

HomeCare

Project Report Submitted By

VIMAL THOMSON

Reg. No : AJC20MCA-2085

In Partial fulfillment for the Award of the Degree Of

**MASTER OF COMPUTER APPLICATIONS (2 Year)
(MCA)**

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala.

Approved by AICTE, Accredited by NAAC with 'A' grade.

Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2020-2022

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**HomeCare**” is the bonafide work of **VIMAL THOMSON (Reg.No:AJC20MCA-2085)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

Ms. Meera Rose Mathew
Internal Guide

Ms. Grace Joseph
Coordinator

Rev.Fr.Dr. Rubin Thottupurathu Jose
Head of the Department

External Examiner

DECLARATION

I hereby declare that the project report “**HomeCare**” is a bonafided work done at Amal Jyothi College of Engineering, towards the partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

Date: 21/07/2022

KANJIRAPPALLY

VIMAL THOMSON

Reg. No: AJC20MCA-2085

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillikutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinators **Rev.Fr.Dr. Rubin Thottupurathu Jose** and **Ms. Grace Joseph** for their valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also like to express sincere gratitude to my guide, **Ms. Meera Rose Mathew** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

VIMAL THOMSON

ABSTRACT

The bulk of established sectors have been disrupted by on-demand applications, which is the subject of this research. The way we travel, eat, buy, and even date has all changed dramatically. So why not our errands and domestic duties. We all require somebody in our life who can quickly do our household works and conduct our errands. Let's start at the very beginning and look at the precise services that on-demand home services applications offer before we get into the details. As the name implies, it provides a platform where you can easily engage specialists to handle your household duties. It has all the necessary functionality built in, just like all other on-Demands. Hire a plumber in your area to quickly address any leaks you may have at home. Got a problem with your air conditioner? Call a professional who can fix it in a matter of minutes. Similar to how they deal with carpentry issues, home appliance issues, and other household issues.

The proposed system consists of actors consisting of a worker, a client and an administrator. The admin has the rights to access and maintain the various users. The customer who wants to avail our services, has to go through registration and login. A client can request for a service by describing their needs. Once a request is made, it is forwarded to the workers, the worker can take the job based on his/her convenience once the job has been completed by the worker, he/she generates the bill. The client can then proceed to make the payment and can also rate the worker's service. The last actor is a service provider who provides a service , they are also supposed to go through the login and registration processes.

CONTENT

Sl. No	Topic	Page No
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	4
2.1	INTRODUCTION	5
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	6
2.4	PROPOSED SYSTEM	6
2.5	BENEFITS OF PROPOSED SYSTEM	6
3	REQUIREMENT ANALYSIS	7
3.1	FEASIBILITY STUDY	8
3.1.1	ECONOMICAL FEASIBILITY	8
3.1.2	TECHNICAL FEASIBILITY	9
3.1.3	BEHAVIORAL FEASIBILITY	9
3.2	SYSTEM SPECIFICATION	10
3.2.1	HARDWARE SPECIFICATION	10
3.2.2	SOFTWARE SPECIFICATION	10
3.3	SOFTWARE DESCRIPTION	10
3.3.1	FLUTTER	10
3.3.2	FIREBASE	11
4	SYSTEM DESIGN	12
4.1	INTRODUCTION	13
4.2	UML DIAGRAM	13
4.2.1	USE CASE DIAGRAM	14
4.2.2	SEQUENCE DIAGRAM	16
4.2.3	STATE CHART DIAGRAM	17
4.2.4	ACTIVITY DIAGRAM	18
4.2.5	CLASS DIAGRAM	19
4.2.6	OBJECT DIAGRAM	20
4.2.7	COMPONENT DIAGRAM	21
4.2.8	DEPLOYMENT DIAGRAM	22

4.3	USER INTERFACE DESIGN USING FIGMA	23
4.4	DATA BASE DESIGN	25
5	SYSTEM TESTING	28
5.1	INTRODUCTION	29
5.2	TEST PLAN	29
5.2.1	UNIT TESTING	29
5.2.2	INTEGRATION TESTING	36
5.2.3	WIDGET TESTING	36
6	IMPLEMENTATION	37
6.1	INTRODUCTION	38
6.2	IMPLEMENTATION PROCEDURE	38
6.3	HOSTING	39
7	CONCLUSION & FUTURE SCOPE	40
7.1	CONCLUSION	41
7.2	FUTURE SCOPE	41
8	BIBLIOGRAPHY	42
9	APPENDIX	44
9.1	SAMPLE CODE	45
9.2	SCREEN SHOTS	56
9.3	PLAGIARISM REPORT	62

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The bulk of established sectors have been disrupted by on-demand applications, which is the subject of this research. The way we travel, eat, buy, and even date has all changed dramatically. So why not our errands and domestic duties. We all require somebody in our life who can quickly do our household duties and conduct our errands. Let's start at the very beginning and look at the precise services that on-demand home services applications offer before we get into the details. As the name implies, it provides a platform where you can easily engage specialists to handle all of your household duties. It has all the necessary functionalities built in, hire a plumber in your area to quickly address any leaks you may have at home. Got a problem with your air conditioner? Call a professional who can fix it in a matter of minutes. Similar to how they deal with carpentry issues, home appliance issues, and other household issues.

1.2 PROJECT SPECIFICATION

The proposed system consists of actors consisting of a worker, a client and an administrator. The admin has the rights to access and maintain the various users. The customer who wants to avail the services, has to go through registration and login. A client can request for a service by describing their needs. Once a request is made, it is forwarded to the workers, the worker can take the job based on his/her convenience once the job has been completed by the worker, he/she generates the bill. The client can then proceed to make the payment and can also rate the worker's service. The last actor is a service provider who provides the service, they are also supposed to go through the login and registration processes.

The system includes 4 modules. They are:

1. Admin Module

The admin can view workers, add workers, remove workers and view the complaints if any registered by the customers. The admin can also add and remove services.

2. Workers Module

The workers are the skilled professionals in the various services provided by the app, like, carpenters, electricians, plumbers etc. they can provide their services to the customers in their desired time.

3. Customers Module

The customers are the users who use the various services provided by the app and avail the services of the various professionals according to their needs, the customers can give the workers a rating after availing their services and can even register a complaint against the workers if they didn't like the workers' services.

4. Payment's Module

The entire process of providing the services ends only after the customer pays the bill via the online mode. So, this is one of the most important modules.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

Data collection and analysis, problem-solving, and system change recommendations are all steps in the process of system analysis. During this problem-solving process, there must be considerable communication between the system users and the system developers. A system analysis or research should be the first step in any system development process. The system analyst acts as an interrogator and examines the operation of the current system in great detail. The system's input is acknowledged, and the system is viewed as a whole. The many processes might be connected to the organisations' outcomes. System analysis involves comprehending the problem, identifying the significant and crucial variables, analysing and synthesising the numerous components, and choosing the best or, at the very least, most acceptable course of action.

Preliminary research is the process of gathering and analysing data in order to use it for upcoming system investigations. Initial research requires strong collaboration between system users and developers since it involves problem-solving. It carries out several feasibility studies. These studies offer a rough idea of the system activities, which can be utilised to choose the methods to employ for effective system research and analysis..

2.2 EXISTING SYSTEM

The current system is not entirely automated, and no suitable program offers the aforementioned functions. The proposed approach corrects this; users just download the app, sign up, and begin using it to access services. The user interface of the app is incredibly user-friendly.

The current system must be altered in order to include new data, increase its efficiency, and make it more adaptable and secure. Customers may examine all information about their desired service, bill, and other items using the new system.

2.3 DRAWBACKS OF EXISTING SYSTEM

- No proper online management of system
- Human effort is needed.
- Important information is challenging to keep up in books.
- More manual labour is required to produce the necessary reports.

2.4 PROPOSED SYSTEM

The actors in the proposed system are a worker, a customer, and an administrator. Initial access and maintenance permissions are granted to the administrator, who must log in to do so. The consumer must complete registration and login in order to use our services. A client can make a service request by outlining their wants. Once a request is made, it is sent to the employees, who can accept the job at their convenience. Once the employee has finished the assignment, he or she generates the bill. Following that, the customer can complete the payment procedure and give the customer service a rating. Finally, there is a service provider who offers a service that requires users to register and log in.

2.5 BENEFITS OF THE PROPOSED SYSTEM

The system is relatively easy to install and develop. The system works in practically all settings and uses very little system resources. It has got following features:

➤ **Confidentiality and privacy: -**

For data to remain confidential and private measures must be taken to prevent unauthorized access. Username and password requirement to sign in ensures privacy.

➤ **Easy to use: -**

The proposed system is easier to use as the application has a very friendly user interface, making the app easier to use for the users of every age column.

➤ **Cloud Database: -**

The product will avoid the burden of hard copy storage. For performing the same activity, we can also save time and resources. The data can be kept for a longer time without losing any information.

As we use secured databases to maintain the data, it will also ensure data security.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study is conducted to determine if the project will, upon completion, fulfil the objectives of the organisation in relation to the labour, effort, and time invested in it. A feasibility study enables the developer to predict the project's usefulness and potential future. A system proposal's workability, which includes the influence on the organisation, capacity to satisfy user demands, and efficient use of resources, is the basis for a feasibility study. As a result, a feasibility analysis is frequently performed before a new application is approved for development.

The paper outlines the project's viability and contains a number of factors that were carefully taken into account throughout this project's feasibility assessment, including its technical, economic, and operational viabilities. The following are its features: -

3.1.1 Economical Feasibility

Cost and benefit analyses are required to support the emerging system. criteria to make sure that focus is placed on the project that will yield the best results the earliest. The price that would be involved in developing a new system is one of the variables.

Some significant financial queries raised during the initial probe include the following:

- The expenses carry out a comprehensive system investigation.
- The price of the software and hardware.
- The advantages come in the shape of lower prices or fewer expensive errors.

The suggested system was created as part of a project; hence, there are no manual expenses associated with it. Additionally, the fact that all of the resources are already at hand indicates that the system may be developed affordably.

The cost of project, Hemecare was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open source software.

3.1.2 Technical Feasibility

The system has to be assessed first from a technical standpoint. An overview design of the system's requirements in terms of input, output, programmes, and processes must serve as the foundation for the assessment of this viability. The inquiry must next advise the kind of equipment, necessary procedure for constructing the system, and means of operating the system once it has been developed after having identified an outline system.

The investigation's observed technical issues include:

- Does the suggested technology work with the current technology?
- Can the system grow if it is developed?

The project should be created in such a way that the required performance and functionality are met within the limitations. This project only has a few minor restrictions. The system was created using the programming languages Dart and Firebase, and it is technically feasible to complete the project. The System used was also of good performance and has an AMD Ryzen 5; RAM 8GB and, Hard disk 1TB

3.1.3 Behavioral Feasibility

The proposed system includes the following queries:

- Is there enough assistance for the users?
- Will the suggested system cause harm of any sorts?

The project would be advantageous because, when created and implemented, it would achieve the goals. The project is deemed to be behaviorally viable after carefully weighing all behavioural factors. The proposed system is evaluated and has deemed beneficial.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - AMD Ryzen 5

RAM - 8 GB

Hard disk - 1 TB

3.2.2 Software Specification

Front End - Flutter, Dart

Back End - Firebase

OS requirements - Windows 7 or above.

Technologies used - Dart, Flutter, Firebase Firestore & Firebase Storage

3.3 SOFTWARE DESCRIPTION

3.3.1 Flutter

In May 2017, Google created and released Flutter, a free and open-source mobile UI framework. Simply put, it makes it possible for you to create a native mobile application from a single codebase. This suggests that you may create two different apps with the same programming language and codebase (for iOS and Android).

Flutter consists of two important parts:

- A collection of tools known as an SDK (Software Development Kit) can help you create applications. Also provided are tools for translating your code to native machine code (code for iOS and Android).
- A framework (UI library based on widgets) is a collection of reusable user interface (UI) elements that you may customise to suit your needs. These elements include buttons, text input fields, sliders, and other objects.

Applications based on Flutter are made using the programming language Dart. Although Google created the language for the first time in October 2011, it has made great progress since then.

A front-end coding framework called Dart could be used to create programmes for both internet and mobile platforms.

3.3.2 Firebase

Firebase is a backend-as-a-service(BaaS). It gives developers a variety of tools and services so they can produce high-quality apps, increase their user base, and earn money. It was developed on the technology platform of Google. Data is stored in documents that resemble JSON in Firebase, a NoSQL database program.

Key features of firebase:

- Authentication: Firebase supports authentication. Passwords, facebook, google, phone number etc can be used to authenticate. We can even integrate more than one sign-in method into our app using firebase.
- Realtime Database: Data is available and synced across all the clients in realtime, even if and when the app goes offline.
- Hosting: Firebase offers the app quick hosting.
- Test lab: The app can be tested on physical devices located on Google's data center or even on virtual devices.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Design is the first step in the development of any engineered system or product. Design is a creative process. A good design is the key to a system that works effectively. "Design" is the process of using many approaches and concepts to thoroughly outline a process or a system so that it can be physically implemented. The process of using several approaches and concepts to specify a tool, a procedure, or a system in sufficient detail to enable its physical actuality is one way to put it. Software design serves as the technical foundation of the software engineering process, regardless of the development paradigm used. The system design generates the architectural detail required to build a system or product. As with any systematic technique, this software underwent the best design phase possible, fine-tuning all efficiency, performance, and accuracy levels. During the design stage, a user-oriented document is transformed into a document for programmers or database employees. Logical design and physical design are the two phases of system design development.

4.2 UML DIAGRAM

UML stands for **Unified Modeling Language**. Compared to other popular programming languages like C++, Java, COBOL, etc., UML is unique. A visual language called UML is used to create software blueprints. A general-purpose visual modelling language for software system visualisation, specification, construction, and documentation is what UML is known as. UML is not just used to represent software systems, despite the fact that this is its most common application. It is also used to model systems that are not software-based. For instance, the manufacturing facility's process flow, etc. Although UML is not a programming language, tools exist that can be used to convert UML diagrams into code in a number of other languages. The analysis and design of object-oriented systems are directly related to UML.

After some standardization, UML has become an OMG standard. A comprehensive UML diagram that depicts a system is made up of all the parts and relationships. The most crucial aspect of the entire process is how the UML diagram looks. It is completed by using all the additional components. The following diagrams are part of UML.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Activity diagram
- Statechart diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

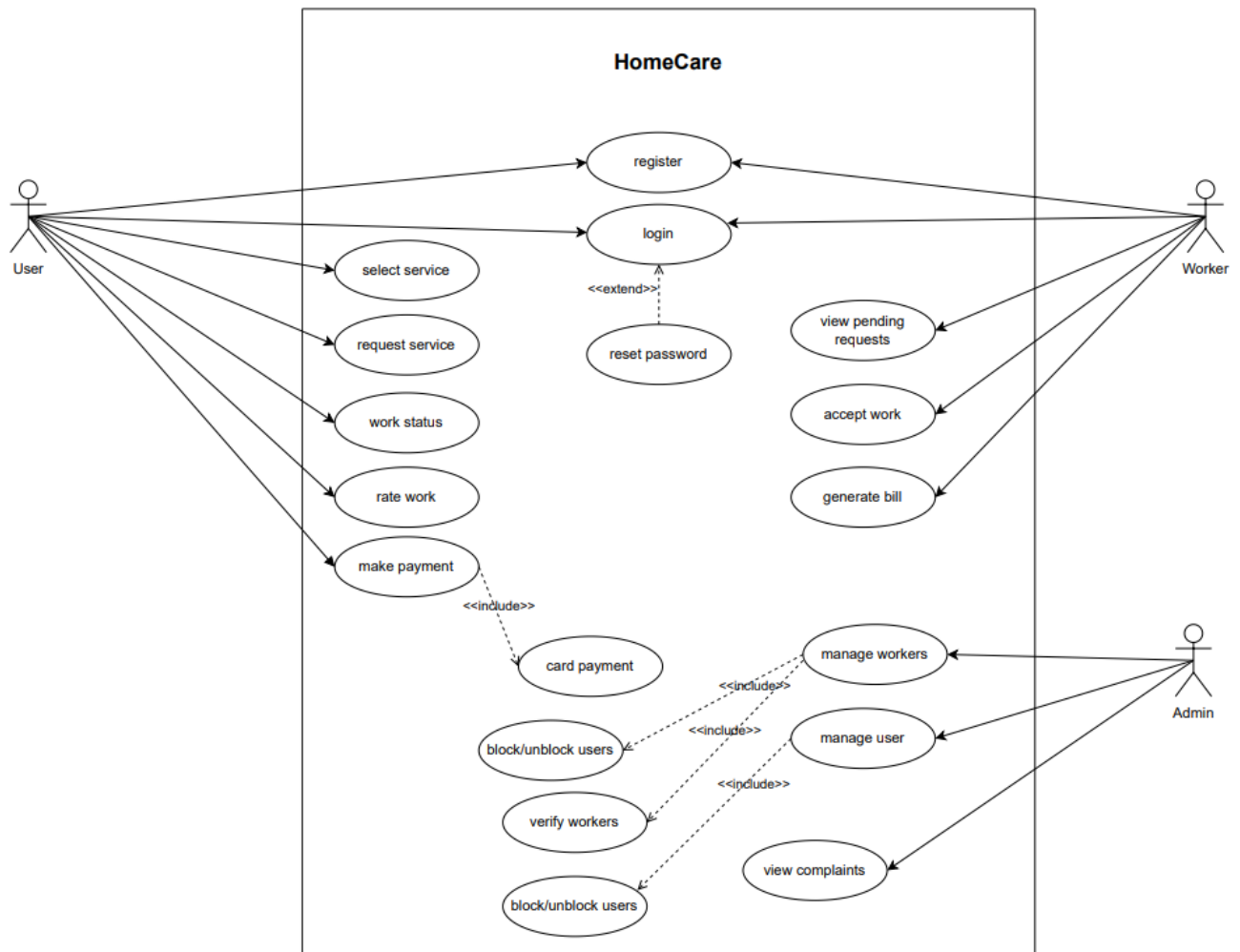
A use case diagram is a graphic representation of how a user might interact with a technology. A use case diagram, which is generally supported by other types of diagrams, displays the numerous various use cases and user types the system has. Either circles or ellipses are used to depict the use cases. Usually, sticks are used to depict the performers.

A use case itself may go into considerable detail about each situation, but a use-case diagram can help by providing a higher-level picture of the system. It has been said before that "Use case diagrams are the blueprints for your system".

Use case diagrams are created to depict a system's functional needs. To create an effective use case diagram after identifying the aforementioned things, we must adhere to the following rules.

- A use case's naming is very significant. The name should be selected in a way that makes it clear what functions are being performed.
- Give a suitable name for actors.
- Clearly depict relationships and dependencies in the diagram.
- Keep in mind that the diagram's primary function is to identify the needs, therefore avoid attempting to include all possible relationships.

Usecase Diagram



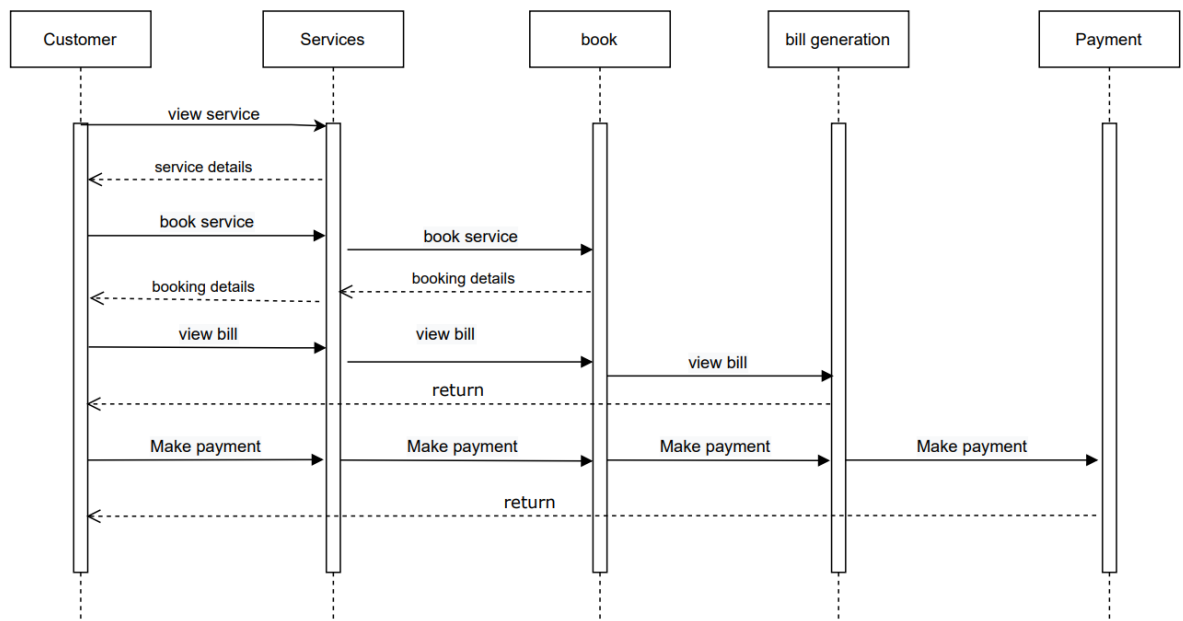
4.2.2 SEQUENCE DIAGRAM

A sequence diagram, commonly referred to as a system sequence diagram (SSD), is a visual representation of process interactions arranged chronologically in the field of software engineering. It illustrates the processes involved and the order in which messages must be exchanged for the processes to provide the functionality. Sequence diagrams and use case realisations are frequently linked in the system-under-4+1 development's architectural viewpoint paradigm. Sequence diagrams are also known as event diagrams and event scenarios.

Uses of sequence diagrams –

- Employed to model and illustrate the reasoning behind a complex function, process, or procedure.
- They are also employed to display information about UML use case diagrams.
- Employed to comprehend the precise operation of present or upcoming systems.
- Visualize the flow of tasks and messages within a system's objects or components.

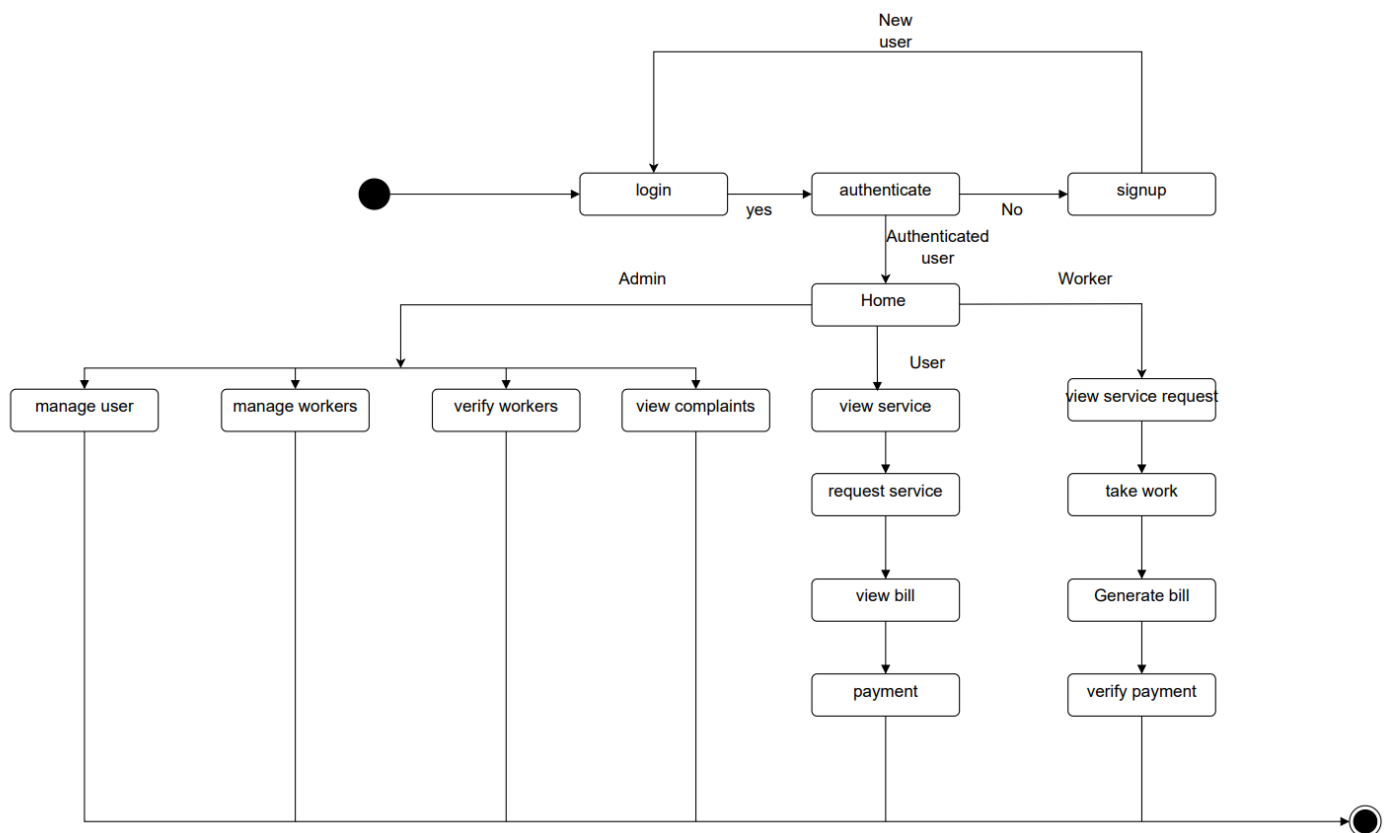
Sequence Diagram



4.2.3 STATE CHART DIAGRAM

A particular form of diagram used in computer science and related subjects to explain how systems behave is called a state diagram. State diagrams call for the system being represented to consist of a finite number of states; occasionally, this is the case, and other times, it's only an acceptable abstraction. State diagrams come in a variety of shapes and sizes, each with a distinct meaning. State diagrams are there to provide a system's behaviour an abstract explanation. In order to evaluate and demonstrate this behaviour, a sequence of events that could occur in one or more hypothetical states is employed. "Each diagram typically depicts objects of a single class and monitor the different states of its objects across the system," according to this.

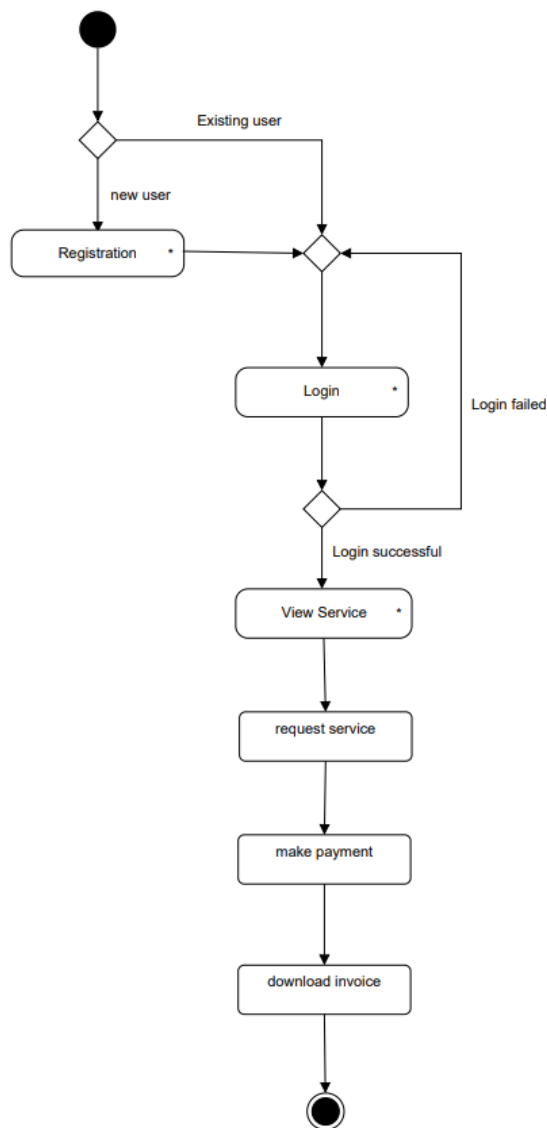
State Chart Diagram



4.2.4 ACTIVITY DIAGRAM

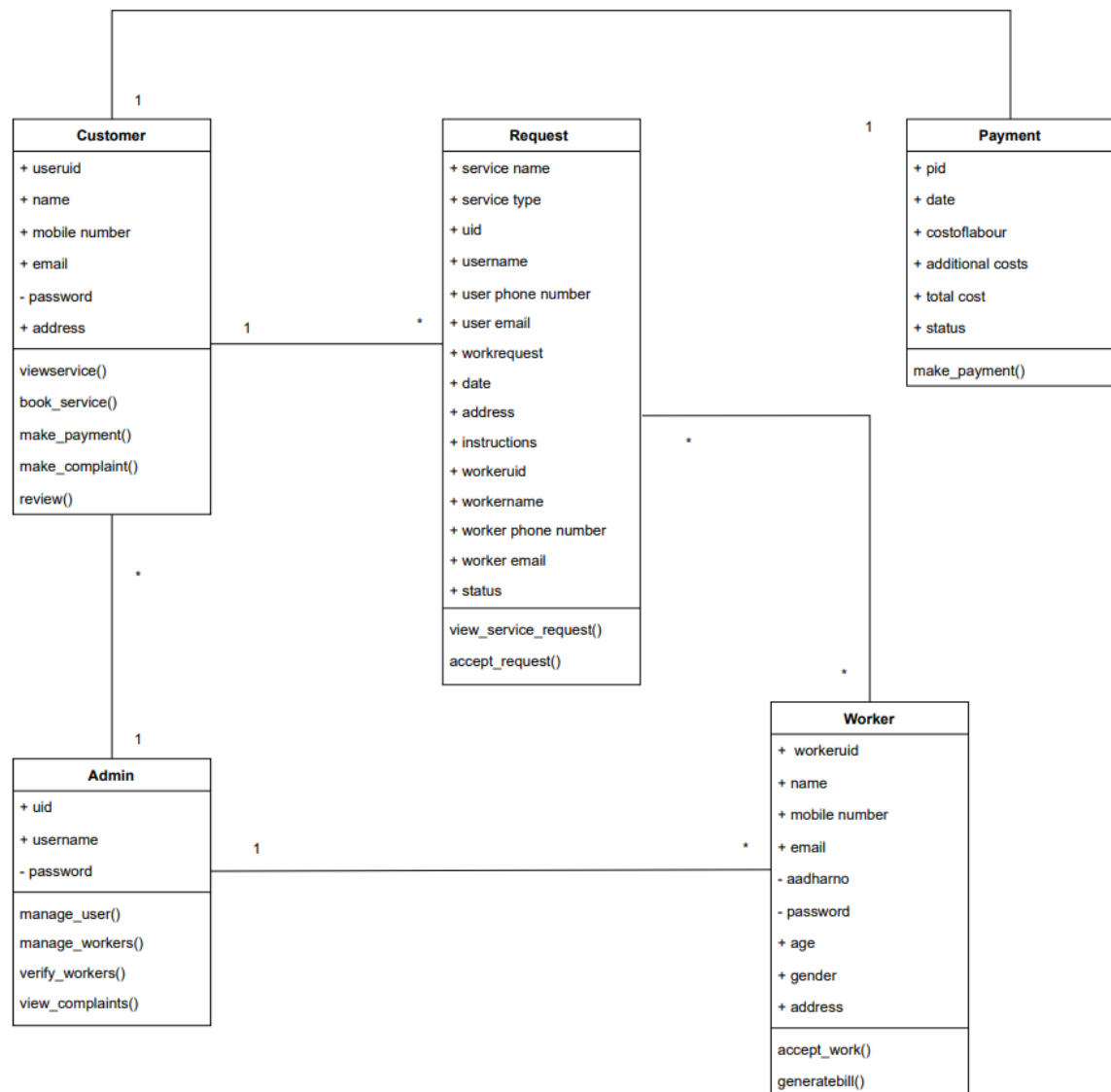
Activity diagrams depict how different levels of abstraction of activities are linked to provide a service. Typically, an event should be completed by some activities, particularly when the activity is intended to do multiple separate goals that need coordination. Another typical requirement is how the events in a single use case interact with one another, particularly in use cases where operations may overlap and require coordination. It may also be used to show how a collection of interrelated use cases interacts to reflect business operations.

Activity Diagram



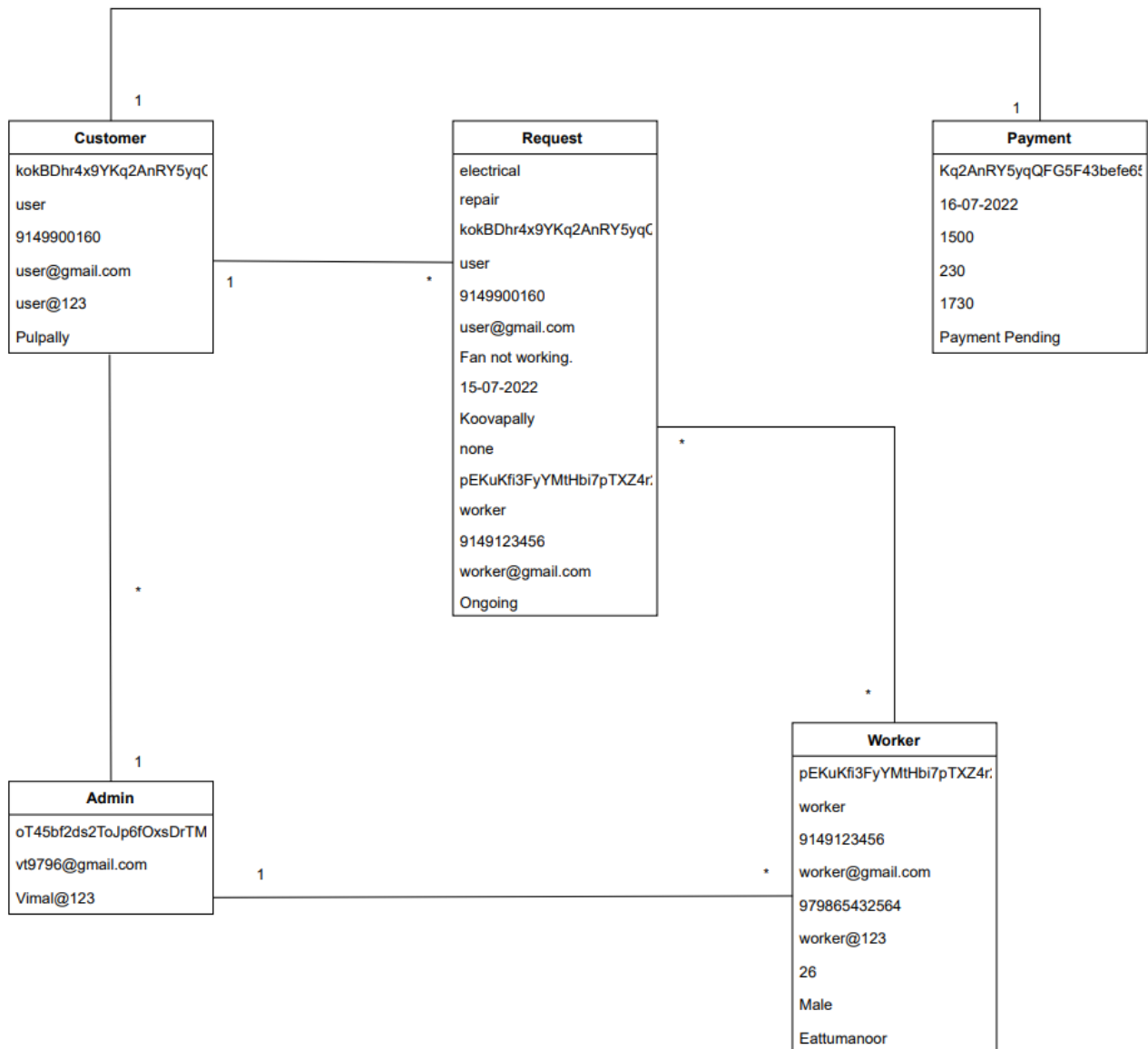
4.2.5 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of the application. Class diagrams are useful for visualising, describing, and documenting various system components as well as for writing executable code for software applications. A class diagram describes the constraints imposed on the system together with the properties and operations of a class. The only UML diagrams that can be directly converted into object-oriented languages are class diagrams, which are extensively utilised in the designing of object-oriented systems. An assortment of classes, interfaces, affiliations, partnerships, and limitations are displayed in a class diagram. It also goes by the name "structural diagram."



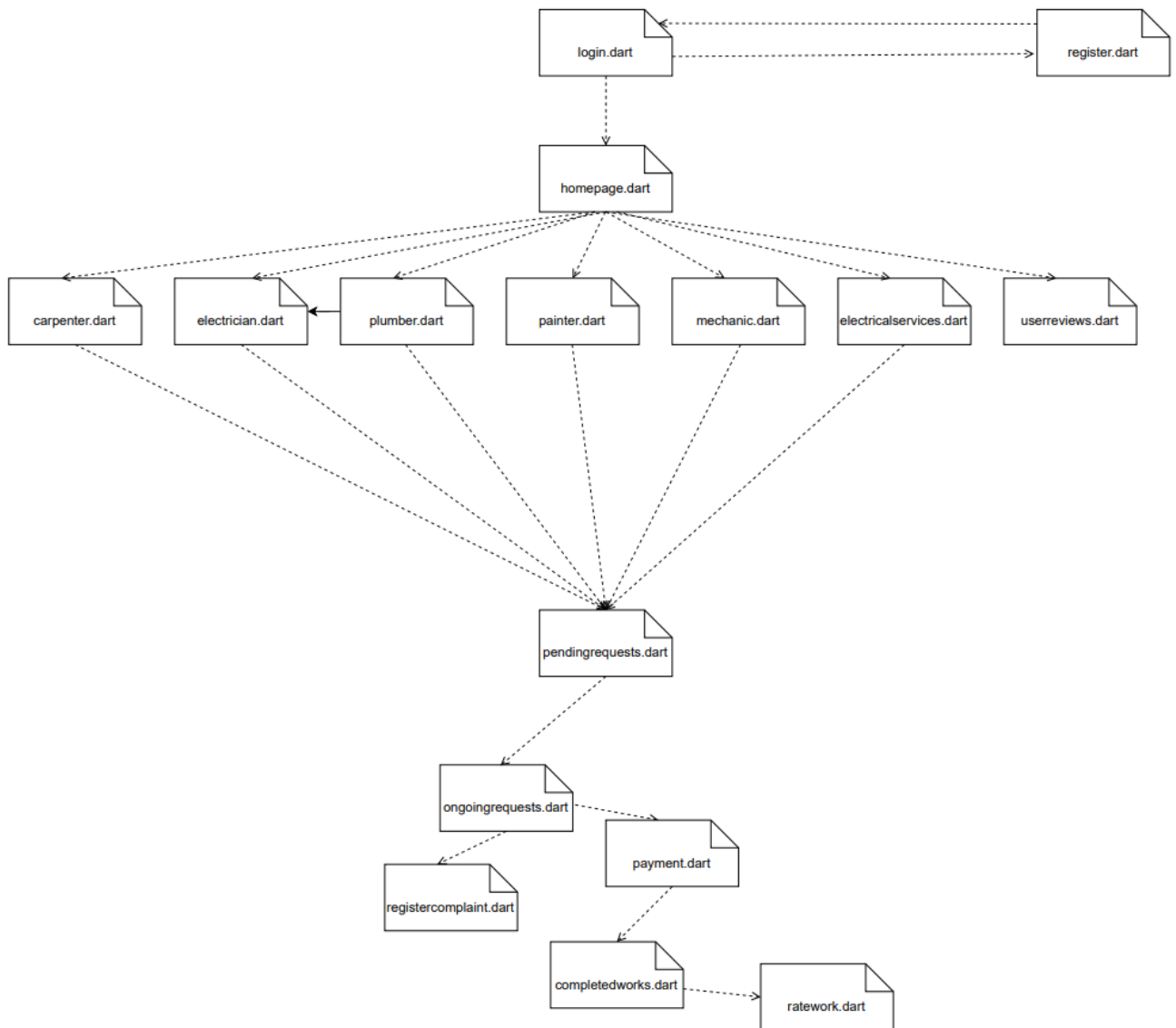
4.2.6 OBJECT DIAGRAM

Class diagrams are necessary before object diagrams can be created since they are the ancestor of object diagrams. An object diagram represents a particular instance of a class diagram. The underlying concepts used in class and object diagrams are the same. Object diagrams may also describe the static view of a system, although this static view only depicts a current state of the system. Object diagrams are used to show links between a set of things.



4.2.7 COMPONENT DIAGRAM

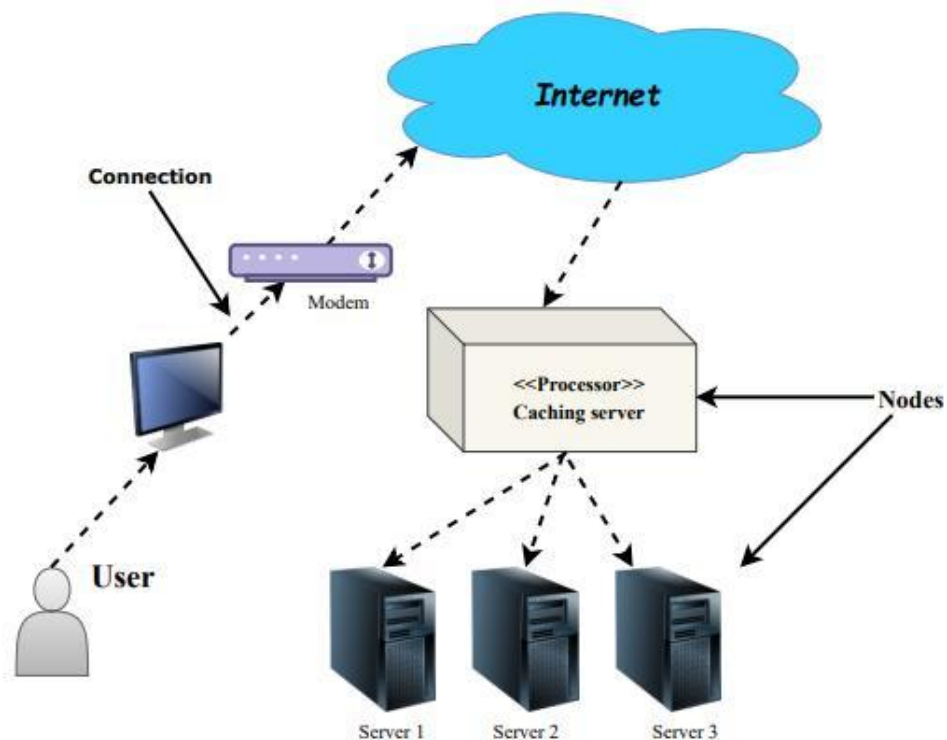
Component diagrams have different behaviours and personalities. The physical parts of the system are represented using component diagrams. Executables, libraries, files, documents, and other items that are physically present in a node are just a few examples. Component diagrams are used to show how the components of a system are connected and arranged. These diagrams may also be used to construct systems that can be run.



4.2.8 DEPLOYMENT DIAGRAM

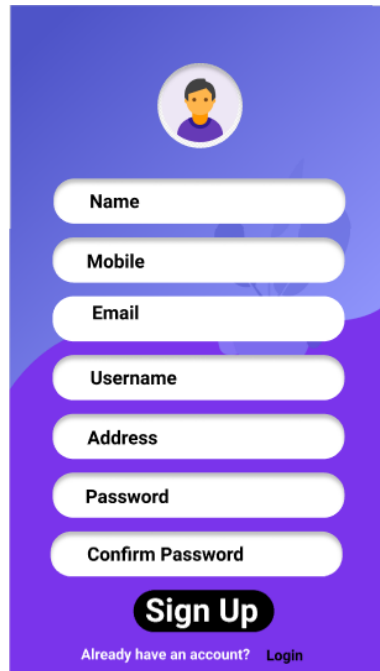
An execution architecture of a system, containing nodes like hardware or software execution environments, and the middleware linking them, is shown in a deployment diagram, a form of UML diagram. Typically, deployment diagrams are used to represent the actual hardware and software of a system.

By using it, you can comprehend how the hardware will physically deliver the system. In contrast to other UML diagram types, which primarily depict the logical components of a system, deployment diagrams assist describe the hardware structure of a system.



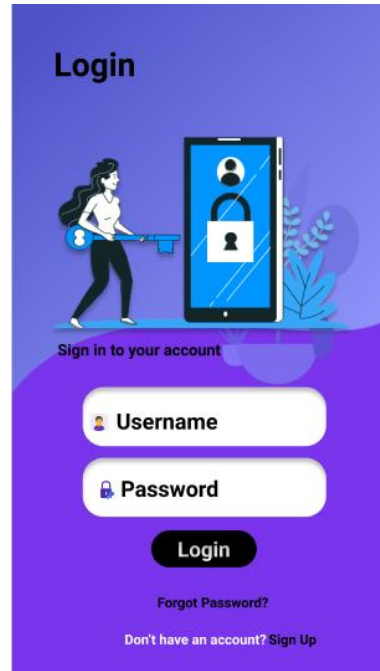
4.5 USER INTERFACE DESIGN USING FIGMA

Registration Page



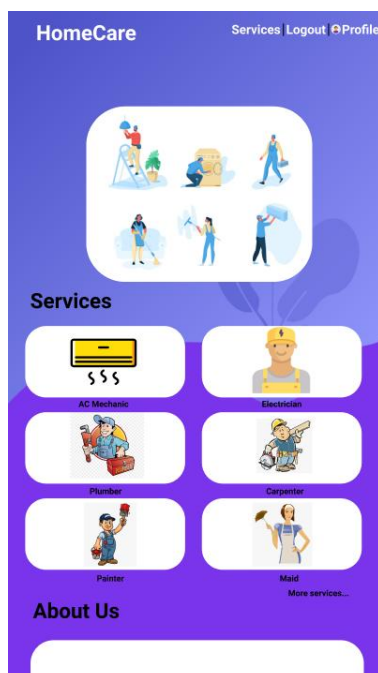
The Registration Page features a purple gradient background. At the top, there is a circular profile icon placeholder. Below it, a series of white rounded rectangular input fields are stacked vertically, labeled: Name, Mobile, Email, Username, Address, Password, and Confirm Password. A prominent black 'Sign Up' button is located at the bottom of the form. Below the button, a link reads 'Already have an account? Login'.

Login Page



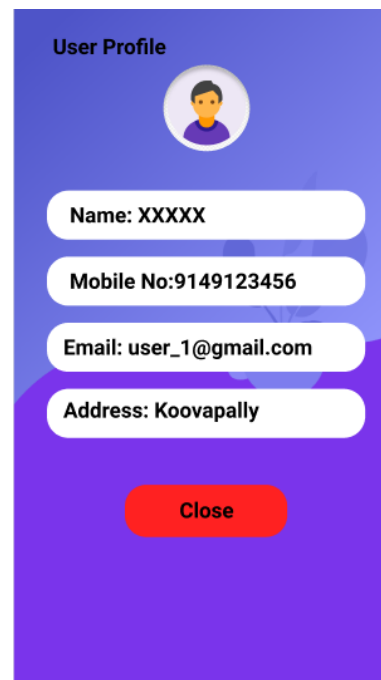
The Login Page has a purple gradient background. It features an illustration of a woman holding a large smartphone with a lock icon on its screen. Below the illustration, the text 'Sign in to your account' is displayed. The login form consists of two white rounded rectangular input fields labeled 'Username' and 'Password', followed by a black 'Login' button. At the bottom, there are two links: 'Forgot Password?' and 'Don't have an account? Sign Up'.

User Homepage



The User Homepage has a purple gradient background. At the top, it shows 'HomeCare' and navigation links for 'Services', 'Logout', and 'Profile'. Below this is a large white rounded rectangle containing six icons representing different services. Underneath, a 'Services' section displays six service cards in a 3x2 grid, each with an icon and a label: AC Mechanic, Electrician, Plumber, Carpenter, Painter, and Maid. At the bottom, there is an 'About Us' section.

User Profile Page



The User Profile Page has a purple gradient background. It features a circular profile icon placeholder at the top. Below it, four white rounded rectangular boxes display the user's information: Name: XXXXX, Mobile No: 9149123456, Email: user_1@gmail.com, and Address: Koovapally. A red 'Close' button is positioned at the bottom of the page.

Book Carpenter




Select Worker

Worker_1	Koovapally	Available
Worker_2	Koovapally	Not Available
Worker_3	Kanjirapally	Available

Home

Worker Profile screen

Worker Profile



Name: XXXXX

Mobile No: 9149123456

Email: wrkr_1@gmail.com


Address: Koovapally

Age: 35

Gender: Male

Close

Booking Screen



Worker_id: 123456

Name: xxxxxxxxxx

Reason for requesting work

Amount: \$100

Submit

4.6. DATABASE DESIGN

A database is a structured system with the capacity to store information and allows users to retrieve stored information quickly and effectively. Any database's primary goal is its data, which demands protection.

There are two stages to the database design process. The user needs are obtained in the first step, and a database is created to as clearly and accurately meet these criteria.

4.6.1 NoSQL Databases

In contrast to relational databases' use of tabular relations for modelling relationships, a NoSQL database provides a mechanism to store and retrieve data. Non-SQL or "non-relational" was the first meaning of the word "noSQL." Despite the fact that such databases have existed since the late 1960s, the word "NoSQL" wasn't actually developed until the start of the twenty-first century in response to the needs of Web 2.0 companies. More and more, NoSQL databases are used by big data and real-time web applications. NoSQL systems may support SQL-like query languages or coexist with SQL databases in polyglot-persistent architectures, as shown by the phrase "Not Only SQL" that is sometimes used to describe them.

The ease of design, easier "horizontal" scaling to machine clusters (which is problematic for relational databases), greater control over availability, and lowering the object-relational impedance mismatch are the driving forces behind this method. NoSQL databases employ different data structures than relational databases by default, which speeds up some operations. Examples of these different data structures include key-value pairs, broad columns, graphs, and documents. Depending on the issue that has to be solved, a specific NoSQL database may or may not be particularly suitable. NoSQL databases' data structures are sometimes seen as "more flexible" than relational databases' tables.

The consistency (as defined by the CAP theorem) of many NoSQL stores is sacrificed in favour of availability, partition tolerance, and speed. The use of low-level query languages (instead of SQL, for example), the inability to do ad hoc joins across tables, the absence of standardised interfaces, and significant prior investments in relational databases are all obstacles to the widespread use of NoSQL stores. True ACID transactions are absent from the majority of NoSQL systems, while some databases have made them a key component of their architectures.

Collection 1

Collection name: users

Field name	Datatype	Description
uid	String	User id automatically generated by firebase
name	String	The name of the user
email	String	The username with which a certain user logs in
gender	String	The gender of the user
phone	String	User's phone number
place	String	User's address
age	String	User's age
profession	String	Worker's profession
profilepicURL	String	URL of the User's Profile picture
rating	String	Rating of the workers done by users
role	String	Role of the certain user
status	String	Determines whether a user can login
aadhar_no	String	Aadhar Number of the worker

Collection 2

Collection name: workrequests

Field name	Datatype	Description
Date	String	The date of the service request
address	String	The address of the work request
instruction	String	Any specific information for the worker
status	String	The status of the service request
username	String	Name of the user who requested the service
useremail	String	Email of the user who requested the service
userphoneno	String	Phone no of the user who requested the service
userid	String	uid of the user who requested the service
work	String	The field of work requested by the user
worktype	String	The type of work requested by the user
workername	String	The name of the worker
workeremail	String	The email of the worker
workerphoneno	String	The phone number of the worker
workeruid	String	The uid of the worker

Collection 3

Collection name: reviews

Field Name	Datatype	Description
Date	String	The date the review has been posted
review	String	The content of the review
username	String	The name of the user who posted the review
useremail	String	The email of the user who posted the review
image	String	Url of the users profile picture

Collection 4

Collection name: complaints

Field Name	Datatype	Description
complaint	String	Content of the complaint
Date	String	The date in which complaint is registered
workername	String	The name of the worker
workeremail	String	The email of the worker
Workerprofession	String	The profession of the worker
username	String	The email of the user who registered the complaint

Collection 5

Collection name: payment

Field name	Datatype	Description
Date	String	The date of the service request
totalrate	String	The total cost of the actual service
costoflabour	String	Total cost of the labour
addcosts	String	Any additional costs
username	String	Name of the user who requested the service
useremail	String	Email of the user who requested the service
userphoneno	String	Phone no of the user who requested the service
userid	String	uid of the user who requested the service
status	String	The status of payment
workername	String	The name of the worker
workeremail	String	The email of the worker
workerphoneno	String	The phone number of the worker
workeruid	String	The uid of the worker

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

The testing portion of an application's development life cycle is crucial. It guarantees the excellent calibre of the application. Testing necessitates meticulous preparation and execution. Additionally, it takes up the most time during the development process.

The Flutter framework and Dart language offer substantial support for an application's automated testing.

Nothing is finished without testing, as it is essential to the system's testing goals. Several guidelines can be used as testing goals, such as:

Executing a program under test conditions with the goal of identifying errors is known as testing.

- A successful test case has a high likelihood of detecting a mistake that has not yet been identified.
- A test that finds a mistake that hasn't been noticed is successful.

5.2 TEST PLAN

A test plan suggests a number of required steps that need be taken in order to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer program, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfills the purpose for which it was intended.

Types of testing:

- ❖ Unit testing
- ❖ Integration Testing
- ❖ Widget Testing

5.2.1 Unit Testing

A unit test examines a single function, method, or class. Confirming that a logic unit is sound under various conditions is the goal of a unit test. External dependencies of the testing unit are frequently mocked out. Generally speaking, unit tests don't display to the screen, read from or write to the disc, or request user input from processes other than the one that is running the test.

5.2.1.1 Test Case 1

```
import ...
void main() {
  test('empty email returns error string', () {
    final result = EmailFieldValidator.validate('');
    expect(result, 'Email cannot be empty');
  });
  test('non-valid email returns error', () {
    final result = EmailFieldValidator.validate('email');
    expect(result, 'Please enter a valid email');
  });
  test('empty password returns error string', () {
    final result = PasswordFieldValidator.validate('');
    expect(result, 'Password cannot be empty');
  });
  test('non-valid password returns error', () {
    final result = PasswordFieldValidator.validate('pass');
    expect(result, 'please enter valid password min. 6 character');
  });
  test('valid email', () {
    final result = EmailFieldValidator.validate('vt9796@gmail.com');
    expect(result, null);
  });
  test('valid password', () {
    final result = PasswordFieldValidator.validate('vimal@123');
    expect(result, null);
  });
}
```

Test Result

✓ Tests passed: 6 of 6 tests – 101 ms

C:\src\flutter\bin\flutter.bat --no-color test --machine --start-paused test\login_test.dart

Testing started at 2:50 PM ...

✓ Test Results	101 ms
✓ login_test.dart	101 ms
✓ empty email returns error string	56 ms
✓ non-valid email returns error	10 ms
✓ empty password returns error string	8 ms
✓ non-valid password returns error	8 ms
✓ valid email	10 ms
✓ valid password	9 ms

Test Case 1					
Project Name: HomeCare					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Vimal Thomson		
Test Priority(Low/Medium/High):High			Test Designed Date: 20-07-2022		
Module Name: Login Screen			Test Executed By : Ms Meera Rose Mathew		
Test Title : Login_test			Test Execution Date: 21-07-2022		
Description: Test the Login Page					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Entering no email		Error string should be displayed	Error string displayed	Pass
2	Provide in valid email	email: email	Error string should be displayed	Error string displayed	Pass
3	Entering no password		Error string should be displayed	Error string displayed	Pass
4	Provide invalid password	Password : pass	Error string should be displayed	Error string displayed	Pass
5	Provide valid email	Email: vt9796@gmail.com	No error string should be displayed	No error string displayed	Pass
6	Provide valid password	Password: Vimal@12	No error string should be displayed	No error string displayed	Pass

5.2.1.2 Test Case 2

```
import 'package:MECARE/pages/screens/job-carpenter.dart';
import 'package:flutter_test/flutter_test.dart';

void main() {
  test('Empty Work description returns error string', () {
    final result = WorkDescValidator.validate('');
    expect(result, "Can't be empty");
  });
  test('Non valid work description returns error ', () {
    final result = WorkDescValidator.validate('Hi');
    expect(result, 'Not Valid');
  });
  test('Empty address field returns error string', () {
    final result = AddressValidator.validate('');
    expect(result, 'Address cannot be empty');
  });
}
```

Test Result

✘ Tests failed: 1, passed: 2 of 3 tests – 485 ms

C:\src\flutter\bin\flutter.bat --no-color test --machine --start-paused test\job-carpenter_test.dart

Testing started at 5:25 AM ...

✘ Test Results	485 ms
✘ job-carpenter_test.dart	485 ms
✓ Empty Work description returns error string	75 ms
✘ Non valid work description returns error	407 ms
✓ Empty address field returns error string	3 ms

Test Case 2					
Project Name: HomeCare					
Login Test Case					
Test Case ID: Test_2			Test Designed By: Vimal Thomson		
Test Priority(Low/Medium/High): High			Test Designed Date: 20-07-2022		
Module Name: job-Carpenter			Test Executed By : Ms Meera Rose Mathew		
Test Title : job-carpenter_test			Test Execution Date: 21-07-2022		
Description: Test the Carpenter Job page					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Entering no work description		Error string should be displayed	Error string displayed	Pass
2	Provide in valid work description	Work description: hi	Error string should be displayed	Error string not displayed	Fail
3	Entering no address		Error string should be displayed	Error string displayed	Pass

5.2.1.3 Test Case 3

```
import 'package:HEMOCARE/pages/workerprofilescreens.dart';
import 'package:flutter_test/flutter_test.dart';

void main() {
  test('Empty age field returns error string', () {
    final result = AgeValidator.validate('');
    expect(result, "Can't be empty");
  });
  test('Invalid age returns error string', () {
    final result = AgeValidator.validate('abc');
    expect(result, "Enter a valid Age");
  });
  test('Empty Aadhar number field returns error string', () {
    final result = AadharValidator.validate('');
    expect(result, "Can't be empty");
  });
  test('Invalid aadhar no returns error string', () {
    final result = AadharValidator.validate('abcd');
    expect(result, 'Enter a valid aadhar number');
  });
}
```

Test Result

✗ Tests failed: 2, passed: 2 of 4 tests – 111 ms

C:\src\flutter\bin\flutter.bat --no-color test --machine --start-paused test\workerprofile_test.dart

Testing started at 7:24 AM ...

✗ Test Results	111 ms
✗ workerprofile_test.dart	111 ms
✗ Empty age field returns error string	82 ms
✓ Invalid age returns error string	6 ms
✗ Empty Aadhar number field returns error string	17 ms
✓ Invalid error returns error string	6 ms

Test Case 3					
Project Name: HomeCare					
Login Test Case					
Test Case ID: Test_3			Test Designed By: Vimal Thomson		
Test Priority(Low/Medium/High): High			Test Designed Date: 20-07-2022		
Module Name: updateworkerprofile			Test Executed By : Ms Meera Rose Mathew		
Test Title : updateprofile_test			Test Execution Date: 21-07-2022		
Description: Test the Update worker profile screen					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Entering no age		Error string should be displayed	Wrong error string displayed	Fail
2	Provide an invalid age	age: abc	Error string should be displayed	Error string displayed	Pass
3	Entering no aadhar no		Error string should be displayed	Wrong error string displayed	Fail
4	Provide an invalid aadhar number	Aadhar Number : abcd	Error string should be displayed	Error string displayed	Pass

5.2.2 Integration Testing

An integration test evaluates software in its entirety or in significant portions. An integration test's objective is to confirm that all the widgets and services under test interact as expected. Integration tests are another tool you may use to validate the functionality of your software.

A genuine device or an OS emulator like iOS Simulator or Android Emulator are frequently used in integration tests. The app being tested is often separated from the test driver code to avoid skewing the findings.

5.2.3 Widget Testing

In other UI frameworks, a widget test—also referred to as a component test—evaluates a single widget. Confirming that a widget's user interface (UI) performs and behaves as intended is the goal of a widget test. Testing widgets entails testing several classes in a test environment that provides the right widget lifecycle context.

For instance, the widget being evaluated should be able to execute layout, accept user events and actions, and respond to them, as well as generate child widgets. A widget test is therefore more in-depth than a unit test. Similar to a unit test, a widget test replaces the environment with a much simpler implementation rather than a full-fledged UI system.

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

The project's implementation phase is where the conceptual design is transformed into a functional system. It can be regarded as the most important stage in creating a successful new system since it gives users assurance that the system will operate as intended and be reliable and accurate. User documentation and training are its main concerns. Usually, conversion happens either during or after the user's training. Implementation is the process of turning a newly revised system design into an operational one, and it simply refers to placing a new system design into operation.

Implementation encompasses all of the steps used to switch from the old system to the new one. The new system could be entirely different, take the place of an existing manual or automated system, or it could be modified to work better. A reliable system that satisfies organisational needs must be implemented properly. System implementation refers to the process of actually using the built system. This comprises all the processes involved in switching from the old to the new system. Only after extensive testing and if it is determined that the system is operating in accordance with the standards can it be put into use. The system personnel assess the system's viability. The effort necessary for system analysis and design to implement the three key components of education and training, system testing, and changeover will increase in proportion to how complicated the system being implemented is.

The implementation state involves the following tasks:

- ☐ Careful planning.
- ☐ Investigation of system and constraints.
- ☐ Design of methods to achieve the changeover.

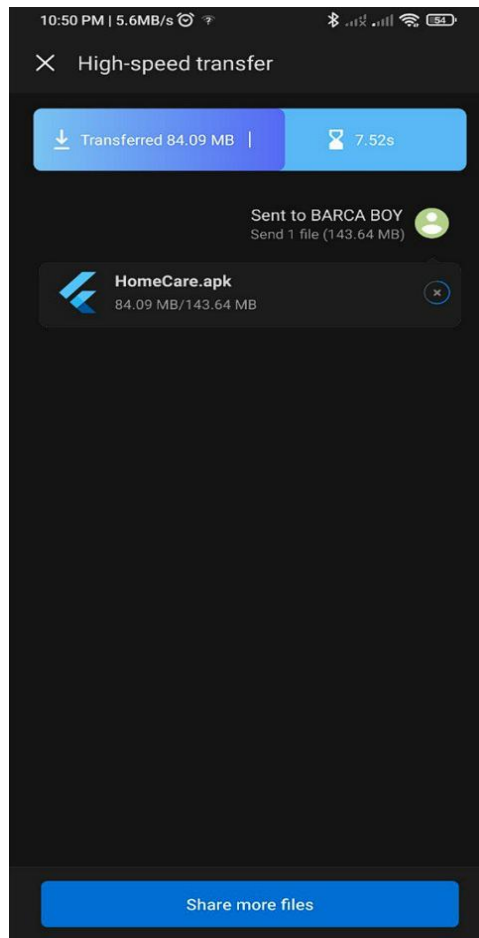
6.2 IMPLEMENTATION PROCEDURES

Software implementation refers to the complete installation of the package in its intended environment, as well as to the system's functionality and satisfaction of its intended applications. The software development project is frequently commissioned by someone who will not be using it.

6.3 HOSTING

Since the application uses cloud database (Firebase), the data can be accessed from any device. So the application can be run on multiple devices in realtime. The apk can be sent to any android device and can be run, if the device runs an ios, the specific ios app can be installed on the ios device and can be run.

Screenshot:



CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The proposed system introduces facility for customer to book simple household services online and view all information related to their order, make payment online and even view the list of all the previous booked services as well. The user can also register a complaint against a particular worker if the user is not satisfied with the worker. To conclude our application provides with the following features:

- Saves time
- Saves money in the long run
- On-demand Availability
- Increases Accessibility

7.2 FUTURE SCOPE

According to a New York Times article, the US online on-demand home services market is worth \$600 billion. It is expanding quickly and will have a CAGR of over 49% by 2021. Since they are also the biggest internet users, Gen Z makes up the greatest segment of customers for online, on-demand home services.

According to Forbes, 75% of the workforce will be made up of millennials who have grown up with the internet and smartphones by 2025. Technology has proven to be and would continue to prove to be a major role in beating the competition. As a result, the need for on-demand home maintenance services is expected to increase over the next few years.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”, 2009.
- Roger S Pressman, “*Software Engineering*”, 1994.
- PankajJalote, “*Software engineering: a precise approach*”, 2006.
- James lee and Brent ware Addison, “Open source web development with LAMP”, 2003
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

- draw.io
- <https://medium.com/>
- <https://console.firebase.com/>
- Pub.dev/
- <https://docs.flutter.dev/development/ui/widgets>

CHAPTER 9

APPENDIX

9.1 Sample Code

loginpage.dart

```
import 'package:MECARE/pages/forgotpassword.dart';
import 'package:MECARE/pages/userhome.dart';
import 'package:MECARE/pages/adminpage.dart';
import 'package:MECARE/pages/userprofile.dart';
import 'package:MECARE/pages/workerhome.dart';
import 'package:MECARE/pages/workerprofilescreens.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:MECARE/common/theme_helper.dart';
import 'package:fluttertoast/fluttertoast.dart';
import '../common/loading.dart';
import 'registration_page.dart';
import 'widgets/header_widget.dart';

class LoginPage extends StatefulWidget{
  const LoginPage({Key? key}): super(key:key);

  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  double _headerHeight = 250;
  bool visible = false;
  bool loading = false;
  final _formKey = GlobalKey<FormState>();
  final TextEditingController email = TextEditingController();
  final TextEditingController password = TextEditingController();

  //Firebase

  final _auth = FirebaseAuth.instance;
  final dbRef = FirebaseDatabase.instance.ref().child('users');
  final fire = FirebaseFirestore.instance.collection('users');
  String role = 'user';
  String status = 'Online';

  @override
  Widget build(BuildContext context) {
    return loading? Loading() : Scaffold(
      backgroundColor: Colors.white,
      body: SingleChildScrollView(
        child: Column(
          children: [
            Container(
              height: _headerHeight,
              child: HeaderWidget(_headerHeight, true,
                Icons.home), //let's create a common header widget
            ),
            SafeArea(
              child: Container(
                padding: EdgeInsets.fromLTRB(20, 10, 20, 10),
                margin: EdgeInsets.fromLTRB(20, 10, 20, 10),
                // This will be the login form
                child: Column(
```

```

        children: [
          Image.asset('assets/images/logo1.jpg'),
          SizedBox(height: 100.0),
          Form(
            autovalidateMode: AutovalidateMode.always,
            key: _formKey,
            child: Column(
              children: [
                Container(
                  child: TextFormField(
                    controller: email,
                    decoration:
ThemeHelper().textInputDecoration(
                      'Email', 'Enter your Email'),

                      validator: (value) {
                        if (value!.length == 0) {
                          return "Email cannot be empty";
                        }
                        if (!RegExp(
                          r"^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-
\.] )+\.([A-Za-z]{2,4})\$")
                            .hasMatch(value)) {
                          return ("Please enter a valid email");
                        } else {
                          return null;
                        }
                      },
                        onSave: (value) {
                          email.text = value!;
                        },
                      ),
                    decoration: ThemeHelper()
                      .inputBoxDecorationShaddow(),
                  ),
                SizedBox(height: 30.0),
                Container(
                  child: TextFormField(
                    controller: password,
                    obscureText: true,
                    decoration:
ThemeHelper().textInputDecoration(
                      'Password', 'Enter your password'),
                    validator: (value) {
                      RegExp regex = new RegExp(r'^.{6,}$');
                      if (value!.isEmpty) {
                        return "Password cannot be empty";
                      }
                      if (!regex.hasMatch(value)) {
                        return ("please enter valid password min.
6 character");
                      } else {
                        return null;
                      }
                    },
                      onSave: (value) {
                        password.text = value!;
                      },
                    ),
                    decoration: ThemeHelper()
                      .inputBoxDecorationShaddow(),
                  ),
                SizedBox(height: 15.0),
                Container(
                  decoration: ThemeHelper().buttonBoxDecoration(

```

```

        context),
        child: ElevatedButton(
          style: ThemeHelper().buttonStyle(),
          child: Padding(
            padding: EdgeInsets.fromLTRB(
              40, 10, 40, 10),
            child: Text('Sign In'.toUpperCase(),
              style: TextStyle(fontSize: 20,
                fontWeight: FontWeight.bold,
                color: Colors.white),),
          ),
          onPressed: () {
            setState(() {
              visible = true;
            });
            signIn(
              email.text, password.text);
          },
        ),
      ),
      Container(
        margin: EdgeInsets.fromLTRB(10, 20, 10, 5),
        //child: Text('Don\'t have an account?
Create'),

        child: Text.rich(
          TextSpan(
            children: [
              TextSpan(
                text: 'Forgot Password?',
                recognizer: TapGestureRecognizer()
                  ..onTap = () {
                    Navigator.push(context,
                      MaterialPageRoute(
                        builder: (context) =>
                          ResetScreen()));
                  },
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  color: Theme
                    .of(context)
                    .accentColor),
              ),
            ],
          ),
        ),
        SizedBox(height: 0.0),
        Container(
          child: Text.rich(
            TextSpan(
              children: [
                TextSpan(
                  text: "Don\'t have an account?
"),

                TextSpan(
                  text: 'Create',
                  recognizer: TapGestureRecognizer()
                    ..onTap = () {
                      Navigator.push(context,
                        MaterialPageRoute(
                          builder: (context) =>
                            RegistrationPage()));
                    },
                  style: TextStyle(
                    fontWeight: FontWeight.bold,
                    color: Theme

```

```

        .of(context)
        .accentColor),
    ),
  ],
),
),
),
),
),
),
),
),
),
),
);
}
void signIn(String email, String password) async {
  if (_formKey.currentState!.validate()) {
    setState(()=>loading = true);
    try {
      UserCredential userCredential = await FirebaseAuth.instance
        .signInWithEmailAndPassword(email: email, password: password);
      if (userCredential != null) {
        User? user = FirebaseAuth.instance.currentUser;
        final DocumentSnapshot snap = await
FirebaseFirestore.instance.collection('users').doc(user?.uid).get();
        setState(() {
          role = snap['role'];
          status = snap['status'];
          setState(() => loading = true);
        });
        if(status == 'Online'){
          if(role == 'User'){
            Navigator.of(context).push(new MaterialPageRoute(
              builder: (BuildContext context) => Userhome()));
          }
          else if(role == 'Admin'){
            Navigator.of(context).push(new MaterialPageRoute(builder:
(BuildContext context)=> AdminPage()));
          } else if(role == 'Worker') {
            Navigator.of(context).push(new MaterialPageRoute(
              builder: (BuildContext context) => WorkerHome()));
          }
        }else if(status == 'Offline'){

          if(role == 'User'){
            Navigator.of(context).push(new MaterialPageRoute(
              builder: (BuildContext context) => UserProfileScreen()));
          } else if(role == 'Worker') {
            Navigator.of(context).push(new MaterialPageRoute(
              builder: (BuildContext context) => WorkerProfileScreen1()));
          }
        }
        else if(status == 'Verification Pending'){
          loading = false;

          Fluttertoast.showToast(
            msg: "Verification not complete yet, please wait",
          );
        }
      }
      else if(role == 'Blah'){
        loading = false;
        Fluttertoast.showToast(

```

```

        msg: "Your application has been rejected.",
      );
    }
    else{
      loading = false;
      Fluttertoast.showToast(
        msg: "You don't have authorization to Login",
      );
    }
  }
} on FirebaseAuthException catch (e) {
  setState(() {
    loading = false;
  });
  if (e.code == 'user-not-found') {
    print("No user found with this email");

    Fluttertoast.showToast(
      msg: "No user found with this email",
    );
  } else if (e.code == 'wrong-password') {
    print("You have entered the Wrong Password");

    Fluttertoast.showToast(
      msg: "You have entered the Wrong Password",
    );
  }
}
} else {
  loading = false;
}
}
}

```


registrationpage.dart

```
import 'package:HEMOCARE/pages/login_page.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:HEMOCARE/common/theme_helper.dart';
import 'package:HEMOCARE/pages/widgets/header_widget.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:form_field_validator/form_field_validator.dart';
import 'package:hexcolor/hexcolor.dart';
import 'package:http/http.dart' as http;
import 'package:firebase_database/firebase_database.dart';
import 'login_page.dart';

class RegistrationPage extends StatefulWidget {
  @override
  State<RegistrationPage> createState() {
    return _RegistrationPageState();
  }
}

class _RegistrationPageState extends State<RegistrationPage> {

  final _formKey = GlobalKey<FormState>();

  TextEditingController fname = new TextEditingController();
  TextEditingController phone = new TextEditingController();
  TextEditingController email = new TextEditingController();
  TextEditingController password = new TextEditingController();
  TextEditingController place = new TextEditingController();
  TextEditingController confirmpassword = new TextEditingController();

  late String txtname, txtphone, txtemail, txtpassword, txtplace, txtgender='',
  txtrole='';

  final dbRef = FirebaseDatabase.instance.ref().child('users');
  CollectionReference collectionReference =
  FirebaseDatabase.instance.collection('users');
  final _auth = FirebaseAuth.instance;

  void register() async {
    try {
      final newUser = _auth.createUserWithEmailAndPassword(
        email: txtemail, password: txtpassword)
        .then((value) {

FirebaseFirestore.instance.collection('users').doc(value.user?.uid).set(
          {"email": value.user?.email,
            "name": txtname,
            "place": txtplace,
            "phone": txtphone,
            "role": txtrole,
            "age": '',
            "profession": '',
            "aadhar_no": '',
            "status": 'Offline',
```

```

"profilepicURL": 'https://cdn.pixabay.com/photo/2015/10/05/22/37/blank-profile-
picture-973460_960_720.png',
    "gender": txtgender));
    },
    );
    if (newUser != null) {
        Navigator.of(context).push(new MaterialPageRoute(builder: (BuildContext
context)=>LoginPage()));
    }
} catch (e) {
    print (e);
    SnackBar(
        content: Text("Username Already exists"),
        backgroundColor: Colors.teal,
    );
    Fluttertoast.showToast(msg: e.toString());
    print(e);
}
}
@override
Widget build(BuildContext context) {
    return Scaffold(
        backgroundColor: Colors.white,
        body: SingleChildScrollView(
            child: Stack(
                children: [
                    Container(
                        height: 150,
                        child: HeaderWidget(150, false, Icons.person_add_alt_1_rounded),
                    ),
                    Container(
                        margin: EdgeInsets.fromLTRB(25, 50, 25, 10),
                        padding: EdgeInsets.fromLTRB(10, 0, 10, 0),
                        alignment: Alignment.center,
                        child: Column(
                            children: [
                                Form(
                                    //autovalidate: true,
                                    autovalidateMode: AutovalidateMode.always,
                                    key: _formKey,
                                    child: Column(
                                        children: [

                                            SizedBox(height: 110,),
                                            Container(
                                                child: TextFormField(
                                                    controller: fname,
                                                    decoration:
ThemeHelper().textInputDecoration('Name', 'Enter your name'),
                                                    onSave: (value) {
                                                        txtname = value!;
                                                    },
                                                    keyboardType: TextInputType.text,
                                                    validator: (val){
                                                        if (val!.length == 0) {
                                                            return "Name cannot be empty";
                                                        }
                                                        if(!(val.isEmpty) && !RegExp(r"^[a-zA-Z][a-zA-
Z\s]{0,20}[a-zA-Z]$").hasMatch(val)){
                                                            return "Enter a valid Name";
                                                        }
                                                        return null;
                                                    },
                                                ),
                                                decoration:
ThemeHelper().inputBoxDecorationShaddow(),
                                            ),

```

```

        SizedBox(height: 30,),

        Container(
          child: TextFormField(
            controller: email,
            decoration: ThemeHelper().textInputDecoration("E-
mail address", "Enter your email"),
            keyboardType: TextInputType.emailAddress,
            onSave: (value) {
              txtemail = value!;
            },
            validator: (value) {
              if (value!.length == 0) {
                return "Email cannot be empty";
              }
              if (!RegExp(r"^([A-Za-z0-9_\-\.])+\@([A-Za-z0-
9_\-\.])+\.([A-Za-z]{2,4})$"))
                .hasMatch(value)) {
                return ("Please enter a valid email");
              } else {
                return null;
              }
            },
          ),
          decoration:
ThemeHelper().inputBoxDecorationShaddow(),
        ),
        SizedBox(height: 20.0),

        DropdownButtonFormField(
          decoration: InputDecoration(
            labelText: 'Gender',
            fillColor: Colors.white,
            filled: true,
            contentPadding: EdgeInsets.fromLTRB(20, 10, 20,
10),
            focusedBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(2.0), borderSide: BorderSide(color: Colors.grey)),
            enabledBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(2.0), borderSide: BorderSide(color:
Colors.grey.shade400)),
            errorBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(2.0), borderSide: BorderSide(color: Colors.red, width:
2.0)),
            focusedErrorBorder:
OutlineInputBorder(borderRadius: BorderRadius.circular(2.0), borderSide:
BorderSide(color: Colors.red, width: 2.0)),
          ),
          validator: (value) {
            if (value == null) {
              return "Select One";
            }
          },
          value: txtgender.isNotEmpty ? txtgender : null,
          items: <String>['Male', 'Female', 'Others']
            .map<DropdownMenuItem<String>>((String value)
            {
              return DropdownMenuItem<String>(
                value: value,
                child: Text(value),);
            }).toList(),
          onChanged: (value) {
            setState(() {
              txtgender = value.toString();
            });
          },
        ),
        SizedBox(height: 20.0),
        Container(

```

```

        child: TextFormField(
          controller: place,
          decoration:
ThemeHelper().textInputDecoration('Address', 'Enter your Address'),
          keyboardType: TextInputType.text,
          onSave: (value) {
            txtplace = value!;
          },
          validator: (val){
            if (val!.length == 0) {
              return "Address cannot be empty";
            }
          },
        ),
        decoration:
ThemeHelper().inputBoxDecorationShaddow(),
      ),
      SizedBox(height: 20.0),
      Container(
        child: TextFormField(
          controller: phone,
          decoration: ThemeHelper().textInputDecoration(
            "Mobile Number",
            "Enter your mobile number"),
          keyboardType: TextInputType.phone,
          onSave: (value) {
            txtphone = value!;
          },
          validator: (val) {
            if (val!.length == 0) {
              return "Phone Number cannot be empty";
            }
            if(! (val.isEmpty) && !RegExp(r"^[0-
9]{10}$").hasMatch(val)){
              return "Enter a valid phone number";
            }
            return null;
          },
        ),
        decoration:
ThemeHelper().inputBoxDecorationShaddow(),
      ),
      SizedBox(height: 20.0),
      Container(
        child: TextFormField(
          controller: password,
          obscureText: true,
          decoration: ThemeHelper().textInputDecoration(
            "Password*", "Enter your password"),
          onSave: (value) {
            txtpassword = value!;
          },
          validator: (val) {
            if (val!.length == 0) {
              return "Password cannot be empty";
            }
            if(! (val.isEmpty) && !RegExp(r"^(?=.*[A-Za-
z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{8,}$").hasMatch(val)){
              return "Please enter a stronger password";
            }
            return null;
          },
        ),
        decoration:
ThemeHelper().inputBoxDecorationShaddow(),
      ),
      SizedBox(height: 15.0),
      Container(

```

```

        child: TextFormField(
          controller: confirmPassword,
          obscureText: true,
          decoration: ThemeHelper().textInputDecoration(
            "Confirm password*", "Re-enter your password"),
          validator: (val){
            if(val!.isEmpty)
            {
              return 'Please re-enter password';
            }
            print(password.text);
            print(confirmPassword.text);
            if(password.text!=confirmPassword.text){
              return "Password does not match";
            }

            return null;
          },

        ),
        decoration:
ThemeHelper().inputBoxDecorationShadow(),
      ),
      SizedBox(height: 20.0),

      DropdownButtonFormField(
        decoration: InputDecoration(
          labelText: 'SignUp as',
          fillColor: Colors.white,
          filled: true,
          contentPadding: EdgeInsets.fromLTRB(20, 10, 20,
10),
          focusedBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(2.0), borderSide: BorderSide(color: Colors.grey)),
          enabledBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(2.0), borderSide: BorderSide(color:
Colors.grey.shade400)),
          errorBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(2.0), borderSide: BorderSide(color: Colors.red, width:
2.0)),
          focusedErrorBorder:
OutlineInputBorder(borderRadius: BorderRadius.circular(2.0), borderSide:
BorderSide(color: Colors.red, width: 2.0)),
        ),
        validator: (value) {
          if (value == null) {
            return "Select One";
          }
        },
        value: txtrole.isNotEmpty ? txtrole : null,
        items: <String>['User', 'Worker']
          .map<DropdownMenuItem<String>>((String value)
          {
            return DropdownMenuItem<String>(
              value: value,
              child: Text(value),);
          }).toList(),
        onChanged: (value){
          setState(() {
            txtrole = value.toString();
          });
        },
      ),

      SizedBox(height: 20.0),
      Container(

        decoration:

```

```

ThemeHelper().buttonBoxDecoration(context),
    child: ElevatedButton(
      style: ThemeHelper().buttonStyle(),
      child: Padding(
        padding: const EdgeInsets.fromLTRB(40, 10, 40,
10),


        child: Text(
          "SignUp".toUpperCase(),
          style: TextStyle(
            fontSize: 20,
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
      ),
      onPressed: () {
        if (_formKey.currentState!.validate()) {
          _formKey.currentState!.save();
          register();
          //register1();
        }
      },
    ),
    SizedBox(height: 20.0),
    Container(
      margin: EdgeInsets.fromLTRB(10,20,10,20),
      //child: Text('Don\'t have an account? Create'),
      child: Text.rich(
        TextSpan(
          children: [
            TextSpan(text: "Already have an account?
"),


            TextSpan(
              text: 'Login',
              recognizer: TapGestureRecognizer()
                ..onTap = (){
                  Navigator.push(context,
MaterialPageRoute(builder: (context) => LoginPage()));
                },
              style: TextStyle(fontWeight:
FontWeight.bold, color: Theme.of(context).accentColor),
            ),
          ],
        ),
      ),
    ),
  );
}
}

```

9.2 Screen Shots


Common Screens





SIGN IN

[Forgot Password?](#)
Don't have an account? [Create](#)




Gender

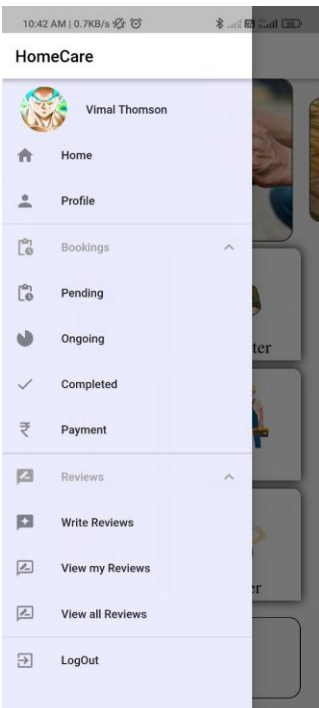
Sign Up as

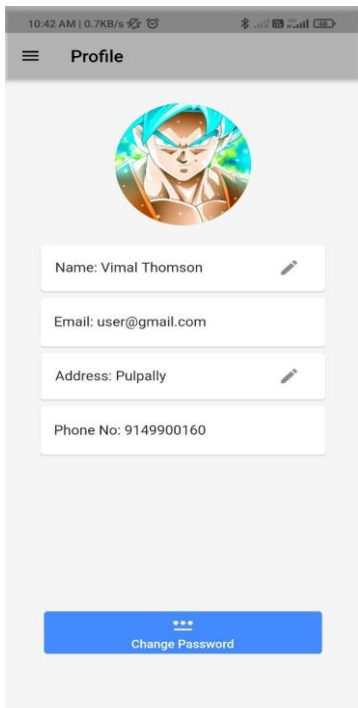
SIGNUP

Already have an account? [Login](#)

User Screens







10:42 AM | 5.4KB/s

Change Password

Change your Password

Enter new Password*

Confirm new password*

Change Password

10:42 AM | 6.6KB/s

Update your details.


Name
Vimal Thomson

Address
Pulpally

UPDATE

10:40 AM | 3.6KB/s

Carpenter Services



HomeCare provides skilled and reliable carpenters that provide quality workmanship at cost you can easily afford. HomeCare offers a comprehensive range of carpentry services from general carpentry, door repair, window repair, lock installation, new furniture making, furniture repair & installation.

Work Description


Can't be empty

Address

Address cannot be empty

10:41 AM | 0.7KB/s

Carpenter Services



HomeCare provides skilled and reliable carpenters that provide quality workmanship at cost you can easily afford. HomeCare offers a comprehensive range of carpentry services from general carpentry, door repair, window repair, lock installation, new furniture making, furniture repair & installation.

Work Description
broken chair

Address
H. No. 11116, Kanjirappally P. O

Work Type
Furniture repair

10:41 AM | 1.6KB/s

Pending Requests

Work : Carpenter, Furniture repair
Address : H. No. 11116, Kanjirappally P. O
Date : 7-18-2022 10:40

Work Request : broken chair
Instructions : None

Cancel Request

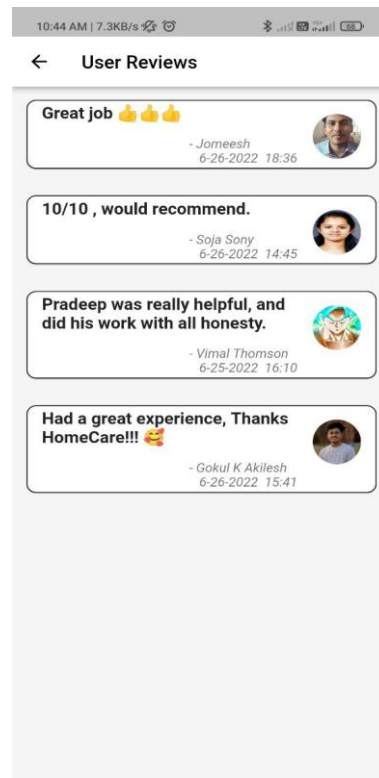
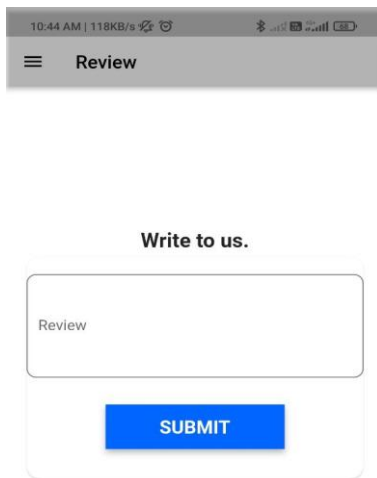
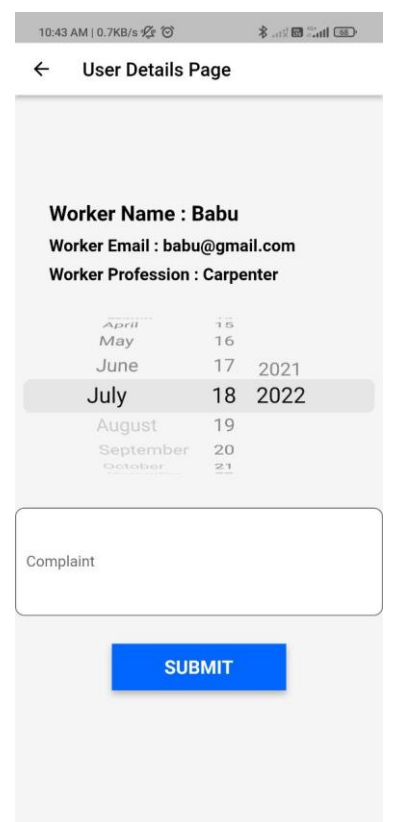
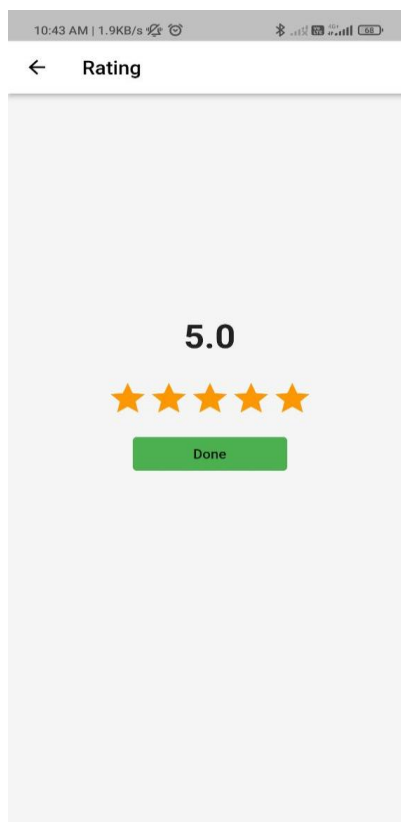
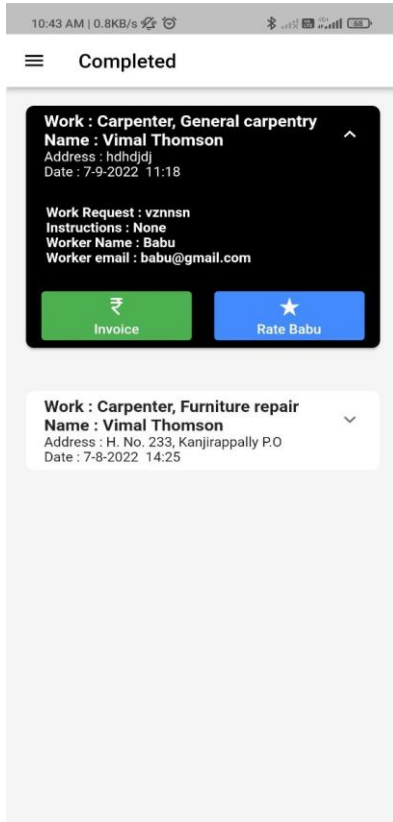
10:42 AM | 2.3KB/s

Work : Carpenter, Furniture repair
Name : Vimal Thomson
Address : H. No. 11116, Kanjirappally P. O
Date : 7-18-2022 10:40

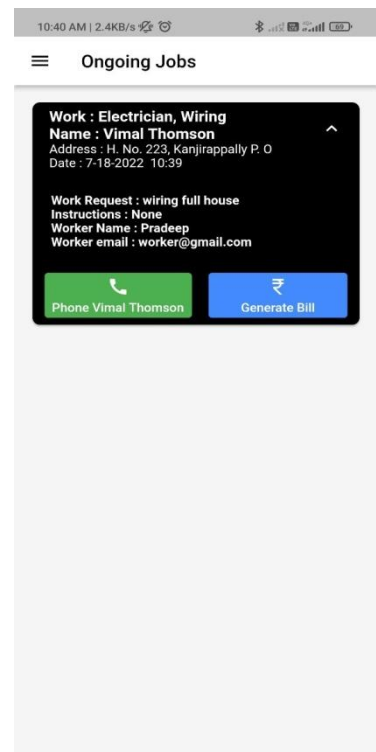
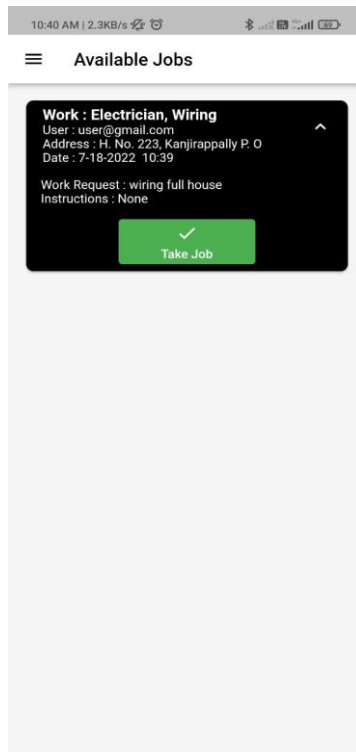
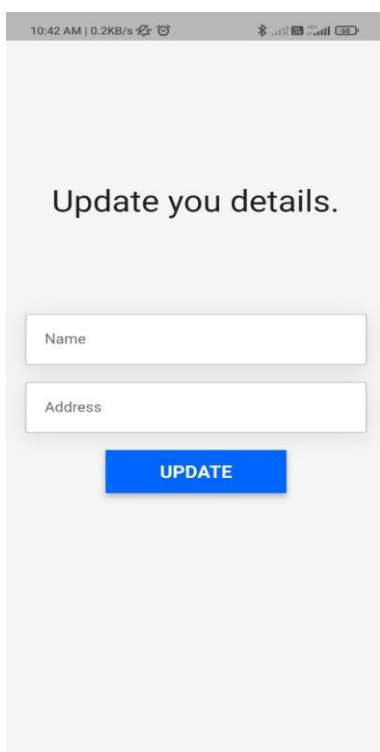
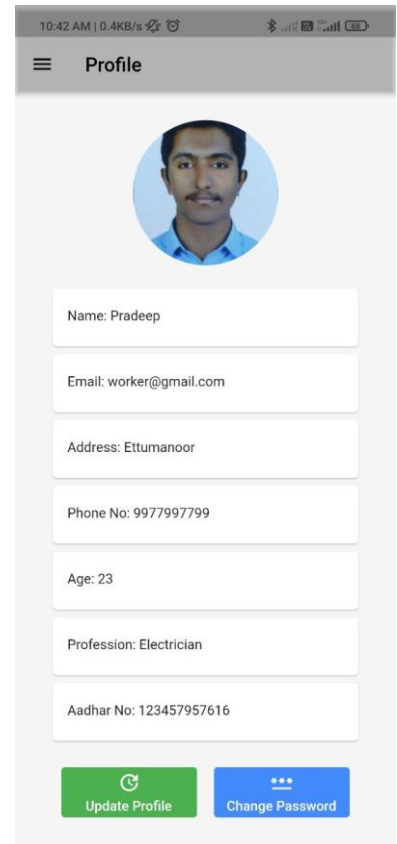
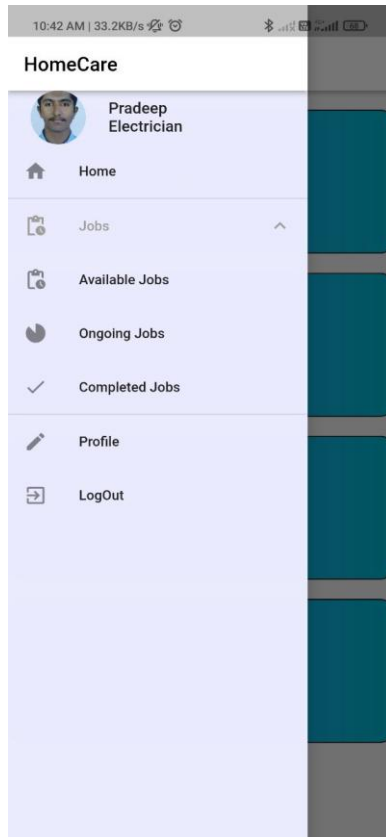
Work Request : broken chair
Instructions : None
Worker Name : Babu
Worker email : babu@gmail.com

Phone Babu

Raise Complaint



Worker Screens



10:40 AM | 1.0KB/s

← **Generate Bill**

Generate Bill

Total work time(in hrs)

12

Other Costs

140

Generate Bill

10:40 AM | 15.5KB/s

← **Compose**

From vt9796@gmail.com

To user@gmail.com

HomeCare Bill

Sir/Mam your work has been successfully completed by Pradeep, the total cost of the work is ₹ 1640, in which labour expense is ₹ 1500 and the cost of other equipments, instruments and other misc costs is ₹ 140. Kindly login to you HomeCare app and complete the payment.

Team HomeCare.

Admin Screens

