

CONNECT ME**PROJECT REPORT**

Submitted to the Calicut University in partial fulfillment of the
requirement for the award of the degree of

Bachelor of Computer Application(BCA)**Submitted by:**

NAJMA MANAF (Reg.No: ARAWBCA042)

JESNA (Reg.No:ARAWBCA039)

RIHANA (Reg.NO:ARAWBCA044)

LIYANA (Reg.No:ARAWBCA040)

Under the guidance

Ms FEMILA K V



PG DEPARTMENT OF COMPUTER SCIENCE

ANSAR WOMEN'S COLLEGE,PERUMPILAVU

2022-2025



ANSAR WOMEN'S COLLEGE , PERUMPILAVU

PG DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the project report entitled “**CONNECT ME**” is a bonafide record of work done by **Ms.NAJMA MANAF (Reg.No:ARAWBCA042), Ms.JESNA(Reg.No:ARAWBCA039),Ms.RIHANA(Reg.No:ARAWBCA044),Ms.LIYANA(Reg.No:ARAWBCA040)** PG Department Computer Science, under the guidance and supervision of **Ms.FEMILA K V** (Assistant prof PG Department of Computer Science) in partial fulfillment of requirements for the award of degree of Bachelor Of Computer Application under the University of Calicut during the year 2022- 2025.

Internal guide:

Head of Department:

The presentation and viva-voce examination was conducted on
..... at Ansar Women's College.

Internal Examiner:

External Examiner:

DATE:



13/02/2025

CERTIFICATE

This is to certify that **NAJMA MANAF** (Reg. No: ARAWBCA042), **LIYANA** (Reg. No: ARAWBCA040), **JESNA O M** (Reg. No: ARAWBCA039), **RIHANA** (Reg. No: ARAWBCA044) have successfully completed their project entitled "**CONNECT ME**" in PYTHON+ANDROID under the guidance of our senior developers during July 2024 to February 2025.

During this period they were found hardworking, punctual & efficient. We wish them successful future.

For RISS TECHNOLOGIES

Project Manager



GMA Building, Manorama Jn. Thrissur-680001, Mob: 9544928881, 9544928882

Web: www.rissttechnologies.com, E-mail: rissthrissur@gmail.com

DECLARATION

We hereby declare that the project report entitled “**CONNECT ME (A NEIGHBORHOOD APP)**”, is the result of my original work done under the guidance of **Ms. FEMILA K V** (Assistant prof PG department of computer science).

I also declare that this work has not been submitted earlier by me too University of Calicut or any other institutions for the fulfillment of the requirement of the course of study. The imperial findings in the report are based on the data collected by ourselves.

Place : PERUMPILAVU

Name: JESNA

LIYANA

NAJMA MANAF

RIHANA

ACKNOWLEDGMENT

At the very outset, we would like to give the first honour to almighty God, who gave the wisdom and knowledge to complete this project.

We owe our deep sense of gratitude to **Ms. Fareedha**, the principal of Ansar Women's College, Perumpilavu for having encouraged us to undertake this project work. We would also like to express sincere thanks to **Ms. Nasheedha**, Head of the Department of Computer Science for making necessary arrangements to enable us to undertake our project with keen interest.

We would also like to express our gratitude to our internal guides **Ms. FEMILA** (Assistant prof PG department of computer science) for their valuable guidance and assistance in completing the project in a determined way within stipulated time .

We express our immense pleasure and thankfulness to our parents, friends and all teaching and non teaching staffs of the Department of Computer Science, Ansar Women's College, Perumpilavu for their encouragement and support

RIHANA
ARAWBCA044

TABLE OF CONTENTS

S1.NO	Contents	Page no
1	INTRODUCTION	1
1.1	SYNOPSIS	2
1.2	ABOUT THE PROJECT	2
1.3	MAIN OBJECTIVE OF THE PROJECT	3
2	SOFTWARE SPECIFICATION	4
2.1	INTRODUCTION TO PYTHON	5
2.2	INTRODUCTION TO SQL	9
3	REQUIREMENT SPECIFICATION	13
3.1	HARDWARE SPECIFICATION	14
3.2	SOFTWARE SPECIFICATION	14
4	PROJECT PLANNING AND SCHEDULING	15
5	SYSTEM ANALYSIS	18

5.1	OVERVIEW	19
5.2	FEASIBILITY ANALYSIS	19
5.2.1	TECHNICAL FEASIBILITY	21
5.2.2	ECONOMICAL FEASABILITY	21
5.2.3	OPERATIONAL FEASABIITY	22
6	SYSTEM DESIGN	24
6.1	INPUT DESIGN	25
6.2	OUTPUT DESIGN	25
6.3	DATAFLOW DIAGRAM	26
6.4	DATABASE DESIGN	32
6.4.1	MEMORY DATABASE	32
7	SYSTEM TESTING AND IMPLEMENTATION	36
7.1	SYSTEM TESTING	37
7.2	SYSTEM IMPLEMENTATION	42
7.3	SYSTEM MAINTENANCE	42

8	FUTURE ENHANCEMENT	45
9	CONCLUSION	46
10	APPENDIX	47
11	BIBLIOGRAPHY	57
12	SCREENSHOTS	58

1. INTRODUCTION

1.1 SYNOPSIS

An innovative Android application is designed to simplify access to essential services and products within a user's locality. It enables users to effortlessly find skilled workers, book services, order from nearby shops, avail pharmacy services, and request emergency assistance. With a user-friendly interface, the platform ensures smooth interaction between users, service providers, and businesses, making everyday tasks more efficient.

The primary goal of **CONNECT ME** is to establish a centralized system that connects users with local workers, shops, and pharmacies. It ensures secure transactions, effective communication, and quick access to essential services, enhancing user convenience and service provider efficiency.

1.2 ABOUT THE PROJECT

CONNECT ME is a user-friendly Android application designed to connect individuals with local services efficiently. It allows users to find skilled workers, book services, order from nearby shops, access pharmacy services, and request emergency assistance.

The platform supports multiple user roles, including users, workers, shop owners, and admins, ensuring seamless interactions. Key features include real-time emergency alerts, secure transactions, sentiment-based reviews, and efficient order management.

Built with the Django Framework for a secure backend and an intuitive Android frontend, Connect Me aims to simplify daily tasks by providing a reliable and accessible service platform.

1.3 MAIN OBJECTIVE OF THE PROJECT

The primary objective of **CONNECT ME** is to develop a comprehensive and user-friendly platform that seamlessly connects users with local service providers, businesses, and pharmacies, simplifying access to essential services and products. The application enables users to effortlessly find skilled workers, book services, order products, and seek emergency assistance, all within a single, intuitive interface. By integrating secure transactions, real-time communication, and a transparent review system, the platform ensures reliability and trust between users and providers. Additionally, **CONNECT ME** empowers service providers and businesses with efficient management tools, allowing them to handle bookings, process orders, track payments, and improve customer engagement. With a strong focus on security, scalability, and ease of use, the project aims to streamline local service accessibility, enhance user convenience, and contribute to the digital transformation of businesses, ultimately fostering a more connected and efficient community.

2. SOFTWARE SPECIFICATION

2.1 INTRODUCTION TO PYTHON



Python is a high-level, general-purpose and a very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry. Python Programming Language is very well suited for Beginners, also for experienced programmers with other programming languages like C++ and Java.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0.[32] Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle detecting garbage collection, reference counting, and Unicode support.

Features of python

1.Easy to Code

Python is a very high-level programming language, yet it is effortless to learn. Anyone can learn to code in Python in just a few hours or a few days. Mastering Python and all its advanced concepts, packages and modules might take some more time. However, learning the basic Python syntax is very easy, as compared to other popular languages like C, C++, and Java.

2.Easy to Read

Python code looks like simple English words. There is no use of semicolons or brackets, and the indentations define the code block. You can tell what the code is supposed to do simply by looking at it.

3.Free and Open-Source

Python is developed under an OSI-approved open-source license. Hence, it is completely free to use, even for commercial purposes. It doesn't cost anything to download Python or to include it in your application.

4.Robust Standard Library

Python has an extensive standard library available for anyone to use. This means that programmers don't have to write their code for every single thing unlike other programming languages. There are libraries for image manipulation, databases, unit testing, expressions and a lot of other functionalities. In addition to the standard library, there is also a growing collection of thousands of components, which are all available in the Python Package Index.

5.Interpreted

When a programming language is interpreted, it means that the source code is executed line by line, and not all at once. Programming languages such as C++ or Java are not interpreted, and hence need to be compiled first to run them. There is no need to compile Python because it is processed at runtime by the interpreter.

6.Portable

Python is portable in the sense that the same code can be used on different machines. Suppose you write a Python code on a Mac. If you want to run it on Windows or Linux later, you don't have to make any changes to it. As such, there is no need to write a program multiple times for several platforms.

7. Object-Oriented and Procedure-Oriented

A programming language is object-oriented if it focuses design around data and objects, rather than functions and logic. On the contrary, a programming language is procedure-oriented if it focuses more on functions (code that can be reused). One of the critical Python features is that it supports both object-oriented and procedure-oriented programming.

8. Extensible

A programming language is said to be extensible if it can be extended to other languages. Python code can also be written in other languages like C++, making it a highly extensible language.

9. Expressive

Python needs to use only a few lines of code to perform complex tasks. For example, to display Hello World, you simply need to type one line - `print ("Hello World")`. Other languages like Java or C would take up multiple lines to execute this.

10. Support for GUI

One of the key aspects of any programming language is support for GUI or Graphical. User Interface. A user can easily interact with the software using a GUI. Python offers various toolkits, such as Tkinter, wxPython and JPython, which allows for GUI's easy and fast development.

2.2 INTRODUCTION TO SQL



SQL is a language to operate databases; it includes Database Creation, Database Deletion, Fetching Data Rows, Modifying & Deleting Data rows, etc.

SQL stands for Structured Query Language which is a computer language for storing, manipulating and retrieving data stored in a relational database. SQL was developed in the 1970s by IBM Computer Scientists and became a standard of the American National Standards Institute (ANSI) in 1986, and the International Organization for Standardization (ISO) in 1987.

SQL is the standard language to communicate with Relational Database Systems. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their Standard Database Language.

Why SQL?

SQL is widely popular because it offers the following advantages –

Allows users to access data in the relational database management systems.

1. Allows users to describe the data.
2. Allows users to define the data in a database and manipulate that data
3. Allows to embed within other languages using SQL modules, libraries & pre compilers.
4. Allows users to create and drop databases and tables.
5. Allows users to create view, stored procedure, functions in a database.
6. Allows users to set permissions on tables, procedures and views.

A Brief History of SQL

1970 – Dr. Edgar F. "Ted" Codd of IBM is known as the father of relational databases. He described a relational model for databases.

1974– Structured Query Language (SQL) appeared.

1978– IBM worked to develop Codd's ideas and released a product named System/R.

1986– IBM developed the first prototype of relational database and standardized by ANSI. The first relational database was released by Relational Software which later came to be known as Oracle.

1987– SQL became the part of the International Organization for Standardization (ISO).

How SQL Works?

When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

There are various components included in this process. These components are –

- ☐ Query Dispatcher
- ☐ Optimization Engines
- ☐ Classic Query Engine
- ☐ SQL Query Engine, etc.

A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.

2.3 INTRODUCTION TO ANDROID



Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies. Android offers a unified Approach to application development for mobile devices which means developers need only for develop Android, and their applications should be able to run on different devices powered by Android. The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

Features of Android

1.User Interface: The user interface of the Android operating system is straight forward, and these features make it very user friendly.

2.Multiple Language Support: Android supports multiple languages in its operating system and one can change the language very easily based on one's requirement, the international languages supported are English, Germany, Chinese, Dutch, French, German, Japanese, Korean, Russian, and many more also some native language of India is also supported like Hindi, Marathi, Gujarati, Punjabi and many more.

3.Multi-tasking: Android provides support to run apps and services in the background with ease which allows the users to use multiple apps at the same time.

4.Connectivity: Android has extensive support to the connectivity and it supports connectivity such as Wi-Fi, Blue tooth, Hotspot, CDMA, GSM, NFC, VOLTE, UBB, VPN, 3G network band, and 4G Network Band.

5.Extensive Application Support: Android have Play store which is used as the major tool to download and update applications on the operating system, however, one can download the installer (often called as APK file) and install it manually, but it is not much recommended as third-party applications could be prone to some security breach in the smartphones.

3. REQUIREMENT SPECIFICATION

3.1 HARDWARE SPECIFICATION

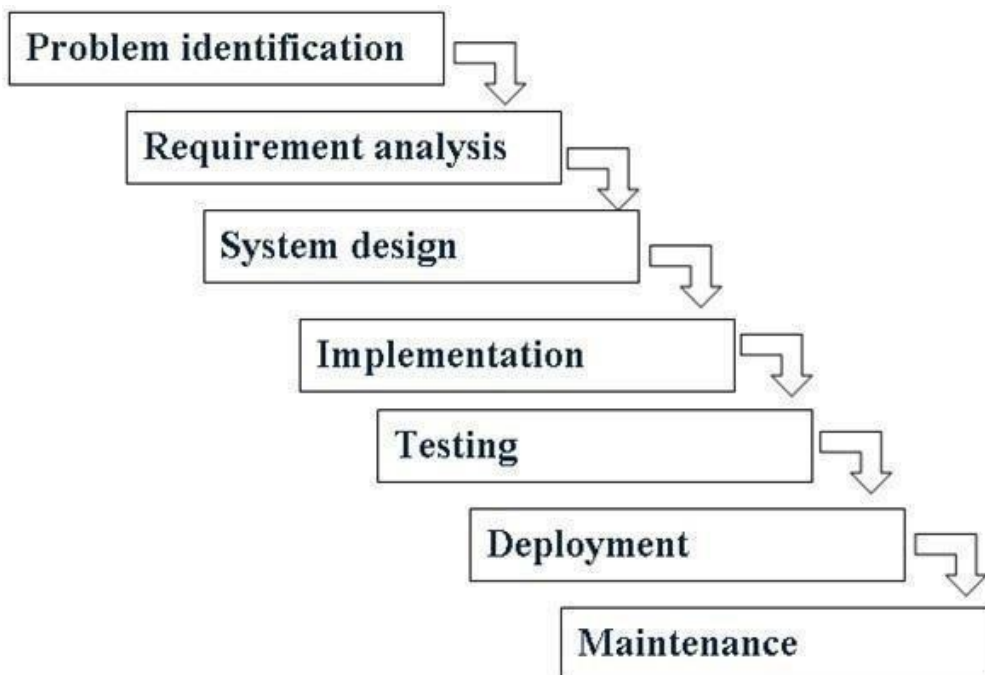
Processor	:	i3 or above
System Bus	:	32Bit or 64Bit
RAM	:	4 GB or above
HDD	:	500 GB or above
Monitor	:	14''LCD or above
Mouse	:	Any Type of Mouse

3.2 SOFTWARE SPECIFICATION

Operating System	:	Windows 10 or above
Front End	:	HTML, CSS, JAVA SCRIPT, XML
Back End	:	Python (Flask), Java (Android)
Data Base	:	MySQL
Software Used	:	Visual Studio Code, Android Studio
Web Browser	:	Internet Explorer/Google Chrome/Firefox

4. PROJECT PLANNING AND SCHEDULING

The design starts after the requirement analysis is complete and the coding begins after the design is complete. Once the programming is completed, the testing is done. In this model the sequence of activities performed in a software development project are.



It was a well-planned and well executed job. The details and the requirements for the system was collected from the client itself and all were properly tabulated. Then the next phase was to design all the outputs given by the client to well manageable design which have all the flexibility and which should withstand the current and upcoming technologies. More than one design was made and from that a well acceptable one which got all the functionalities of the user requirements was selected. And we gathered information about the front end and the back end that will suit for our job.

GANTT

Tasks	Duration	MONTH 1				MONTH 2				MONTH 3				MONTH 4		
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3
Project	4months															
System study	2weeks															
System analysis	4weeks															
Design	3weeks															
Coding and Testing	4weeks															
Implementation	1week															

5. SYSTEM ANALYSIS

System Analysis is conducted with the following objectives.

- Identify the customers need.
- Evaluate the system concept for feasibility.
- Perform economic analysis.
- Technical analysis.
- Allocate function to hardware, software, people, database and other system elements.
- Establish cost and schedule constraints.
- Create system definition that forms the foundation for all subsequent engineering work.

5.1 OVERVIEW

System analysis is concerned with studying the existing system identifying the problems and creation of the requirement specification document. In the system analysis I study about the system to be computerized and the entire problem with the present are pin pointed. The system analysis was planned and conducted in 3 stages, involving an initial investigation, feasibility and detailed analysis.

5.2 FEASIBILITY ANALYSIS

Feasibility study is the process of determination of whether or not a project is worth doing. Feasibility studies are undertaken within tight time constraints and normally culminate in a written and oral feasibility report. The contents and recommendations of this feasibility study helped us as a sound basis for deciding how to precede the project. It helped in taking decisions such as which software to use, hardware combinations, etc.

The Feasibility analysis starts with the user set of requirements. With this, the existing system is also observed. The next step is to check for the deficiencies in the existing system. By evaluating the above points a fresh idea is conceived to define and quantify the required goals. The user consent is very important for the new plan.

Besides that, a set of alternatives and their feasibility is also considered in case of any failure in the proposed system. Thus, feasibility study is an important part in software development.

To perform the feasibility study, the software engineer must first analyse the problem at global level. Indeed, the more the problem is understood, the better alternative solutions, the cost and their potential benefits for the user can be identified. Therefore, ideally, one should perform as much analysis of the problem as is needed to do a well-founded feasibility study. Since software developers cannot be sure that the offer will be accepted, they have a limited incentive for investing resource into analysing the program. On the other hand, if the study produces results that are inaccurate, it may underestimate the resource needed to develop the application and that will result in serious budget problem.

The requirements analysis phase of system and take advantages of the opportunities identified during scope definition and it satisfies the requirement identified in the requirements analysis phase of system development. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study will decide if the proposed system will be cost effective from a business point of view and if it can be developed given existing budgetary constraints. The key considerations involved in the feasibility analysis are economic, technical, behavioural and operational.

Three tests for project feasibility namely: Technical, Economical and Operational feasibilities.

5.2.1 TECHNICAL FEASIBILITY

Technical feasibility determines whether the work for the project can be done with the existing equipment, software technology and available personnel. Technical feasibility is concerned with specifying equipment and software that will satisfy the user requirement.

This project is feasible on technical remarks also, as the proposed system is more beneficiary in terms of having a sound proof system with new technical components installed on the system. The proposed system can run on any machines supporting Windows and Internet services and works on the best software and hardware

5.2.2 ECONOMICAL FEASIBILITY

Economical feasibility determines whether there are sufficient benefits in creating to make the cost acceptable, or is the cost of the system too high. As this signifies cost-benefit analysis and savings. On the behalf of the cost-benefit analysis, the proposed system is feasible and is economical regarding its pre-assumed cost for making a system.

Economical feasibility has great importance as it can outweigh other feasibilities because costs affect organization decisions. The concept of Economic Feasibility deals with the fact that a system that can be developed and will be used on installation must be profitable for the Organization. The cost to conduct a full system investigation, the cost of hardware and software, the benefits in the for of reduced expenditure are all discussed during the economic feasibility.

During the economical feasibility test we maintained the balance between the Operational and Economical feasibilities, as the two were conflicting. For example, the solution that provides the best operational impact for the end-users may also be the most expensive and, therefore, the least economically feasible.

As we know that the system development costs are usually one-time costs that will not recur after the project has been completed. For calculating the Development costs, we evaluated certain cost categories viz.

- (i) Personnel costs
- (ii) Computer usage
- (iii) Training
- (iv) Supply and equipment costs
- (v) Cost of any new computer equipment and software.

In order to test whether the Proposed System is cost-effective or not we evaluate it through three techniques viz.

- Payback analysis
- Return on Investment
- Net Present value

5.2.3 OPERATIONAL FEASIBILITY

Operation feasibility is a measure of how people feel about the system. Operational Feasibility criteria measure the urgency of the problem or the acceptability of a solution. Operational Feasibility is dependent upon determining human resources for the project.

It refers to projecting whether the system will operate and be used once it is installed. If the ultimate users are comfortable with the present system and they see no problem with its continuance, then resistance to its operation will be zero. Behaviourally also the proposed system is feasible. A particular application may be technically sound but may fail to produce the forecasted benefits, because the company is not able to get it to work. For the system, it is not necessary that the user must be a computer expert, but any computer operator given a little bit of knowledge and training can easily operate.

Our Project is operationally feasible since there is no need for special training of staff member. This project is being developed keeping in mind the general people who have very little knowledge of mobile operation, but can easily access their required database and other related information. The redundancies can be decreased to a large extent as the system will be fully automated.

6.SYSTEM DESIGN

INPUT DESIGN

The input design is the link between the computers and the users. It composes developing specification and procedure for data preparation and those steps that are necessary to put transaction data into a usable form for processing data entry. Input design is one of the most expensive phase of the development of a system. A large number of problem with a system can usually be raised due to the fault input design and method. Therefore needed to say, the input data is the life block of a system and has to be analysed and designed with the most consideration.

- The decision made during input design must help.
- To provide cost effective method of input.
- To achieve highest level of accuracy.

System analysis decide the following input design details like, what data item to input, how the data should be arranged, coded data items and transactions needing validations to detect errors and finally dialogue to guide the users in providing input. The design of input involves identifying the data needed, specifying the characteristic of each data item, capturing and for computer processing and ensuring correctness of data.

OUTPUTDESIGN

The output design refers to the result and information that are generated by the system. For many users output is the main reason for developing a system and the basis on which they evaluate the usefulness of an application. The objective of a system leads to the determination of output. The analysis of a system leads to the determination of output. Output of a system can take various forms. The most common reports are, screen displays, printed forms, graphical forms etc. the output also varies in terms of their contents, frequency, timing and format. The users of the output, its purpose and sequence of details to be printed are all considered. The output form a system is the justification for its existence. If the output is inadequate in anyway, the system itself in inadequate. The basics requirements of the output are that it should be accurate, timely and appropriate, in terms of content, medium and layout for its intended purpose. Hence it is necessary to design output so that the objective of the system is met in the best possible manner.

DATA FLOW DIAGRAM

A data flow diagram (DFD) is graphical representation of the “flow” of data through an information system. DFD can also be used for the visualization of the data processing (structured design). ADFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. Data-flow diagrams can be used to provide the end user with physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to record. How many systems are developed can be determined through a data flow diagram. The circle or bubble represents a transformation process and the label inside the bubble describes the process, using an active verb. Data flows are directed lines that identify the input data flow and output data flows at each process bubble. Data storage is represented by an open ended rectangle with a label that identifies the datastore or file. The square is labelled to identify an external entity that is a source or destination of a dataflow. Data flow diagrams are used to define the flow of the system and its resources such as information. DFD are way of expressing system requirements in a graphical manner. DFD represents one of the most ingenious tools used for the structured analysis. It has the purpose of clarifying system requirements and identifying major transformation that will become programs in the system design. It is the major point in the design phase that functionality decomposes the requirements specification down the lowest level of details.

Flow diagrams in general are usually designed using simple symbols such as a rectangle, an oval or a circle depicting a processes, data stored or an external entity, and arrows are generally used to depicts the data flow one step to another. Data flow diagrams present the logical flow of information through a system in graphical or pictorial form.

BAISIC DATAFLOW DIAGRAM SYMBOLS

DATAFLOW

Used to represent functions or process Packets of data to travel from one point to another. Data may flow from a source to process and from data store or process. An arrow line depicts the flow, with arrow head pointing in the direction of the flow. Arrow head Indicating the direction in which the data is flowing. Can have two arrow heads when a process is altering existing records in a datastores.

Process

Circle stands for processes that convert data in to information. A process represents transformation where incoming data flows. A data flow is a route, which enables are changed in to outgoing data flows. Must have at least one input and at least one output. When naming process, avoid glossing over them, without really understanding their roles.

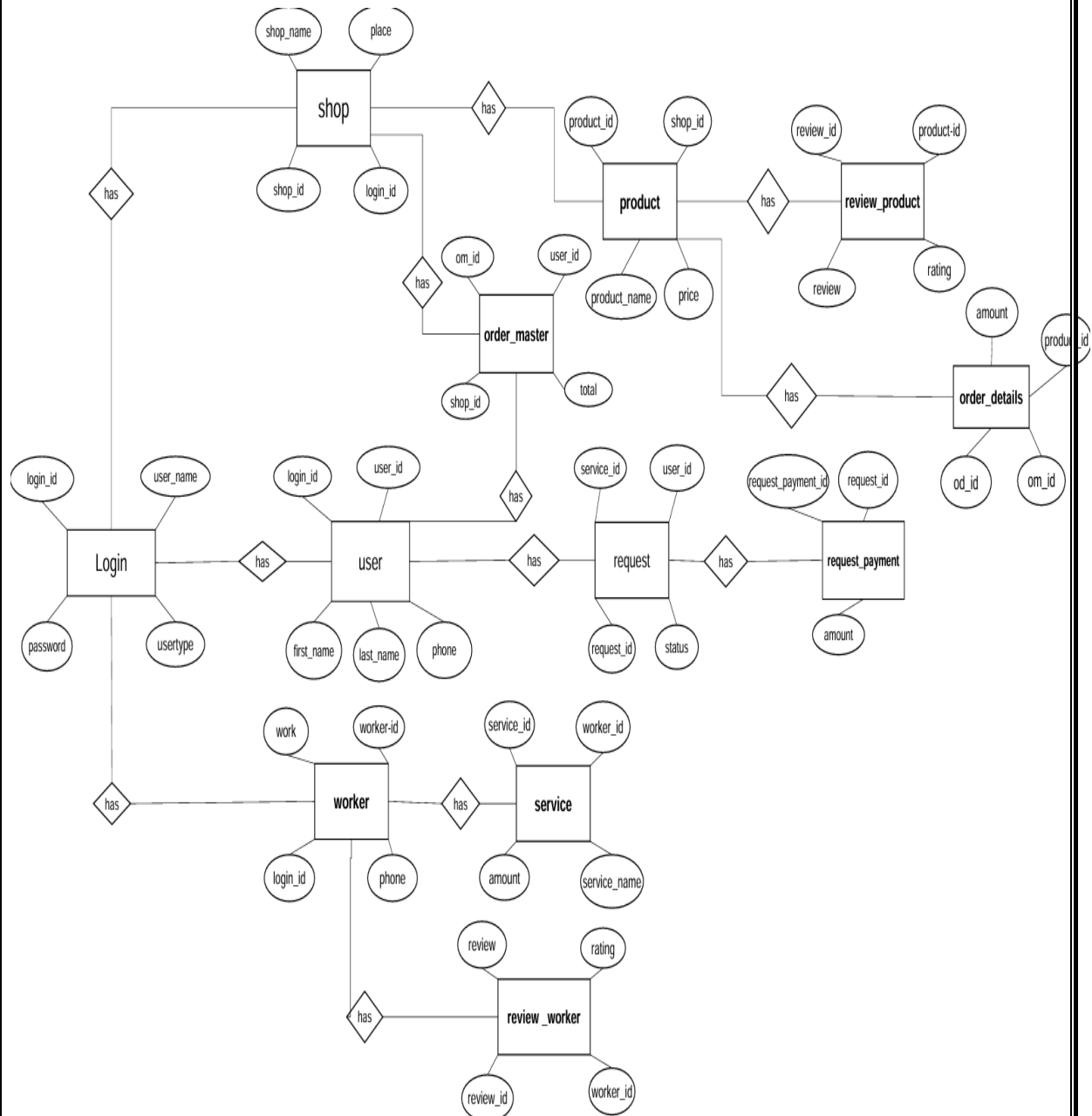
Datastore

A datastore is a repository of data that is to be stored for use by a one or more process may be as simple as buffer or queue or sophisticated as relational data base. They should have clear name. If a process merely uses the content of store and does not alter it.

Source/Destination

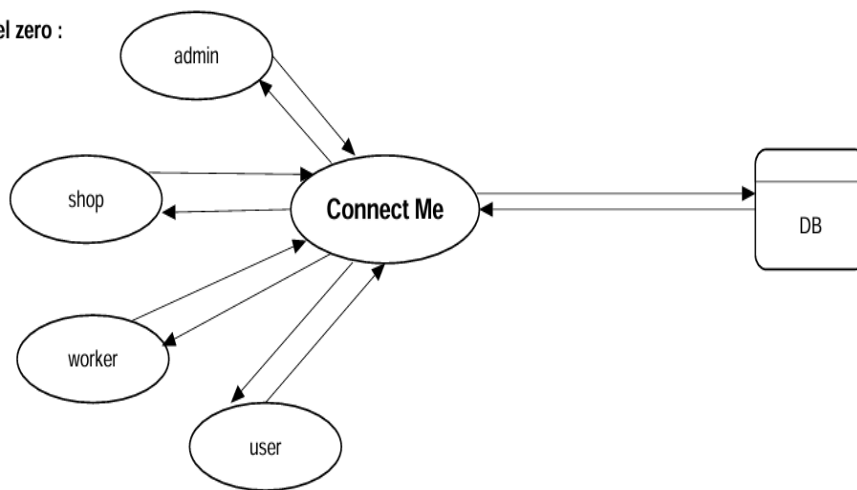
A source or sink is a person or part of an organization which enters or receives information from the system, but is considered to be outside the content of data flow model. It is normal for all the information represented with a system to have been obtained from or to be passed on to an external source or recipient. These external entities may be duplicated on a diagram, to avoid crossing data flow lines, they are duplicated a stripe is drawn across the left hand corner.

ENTITY DIAGRAM

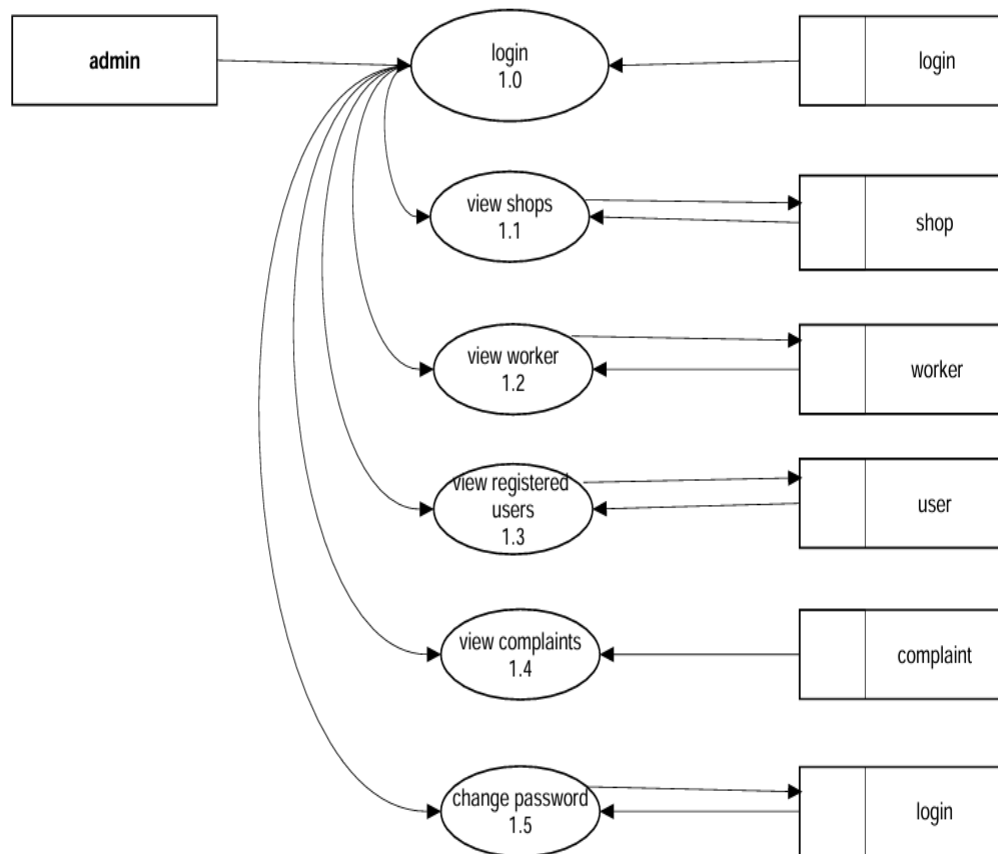


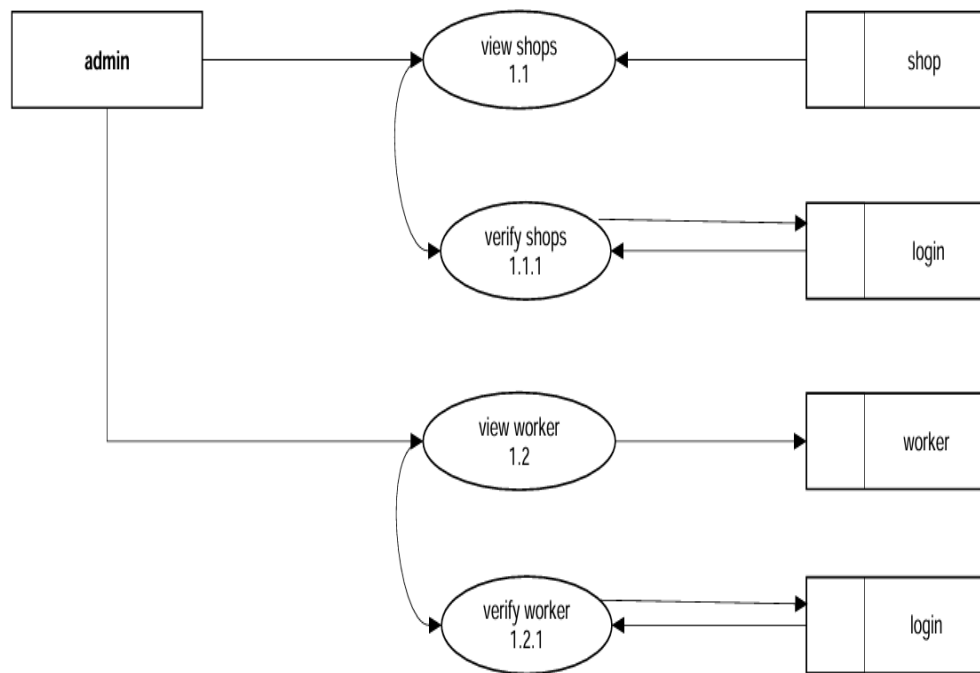
DATA FLOW DIAGRAM

Level zero :



28

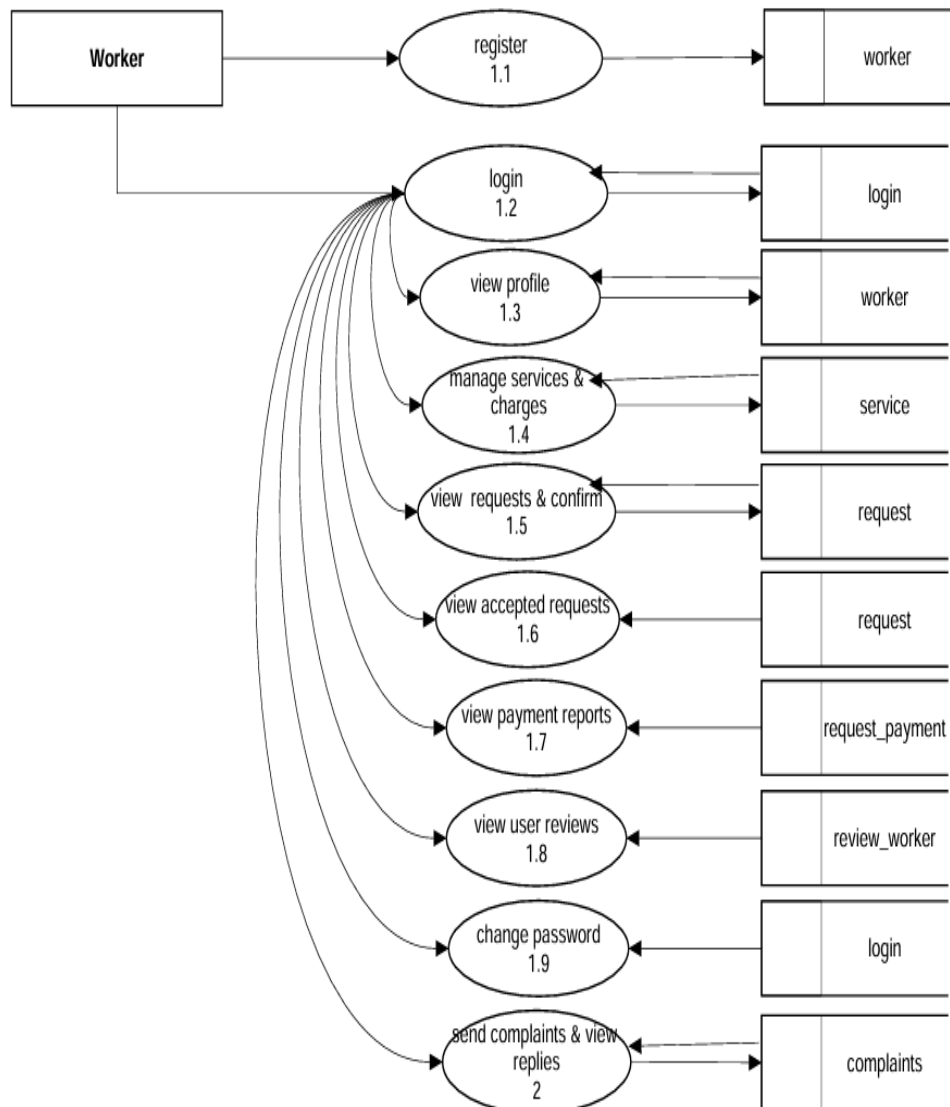
Admin Level One :

Admin Level Two :

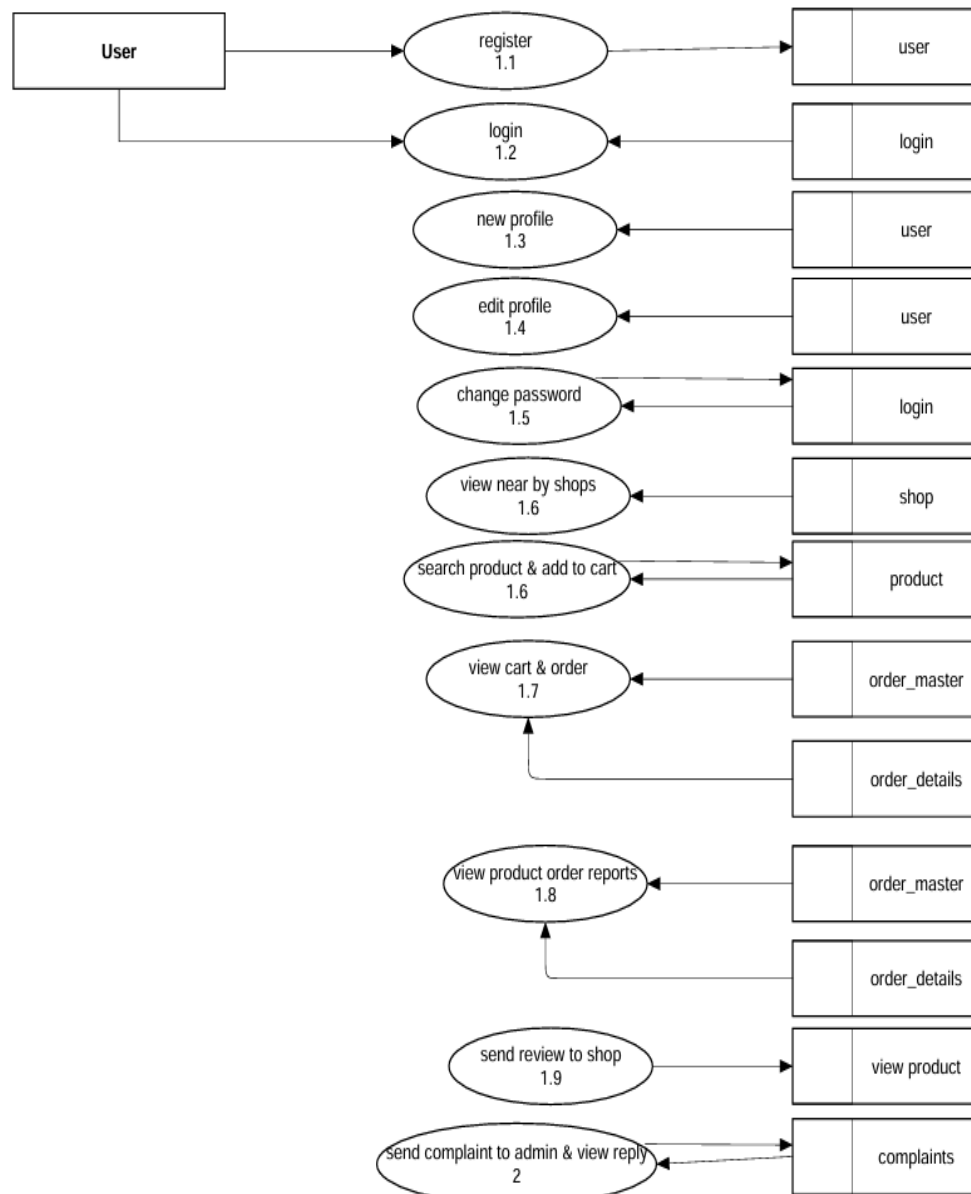
30

Shop Level One :

31

Worker Level One:

32

User Level One :

DATABASE DESIGN

A database is a collection of inter related data. A database system provides the enterprise centralized control of its operational data. In this project the data are stored at specially designed in-memory database.

In Memory Database

Tables in the database are implemented as flat files which can be accessed in random mode. Two base tables of the database are DICTNAME and DICTDICT. Dictname stores the names of all the tables in the database with a table number. DICTDICT stores table number , fieldname, field type , field length and whether a field is part of the primary key. On mounting the database , complete records of all the tables are read in to memory. Request for data is served from the memory copy and any updates are first written in the memory copy and subsequently flushed in to the physical file.

TABLE STRUCTURE

In memory data

Table 1: LOGIN

FIELD NAME	DATA TYPE	CONSTRAINTS
Login_id	int	Primary key
Username	varchar	Not null
Password	varchar	Not null
Usertype	varchar	Not null

Table 2: SHOP

FIELD NAME	DATA TYPE	CONSTRAINTS
Shop_id	int	Primary key
Login_id	int	Not null
Shop_name	varchar	Not null
Place	varchar	Not null
Pin	varchar	Not null
Email	varchar	Not null
Phone	varchar	Not null
Latitude	varchar	Not null
Longitude	varchar	Not null
type	varchar	Not null

Table 3: USER

FIELD NAME	DATA TYPE	CONSTRAINTS
User_id	int	Primary key
Login_id	int	Not null
First_name	varchar	Not null
Last_name	varchar	Not null
Place	varchar	Not null
Email	varchar	Not null
Phone	varchar	Not null
Latitude	varchar	Not null
Longitude	varchar	Not null

Table 4: WORKER

FIELD NAME	DATA TYPE	CONSTRAINTS
Worker_id	int	Primary key
Login_id	int	Not null
First_name	varchar	Not null
Last_name	varchar	Not null
Place	varchar	Not null
Phone	varchar	Not null
Email	varchar	Not null
Work	varchar	Not null
Latitude	varchar	Not null
Longitude	varchar	Not null

Table 5: ORDER_MASTER

FIELD NAME	DATA TYPE	CONSTRAINTS
Om_id	int	Primary key
User_id	int	Not null
Shop_id	int	Not null
Total	varchar	Not null
Date_time	varchar	Not null
Status	varchar	Not null

Table 6: ORDER_DETAILS

FIELD NAME	DATA TYPE	CONSTRAINTS
Od_id	int	Primary key
Om_id	int	Not null
Product_id	int	Not null
Quantity	varchar	Not null
Amount	varchar	Not null
Date_time	varchar	Not null

Table 7: ORDER_PAYMENT

FIELD NAME	DATA TYPE	CONSTRAINTS
Payment_id	int	Primary key
Om_id	int	Not null
Amount	varchar	Not null
Date_time	varchar	Not null
Status	varchar	Not null

Table 8: PRODUCT

FIELD NAME	DATA TYPE	CONSTRAINTS
Product_id	int	Primary key
Shop_id	int	Not null
Product_name	varchar	Not null
Price	varchar	Not null
Quantity	varchar	Not null
Description	varchar	Not null

Table 9: REVIEW_PRODUCT

FIELD NAME	DATA TYPE	CONSTRAINTS
Review_id	int	Primary key
Product_id	int	Not null
Review	varchar	Not null
Rating	varchar	Not null

Table 10: REVIEW_WORKER

FIELD NAME	DATA TYPE	CONSTRAINTS
Review_id	int	Primary key
Worker_id	int	Not null
Review	varchar	Not null
Rating	varchar	Not null

Table 11: SERVICE

FIELD NAME	DATA TYPE	CONSTRAINTS
Service_id	int	Primary key
Worker_id	int	Not null
Service_name	varchar	Not null
Description	varchar	Not null
Amount	varchar	Not null

Table 12: REQUEST

FIELD NAME	DATA TYPE	CONSTRAINTS
Request_id	int	Primary key
Service_id	int	Not null
User_id	int	Not null
Title	varchar	Not null
Amount	varchar	Not null
Status	varchar	Not null
Date	varchar	Not null

Table 13: REQUEST_PAYMENT

FIELD NAME	DATA TYPE	CONSTRAINTS
Request_Payment_id	int	Primary key
Request_id	int	Not null
Amount	varchar	Not null
Date_time	varchar	Not null
Status	varchar	Not null

7. SYSTEM TESTING AND IMPLEMENTATION

1 SYSTEM TESTING

Software testing is defined as the process by which one detects the defects in the software. It is considered as the final opportunity for covert/rectify and us to detect any defects that were in the software. Testing is a process which is done with the explicit intention of finding errors that make the program fail. In short, system and quality assurance is a review of the software products and related documentation for completion, correctness, reliability and maintainability. The first step in system testing is to prepare a plan that will test all aspects of the system. System testing is an expensive, but critical process that may take as much as 50% of the budget for program development.

Testing performs a very critical role for quality assurance and for ensuring their liability of software. In a software development project, errors can be injected at any stage during development. Testing is the phase where the errors remaining from the earlier phases also must be detected. The common view of testing is that it is performed to prove that there are no errors in the program. But this is quite difficult since the analyst cannot prove that software is free from all sorts of errors. Therefore, the most useful and practical approach is with the understanding that testing is the process of executing program with the explicit intension of finding errors that is to make the program fail. A successful test can be therefore, one that finds an error. The philosophy behind testing is finding errors. Test cases are devised with this purpose in mind. A test case is a set of data that the system will process as normal input. However, the data is created with the express intent of determining whether the system will process it correctly. Each test case is designed with the intent of finding errors in the way the system will process it. There are two general strategies for testing software, code testing and specification testing.

Code Testing

The code testing strategy examines the logic of the program. To follow this testing method, the analyst develops test cases that result in executing every instruction in the program or module, that is, path through the program is tested. A path is

specific combination of conditions that is handled by the program. On the surface, code testing seems to be an ideal method for testing software. However, even if code testing can be performed in its entirety, it does not guarantee against software failures. This testing strategy does not indicate whether the code meets its specification nor does it determine whether all aspects are even implemented. Code testing also does not check the range of data that the program will accept, even though, when software failures occur in actual case, it is frequently because users submitted data outside of expected ranges.

Specification Testing

To perform specification testing, the analyst examines the specifications stating what the program should do, and how it should perform under various conditions. Then test cases are developed for each condition or combination of conditions and submitted for processing. By examining the results, the analyst can determine whether the program performs according to its specified requirements. Software testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before the live operation commences. It is the critical element of software quality assurance and ultimate review of specification, design and coding. The debugging process is the most unpredictable part of the testing procedure.

Syntax Testing

We use syntax testing to eliminate errors in the software. In the system, we have input fields like text, numeric fields. We should allow only numeric fields.

Unit Testing

Unit testing comprises the set of tests performed by an individual programmer prior to integration of the unit into a layer system. These are four categories of tests that can be performed on a program unit.

-Functional Unit

-Performance Unit

-Stress Unit

-Structure Unit

Integration Testing

Integration testing involves bottom-up integration, top-down integration and sandwich integration strategy. Bottom-up integration starts with the traditional strategy used to integrate the components. Top-down integration starts with the main routine and one or two immediate subroutines in the system structure. Sandwich integration is predominantly top-down, bottom-up techniques are used in some modules and systems.

Acceptance Testing

Acceptance testing involves planning and execution of functional tests, performance tests and stress tests in order to demonstrate the implemented system satisfies its requirements. Functional test cases involve exercising the code with nominal files for which the expected outputs are known. Thus, the software system developed in the above manner is one that satisfies the user needs, confirms to its requirements and the design satisfactions and exhibits an absence of errors.

Validation Testing

Validation refers to the process of using the new software for the developed system live environment. i.e. new software inside the organization, in order to find out the error the validation phase reveals the failure and the bugs in the developed system. It will come to know about the practical difficulties the system focus. When operated in the true environment by testing the code of the implemented software, the logic of the program can be examined. A specifications test is conducted to check whether and specifications starting the program are performing under various conditions. Apart from these tests are some special tests conducted which are given below:

Peak load test: This determines whether the new system will handle the volume of activities demand. The test has revealed that, the new software for the college is capable of handling demands at the peak time.

Storage testing: This determines the capacity of the new system to store transaction data on a disk or on other files. The proposed software has the required storage space available, because of the use of a number of hard disks.

Performance time testing: This test determines the length of the time used by the system to process transaction data.

White Box Testing

White box testing is a test case design method that causes the control structure of the procedural design to derive test cases. Using white box testing methods, the software engineer can derive test cases that

- a. All independent paths within a module have been exercised at least once.
- b. Exercise all logical decisions on their true and false sides.
- c. Execute all loops at their boundaries and within their Operational bounds
- d. Exercise internal data structures to ensure their validity.

Black Box Testing Black box testing methods focus on the functional requirement of the software. That is, Black box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black box testing attempts to find errors in the following categories:

- a. Incorrect or missing functions
- b. Interface errors
- c. Errors in data structures or external database access.
- d. Performance errors and
- e. Initialization and termination errors

Black box testing is not an alternative to White box testing. Rather it is complimentary approach that is likely to uncover a different class of errors than white box methods. Unlike white box testing, that is performed early in the testing process. Black box testing to be applied during later stages of testing.

7.2 SYSTEM IMPLEMENTATION

Implementation is a process of ensuring that the information system is operational. Implementation allows the users to take over its operation for use and evaluation. It involves training the users to handle the system and plan for a smooth conversion. The personnel in the system must know in detail what their roles will be, how they can use the system, and what the system will or will not do. The success or failure of well-designed and technically elegant systems can depend on the way they are operated and used.

Systems operators must be trained properly such that they can handle all possible operations, both routine and extraordinary. The operators should be trained in what common malfunctions may occur, how to recognize them, and what steps to take when they come. Training involves creating trouble shooting lists to identify possible problems and remedies for them, as well as the names and telephone numbers of individuals to contact when unexpected or unusual problems arise. Training also involves familiarization with run procedures, which involves working through the sequence of activities needed to use a new system.

7.3 SYSTEM MAINTENANCE

Software is always evolving and it is never finished as long as it is used; partly to accommodate for the ever-changing world we live in. The evolution of your software might be motivated by a variety of reasons; to keep the software up and running, to upgrade to the latest release, enhance features or to rework the system for future maintainability. No matter the motivation, software change is vital for the evolution and success of it. Therefore, software will have to undergo changes, and understanding the different types of changes your software can go through is important to realize that software maintenance is more than just bug fixing. Infact, a study suggests that over 80% of software change is attributed to non-bug related changes,

There are four categories of software maintenance:

- Corrective maintenance
- Adaptive maintenance

- Perfective maintenance
- Preventive maintenance

CORRECTIVE MAINTENANCE

Corrective change, most commonly referred to as “bugs,” is the most typical change associated with maintenance work. Corrective changes address errors and faults in your software that could affect various areas of your software; design, logic or code. Most commonly, these changes are sprung by bug reports created by users. It is important to note that sometimes problem reports submitted by users are actually enhancements of the system not bugs.

ADAPTIVE MAINTENANCE

Adaptive change is triggered by changes in the environment your software lives in. An adaptive change can be triggered by changes to the operating system, hardware, software dependencies and even organizational business rules and policies. These modifications to the environment can trigger changes within other parts of your software. For example, updating the server, compilers, etc or modifications to shipping carriers and payment processors can affect functionality in your software.

PERFECTIVE MAINTENANCE

Perfective changes refer to the evolution of requirements and features in your existing system. As your software gets exposed to users, they will think of different ways to expand the system or suggest new features that they would like to see as part of the software, which in turn can become future enhancements to the system. Perfective changes also include removing features from a system that are not effective and functional to the end goal of

the system. Surprisingly, 50-55% of most maintenance work is attributed to perfective changes.

PREVENTIVE MAINTENANCE

Preventive changes refer to changes made to increase the understanding and maintainability of your software in the long run. Preventive changes are focused in decreasing the deterioration of your software in the long run. Restructuring, optimizing code and updating documentation are common preventive changes. Executing preventive changes reduces the amount of unpredictable effects a software can have in the long term and helps it become scalable, stable, understandable and maintainable.

8. FUTURE ENHANCEMENT

1. AI Recommendations – Suggest services and products based on user preferences.
2. In-App Chat & Video Calls – Enable real-time messaging and consultations.
3. Premium Subscriptions – Offer priority bookings and exclusive deals.
4. Multilingual Support – Expand accessibility with multiple languages.
5. AI Chatbot Support – Provide 24/7 automated assistance.
6. Secure Blockchain Transactions – Ensure transparency and security.

9. CONCLUSION

‘Connect Me’ is a comprehensive solution designed to bridge the gap between users, local workers, shops, and pharmacies. By providing a seamless platform for service booking and product ordering, it enhances accessibility, efficiency, and security. With its user-friendly interface and robust backend, the application ensures smooth interactions and reliable transactions. As the platform evolves with future enhancements, it has the potential to become an essential tool for local commerce and daily services, improving convenience for both users and service providers.

10. APPENDIX

CONNECT ME

Public.py

```
from flask import *
from database import *
public=Blueprint('public',_name_)

@public.route('/')
def home():
    return render_template("home.html")

@public.route('/login',methods=['get','post'])
def login():
    if 'login' in request.form:
        username=request.form['username']
        passwd=request.form['password']
        qry="select * from login where user_name='%s' and password='%s'"%(username,passwd)
        res=select(qry)

    if res:
        session['login_id']=res[0]['login_id']
        print(session['login_id'])

        if res[0]['user_type']=='admin':
            return redirect(url_for('admin.admin_home'))
```

```
if res[0]['user_type']=='user':
    return redirect(url_for('user.user_home'))
if res[0]['user_type']=='shop':
    qrt="select * from shop where login_id='%s'"%(session['login_id'])
    ress=select(qrt)
    if ress:
        session['shop_id']=ress[0]['shop_id']
        return redirect(url_for('shop.shop_home'))
if res[0]['user_type']=='worker':
    qrt="select * from worker where login_id='%s'"%(session['login_id'])
    ress=select(qrt)
    if ress:
        session['worker_id']=ress[0]['worker_id']
        return redirect(url_for('worker.worker_home'))
return render_template("login.html")

@public.route('/shop_register',methods=['get','post'])
def shop_register():
    if 'register' in request.form:

        shopname=request.form['shop name']
        place=request.form['place']
        pin=request.form['pin']
        email=request.form['email']
        phone=request.form['phone']
        lat=request.form['latitude']
        long=request.form['longitude']
        username=request.form['username']
        passw=request.form['password']

    qry="insert into login values(null,'%s','%s','shop')"%(username,passw)
```

```
res=insert(qry)

qry1="insert into shop
values(null,'%s','%s','%s','%s','%s','%s','%s','%s','shop')"%(res,shopname,place,pin,email,phone,lat,long)
insert(qry1)

return render_template("shop_register.html")

@public.route('/worker_register',methods=['get','post'])
def worker_register():
    if 'register' in request.form:

first=request.form['first_name']
    last=request.form['last_name']
    place=request.form['place']
    phone=request.form['phone']
    email=request.form['email']
    work=request.form['work']
    lat=request.form['latitude']
    long=request.form['longitude']
    username=request.form['username']
    passw=request.form['password']

    qry="insert into login values(null,'%s','%s','worker')"%(username,passw)
    res=insert(qry)

    qry1="insert into worker
values(null,'%s','%s','%s','%s','%s','%s','%s','%s','%s')"%(res,first,last,place,phone,email,work,
lat,long)
    insert(qry1)
    return render_template("worker_register.html")
```

Admin.py

```
from flask import *
from database import *
admin=Blueprint('admin',_name_)
@admin.route('/admin_home')
def admin_home():
    return render_template('admin_home.html')

@admin.route("/admin_view_new_shops")
def admin_view_new_shops():
    data={ }
    qry="SELECT * FROM shop INNER JOIN login USING(login_id)"
    res=select(qry)
    data['view']=res

    if 'action' in request.args:
        action=request.args['action']
        id=request.args['log_id']

    if action=='verify':
        qry2="update login set user_type='shop' where login_id='%s'"%(id)
        res2=update(qry2)
        if res2:
            return "<script>alert('verified
success');window.location='/admin_view_new_shops'</script>"

        if action=='reject':
            qry3="update login set user_type='reject' where login_id='%s'"%(id)
            res3=update(qry3)
            if res3:
```



```
return "<script>alert('rejected
success');window.location='/admin_view_new_shops'</script>"

print(action,id,"++++++++++++++++++++")

return render_template('admin_view_new_shops.html',data=data)

@admin.route("/admin_view_new_worker")
def admin_view_new_worker():
    data={ }
    qry="SELECT * FROM worker INNER JOIN login USING(login_id)"
    res=select(qry)
    data['view']=res

if 'action' in request.args:
    action=request.args['action']
    id=request.args['log_id']

    if action=='verify':
        qry2="update login set user_type='worker' where login_id='%s'"%(id)
        res2=update(qry2)
        if res2:
            return "<script>alert('verified
successfully');window.location='/admin_view_new_worker'</script>"

    if action=='reject':
        qry3="update login set user_type='reject' where login_id='%s'"%(id)
        res3=update(qry3)
        if res3:
            return "<script>alert('rejected
successfully');window.location='/admin_view_new_worker'</script>"
```

```
return render_template('admin_view_new_worker.html',data=data)

@admin.route("/view_complaints")
def view_complaints():
    data={ }
    qry="select * from complaints"
    res=select(qry)
    data['view']=res
    return render_template('view_complaints.html',data=data)

@admin.route("/admin_send_reply",methods=['get','post'])
def admin_send_reply():
    id=request.args['id']

    if 'complt' in request.form:
        reply=request.form['comp']

        qry="update complaints set reply='%s' where complaint_id='%s' "%(reply,id)
        update(qry)

    return render_template('admin_send_reply.html')

@admin.route('/contact')
def contact():
    return render_template('contact.html')
```

main.py

```
from flask import *
from public import public
```

```
from admin import admin
from user import user
from shop import shop
from worker import worker
```

```
app=Flask(__name__)
```

```
app.secret_key="abcde"
```

```
app.register_blueprint(public)
app.register_blueprint(admin)
app.register_blueprint(user)
app.register_blueprint(shop)
app.register_blueprint(worker)
```

```
app.run(debug=True,host="0.0.0.0",port=5007)
```

shop.py

```
import uuid
from flask import *
from database import *

shop=Blueprint('shop',__name__)
```

```
@shop.route('/shop_home')
def shop_home():
    return render_template('shop_home.html')
```

```
@shop.route('/view_shop_profile')
def view_shop_profile():
    data={ }
    qry="SELECT * FROM shop where shop_id='%s'"%(session['shop_id'])
    res=select(qry)
    data['view']=res
    return render_template('view_shop_profile.html',data=data)

@shop.route('/product_management',methods=['get','post'])
def product_management():
    if 'submit' in request.form:

        pname=request.form['product name']
        price=request.form['price']
        qty=request.form['quantity']
        image=request.files['image']
        path="static/product/"+str(uuid.uuid4())+image.filename
        image.save(path)
        descr=request.form['description']

        qry1="insert into product
values(null,'%s','%s','%s','%s','%s','%s')"%(session['shop_id'],pname,price,qty,path,descr)
        insert(qry1)
        data={ }
        qry="SELECT * FROM product where shop_id='%s'"%(session['shop_id'])
        res=select(qry)
        data['view']=res
        if 'action' in request.args:
            value=request.args['action']
            id=request.args['product_id']
        if value=='update':
            print("sdfaa")
```

61

```

a="select * from product where product_id='%s'%(id)
b=select(a)
if b:
    data['up']=b
    if 'update' in request.form:
        pname=request.form['product name']
        price=request.form['price']
        qty=request.form['quantity']
        image=request.form['image']
        descr=request.form['description']
        d="update product set
product_name='%s',price='%s',quantity='%s',image='%s',description='%s' where
product_id='%s'"%(pname,price,qty,image,descr,id)
        e=update(d)
        if e:
            return "<script>alert('updated
successfully');window.location='/product_management'</script>"

if value=='delete':
    qry2="delete from product where product_id='%s'%(id)
    res2=delete(qry2)
    if res2:
        return "<script>alert('deleted
successfully');window.location='/product_management'</script>"

return render_template("product_management.html",data=data)

@shop.route('/view_order')
def view_order():
    data={}

    qry="SELECT * FROM order_details INNER JOIN order_master USING(om_id) INNER
JOIN product USING(product_id) INNER JOIN user USING(user_id) "
    res=select(qry)

```

```
data['view']=res
return render_template('view_order.html',data=data)

@shop.route('/view_payment')
def view_payment():
    data={}
    id=request.args['id']
    qry="SELECT *,order_payment.amount AS total FROM order_payment INNER JOIN
order_details USING(om_id) WHERE om_id='%s'"%(id)
    res=select(qry)
    data['view']=res
    return render_template('view_payment.html',data=data)

@shop.route("/shop_complaint",methods=['get','post'])
def shop_complaint():
    data={}
    if 'submit' in request.form:
        comp=request.form['comp']

        qry1="insert into complaints
values(null,'%s',curdate(),'pending','%s')"%(session['shop_id'],comp)
        insert(qry1)

        qry="select * from complaints where sender_id='%s'"%(session['shop_id'])
        res=select(qry)

        print(res,'////////////////////////////////////')
        if res:
            data['view']=res
            return render_template('shop_complaint.html',data=data)
```

```
@shop.route('/view_reviews')
def view_reviews():
    data={ }
    qry="select * from review_product"
    res=select(qry)
    data['view']=res
    return render_template('view_reviews.html',data=data)
```

Worker.py

```
from flask import *
from database import *

worker=Blueprint('worker',_name_)

@worker.route('/worker_home')
def worker_home():
    return render_template('worker_home.html')

@worker.route('/view_worker_profile')
def view_worker_profile():
    data={ }
    qry="SELECT * FROM worker where worker_id='%s'"%(session['worker_id'])
    res=select(qry)
    data['view']=res
    return render_template('view_worker_profile.html',data=data)

@worker.route('/manage_services_charges',methods=['get','post'])
def manage_services_charges():

    if 'services' in request.form:
        sname=request.form['service_name']
        desc=request.form['description']
        amt=request.form['amount']

    qry1t="insert into service
values(null,'%s','%s','%s','%s')"%(session['worker_id'],sname,desc,amt)

    insert(qry1t)
```



```
data={ }
qry="SELECT * FROM service where worker_id='%s'"%(session['worker_id'])
res=select(qry)
data['view']=res

if 'action' in request.args:
    value=request.args['action']
    id=request.args['id']

else:
    value=None

if value=='delete':
    qry2="delete from service where service_id='%s'"%(id)
    res2=delete(qry2)
    return "<script>alert('deleted successfully');window.location='/manage_services_charges'</script>"

if value=='update':
    print("sdfaa")
    a="select * from service where service_id='%s'"%(id)
    b=select(a)
    data['up']=b

if 'update' in request.form:
    sname=request.form['service_name']
    descp=request.form['description']
    amt=request.form['amount']
    d="update service set service_name='%s',description='%s',amount='%s' where service_id='%s'"%(sname,descp,amt,id)
    e=update(d)
```

66

```
return "<script>alert('updated
successfully');window.location='/manage_services_charges'</script>"
```

```
    return render_template('manage_services_charges.html',data=data)
@worker.route('/view_user_reviews')
def view_user_reviews():
    data={ }
    qry="select * from review_worker"
    res=select(qry)
    data['view']=res
    return render_template('view_user_reviews.html',data=data)
```

```
@worker.route("/worker_complaint",methods=['get','post'])
def worker_complaint():
    data={ }
    if 'submit' in request.form:
        comp=request.form['comp']

        qry1="insert into complaints
values(null,'%s',curdate(),'pending','%s')"%(session['worker_id'],comp)
        insert(qry1)

    return "<script>alert('Add successfully');window.location='/worker_complaint'</script>"
```

```
qry="select * from complaints where sender_id='%s'"%(session['worker_id'])
res=select(qry)
print(res)
if res:
    data['view']=res
    return render_template('worker_complaint.html',data=data)
```

```
@worker.route('/view_user_request',methods=['get','post'])
```

61

```

def view_user_request():
    data={ }
    qry1="SELECT * FROM request left JOIN service USING(service_id) where
worker_id='%s' and status='pending'"%(session['worker_id'])
    res=select(qry1)
    data['view']=res
    if 'action' in request.args:
        value=request.args['action']
        id=request.args['id']
    else:
        value=None

    if value=='accept':
        qry="update request set status='accepted' where request_id='%s'"%(id)
        update(qry)
        return "<script>alert('request accepted');window.location='/view_user_request'</script>"
    if value=='reject':
        qry="update request set status='rejected' where request_id='%s'"%(id)
        update(qry)
        return "<script>alert('request rejected');window.location='/view_user_request'</script>"

    data1={ }
    qry2="SELECT * FROM request left JOIN service USING(service_id) where
worker_id='%s' and status='accepted'"%(session['worker_id'])
    res2=select(qry2)
    data1['accepted']=res2
    return render_template("view_user_request.html",data=data,data1=data1)

@worker.route('/view_payment_report',methods=['get','post'])
def view_payment_report():

```

```
data={ }

qry1="SELECT *,request_payment.status as pay_status,request_payment.amount as
total_amount FROM request_payment INNER JOIN request using(request_id) INNER JOIN
service USING(service_id) where worker_id='%s'"%(session['worker_id'])

res=select(qry1)
data['view']=res

return render_template("view_payment_report.html",data=data)
```

user.py

```
from flask import *
from database import *

user=Blueprint('user',_name_)

@user.route('/user_login')
def user_login():
    data={ }

    uname=request.args['u']
    psw=request.args['p']

    qry="select * from login inner join user using(login_id) where user_name='%s' and
password='%s'"%(uname,psw)

    res=select(qry)
    if res:
        data['status']="success"
        data['data']=res
    else:
        data['status']="failed"
```

```
return str(data)

@user.route('/user_register')
def user_register():
    data={ }
    fname=request.args['f']
    lname=request.args['l']
    place=request.args['pl']
    email=request.args['em']
    phone=request.args['ph']
    uname=request.args['u']
    psw=request.args['p']
    lati=request.args['lati']
    longi=request.args['logi']

    print(lati,longi,"//////////////////////////////////////=====")

    a="select * from login where user_name='%s'"%(uname)
    b=select(a)

    if b:
        data['status']='username'

    else:

        qry="insert into login values(null,'%s','%s','user')"%(uname,psw)
        id=insert(qry)
        qry1="insert into user
values(null,'%s','%s','%s','%s','%s','%s','%s')"%(id,fname,lname,place,email,phone,lati,lo
ngi)
        c=insert(qry1)
```

```
if c:
    data['status']='success'

else:
    data['status']='failed'

return str(data)
```

```
@user.route('/user_shop')
def user_shop():
    # data={ }

    # qry="select * from shop "
    # res=select(qry)
    # print(res,"////////////////")
    # if res:
    #     data['status']="success"
    #     data['data']=res
    # else:
    #     data['status']="failed"

data={ }

lati=request.args['lati']
logi=request.args['logi']

print(lati,"_")
```

```
print(logi, "_")

q="""SELECT *,
      (3959 * ACOS(
          COS(RADIANS(%s)) * COS(RADIANS(latitude)) * COS(RADIANS(longitude) -
RADIANS(%s)) +
          SIN(RADIANS(%s)) * SIN(RADIANS(latitude))
      )) AS user_distance
FROM shop
HAVING user_distance < 11.068
ORDER BY user_distance

"""% (lati, logi, lati)
res=select(q)
data['view']=res

print(res, "-")
if res:
    data['status']='success'
    data['data']=res
else:
    data['status']='failed'
data['method']='view'

return str(data)
```

11. BIBLIOGRAPHY

References

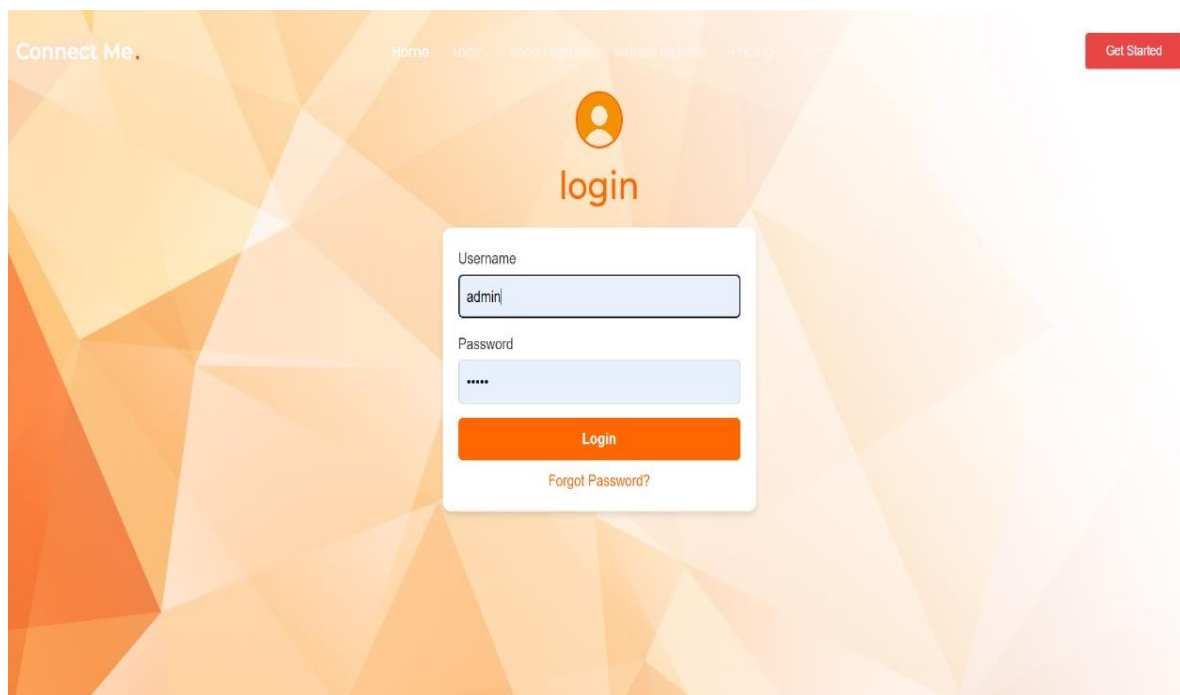
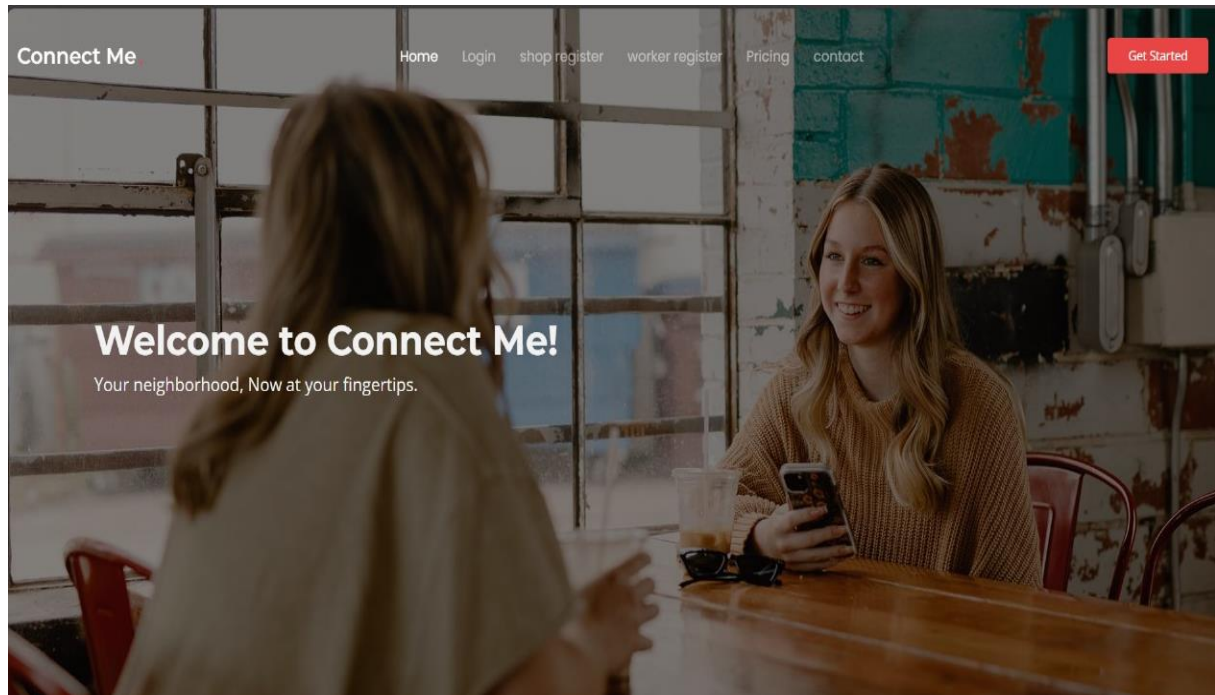
"Local Services" on GitHub: A collection of open-source projects and code samples for local service discovery.

URL

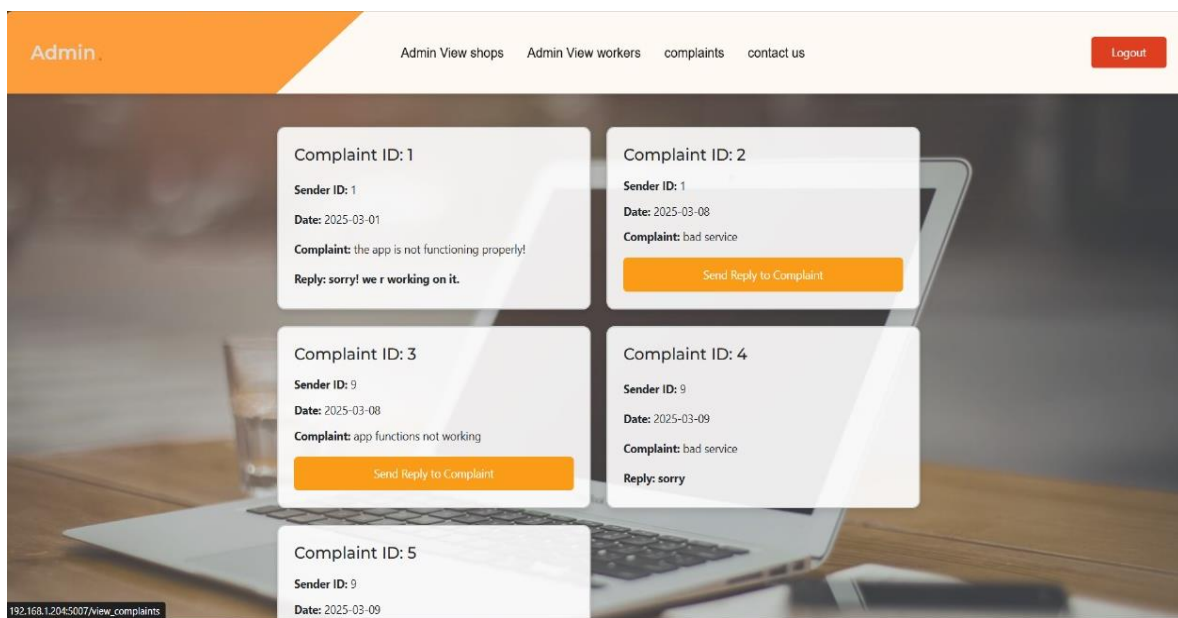
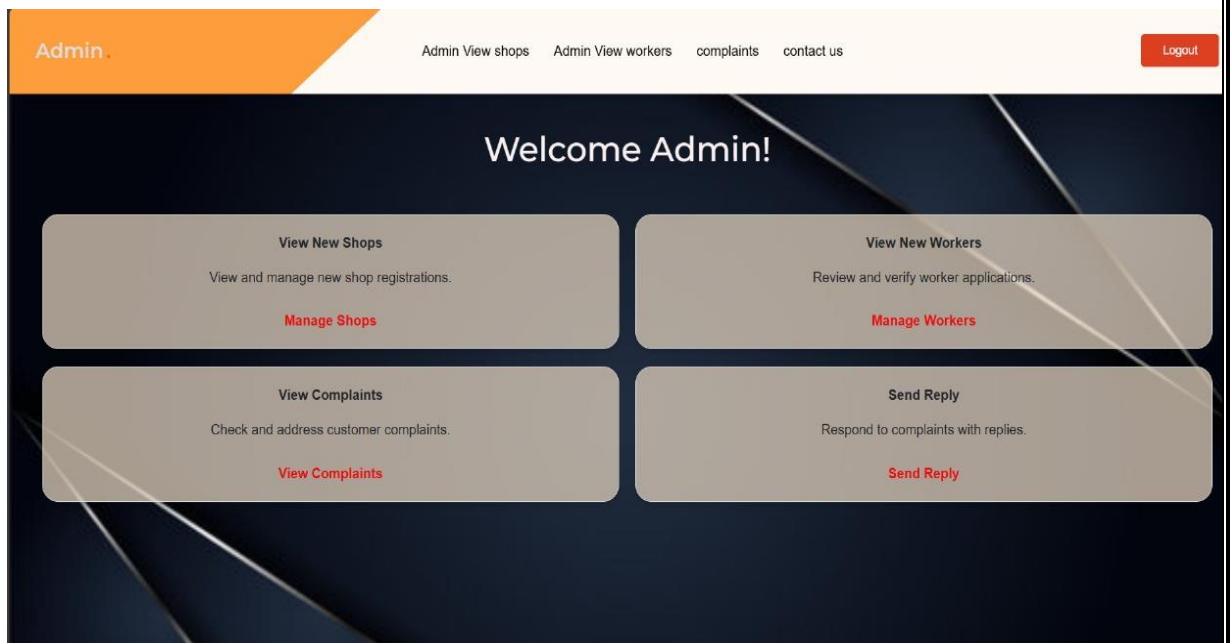
https://github.com/angrycoding/js-local-service-discovery?utm_source=chatgpt.com

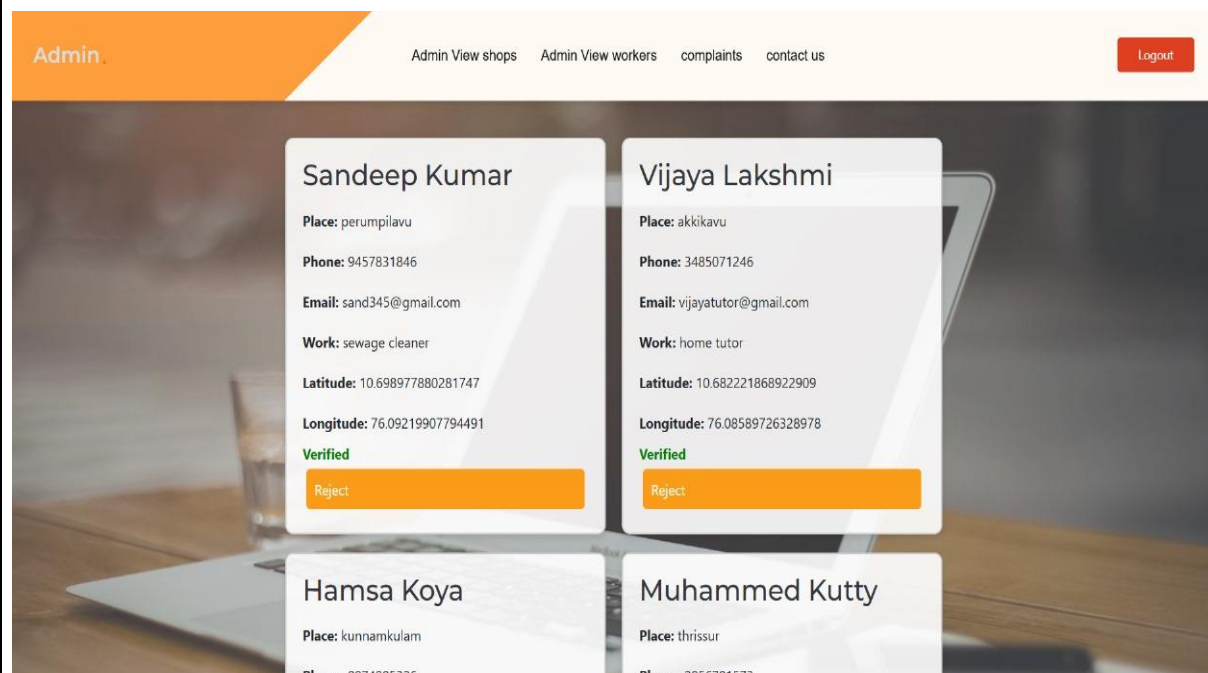
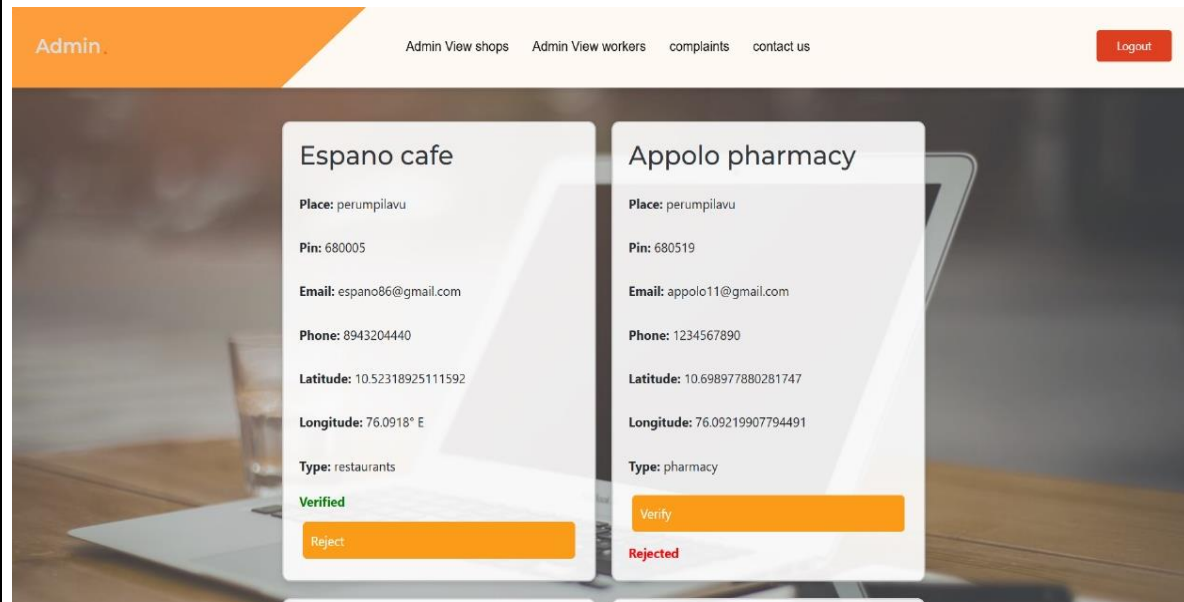
12.SCREENSHORT

Admin interface



75





How can we help you today?



SALES
sales@example.com
Call us: +91-91-7076-3758
(8 AM to 8 PM IST, Monday to Saturday)



SUPPORT
support@example.com
Call us: +91-91-7076-3758
(8 AM to 8 PM IST, Monday to Saturday)

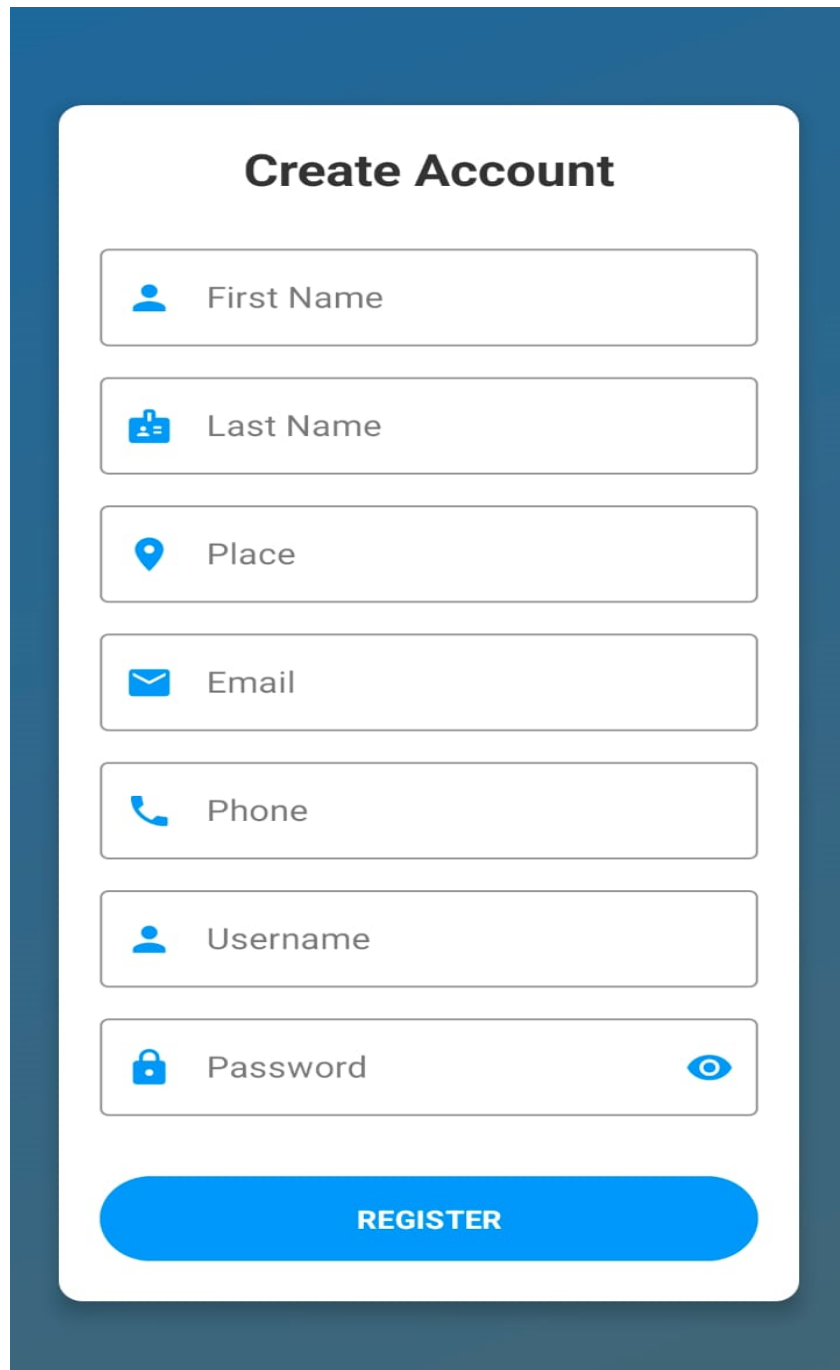


BILLING
billing@example.com
Call us: +91-91-7076-3758
(8 AM to 8 PM IST, Monday to Saturday)




COMPLAINTS
[Submit Complaint for Abuse Activity](#)
[Submit Complaint for False Whois](#)
[Report Spam](#)


User interface





The image displays a 'Create Account' form within a dark blue container. The form itself is a white rounded rectangle with a blue shadow. It features a title 'Create Account' at the top. Below the title are seven input fields, each with a blue icon on the left and a label: 'First Name' (person icon), 'Last Name' (ID card icon), 'Place' (location pin icon), 'Email' (envelope icon), 'Phone' (phone handset icon), 'Username' (person icon), and 'Password' (lock icon). The 'Password' field includes a blue eye icon on the right for toggling visibility. At the bottom of the form is a large, rounded blue button with the text 'REGISTER' in white capital letters.


Create Account


 First Name



 Last Name

 Place


 Email

 Phone


 Username



 Password 

REGISTER



Welcome Back!

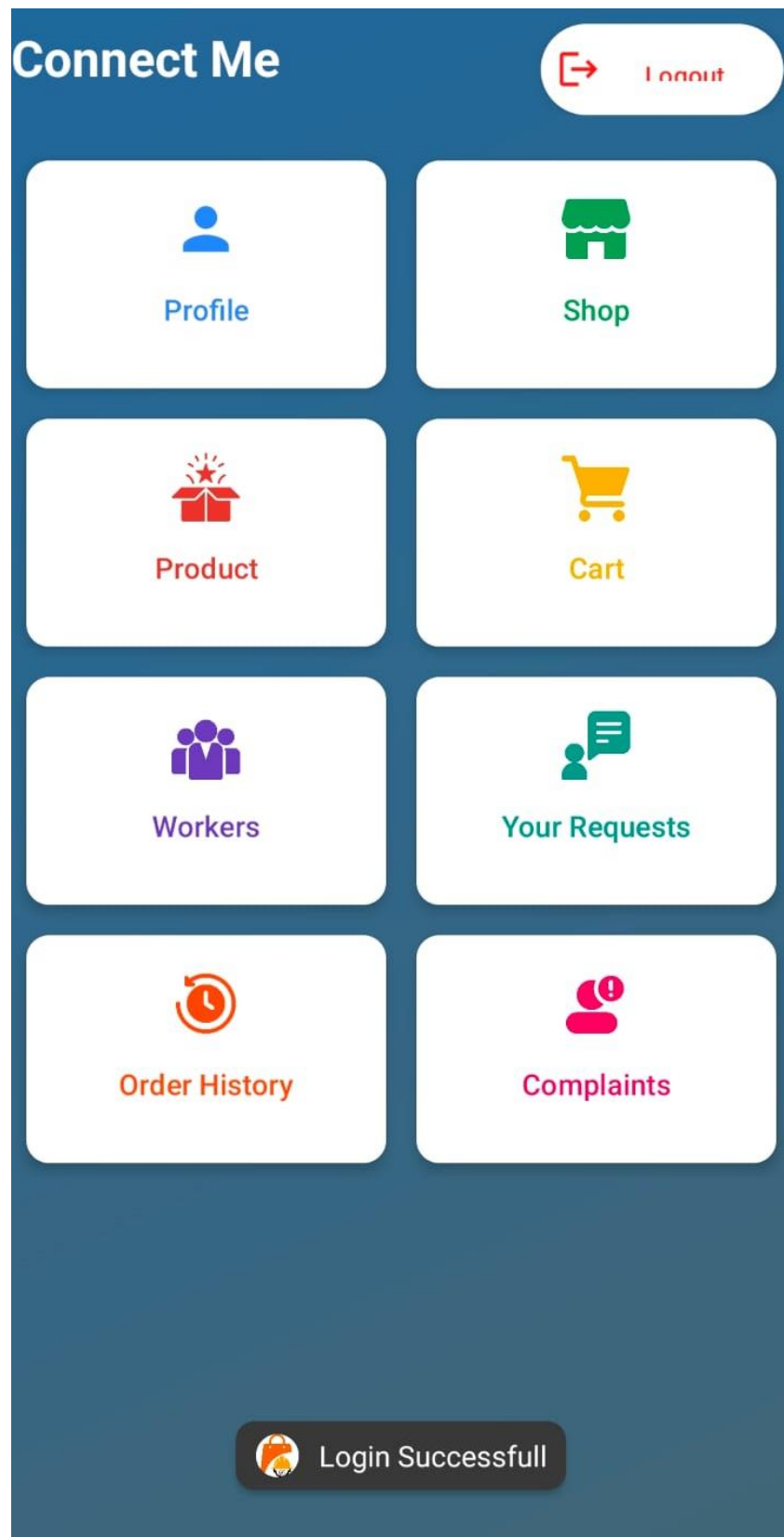
 Username

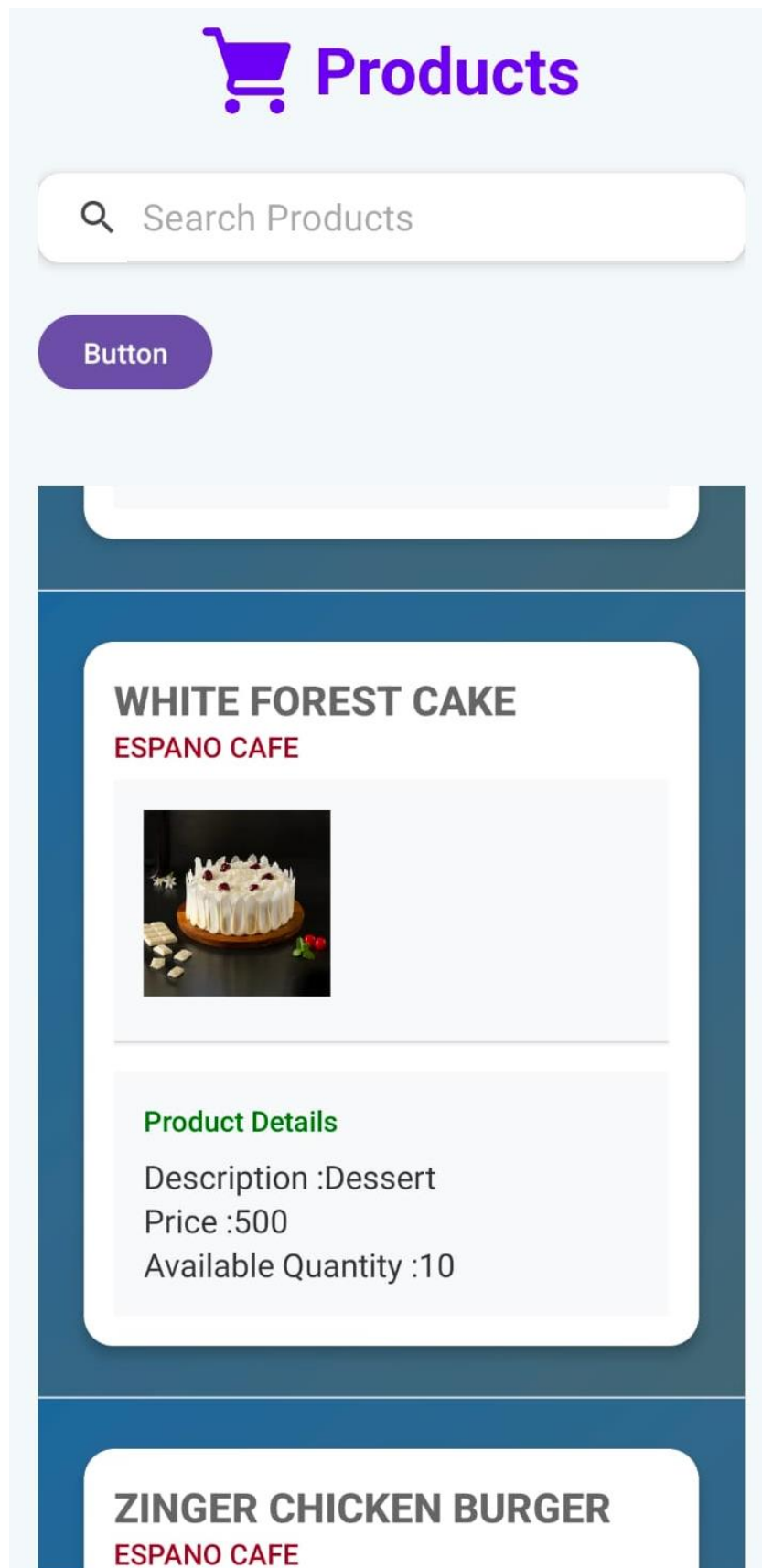
 Password 


LOGIN

SIGN UP


[Forgot Password?](#)











Sandeep Kumar
WORK: **sewage cleaner**


**Contact Details**

 Phone: **9457831846**


 Email: **sand345@gmail.com**


**Location Details**


 Place: **perumpilavu**




Vijaya Lakshmi
WORK: **home tutor**

**Contact Details**

 Phone: **3485071246**

 Email: **vijayatutor@gmail.com**

**Location Details**

ANSAR WOMEN'S COLLEGE PERUMPILAVU

PROJECT REPORT 24

order history

WHITE FOREST CAKE

ESPANO CAFE



Product Details

Description :Dessert

Price :500

Quantity :1

CHICKEN LOADED FRIES

ESPANO CAFE



Product Details

Description :non veg

Price :350

My Requests

10/03/2025

hospital emergency

Amount

280

Response

pending

Complaints

Enter your complaints

Submit Complaint

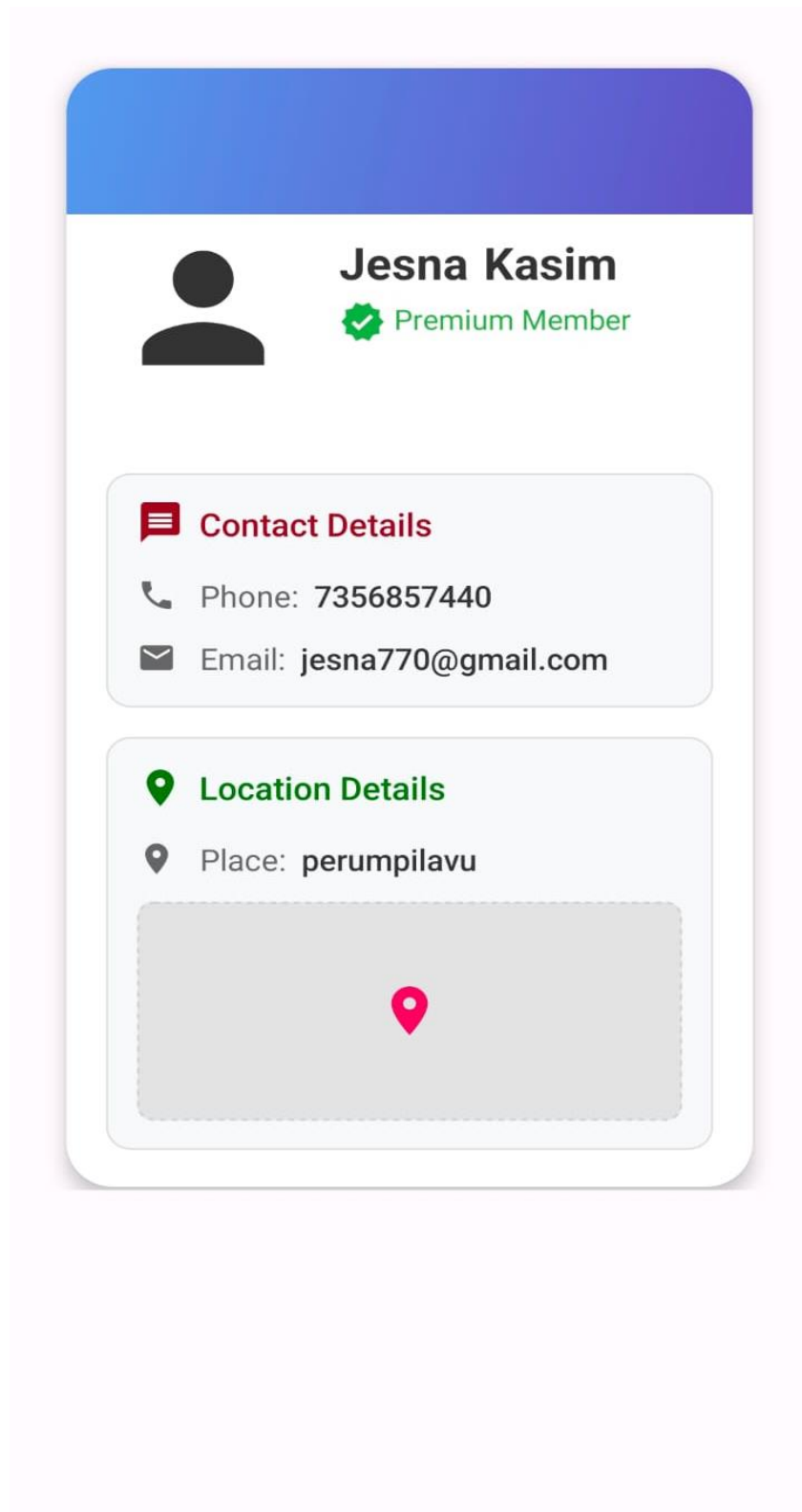
2025-03-08

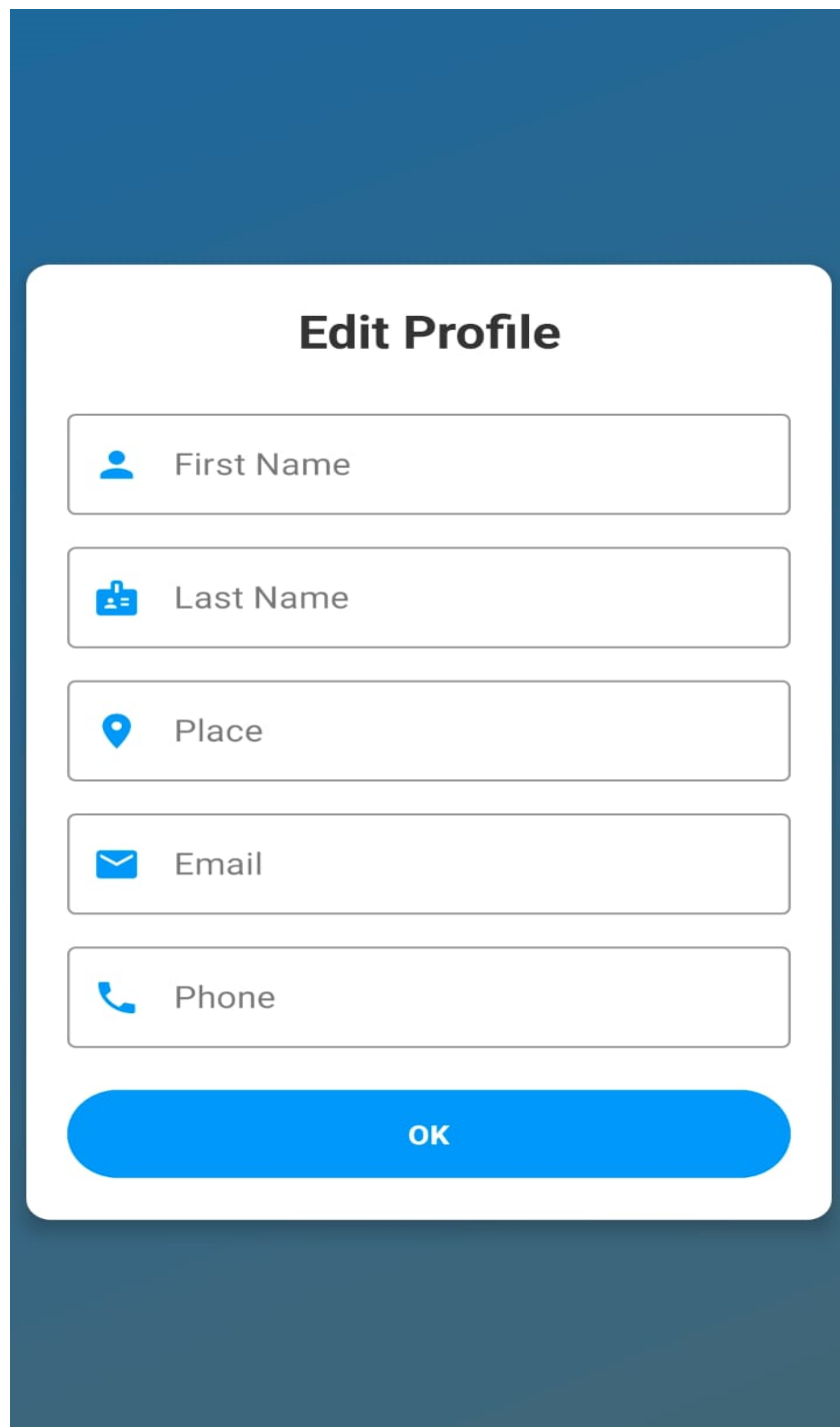
Complaint

app functions not working


Response


pending





A mobile application interface for editing a profile. It features a dark blue header and a white rounded rectangle containing the form. The form has five input fields, each with a blue icon and a label: a person icon for 'First Name', a name tag icon for 'Last Name', a location pin icon for 'Place', an envelope icon for 'Email', and a telephone handset icon for 'Phone'. Below these fields is a large blue button with the text 'OK' in white.


Edit Profile

 First Name

 Last Name

 Place

 Email

 Phone

OK
