**Services Provider**

****

**Session (2017-2021)**

**Program**BS (SE)

**Submitted By**

Asfand Yar Mumtaz 5482

Ethisham Ahmad Khan 5493

**Project Supervisor**

Miss Sadia Waheed

Lecturer

**Department of Computer Science**

**Abbottabad University of Science & Technology**

**APPROVAL**

This is to certify that Asfand Yar Mumtaz (5482) and Ethisham Ahmad Khan (5493) have worked on and completed their Software Project at, Department of Computer Science, Abbottabad University of Science & Technology in partial fulfillment of the requirement for the degree of BSCS under the guidance and supervision of Miss Sadia Waheed.

In our opinion, it is satisfactory and up to the mark and therefore fulfills the requirements of BSCS.

**Supervisor**

Miss Sadia Waheed \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lecturer (Signature)

Department of Computer Science

**FYP Committee Convener:**

<<Convener Name>> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Designation (Signature)

Department of Computer Science

**Internal Examiner**

<<Name>> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature)

**External Examiner**

<<External Name>> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Designation (Signature)

Department of Computer Science

**CHAIRMAN:**

Dr. Muhammad Naeem \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HOD (Signature)

Department of Computer Science

**EXORDIUM**

In the name of Allah, the Compassionate, the Merciful. Praise be to Allah, Lord of Creation, The Compassionate, the Merciful, King of Judgment-day!

You alone we worship, and to You alone we pray for help, Guide us to the straight path The path of those who You have favored, Not of those who have incurred Your wrath, Nor of those who have gone astray.

**DEDICATION**

**Dedicated to our Parents and Teachers**

**ACKNOWLEDGEMENT**

With the great name of ALLAH, the most gracious and merciful, who gifted us blessings, strength and mental powers, without which we could not complete this project. Before we go into depth of the things we would like to add a few heartfelt words for the people who were part of this project in numerous ways. People who gave unending support right from the stage project idea were conceived. In particular we are extremely indebted our teacher and supervisor of this project, and also we are thankful to our staff and fellows. We are also very thankful to senior for their guidance & help. We are extremely thankful to our beloved Parents and family whose prayers and continuous encouragement made the successful completion of this project possible.

**PREFACE**

This Project Report has been prepared in partial fulfillment of the requirement for the Subject: Computer Science of the program BS(CS) in the academic year 2021. The blend of learning and knowledge acquired during our practical studies at the company is presented in this Project Report.

**TABLE OF CONTENTS**

[CHAPTER 1 Gathering & Analyzing Info 1](#_Toc50449811)

[1.1 Introduction 2](#_Toc50449812)

[1.2 Purpose 2](#_Toc50449813)

[1.3 Scope 2](#_Toc50449814)

[1.4 Definitions, acronyms and abbreviations 2](#_Toc50449815)

[1.5 Use cases and usage scenarios 3](#_Toc50449816)

[1.5.1 Use Case Diagram 3](#_Toc50449817)

[1.5.2 Usage Scenarios 4](#_Toc50449818)

[1.6 Tools Used 9](#_Toc50449819)

[CHAPTER 2 Theoretical Background 11](#_Toc50449820)

[2.1 Introduction 12](#_Toc50449821)

[2.2 Limitation of Existing System 12](#_Toc50449822)

[2.2.1 Inconsistency 12](#_Toc50449823)

[2.2.2 Time Consumption 12](#_Toc50449824)

[2.2.3 Man Power 12](#_Toc50449825)

[2.2.4 Not Up to Date 12](#_Toc50449826)

[2.2.5 Takes Up a Lot of Space 13](#_Toc50449827)

[2.2.6 Hard to Make Changes 13](#_Toc50449828)

[2.2.7 Lack of Security 13](#_Toc50449829)

[CHAPTER 3 Proposed System 14](#_Toc50449830)

[3.1 Introduction 15](#_Toc50449831)

[3.2 Modules 15](#_Toc50449832)

[3.3 Functional Requirements 15](#_Toc50449833)

[3.4Non Functional / Supplementary requirements 16](#_Toc50449834)

[1.4.1 Usability 16](#_Toc50449835)

[3.4.2 Reliability 16](#_Toc50449836)

[3.4.3 Supportability 17](#_Toc50449837)

[3.4.4 System Requirements 17](#_Toc50449838)

[3.5 Feasibility Study 17](#_Toc50449839)

[3.5.1 Technical feasibility 18](#_Toc50449840)

[3.5.2 Operational feasibility 18](#_Toc50449841)

[3.5.3 Economic feasibility 19](#_Toc50449842)

[3.6 Methodology 19](#_Toc50449843)

[3.6.1 Chosen Methodology 19](#_Toc50449844)

[3.7 Reasons for Chosen Methodology 20](#_Toc50449845)

[3.8 Work Plan 20](#_Toc50449846)

[3.9 Project Structure 21](#_Toc50449847)

[3.9.1 Team Structure 21](#_Toc50449848)

[3.9.3 Project Schedule 21](#_Toc50449849)

[CHAPTER 4 System Design 22](#_Toc50449850)

[4.1 Introduction 23](#_Toc50449851)

[4.2 Purpose 23](#_Toc50449852)

[4.3 Scope 24](#_Toc50449853)

[4.4 Definitions, acronyms and abbreviations 24](#_Toc50449854)

[4.5 Architectural Representation (Architecture Diagram) 24](#_Toc50449855)

[4.6 Dynamic Model: Sequence Diagrams 26](#_Toc50449856)

[4.7 Class Diagram 28](#_Toc50449857)

[4.8 Activity Diagram 29](#_Toc50449858)

[4.9 DFD 30](#_Toc50449859)

[4.9.1 DFD Level 0 31](#_Toc50449860)

[4.9.2 DFD Level 1 31](#_Toc50449861)

[4.9.3 DFD Level 2 32](#_Toc50449862)

[4.10 ERD 32](#_Toc50449863)

[4.11 Normalization 33](#_Toc50449864)

[4.12 Data Flow Chart 33](#_Toc50449865)

[4.14 Graphical User Interfaces 35](#_Toc50449866)

[CHAPTER 5 System Testing 39](#_Toc50449867)

[5.1 Introduction 40](#_Toc50449868)

[5.2 Purpose of Testing 40](#_Toc50449869)

[5.3 Objective of Testing 40](#_Toc50449870)

[5.4 Testing Techniques 40](#_Toc50449871)

[5.4.1 Black Box Testing 41](#_Toc50449872)

[5.4.2 White Box Testing 41](#_Toc50449873)

[5.5 Unit Testing 42](#_Toc50449874)

[5.5 Integration Testing 42](#_Toc50449875)

[5.7 System Testing 42](#_Toc50449876)

[5.7.1 Test Cases 42](#_Toc50449877)

[5.7.2 Trace-ability 44](#_Toc50449878)

[Chapter 6 Conclusions 45](#_Toc50449879)

[Appendix REFERENCES 47](#_Toc50449880)

**LIST OF FIGURES**

Figure 1.1 Use Case Diagram 3

Figure 3.1 Types of Feasibility 18

Figure 3.2: Incremental Model 20

Figure 3.3: Project Schedule 21

Figure 4.1: Architecture Diagram 25

Figure 4.2: Sequence Diagram 27

Figure 4.3: Class Diagram 29

Figure 4.4: Activity Diagram 30

Figure 4.5: DFD Level 0 31

Figure 4.6: DFD Level 1 30

Figure 4.7: DFD Level 2 32

Figure 4.8: ERD 33

Figure 4.9: Flowchart 34

Figure 4.10: Home Page 35

Figure 4.11: View Service Provider 36

Figure 4.12: Service Provider Profile 37

Figure 4.13: Service Detail 38

Figure 5.1: Black Box Testing 41

Figure 5.2: White Box Testing 41

**LIST OF TABLES**

Table 1.1 Use Case 01 4

Table 1.2 Use Case 02 5

Table 1.3 Use Case 03 6

Table 1.4 Use Case 04 7

Table 1.5 Use Case 05 8

Table 1.6 Use Case 06 9

Table 3.1: Team Structure 21

Table 5.1: Test Cases 44

Table 5.2: Traceability Matrix 44

# CHAPTER 1 Gathering and Analyzing Information

## 1.1 Introduction

In our Country there is problem that if somebody needs a service or a task to be done for his personal use or for profession, he was unable to find a good service provider in his location because he was unable to select a service provider who is good for his job because he was unaware of good service providers around his location and if they found a good service provider, they were unable to afford them because of their high prices and demands.

So, with the help of our product user who needs a service can easily get a good service around his location. A new user simply needs to sign up and then log in and then after that he can hire a service provider by simply searching the service provider and then by selecting the service which he needs

Our target market is users who need any service and service providers who can provide any service. Through our web-based application users will find available service provider in their area and easily gets the details about service provider by checking his profile, services, portfolio and reviews without asking anyone.

## 1.2 Purpose

The Proposed system is Service-Finder, which provides facility to users to search their service easily in nearest location in any area also give facility to service provider to enhance their selling services. Our product will help the user to find any service in any area or nearest his location.

## 1.3 Scope

To enhance the proposed work for other similar projects like food-ordering app or for fruit & vegetables ordering app or for door to door services.

## 1.4 Definitions, acronyms and abbreviations

**Abbreviations**

* ID Identification
* AC Air Conditioner

## 1.5 Use cases and usage scenarios

### 1.5.1 Use Case Diagram

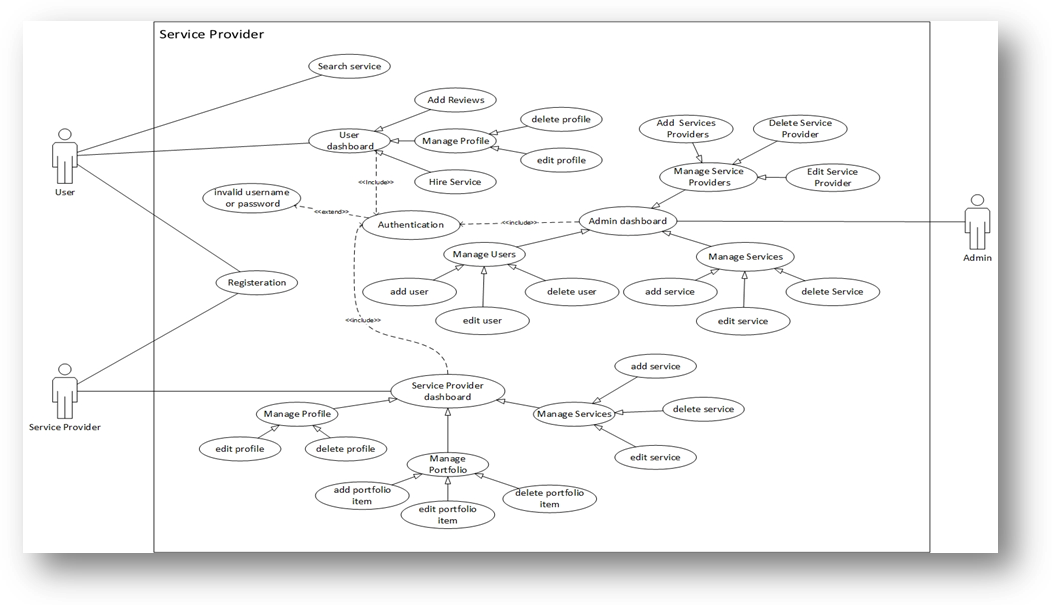


Figure 1.1: Use Case Diagram

### 1.5.2 Usage Scenarios

|  |  |  |
| --- | --- | --- |
| **Use Case Title** | Login | |
| **Abbreviated Title** | Customer & Service Provider | |
| **Use Case Id** | Use Case 01 | |
| **Requirement Id** | Registration ID | |
| **Description:** Customer & Service Provider attempts to login to the system to perform required operations. | | |
| **Pre-Conditions:** Customer & Service Provider must have an account. | | |
| **Task Sequence** | | **Exceptions** |
| 1. User Request for login Form. | | 1- An exception occurs when Form not found. |
| 1. System will ask the user to enter his user ID and password | | 2- The system generates an implicit exception, and gives message to user to provide his correct name and password. |
| 1. If the User ID and password is correct the system will allow the user to access the required Forms/modules. | | 3. If User Record not found system provides exception. |
| **Post Conditions:**   1. Customer & Service Provider successfully logged in to the system. | | |
| **Unresolved issues:** None | | |
| **Authority:** | | |
| **Modification history:**  **Author:** Asfand Yar Mumtaz & Ethisham Ahmad Khan | | |

Table 1.1 Use Case 01

|  |  |  |
| --- | --- | --- |
| **Use Case Title** | View Services | |
| **Abbreviated Title** | View Services | |
| **Use Case Id** | Use Case 02 | |
| **Requirement Id** | Customer | |
| **Description:** Customer View the Services. | | |
| **Pre-Conditions:** Customer must be login into their account. | | |
| **Task Sequence** | | **Exceptions** |
| 1. Click on Services link. | | 1- Invalid filter |
| 1. Services Saved are shown. | | 2- Record Not Found. |
| **Post Conditions:**   1. Customer can request for Services request. | | |
| **Unresolved issues:** None | | |
| **Authority:** | | |
| **Modification history:**  **Author:** Asfand Yar Mumtaz & Ethisham Ahmad Khan | | |

Table 1.2 Use Case 02

|  |  |  |
| --- | --- | --- |
| **Use Case Title** | Add Services | |
| **Abbreviated Title** | Add Services | |
| **Use Case Id** | Use Case 03 | |
| **Requirement Id** | Administrator | |
| **Description:** Administrator enter the Services detail. | | |
| **Pre-Conditions:** Administrator must be logged on to system. | | |
| **Task Sequence** | | **Exceptions** |
| 1. Click on Add Services | | 1- Exception occurs if user enter invalid data. |
| 1. Enter Services Detail. | | 2- Exception occur if user did not provides all values. |
| 3. Click on save button. | |  |
| **Post Conditions:**   1. Services are visible by the customers. | | |
| **Unresolved issues:** None | | |
| **Authority:** | | |
| **Modification history:**  **Author:** Asfand Yar Mumtaz & Ethisham Ahmad Khan | | |

Table 1.3 Use Case 03

|  |  |  |
| --- | --- | --- |
| **Use Case Title** | Add Service Request | |
| **Abbreviated Title** | Add Service Request | |
| **Use Case Id** | Use Case 04 | |
| **Requirement Id** | Customer | |
| **Description:** Customer request for services. | | |
| **Pre-Conditions:** Customer must be logged on to system. | | |
| **Task Sequence** | | **Exceptions** |
| 1. Select Service | | 1. Exception occur if record not Exist |
| 1. Click on Order Now Link | |  |
| 1. Fill Order Form | |  |
| 1. Click on Save | |  |
| **Post Conditions:**  Designer View the order placed by the customer. | | |
| **Unresolved issues:** None | | |
| **Authority:** | | |
| **Modification history:**  **Author:** Asfand Yar Mumtaz & Ethisham Ahmad Khan | | |

Table 1.4 Use Case 04

|  |  |  |
| --- | --- | --- |
| **Use Case Title** | Change Password | |
| **Abbreviated Title** | Change Password | |
| **Use Case Id** | Use Case 05 | |
| **Requirement Id** | Designer & Customer | |
| **Description:** Designer & Customer can change their user account password. | | |
| **Pre-Conditions:** Designer & Customer must be logged in into the system. | | |
| **Task Sequence** | | **Exceptions** |
| 1. Click on Change Password Link | | 1- Exception message will be provided if form not loaded correctly. |
| 1. Enter Old and New Password. | | 1. Exception occur if Incorrect Information is Provided |
| 1. Click on Update Button | |  |
| **Post Conditions:**  Designer & Customer must use new password for login. | | |
| **Unresolved issues:** None | | |
| **Authority:** | | |
| **Modification history:**  **Author:** Asfand Yar Mumtaz & Ethisham Ahmad Khan | | |

Table 1.5 Use Case 05

|  |  |  |
| --- | --- | --- |
| **Use Case Title** | Logout | |
| **Abbreviated Title** | Logout from system | |
| **Use Case Id** | Use Case 06 | |
| **Requirement Id** | Designer & Customer | |
| **Description:** Designer & Customer of system attempt to logout from system. | | |
| **Pre-Conditions:** Registered user must have logged in to system. | | |
| **Task Sequence** | | **Exceptions** |
| 1. User select logout link | |  |
| **Post Conditions:**  User successfully logout from application | | |
| **Unresolved issues:** None | | |
| **Authority:** | | |
| **Modification history:**  **Author:** Asfand Yar Mumtaz & Ethisham Ahmad Khan | | |

Table 1.6 Use Case 06

## 1.6 Tools Used

**Front End:**

* HTML
* CSS
* JavaScript
* jQuery
* Bootstrap
* React JS

**Back End:**

* Node JS
* Express JS
* SQL

**DBMS**

* MySQL

**Graphics**

* Photoshop CS6

**Project Documentation**

* MS Word 2015

# CHAPTER 2 Theoretical Background

## 2.1 Introduction

In this modern technological age and ever growing technology where home services needs is the major factor and huge need for the people and user can book home services by using Services Provider to save their valued time and get their work done on time and the service provider earns a huge capital by doing work. But unfortunately it becomes very hectic when a needy person doesn’t get this in time. To find a Professional at time of need is now very hard and it is also a headache for Professionals to find job. So to resolve this problem we have an idea to make an app to solve this problem technologically.

## 2.2 Limitation of Existing System

The current system is manual all record kept manually in registers or in traditional file system. The user store each record in different registers which is very exhaustive task, this involve a lot of labor work. The manual system is easy if number of record are limited but it is complicated if number of transactions increases.

## 2.2.1 Inconsistency

In existing system there is an Inconsistency in record management. There is no proper relationship in between the data and information.

## 2.2.2 Time Consumption

Existing system require more time because there is manual record keeping system used by the user.

## 2.2.3 Man Power

Existing system takes lot of man power to accomplish the task. Lack of modern technology also causes the problems.

## 2.2.4 Not Up to Date

Existing system does not keep updated information.

### 2.2.5 Takes Up a Lot of Space

The biggest downfall to manual document filing is the amount of space it can take up. While at first your business will be small enough that it’s not a huge deal, once you start growing you are going to want to find a new way to store files. Otherwise you might find yourself taking up rooms just to fit the files in. This also means you have to go out of your way to hunt down a file or a client. This is just one big headache you don’t need. It interrupts your productivity levels.

### 2.2.6 Hard to Make Changes

When you are working with paper documents it is much harder to make changes. Every time you want to make a change you will have to make a copy, so you don’t destroy the original with any edits or comments you might add. This means the editing process is more time consuming than if you were working with digital copies.

### 2.2.7 Lack of Security

Paper document filing can be less secure than electronic filing systems. Misplaced documents can easily be placed in the wrong hands. Clients expect their information to be secure in your hands. If you can’t keep this safe, you are at risk for losing them. A cabinet filled with files is way easier to access than a computer which requires a password and credentials to get into.

# CHAPTER 3 Proposed System

## 3.1 Introduction

The Proposed system is Service-Finder, which provides facility to users to search their service easily in nearest location in any area also give facility to service provider to enhance their selling services. Our product will help the user to find any service in any area or nearest his location.

## 3.2 Modules

* Admin Dashboard
* Service Provider Dashboard
* User Dashboard
* Registration
* Search Service
* Login
* User can hire services and can add reviews on it
* User can edit or delete their profiles
* Service Provider can edit and deletes their profiles
* Service Provider manages their portfolio by adding, modifying, deleting their portfolio items.
* User can easily find the service.
* Search engine display those services which are nearest the user's location

## 3.3 Functional Requirements

The project deals with the following modules of proposed system.

**Admin Dashboard.**

Through Admin Dashboard we can check record of all the service providers and users who are registered on our app.

**Service Provider Dashboard.**

Thorough Service Provider Dashboard we give access service provider to manage their profile and portfolio by adding the new service and portfolio items or editing the previous one. Through this dashboard service provider can also chat with users.

**User Dashboard.**

Thorough user dashboard we give access to users to manage their profile and to search for service providers and their services. Through this dashboard users can chat with service provider. Users can select services and can add reviews on it.

**Registration.**

Through Registration user and service provider must register them before accessing their Dashboards.

**Search Service.**

By using our project user can search for a service or a service provider near his location.

**Login.**

Through Login our system verifies the user through Authentication then checks its role either it is Admin Service Provider or User and then he is redirected to their dashboard.

## 3.4 Non Functional / Supplementary requirements

### 1.4.1 Usability

Usability is the degree to which the software is easy to use as indicated by the following substitutes. Usability minimizes the efforts required to learn, operate, prepare, input and interpret output of a program. It is the combination of fitness for the purpose, ease of use and ease of learning that makes a product effective. It focuses on determining if the product is easy to learn, satisfying to use and contains functionality that the users desire.

### 3.4.2 Reliability

The proposed system is less error prone. Human mistakes occurred during day to day jobs are minimized due to minimum data entry.

### 3.4.3 Supportability

The proposed system has all the features that a web project should have. It supports all type of client platforms for browsing.

### 3.4.4 System Requirements

* For Hosting
  + Windows / LINUX Server with SQL Database
* For Client
  + Web Browser

## 3.5 Feasibility Study

Feasibility is defined as the practical extent to which a project can be performed successfully. To evaluate feasibility, a feasibility study is performed, which determines whether the solution considered to accomplish the requirements is practical and workable in the software. Information such as resource availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered during the feasibility study. The objective of the feasibility study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards.

Various other objectives of feasibility study are listed below.

* To analyze whether the software will meet organizational requirements.
* To determine whether the software can be implemented using the current technology and within the specified budget and schedule.
* To determine whether the software can be integrated with other existing software.

**Types of Feasibility**

Various types of feasibility that are commonly considered include technical feasibility, operational feasibility, and economic feasibility.

Types of Feasibility

Technical Feasibility

Operational Feasibility

Economical Feasibility

Figure 3.1 Types of Feasibility

### 3.5.1 Technical feasibility

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

* Analyzes the technical skills and capabilities of the software development team members.
* Determines whether the relevant technology is stable and established.
* Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

### 3.5.2 Operational feasibility

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

* Determines whether the problems anticipated in user requirements are of high priority.
* Determines whether the solution suggested by the software development team is acceptable.
* Analyzes whether users will adapt to a new software.
* Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

### 3.5.3 Economic feasibility

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

* Cost incurred on software development to produce long-term gains for an organization.
* Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis).
* Cost of hardware, software, development team, and training.

## 3.6 Methodology

A software process (also known as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

### 3.6.1 Chosen Methodology

Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance. Each iteration passes through the **requirements, design, coding and testing phases**.And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented. The system is put into production when the first increment is delivered. The first increment is often a core product where the basic requirements are addressed, and supplementary features are added in the next increments. Once the core product is analyzed by the client, there is plan development for the next increment.



Figure 3.2: Incremental Model

## 3.7 Reasons for Chosen Methodology

* System development is broken down into many mini development projects
* Partial systems are successively built to produce a final total system
* Highest priority requirement is tackled first
* Once the requirement is developed, requirement for that increment are frozen

## 3.8 Work Plan

Database design is the process of producing a detailed data model of database. This data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

## 3.9 Project Structure

### 3.9.1 Team Structure

|  |  |
| --- | --- |
| **Members** | **Tasks** |
| Asfand Yar Mumtaz and Ethisham Ahmad Khan | Analysis, Design, Coding, Testing and Implementation. |

Table 3.1: Team Structure

### 3.9.3 Project Schedule

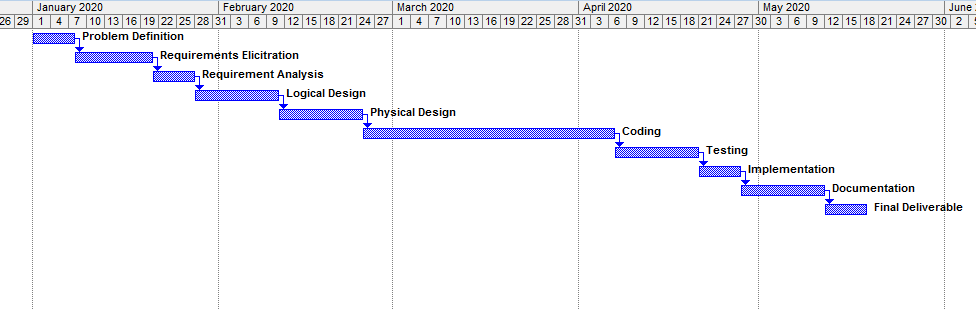


Figure 3.3: Project Schedule

# CHAPTER 4 System Design

## 4.1 Introduction

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either "all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" or "the activity following requirements specification and before programming, as in a stylized software engineering process."

Software design usually involves problem solving and planning a software solution. This includes both a low-level component and algorithm design and a high-level, architecture design.

## 4.2 Purpose

The design concepts provide the software designer with a foundation from which more sophisticated methods can be applied. A set of fundamental design concepts has evolved. They are as follows:

1. Abstraction - Abstraction is the process or result of generalization by reducing the information content of a concept or an observable phenomenon, typically in order to retain only information which is relevant for a particular purpose.It is an act of Representing essential features without including the background details or explanations.
2. Refinement - It is the process of elaboration. A hierarchy is developed by decomposing a macroscopic statement of function in a step-wise fashion until programming language statements are reached. In each step, one or several instructions of a given program are decomposed into more detailed instructions. Abstraction and Refinement are complementary concepts.
3. Modularity - Software architecture is divided into components called modules.
4. Software Architecture - It refers to the overall structure of the software and the ways in which that structure provides conceptual integrity for a system. Good software architecture will yield a good return on investment with respect to the desired outcome of the project, e.g. in terms of performance, quality, schedule and cost.
5. Control Hierarchy - A program structure that represents the organization of a program component and implies a hierarchy of control.
6. Structural Partitioning - The program structure can be divided both horizontally and vertically. Horizontal partitions define separate branches of modular hierarchy for each major program function. Vertical partitioning suggests that control and work should be distributed top down in the program structure.
7. Data Structure - It is a representation of the logical relationship among individual elements of data.
8. Software Procedure - It focuses on the processing of each module individually.
9. Information Hiding - Modules should be specified and designed so that information contained within a module is inaccessible to other modules that have no need for such information.

## 4.3 Scope

To develop an optimal android mobile application that helps the customers to request for a home service and get facilitated.

## 4.4 Definitions, acronyms and abbreviations

* IDE Integrated Development Environment
* ERD Entity Relationship Diagram
* CLR Common Language Runtime
* UML Unified Modeling Language
* ASP Active Server Pages
* HTTP Hyper Text Transfer Protocol

## 4.5 Architectural Representation (Architecture Diagram)

Node JS Platform uses “Single Threaded Event Loop” architecture to handle multiple concurrent clients. Then how it really handles concurrent client requests without using multiple threads. What is Event Loop model? We will discuss these concepts one by one.

Before discussing “Single Threaded Event Loop” architecture, first we will go through famous “Multi-Threaded Request-Response” architecture.

Any Web Application developed without Node JS, typically follows “Multi-Threaded Request-Response” model. Simply we can call this model as Request/Response Model.

Client sends request to the server, then server do some processing based on clients request, prepare response and send it back to the client.

This model uses HTTP protocol. As HTTP is a Stateless Protocol, this Request/Response model is also Stateless Model. So we can call this as Request/Response Stateless Model.

However, this model uses Multiple Threads to handle concurrent client requests. Before discussing this model internals, first go through the diagram below.

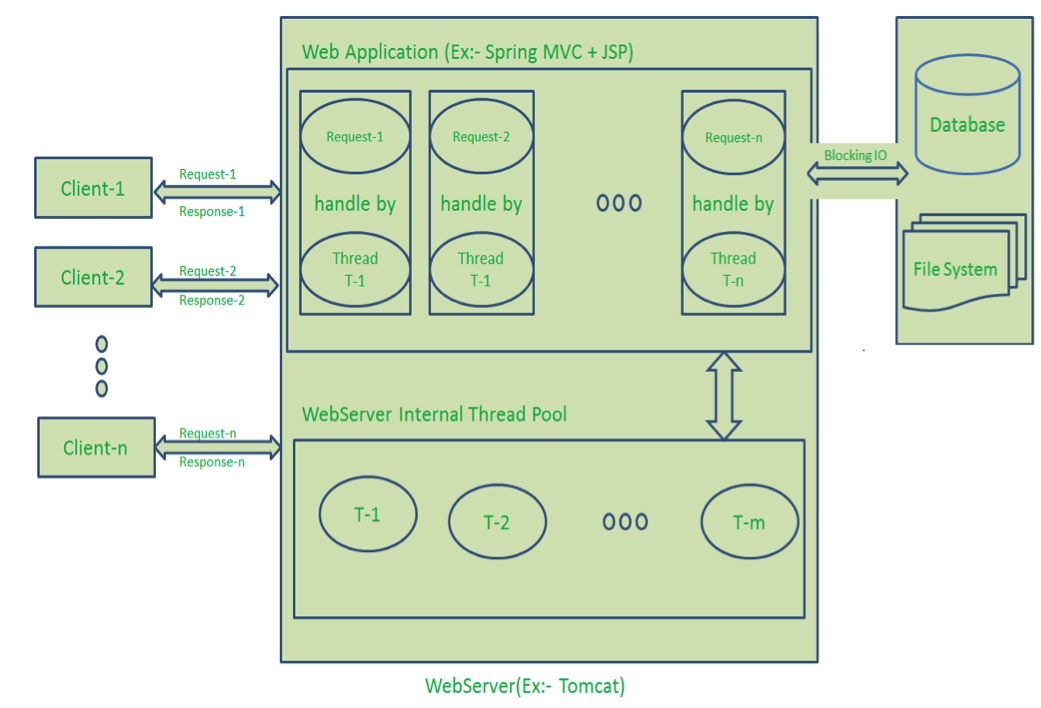


Figure 4.1: Architecture Diagram

 Here “n” number of Clients Send request to Web Server. Let us assume they are accessing our Web Application concurrently.

 Let us assume, our Clients are Client-1, Client-2… and Client-n.

 Web Server internally maintains a Limited Thread pool. Let us assume “m” number of Threads in Thread pool.

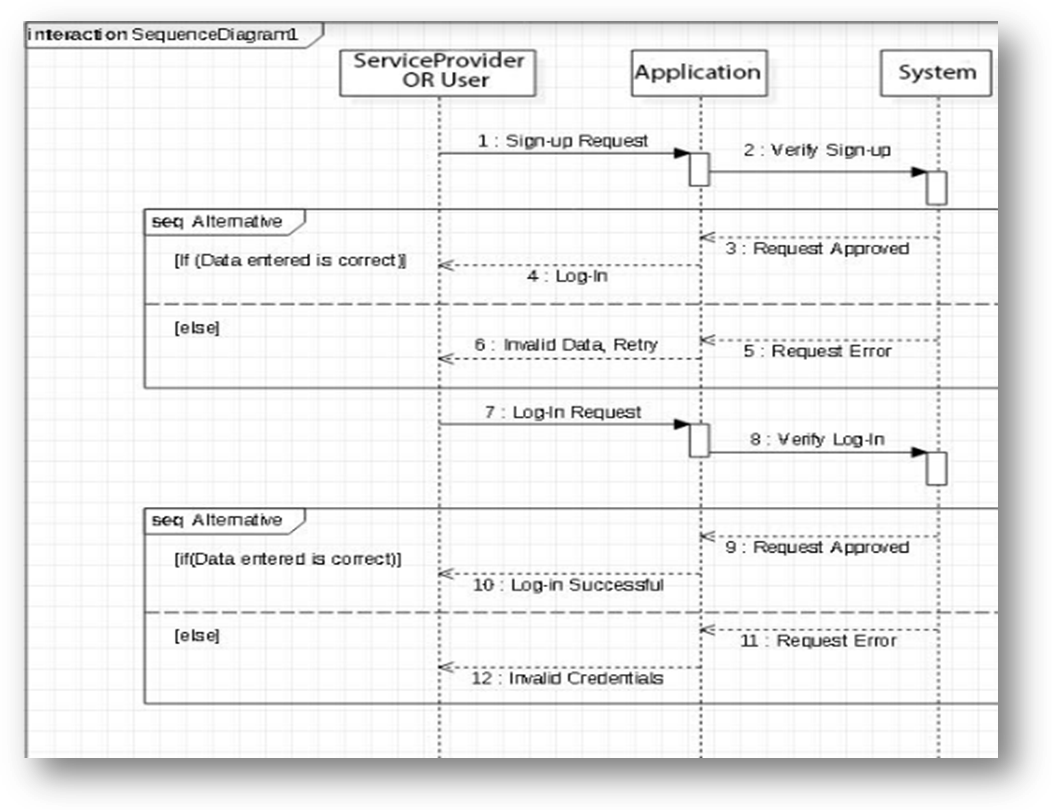
 Web Server receives those requests one by one.

* Web Server pickup Client-1 Request-1, Pickup one Thread T-1 from Thread pool and assign this request to Thread T-1
  + Thread T-1 reads Client-1 Request-1 and process it
  + Client-1 Request-1 does not require any Blocking IO Operations
  + Thread T-1 does necessary steps and prepares Response-1 and send it back to the Server
  + Web Server in-turn send this Response-1 to the Client-1
* Web Server pickup another Client-2 Request-2, Pickup one Thread T-2 from Thread pool and assign this request to Thread T-2
  + Thread T-2 reads Client-1 Request-2 and process it
  + Client-1 Request-2 does not require any Blocking IO Operations
  + Thread T-2 does necessary steps and prepares Response-2 and send it back to the Server
  + Web Server in-turn send this Response-2 to the Client-2
* Web Server pickup another Client-n Request-n, Pickup one Thread T-n from Thread pool and assign this request to Thread T-n
  + Thread T-n reads Client-n Request-n and process it
  + Client-n Request-n require heavy Blocking IO and computation Operations
  + Thread T-n takes more time to interact with external systems, does necessary steps and prepares Response-n and send it back to the Server
  + Web Server in-turn send this Response-n to the Client-n

## 4.6 Dynamic Model: Sequence Diagrams

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart.

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.



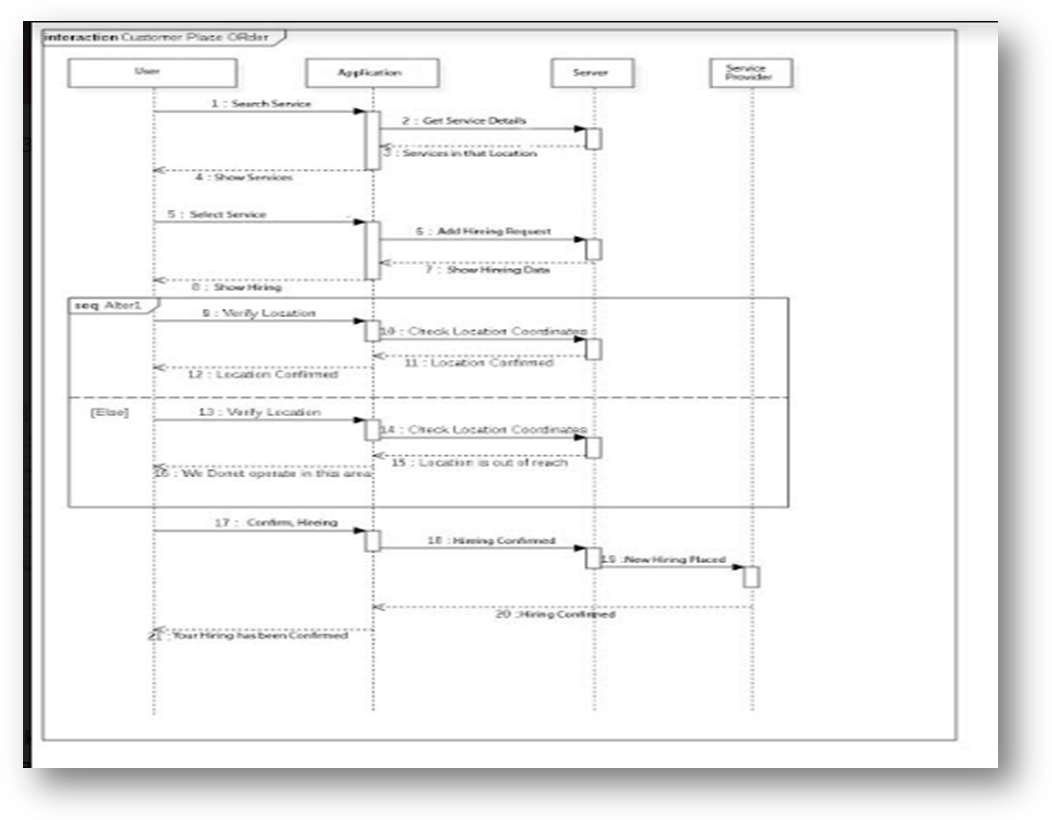


Figure 4.2: Sequence Diagram

## 4.7 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

* The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
* The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
* The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. With detailed modeling, the classes of the conceptual design are often split into a number of subclasses.

In order to further describe the behavior of systems, these class diagrams can be complemented by a state diagram or UML state machine.

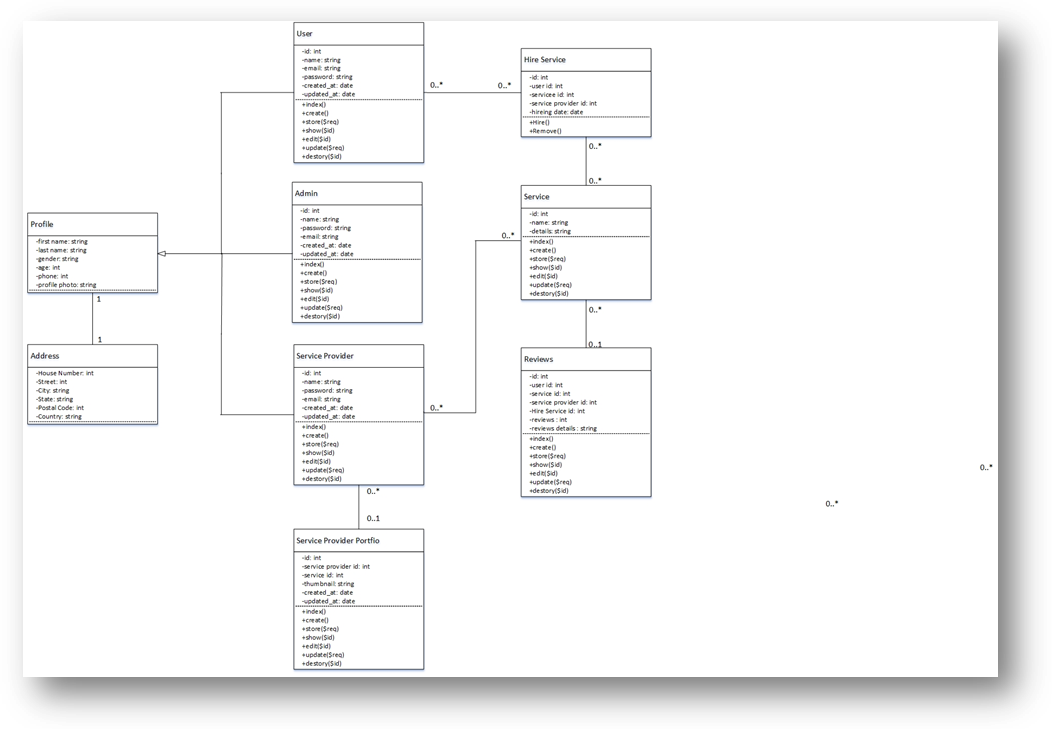
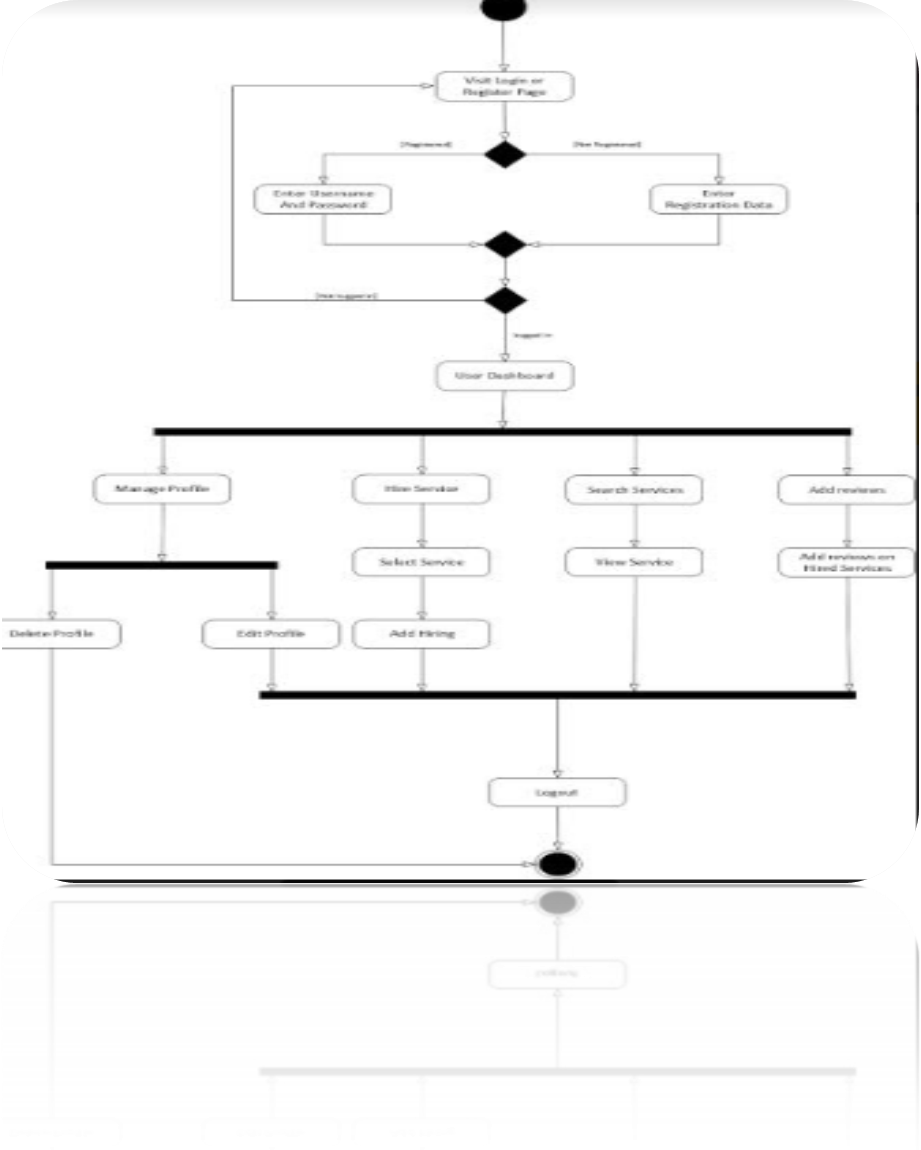


Figure 4.3: Class Diagram

## 4.8 Activity Diagram

We use Activity Diagrams to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system. An activity diagram is very similar to a flowchart.

 Figure 4.4: Activity Diagram

## 4.9 DFD

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

### 4.9.1 DFD Level 0

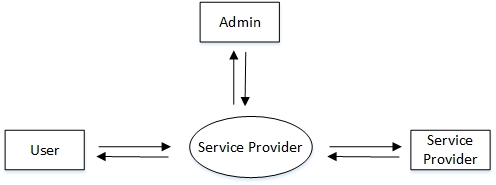


Figure 4.5: DFD Level 0

### 4.9.2 DFD Level 1

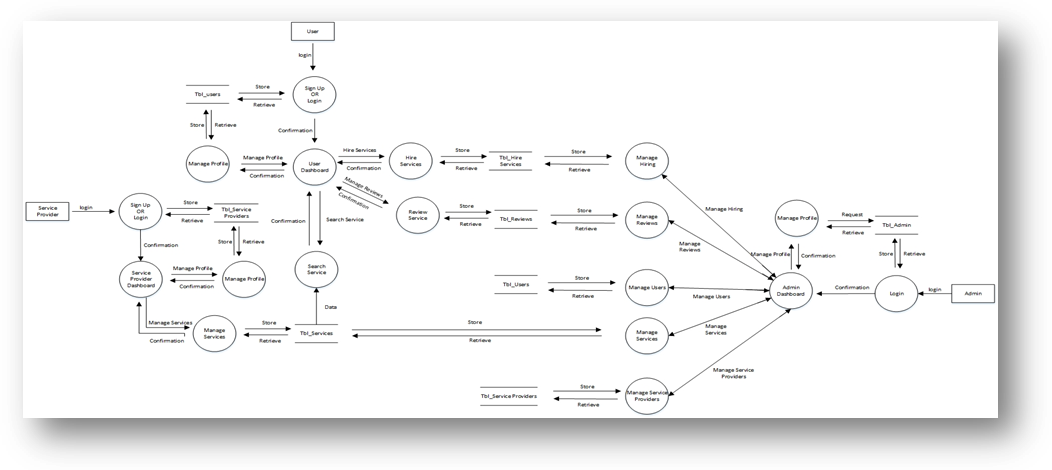


Figure 4.6: DFD Level 1

### 4.9.3 DFD Level 2

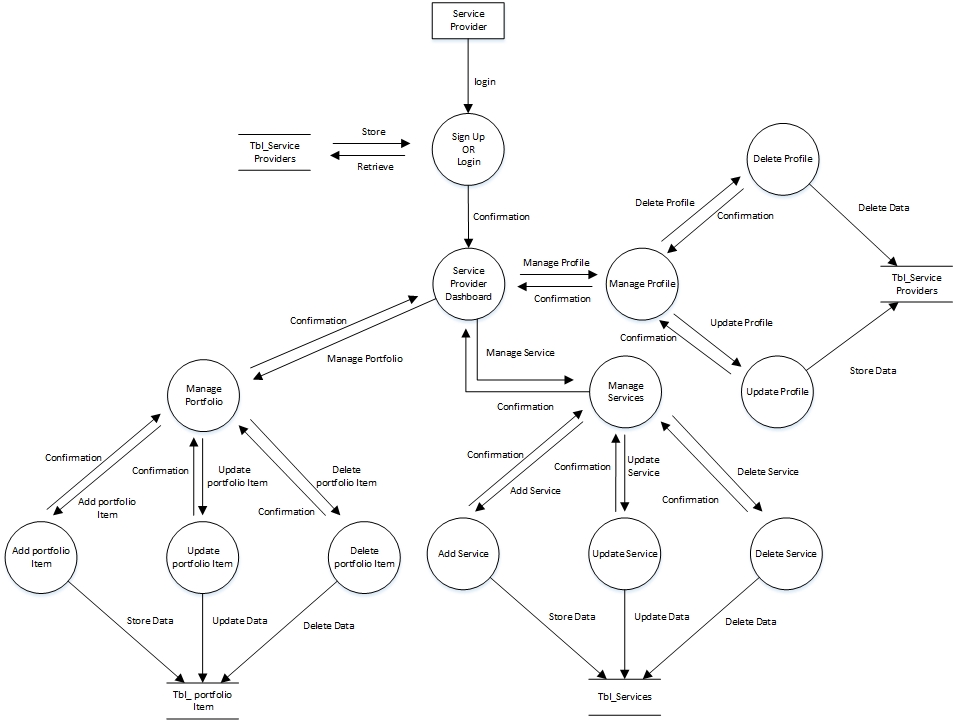


Figure 4.7: DFD Level 2

## 4.10 ERD

An entity–relationship model (or ER model) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types).

In software engineering, an ER model is commonly formed to represent things a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model, that defines a data or information structure which can be implemented in a database, typically a relational database.

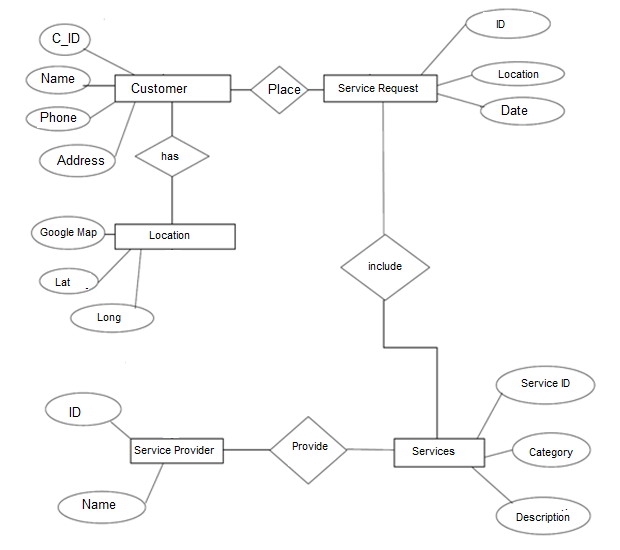


Figure 4.8: ERD

## 4.11 Normalization

Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. It was first proposed by Edgar F. Codd as part of his relational model.

Normalization entails organizing the columns (attributes) and tables (relations) of a database to ensure that their dependencies are properly enforced by database integrity constraints. It is accomplished by applying some formal rules either by a process of synthesis (creating a new database design) or decomposition (improving an existing database design).

## 4.12 Data Flow Chart

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

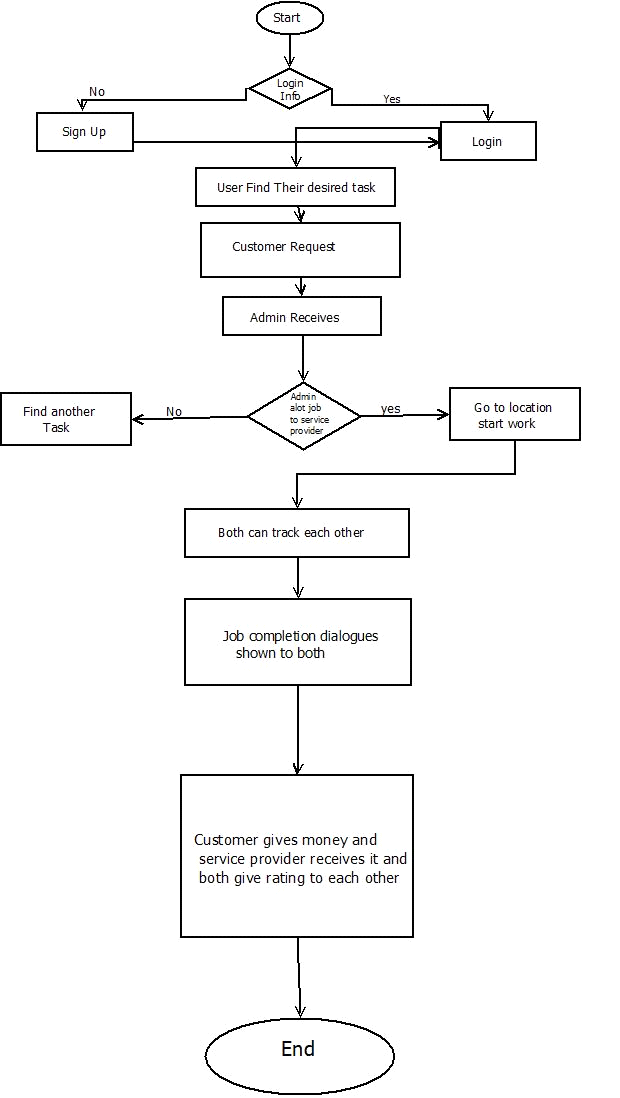


Figure 4.9: Flowchart

## 4.14 Graphical User Interfaces

User interface design (UI) or user interface engineering is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing usability and the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).

**Home Page**

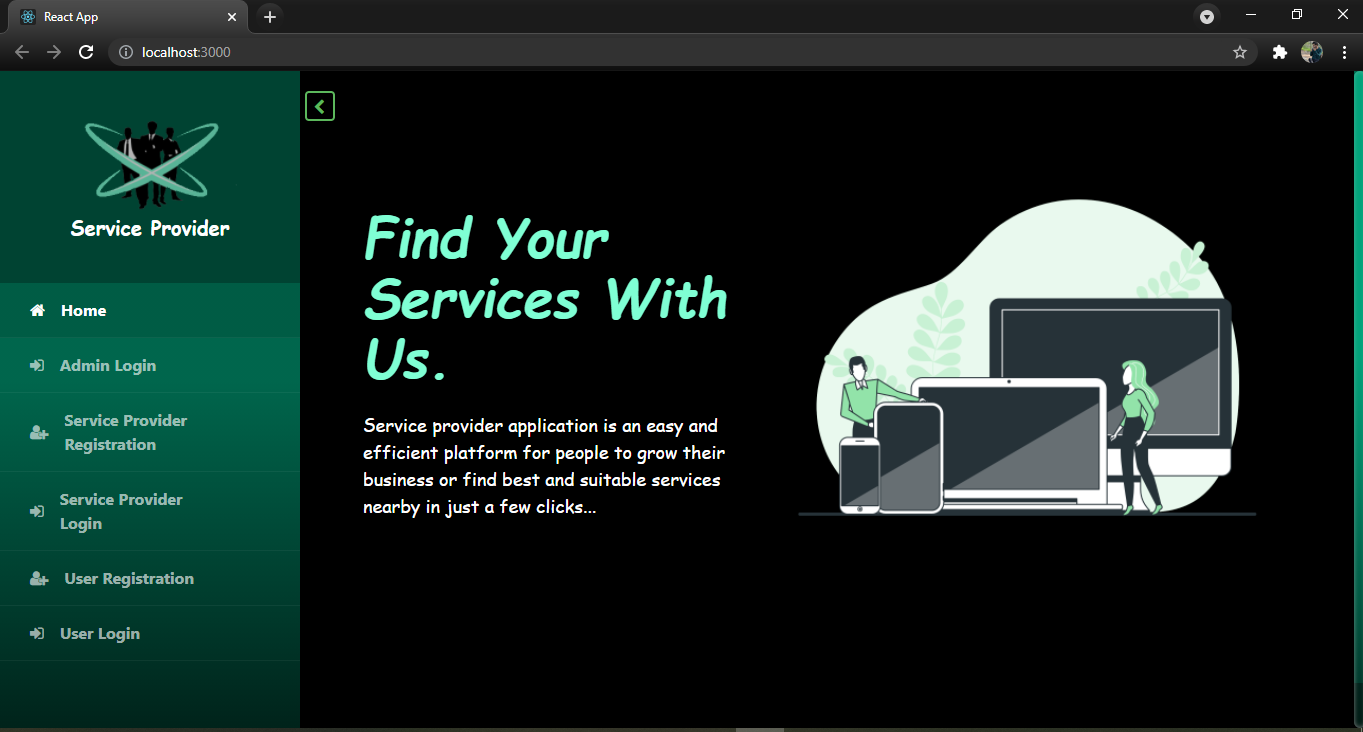


Figure 4.10: Home Page

**View Service Providers**

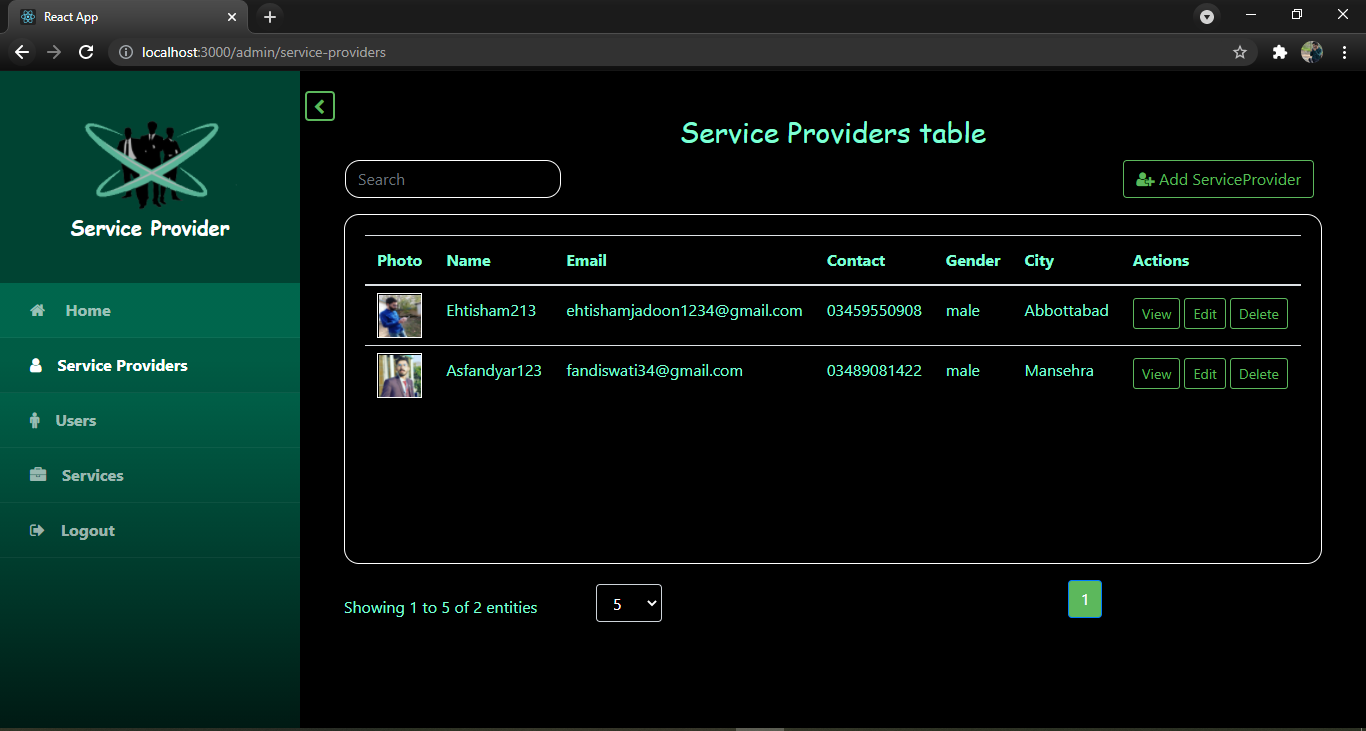


Figure 4.11: View Service Providers

**Service Provider Profile**

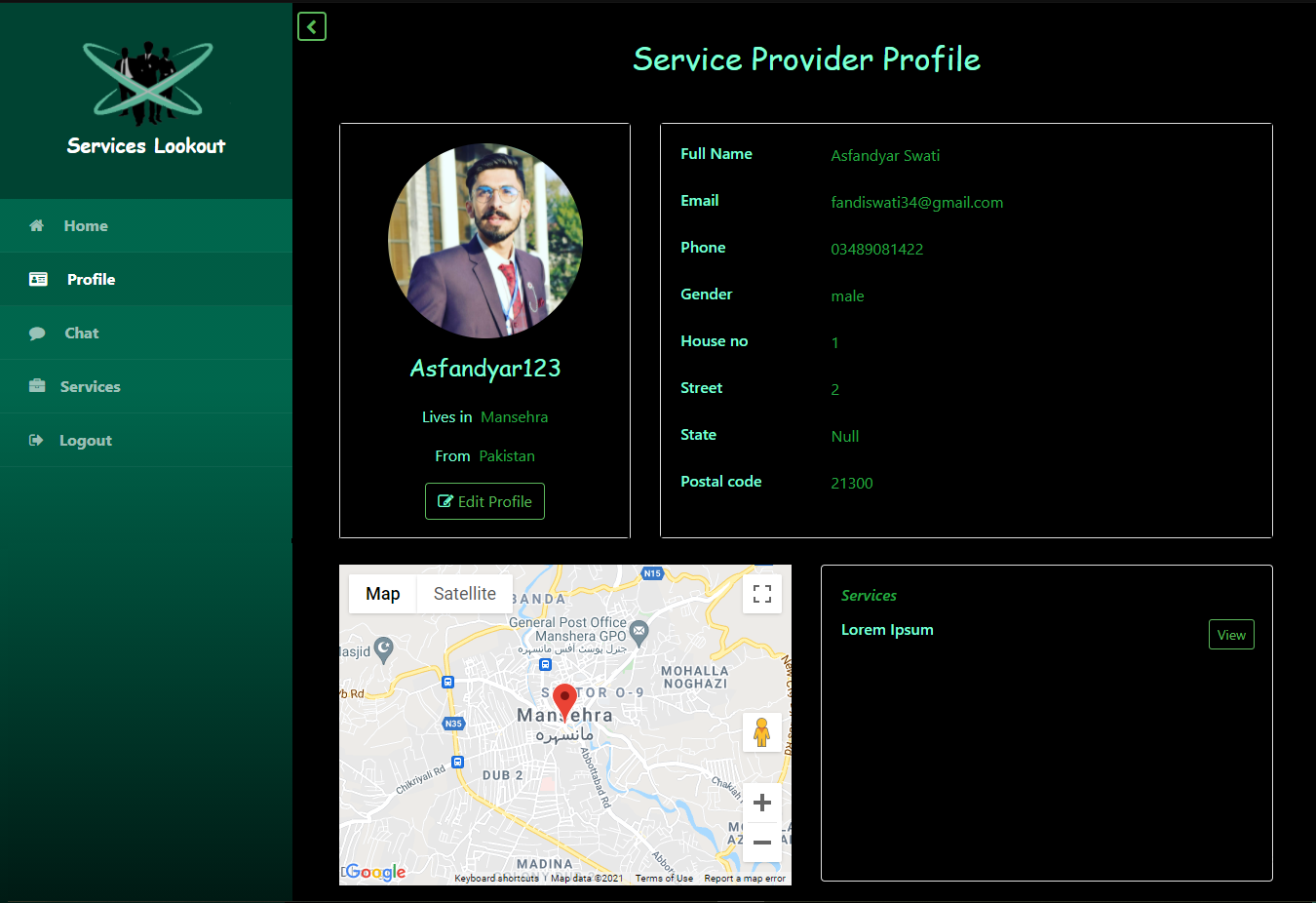


Figure 4.12: Service Provider Profile

**Service Detail**

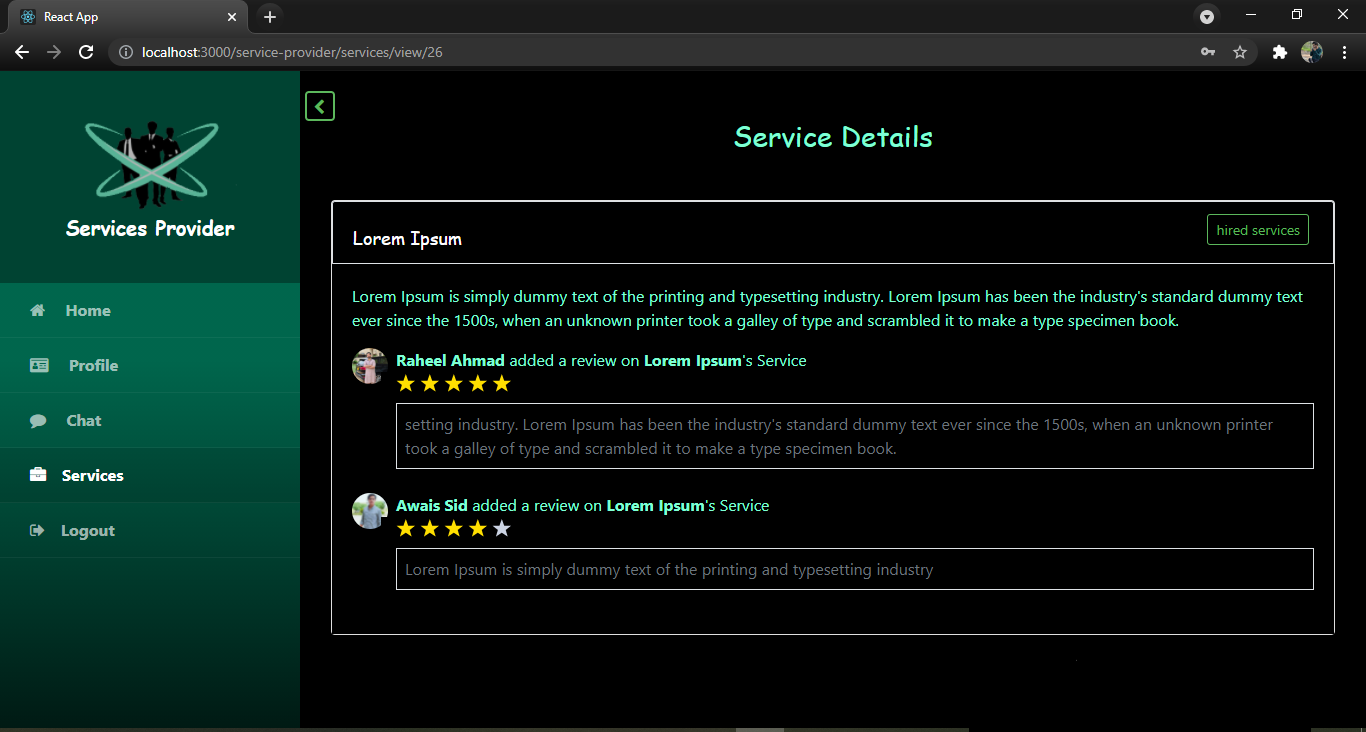


Figure 4.13: Service Detail

# CHAPTER 5 System Testing

## 5.1 Introduction

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements.

System testing takes, as its input, all of the integrated components that have passed integration testing. The purpose of integration testing is to detect any inconsistencies between the units that are integrated together (called assemblages). System testing seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. The actual result is the behavior produced or observed when a component or system is tested.

## 5.2 Purpose of Testing

1. Finding defects which may get created by the programmer while developing the software.
2. Gaining confidence in and providing information about the level of quality.
3. To prevent defects.
4. To make sure that the end result meets the business and user requirements.
5. To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
6. To gain the confidence of the customers by providing them a quality product.

## 5.3 Objective of Testing

Software testing helps in finalizing the software application or product against business and user requirements. It is very important to have good test coverage in order to test the software application completely and make it sure that it’s performing well and as per the specifications.

## 5.4 Testing Techniques

Software Testing Techniques help you design better test cases. Since exhaustive testing is not possible; Manual Testing Techniques help reduce the number of test cases to be executed while increasing test coverage. They help identify test conditions that are otherwise difficult to recognize.

### 5.4.1 Black Box Testing

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

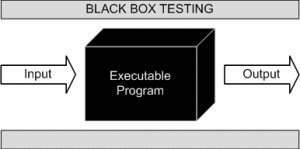


Figure 5.1: Black Box Testing

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see.

### 5.4.2 White Box Testing

White Box Testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

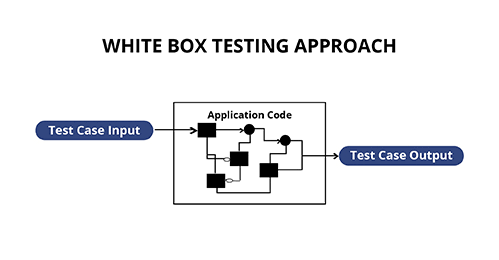


Figure 5.2: White Box Testing

This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

## 5.5 Unit Testing

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

## 5.5 Integration Testing

Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

## 5.7 System Testing

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements.

System testing takes, as its input, all of the integrated components that have passed integration testing. The purpose of integration testing is to detect any inconsistencies between the units that are integrated together (called assemblages). System testing seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. The actual result is the behavior produced or observed when a component or system is tested.

### 5.7.1 Test Cases

A test case, is a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. A test case could simply be a question that you ask of the program. The point of running the test is to gain information, for example whether the program will pass or fail the test. Test case is the cornerstone of Quality Assurance whereas they are developed to verify quality and behavior of a product.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id** | **Test Case ID with Title** | **Test case Description** | **Expected Result** | **Test Method** |
| 1 | Test Case 01 (Login) | Administrator, Service Provider & Customer Requests the System for Login | Successful Login | White Box and Black Box testing is performed |
| 2 | Test Case 02 (View Services) | Customer view and search the desired service | Successful in finding and viewing services | White Box and Black Box testing is performed |
| 3 | Test Case 03 (User Registration) | By using this form new user can register himself | Successful in user registration | White Box and Black Box testing is performed |
| 4 | Test Case 04 (Add Services) | Service Provider add new service | Successful in Adding New service | White Box and Black Box testing is performed |
| 5 | Test Case 05 (Update Record) | Users attempt to update specific record | Successful in Updating record | White Box and Black Box testing is performed |
| 6 | Test Case 06 (Add Order) | Customer place order on the item | Successful in Adding order | White Box and Black Box testing is performed |
| 7 | Test Case 07 (View Order) | Service Provider view the order placed on specific service | Successful in viewing order | White Box and Black Box testing is performed |
| 8 | Test Case 08 (Change Password) | User can change their user account password | Successful in Changing Password | White Box and Black Box testing is performed |
| 9 | Test Case 0 (Logout) | User of system attempt to logout from system. | Successful in logout | White Box and Black Box testing is performed |

Table 5.1: Test Cases

### 5.7.2 Trace-ability

Requirement Traceability Matrix (RTM) is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle. The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Test** | **Result** |
| Test 1 | Verify that user can access all pages of the web portal | Test has passed successfully. |
| Test 2 | Verify that user can Access all modules | Test has passed successfully. |
| Test 3 | Verify that user get error messages on wrong entry | Test has passed successfully. |
| Test 4 | Verify that user gets error message on entering wrong password | Test has passed successfully. |
| Test 5 | Verify that all links are linked with the desired forms | Test has passed successfully. |
| Test 6 | Verify that user can submit the records in desired fields of each form. | Test has passed successfully. |
| Test 7 | Verify that user can perform all database operations. | Test has passed successfully. |
| Test 8 | Verify that user can see, print, exports and search all reports. | Test has passed successfully. |

Table 5.2: Traceability Matrix

# Chapter 6 Conclusions

In this thesis, we addressed the problem of current system that are used for the existing system. We also address the proposed system and also developed an android application for the Services Provider.

# Appendix REFERENCES

* Jaffery A Hoffer, Mary B Prescott & Fred R McFadden - Modern Database Management (Eighth Edition).
* Kenneth E Kendall & Julie - System Analysis and Design.
* Rojer Pressman - Software Engineering
* Software Engineering- A practitioner's Approach by Roger S. Pressman: 6th edition McGraw Hill, 2005
* An Integrated Approach to Software Engineering by Pankaj Jalote:3rd edition Springer, 2005
* Rojer Pressman - Software Engineering Fifth Edition
* Software engineering A Beginner’s Guide, 2nd Edition by Wendy Willard.
* Wrox - Beginning Java Database Programming.
* Android, A Programmers Guide by J.F. DiMarzio
* http://www.wikipedia.org