

EASY WAY
A PROJECT REPORT
for
Major Project (KCA451)
Session (2023-24)

Submitted by

Arpita Vaish (University Roll No: 2200290140038)
Anshuman Patek (University Roll No: 2200290140036)
Anshu (University Roll No: 220290140035)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Mr Praveen Kumar Gupta
Assistant Professor



Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(MAY 2024)

CERTIFICATE

Certified that **Arpita Vaish (2200290140038)**, **Anshuman Patek (2200290140036)**, **Anshu (2200290140035)** has/ have carried out the project work having “**Easy Way**” (**Major Project-KCA451**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Arpita Vaish (2200290140038)

Anshuman Patek (2200290140036)

Anshu (2200290140035)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Mr Praveen Kumar Gupta
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

Our project focuses on creating a service portal. Users can easily input their needs, and the system promptly provides the required services. The portal prioritizes simplicity and efficiency for a seamless user experience.

This project introduces a user-friendly service portal emphasizing intuitive interaction and effortless navigation. We prioritize a seamless experience by minimizing user input through a clear interface designed to facilitate the articulation of specific needs.

A robust backend system leverages intelligent automation to match user requests with the most appropriate services. This eliminates the need for complex navigation and ensures a real-time response, delivering a swift resolution to user inquiries.

This user-centric approach fosters increased user satisfaction and promotes a positive brand experience by prioritizing efficiency and minimizing user effort.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Mr. Praveen Kumar Gupta** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Arpita Vaish (2200290140038)

Anshuman Patek (2200290140036)

Anshu (2200290140035)

TABLE OF CONTENT

	CERTIFICATE	II
	ABSTRACT	III
	ACKNOWLEDGEMENT	IV
	TABLE OF CONTENT	V-VII
	LIST OF FIGURES	VIII
1	INTRODUCTION	1-5
	1.1 OVERVIEW	1
	1.2 NEED	2
	1.3 PROJECT DESCRIPTION	3
	1.4 PROPOSED SYSTEM	3
	1.5 FEATURES OF THE SYSTEM	4
	1.6 PROJECT SCOPE	4-5
2	LITERATURE REVIEW	6-10
	2.1 LITERATURE REVIEW	6-9
	2.1.1 RELATED CONCEPTS	6-7
	2.1.2 E-SERVICE QUALITY	7-8
	2.2.3 FUTURE SCOPE	8-9
	2.2 LITERATURE SURVEY	9-10
3	FEASIBILITY STUDY	11-14
	3.1 TECHNICAL FEASIBILITY	11-12
	3.1.1 HARDWARE FEASIBILITY	12
	3.1.2 SOFTWARE FEASIBILITY	12
	3.2 ECONOMIC FEASIBILITY	13
	3.3 LEGAL FEASIBILITY	13
	3.4 OPERATIONAL FEASIBILITY	13
	3.5 BEHAVIORAL FEASIBILITY	13-14
	3.6 RISK ANALYSIS	14
4	DATABASE DESIGN	15-34
	4.1 FLOWCHART	16-18
	4.1.1 BASIC SYMBOLS USED IN FLOWCHART DESIGN	16-18
	4.1.2 FLOWCHART ABOUT THE WORKING OF APPLICATION	18
	4.2 USE CASE DIAGRAM	18-22
	4.3 DATA FLOW DIAGRAM	22-25
	4.3.1 DFD LEVELS	23
	4.4 E-R DIAGRAM	25-31

	4.4.1 USES OF ER DIAGRAM	26
	4.4.2 COMPONENTS AND FEATURES OF ER DIAGRAM	26-30
	4.4.3 ER DIAGRAM OF THE APPLICATION	30
	4.5 LOGICAL DATABASE	31-33
	4.6 PHYSICAL DATABASE	33-34
5	IMPLEMENTATION	35-39
	5.1 IMPLEMENTATION	35
	5.1.1 OVERVIEW OF TECHNOLOGIES USED	35-36
	5.1.2 DEVELOPMENT ENVIRONMENT SETUP	36
	5.1.3 DATABASE DESIGN	36
	5.1.4 FRONTEND DEVELOPMENT	36-38
	5.1.5 BACKEND DEVELOPMENT	38-39
6	TESTING	40-49
	6.1 UNIT TESTING	40-42
	6.1.1 TECHNIQUES OF UNIT TESTING	41
	6.1.2 UNIT TEST CASES OF EACH MODULE	41-42
	6.2 INTEGRATION TESTING	42-44
	6.2.1 TECHNIQUES OF INTEGRATION TESTING	43
	6.2.2 INTEGRATION TEST CASES FOR THE SYSTEM	43-44
	6.3 SYSTEM TESTING	45-46
	6.3.1 TYPES OF SYSTEM TESTING	45
	6.3.1 TEST CASES FOR ENTIRE SYSTEM	45-46
	6.4 ACCEPTANCE TESTING	47
	6.4.1 ACCEPTANCE TEST CASES FOR APPLICATION	47
	6.5 SOFTWARE VERIFICATION AND VALIDATION	48
	6.5.1 SOFTWARE VERIFICATION	48
	6.5.2 SOFTWARE VALIDATION	48
	6.6 TEST PROCEDURE	49
	6.7 TEST CASES	49
7	DESIGN	50-55
	7.1 LOGIN	51
	7.2 REGISTRATION	52
	7.3 HOME	53
	7.4 SERVICES	53
	7.5 CONTACT US	54
	7.6 ABOUT US	54
	7.7 FEEDBACK	55
	7.8 USER REVIEW	55
8	EVALUATION AND CONCLUSION	56-60
	8.1 EVALUATION	56
	8.2 LIMITATION OF THE SYSTEM	56-57
	8.3 PROBLEMS ENCOUNTERED	57-59

	8.4 RECOMMENDATION/FUTURE SCOPE	59-60
	8.5 CONCLUSION	61
	REFERENCES & BIBLIOGRAPHY	62

LIST OF FIGURES

4.1.2	Flowchart About Working of Application	18
4.2.1	Use Case Diagram for Admin	20
4.2.2	Use Case Diagram for User	21
4.2.3	Use Case Diagram for Worker	22
4.3.1	Context Level DFD for Easy Way Application	24
4.3.2	Level-1 DFD For Easy Way Application	25
4.4.3	ER Diagram for The Application	31
4.5.1	Logical Database for User	32
4.5.2	Logical Database for Worker	32
4.5.3	Logical Database for Feedback	33
4.6.1	Physical Database of Application	34
4.6.2	Physical Database of Application	34
7.1	Login	51
7.2	Registration	52
7.3	Home Page	53
7.4	Services	53
7.5	Contact Us	54
7.6	About Us	54
7.7	Feedback	55
7.8	User Review	55

CHAPTER 1

INTRODUCTION

1.1 Overview

In the ever-evolving landscape of service interactions, user experience reigns supreme. Gone are the days of navigating complex menus and enduring convoluted procedures. Today's users demand effortless access, intuitive interfaces, and prompt fulfilment of their needs. This project takes centre stage in this transformation by introducing a revolutionary service portal designed with simplicity and efficiency as its core tenets. Our user-friendly platform goes beyond mere service delivery; it empowers individuals to navigate with ease, articulate their needs with clarity, and obtain the desired services with minimal effort. This translates to a seamless experience that fosters increased user satisfaction and builds a positive brand image. We envision a portal that anticipates your needs, not one that creates hurdles.

Unveiling the Power of Intuitive Interaction:

This service portal prioritizes intuitive interaction. We believe in interfaces that are as clear as daylight, devoid of technical jargon, and built with a focus on effortless navigation. Our design philosophy revolves around a user-centric approach, placing the user's comfort at the forefront. Gone are the days of deciphering confusing menus and grappling with complex forms. Our portal streamlines the process with clear instructions, readily identifiable options, and a layout designed for optimal user flow. Users intuitively navigate through the portal, minimizing cognitive load and ensuring a frictionless experience.

Effortless Input:

Articulating Your Needs with Clarity We understand that time is a valuable commodity. Our service portal eliminates the need for lengthy data entry and cumbersome form-filling. We believe in the power of effortless input. The platform utilizes a clear interface

designed to facilitate the articulation of specific needs. Whether it's requesting technical assistance, scheduling an appointment, or seeking information, users can clearly and concisely convey their requirements. This empowers users to quickly and efficiently engage with the system, eliminating unnecessary steps and fostering a more productive user experience.

The Power of Intelligent Automation:

Matching Needs with Solutions Behind the scenes, the service portal harnesses the power of intelligent automation. A robust backend system acts as the brain of the operation, seamlessly matching user requests with the most appropriate services available. This eliminates the need for complex navigation and ensures a real-time response. Imagine a system that understands your needs and directs you to the right solution instantly. This intelligent automation streamlines service delivery, eliminates unnecessary delays, and delivers a swift resolution to user inquiries.

The Gateway to User Satisfaction:

A Seamless Service Experience By prioritizing simplicity, intuitive interaction, and effortless input, our service portal paves the way for a seamless user experience. Users can confidently navigate the system, clearly articulate their needs, and receive prompt and efficient service delivery. This user-centric approach translates to increased user satisfaction. When users feel valued and empowered, their perception of the brand improves significantly. This fosters a positive brand experience, ultimately leading to higher customer retention and advocacy.

1.2 Need

The need your service portal addresses is reducing friction in the process of obtaining a service. Here's how it breaks down:

- **Complexity:** Many services, especially those offered by governments or large organizations, can be complex to navigate. Users often have to wade through layers of information or unclear instructions to find what they need.
- **Inefficiency:** Traditional methods of obtaining services, like phone calls or in-person visits, can be time-consuming and inefficient. Users may have to wait on hold or stand in line for extended periods.
- **Seamless User Experience:** In today's digital world, users expect a smooth and intuitive experience when interacting with any service. A clunky or confusing interface can quickly frustrate users and deter them from using the portal.

Our service portal tackles these issues by offering a simple and efficient way for users to access the services they need. By streamlining the process and prioritizing user-friendliness, your portal can save users time and frustration, ultimately leading to a higher level of satisfaction.

1.3 Project Description

Our project addresses the challenges which occur when local worker is required to finish their daily work, such as repairing electronic items, cleaning houses and offices and plumbing-related work. On the other side, workers also have to waste their time searching for work. Currently, valuable time is spent searching for suitable workers, leading to inefficiency and frustration. To tackle this problem, we have taken the initiative to develop a user-friendly portal that allows users to register and access a database of local workers easily. The portal provides comprehensive information about each worker, including their experience and ratings, enabling users to make informed decisions when selecting a worker for a particular task. The primary objective of this initiative portal is to digitize the local worker ecosystem, making it convenient for users to search for and contact workers with relevant skills and expertise. By implementing an efficient system, we aim to streamline the process and eliminate the hassles of finding reliable local workers. Ultimately it will improve the overall experience of both users as well as workers.

1.4 Proposed System

User View:

1. Login/Register

Users can register on the website through basic information like name, email, password, location, and more. Moreover, workers can register through some other information like occupation and experience.

2. Top workers

Top workers are listed on the basis of their ratings and review. Top Workers are also categorized as per their occupation.

3. Filtering workers by occupation

Here service page of this portal provides the feature of searching for a worker through occupation.

4. Request for service

If the user wants to take service, they can contact the worker through the portal by generating the request, and this request will be shown to the worker; if the worker is interested in this request, then the worker can accept it or reject it.

5. Review and rating for workers

It provides a feature for users to write their experience and provide ratings to workers based on their work.

6. Contact us

The portal provides a feature for users and workers to communicate with the Admin for any type of query regarding the portal. Users/workers can fill out the contact us form and mention their issues.

7. Feedback form

A feedback form is a tool to gather information about a user's opinion or experience regarding a specific service.

1.5 Features Of This System

Service Portal Features: User Experience

Here's a breakdown of features focusing on a user's experience within your service portal:

Core Features:

- Login/Signup: Easy and secure user registration with options like social media logins.
- Search for Workers:
- Search by Occupation: Users can find workers based on specific occupations (plumber, electrician, etc.).
- Browse by Category: Allow browsing for general service categories (handyman services, IT support).
- Filter Options: Refine searches by location, availability, price range, user ratings, or specific skills (licensed electrician).
- Worker Profiles: Detailed profiles showcasing worker information including:
 - Experience and Qualifications (years of experience, relevant certifications).
 - Skills and Services Offered (list of specific skills and the services they provide).
 - User Reviews and Ratings (to build trust and credibility).
 - Contact Information (optional - may be managed through messaging system).
- Requesting Service:
 - Simple and clear process for users to request services from chosen workers.
 - Ability to specify job details, desired timeframe, and budget.
 - Secure communication channel to discuss project details with workers (messaging system).

1.6 Project Scope

The project aims to develop a comprehensive and sustainable service-based web application with the following scope:

- **User View:**
 - Login/Signup with secure user authentication.
 - Search for workers by occupation, category, and filter options (location, availability, price, ratings, skills).
 - View detailed worker profiles with qualifications, skills, and potentially user reviews.
 - Secure messaging system for users to communicate with workers.
 - Simple process to request services from chosen workers, specifying details, timeframe, and budget.

- **Admin View:**
 - Basic functionalities for managing worker applications (approve/reject).
 - This can be expanded upon in future iterations.
- **Security:** Secure login protocols and data encryption for user and worker information.
- **Additional Considerations:**
 - Define the target user base (who will be using the service portal).
 - Determine the initial service categories to be offered.
 - Establish a clear timeline for development and launch.
 - Set a budget for development, maintenance, and potential ongoing costs.

This project scope provides a clear roadmap for building a functional service portal with a focus on user experience. It allows for controlled development while leaving room for future expansion based on user needs and project success. Remember, this is just a starting point, and you can adjust it based on your specific requirements and priorities.

CHAPTER 2

Literature Review

2.1 Literature Review

Literature review is an expressive study based on the detailed review of earlier pertinent studies related to the various concepts of online shopping to discover the concept of online shopping. It highlights the status of online shopping, importance and problems of online shopping, factors affecting online shopping and a critical review of the privacy and security issues in online shopping.

2.1.1 Related Concepts

Electronic service (e-service) refers to providing services on the Internet, including electronic commerce. E-services also involve non-commercial services (e.g., e-government). Besides, electronic services are a collection of services that are invariably available on the Internet to provide online purchases and sale transactions. This kind of website is distinct from the traditional ones, which only provide descriptive information. Since the 2000s, many researchers have become interested in e-services. The Net Offer model proposed by Gronroos et al. recommends different concepts of core services in a service pack that is constantly available in the simulated market; nonetheless, it is still a value-added service that should involve both customers and media. Although expectations are a less critical factor in e-commerce, customers can still employ experience-based criteria, and traditional services as benchmarks for e-services. VanRiel et al. suggested five different elements in electronic services: core services; conditional services; support services; additional services; and user interfaces which allow purchasers to access to a variety of electronic services. Service quality is the association between the buyer's expectations with the response performance. High service quality will satisfy the various customer needs and increase competitive advantages. This showed that when service performance was evidently sufficient or above the customers' expectation, the service quality was satisfactory. If the

performance is below expectations, it can be inferred that the service was developed with inferior quality. Gummesson discussed the theory of service quality associated with perception and trust. Gronroos stated a concept related to the complete service quality as the customer's perception of the distinction occurring amongst the expected and perceived services. Service quality is generally understood as a scale of service levels that matches the customer's expectation. Besides, Parasuraman et al. define service quality as the assessment of a particular service firm that resulted from differentiating performance with purchaser's expectations of firms in the industry. This notion was promoted into a tool for service quality evaluation called SERVQUAL. Two main streams related to the research field on the service quality scales are: the Nordic perspective, using generic service terms to define service quality (e.g., functional, technical) North American perspective, using service attributes to describe service quality in scales (e.g., reliability, responsiveness, empathy, assurance, materialization). In addition, the different research models related to service quality which have gain much attention, for example, technical and functional quality, GAP model, aggregate service quality, ideal value of service quality, benchmarked quality, associations with information technology, overall attributes, retail service quality, internal service quality, e-SQ.

2.1.2 E-Service Quality(E-SQ)

E-service quality (e-SQ) is the assessment of service quality in the virtual market. E-SQ also determines the success and effectiveness of websites, customer satisfaction. Parasuraman et al. stated e-SQ as the "extent to which a website facilitates efficient and effective shopping, purchasing, and delivery." This statement indicates that the concept of e-SQ rooted from the repurchase intention (e.g., ease of use, product information, ordering information, and protection of personal information) to the post-purchase stage (e.g., distribution, customer service, and return policy). Besides, Parasuraman et al. indicate two different scales for measuring e-SQ, including the basic E-S-QUAL scale and the extended E-RECs Qual scale. The E-S-QUAL consists of different dimensions including system availability, privacy, fulfilment, and efficiency. Remarkably, the efficiency mentions the ease as well as the pace of evaluating and employing the websites. The attainment means the growth of the site's intention about the distribution of the commodity's delivery as well as the item availability is fulfilled at a certain level. System availability is related to how precisely the website can place on its technical functions. Privacy mentions the proportion to how safe the website is or not and how it can conduct different protective interface to customer information. It can be seen that E-S-QUAL is related to the overall customers of a website. E-RECs-QUAL is an inferior branch of E-S-QUAL, which focuses on handling services' issues and inquiries. E-RECs-Qual only gives customers non-routed contact with websites, and this scale includes three dimensions as follows: responsiveness, compensation, and contact. In particular, responsiveness means the effective handling of problems and how it is returned to the website. Compensation mentions how a website could compensate its purchasers for their emerge issues. Contact means the process of proactive support

with tele-phone or virtual agents. From the literature review of related studies on e-SQ, the author summarizes and interprets the concept of e-SQ as follows: “e-SQ includes ease of navigation, site aesthetics and customization, reliability, responsiveness, security/privacy, and information quality. “For customers, high-standard e-SQ is a tool by which the potential benefits of the Internet could bring. In addition, Parasuraman suggested themes in the online environment, particularly the positive ones (e.g., flexibility, convenience, efficiency...); negative themes (e.g., security failure, mal-control, risk of obsolescence...), these factors are closely related to service quality in many ways. According to Santos, the interactive nature of the Internet facilitates the search and integration of information to meet customer requirements effectively. Besides, online products ‘technical features and prices are much easier to compare than traditional channels, so online service quality is crucial to customers. Thus, online customers expect the same service quality as the traditional providers could bring. E-SQ can make a valuable difference for business development for online service providers. Website interactive features, various media contents, and customizability attract the attention of businesses in e-commerce. In addition, online companies also provide better services than their rivals which is a determining factor in earning customer loyalty. Hence, focusing on e-SQ is the primary concern in e-commerce. If the Internet is used correctly, the Internet is a powerful tool for increasing the overall quality of service and setting entirely higher standards in various industries. A high-quality e-SQ is going to supply extended-term benefits for a company. An overall e-SQ model provides a comprehensive framework for e-SQ. It includes distinct trends classified into six positive dimension trends and five recovery dimension trends. Specifically, the recovery dimension is stated as the suitable sketch of how a technology website is utilized in order to offer its customers with comfort-able experience, knowledge, and attractiveness of websites. Most of the factors in the recovery dimension can be developed before launching the website as follows: ease of use, alignment, appearance, arrangement, structure, and content. The positive dimensions must be consistent throughout the life cycle of a website. This factor is likely to improve the customer maintenance and enhance the effects of word of mouth (WoM). Besides, suggested a conceptual framework related to disparate elements of e-service quality to its outcomes, including WoM, customer satisfaction, and repurchase intentions. This structure is originated from the means-end chain theory. This model provides a complete comprehension of the elements, sequences, and moderators of e-SQ which serves as a theoretical basis for forthcoming experimental studies

2.1.3 Future Work

The conceptualization and measurement of quality are a major problem for e-service studies. Many authors have studied e-SQ and its theoretical gaps [68]. The concepts and e-SQ scale still have many theoretical gaps, and this is a challenge and an opportunity for researchers to continue exploring. Therefore, Barrutia et al. [82] pointed out some challenges in e-SQ research as follows: (1) several e-SQ definitions are still vague and there is no consensus on the main factors in measuring e-SQ; (2) there is a

lack of theoretical studies and supporting theories on e-SQ; (3) most of the scales tend to incompletely reflect consumers' perceptions of e-SQ; (4) e-SQ studies in a single-channel direction and it failed to genuinely comprehend the perception of consumers of e-services in the multi-channel context; (5) limitations in the researcher's access to information sources; (6) the positive randomization properties of the scale may not be considered in studies; (7) limitations in receiving information of traditional; (8) there is a growing need for alternative research methods that exist but remain largely unexplored; (9) More in-depth qualitative research is needed to grasp why certain concepts related to e-SQ are needed; (10) the results of e-SQ have not been entirely and comprehensively developed. Many authors have developed and tested several research models related to e-SQ in different contexts. For example, e-TailQ, SITEQUAL, WEBQUAL, ES-QUAL and E-RECs-QUAL, E-Trans-Qual, Pe-SQ, E-SERVCON, and IS-SERVQUAL, these models have been widely applied and validated. Some new research directions that could be implemented in the future are to reconstruct the e-SQ scales in distinct contexts which enhance their external validity. In particular, the IS-Seroquel model should be deemed as the latest scale to evaluate the quality of information systems because the SERVQUAL model is outdated and no longer suitable for modern information systems. The details of the summary of research models are presented in Table 1. Accordingly, the researchers are carried out in the direction of approaching mainly and businesses. In addition, there is an expert approach. Indeed, the majority of related research looks at the relationship between consumers and businesses (B2C). Therefore, the development of e-SQ scales according to the business-to-business (B2B) relationship will be precious. Obviously, customers can be individuals or organizations, so the nature of e-SQ scale settings is conducted as usual with considering the multidimensionality and multi-level.

2.2 Literature Survey

The purpose of this study is to empirically investigate the dimensions and drivers of entrepreneurial perceptions in the pursuit of emerging e-business opportunities for traditional (or offline) firms. This study introduces the subjectivist theory of entrepreneurship into the IS research context, and identifies three dimensions that make up entrepreneurial perceptions: collaboration perception, planning perception, and operation perception. The authors tested the proposed research model using structural equation modelling (SEM) with survey data collected from 203 firms in China. Results reveal that external pressures and IT infrastructure maturity are positively and significantly related to entrepreneurial perceptions. The results also suggest that IT infrastructure maturity has stronger effects on collaboration perception and planning perception than external pressures. This paper provides clear guidance for entrepreneurs to understand the three entrepreneurial perceptions for emerging e-business opportunity discovery and the driving forces to the entrepreneurial perceptions.

In, Information and communication technology (ICT) has helped to drive increasingly intense global Competition. In the world history the most of the countries are most developed because of they are financially very clear for how to use the high amount of

money in the developing processing own country. We also use the SOA architecture for providing the scalable and reliable service therefor we studied related to the SOA architecture to know how we use to implementation process in our project using Service Oriented Architectures (SOA). We also refer the paper who give the case study information about Scandinavian bank and a Swiss bank These two banks are working on the basis of service-oriented architecture for providing the service for the customer. SOA provides potential for greater organizational agility (and thereby competitiveness). In, in the second paper we learn which type of problems are created in banking system during the different types of transactions. Here discuss about if any region the transaction may be fail then how to avoid it and fixed it. We also studied about Firms in Italy defaulted more against banks with high levels of past losses. This 'selective' default increases where legal enforcement is weak. Poor enforcement thus can create a systematic transaction risk by encouraging banking users to defaulted masse once the continuation value of their bank relationships comes into doubt. In banking sector, the security also must and when we talk about money or property this case is more sensational than we found the security is the major thing to do in banking system. In our project we provide the security questions when customer login with account to prevent the fraud and provide the best security in the easy way application.

CHAPTER 3

FEASIBILITY STUDY

The feasibility study for the proposed Easy Way application reveals promising outcomes across technical, economic, operational, and scheduling dimensions. From a technical standpoint, the required software, hardware, and expertise are readily available, facilitating seamless development and integration. Economic feasibility is supported by a favorable cost-benefit analysis, showcasing a reasonable return on investment over time. Operationally, the study indicates a positive outlook, with users displaying openness to the system, and its alignment with organizational goals is evident. The scheduling feasibility is reinforced by a well-structured project timeline, accounting for dependencies and potential risks. The project's regulatory and legal aspects are deemed feasible, with a commitment to compliance and data privacy measures. The system exhibits scalability and flexibility, ensuring adaptability to future demands. Overall, the feasibility study indicates a strong foundation for the successful development and implementation of the Easy Way service portal.

3.1 Technical Feasibility

Technical Feasibility (TF) analysis in software development can be carried out to verify this hypothesis and learn more about the potential outcomes of the proposed project. This applies across all sectors, providing a brighter future for software development.

The technical feasibility study in software engineering is conducted in various ways depending on the company. Some people may do it in a precise and organized method, while others may do it as needed. However, you must have the following resources -

- Hardware
- Software
- Technology
- Skills and knowledge
- Time and budget for development
- Specialists
- Software development tools

3.1.1 Hardware Feasibility

- Processor must be i3 or more
- Processor: A processor with a minimum clock speed of 1 GHz
- RAM: At least 1 GB RAM

3.1.2 Software Feasibility

Library: React

- Front-end: HTML, CSS, JAVASCRIPT
- Back-end Side Technologies: Node.js, Express.js
- Database Server: MongoDB
- Operating System: Microsoft Windows/Linux

Since the above hardware requirements are available to me and my team and the application, which has been developed using MERN stack i.e. library of React which include HTML, CSS, JavaScript, we have used Visual Studio Code because me and my team member were comfortable in using that software tool for developing the project.

MERN Stack is a JavaScript Stack that is used for easier and faster deployment of full-stack web applications. MERN Stack comprises of 4 technologies namely: MongoDB, Express, React and Node.js. It is designed to make the development process smoother and easier.

- MongoDB: Non-Relational Database
- Express: Node.js web server
- React: JavaScript Frontend Framework
- Node: JavaScript Web Server

3.2 Economic Feasibility

The economic viability of a project considers both the costs and potential returns. Therefore, making a ROM (Rough Order of Magnitude) estimate is normal practice to ascertain financial viability. In the Economic Feasibility study cost and benefit of the project are analysed. This means under this feasibility study a detailed analysis is carried out will be the cost of the project for development which includes all required costs final development hardware and software resources required, design and development costs and operational costs, and so on. After that, it is analysed whether the project will be beneficial in terms of finance for the organization or not. we find the total cost and benefit of the proposed system over the current system. For this project, the main cost is the documentation cost.

3.3 Legal Feasibility

Considering the product's legal viability ensures it won't get you in trouble. For instance, HIPAA compliance is required for any medical software that handles PHI (Protected Health Information). In addition, you must investigate the potential legal threats to your project and how best to mitigate them.

3.4 Operational Feasibility

Implementing a project within the current business environment might impact daily operations. Operational feasibility involves analysing the practicality of implementing the project within the current business environment and determining how it will impact daily operations. This assessment involves undertaking a study to analyse and determine whether and how well the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development. It is the ease and simplicity of operation of the proposed system. The system does not require any special skill set for users to operate it. This shows the management and organizational structure of the project. This project is not built by a team. The management tasks are all to be carried out by a single person. That won't create any management issues and will increase the feasibility of the project.

3.5 Behavioural Feasibility

It evaluates and estimates the user attitude or behaviour toward the development of the new system. It helps in determining if the system requires special effort to educate, retrain, transfer, and change an employee's job status on new ways of conducting business. Establishing the cost-effectiveness of the proposed system i.e. if the benefits do not outweigh the costs, then it is not worth going ahead. In the fast-paced world today

there is a great need for online social networking facilities. Thus, the benefits of this project in the current scenario make it economically feasible. The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis.

3.6 Risk Analysis

Identifies potential risks associated with the project and assesses the impact of those risks on project success. In this a risk mitigation plan is also developed to cope up with the upcoming risks.

Potential Risks:

- Security breaches, system performance issues, and low user adoption.
- Mitigation strategies include security audits, performance testing, and user marketing campaigns.

CHAPTER 4

DATABASE DESIGN

A properly designed database provides you with access to up-to-date, accurate information. Because a correct design is essential to achieving your goals in working with a database, investing the time required to learn the principles of good design makes sense. In the end, you are much more likely to end up with a database that meets your needs and can easily accommodate change. This article provides guidelines for planning a desktop database. You will learn how to decide what information you need, how to divide that information into the appropriate tables and columns, and how those tables relate to each other. You should read this article before you create your first desktop database. Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of an enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept for designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored. The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate on this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keeping control of physical media using hardware resources and software systems such as Database Management System (DBMS).

4.1 Flowchart

A flowchart is a graphical representation of an algorithm. Programmers often use it as a program- planning tool to solve a problem. It makes use of symbols that are connected among them to indicate the flow of information and processing. The process of drawing a flowchart for an algorithm is known as “flowcharting”.

4.1.1 Basic Symbols used in Flowchart Designs

4.1.1.1 Terminal:

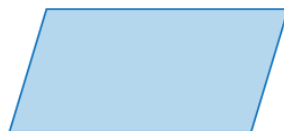
The oval symbol indicates Start, Stop, and Halt in a program’s logic flow. A pause/halt is generally used in a program logic under some error conditions. The terminal is the first and last symbol in the flowchart.



Terminal

4.1.1.2 Input/Output:

A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelograms in a flowchart.



4.1.1.3 Processing:

A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication, and division are indicated by action or process symbol.



Process

4.1.1.4 Decision:

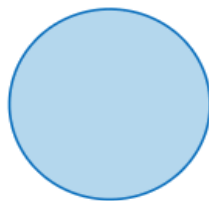
Diamond symbol represents a decision point. Decision-based operations such as yes/no questions or true/false are indicated by diamonds in the flowchart.



Decision

4.1.1.5 Connectors:

Whenever the flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.



4.1.1.6 Flow lines:

Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of the flow of control and the relationship among different symbols of the flowchart.



4.1.2 Flowcharts about the working of the application

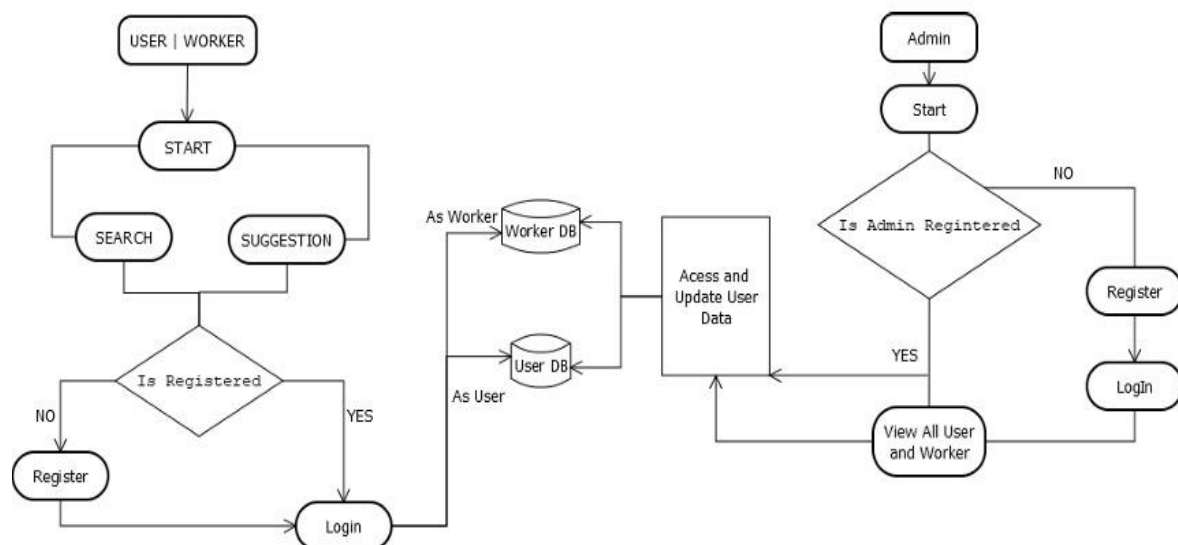


Fig 4.1.2

4.2 Use Case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. Use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

The purpose of a use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as the other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from the other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements actors.

4.2.1 Use case diagram components.

To answer the question, "What is a use case diagram?" you need to first understand its building blocks. Common components include:

Actors: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.

System: A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.

Goals: The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

4.2.2 Use case diagram symbols and notation.

The notation for a use case diagram is pretty straight forward and doesn't involve as many types of symbols as other UML diagrams.

Use cases: Horizontally shaped ovals that represent the different uses that a user might have.

Actors: Stick figures that represent the people actually employing the use cases. ·
Associations: A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.

System boundary boxes: A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.

Packages: A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.

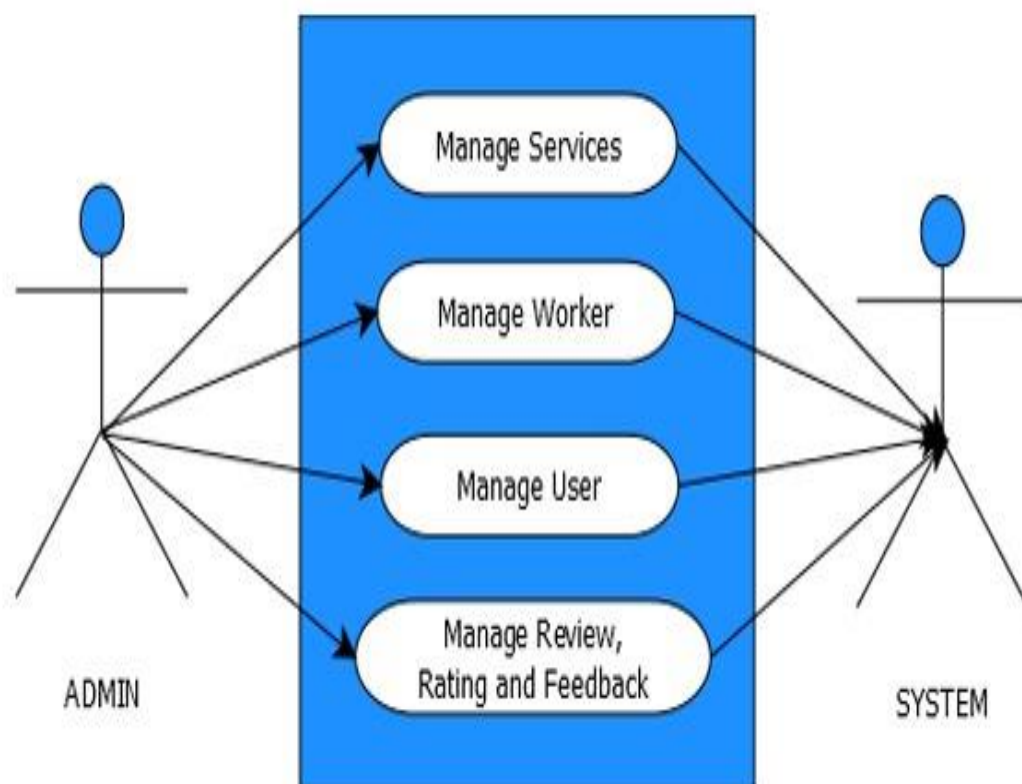


Fig 4.2.1 UCD for admin

The above is the use case diagram of admin which has actors are admin and the system or application where admin can manage the services available in the application, manage workers present in the application or those who have registered at the portal, manages users in the portal who are registering to get the resources of the application and also manages reviews, ratings and feedback of the workers and applications.

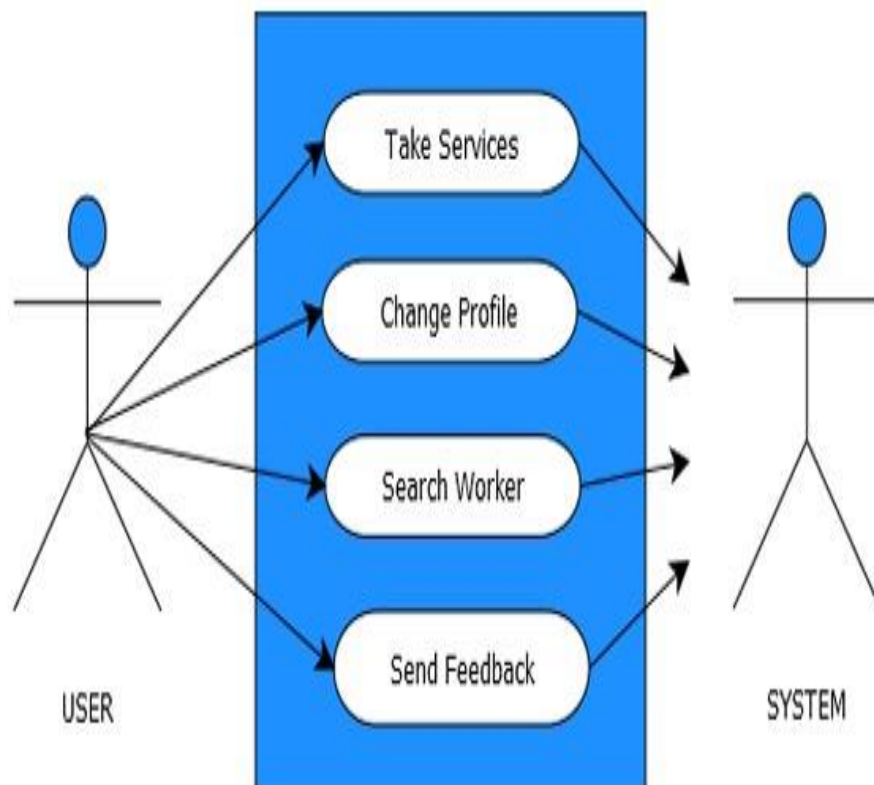


Fig 4.2.2 UCD for user

The above is the use case diagram of the user who can take services from the system or application, change profile on the system, Search for the worker in the system according to their requirement, and can Send feedback to the system regarding the services and the worker.

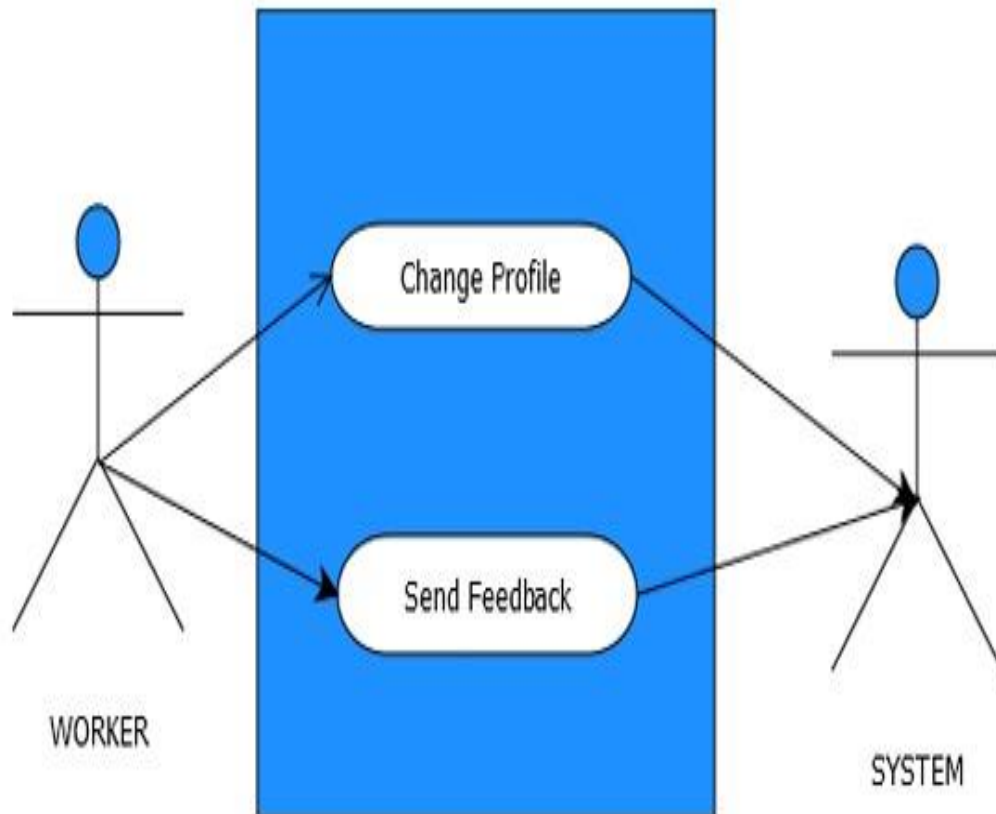


Fig 4.2.3 UCD for worker

The above shows the use case diagram for worker where the actors are worker and the system or the application in which the worker can change their profiles and can send feedback regarding the application experience.

4.3 Data Flow Diagram:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

4.3.1 DFD Levels:

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond.

- Level 0 DFD:

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analysed or modelled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

- Level 1 DFD:

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.

- Level 2 DFD:

DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system's functioning.

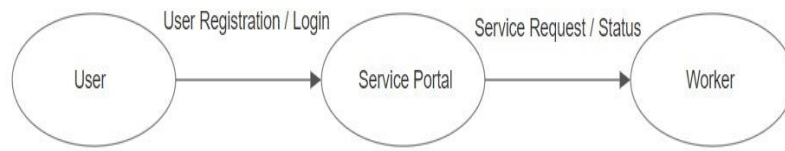


Fig 4.3.1 Context Diagram for Easy Way

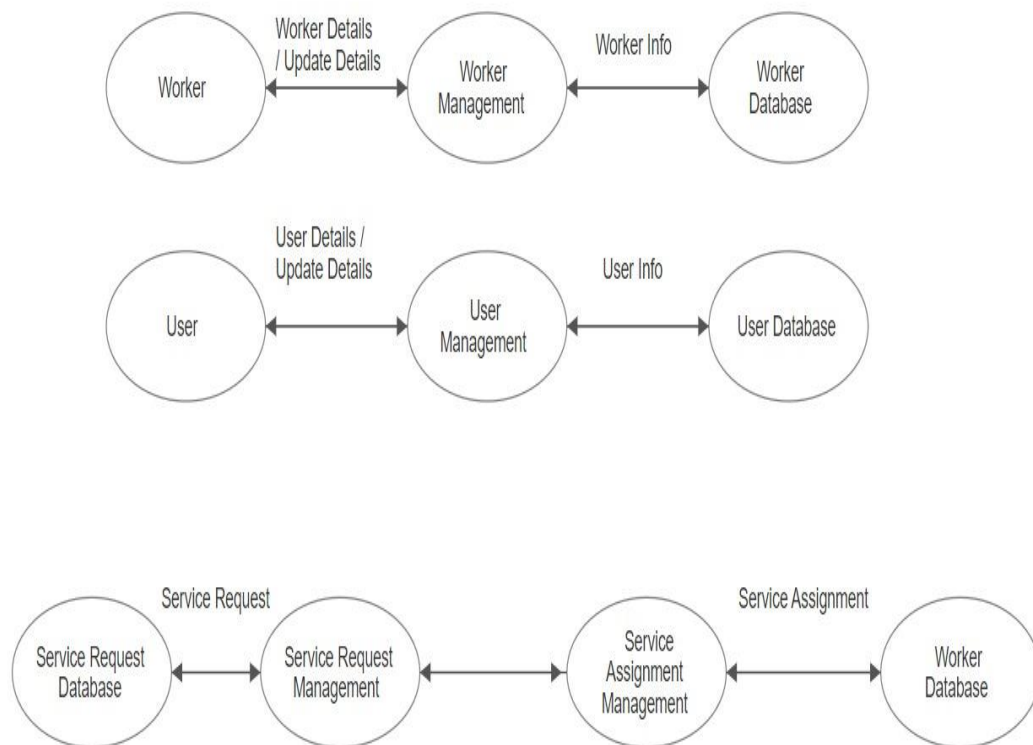


Fig 4.3.2 Level 1 DFD for Easy Way

4.4 ER diagram:

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs. ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

4.4.1 Uses of Entity-Relationship diagram

- **Database design:**

ER diagrams are used to model and design relational databases, in terms of logic and business rules (in a logical data model) and in terms of the specific technology to be implemented (in a physical data model.) In software engineering, an ER diagram is often an initial step in determining requirements for an information systems project. It's also later used to model a particular database or databases. A relational database has an equivalent relational table and can potentially be expressed that way as needed.

- **Database troubleshooting:**

ER diagrams are used to analyse existing databases to find and resolve problems in logic or deployment. Drawing the diagram should reveal where it's going wrong.

- **Business information systems:**

The diagrams are used to design or analyse relational databases used in business processes. Any business process that uses fielded data involving entities, actions and interplay can potentially benefit from a relational database. It can streamline processes, uncover information more easily and improve results.

- **Business process re-engineering (BPR):**

ER diagrams help in analysing databases used in business process re-engineering and in modelling a new database setup.

- **Education:**

Databases are today's method of storing relational information for educational purposes and later retrieval, so ER Diagrams can be valuable in planning those data structures.

- **Research:**

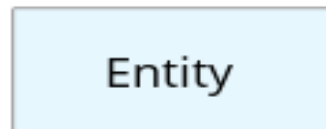
Since so much research focuses on structured data, ER diagrams can play a key role in setting up useful databases to analyse the data.

4.4.2 Components and Features of ER Diagram

ER Diagrams are composed of entities, relationships and attributes. They also depict cardinality, which defines relationships in terms of numbers. Here's a glossary:

Entity

A definable thing—such as a person, object, concept or event—that can have data stored about it. Think of entities as nouns. Examples: a customer, student, car or product. Typically shown as a rectangle.



Entity type:

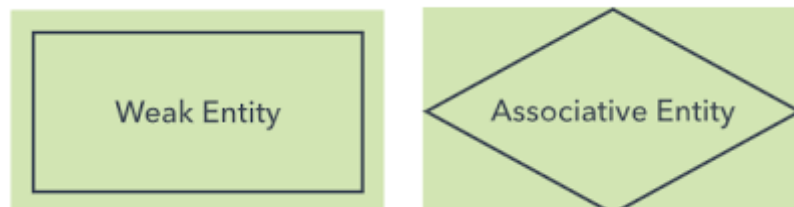
A group of definable things, such as students or athletes, whereas the entity would be the specific student or athlete. Other examples: customers, cars or products.

Entity set:

Same as an entity type, but defined at a particular point in time, such as students enrolled in a class on the first day. Other examples: Customers who purchased last month, cars currently registered in Florida. A related term is instance, in which the specific person or car would be an instance of the entity set.

Entity categories:

Entities are categorized as strong, weak or associative. A **strong entity** can be defined solely by its own attributes, while a **weak entity** cannot. An associative entity associates entities (or elements) within an entity set.



Entity keys:

Refers to an attribute that uniquely defines an entity in an entity set. Entity keys can be super, candidate or primary.

Super key: A set of attributes (one or more) that together define an entity in an entity set.

Candidate key: A minimal super key, meaning it has the least possible number of attributes to still be a super key. An entity set may have more than one candidate key.

Primary key: A candidate key chosen by the database designer to uniquely identify the entity set.

Foreign key: Identifies the relationship between entities.

Relationship:

How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines.

Attribute

A property or characteristic of an entity. Often shown as an oval or circle.



Descriptive attribute: A property or characteristic of a relationship (versus of an entity.)

Attribute categories: Attributes are categorized as simple, composite, derived, as well as single-value or multi-value.

Simple: Means the attribute value is atomic and can't be further divided, such as a phone number.

Composite: Sub-attributes spring from an attribute.

Derived: Attributed is calculated or otherwise derived from another attribute, such as age from a birthdate.



Multi-value:

More than one attribute value is denoted, such as multiple phone numbers for a person.



Single-value:

Just one attribute value. The types can be combined, such as: simple single-value attributes or composite multi-value attributes.

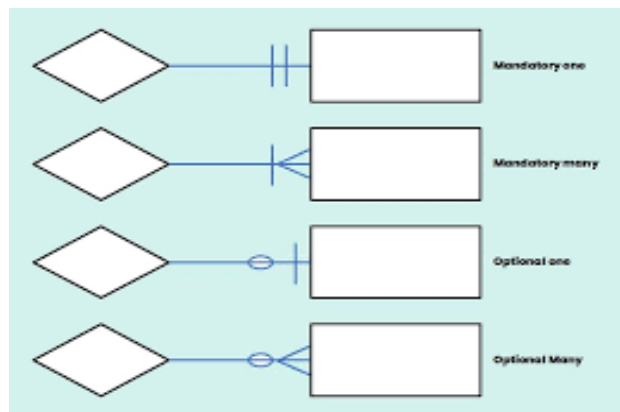
Cardinality:

Defines the numerical attributes of the relationship between two entities or entity sets. The three main cardinal relationships are one-to-one, one-to-many, and many-many.

One-to-one example would be one student associated with one mailing address.

One-to-many example (or many-to-one, depending on the relationship direction): One student register for multiple courses, but all those courses have a single line back to that one student.

Many-to-many example: Students as a group are associated with multiple faculty members, and faculty members in turn are associated with multiple students.



4.4.3 ER Diagram for the Easy Way Application:

Below is the Entity relationship diagram for the Easy way application where all the entities present in the application are mentioned with their relationships with other entities and each entity has its own primary key and attributes.

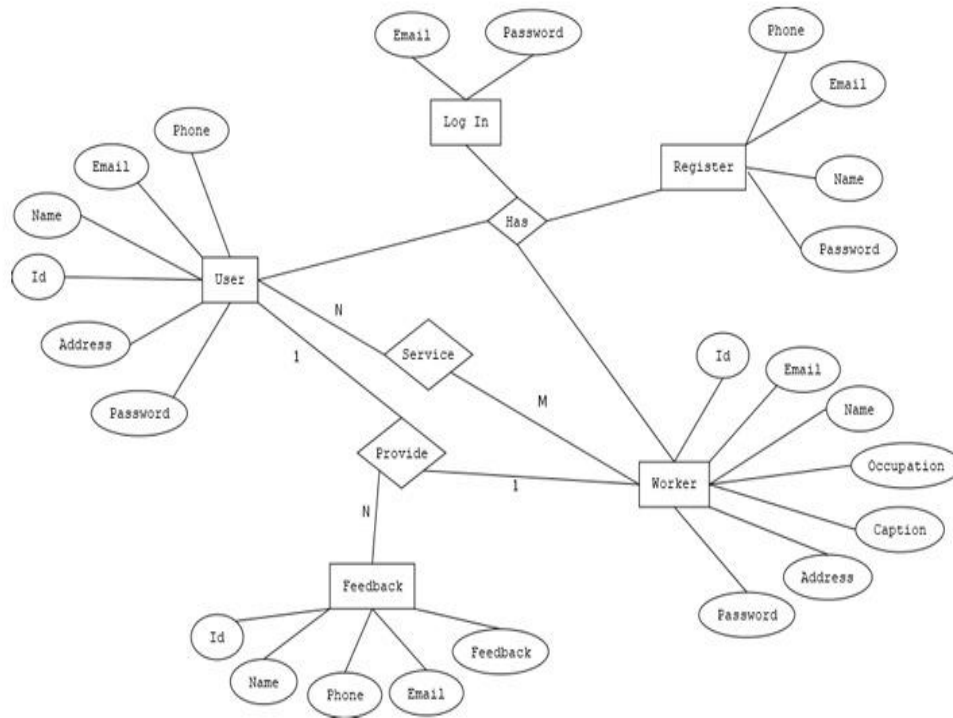


Fig. 4.4.3 ERD for Easy Way

Entities: User, Log In, Register, Worker, Feedback.

Attributes of each entity:

- User: Id, Name, Email, Phone, Address, Password
- Log In: Email, Password
- Register: Name, Email, Phone, Password
- Worker: Id, Email, Name, Occupation, Caption, Address, Password
- Feedback: Id, Name, Phone, Email, Feedback

4.5 Logical Database:

The logical database design is meant to describe the representation of the database in terms of its entities in form of tables and the existing relationships. Below is an illustration of the systems logical design as generated by the visual code studio IDE:

```
const userSchema = mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please add a username!"],
  },

  email: {
    type: String,
    required: [true, "Please add the user email"],
    unique: [true, "Email address is already used!"],
  },

  phone: {
    type: String
  },

  address: {
    type: String,
    required: true
  },

  password: {
    type: String,
    required: [true, "Please add the user password!"],
  },

  imageUrl: {
    type: String
  }
}, {
  timestamps: true,
});
```

Fig 4.5.1 Logical Database Design for User

```
const workerSchema = mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please add a name"],
  },

  email: {
    type: String,
    required: [true, "Please enter an email"],
    unique: true
  },

  phone: {
    type: String,
    required: [true, "Please add a phone number"],
  },

  address: {
    type: String,
    required: true
  },

  password: {
    type: String,
    required: [true, "Please provide a password"],
  },

  occupation: {
    type: String,
  },

  serviceList: [{
    type: String,
  }],

  description: {
    type: String
  }
});
```

Fig 4.5.2 Logical database design for worker


```

const mongoose = require('mongoose');

const userFeedback = mongoose.Schema({
  name : {
    type: String,
    required: [true, "Please add user name"],
  },

  email:{
    type: String,
    required : [true, "Please add user email"],
    unique : [true, "email id already user"],
  },

  rating :{
    type : String,
    required: [true, "Please add some rating"],
  },

  user_message : {
    type : String,
  },
},
{
  timestamps : true,
})

module.exports = mongoose.model("userFeedback", userFeedback);

```

Fig 4.5.3 Logical Database design for feedback

4.6 Physical Database:

As one of the core elements of an Easy Way Application, the database had to be designed in a meticulous systematic manner. This process started at the analysis phase of the project. From the analysis, the researcher was able to identify the necessary collections required for the database and the associated field names, format and length of each table. Below are the images of these collections:

allusers				
Storage size: 32.77 kB	Documents: 159	Avg. document size: 193.00 B	Indexes: 1	Total index size: 36.86 kB
otprecords				
Storage size: 8.19 kB	Documents: 0	Avg. document size: 0 B	Indexes: 1	Total index size: 12.29 kB
requests				
Storage size: 20.48 kB	Documents: 4	Avg. document size: 104.00 B	Indexes: 1	Total index size: 36.86 kB
userfeedbacks				
Storage size: 20.48 kB	Documents: 4	Avg. document size: 276.00 B	Indexes: 2	Total index size: 73.73 kB

Fig 4.6.1 Physical database for Easy Way

otprecords				
Storage size: 8.19 kB	Documents: 0	Avg. document size: 0 B	Indexes: 1	Total index size: 12.29 kB
requests				
Storage size: 20.48 kB	Documents: 4	Avg. document size: 104.00 B	Indexes: 1	Total index size: 36.86 kB
userfeedbacks				
Storage size: 20.48 kB	Documents: 4	Avg. document size: 276.00 B	Indexes: 2	Total index size: 73.73 kB
users				
Storage size: 20.48 kB	Documents: 2	Avg. document size: 377.00 B	Indexes: 2	Total index size: 73.73 kB
workers				
Storage size: 20.48 kB	Documents: 7	Avg. document size: 438.00 B	Indexes: 2	Total index size: 73.73 kB

Fig 4.6.2 Physical database for Easy Way Application

CHAPTER 5

IMPLEMENTATION

5.1 Implementation

Implementation is the realization of an application, execution of a plan, idea, model, design, specification, standard, algorithm, policy, the administration or management of a process or objective.

5.1.1 Overview of Technologies Used

The Easy Way is developed using MERN Stack where React library is used for front-end and Node.js and Express.js for back-end and MongoDB as database server. All of them played very crucial role in building the project.

MERN stack is a collection of technologies that enables faster application development. It is used by developers worldwide. The main purpose of using MERN stack is to develop apps using JavaScript only. This is because the four technologies that make up the technology stack are all JS-based. Thus, if one knows JavaScript (and JSON), the backend, frontend, and database can be operated easily.

MERN Stack is a compilation of four different technologies that work together to develop dynamic web apps and websites.

It is a contraction for four different technologies as mentioned below:

- M - MongoDB
- E – Express.JS
- R - ReactJS
- N - NodeJS

Considering you've downloaded and configured all the four aforementioned technologies; you need to know how to create a new project folder to get started with the MERN stack. Next, you need to open the folder on the CMD (or terminal) and enter the following command to initialize a package .Json file:

```
npm init;
```

You can install modules by using:

```
npm install module_name -save
```

MERN Stack Components

There are four components of the MERN stack. Let's discuss each of them one by one.

- The first component is MongoDB, which is a NoSQL database management system.
- The second MERN stack component is ExpressJS. It is a backend web application framework for NodeJS.
- The third component is ReactJS, a JavaScript library for developing UIs based on UI components.
- The final component of the MERN stack is NodeJS. It is a JS runtime environment, i.e., it enables running JavaScript code outside the browser.

5.1.2 Development Environment Setup

The initial phase started with setting up the environment in Visual Studio Code IDE by creating a collection. This setup provided the server environment facilitating seamless testing and development.

5.1.3 Database Design

The backbone of the Easy Way relied on a well-structured database. The design incorporated multiple tables enabling efficient data storage and retrieval.

5.1.4 Frontend Development

The frontend development was entirely based on library of React. The React Library helped in creating a user-friendly user interface for the users and workers. Key aspects of this are:

React:

- **React** (also known as **React.js** or **ReactJS**) is a free and open-source front-end JavaScript library for building user interfaces based on components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies.
- React can be used to develop single-page, mobile, or server-rendered applications with frameworks like Next.js. Because React is only concerned with the user interface and rendering components to the DOM, React applications often rely on libraries for routing and other client-side functionality. A key advantage of React is that it only re-renders those parts of the page that have changed, avoiding unnecessary re-rendering of unchanged DOM elements.

HTML:

- **Hypertext Markup Language (HTML)** is the standard markup language for documents designed to be displayed in a web browser. It defines the content and structure of web content. It is often assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
- Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for its appearance.
- HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as **** and **<input>** directly introduce content into the page. Other tags such as **<p>** and **</p>** surround and provide information about document text and may include sub-element tags. Browsers do not display the HTML tags but use them to interpret the content of the page.
- HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. A form of HTML, known as HTML5, is used to display video and audio, primarily using the **<canvas>** element, together with JavaScript.

CSS:

Cascading Style Sheets (CSS) is a style sheet language used for specifying the presentation and styling of a document written in a markup language such

as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of content and presentation, including layout, colors, and fonts. This separation can improve content accessibility provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name *cascading* comes from the specified priority scheme to determine which declaration applies if more than one declaration of a property match a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL. CSS is also used in the GTK widget toolkit.

JavaScript:

JavaScript, often abbreviated as **JS**, is a programming language and core technology of the Web, alongside HTML and CSS. 99% of websites use JavaScript on the client side for webpage behaviour.

Web browsers have a dedicated JavaScript engine that executes the client code. These engines are also utilized in some servers and a variety of apps. The most popular runtime system for non-browser usage is Node.js.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard.^[11] It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

5.1.5 Backend Development

The backend was developed using MongoDB, Node.js, Express.js which helped in connectivity of frontend components with backend.

MongoDB:

MongoDB is a source-available, cross-platform, document-oriented database program. Classified as a NoSQL database product, MongoDB utilizes JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and current versions are licensed under the Server Side Public License (SSPL). MongoDB is a member of the MACH Alliance.

Node.js:

Node.js is an open-source server environment that uses JavaScript on server. A Node.js application runs within a single process, without generating a new thread for each request. Node.js includes asynchronous I/O primitives as a part of its standard library, which prevents JavaScript code from blocking and, in general, libraries in Node.js are developed using non-blocking paradigms. This makes blocking behaviour the exception instead of the rule.

It is developed by Ryan Dahi in the year 2009 and v20.9 is the latest version of Node.js. Because it is cross-platform one can easily run on Windows, Linux, Unix, macOS and more.

Node.js has a unique advantage because millions of frontend developers who write JavaScript for the browser can now write server-side code without needing to learn a completely new language. Node.js is one of the popular choices for developing RESTful APIs, microservices and web application.

Express JS:

Express.js is a fast, flexible and minimalist web framework for Node.js. It's effectively a tool that simplifies building web applications and APIs using JavaScript on the server side. Express is an open-source that is developed and maintained by the Node.js foundation.

Express.js offers a robust set of features that enhance your productivity and streamline your web application. It makes it easier to organize your application's functionality with middleware and routing. It adds helpful utilities to Node HTTP objects and facilitates the rendering of dynamic HTTP objects.

Express is a user-friendly framework that simplifies the development process of Node applications. It uses JavaScript as a programming language and provides an efficient way to build web applications and APIs. With Express, you can easily handle routes, requests, and responses, which makes the process of creating robust and scalable applications much easier.

CHAPTER 6

TESTING

Testing is critical for a newly developed system as a prerequisite for it being put into an environment where the end users can use it. Exhaustive testing is conducted to ensure accuracy and reliability and to ensure that bugs are detected as early as possible. In the process of designing the RMS, three levels of testing were conducted, namely, unit testing, integration testing and system testing.

6.1 Unit Testing

Unit test is where the system is tested partially and independently, component by component, to ensure that particular portion or module is workable within it. In the development of the records management system, each component was tested independently before finally integrating each of them into one system. This test was used by the researcher to verify that every input of data was assigned to the appropriate tables and fields.

Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.

A unit is a single testable part of a software system and tested during the development phase of the application software.

The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. White box testing approach used for unit testing and usually done by the developers.

Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application

independently or one by one, and this process is known as Unit testing or components testing

6.1.1 Techniques of Unit Testing:

- **White Box Testing:** White box testing techniques analyse the internal structures the used data structures, internal design, code structure and the working of the software rather than just the functionality as in Blackbox testing. It is also called glass box testing or clear box testing or structural testing.
- **Black Box Testing:** Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.
- **Grey Box Testing:** Grey box testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a combination of black box and white box testing because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing

6.1.2 Unit Testcases for each module:

Module: Home

Test Case 1: Verify that the home page loads successfully and displays the intended content (e.g., welcome message, service categories).

Test Case 2: Verify that navigation links to other modules (Services, User Reviews, Contact, etc.) are functional and direct users to the correct pages.

Module: Services

Test Case 3: Verify that the services page displays a list of available services with descriptions.

Test Case 4: Verify that clicking on a specific service redirects the user to a detailed page about that service.

Test Case 5: (Optional) If there's a search functionality, test that it allows users to find specific services by keyword.

Module: User Reviews

Test Case 6: Verify that users can leave reviews for completed services. (This might require user login functionality)

Test Case 7: Verify that user reviews are displayed correctly on the page, including rating, username (if applicable), and review text.

Test Case 8: Verify that appropriate validation is in place for user reviews (e.g., minimum character length, preventing spam).

Module: Contact

Test Case 9: Verify that the contact page displays contact information (e.g., email address, phone number, contact form).

Test Case 10: Verify that the contact form functionality works correctly, allowing users to submit messages.

Test Case 11: Verify that form validation ensures required fields are filled and data is submitted in the expected format (e.g., valid email address).

Module: Worker (Assuming this is for worker registration/management)

Test Case 12: Verify that workers can register on the platform with required information (e.g., name, skills, qualifications).

Test Case 13: Verify that worker registration validates user input (e.g., email format, strong password).

Test Case 14: (Optional) Verify that worker profiles are displayed correctly on the platform, showcasing skills and qualifications (if applicable).

Module: Registration

Test Case 15: Verify that user registration allows creating new accounts with required information (e.g., name, email, password).

Test Case 16: Verify that registration validates user input (e.g., email format, strong password).

Test Case 17: Verify that successful registration redirects users to a confirmation page or logs them in automatically.

6.2 Integration Test

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit Testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software

modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.

Integration testing for the Easy Way application involved verifying the interaction and interoperability of different modules and functionalities to ensure they work together seamlessly.

6.2.1 Techniques of Integrating Testing:

- **Incremental Approach:** In the Incremental Approach, modules are added in ascending order one by one or according to need. The selected modules must be logically related. Generally, two or more than two modules are added and tested to determine the correctness of functions. The process continues until the successful testing of all the modules.
- **Top-down Approach:** The top-down testing strategy deals with the process in which higher level modules are tested with lower-level modules until the successful completion of testing of all the modules. Major design flaws can be detected and fixed early because critical modules tested first. In this type of method, we will add the modules incrementally or one by one and check the data flow in the same order.
- **Bottom – Up Approach:** The bottom to up testing strategy deals with the process in which lower-level modules are tested with higher level modules until the successful completion of testing of all the modules. Top-level critical modules are tested at last, so it may cause a defect. Or we can say that we will be adding the modules from bottom to the top and check the data flow in the same order.
- **Big Bang Approach:** In this approach, testing is done via the integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult. Since this testing can be done after completion of all modules due to that testing team has less time for execution of this process so that internally linked interfaces and high-risk critical modules can be missed easily.

6.2.2 Integrated testcases for application:

Here are some integration testing test cases for the Easy Way service portal, building upon the unit tests you already have:

Focus on Interactions Between Modules

Integration testing aims to verify how different modules work together. Here's how we can test interactions between the modules you mentioned:

Home & Services:

- **Test Case 1:** Verify that clicking a service category on the home page displays a filtered list of relevant services on the Services page.
- **Test Case 2:** Verify that clicking a specific service from the home page successfully redirects to the detailed service page with information retrieved from the Services module.

Services & User Reviews:

- **Test Case 3:** Verify that after completing a service (simulated or through a test account), users can access the appropriate service page and leave a review. (This might involve testing user login integration)
- **Test Case 4:** Verify that user reviews for a specific service are displayed on the corresponding service detail page, integrating data from Services and User Reviews modules.

Services & Contact:

- **Test Case 5:** Verify that the contact information displayed on the Contact page includes an email address linked to the service provider associated with a chosen service (if applicable).
- **Test Case 6:** (Optional) Verify that the contact form on the Contact page allows users to specify a chosen service when submitting a message, potentially routing it to the relevant service provider (testing integration with Services module).

Worker & Services:

- **Test Case 7:** Verify that registered workers can log in and access a dashboard or profile where they can manage the services they offer (assuming worker management functionality exists). (Tests integration between Worker and Services modules)
- **Test Case 8:** Verify that worker profiles displayed on the platform (if applicable) link to the services they offer, integrating data from Worker and Services modules.

Registration & Other Modules:

- **Test Case 9:** Verify that after successful user registration, users can log in and access functionalities relevant to their role (e.g., service booking, leaving reviews). This tests login integration with various modules.
- **Test Case 10:** (Optional) Verify that during registration, users can choose a service category they specialize in, potentially linking their profile to relevant services upon successful registration (testing integration between Registration and Services).

6.3 SYSTEM TESTING

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.

To check the end-to-end flow of an application or the software as a user is known as System testing. In this, we navigate (go through) all the necessary modules of an application and check if the end features or the end business works fine, and test the product as a whole system.

It is end-to-end testing where the testing environment is similar to the production environment.

6.3.1 Types of System Testing:

- **Performance Testing:** Performance Testing is a type of software testing that is carried out to test the speed, scalability, stability and reliability of the software product or application.
- **Load Testing:** Load Testing is a type of software Testing which is carried out to determine the behaviour of a system or software product under extreme load.
- **Stress Testing:** Stress Testing is a type of software testing performed to check the robustness of the system under the varying loads.
- **Scalability Testing:** Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request load.

6.3.2 System Test Cases of the entire system:

System testing focuses on verifying the entire Easy Way service portal as a whole, ensuring it meets user needs and functional requirements. Here are some test cases to consider:

User Registration & Login:

- **Test Case 1:** Verify that new users can register successfully using a valid email address and strong password.
- **Test Case 2:** Verify that existing users can log in with their registered credentials.

- **Test Case 3:** Verify error handling for failed login attempts due to incorrect username/password or account lockout.

Service Booking & Management (if applicable):

- **Test Case 4:** Verify that registered users can browse services, select a desired service, and complete a booking process.
- **Test Case 5:** Verify that service providers (workers) receive notifications or can view upcoming bookings in their dashboard (if applicable).
- **Test Case 6:** Verify that users can cancel or reschedule bookings within a specific timeframe (if allowed by the system).

Search Functionality:

- **Test Case 7:** Verify that the search function allows users to find specific services by keyword, category, or other relevant filters.
- **Test Case 8:** Verify that search results are displayed accurately and relevant to the user's search query.

User Reviews & Ratings:

- **Test Case 9:** Verify that registered users can leave reviews and ratings for completed services.
- **Test Case 10:** Verify that user reviews are moderated or filtered to prevent inappropriate content (if applicable).
- **Test Case 11:** Verify that average service ratings are calculated and displayed correctly.

Contact & Support:

- **Test Case 12:** Verify that users can submit inquiries or feedback through the contact form on the Contact page.
- **Test Case 13:** Verify that the system sends email notifications or generates support tickets for submitted contact messages.
- **Test Case 14:** (Optional) Verify that a live chat functionality (if implemented) allows users to connect with customer support representatives for real-time assistance.

Security Testing:

- **Test Case 15:** Verify that user passwords are stored securely using encryption techniques.
- **Test Case 16:** Verify that the system is protected against common security vulnerabilities like SQL injection or cross-site scripting (XSS).
- **Test Case 17:** Verify that user data (personal information, payment details) is transmitted securely using HTTPS protocol.

6.4 ACCEPTANCE TESTING

Acceptance testing is formal testing based on user requirements and function processing. It determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of Black Box testing where the number of required users involved testing the acceptance level of the system. It is the fourth and last level of software testing. User acceptance testing (UAT) is a type of testing, which is done by the customer before accepting the final product. Generally, UAT is done by the customer (domain expert) for their satisfaction, and check whether the application is working according to given business scenarios, real-time scenarios. In this, we concentrate only on those features and scenarios which are regularly used by the customer or mostly user scenarios for the business or those scenarios which are used daily by the end-user or the customer.

6.4.1 Acceptance Testing Test Cases for Easy Way Service Portal

Acceptance testing involves real users or stakeholders validating the Easy Way service portal meets their expectations and business requirements. Here are some acceptance test cases focusing on user experience and business goals:

User-Centric Testing:

- **Test Case 1:** Verify that users can easily navigate the portal and find the information or services they need.
- **Test Case 2:** Verify that the user interface is intuitive and user-friendly, with clear instructions and labels.
- **Test Case 3:** Verify that the registration and login process is simple and straightforward for new users.
- **Test Case 4:** Verify that users can complete desired actions (booking services, leaving reviews, contacting support) efficiently and without errors.
- **Test Case 5:** Verify that the portal is responsive and performs well on different devices (desktop, mobile, tablet).

Business-Oriented Testing:

- **Test Case 6:** Verify that the portal facilitates easy service discovery for potential customers.
- **Test Case 7:** Verify that the platform allows for efficient service booking and management for both users and service providers (if applicable).
- **Test Case 8:** Verify that the user review system helps build trust and credibility for service providers.
- **Test Case 9:** Verify that the contact form and support features provide effective channels for user communication.
- **Test Case 10:** Verify that the portal generates reports or analytics that can be used to track user activity, identify trends, and measure overall platform effectiveness.

6.5 SOFTWARE VERIFICATION AND VALIDATION

6.5.1 Software Verification:

Verification testing includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product. It is also known as static testing, where we are ensuring that "we are developing the right product or not". And it also checks that the developed application fulfilling all the requirements given by the client.

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfils the requirements that we have. Verification is Static Testing.

Activities involved in verification:

- Inspections
- Reviews
- Walkthroughs
- Desk-checking

6.5.2 Software Validation

Validation testing is testing where tester performed functional and non-functional testing. Here functional testing includes Unit Testing (UT), Integration Testing (IT) and System Testing (ST), and non-functional testing includes User acceptance testing (UAT). Validation testing is also known as dynamic testing, where we are ensuring that "we have developed the product right." And it also checks that the software meets the business needs of the client.

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the Dynamic Testing.

Activities involved in validation:

- Black box testing
- White box testing
- Unit testing
- Integration testing

6.6 TEST PROCEDURE

A test procedure is a formal specification of test cases to be applied to one or more target program modules. Test procedures are executable. A process called the VERIFIER applies a test procedure to its target modules and produces an exception report indicating which test cases, if any, failed.

Test procedures facilitate thorough software testing by allowing individual modules or arbitrary groups of modules to be thoroughly tested outside the environment in which they will eventually reside. Test procedures are complete, self-contained, self-validating and execute automatically. Test procedures are a deliverable product of the software development process and are used for both initial checkout and subsequent regression testing of target program modifications.

Test procedures are coded in a new language called TPL (Test Procedure Language). The paper analyses current testing practices, describes the structure and design of test procedures and introduces the Fortran Test Procedure Language.

6.7 TEST CASES

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

CHAPTER 7

DESIGN

Web design encompasses many different skills and disciplines in the production and maintenance of websites. The different areas of web design include web graphic design; user interface design (UI design); authoring, including standardised code and proprietary software; user experience design (UX design); and search engine optimization. Often many individuals will work in teams covering different aspects of the design process, although some designers will cover them all. The term "web design" is normally used to describe the design process relating to the front-end (client side) design of a website including writing markup. Web design partially overlaps web engineering in the broader scope of web development. Web designers are expected to have an awareness of usability and be up to date with web accessibility guidelines.

Here's why implementing these practices is important for your e-Commerce website:

- **To provide the best customer experience possible:** A great customer experience is about making it really simple for customers to buy products from your website, while also making it a great pleasure for them. These practices will help you create a website experience your customers will love.
- **To boost sales through conversion rate optimisation:** Great design will help you convert more website visitors into paying customers, boosting your sales and enhancing business performance.
- **To improve customer retention:** You don't want website visitors or existing customers to leave your website and buy the same product from someone else. A great design will help you grab the attention of your website visitors, and encourage your customers to be loyal towards your brand.
- **To reinforce your brand:** Great design will speak volumes about who you are, what your brand stands for, and help you be perceived appropriately. At the end of the day, design is about great communication!

- **To build customer relationships:** People tend to trust websites that are designed well, and therefore, will want to engage more with them. This helps with building great customer relationships with trust at its foundation.

7.1 Login:

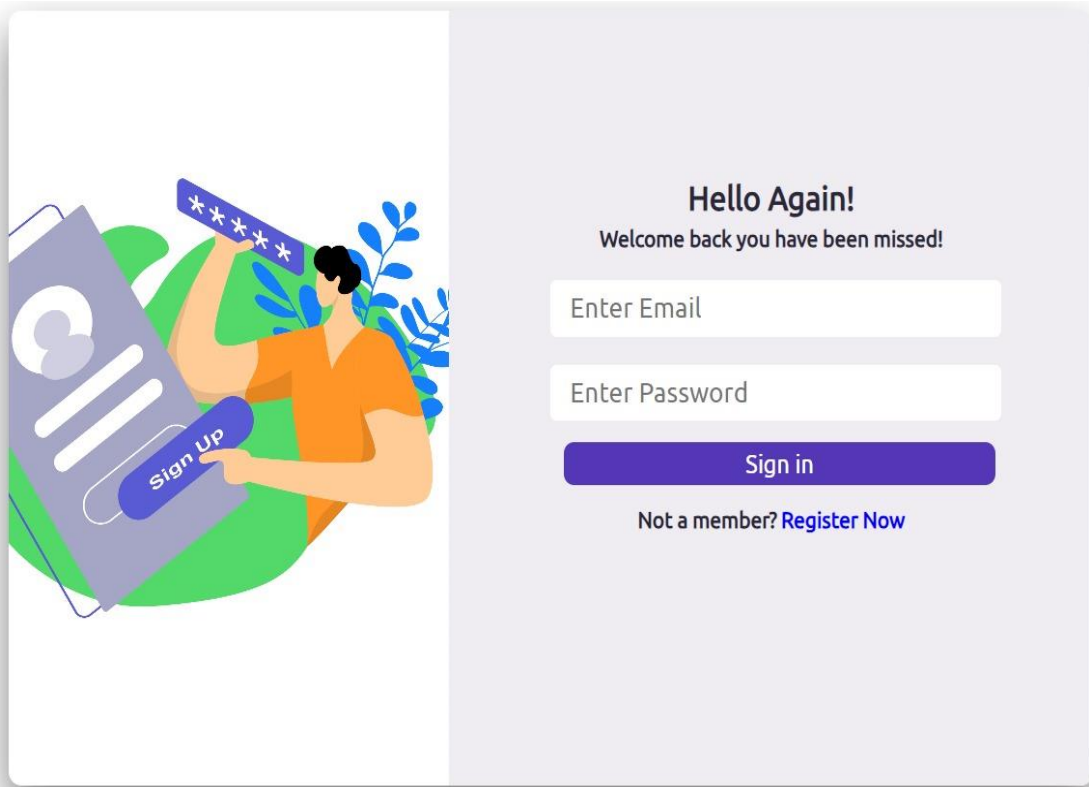


Fig 7.1 Login Page

7.2 Registration:

— Register —

Type *
User

Name *

Phone *

User Email *

Address *

Password *

Profile Image *
Choose File No file chosen

OTP

Fig 7.2 Registration page

7.3 Home Page:



Fig 7.3 Home page

7.4 Services:



Fig 7.4

7.5 Contact Us:

Contact Us

Get In Touch With Us

Welcome to the EasyWay services we provide an efficient way to find the worker for daily home services like electrician, cleaner, plumber, etc. Contact us for any queries.




Phone Number
(+91)8210755399



Email ID
AnshumanPatek369@gmail.com

Fig 7.5 Contact Us

7.6 About Us:



About Us

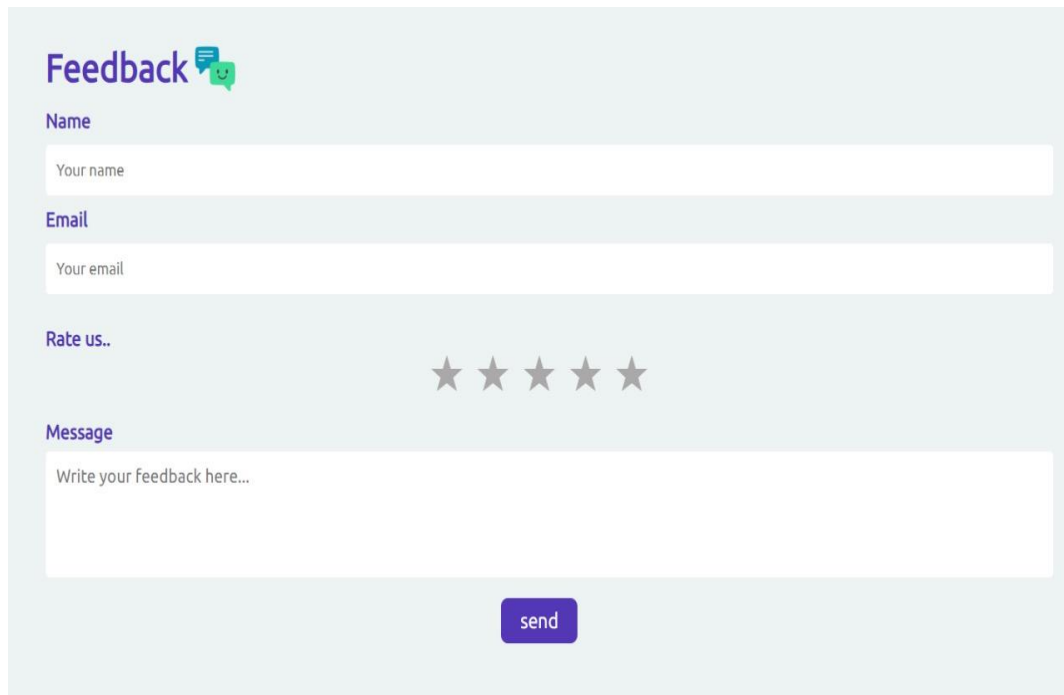
Why will you Choose our services?

Efficient, Friendly, Residential & Commercial Workers. Your Satisfaction is guaranteed

- ✓ Easy to get help
- ✓ Saves your time
- ✓ Seamless Communication
- ✓ Provide professional workers

Fig 7.6 About Us

7.7 Feedback:



The feedback form is titled "Feedback" with a speech bubble icon. It contains four input fields: "Name" (placeholder: "Your name"), "Email" (placeholder: "Your email"), "Rate us.." (with five star icons), and "Message" (placeholder: "Write your feedback here..."). A purple "send" button is located at the bottom right.

Fig 7.7 Feedback

7.8 User Reviews:

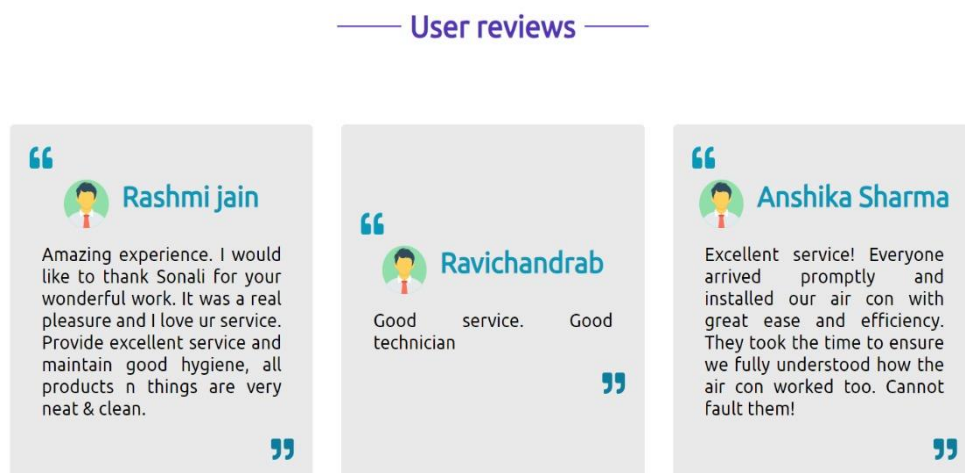


Fig 7.8 User reviews

CHAPTER 8

EVALUATION AND CONCLUSION

8.1 Evaluation

In the attempt to evaluate the designed system, it is imperative that the researcher look back at the predefined functionalities, goals and objectives and analyse those in relation to the expectations met by the system. The Bank Management System was evaluated based on the set of predefined objectives and expected functionalities it was able to fulfil. The Bank Management System was designed to facilitate efficient transaction management in banking system by providing an efficient, reliable computerized banking management system and after a careful evaluation process; it met a considerable portion of those expectations.

The main objective was to design a system that enables faster and more efficient storage, retrieval and updating of customer's account's records. As far as this is concerned, the system met this expectation by giving direct benefit to the customer such as fast cash deposition, withdrawal etc. Analysis was successfully completed. This evaluation is based on the fact that data requirements were collected that successfully enabled the design and development of the system.

The design objectives of creating an efficient bank management system was further accomplished with the creation of add, delete, search and edit functionalities in the system that not only enable computerized but rather efficient, reliable and fast data entry. All these functionalities possess a relatively high level of accuracy. In evaluating this objective in relation to the system's performance, it would therefore be accurate to state that it was achieved to a large extent.

8.2 Limitations of the System

Acceptance testing, while crucial for ensuring a successful product, has its limitations. Here are some key points to consider:

Scope Limitations:

Focus on User Experience: Acceptance testing primarily focuses on validating the user experience and ensuring the system meets user expectations. It might not comprehensively test underlying technical aspects or catch intricate software bugs.

Real-world Challenges:

Limited User Pool: Acceptance testing often involves a limited group of users. This might not capture the entire range of user behaviours or diverse use cases that could arise in a real-world scenario with a broader user base.

Subjectivity in Feedback: User feedback during acceptance testing can be subjective. It's important to analyse it carefully and identify recurring themes or areas needing improvement.

Cost and Time:

Resource Intensive: Involving real users and stakeholders can be time-consuming and resource-intensive. It's important to balance the scope of acceptance testing with project timelines and budget constraints.

Incomplete Picture:

Not a Replacement for Other Testing: Acceptance testing should not be seen as a replacement for unit, integration, or system testing. It complements these other testing methods by providing a user-centric perspective on the final product.

Here's how we can address these limitations:

Combine with Other Testing: Conduct thorough unit, integration, and system testing before user acceptance testing.

Refine Test Cases: Use a risk-based approach to prioritize acceptance test cases based on critical functionalities and potential user impact areas.

Diverse User Groups: Try to involve a diverse set of users in acceptance testing to get a broader range of perspectives.

Structured Feedback: Use standardized surveys or questionnaires to gather more objective user feedback during acceptance testing.

Iterative Process: Be prepared to iterate on the system based on user feedback and address any issues identified during acceptance testing.

By understanding these limitations and taking steps to mitigate them, you can ensure that acceptance testing is a valuable tool for launching a successful Easy Way service portal.

8.3 Problems Encountered

Here are some common problems encountered during acceptance testing (UAT) and how to address them:

Insufficient Planning and Coordination:

Problem: Lack of clear goals, poorly defined test cases, or inadequate communication between development, testing, and stakeholders can lead to confusion and delays during UAT.

Solution: Establish a clear UAT plan with defined objectives, test scope, and timelines. Ensure proper communication channels exist for all involved parties.

Inadequate User Involvement:

Problem: Not involving the right users or having a limited number of testers can result in overlooking user needs and missing critical usability issues.

Solution: Involve representative users from the target audience in UAT. Encourage them to explore the system freely and provide feedback based on their real-world experience.

Unrealistic Expectations:

Problem: Users or stakeholders might have unrealistic expectations about the system's capabilities, leading to frustration during UAT.

Solution: Clearly communicate the system's functionalities and limitations to users beforehand. Manage expectations and focus on testing functionalities as designed.

Ambiguous Requirements:

Problem: Unclear or poorly defined requirements can make it difficult to determine if the system meets expectations. Testers might struggle to decide whether certain behaviours are defects or intended functionality.

Solution: Ensure requirements are well-defined, documented, and accessible to all parties involved in UAT. Clarify any ambiguities before testing commences.

Limited Timeframe:

Problem: Rushing through UAT to meet deadlines can lead to missed issues and a higher risk of bugs escaping into production.

Solution: Allocate sufficient time for UAT based on the project's complexity. Prioritize critical functionalities for testing and address them first.

Poor Defect Management:

Problem: Inefficient reporting, tracking, and resolution of defects identified during UAT can create bottlenecks and hinder progress.

Solution: Establish a clear process for reporting, prioritizing, and resolving defects. Use a defect management tool to track issues and ensure timely fixes.

By being aware of these potential problems and taking proactive measures to address them, you can ensure a smooth and effective UAT process for your Easy Way service portal.

8.4 Recommendation/Future Search

Here are some recommendations to ensure successful acceptance testing for your Easy Way service portal:

Prior to Acceptance Testing:

Solid Foundation: Ensure thorough unit, integration, and system testing have been conducted before involving users in acceptance testing. This reduces the likelihood of encountering basic bugs during UAT.

Refined Test Cases: Focus acceptance testing on user experience and critical functionalities. Prioritize test cases based on potential user impact and business goals.

Diverse User Group: Recruit a representative group of users from your target audience for acceptance testing. This helps identify usability issues relevant to real-world usage patterns.

Clear Communication: Clearly communicate the goals, scope, and limitations of acceptance testing to all stakeholders (users, testers, developers). This ensures everyone has aligned expectations.

Structured Feedback: Prepare surveys or questionnaires to gather user feedback during acceptance testing. This helps capture user experience in a structured and quantifiable manner.

During Acceptance Testing:

Training and Support: Provide users with basic training on the system's functionalities and how to participate in acceptance testing effectively. Offer support throughout the process to address any questions or concerns.

Bug Reporting: Establish a clear and efficient process for users to report bugs or usability issues encountered during testing. Use a defect management tool to track and prioritize these issues.

Iterative Approach: Be prepared to iterate on the system based on user feedback. Address high-priority bugs and usability issues promptly.

Following Acceptance Testing:

Defect Resolution: Ensure all critical defects identified during acceptance testing are addressed and resolved before launch. Track the resolution process and communicate updates to stakeholders.

Post-Launch Monitoring: Continue to monitor user feedback and system performance after launch. This helps identify any lingering issues or areas for improvement.

Future Search:

Easy Way Services provides various features which complement the information system and increases the productivity of the Worker service. There are various possibilities which we will be looking forward to implement to enhance the application's functionality and user experience.

- We will add location filter so that user can access the details of workers nearby.
- Login/Register using social media authentication like Google, Facebook and twitter
- We will add live location tracker, so that the user can track the location of the worker.
- We will launch this web-app as an Android app in the future.
- Digital payment options can be added for the convenience of both the user and the worker.

8.5 Conclusion

The project entitled” Easy Way” is developed using HTML, CSS, and Library React as the front-end, Node JS and its Library for the back-end, and MongoDB as a database to computerize the process of Worker Services. This project covers the essential features required by Workers and Users.

BIBLIOGRAPHY AND REFERENCES

React documentation <https://react.dev/>

Node Package Manager documentation <https://www.npmjs.com/>

Node JS documentation <https://nodejs.org/en/docs>

Firebase documentation <https://firebase.google.com/docs/storage/web/upload-files>

Node Mailer documentation <https://nodemailer.com/about/>

Vite React Guide <https://vitejs.dev/guide/>

JSON Web Token documentation <https://jwt.io/introduction>

Cookie-Parser documentation <https://www.npmjs.com/package/cookie-parser>