true,
backgroundColor: Colors.white,
snap: false,
centerTitle: false,
title: Column(
  mainAxisAlignment: MainAxisAlignment.start,
  children: [
    Container(
      alignment: Alignment.centerLeft,
      margin: EdgeInsets.fromLTRB(0, 0, 0, 0),
      child: Text(
        "Welcome to myBedra",
        style: GoogleFonts.bebasNeue(
          fontSize: 30, color: Colors.black54),
        textAlign: TextAlign.left,)),),
    actions: [
      Padding(
        padding: const EdgeInsets.all(8.0),
        child: Image.asset(
          'assets/images/cmlogo.png',
          width: 80,
          height: 80,)),),
    // Other Sliver Widgets
  ],
),
SliverList(
  delegate: SliverChildListDelegate([
    Column(
      children: [
        Container(
          margin: EdgeInsets.fromLTRB(0, 12, 0, 12),
          child: Stack(
            alignment: AlignmentDirectional.bottomCenter,
            children: [

```

```

        MyImageCarousel(imageUrls: _imageUrls),]),),
    Container(
      margin: EdgeInsets.all(5),
      alignment: Alignment.centerLeft,
      child: Text("Search by Categories", style: GoogleFonts.bebasNeue(fontSize:
25,color: Colors.grey),)),
    Container(
      child: Categories(),
      alignment: Alignment.centerLeft,),
    Container(
      margin: EdgeInsets.all(5),
      alignment: Alignment.centerLeft,
      child: Text("Digital Guide", style: GoogleFonts.bebasNeue(fontSize: 35,color:
Colors.grey),)),
    Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Expanded(child: _buildProgrammCardBus(context)),
        Expanded(child: _buildProgrammCardAuto(context)),],),
    Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Expanded(child: _buildProgrammCardFood(context)),
        Expanded(child: _buildProgrammCardGross(context)),],),],),),),);}}
Widget _buildProgrammCardAuto(context) {
  return Container(
    margin: EdgeInsets.fromLTRB(4, 0, 2, 0),
    height: 120,
    child: Card(
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(15.0)),
      color: yellowColor,
      child: InkWell(
        splashColor: Colors.white.withAlpha(30),

```



```

margin: EdgeInsets.fromLTRB(4, 0, 2, 0),
height: 120,
child: Card(
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(15.0)),
  color: blueColorDark,
  child: InkWell(
    splashColor: Colors.white.withAlpha(30),
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => BusSelectPage(),),),},
  child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: Column(
      children: [
        Expanded(
          child: Row(
            children: [
              Expanded(
                child: Padding(
                  padding: const EdgeInsets.fromLTRB(8, 0, 0, 0),
                  child: Container(
                    alignment: Alignment.topLeft,
                    height: 50,
                    width: 50,
                    child: Image.asset(
                      'assets/images/bussidelogo.png',
                      color: Colors.white,
                      height: 70,)),),),
              Icon(
                CupertinoIcons.forward,
                color: Colors.white,)),),

```



```

String searchItem = "";
void openCartView(BuildContext context) {
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => ImageSearch(searchItem: searchItem),),);}
@override
Widget build(BuildContext context) {
  return SingleChildScrollView(
    scrollDirection: Axis.horizontal,
    child: Container(
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        children: [
          Container(
            alignment: Alignment.centerLeft,
            margin: EdgeInsets.all(8),
            child: ClipRRect(
              borderRadius: BorderRadius.circular(8.0),
              child: Column(
                children: [
                  Padding(
                    padding: const EdgeInsets.fromLTRB(4, 4, 4, 0),
                    child: GestureDetector(
                      onTap: () {
                        searchItem = "Rest";
                        openCartView(context);},
                      child: Image.asset(
                        'assets/images/dinner.png',
                        width: 50.0,
                        height: 50.0,
                        fit: BoxFit.fill,)),),
                  SizedBox(
                    height: 4,),

```

```

Container(
  alignment: Alignment.center,
  width: 54,
  child: new Text("Restaurants",
    style: new TextStyle(
      overflow: TextOverflow.ellipsis,
      fontSize: 8,
      color: Colors.white,
      fontWeight: FontWeight.w600)),
  decoration: new BoxDecoration(
    borderRadius:
      new BorderRadius.all(new Radius.circular(10.0)),
    color: Colors.black54),
  padding: new EdgeInsets.fromLTRB(8, 4, 8, 4),),),),}

```

imagesearch.dart

```

import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:http/http.dart' as http;
import 'package:url_launcher/url_launcher.dart';
import 'GetPath.dart';
class ImageSearch extends StatefulWidget {
  final String searchItem;
  const ImageSearch({
    Key? key,
    required this.searchItem,
  }) : super(key: key);
  @override
  _ImageSearchState createState() => _ImageSearchState();
class _ImageSearchState extends State<ImageSearch> {
  List<Map<String, dynamic>> searchedItems = [];
  @override
  void initState() {

```

```

super.initState();
searchItems(widget.searchItem);}
Future<void> searchItems(String searchTerm) async {
  GetUrl serverPath = GetUrl();
  final url = serverPath.url+'categories_search?searchString=$searchTerm';
  final response = await http.get(Uri.parse(url), headers: {'Content-Type': 'application/json'});
  if (response.statusCode == 200) {
    final data = json.decode(response.body) as List;
    setState(() {
      searchedItems = data.map((item) => Map<String, dynamic>.from(item)).toList();});
  } else {
    print('Failed to fetch category data. Status code: ${response.statusCode}');}}
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Search Results',style: GoogleFonts.bebasNeue(fontSize: 20),),),
    body: ListView.builder(
      itemCount: searchedItems.length,
      itemBuilder: (context, index) {
        final result = searchedItems[index];
        return ListTile(
          title: Text(result['cname'] ?? "",style: GoogleFonts.bebasNeue(fontSize: 20),),
          subtitle: Text(result['contact1'] ?? result['contact2'] ?? "",style:
GoogleFonts.bebasNeue(fontSize: 20),),
          onTap: () {
            String phoneNumber = result['contact1'] ?? result['contact2'] ?? "";
            launch('tel:$phoneNumber');},),),)}

```

BusSelectPage.dart

```

import 'dart:async';
import 'dart:convert';
import 'package:google_fonts/google_fonts.dart';
import 'package:mybedra/GetPath.dart';

```

```

import 'package:mybedra/Models/BusDestinationModel.dart';
import 'package:mybedra/TextStyles.dart';
import 'package:mybedra/consts.dart';
import 'package:mybedra/dashBoard.dart';
import 'package:mybedra/themes.dart';
import 'package:flutter/cupertino.dart';
import 'package:mybedra/getBuses.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:intl/intl.dart';
import 'package:velocity_x/velocity_x.dart';
class BusSelectPage extends StatefulWidget {
  BusSelectPage({Key? key}) : super(key: key);
  @override
  State<BusSelectPage> createState() => _BusSelectPageState();}
class _BusSelectPageState extends State<BusSelectPage> {
  late Timer timer;
  bool _isSelected = false;
  bool _isLogin = false;
  String mangloreTime = "";
  String karkalaime = "";
  String bcRoadTime = "";
  String belthangadyTime = "";
  String MangaluruNextBusName = "";
  String KarkalaNextBusName = "";
  String BCRoadNextBusName = "";
  String BelthangadyNextBusName = "";
  List<String> dropdownValues = [];
  late String _dropDownValue = "Select location";
  Future<void> fetchData() async {
    // Make an HTTP GET request to your API endpoint
    GetUrl serverPath= GetUrl();
    var response =
      await http.get(Uri.parse(serverPath.url+'api/destinations/bus'));

```

```

if (response.statusCode == 200) {
  final jsonData = json.decode(response.body);
  final List<dynamic> destinations = jsonData['destinations'];
  setState() {
    dropdownValues = destinations.cast<String>();
  });
} else {
  // Handle the error case
  print('Request failed with status: ${response.statusCode}');}}

@override
void initState() {
  super.initState();
  fetchData();
  loadDatabuses('Mangaluru');
  loadDatabuses('Karkala');
  loadDatabuses('BC Road');
  loadDatabuses('Belthangady');
  timer = Timer.periodic(Duration(seconds: 15), (Timer t) => SetDefault());}
SetDefault() {
  if (mangloreTime == "") mangloreTime = 'No Buses Available';
  if (karkalaime == "") karkalaime = 'No Buses Available';
  if (bcRoadTime == "") bcRoadTime = 'No Buses Available';
  if (mangloreTime == "") mangloreTime = 'No Buses Available';}

@override
Widget build(BuildContext context) {
  return Scaffold(
    floatingActionButtonLocation: FloatingActionButtonLocation.endTop,
    appBar: AppBar(
      backgroundColor: blueColorDark,
      leading: BackButton(color: Colors.white),
      title: Text(
        'Buses from Moodbidri',
        style: GoogleFonts.bebasNeue(fontSize: 30, color: Colors.white),),),
    body: Container(

```

```

alignment: Alignment.center,
child: SingleChildScrollView(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      SizedBox(
        height: 70,),
      Container(
        alignment: Alignment.center,
        child: Stack(
          children: [
            Image.asset(
              'assets/images/backk.png',
              color: Colors.blueGrey,
              height: 100,),
            Container(
              margin: EdgeInsets.fromLTRB(30, 0, 10, 0),
              child: Image.asset(
                'assets/images/blue-bus.png',
                height: 95,),)],),
      SizedBox(
        height: 30,),
      Text(
        'Select your route',
        style:
          GoogleFonts.bebasNeue(fontSize: 18, color: Colors.blueGrey),),
      Padding(
        padding: EdgeInsets.fromLTRB(36, 12, 36, 6),
        child: Center(
          child: DropdownButton<String>(
            borderRadius: BorderRadius.circular(12)),
            hint: _dropDownValue == null
              ? Text('Dropdown')
              : Text(

```



```

        _dropDownValue,
        style: GoogleFonts.bebasNeue(
            fontSize: 18, color: Colors.blueGrey),),
    isExpanded: true,
    iconSize: 30.0,
    style: TextStyle(color: Colors.blueGrey),
    items: dropdownValues
        .map<DropdownMenuItem<String>>((String value) {
    return DropdownMenuItem<String>(
        value: value,
        child: Text(
            value,
            style: GoogleFonts.bebasNeue(
                fontSize: 18, color: Colors.blueGrey),),),
    }).toList(),
    onChanged: (String? value) {
        // This is called when the user selects an item.
        setState(() {
            _dropDownValue = value!;
            _isSelected = true;});},),),),
    SizedBox(
        height: 12,),
    Container(
        margin: EdgeInsets.fromLTRB(24, 0, 24, 12),
        height: 45,
        width: double.infinity,
        child: _isLogin
            ? Center(
                child: CircularProgressIndicator(),)
            : ElevatedButton(
                style: ElevatedButton.styleFrom(
                    primary: _isSelected ? blueColorDark : blueColor,
                    onSurface: blueColorDark,
                    elevation: 0,

```

```

        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(8)),),
        onPressed: _isSelected? () {
          setState(() {
            _isSelected = true;
            _isLogin = true;});
          Future.delayed(Duration(milliseconds: 100), () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => Dashboard(
                  BusRoute: _dropDownValue,)),);
            setState(() {
              _isLogin = false;});});}
        : null,
        child: Text(
          'Serach Buses',
          style: GoogleFonts.bebasNeue(
            fontSize: 18, color: Colors.white)),),),
        Container(
          margin: EdgeInsets.all(12),
          alignment: Alignment.centerLeft,
          child: NormalText("Quick Access", kgreyDark, 28)),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            Expanded(
              child: _buildBusCard(context, 'Mangaluru', mangloreTime,
                MangaluruNextBusName)),
            Expanded(
              child: _buildBusCard(
                context, 'Karkala', karkalaime, KarkalaNextBusName)),],),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,

```

```

        children: [
          Expanded(
            child: _buildBusCard(
              context, 'BC Road', bcRoadTime, BCRoadNextBusName)),
          Expanded(
            child: _buildBusCard(context, 'Belthangady',
              belthangadyTime, BelthangadyNextBusName)),],),),),),),);}
Widget timeWidget(context, String Timing, String NextBusName) {
  return Column(
    children: [
      Container(
        alignment: Alignment.topLeft,
        child: Text(
          NextBusName,
          overflow: TextOverflow.ellipsis,
          textAlign: TextAlign.left,
          maxLines: 1,
          style: GoogleFonts.bebasNeue(fontSize: 18, color: Colors.white),),),
      Container(
        alignment: Alignment.topLeft,
        child: Text(
          Timing,
          overflow: TextOverflow.ellipsis,
          maxLines: 1,
          style: GoogleFonts.bebasNeue(fontSize: 18, color: Colors.white),),),]);}
Widget _buildBusCard(
  context, String DestinationName, String NextBusTime, String NextBusName) {
  return Container(
    margin: EdgeInsets.fromLTRB(4, 0, 2, 0),
    height: 130,
    child: Card(
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(15.0)),
      color: blueColorDark,

```

```

child: InkWell(
  splashColor: Colors.white.withAlpha(30),
  onTap: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => Dashboard(BusRoute: DestinationName),),),},
child: Padding(
  padding: const EdgeInsets.all(8.0),
  child: Column(
    children: [
      Expanded(
        child: Row(
          children: [
            Expanded(
              child: Padding(
                padding: const EdgeInsets.only(
                  bottom: 8.0, left: 8.0, right: 8.0),
                child: Container(
                  alignment: Alignment.topLeft,
                  height: 80,
                  child: Column(
                    children: [
                      if (NextBusTime != "")
                        timeWidget(context, NextBusTime, NextBusName)
                      else
                        CircularProgressIndicator(
                          color: Colors.white,
                        ).centered().expand(),],),),),
            Icon(
              CupertinoIcons.forward,
              color: Colors.white,)],),),
      Padding(
        padding: const EdgeInsets.all(8.0),

```

```

        child: Container(
          alignment: Alignment.bottomLeft,
          height: 30,
          child: Text(
            DestinationName,
            style: GoogleFonts.bebasNeue(
              fontSize: 24, color: Colors.white),),),),], ),),),),);}
loadDatabuses(String BusRoute) async {
  GetUrl serverPath = GetUrl();
  final busJson =
    await http.get(Uri.parse(serverPath.url + 'bus/' + BusRoute));
  print(busJson);
  var busdecodedData = jsonDecode(busJson.body);
  var busproductsData = busdecodedData;
  busModel.buslist = List.from(busproductsData)
    .map<Buses>((buslist) => Buses.fromMap(buslist))
    .toList();
  var now = DateTime.now();
  var formatterTime = DateFormat('HHmm');
  String actualTime = formatterTime.format(now);
  String strNextBus = "";
  String strNxtbusName = "";
  int nextBustime;
  for (int i = 0; i < busModel.buslist.length; i++) {
    busModel.buslist[i];
    if ((busModel.buslist[i].bus_time).toString().length == 3) {
      nextBustime = int.parse('0' + busModel.buslist[i].bus_time);
    } else {
      nextBustime = int.parse(busModel.buslist[i].bus_time);}
    int currtime = int.parse(actualTime);
    if (nextBustime > currtime) {
      strNextBus = busModel.buslist[i].bus_time;
      strNxtbusName = busModel.buslist[i].bus_name;
      break;}}

```

```

if (BusRoute == 'Mangaluru') {
  mangloreTime = utcTo12HourFormat(strNextBus);
  MangaluruNextBusName = strNxtbusName;
} else if (BusRoute == 'Karkala') {
  karkalaime = utcTo12HourFormat(strNextBus);
  KarkalaNextBusName = strNxtbusName;
} else if (BusRoute == 'BC Road') {
  bcRoadTime = utcTo12HourFormat(strNextBus);
  BCRoadNextBusName = strNxtbusName;
} else if (BusRoute == 'Belthangady') {
  belthangadyTime = utcTo12HourFormat(strNextBus);
  BelthangadyNextBusName = strNxtbusName;}
setState(() {});}

String utcTo12HourFormat(String bigTime) {
  if (bigTime.length == 3) {
    bigTime = '0' + bigTime;}
  if (bigTime != "") {
    bigTime = bigTime.substring(0, 2) + ':' + bigTime.substring(2, 4);
    DateTime tempDate = DateFormat("HH:mm").parse(bigTime);
    final DateFormat formatter = DateFormat('h:mm a');
    final String formatted = formatter.format(tempDate);
    print(formatted);
    return formatted;
  } else {
    return 'Bus not available';}}}

```

autoselect.dart

```

import 'dart:convert';
import 'package:google_fonts/google_fonts.dart';
import 'package:mybedra/AutoListMainPage.dart';
import 'package:mybedra/GetPath.dart';
import 'package:http/http.dart' as http;
import 'package:mybedra/themes.dart';
import 'package:flutter/cupertino.dart';

```

```

import 'package:flutter/material.dart';
class AutoSelect extends StatefulWidget {
  AutoSelect({Key? key}) : super(key: key);
  @override
  State<AutoSelect> createState() => _AutoSelectState();}
class _AutoSelectState extends State<AutoSelect> {
  @override
  void initState() {
    super.initState();
    fetchData();}
  bool _isSelected = false;
  bool _isLogin = false;
  List<String> dropdownValues = [];
  Future<void> fetchData() async {
    GetUrl serverPath=GetUrl();
    // Make an HTTP GET request to your API endpoint
    var response = await http
      .get(Uri.parse(serverPath.url+'api/destinations/auto'));
    if (response.statusCode == 200) {
      final jsonData = json.decode(response.body);
      final List<dynamic> destinations = jsonData['destinations'];

      setState() {
        dropdownValues = destinations.cast<String>();});
    } else {
      // Handle the error case
      print('Request failed with status: ${response.statusCode}');}}
  late String _dropDownValue = "Select location";
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: yellowColor,
        leading: BackButton(color: Colors.white),

```

```

title: Text(
  ' Select your location',
  style: GoogleFonts.bebasNeue(fontSize: 20, color: Colors.white),),
centerTitle: true,)),
body: Container(
  alignment: Alignment.center,
  child: SingleChildScrollView(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        SizedBox(
          height: 30,),
        Container(
          alignment: Alignment.center,
          child: Stack(
            alignment: Alignment(-0.8, 0.2),
            children: [
              Image.asset(
                'assets/images/backk.png',
                color: Colors.green,
                height: 300,),
              Image.asset(
                'assets/images/y-auto.png',
                height: 140,)],)),
        SizedBox(
          height: 30,),
        Text(
          'Auto stand location?',
          style:
            GoogleFonts.bebasNeue(fontSize: 20, color: Colors.blueGrey),),
        Padding(
          padding: EdgeInsets.fromLTRB(36, 12, 36, 6),
          child: Center(
            child: DropdownButton<String>(

```



```

borderRadius: BorderRadius.all(Radius.circular(12)),
hint: _dropDownValue == null
  ? Text('Dropdown')
  : Text(
    _dropDownValue,
    style: GoogleFonts.bebasNeue(
      fontSize: 18, color: Colors.blueGrey),),
isExpanded: true,
iconSize: 30.0,
style: TextStyle(color: Colors.blueGrey),
items: dropdownValues
  .map<DropdownMenuItem<String>>((String value) {
return DropdownMenuItem<String>(
  value: value,
  child: Text(
    value,
    style: GoogleFonts.bebasNeue(
      fontSize: 18, color: Colors.blueGrey),),),
}).toList(),
onChanged: (String? value) {
  // This is called when the user selects an item.
  setState(() {
    _dropDownValue = value!;
    _isSelected = true;});}, ),),
SizedBox(
  height: 12,),
Container(
  margin: EdgeInsets.fromLTRB(24, 0, 24, 12),
  height: 45,
  width: double.infinity,
  child: _isLogin
    ? Center(
      child: CircularProgressIndicator(),)
    : ElevatedButton(

```

```

style: ElevatedButton.styleFrom(
    primary: _isSelected ? yellowColor : yellowColor,
    onSurface: yellowColor,
    elevation: 0,
    shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(8)),),
onPressed: _isSelected? () {
    setState() {
        _isSelected = true;

        _isLogin = true;});
Future.delayed(Duration(milliseconds: 100), () {
    setState() {
        Navigator.push(
            context,
            MaterialPageRoute(
                builder: (context) =>
                    AutoListMainPage(
                        StandName: _dropDownValue,));),);
        _isLogin = false;});});}
: null,
child: Text(
    'Seach Auto',
    style: GoogleFonts.bebasNeue(
        fontSize: 18, color: Colors.white),),),),),),),),);}}

```

food.dart

```
import 'dart:convert';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:mybedra/pages/profile.dart';
import 'package:http/http.dart' as http;
import 'package:collection/collection.dart';
```

```

import 'package:firebase_auth/firebase_auth.dart';
import 'package:mybedra/themes.dart';
import 'package:mybedra/GetPath.dart';
class FoodScreen extends StatefulWidget {
  @override
  FoodScreenState createState() => FoodScreenState();}
class FoodScreenState extends State<FoodScreen> {
  List<Map<String, dynamic>> groceryItems = [];
  List<Map<String, dynamic>> cartItems = [];
  TextEditingController searchController = TextEditingController();
  bool isSearching = false;
  List<Map<String, dynamic>> searchedItems = [];
  @override
  void initState() {
    super.initState();
    fetchData();}
  Future<void> fetchData() async {
    GetUrl serverPath = GetUrl();
    final response =
      await http.get(Uri.parse(serverPath.url +'food_fetch'));
    if (response.statusCode == 200) {
      final data = json.decode(response.body) as List;
      setState() {
        groceryItems =
          data.map((item) => Map<String, dynamic>.from(item)).toList();});} }
  void signUserOut() {
    FirebaseAuth.instance.signOut();}
  void openProfileView() {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => ProfilePage(),),);}
  void openCartView(BuildContext context) {
    Navigator.push(

```

```

    context,
    MaterialPageRoute(
      builder: (context) => CartScreen(cartItems: cartItems),),);}
void searchItems(String searchTerm) async {
  setState(() {
    isSearching = true;});
  GetUrl serverPath = GetUrl();
  final url = serverPath.url+'food_search';
  final response = await http.post(
    Uri.parse(url),
    headers: {'Content-Type': 'application/json'},
    body: json.encode({'searchTerm': searchTerm}),);
  if (response.statusCode == 200) {
    final data = json.decode(response.body) as List;
    setState(() {
      searchedItems =
        data.map((item) => Map<String, dynamic>.from(item)).toList();});
  } else {
    // Handle the API error
    // ...}}
void _performSearch(String searchTerm) {
  if (searchTerm.isNotEmpty) {
    searchItems(searchTerm);
    setState(() {
      isSearching = true; // Set isSearching to true when performing search});
  } else {
    setState(() {
      searchedItems = List<Map<String, dynamic>>.from(groceryItems);
      isSearching = false; // Set isSearching to false when not searching});}
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.grey[100],
    appBar: AppBar(

```



```

        quantities[index] -= 0.25;}}});}

void placeOrder() async {
    double totalPrice = 0.0;
    List<Map<String, dynamic>> orderedItems = [];
    // Fetch user data from Firebase Firestore
    DocumentSnapshot userData = await FirebaseFirestore.instance
        .collection("User")
        .doc(user.email)
        .get();
    for (int i = 0; i < widget.cartItems.length; i++) {
        final item = widget.cartItems[i];
        final quantity = quantities[i];
        final price = item['price']; // Parse the price as an integer
        totalPrice += price * quantity;
        if (userData.exists) {
            if (userData['address'] != "" && userData['phoneNumber'] != "") {
                Map<String, dynamic> orderedItem = {
                    'user_name': user.email!,
                    'item_name': item['name'],
                    'hotel_name': item['hotel'],
                    'item_price': price,
                    'quantity': quantity,
                    'address': userData['address'],
                    'phoneNumber': userData['phoneNumber'],};
                orderedItems.add(orderedItem);
            } else {
                showDialog(
                    context: context,
                    builder: (BuildContext context) {
                        return Theme(
                            data: Theme.of(context).copyWith(
                                dialogTheme: DialogTheme(
                                    elevation: 0, // Set the elevation to 0 to remove the shadow),),
                            child: AlertDialog(

```



```

child: TextField(
  controller: widget.controller,
  style: TextStyle(color: Colors.blueGrey),
  decoration: InputDecoration(
    hintText: 'Search...',
    hintStyle: TextStyle(color: Colors.blueGrey),
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.blueGrey),),
    enabledBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.blueGrey),),
    focusedBorder: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
      borderSide: BorderSide(color: Colors.blueGrey),),),),
  IconButton(
    icon: Icon(Icons.search),
    onPressed: _handleSearch,
    color: Colors.blueGrey,
    disabledColor: Colors.black, // Add this line),], ),);}

```

grocery.dart

```

import 'dart:convert';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:mybedra/pages/profile.dart';
import 'package:http/http.dart' as http;
import 'package:collection/collection.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:mybedra/themes.dart';
import 'GetPath.dart';
class GroceryScreen extends StatefulWidget {
  @override

```

```

_GroceryScreenState createState() => _GroceryScreenState();}
class _GroceryScreenState extends State<GroceryScreen> {
  List<Map<String, dynamic>> groceryItems = [];
  List<Map<String, dynamic>> cartItems = [];
  TextEditingController searchController = TextEditingController();
  bool isSearching = false;
  List<Map<String, dynamic>> searchedItems = [];
  @override
  void initState() {
    super.initState();
    fetchData();}
  Future<void> fetchData() async {
    GetUrl serverPath = GetUrl();
    final response =
      await http.get(Uri.parse(serverPath.url +'grocery_fetch'));
    if (response.statusCode == 200) {
      final data = json.decode(response.body) as List;
      setState() {
        groceryItems =
          data.map((item) => Map<String, dynamic>.from(item)).toList();});}
  void signUserOut() {
    FirebaseAuth.instance.signOut();}
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      appBar: AppBar(
        elevation: 0,
        backgroundColor: GreenColor,
        title: Text(
          'Grocery',
          style: GoogleFonts.bebasNeue(fontSize: 30),),
        actions: [
          IconButton(

```

```

        icon: Icon(Icons.shopping_cart),
        onPressed: () {
          openCartView(context);},),]),
body: Column(
  children: [
    CustomSearchBar(
      controller: searchController,
      isSearching: isSearching,
      onSearch: _performSearch,),
    Expanded(
      child: Container(
        child: Padding(
          padding: const EdgeInsets.all(8.0),
          child: GridView.builder(
            gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
              crossAxisCount: 2,
              crossAxisSpacing: 12.0,
              mainAxisSpacing: 12.0,
              mainAxisExtent: 250,),
            itemCount:
              isSearching ? searchedItems.length : groceryItems.length,
            itemBuilder: (ctx, index) {
              final item = isSearching
                ? searchedItems[index]
                : groceryItems[index];
              final String imagePath = item['item_image'];
              return Card(
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(16),
                  side: BorderSide(color: GreenColor)),
                child: Column(
                  children: [
                    Padding(
                      padding: const EdgeInsets.only(

```



```

class _CartScreenState extends State<CartScreen> {
  List<double> quantities = [];
  @override
  void initState() {
    super.initState();
    quantities = List<double>.filled(widget.cartItems.length, 1.0);}
  final user = FirebaseAuth.instance.currentUser!;
  void incrementQuantity(int index) {
    setState() {
      quantities[index] += 0.25;});}
  void decrementQuantity(int index) {
    setState() {
      if (quantities[index] > 0.25) {
        quantities[index] -= 0.25;});});}
  void placeOrder() async {
    double totalPrice = 0.0;
    List<Map<String, dynamic>> orderedItems = [];
    // Fetch user data from Firebase Firestore
    DocumentSnapshot userData = await FirebaseFirestore.instance
      .collection("User")
      .doc(user.email)
      .get();
    for (int i = 0; i < widget.cartItems.length; i++) {
      final item = widget.cartItems[i];
      final quantity = quantities[i];
      final price = item['price']; // Parse the price as an integer
      totalPrice += price * quantity;
      if (userData.exists) {
        if (userData['address'] != "" && userData['phoneNumber'] != "") {
          Map<String, dynamic> orderedItem = {
            'user_name': user.email!,
            'item_name': item['name'],
            'item_price': price,
            'quantity': quantity,

```

```

        'address': userData['address'],
        'phoneNumber': userData['phoneNumber'],});
orderedItems.add(orderedItem);
} else {
showDialog(
  context: context,
  builder: (BuildContext context) {
    return Theme(
      data: Theme.of(context).copyWith(
        dialogTheme: DialogTheme(
          elevation: 0, // Set the elevation to 0 to remove the shadow),),
      child: AlertDialog(
        title: Text(
          "Incomplete Profile Information",
          style: GoogleFonts.bebasNeue(
            fontSize: 20, color: Colors.redAccent),),
        content: Text(
          'We noticed that some required information is missing from your profile. Please
provide your phone number and address to complete your profile. This information is essential
for us to provide you with the best possible service and ensure smooth communication. Thank
you for your cooperation!',
          style: GoogleFonts.bebasNeue(fontSize: 20),),
        actions: [
          ElevatedButton(
            onPressed: () {
              Navigator.pop(context);},
            child: Text(
              'OK',
              style: GoogleFonts.bebasNeue(fontSize: 20),),),),),),),),);break;}}
if (userData['address'] != "" && userData['phoneNumber'] != "") {
showDialog(
  context: context,
  builder: (BuildContext context) {
    return Theme(

```



```

        style: GoogleFonts.bebasNeue(fontSize: 30),),),
body: Padding(
  padding: const EdgeInsets.all(8.0),
  child: GridView.builder(
    gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: 1,
      crossAxisSpacing: 12.0,
      mainAxisSpacing: 12.0,
      mainAxisExtent: 120,),
    itemCount: widget.cartItems.length,
    itemBuilder: (ctx, index) {
      final item = widget.cartItems[index];
      final quantity = quantities[index];
      return Card(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12.0)),
        child: Padding(
          padding: const EdgeInsets.only(top: 8.0),
          child: Column(
            children: [
              Text(
                item['name'],
                style: GoogleFonts.bebasNeue(
                  fontSize: 20, color: Colors.black),),
              Text(
                '₹${item['price']}',
                style: GoogleFonts.bebasNeue(
                  fontSize: 20, color: Colors.black),),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  IconButton(
                    onPressed: () {
                      decrementQuantity(index);},

```



```

const item_list_final = [];
app.use(cors());
const port = process.env.PORT || 600;
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json()); // Remove
//app.use(express.json()); // New
const pool = mysql.createPool({
  connectionLimit: 10000,
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'connectm' //'nodejs_beers'})
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, 'uploads/'),
  filename: function (req, file, cb) {
    cb(null, Date.now() + '-' + file.originalname)}});
const fileFilter = function (req, file, cb) {
  const allowedTypes = ['image/jpeg', 'image/png'];
  if (!allowedTypes.includes(file.mimetype)) {
    const error = new Error('Wrong file type');
    error.code = 'LIMIT_FILE_TYPES';
    return cb(error, false);}
  cb(null, true);});
const upload = multer({
  storage: storage,
  fileFilter: fileFilter,
  limits: {
    fileSize: 1024 * 1024 * 5 // 5 MB (max file size)}})
app.post('/upload', upload.single('file'), (req, res) => {
  pool.getConnection((err, connection) => {
    console.log(req.file);
    const { filename, path, mimetype, size } = req.file;
    const sql = `INSERT INTO images (filename, path, mimetype, size) VALUES (?, ?, ?, ?)`;

```

```

connection.query(sql, [filename, path, mimetype, size], (err, result) => {
  if (err) {
    console.error(err);
    return res.status(500).send('Internal Server Error');})
// req.file contains information about the uploaded file
res.send('File uploaded successfully!');})
});
//GET by ID
app.get('/bus/:destination_name', (req, res) => {
  pool.getConnection((err, connection) => {
    var keyword = ("% " + req.query.destination_name + "%")
    if (err) throw err
    console.log('connected as id ${connection.threadId}')
    connection.query('select bus_name,bus_time,destination_name from bus_table where
destination_name=? ', [req.params.destination_name], (err, rows) => {
      connection.release() // return the connection to pool
      if (!err) {
        res.send(rows);}
    else {console.log(err)}}))})
app.get('/pattern/:cname', (req, res) => { // path as the above path is same.
  pool.getConnection((err, connection) => {
    console.log(req.params.cname);
    if (err) throw err
    console.log('connected as id ${connection.threadId}')
    connection.query('select * from connect where cname like ? or category like ? order by
cname ASC ', ['%' + req.params.cname + '%', '%' + req.params.cname + '%'], (err, rows) => {
      connection.release() // return the connection to pool
      if (!err) {res.send(rows);}else {console.log(err)}}))})
app.get('/autos/:auto_stand', (req, res) => { // path as the above path is same.
  pool.getConnection((err, connection) => {
    console.log("% " + req.params.auto_stand + "%")
    if (err) throw err
    console.log('connected as id ${connection.threadId}')

```

```

    connection.query('select  autodriver_name,auto_number,auto_stand  from  auto_rikshaw
where auto_stand = ? ', [req.params.auto_stand], (err, rows) => {
    connection.release() // return the connection to pool
    if (!err) {res.send(rows);}else {console.log(err)}}))
app.get('/autostands/', (req, res) => { // path as the above path is same.
    pool.getConnection((err, connection) => {
        if (err) throw err
        console.log('connected as id ${connection.threadId}')
        connection.query('select  autodriver_name,auto_number,auto_stand  from  auto_rikshaw ',
(err, rows) => {
            connection.release() // return the connection to pool
            if (!err) {res.send(rows);}else {console.log(err)}}))
app.get('/getCat/:cat', (req, res) => {
    pool.getConnection((err, connection) => {
        var keyword = ('%' + req.params.cat + '%');
        if (err) throw err
        console.log('connected as id ${connection.threadId}')
        connection.query('SELECT distinct (category) FROM connect WHERE category like ? ',
keyword, (err, rows) => {
            connection.release() // return the connection to pool
            if (!err) {res.send(rows);}else {console.log(err)}}));
const url = require('url');
const grocery_upload = multer({ dest: 'mybedra/img/' });
app.post('/grocery_upload', grocery_upload.single('item_pic'), (req, res) => {
    const { item_name, item_price } = req.body;
    const item_image = req.file; // Get the uploaded file
    const uploadDir = path.join(__dirname, '../mybedra/img'); // Specify the upload directory
    outside the src folder
    const imageName = `${Date.now()}-${item_image.originalname}`; // Generate a unique
image name
    const imagePath = path.join(uploadDir, imageName); // Set the image path
    // Move the uploaded file to the specified directory
    const fs = require('fs');
    fs.rename(item_image.path, imagePath, (err) => {

```

```

if (err) {
  console.error('Error saving image:', err);
  res.status(500).send('Error saving image');
} else {
  const sql = 'INSERT INTO grocery (item_name, item_price, item_image) VALUES (?, ?, ?)';
  const values = [item_name, item_price, `img/${imageName}`]; // Store the image path in the database
  pool.query(sql, values, (err, result) => {
    if (err) {
      console.error('Error uploading data:', err);
      res.status(500).send('Error uploading data');
    } else {
      console.log('Data uploaded successfully');
      res.redirect('http://localhost/mybedra/cm/grocery.php'); // Redirect the client to the specified URL}}}}));
const food_upload = multer({ dest: 'mybedra/img/' });
app.post('/food_upload', food_upload.single('item_pic'), (req, res) => {
  const { item_name, item_price, hotel_name } = req.body;
  const item_image = req.file; // Get the uploaded file
  const uploadDir = path.join(__dirname, '../mybedra/img'); // Specify the upload directory outside the src folder
  const imageName = `${Date.now()}-${item_image.originalname}`; // Generate a unique image name
  const imagePath = path.join(uploadDir, imageName); // Set the image path
  // Move the uploaded file to the specified directory
  const fs = require('fs');
  fs.rename(item_image.path, imagePath, (err) => {
    if (err) {
      console.error('Error saving image:', err);
      res.status(500).send('Error saving image');
    } else {
      const sql = 'INSERT INTO foods (item_name, item_price, hotel_name, item_image) VALUES (?, ?, ?, ?)';

```

```

    const values = [item_name, item_price, hotel_name, `img/${imageName}`]; // Store the
image path in the database
    pool.query(sql, values, (err, result) => {
        if (err) {
            console.error('Error uploading data:', err);
            res.status(500).send('Error uploading data');
        } else {
            console.log('Data uploaded successfully');
            res.redirect('http://localhost/mybedra/cm/food.php');});});});});
// Fetch data from the database
app.get('/grocery_fetch', (req, res) => {
    const sql = 'SELECT * FROM grocery';

    pool.query(sql, (err, result) => {
        if (err) {
            console.error('Error fetching data:', err);
            return res.status(500).send('Error fetching data'); // Add return statement here}
            console.log('Data fetched successfully');
            res.status(200).json(result);});});
app.get('/food_fetch', (req, res) => {
    const sql = 'SELECT * FROM foods';
    pool.query(sql, (err, result) => {
        if (err) {
            console.error('Error fetching data:', err);
            return res.status(500).send('Error fetching data'); // Add return statement here}
            console.log('Data fetched successfully');
            res.status(200).json(result);});});
app.post('/grocery_ordered', (req, res) => {
    const orderedItems = req.body;
    // Display the received contents
    console.log('Ordered Items:');
    console.log('Type of orderedItems:', typeof orderedItems);
    console.log('Value of orderedItems:', orderedItems);
    // Insert data into the "order" table

```

```

orderedItems.forEach(item => {
  const { user_name, item_name, item_price, quantity, address, phoneNumber } = item;
  const query = `INSERT INTO groceryorder (user_id, item_name, item_price, item_quantity,
status, user_address, user_phoneNumber) VALUES (?, ?, ?, ?, ?,?,?)`;
  const values = [user_name, item_name, item_price, quantity, "ordered", address,
phoneNumber];
  // Execute the SQL query
  pool.query(query, values, (error, results) => {
    if (error) throw error;
    console.log('Inserted data into "order" table.');
```

```

pool.query(query, (err, results) => {
  if (err) throw err;
  res.json(results);});});
app.get('/food_userorderlist', (req, res) => {
  const query = `SELECT * FROM foodorder WHERE status='ordered';

pool.query(query, (err, results) => {
  if (err) throw err;
  res.json(results);});});
app.post('/grocery_acceptOrders', (req, res) => {
  const { userName } = req.body;
  // Update the status in the database
  const sql = 'UPDATE groceryorder SET status = "accepted" WHERE user_id = ?';
  pool.query(sql, [userName], (error, results) => {
    if (error) {
      console.error('Failed to update order status:', error);
      res.status(500).json({ message: 'Failed to update order status' });
    } else {
      console.log('Order status updated');
      res.status(200).json({ message: 'Order status updated successfully' });});});});
app.post('/food_acceptOrders', (req, res) => {
  const { userName } = req.body;
  // Update the status in the database
  const sql = 'UPDATE foodorder SET status = "accepted" WHERE user_id = ?';
  pool.query(sql, [userName], (error, results) => {
    if (error) {
      console.error('Failed to update order status:', error);
      res.status(500).json({ message: 'Failed to update order status' });
    } else {
      console.log('Order status updated');
      res.status(200).json({ message: 'Order status updated successfully' });});});});
app.post('/grocery_search', (req, res) => {
  const searchTerm = req.body.searchTerm;
  // Perform the search query

```



```

const query = `SELECT * FROM grocery WHERE item_name LIKE '%${searchTerm}%'`;
pool.query(query, (err, results) => {
  if (err) {
    console.error('Error executing the search query:', err);
    res.status(500).json({ error: 'Internal server error' });
    return;
  }
  console.log('Searched .');
  // Send the matched results back to the client
  res.json(results);});});
app.post('/food_search', (req, res) => {
  const searchTerm = req.body.searchTerm;
  // Perform the search query
  const query = `SELECT * FROM foods WHERE item_name LIKE '%${searchTerm}%' OR
hotel_name LIKE '%${searchTerm}%'`;
  pool.query(query, (err, results) => {
    if (err) {
      console.error('Error executing the search query:', err);
      res.status(500).json({ error: 'Internal server error' });
      return;
    }
    console.log('Searched .');
    // Send the matched results back to the client
    res.json(results);});});
app.get('/categories_search', (req, res) => {
  const searchString = req.query.searchString;
  // Perform the database query
  const query = `SELECT cname, contact1, contact2 FROM connect WHERE category LIKE
'${searchString}'`;
  pool.query(query, (err, results) => {
    if (err) {
      console.error('Error executing query: ', err);
      res.status(500).json({ error: 'Error executing query' });
      return;
    }
    res.json(results);
    console.log(results);});});

```

```

app.get('/getbuses', (req, res) => {
  const route = req.query.route;
  const query = `SELECT bus_name, bus_time, destination_name FROM bus_table WHERE
destination_name='${route}'`;
  pool.query(query, (err, result) => {
    if (err) {
      res.status(500).json({ error: 'An error occurred' });
    } else {
      const item_list = result.map((row) => {
        return {
          bus_name: row.bus_name,
          bus_time: row.bus_time,
          destination_name: row.destination_name;});
      res.json(item_list);});});
// Initialize Firebase Admin SDK
const serviceAccount = require('../mybedraauth.json');
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount)});
// Firebase Firestore
const db = admin.firestore();
app.get('/api/fetch-data', async (req, res) => {
  try {
    const snapshot = await db.collection('User').get();
    snapshot.forEach((doc) => {
      const docData = doc.data();
      const docId = doc.id;
      const username = docData.username;
      const phoneNumber = docData.phoneNumber;
      const address = docData.address;
      const email = docId;
      const selectQuery = `SELECT * FROM user_details WHERE email = ?`;
      const selectValues = [email];
      pool.query(selectQuery, selectValues, (error, results) => {
        if (error) {

```

```

        console.error('Error selecting data from MySQL:', error);
    } else {
        if (results.length > 0) {
            // Email exists, perform update
            const updateQuery = `UPDATE user_details SET user_name = ?, phone_number = ?,
location = ? WHERE email = ?`;
            const updateValues = [username, phoneNumber, address, email];
            pool.query(updateQuery, updateValues, (error, results) => {
                if (error) {console.error('Error updating data in MySQL:', error);});
            } else {
                // Email does not exist, perform insert
                const insertQuery = `INSERT INTO user_details (user_name, email, phone_number,
location) VALUES (?, ?, ?, ?)`;
                const insertValues = [username, email, phoneNumber, address];
                pool.query(insertQuery, insertValues, (error, results) => {
                    if (error) {console.error('Error inserting data into MySQL:', error);});});
                console.log('Data has been fetched from Firestore and stored in the SQL table');
                res.status(200).json({ message: 'Data fetched and stored successfully' });
            } catch (error) {
                console.error('Error fetching Firestore data:', error);
                res.status(500).json({ error: 'An error occurred while fetching data' });});
        //auto search
        app.get('/search_auto_list', (req, res) => {
            const stand = req.query.standName;
            const query = `SELECT autodriver_name, auto_number, auto_stand FROM auto_rikshaw
WHERE auto_stand = '${stand}'`;
            pool.query(query, (error, results) => {
                if (error) {
                    console.error('Error executing MySQL query:', error);
                    res.status(500).json({ error: 'An error occurred while fetching data' });
                } else {
                    const item_list = results.map((row) => {
                        return {
                            auto_name: row.autodriver_name,

```

```

        auto_number: row.auto_number,
        auto_stand: row.auto_stand}}});
    console.log(item_list);
    res.json(item_list)}}});
app.get('/api/destinations/auto', (req, res) => {
    const query = 'SELECT DISTINCT auto_stand FROM auto_rikshaw';
    // Execute the query
    pool.query(query, (error, results) => {
        if (error) {
            console.error('Error executing the query: ', error);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }
        // Process the query results
        const destinations = results.map((row) => row.auto_stand);
        res.json({ destinations }}));
    });
app.get('/api/destinations/bus', (req, res) => {
    const query = 'SELECT DISTINCT destination_name FROM bus_table';
    // Execute the query
    pool.query(query, (error, results) => {
        if (error) {
            console.error('Error executing the query: ', error);
            res.status(500).json({ error: 'Internal server error' });return;}
        // Process the query results
        const destinations = results.map((row) => row.destination_name);
        res.json({ destinations }}));
    });
app.listen(port, () => console.log(`Listening on port ${port}`))

```

NOTE: Admin code was given by the company.

CHAPTER

-7-

INTERFACES

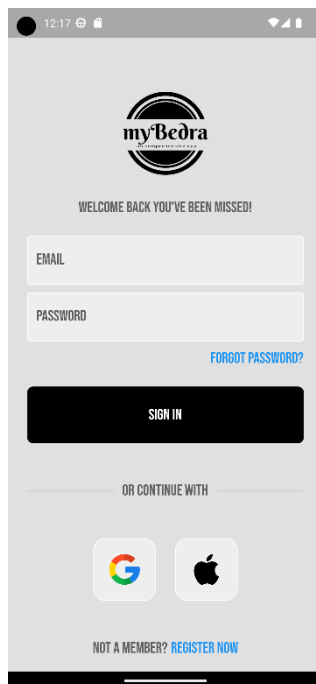
7. Interfaces

7.1. Android View

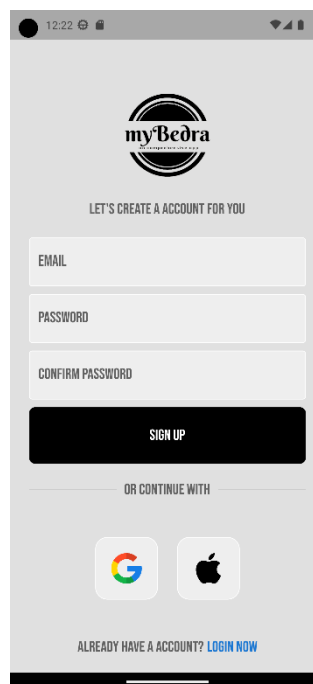
7.1.1. Logo



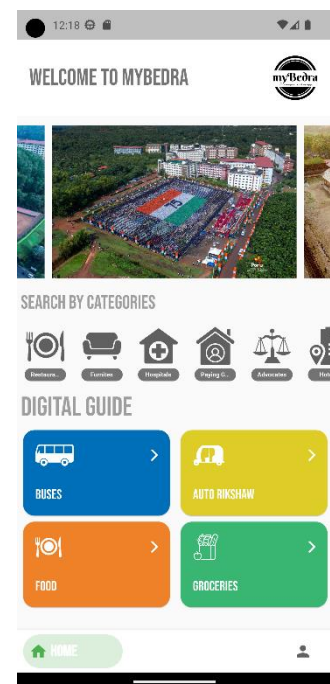
7.1.2. Login Page



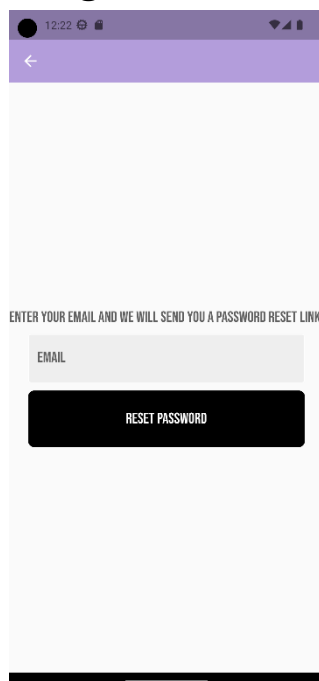
7.1.3. Registration Page



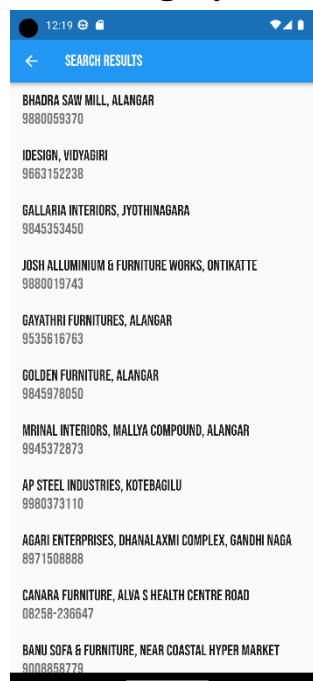
7.1.4. Home Page



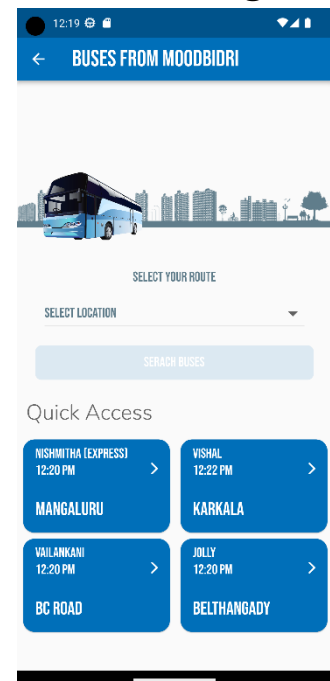
7.1.5. Forgot Password Page



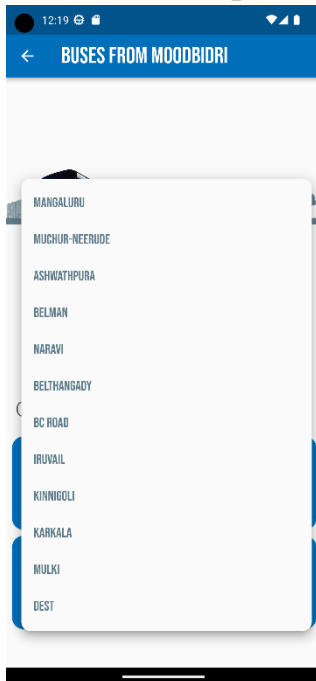
7.1.6. Category Search



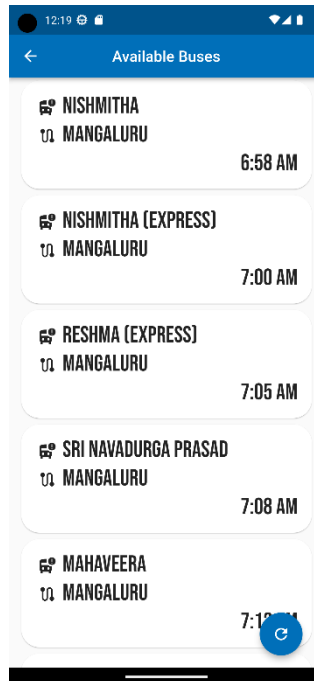
7.1.7. Bus Page



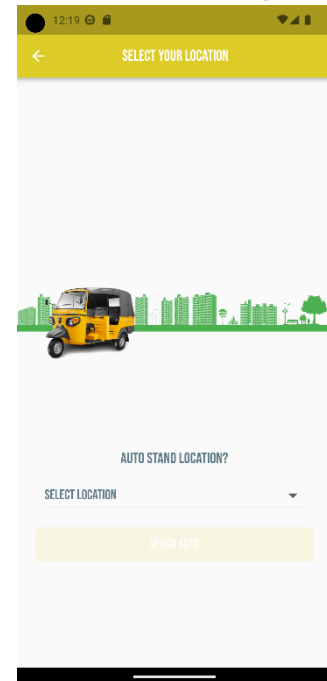
7.1.8. Bus Dropdown



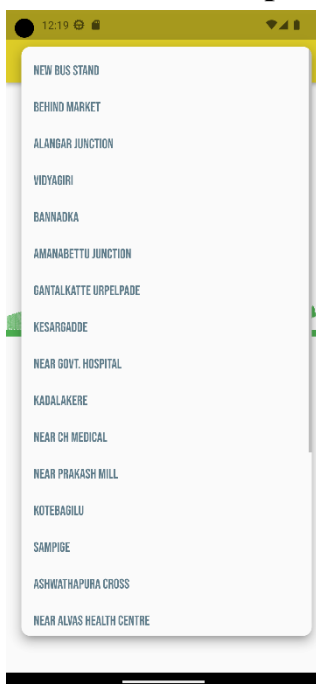
7.1.9. Bus Search



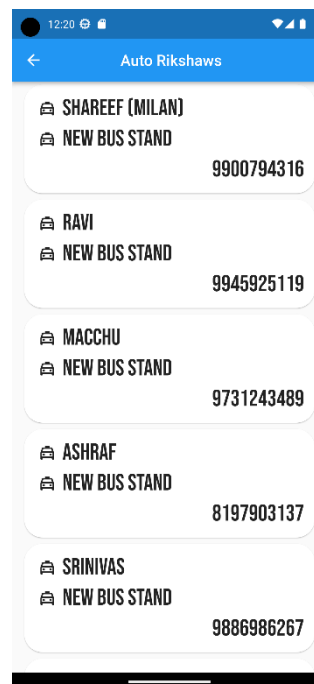
7.1.10. Auto Page



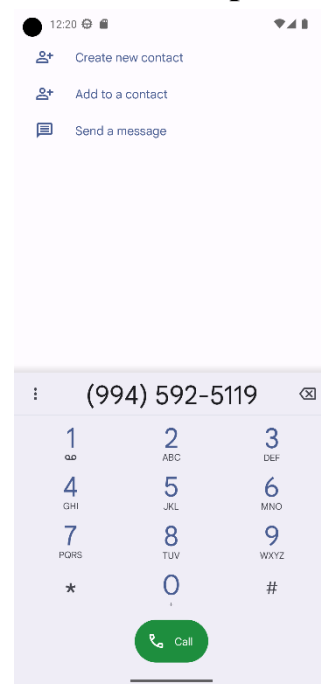
7.1.11. Auto Dropdown



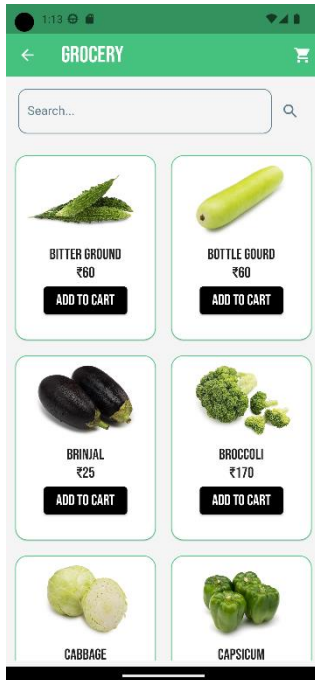
7.1.12. Auto Search



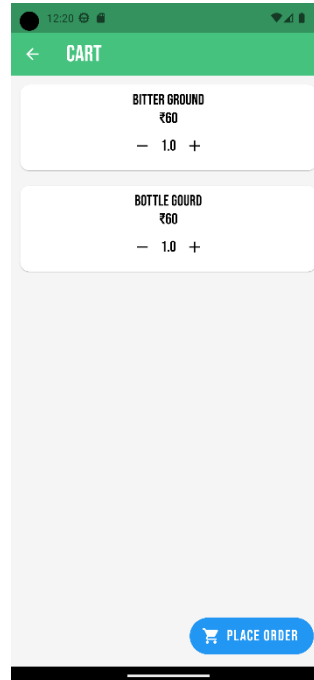
7.1.13. On Tap Call



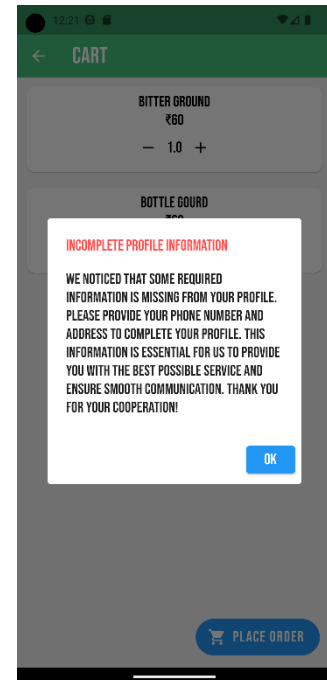
7.1.14. Grocery Page



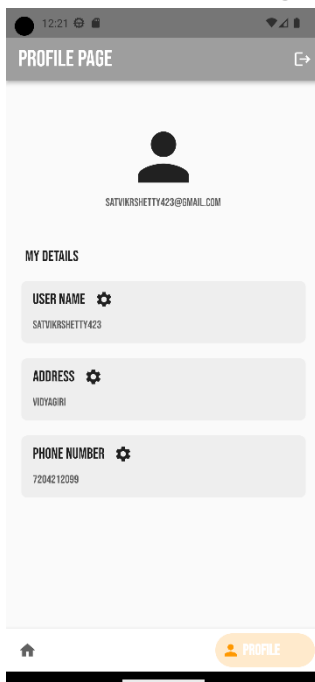
7.1.15. Cart Page



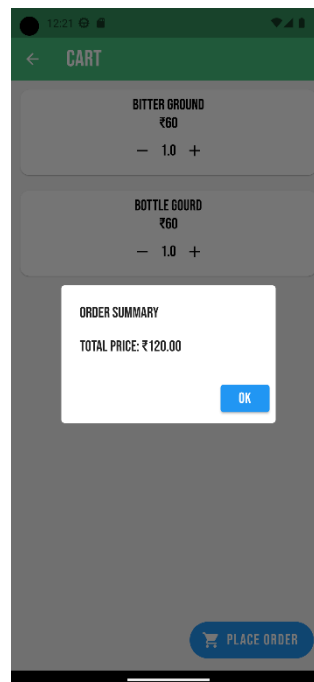
7.1.16. Error Page



7.1.17. Profile Page

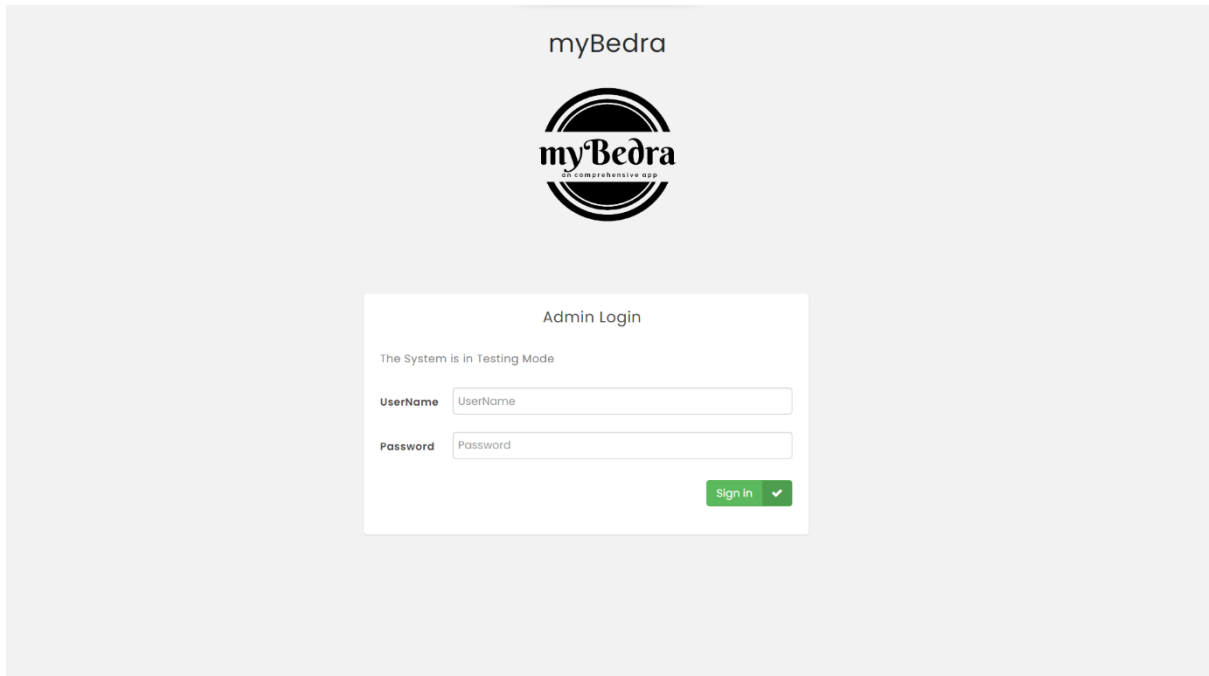


7.1.18. Successful Order



7.2. Website View

7.2.1. Login Page



The image shows the Admin Login page for myBedra. At the top, the myBedra logo is displayed, featuring a circular emblem with the text "myBedra" and "AN COMPREHENSIVE APP" below it. Below the logo, the title "Admin Login" is centered. A message states "The System is in Testing Mode". There are two input fields: "UserName" and "Password". A green "Sign In" button with a checkmark is located at the bottom right of the login form.

myBedra

myBedra
AN COMPREHENSIVE APP

Admin Login

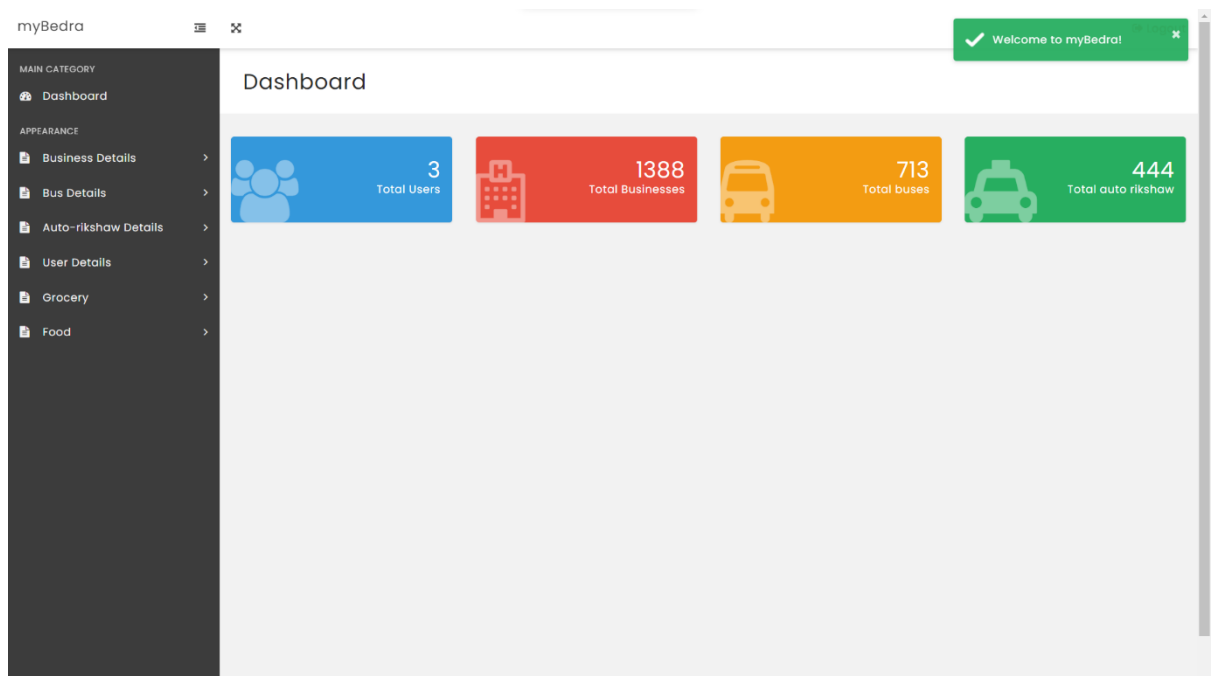
The System is in Testing Mode

UserName

Password

Sign In ✓

7.2.2. Dashboard



7.2.3. Add Business

myBedra

Logout

MAIN CATEGORY

Dashboard

APPEARANCE

Business Details

Bus Details

Auto-rikshaw Details

User Details

Grocery

Food

Add Business Record

Home / Add Business Record

Fill details to add Business record

Business name

Full address

Category

Contact1

Contact2

Add

7.2.4. Display Business Listing

myBedra

Logout

MAIN CATEGORY

Dashboard

APPEARANCE

Business Details

Bus Details

Auto-rikshaw Details

User Details

Grocery

Food

ALL LISTS

Home / Listings / Display

View Classes Info

Show 10 entries

Search:

#	name	category	contact1	contact2	Action
1	LG Fancy, Hosangady	1 Gram Gold	9481919888	sdds	
2	Shrinidhi Refrigeration, Swaraj Residency	AC / Refrigerator - Sales & Service	9880925420	08258-238622	
3	Jinaprabha Refrigeration, Swaraj Maidan Rd.	AC / Refrigerator - Sales & Service	9663860476	457896542	
4	JMJ Weather Cooler, Alangar	AC / Refrigerator - Sales & Service	9902196277		
5	Padma Refrigeration, Kalsanka	AC / Refrigerator - Sales & Service	9945619496		
6	Dhanraj Refrigeration, Opp. Mescom	AC / Refrigerator & Service	9845353451		
7	Total Alluminium Material Centre, Near Shalimar Ha	ACP / Alluminium / PVC Doors	8310285598		
8	MJ Impress Dance Group & Costume Centre, Bus Stand	Acting Classes	9880960114	8296011535	
9	Samparka Madhyama, Alva s Health Centre Campus	Ad Agency - Udayavani	9448726972	08258-238969	
10	Sharath D. Shetty, Panchami Complex	Advocates	9845593123		
#	name	category	contact1	contact2	Action

Showing 1 to 10 of 1,388 entries

Previous

1

2

3

4

5

...

139

Next

7.2.5. Add Bus

myBedra ☰ ✖ Logout

MAIN CATEGORY
Dashboard

APPEARANCE
Business Details >
Bus Details >
Auto-rikshaw Details >
User Details >
Grocery >
Food >

Add bus record

Home / Add bus record

Fill details to add Bus record

Bus name

Bus time

Destination name

7.2.6. Display Bus Listing

myBedra ☰ ✖ Logout

MAIN CATEGORY
Dashboard

APPEARANCE
Business Details >
Bus Details >
Auto-rikshaw Details >
User Details >
Grocery >
Food >

Display all Bus Records

Home / Bus data / Display

View Classes Info

Show entries Search:

#	bus name	bus time	destination name
1	Vishal (EXPRESS)	450	Mangaluru
2	Reshma (EXPRESS)	540	Mangaluru
3	Sri Navadurga Prasad	600	Mangaluru
4	Nishmitha	635	Mangaluru
5	Christa Prasad	640	Mangaluru
6	Nishmitha (EXPRESS)	650	Mangaluru
7	Nishmitha	658	Mangaluru
8	Nishmitha (EXPRESS)	700	Mangaluru
9	Reshma (EXPRESS)	705	Mangaluru
10	Sri Navadurga Prasad	708	Mangaluru
#	bus name	bus time	destination name

Showing 1 to 10 of 713 entries

Previous **1** 2 3 4 5 ... 72 Next

7.2.7. Add Auto Rickshaw

myBedra

Logout

MAIN CATEGORY

Dashboard

APPEARANCE

Business Details

Bus Details

Auto-rikshaw Details

User Details

Grocery

Food

Add auto records

Home / Add rikshaw details

Fill details to add rikshaw record

Driver name

contact number

Auto-stand name

Add

7.2.8. Display Auto Listing

myBedra

Logout

MAIN CATEGORY

Dashboard

APPEARANCE

Business Details

Bus Details

Auto-rikshaw Details

User Details

Grocery

Food

Display all auto records

Home / auto details / Display

View Job Info

Show 10 entries Search:

#	auto driver name	contact number	auto stand
1	Shareef (Milan)	9900794316	New Bus Stand
2	Ravi	9945925119	New Bus Stand
3	Macchu	9731243489	New Bus Stand
4	Ashraf	8197903137	New Bus Stand
5	Srinivas	9886986267	New Bus Stand
6	Dolli	9481757982	New Bus Stand
7	Vishwa	9741444673	New Bus Stand
8	Manju	9901328209	New Bus Stand
9	Prashanth	9449555664	New Bus Stand
10	Naveen	9731920924	New Bus Stand
#	auto driver name	contact number	auto stand

Showing 1 to 10 of 444 entries

Previous12345...45Next

7.2.9. Display User Details

The screenshot shows the 'Display all user records' page in the myBedra application. The left sidebar contains a 'MAIN CATEGORY' section with 'Dashboard' and an 'APPEARANCE' section with 'Business Details', 'Bus Details', 'Auto-rikshaw Details', 'User Details', 'Grocery', and 'Food'. The main content area has a header 'Display all user records' and a breadcrumb 'Home / User data / Display'. Below this is a 'View User info' section with a 'Show 10 entries' dropdown and a 'Search:' input field. A table displays user records with columns: #, User name, Email, Phone number, and Location. The table contains three entries. At the bottom, it says 'Showing 1 to 3 of 3 entries' and has 'Previous', '1', and 'Next' navigation buttons.

#	User name	Email	Phone number	Location
1	m.thouseef03	m.thouseef03@gmail.com	8971548797	karkaLA
2	ashwathiruvail	ashwathiruvail@gmail.com	7483812978	Iruvail
3	oskullxcalibreo	oskullxcalibreo@gmail.com	7204212099	vidyagiri

7.2.10. Add Grocery

The screenshot shows the 'Add Grocery' page in the myBedra application. The left sidebar is identical to the previous screenshot. The main content area has a header 'Add Grocery' and a breadcrumb 'Home / Add Grocery'. Below this is a 'Fill details to add Grocery' section with three input fields: 'Item Name', 'Item Price', and 'Item Image'. The 'Item Image' field has a 'Choose File' button and the text 'No file chosen'. An 'Add' button is at the bottom of the form.

7.2.11. Add Food

myBedra ☰ ✖ Logout

MAIN CATEGORY

- Dashboard

APPEARANCE

- Business Details
- Bus Details
- Auto-rikshaw Details
- User Details
- Grocery
- Food

Add Food

Home / Add Food

Fill details to add Food

Item Name

Item Price

Hotel Nam

Item Image No file chosen

7.2.12. Display Order List

myBedra ☰ ✖ Logout

MAIN CATEGORY

- Dashboard

APPEARANCE

- Business Details
- Bus Details
- Auto-rikshaw Details
- User Details
- Grocery
- Food

Display Orders

Home / User data / Display

View User Info

User Name	Item Name	Item Price	Item Quantity	Status	Accept
oskullxcalibre@gmail.com	Cabbage	30	1	ordered	<input type="button" value="Accept"/>
	Capsicum	70	1	ordered	
	Carrot	90	1	ordered	
	Cauliflower	40	2	ordered	
satvikrshetty423@gmail.com	Tomato	40	1	ordered	<input type="button" value="Accept"/>
	Onion	120	1	ordered	
	Garlic	200	0	ordered	

7.2.13. Display Accepted Order List

myBedra

Logout

MAIN CATEGORY

Dashboard

APPEARANCE

Business Details

Bus Details

Auto-rikshaw Details

User Details

Grocery

Food

Display Accepted Orders

Home / User data / Display

View User info

User Name	Item Name	Item Price	Item Quantity	Status	Address	Phone Number
oskullxcalibre@gmail.com	Cabbage	30	1	accepted	vidyagiri	7204212099
	Capsicum	70	1	accepted	vidyagiri	7204212099
	Carrot	90	1	accepted	vidyagiri	7204212099
satvikshetty423@gmail.com	Cauliflower	40	2	accepted	vidyagiri	7204212099
	Tomato	40	1	accepted	Alangar	8762080989
	Onion	120	1	accepted	Alangar	8762080989
	Garlic	200	0	accepted	Alangar	8762080989

CHAPTER

-8-

TESTING

8. Testing

8.1. Introduction

Testing is a fundamental and systematic process that involves running a program with the explicit purpose of discovering errors. Its primary objective is to evaluate the correctness, completeness, and overall quality of computer software. Software testing encompasses various approaches, but for intricate products, it is an investigative process that delves into the intricacies of the software. Through testing, we can ascertain whether the software performs as intended, thereby ensuring its verification and validation. This entails employing both static and dynamic methodologies to examine the application thoroughly. When it comes to designing test cases, two distinct methods are commonly utilized. These methods aid in structuring and organizing the testing process to achieve comprehensive and effective evaluation of the software.

Software testing serves three primary purposes:

1. **Verification:** The verification process ensures that the software aligns with its technical specifications. These specifications define the desired functions and their expected measurable outputs based on specific input values and preconditions. By conducting verification, software testers can confirm that the software operates as intended according to the defined specifications.
2. **Validation:** The validation process focuses on confirming that the software meets the business requirements. It ensures that the software satisfies the needs and expectations of the stakeholders and effectively supports the desired business processes. Through validation, software testers verify that the software is suitable for its intended purpose and aligns with the overall business objectives.
3. **Defect Management:** Defects are discrepancies between the expected and actual results. They can originate from faults introduced during the specification, design, or development phases. However, not all defects lead to failures or malfunctions in the software. Effective defect management involves identifying and documenting defects, investigating their root causes, and prioritizing their resolution based on their impact on the software's functionality and usability.

Software testing encompasses two main types: white box testing and black box testing.

- **White Box Testing:** White box testing focuses on the internal logic and structure of the code. It involves understanding the code's internal workings and writing tests based on code coverage, branches, statements, and logic. Testers proficient in coding and logic are required to carry out white box testing.
- **Advantages:**
 - Knowledge of the code structure facilitates effective test case design.
 - Optimization and removal of redundant code can be achieved.
- **Disadvantages:**
 - Skilled testers with knowledge of code internals are needed, increasing the cost.
 - It may be challenging to uncover hidden errors in every line of code, potentially leading to application failures.
- **Black Box Testing:** Black box testing examines the test object from an external perspective, without knowledge of its internal structure. Test cases are derived based on valid and invalid inputs, with the objective of determining correct outputs. This method applies to various levels of testing, including unit, internal, functional, system, and acceptance testing.
- **Advantages:**
 - Black box tests are reproducible.
 - The software environment is tested alongside the program.
 - Efforts invested in test design can be reused.
- **Disadvantages:**
 - Test results may sometimes overestimate the actual software performance.
 - Not all software properties can be thoroughly tested.
 - Root causes of failures may remain unidentified.

8.2. Test Reports

8.2.1. Unit Testing

Unit testing involves the meticulous examination of individual units or components within a software application. Primarily conducted by developers, this testing method ensures that each unit functions as intended. Typically automated, unit

tests target specific sections of code, such as particular functions or methods. Unit testing takes place at the lowest level of the software development process, allowing isolated testing of individual code units.

8.2.2. Integration Testing

Integration testing focuses on assessing how different units or components of a software application interact with each other. By combining these units, integration testing identifies and resolves any potential issues. Typically performed after unit testing and before functional testing, this process verifies the seamless collaboration of diverse software units.

8.2.3. System Testing

System testing is conducted on the entire software system, considering either system requirement specifications or functional requirement specifications, or both. It ensures that the software operates smoothly across different operating systems. System testing falls under the purview of black box testing, examining the system as a whole without focusing on internal implementation details.

8.3. Test Cases

User Login

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	If user clicks on signin button without entering username and password.	Inputs are not given	Please fill in all fields	Please fill in all fields	Pass
2	Enter correct username & Password and hit signin button	username test@xyz.com Password: *****	Login success	Login success	Pass
3	If username is blank but password is entered.	Password: *****	Please fill in all fields	Please fill in all fields	Pass
4	If password is blank but username is entered.	Username: abcd	Please fill in all fields	Please fill in all fields	Pass
5	If the username or password is incorrect.	Username: abcd Password: *****	Wrong password	Wrong password	Pass

User Signup

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	If user clicks signup button	Inputs are not given	Please fill in all fields	Please fill in all fields	pass

	without entering Email				
2	If user clicks signup button without entering Password	Inputs are not given	Please fill in all fields	Please fill in all fields	pass
3	If user enters all the fields and clicks on signup button	satvikrshetty423@gmail.com , *****	Successfully completed	Successfully completed	pass
4	If user inputs invalid mail format	Adsaddas	Invalid email	Invalid email	pass
5	If user inputs password less than 8 character	*****	Password must have at least 8 characters	Password must have at least 8 characters	pass
6	If user inputs password 8 character or more but does not have special character	*****	Password must contain at least one special character	Password must contain at least one special character	pass
7	If user inputs password 8 character or more but does not have number	*****	Password must contain at least one number	Password must contain at least one number	pass

HomePage

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
------	--------	--------	-----------------	---------------	-------------

1	If user clicks on Buses Button	No inputs	Bus module should open	Bus module should open	Pass
2	If user clicks on Auto Rickshaw Button	No inputs	Auto module should open	Auto module should open	Pass
3	If user clicks on Food Button	No inputs	Food module should open	Food module should open	Pass
4	If user clicks on Grocery Button	No inputs	Grocery module should open	Grocery module should open	Pass
5	If user clicks on category icons	No inputs	Respected module should open	Respected module should open	Pass

Buses page

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	If user clicks on dropdown menu	No inputs	Dropdown with bus routes should be displayed	Dropdown with bus routes should be displayed	Pass
2	If user clicks on dropdown when the list is empty	No inputs	Dropdown should be disabled	Dropdown should be disabled	Pass
3	If user chooses a bus route from dropdown and clicks on search bus button	No inputs	Bus list of respective routes should be displayed	Bus list of respective routes should be displayed	Pass
4	If user clicks on quick access cards	No inputs	Bus list of respective routes card should be displayed	Bus list of respective routes card should be displayed	Pass

5	If user click on search busses button without choosing a route	No inputs	Button should be disabled	Button should be disabled	Pass
---	--	-----------	---------------------------	---------------------------	------

Auto Rickshaw page

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	If user clicks on dropdown menu	No inputs	Dropdown with bus routes should be displayed	Dropdown with bus routes should be displayed	Pass
2	If user clicks on dropdown when the list is empty	No inputs	Dropdown should be disabled	Dropdown should be disabled	Pass
3	If user click on search auto button without choosing an auto stand	No inputs	Button should be disabled	Button should be disabled	Pass
4	If user chooses a auto stand from dropdown and clicks on search auto button	No inputs	Auto list of respective auto stand should be displayed	Auto list of respective auto stand should be displayed	Pass

Auto Rickshaw (after search) page

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	If user clicks on driver card	No inputs	Phone's default phone app should open with	Phone's default phone app should open with driver	Pass

			driver number pre written in it	number pre written in it	
--	--	--	------------------------------------	-----------------------------	--

Food page

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	If user clicks on add to cart button	No inputs	Item should be added to cart	Item should be added to cart	Pass
2	If user searches for an item	Item name	Searched item should be displayed	Searched item should be displayed	Pass
3	If user searches without inputs	No inputs	Entire item list should be displayed	Entire item list should be displayed	Pass

Grocery page

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	If user clicks on add to cart button	No inputs	Item should be added to cart	Item should be added to cart	Pass
2	If user searches for an item	Item name	Searched item should be displayed	Searched item should be displayed	Pass
3	If user searches without inputs	No inputs	Entire item list should be displayed	Entire item list should be displayed	Pass

Cart page

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	If user clicks on place order button	Items that are added to cart	Order successful	Order successful	Pass
2	If user clicks on place order button when no item is added to cart	No input	No item. Add items to cart to place order	No item. Add items to cart to place order	Pass
3	If user clicks on place order button when his profile is incomplete	Items that are added to cart	Incomplete profile information error should be displayed	Incomplete profile information error should be displayed	Pass

CONCLUSION

In conclusion, the proposed system for Moodbidri seeks to achieve several key objectives. Firstly, it aims to provide convenient information access to the residents, allowing them easy and quick access to information about bus schedules, autorickshaws, food options, grocery stores, and local businesses. This ensures that individuals can effortlessly find the information they need to navigate the town.

Secondly, the system aims to enhance the commuting experience by offering real-time updates on bus routes, schedules, and arrival times. Additionally, it facilitates the location and booking of autorickshaws, making commuting more efficient and convenient for residents.

Moreover, the system strives to promote and support local businesses by providing a platform for residents to discover and engage with a diverse range of products and services available within Moodbidri. This helps to strengthen the local economy and foster economic growth within the town.

LIMITATIONS

- **Limited Coverage:** myBedra is limited to serving the residents of Moodbidri town only, excluding users from other areas.
- **Connectivity Requirements:** Accessing and using myBedra may require a stable internet connection, which may not always be available in certain areas.
- **Reliance on Service Providers:** The quality of services such as transportation and food delivery depend on the performance and reliability of local service providers.
- The app is not multi linguistic and can not be used by disabled poeple

FUTURE SCOPE

The myBedra app has immense potential to grow. Payment system can be implemented for bus module, autorickshaw module, groceries module and restaurants module. Order tracking system can be added to groceries and restaurants module. Vehicle tracking system can be added to bus and autorickshaw module.

ABBREVIATION AND ACRONYMS

Abbreviations

- **DFD** - Data Flow Diagram
- **CFD** - Context Flow Diagram
- **DBMS** - Database Management System.
- **SRS** - Software Requirement specification
- **ER** - Entity Relation
- **MySQL** - Structured Query Language
- **GUI** - Graphical User Interface

Acronyms

- **DFD:** It is a way of representing a flow of data of a process or a system.
- **DBMS:** Database Management System is a software for storing and retrieving user's data while considering appropriate security measures.
- **CFD:** It is a tool popular among Business Analysts who use it to understand the details and boundaries of the system to be designed in a project.
- **SQL:** It is a standard language for storing, manipulating and retrieving data in database.

BIBLIOGRAPHY / REFERENCES

Websites

- <http://www.tutorialspoint.com>
- <http://www.youtube.com>
- <http://www.w3schools.com>
- <http://www.geeksforgeeks.com>
- <http://chat.openai.com>