



FINAL YEAR PROJECT REPORT

BS (SOFTWARE ENGINEERING)

“SECURE GUARD HUB”

SUBMITTED BY

HAMZA MUZAFFAR	51205
ALI HARIS KHAN	50753
RAJA SHAZIL	51747
MUHAMMAD UMAR	51955

SUPERVISOR

FAHAD NAJEEB

COORDINATOR

DR. AARIJ MAHMOOD HUSSAAN

FACULTY OF ENGINEERING, SCIENCE AND TECHNOLOGY

IQRA UNIVERSITY, KARACHI

MARCH 2023



FACULTY OF ENGINEERING, SCIENCE AND TECHNOLOGY

DEPARTMENT OF SOFTWARE ENGINEERING

FINAL YEAR PROJECT REPORT

BACHELOR OF SOFTWARE ENGINEERING

HAMZA MUZAFFAR (51205)

ALI HARIS KHAN (50753)

RAJA SHAZIL (51747)

MUHAMMAD UMAR (51955)

PROJECT:

Online Security Guard Services

Management System

SUPERVISOR:

FAHAD NAJEEB

MARCH 2024

ABSTRACT

In today's dynamic security landscape, efficient management of guard services is paramount for ensuring safety and security. This project presents an Online Guard Reservation and Management System developed using Python Flask, aimed at providing a comprehensive solution for clients to reserve guards, manage reservations, generate invoices, view guard schedules, and monitor guard attendance.

The system facilitates clients to reserve one or more guards through an intuitive web interface. Clients can easily browse available guard profiles, select suitable guards based on their preferences and requirements, and make reservations seamlessly. Furthermore, clients have the flexibility to cancel reservations as needed, with the system automatically handling the necessary updates and notifications.

Invoice generation is an essential feature of the system, allowing clients to receive detailed invoices for the services availed. The invoicing process is streamlined and conducted entirely online, eliminating manual errors and delays. Clients can access and download invoices conveniently from their dashboard, enhancing transparency and accountability in billing.

The system also offers a comprehensive view of guard schedules, enabling clients to plan and coordinate security arrangements effectively. Guard schedules are dynamically updated based on reservations, cancellations, and other relevant factors, ensuring accuracy and reliability.

Real-time location sharing is facilitated through a dedicated mobile application used by guards. Guards can share their live location with supervisors, providing real-time insights into their whereabouts and activities. This feature enhances supervision capabilities, enabling supervisors to respond promptly to emergencies and ensure optimal deployment of resources.

Additionally, supervisors can mark guard attendance online, streamlining the attendance tracking process. By leveraging the power of web-based technologies, supervisors can efficiently monitor guard attendance, identify patterns, and address any discrepancies promptly.

Overall, the Online Guard Reservation and Management System developed using Python Flask offer a robust and user-friendly platform for managing guard services efficiently. By integrating essential features such as reservation management, invoicing, schedule viewing, live location sharing, and attendance tracking, the system enhances operational efficiency, transparency, and security in guard management processes.

This manuscript has been reviewed and approved for submission and presentation as fulfillment of Bachelor - of Software Engineering/ Computer Science requirements.

Supervisor: Fahad Najeeb

Date: 11-03-2024

Project Coordinator: Dr. Aarij Mahmood

Date: 11-03-2024

DECLARATION

I hereby declare that the work presented in this report has been conducted by myself to fulfill the requirements of the BS (Software Engineering) degree. I affirm that no portion of this work has been submitted in support of any application for any other degree or qualification, either at this university or any other institute of learning.

Furthermore, I declare that should any infringement of the provisions of the Act occur, whether knowingly or unknowingly, the university shall not be held liable in any manner. I undertake to indemnify and hold the university harmless against all claims and actions arising from such infringements.

© HAMZA MUZAFFAR [51205]

© ALI HARIS KHAN [50753]

© RAJA SHAZIL [51747]

© MUHAMMAD UMAR [51955]

ACKNOWLEDGEMENT

First and foremost, we extend our gratitude to Almighty Allah, whose blessings have endowed us with the capability to think, work, and fulfill our assigned tasks. We are indebted to our **supervisor, Mr. Fahad Najeeb**, for his invaluable guidance and support throughout the project. Additionally, we acknowledge the contributions of our instructor, **Mr. Aarij Mahmood Hussaan**, whose insights and direction have been instrumental in the successful completion of our work.

We are also thankful to all our teachers, departmental staff, and university personnel for their unwavering assistance and guidance. Special thanks to our families and friends for their constant support and encouragement throughout this journey. Lastly, we express our appreciation to our colleagues at IQRA University for their suggestions and words of encouragement, which have played a significant role in bringing this project to fruition.

LIST OF ACRONYMS

1. GUI - Graphical User Interface
2. API - Application Programming Interface
3. CRUD - Create, Read, Update, Delete
4. SQL - Structured Query Language
5. HTTPS - Hypertext Transfer Protocol Secure
6. AJAX - Asynchronous JavaScript and XML
7. JSON - JavaScript Object Notation
8. CSS - Cascading Style Sheets
9. HTML - Hypertext Markup Language
10. SSL - Secure Sockets Layer
11. UI - User Interface
12. UX - User Experience
13. CDN - Content Delivery Network
14. HTTPS - Hypertext Transfer Protocol Secure
15. URL - Uniform Resource Locator
16. API - Application Programming Interface
17. ORM – Object Relational Mapping

TABLE OF CONTENTS

CHAPTER – 1	1
1.0 Introduction.....	12
1.1 Problem Statement.....	13
1.2 Motivation.....	13
1.3 Objective.....	14
1.4 Challenges.....	15
1.5 Structure of Report	16
1.5.1 Chapter 2: Technology Background	16
1.5.2 Chapter 3: Requirements & Methodology	16
1.5.3 Chapter 4: Project Plan & Initial Design.....	16
1.5.4 Chapter 5: Project Design & Development.....	16
1.5.5 Chapter 6: Testing.....	17
1.5.6 Chapter 7: Conclusion.....	17
CHAPTER – 2.....	18
2. Technology Background.....	18
2.1 Background of the technology	18
2.1.1 Android Studio.....	18
2.1.2 Python	18
2.3 Literature Review	20
CHAPTER – 3.....	22
3.1 Introduction.....	22
3.2 Project Plan:.....	22
3.3 Functional Requirements:	23
3.4 Non-Functional Requirements	24
3.5 Summary.....	25
CHAPTER – 4.....	26
4.1 Introduction.....	26
4.2 Entity Relationship Diagram.....	27
4.3 Use Cases.....	28
4.5 Summary.....	43

CHAPTER – 5.....	44
5.1 Introduction.....	44
5.2 Database Queries:	45
5.3 Screenshots:	54
5.4 Summary.....	62
CHAPTER – 6.....	63
6.1 Introduction.....	63
6.2 Test Cases	64
6.3 Summary.....	72
CHAPTER – 7.....	73
7.1 Introduction.....	73
7.2 System Limitations and Challenges:.....	73
7.3 Future Work.....	74
7.4 Conclusion:	73
REFERENCES	75
APPENDIX	76

LIST OF TABLES

Table 1: Sign Up	29
Table 2: Login.....	30
Table 3: Apply For Job	31
Table 4: Guard Reservation.....	32
Table 5: Show Guard Attendance	33
Table 6: Show Guard Schedule	34
Table 7: Show Live Guard Location	35
Table 8: Supervisor login.....	37
Table 9: Reservation Request	38
Table 10: Mark Attendance	39
Table 11: Incident	40
Table 12: Live Guard Location.....	41
Table 13: Payroll.....	42
Table 14: Test case 1.....	53
Table 15: Test case 2.....	54
Table 16: Test case 3.....	54
Table 17: Test case 4.....	55
Table 18: Test case 5.....	55
Table 19: Test case 6.....	56
Table 20: Test case 7.....	56
Table 21: Test case 8.....	57
Table 22: Test case 9.....	57

LIST OF FIGURES

Figure 1: Project Plan.....	22
Figure 2: Gantt Chart	23
Figure 3: Entity Relationship Diagram	27
Figure 4: User Dashboard	28
Figure 5: Sign Up Use case	29
Figure 6: Login Use case	30
Figure 7: Apply For Job Use case.....	31
Figure 8: Guard Reservation Use case	32
Figure 9: Show Guard Attendance Use case.....	33
Figure 10: Show Guard Schedule Use case	34
Figure 11: Show Live Guard Location Use case	35
Figure 12: Supervisor dashboard	36
Figure 13: Supervisor login Use case	37
Figure 14: Reservation Request Use case.....	38
Figure 15: Mark Attendance Use case	39
Figure 16: Incident Use case.....	40
Figure 17: Live Guard Location Use case	41
Figure 18: Payroll Use case	42
Figure 19: Splash Screen	54
Figure 20: Login Screen	55
Figure 21: Logged in Screen.....	56
Figure 22: Web login.....	57
Figure 23: Client dashboard.....	57
Figure 24: Guard Schedule	58
Figure 25: Cancel Reservation.....	58
Figure 26: Guard Live Tracking.....	59
Figure 27: Apply for job	59
Figure 28: Attendance Marking	60
Figure 29: Search Attendance	60
Figure 30: Supervisor Dashboard.....	61
Figure 31: Insert Incident Report	61
Figure 32: Reservation Report	62

CHAPTER – 1

1.0 Introduction

Security guards serve as the frontline defenders of safety and security, playing a crucial role in protecting lives, property, and assets. In the contemporary landscape, safeguarding assets and ensuring the safety of individuals stands as a critical priority. Yet, the traditional methods of sourcing and managing security personnel often prove arduous and time-intensive. Recognizing this challenge, our SecureGuard Hub emerges as a transformative solution, reshaping the accessibility and administration of security services. Our platform empowers users to seamlessly book security guards anytime, anywhere, through a user-friendly mobile or web application. Say goodbye to the complexities of endless searches and cumbersome phone calls – whether for short-term requirements or prolonged security needs, our system facilitates guard reservations effortlessly. Central to our system is real-time tracking functionality, granting users the ability to monitor the live location of deployed guards. This transparency not only instills peace of mind but also ensures the seamless orchestration of security operations.

Furthermore, our comprehensive suite of features encompasses various facets of security management, ranging from job applications and guard registration to attendance tracking, scheduling, and reservation. Additionally, the system facilitates corrective action requests, incident reporting, and invoicing, simplifying administrative processes and fostering a proactive approach to security oversight. Harnessing the power of GPS tracking technology, our system provides unparalleled visibility into guard movements, enabling swift responses to emergent situations. Moreover, dedicated portals for clients and guards foster enhanced communication and collaboration, enriching the overall security service experience.

In essence, our SecureGuard Hub empowers individuals and businesses to access reliable security services with unparalleled ease. By embracing cutting-edge technology, we endeavor to redefine the standards of security service management, ushering in a new era of efficiency and efficacy in safeguarding assets and ensuring the safety of all stakeholders.

CHAPTER – 1

1.1 Problem Statement

In today's dynamic and often unpredictable world, the need for robust security measures has become increasingly paramount. However, many individuals and organizations struggle to effectively address security concerns due to a lack of dedicated personnel and resources. This gap in security infrastructure leaves properties, assets, and individuals vulnerable to various threats, including theft, vandalism, and unauthorized access. Moreover, traditional security methods may prove inadequate in addressing the evolving nature of security risks, such as cyber threats and sophisticated criminal activities. Consequently, there is a pressing need for comprehensive security solutions that prioritize proactive deterrence, rapid response, and effective risk mitigation. Addressing these challenges requires a reevaluation of existing security strategies and the implementation of innovative approaches that leverage technology, expertise, and collaboration to safeguard lives and property effectively.

1.2 Motivation

According to a survey, Our project endeavors to address the challenges individuals encounter when hiring security guards by providing accessible, swift, and secure security guard services tailored to the needs of both individuals and businesses.

We understand that ensuring safety and security is paramount to the well-being of individuals, communities, and assets. Everyone deserves to feel secure in their environment, whether at home, work, or in public spaces. By implementing effective security measures, we not only protect lives and property but also instill confidence and peace of mind among residents, employees, and visitors.

Furthermore, addressing security challenges is vital for fostering stability and prosperity within businesses and society as a whole. It promotes economic growth and social cohesion by creating safe environments conducive to living, working, and thriving.

As security threats evolve, we remain committed to constant vigilance and innovation. Our goal is to develop and adopt advanced security solutions capable of adapting to emerging risks and vulnerabilities, ensuring our communities remain resilient in the face of adversity.

One of the key strengths of our system/application lies in its user-friendly interface, facilitating a seamless experience without any hurdles. Users can quickly reserve guards with ease and efficiency, enabling swift access to security services when needed.

CHAPTER – 1

1.3 Objective

Our project aims to revolutionize the accessibility and effectiveness of security guard services by providing a comprehensive SecureGuard Hub. The primary focus is on facilitating seamless access to security services for individuals and businesses, ensuring their safety and peace of mind. Our objective is to streamline the process of hiring security guards and enhance the overall security management experience.

The concept revolves around developing a robust online platform equipped with advanced features to manage security guard services efficiently. Through our system, users will be able to easily browse, select, and book security guards tailored to their specific needs. Our objective is to simplify the reservation process and provide real-time monitoring capabilities, ensuring optimal security outcomes for our clients.

- i. Facilitating quick and efficient reservation of security guards through our online platform.
- ii. Providing real-time monitoring functionalities to track the live location of deployed guards.
- iii. Offering comprehensive management tools, including guard registration, scheduling, and incident reporting.
- iv. Enhancing communication and collaboration between clients and security guards through dedicated portals within our system.

CHAPTER – 1

Research Objectives:

Through our research efforts, we aim to raise awareness about our SecureGuard Hub and highlight its significance in modern security management practices. We strive to understand the evolving needs and preferences of users and adapt our system accordingly to ensure optimal usability and effectiveness.

Academic Objectives:

Our academic objectives involve exploring the impact of technology on security management and evaluating the effectiveness of our system in enhancing security outcomes. [4] We aim to contribute to the academic discourse surrounding security management practices and provide insights into the potential benefits of adopting advanced technological solutions.

Management Objectives:

The management's primary objective is to oversee the development and implementation of our SecureGuard Hub. This involves organizing meetings for discussions, monitoring project progress, and ensuring timely project completion to meet the needs of our clients effectively.

1.4 Challenges

Embarking on a project without precedence in the form of existing websites or software presented us with numerous challenges, particularly in the realms of backend and database development. Also a significant portion of the population lacks familiarity with smartphone usage, posing a challenge in adopting a mobile-based security management system.

- i. In areas with limited internet connectivity, accessing and utilizing a mobile-based management system becomes challenging, hindering effective security operations.
- ii. The cost of smartphones or tablets may be prohibitive for some individuals or businesses, limiting their ability to leverage mobile-based security solutions.
- iii. The absence of similar websites or software posed a unique challenge as it was our first-time experience dealing with such a project. Without established guidelines or frameworks to reference, we had to navigate uncharted territory.

CHAPTER – 1

1.5 Structure of Report

This marks the completion of chapter one of the SecureGuard Hub project. This chapter highlighted the overall concept of the SecureGuard Hub application. The introduction section provides a detailed review regarding major details of idea hunting. The problem statement specifies, what kind of issue we are tackling with this application and how it will be beneficial for the society. The motivation heading provides a clear definition of what motivated us to come up with the idea of SecureGuard Hub application.

As we go further into the content of chapter one the major descriptions regarding the objective of the project can be seen. This is where you will find vital information such as research objectives, academic objectives as well as management objectives. Furthermore, we have provided content regarding the challenges that can occur with the progression of the application as well as in the usage of the application. Lastly, the remaining structure of the project report is given as follows:

1.5.1 Chapter 2: Technology Background

This chapter will consist of our well researched literature review regarding all the related prior work of our project subject and technologies.

1.5.2 Chapter 3: Requirements & Methodology

This Chapter will discuss basic models of the system, in addition to that the chapter will also host functional and nonfunctional requirements of our project.

1.5.3 Chapter 4: Project Plan & Initial Design

This chapter will consist of all the detailed designs of the project that will help the developer in understanding the project implementation and creating an easy route in development of the system.

1.5.4 Chapter 5: Project Design & Development

This is the most significant chapter since it details the actual design and implementation of the concept. i.e., the phases of design and development.

CHAPTER – 1

1.5.5 Chapter 6: Testing

We will construct test cases in this chapter.

1.5.5.1 Perform front end (design testing), which may include user control testing, spelling checks, and alignment, among other things.

1.5.5.2 Carry out backend testing (source code)

1.5.5.3 Use a tool to conduct testing and incorporate the results in report.

1.5.6 Chapter 7: Conclusion

In this last chapter, we will conclude our work, share results including facts and figures, tables, and graphs to show your findings.

- i Discuss limitations and challenges.
- ii Discuss the work that will be done in the future.

CHAPTER – 2

2. Technology Background

In today's digital age, the effective management of guard services demands streamlined and technologically advanced solutions. The Online Guard Reservation and Management System leverages a suite of modern technologies to provide clients, guards, and supervisors with a comprehensive platform for efficiently managing guard services. This introduction outlines the key technologies that form the foundation of the project, enabling seamless interactions and enhancing the overall security management process.

2.1 Background of the technology:

The web application overview begins with a homepage containing company-related details. Without logging in, users can view About, Contact, Blog, and Security Officers sections. Additionally, if there's a job requirement, users can apply via the "Apply for Job" option. After logging in or signing up, users can easily submit guard reservation requests.

2.1.1 Android Studio

Android Studio stands as the cornerstone of the mobile application's development, offering a powerful integrated development environment (IDE) specifically tailored for Android app creation. Leveraging Android Studio's rich set of tools and libraries, developers can design, code, and debug applications efficiently, ensuring optimal performance across a wide range of Android devices.

Java serves as the primary programming language for building the Android application, providing developers with robust frameworks and APIs for implementing various functionalities. Whether it's handling user authentication, retrieving guard location data, or interfacing with external services, Java or Kotlin empowers developers to create feature-rich and responsive mobile applications.

Supported platforms

- i. Mobile platforms
- ii. Internet platform

2.1.2 Python Flask

Python Flask serves as the backbone of the system, providing a lightweight and flexible framework for web application development. Known for its simplicity and extensibility, Flask facilitates rapid development and easy integration with other technologies. Through Flask, the system can handle routing, request handling, and template rendering, ensuring a smooth user experience.

CHAPTER – 2

Identifying Features:

- i. Lightweight and flexible framework for web application development
- ii. Simplicity and ease of use, ideal for rapid development
- iii. Seamless integration with other technologies
- iv. Capable of handling routing to direct incoming requests to the appropriate endpoints
- v. Efficient request handling for processing user interactions and data manipulation
- vi. Built-in support for template rendering, enabling the creation of dynamic and interactive user interfaces
- vii. Facilitates a smooth user experience through its robust features and functionalities

2.1.3 HTML, CSS, JavaScript

HTML, CSS, and JavaScript form the frontend stack of the system, enabling the creation of intuitive and interactive user interfaces. HTML structures the content of web pages, while CSS styles and layouts enhance their visual appeal. JavaScript adds dynamic behavior to the web pages, enabling features such as real-time updates and interactive elements.

2.1.4 SQLAlchemy

SQLAlchemy, an Object-Relational Mapping (ORM) library, simplifies database operations by allowing developers to work with database entities as Python objects. This abstraction layer over the database streamlines tasks such as querying, inserting, updating, and deleting data, ensuring efficient data management within the system.

CHAPTER – 2

2.3 Literature Review:

In recent years, there has been a noticeable surge in the integration of technology-driven solutions within the security industry, leading to the advent of various online guard reservation and management systems. These systems have garnered attention due to their potential to revolutionize traditional security operations by introducing efficiency, transparency, and accountability into the management processes. Research conducted by Lai and To (2019) underscores the significance of technological advancements, particularly online platforms and mobile applications, in reshaping security management practices. This shift towards technology adoption aligns with broader trends across industries, emphasizing the importance of leveraging digital solutions to optimize operational workflows and enhance service delivery.

One of the foundational concepts underpinning these systems is the notion of online reservation platforms. These platforms, widely utilized in industries such as hospitality, transportation, and healthcare, enable users to remotely book services or resources with ease and convenience. According to Sharma and Singh (2020), online reservation systems offer several benefits, including improved resource utilization and enhanced customer satisfaction. Such systems provide users with the flexibility to schedule and manage their bookings efficiently, contributing to a smoother and more streamlined user experience.

Guard management systems represent another significant component of this technological landscape. These systems focus on optimizing the deployment, monitoring, and coordination of security personnel, thereby enhancing overall security effectiveness. Commercial solutions like TrackTik and Silvertrac Software offer comprehensive platforms for managing guard activities, ranging from scheduling and incident reporting to client communication. Research conducted by Chen et al. (2018) highlights the positive impact of guard management systems on operational efficiency and response times, indicating their value in modern security operations.

Furthermore, the advent of mobile applications has introduced new possibilities for guard supervision and communication. Mobile applications enable real-time location tracking, communication, and incident reporting among security personnel, enhancing situational awareness and facilitating timely decision-making by supervisors. Studies by O'Connor et al. (2017) underscore the role of mobile applications in modern security operations, emphasizing their potential to streamline communication channels and improve operational efficiency.

CHAPTER – 2

However, despite the potential benefits offered by these systems, several challenges persist. Security concerns, data privacy issues, and resistance to technology adoption are commonly cited challenges in the literature (Oliveira et al., 2019). Additionally, ensuring interoperability with existing systems and addressing user training needs are critical considerations for successful implementation. Nonetheless, integrated solutions that combine web-based reservation systems with mobile applications offer enhanced functionality and accessibility, paving the way for more efficient and effective security management practices (Zhu et al., 2021).

CHAPTER – 3

3.1 Introduction:

In this chapter we will discuss about how much work is done on the development of our project according to the project plan. This chapter will cover the in-detail process and objective of the project. As our app is built on windows that are being reused within the system itself the developers had to take a systematic approach for the webapp to work smoothly. Our project plan is strategically planned with the Gantt chart and other organizational tools. Each activity has specific time period allotted according to the complexity of the task, which is why the days in work may vary.

We will also discuss in detail about the Functional, Non-Functional and Hardware requirements of our project. The functional requirement are taken in to full consideration as they are the necessary part in order to get the basic requirement by the project such as, location detection and reservation objective, while on the other hand, the non-functional requirement such as, settings and feedback are also thoroughly planned.

3.2 Project Plan:

The Division of Tasks Among Team-Members					
Phase		Raja Shazil	Ali Haris Khan	Hamza Muzaffar	Muhammad Umar
Requirement Gathering		Software & Hardware Requirement			
Design		User Interface	Use Case diagram	Sequence Diagram	Class Diagram
Creation	Guard Scheduling	Documentation, Front-End, Back-End, Database			
	Time & Attendance Management		Documentation, Front-End, Back-End, Database		
	GPS Tracking			Documentation, Front-End, Back-End, Database	
	Guard Profile				Documentation, Front-End, Back-End, Database
	Communication System	Documentation, Front-End, Back-End, Database			
	Billing and invoicing			Documentation, Front-End, Back-End, Database	
Testing		Unit , User & Component Testing			

Figure 1: Project Plan

CHAPTER – 3

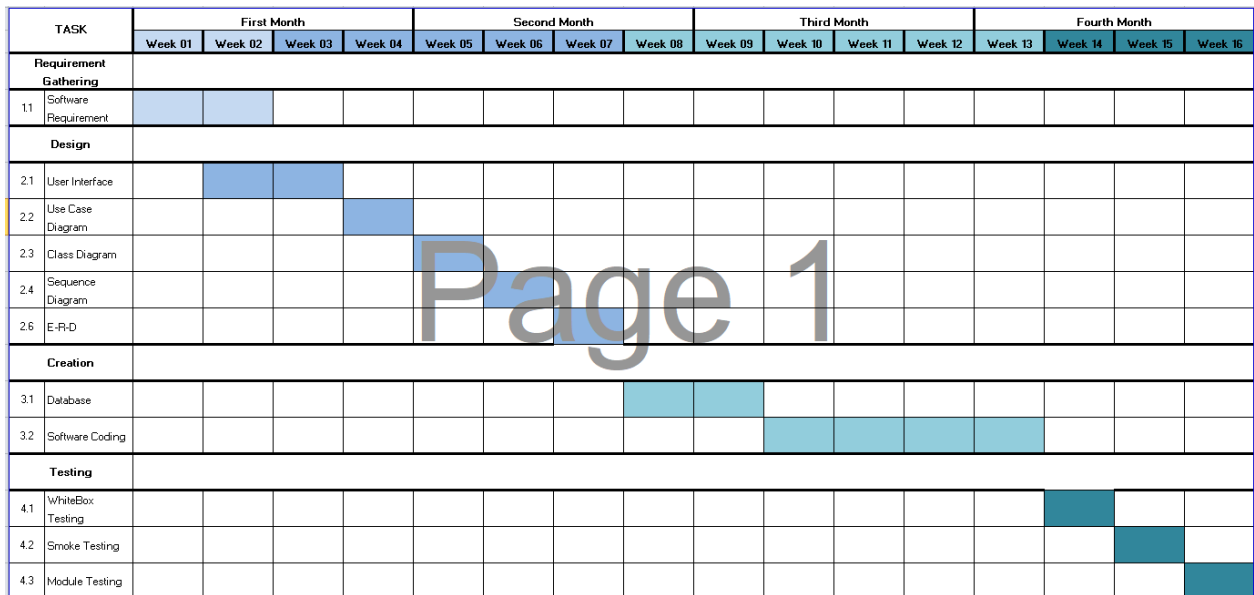


Figure 2: Gantt Chart

3.3 Functional Requirements:

- 1. Apply for job:** It involves the process of recruiting and onboarding security guards into the system. It includes activities such as job postings, candidate screening, interviews, background checks, and documentation. This requirement ensures that the system has a pool of qualified and trained guards available for deployment when needed.
- 2. Guard Reservation:** That enables clients or authorized users to reserve security guards for specific assignments or time periods. It involves the process of requesting and scheduling guard services based on the client's requirements. This feature ensures that clients can book security guards in advance for their desired dates, locations, and durations.
- 3. Guard Scheduling:** Allow the scheduling of security guards for different shifts and assignments based on their availability, skills, and location.
- 4. Time and Attendance Management:** Track the attendance of security guards, including their clock-in and clock-out times, and their hours worked, to ensure that they are meeting their work requirements and getting paid accurately.
- 5. Incident Reporting:** Allow security guards to report incidents, such as theft, vandalism, or accidents, and to enable the management team to review and respond to these reports.
- 6. GPS Tracking:** Track the location of security guards using GPS technology, to ensure that they are following their assigned patrol routes and to enable quick response to emergencies.

CHAPTER – 3

7. **Guard Profile Management:** Manage and maintain the profiles of security guards, including their personal and professional information, employment history, and training records.
8. **Communication System:** Provide a communication system, such as two-way radios or messaging apps, to enable security guards and the management team to communicate quickly and effectively.
9. **Billing and Invoicing:** Generate invoices and billing statements for clients based on the services provided by the security guards.
10. **Client Portal:** Provide a secure portal for clients to access their account information, view reports and analytics, and communicate with the management team.

3.4 Non-Functional Requirements:

1. **Availability:** The system should be available to users 24/7 without any downtime, and should have a failover mechanism in place in case of a system failure.
2. **Performance:** The system should be able to handle a large number of users and requests without slowing down or crashing.
3. **Scalability:** The system should be scalable and able to handle an increasing number of users, requests, and data without affecting performance.
4. **Usability:** The system should be easy to use and navigate, and should provide clear and concise instructions for users.
5. **Reliability:** The system should be reliable and able to withstand attacks, failures, or other unexpected events.
6. **Maintainability:** The system should be easy to maintain, update, and upgrade, and should have a modular architecture that allows for easy customization and integration.
7. **Security:** The system should provide a high level of security to protect against unauthorized access, data breaches, or other security threats.
8. **Accessibility:** The system should be accessible to users with disabilities, such as those with visual impairments, hearing impairments, or mobility impairments.
9. **Privacy:** The system should respect user privacy and protect their personal information according to applicable privacy laws and regulations.

CHAPTER – 3

3.5 Summary:

In this chapter, a detailed Project Plan, Functional, Non-Functional requirements and other planning mechanisms are discussed in detail that will be required in our project. We have also mentioned an introduction regarding our Web-application how we can perform our task so we make a milestone chart in this first we describe our task week wise in summary activity and then we make a Gantt chart according to summary activity. In Gantt chart we were describing task name or duration for implementation of our “SecureGuard Hub” After Gantt chart we describe Non-Functional requirements of our Web-application the requirements are system settings and feedback

CHAPTER – 4

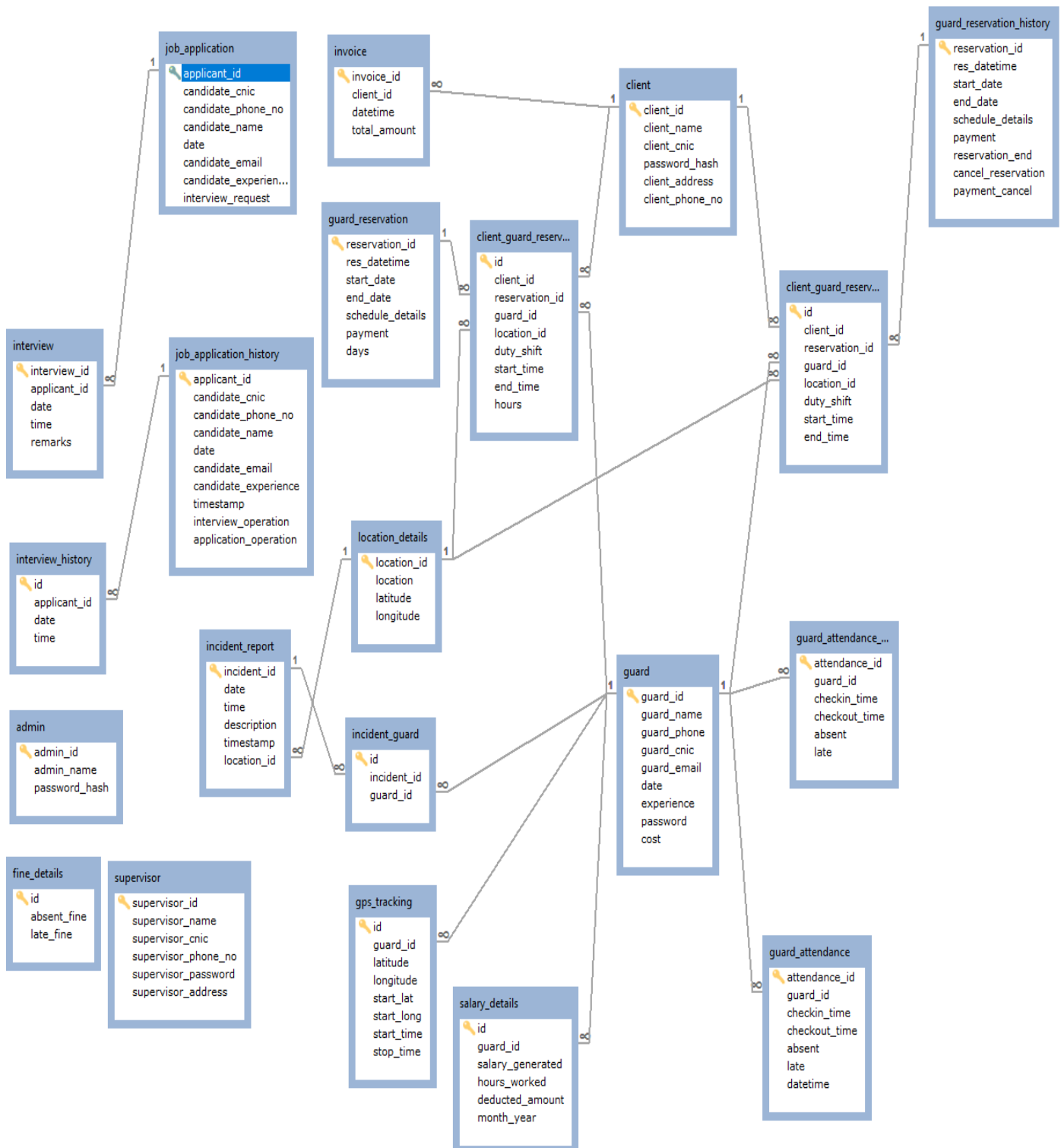
4.1 Introduction:

In this chapter, we are going to discuss about the design and specification of our project, in which we elaborate our project deeply with the help of diagrams like we gather all the information related to our Web-application then set the framework to show the flow of the Web-application so that the Web-application flow will be easily understand. We have used different diagrams for the complete flow of our application to make it understandable to user. In this phase, we also discussing the entity relationship diagram (ERD'S), UML in detailed as according to our Web-application. These diagrams show the system work flow and specification of our application to make it user friendly. It also shows how every screen work flow is working with the help of diagram. After all information has been gathered and design has been created so now, the development has started in order to make sure that it is able to be used by user.

The purpose for making the entity diagram to guide the direction of our system that how we perform each and every thing and also show the flow of our Web-application specifically like in implementation. In detail this will provide a clear understanding of the overall coding of the system for the people who are on user bases. Each diagram is detail with all functional input and output of the system, making sure that the system runs smoothly.

CHAPTER – 4

4.2 Entity Relationship Diagram



CHAPTER – 4

Figure 3: Entity Diagram Relationship

4.3 Use Cases

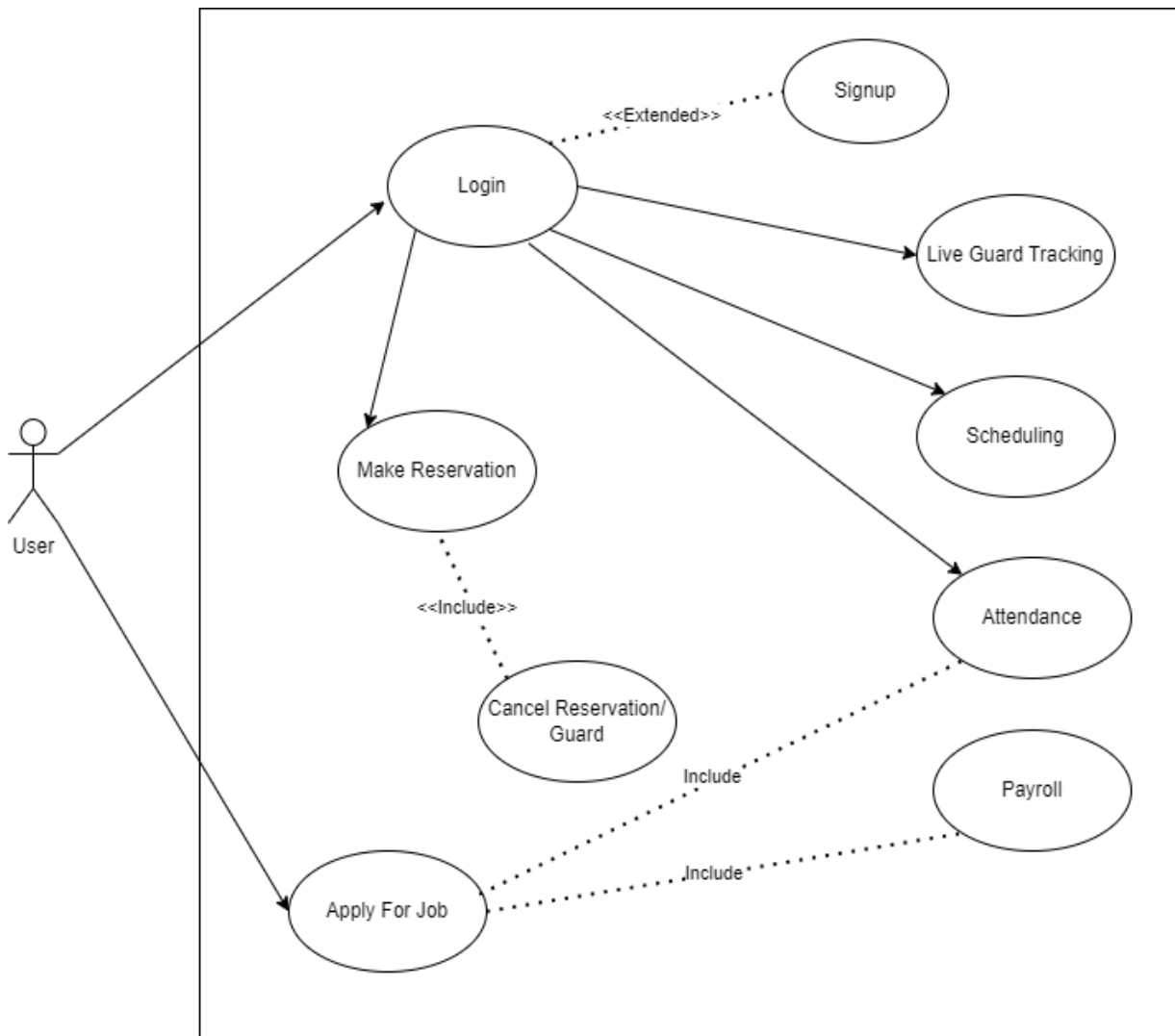


Figure 4: User Dashboard

CHAPTER – 4

1. Signup



Figure 5: Sign Up Use case

Use Case Name:	Sign up	
ID:	01	
Priority:	High	
Actors Involved:	User	
Brief Description:	If the actor is not registered then they will fill all the fields of the signup form, after that they can see and avail all the features in the application.	
Pre- Condition:		
Post- Condition:		
Normal Flow of Events:	Actor Actions: i Enter name, CNIC, address, password and mobile number, type. ii Clicks the 'signup' Button.	System Response: iii System will check and validate the username that it is exist in the database or not. iv If username already exist in the database, then it will show error on the actor's screen that 'this username is already exist'. If actor put unique username, then the System will create the account. v System displays the actor's main page on successful login.

Table 1: Sign Up

CHAPTER – 4

2. Login



Figure 6: Login Use case

Use Case Name:	Login	
ID:	02	
Priority:	High	
Actors Involved:	User	
Brief Description:	Actors enter the Email and password to login to the application.	
Pre-Condition:	Use Case ID: 01	
Post-Condition:	Enter into the application.	
Normal Flow of Events:	Actor Actions: i Enter Email. ii Enter Password iii Clicks the login Button.	System Response: i System displays the actor's Client Dashboard page on successful login. ii System displays error message on invalid login.

Table 2: Login

CHAPTER – 4

3. Apply For Job

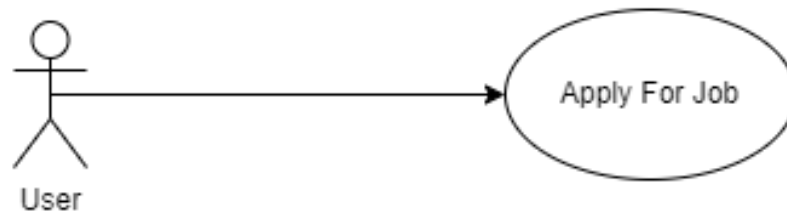


Figure 07: Apply For Job

Use Case Name:	Apply for job	
ID:	03	
Priority:	High	
Actors Involved:	User	
Brief Description:	Actors Apply for Job	
Pre-Condition:	Enter into the Web-application.	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: i Click on Actor ii Apply For Job	System Response: i Open Form of job application

Table 3: Apply For Job

CHAPTER – 4

4. Reservation

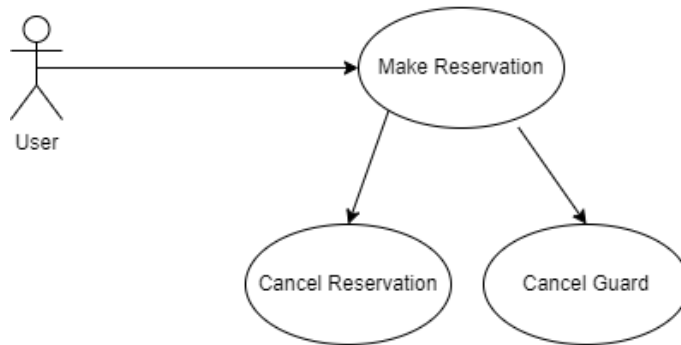


Figure 08: Make Reservation Use case

Use Case Name:	Make Reservation	
ID:	04	
Priority:	High	
Actors Involved:	User	
Brief Description:	Actors Make a Guard Reservation for security.	
Pre-Condition:	Use Case ID: 02	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: <ul style="list-style-type: none"> i. Make Reservation ii. Cancel Reservation iii. Cancel Single Guard 	System Response: <ul style="list-style-type: none"> iv Show Available Guards v Show Invoices vi Maintain Data

Table 4: Reservation

CHAPTER – 4

5. Show Guard Attendance

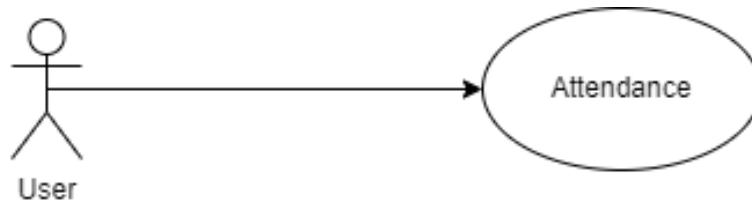


Figure 09: Show Guard Attendance Use case

Use Case Name:	Show Guard Attendance	
ID:	05	
Priority:	High	
Actors Involved:	User	
Brief Description:	Actors can see Guard Attendance.	
Pre-Condition:	Use Case ID: 02	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: i Login ii See Guard Attendance	System Response: i Show Guard Attendance on the bases of their Time In and Out.

Table 5: Show Guard Attendance

CHAPTER – 4

6. Scheduling



Figure 10: Show Guard Schedules Use case

Use Case Name:	Show Guard Schedules	
ID:	06	
Priority:	High	
Actors Involved:	User	
Brief Description:	Actors can see the Guard Schedules.	
Pre-Condition:	Use Case ID: 03	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: i. Login ii. Open Scheduling	System Response: i. Show Guard Schedules

Table 6: Show Guard Schedules

CHAPTER – 4

7.Live Guard Tracking

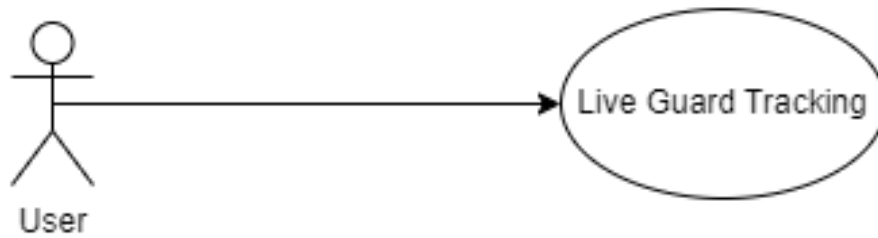


Figure 11: Show Live Guard Location Use case

Use Case Name:	Show Live Guard Location	
ID:	06	
Priority:	High	
Actors Involved:	User	
Brief Description:	Actors can see the Live Guard Location.	
Pre-Condition:	Use Case ID: 03	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: iii. Login iv. Open Live guard tracking	System Response: i. Show Guard Location.

Table 7: Show Live Guard Location

CHAPTER – 4



Figure 12: Supervisor Dashboard

CHAPTER – 4

1. Login

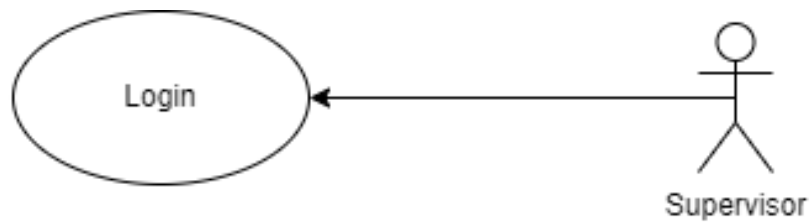


Figure 13: Login Use case

Use Case Name:	Login	
ID:	01	
Priority:	High	
Actors Involved:	Supervisor	
Brief Description:	Actors enter the Email and password to login to the Web-application.	
Pre-Condition:	Use Case ID: 01	
Post-Condition:	Enter into the web-application.	
Normal Flow of Events:	Actor Actions: i Enter Email. ii Enter Password iii Clicks the login Button.	System Response: iii System displays the actor's Supervisor Dashboard page on successful login. iv System displays error message on invalid login.

Table 8: Login

CHAPTER – 4

2. Reservation

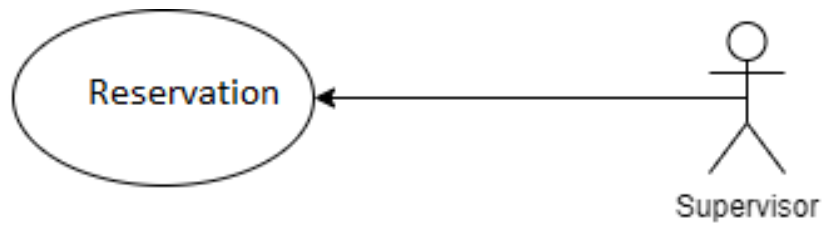


Figure 14: Reservation Request

Use Case Name:	Reservation Request	
ID:	02	
Priority:	High	
Actors Involved:	Supervisor	
Brief Description:	Actors can accept or reject the client reservation request	
Pre-Condition:	Use Case ID: 02	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: i Accept reservation. ii Reject reservation	System Response: v Show reservation receipt. vi Show invoice.

Table 9: Reservation

CHAPTER – 4

3. Mark Attendance

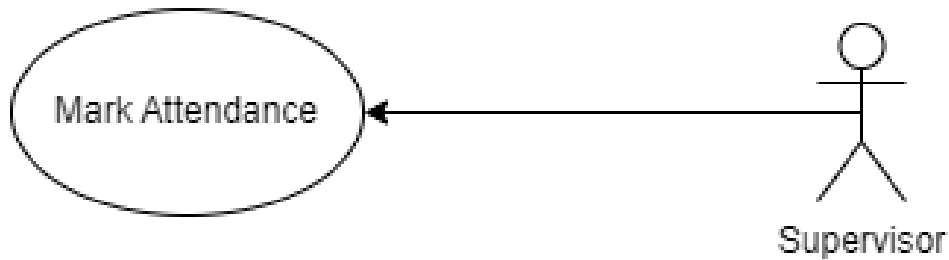


Figure 15: Mark Attendance

Use Case Name:	Mark Attendance	
ID:	03	
Priority:	High	
Actors Involved:	Supervisor	
Brief Description:	Actors can mark the Guard Attendance.	
Pre-Condition:	Use Case ID: 03	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: i Login ii Mark Guard attendance	System Response: i Show Guard In / out time. ii Show attendance record.

Table 10: Mark Attendance

CHAPTER – 4

4. Incident

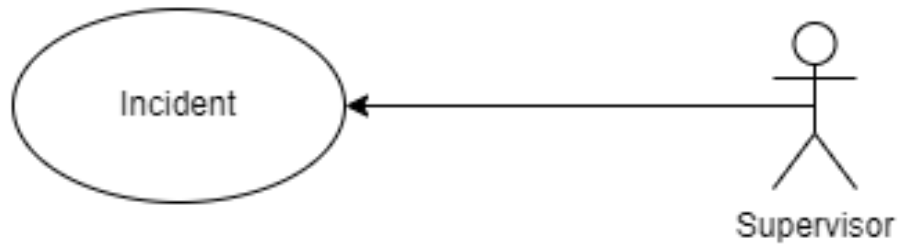


Figure 16: Incident

Use Case Name:	Insert Incident	
ID:	04	
Priority:	High	
Actors Involved:	Supervisor	
Brief Description:	Actors insert the incident reports.	
Pre-Condition:	Use Case ID: 04	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: v. Login vi. Open Incident	System Response: i. Show Incident report.

Table 11: Incident

CHAPTER – 4

5. Live Guard Tracking

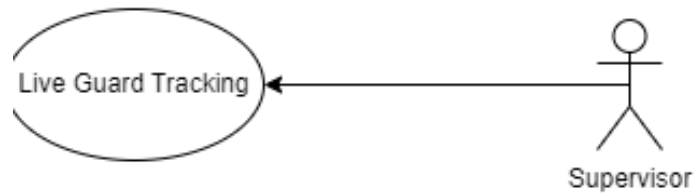


Figure 17: Live Guard Tracking

Use Case Name:	Show Live Guard Location	
ID:	05	
Priority:	High	
Actors Involved:	Supervisor	
Brief Description:	Actors can see the Live Guard Location.	
Pre-Condition:	Use Case ID: 05	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: vii. Login viii. Open Live guard tracking	System Response: i. Show Guard Location.

Table 12: Live Guard Location

CHAPTER – 4

6. Payroll

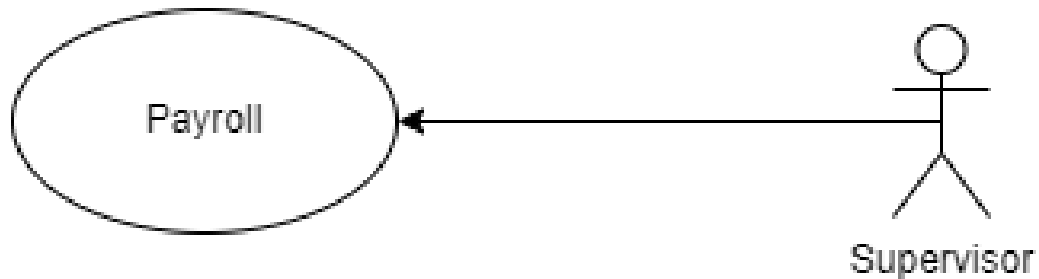


Figure 18: Pay roll

Use Case Name:	Show Payroll	
ID:	06	
Priority:	High	
Actors Involved:	Supervisor	
Brief Description:	Actors can see the salary slips	
Pre-Condition:	Use Case ID: 06	
Post-Condition:	Enter into the Web-application.	
Normal Flow of Events:	Actor Actions: ix. Login x. Open Payroll	System Response: i. Show Salary slip.

Table 13: Payroll

4.5 Summary:

In this chapter, we met with the entity relationship diagrams and uml diagram. Our Web-application covered the all-major modules, which were used to fulfill the requirements so the entity relationship diagram (erd's) elaborate this in detailed because when we gathered the information so it show the flow of our system specifically like in implementation we make sure that the development is start or not then after this testing process will be occurred so if there is any error occur so it must be solved at that time then we monitor our Web-application by time to time to make sure that everything is complete or not that why we make entity relationship diagram to focus on our mistake. In spite of everything, information has been accumulated and layout has been created so now, the implementation has been initiated according to the web-applications' requirement to make sure that it is beneficial for the user on a long run. This application has a highly responsive nature because of the fact that the major specification of this web-application is to connect with internet, GPS, scheduling, attendance of the guard.

CHAPTER – 5

5.1 Introduction:

In this chapter we are discussing aspects which are used in our project, which is generally used to evaluate a new design to enhance precision by system analysts and users and front-end and back-end design of our project. We are also discussing about the database queries which are used in fire-base and some external libraries and we are showing screen-shots of our web-application as it fulfills user requirements. We have briefly provided a few clarifications about the sort of functionalities available on the system, source code of validation and etc.

CHAPTER – 5

5.2 Database Queries:

/*Table structure for table `admin` */

```
CREATE TABLE `admin` (  
  `admin_id` int(11) NOT NULL,  
  `admin_name` varchar(50) DEFAULT NULL,  
  `password_hash` varchar(100) NOT NULL,  
  PRIMARY KEY (`admin_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `client` */

```
CREATE TABLE `client` (  
  `client_id` int(11) NOT NULL,  
  `client_name` varchar(50) DEFAULT NULL,  
  `client_cnic` varchar(50) DEFAULT NULL,  
  `password_hash` varchar(50) DEFAULT NULL,  
  `client_address` varchar(100) DEFAULT NULL,  
  `client_phone_no` varchar(20) DEFAULT NULL,  
  PRIMARY KEY (`client_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `client_guard_reservation` */

```
CREATE TABLE `client_guard_reservation` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `client_id` int(11) DEFAULT NULL,  
  `reservation_id` int(11) DEFAULT NULL,  
  `guard_id` int(11) DEFAULT NULL,  
  `location_id` int(11) DEFAULT NULL,  
  `duty_shift` varchar(50) DEFAULT NULL,  
  `start_time` time DEFAULT NULL,  
  `end_time` time DEFAULT NULL,  
  `hours` double DEFAULT NULL,
```

CHAPTER – 5

```
PRIMARY KEY (`id`),
KEY `guard_id` (`guard_id`),
KEY `location_id` (`location_id`),
KEY `reservation_id` (`reservation_id`),
KEY `client_id` (`client_id`),

CONSTRAINT `client_guard_reservation_ibfk_3` FOREIGN KEY (`guard_id`) REFERENCES `guard` (`guard_id`) ON
DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `client_guard_reservation_ibfk_4` FOREIGN KEY (`location_id`) REFERENCES `location_details`
(`location_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `client_guard_reservation_ibfk_5` FOREIGN KEY (`reservation_id`) REFERENCES `guard_reservation`
(`reservation_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `client_guard_reservation_ibfk_6` FOREIGN KEY (`client_id`) REFERENCES `client` (`client_id`) ON
DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

/*Table structure for table `client_guard_reservation_history` */

CREATE TABLE `client_guard_reservation_history` (
  `id` int(11) NOT NULL,
  `client_id` int(11) DEFAULT NULL,
  `reservation_id` int(11) DEFAULT NULL,
  `guard_id` int(11) DEFAULT NULL,
  `location_id` int(11) DEFAULT NULL,
  `duty_shift` varchar(50) DEFAULT NULL,
  `start_time` time DEFAULT NULL,
  `end_time` time DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `guard_id` (`guard_id`),
  KEY `location_id` (`location_id`),
  KEY `reservation_id` (`reservation_id`),
  KEY `client_id` (`client_id`),
  CONSTRAINT `client_guard_reservation_history_ibfk_1` FOREIGN KEY (`reservation_id`) REFERENCES
`guard_reservation_history` (`reservation_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `client_guard_reservation_history_ibfk_2` FOREIGN KEY (`client_id`) REFERENCES `client` (`client_id`)
ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `client_guard_reservation_history_ibfk_3` FOREIGN KEY (`location_id`) REFERENCES `location_details`
(`location_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
```

CHAPTER – 5

```
CONSTRAINT `client_guard_reservation_history_ibfk_4` FOREIGN KEY (`guard_id`) REFERENCES `guard` (`guard_id`)
ON DELETE NO ACTION ON UPDATE NO ACTION
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
/*Table structure for table `fine_details` */
```

```
CREATE TABLE `fine_details` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `absent_fine` double DEFAULT NULL,
  `late_fine` double DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
/*Table structure for table `gps_tracking` */
```

```
CREATE TABLE `gps_tracking` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `guard_id` int(11) DEFAULT NULL,
  `latitude` double DEFAULT NULL,
  `longitude` double DEFAULT NULL,
  `start_lat` double DEFAULT NULL,
  `start_long` double DEFAULT NULL,
  `start_time` datetime DEFAULT NULL,
  `stop_time` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `guard_id` (`guard_id`),
  CONSTRAINT `gps_tracking_ibfk_1` FOREIGN KEY (`guard_id`) REFERENCES `guard` (`guard_id`) ON DELETE
  CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
/*Table structure for table `guard` */
```

```
CREATE TABLE `guard` (
  `guard_id` int(11) NOT NULL AUTO_INCREMENT,
  `guard_name` varchar(50) DEFAULT NULL,
```

CHAPTER – 5

```
`guard_phone` varchar(20) DEFAULT NULL,  
`guard_cnic` varchar(50) DEFAULT NULL,  
`guard_email` varchar(50) NOT NULL,  
`date` date NOT NULL,  
`experience` varchar(50) NOT NULL,  
`password` varchar(100) DEFAULT NULL,  
`cost` double DEFAULT NULL,  
PRIMARY KEY (`guard_id`)  
) ENGINE=InnoDB AUTO_INCREMENT=8012 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `guard_attendance` */

```
CREATE TABLE `guard_attendance` (  
  `attendance_id` int(11) NOT NULL AUTO_INCREMENT,  
  `guard_id` int(11) DEFAULT NULL,  
  `checkin_time` datetime DEFAULT NULL,  
  `checkout_time` datetime DEFAULT NULL,  
  `absent` tinyint(1) DEFAULT 0,  
  `late` tinyint(1) DEFAULT 0,  
  `datetime` datetime DEFAULT NULL,  
  PRIMARY KEY (`attendance_id`),  
  KEY `guard_id` (`guard_id`),  
  CONSTRAINT `guard_attendance_ibfk_1` FOREIGN KEY (`guard_id`) REFERENCES `guard` (`guard_id`) ON DELETE  
  CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `guard_attendance_history` */

```
CREATE TABLE `guard_attendance_history` (  
  `attendance_id` int(11) NOT NULL AUTO_INCREMENT,  
  `guard_id` int(11) DEFAULT NULL,  
  `checkin_time` datetime DEFAULT NULL,  
  `checkout_time` datetime DEFAULT NULL,  
  `absent` tinyint(1) DEFAULT 0,  
  `late` tinyint(1) DEFAULT 0,
```


CHAPTER – 5

```
PRIMARY KEY (`attendance_id`),  
KEY `guard_id` (`guard_id`),  
CONSTRAINT `guard_attendance_history_ibfk_1` FOREIGN KEY (`guard_id`) REFERENCES `guard` (`guard_id`) ON  
DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `guard_reservation` */

```
CREATE TABLE `guard_reservation` (  
  `reservation_id` int(11) NOT NULL AUTO_INCREMENT,  
  `res_datetime` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),  
  `start_date` date DEFAULT NULL,  
  `end_date` date DEFAULT NULL,  
  `schedule_details` text DEFAULT NULL,  
  `payment` tinyint(1) DEFAULT 0,  
  `days` int(11) DEFAULT NULL,  
  PRIMARY KEY (`reservation_id`)  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `guard_reservation_history` */

```
CREATE TABLE `guard_reservation_history` (  
  `reservation_id` int(11) NOT NULL,  
  `res_datetime` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),  
  `start_date` date DEFAULT NULL,  
  `end_date` date DEFAULT NULL,  
  `schedule_details` text DEFAULT NULL,  
  `payment` tinyint(1) DEFAULT 0,  
  `reservation_end` tinyint(1) DEFAULT NULL,  
  `cancel_reservation` tinyint(1) DEFAULT NULL,  
  `payment_cancel` tinyint(1) DEFAULT NULL,  
  PRIMARY KEY (`reservation_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `incident_guard` */

CHAPTER – 5

```
CREATE TABLE `incident_guard` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `incident_id` int(11) DEFAULT NULL,  
  `guard_id` int(11) DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `incident_id` (`incident_id`),  
  KEY `guard_id` (`guard_id`),  
  CONSTRAINT `incident_guard_ibfk_1` FOREIGN KEY (`incident_id`) REFERENCES `incident_report` (`incident_id`) ON  
  DELETE CASCADE ON UPDATE CASCADE,  
  CONSTRAINT `incident_guard_ibfk_2` FOREIGN KEY (`guard_id`) REFERENCES `guard` (`guard_id`) ON DELETE  
  CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `incident_report` */

```
CREATE TABLE `incident_report` (  
  `incident_id` int(11) NOT NULL AUTO_INCREMENT,  
  `date` date DEFAULT NULL,  
  `time` time DEFAULT NULL,  
  `description` text DEFAULT NULL,  
  `timestamp` datetime DEFAULT NULL,  
  `location_id` int(11) DEFAULT NULL,  
  PRIMARY KEY (`incident_id`),  
  KEY `location_id` (`location_id`),  
  CONSTRAINT `incident_report_ibfk_1` FOREIGN KEY (`location_id`) REFERENCES `location_details` (`location_id`) ON  
  DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `interview` */

```
CREATE TABLE `interview` (  
  `interview_id` int(11) NOT NULL AUTO_INCREMENT,  
  `applicant_id` int(11) DEFAULT NULL,  
  `date` date NOT NULL,  
  `time` time DEFAULT NULL,
```

CHAPTER – 5

```
`remarks` text NOT NULL,  
PRIMARY KEY (`interview_id`),  
KEY `applicant_id` (`applicant_id`),  
CONSTRAINT `interview_ibfk_1` FOREIGN KEY (`applicant_id`) REFERENCES `job_application` (`applicant_id`) ON  
DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `interview_history` */

```
CREATE TABLE `interview_history` (  
  `id` int(11) NOT NULL,  
  `applicant_id` int(11) DEFAULT NULL,  
  `date` date DEFAULT NULL,  
  `time` time DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `applicant_id` (`applicant_id`),  
  CONSTRAINT `interview_history_ibfk_1` FOREIGN KEY (`applicant_id`) REFERENCES `job_application_history`  
  (`applicant_id`) ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `invoice` */

```
CREATE TABLE `invoice` (  
  `invoice_id` int(11) NOT NULL,  
  `client_id` int(11) DEFAULT NULL,  
  `datetime` datetime DEFAULT NULL,  
  `total_amount` double DEFAULT NULL,  
  PRIMARY KEY (`invoice_id`),  
  KEY `invoice_ibfk_2` (`client_id`),  
  CONSTRAINT `invoice_ibfk_2` FOREIGN KEY (`client_id`) REFERENCES `client` (`client_id`) ON DELETE CASCADE  
  ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `job_application` */

```
CREATE TABLE `job_application` (  

```

CHAPTER – 5

```
`applicant_id` int(11) NOT NULL AUTO_INCREMENT,  
`candidate_cnic` varchar(50) DEFAULT NULL,  
`candidate_phone_no` varchar(20) DEFAULT NULL,  
`candidate_name` varchar(50) DEFAULT NULL,  
`date` datetime NOT NULL,  
`candidate_email` varchar(50) NOT NULL,  
`candidate_experience` varchar(50) NOT NULL,  
`interview_request` tinyint(1) DEFAULT 0,  
PRIMARY KEY (`applicant_id`)  
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `job_application_history` */

```
CREATE TABLE `job_application_history` (  
  `applicant_id` int(11) NOT NULL,  
  `candidate_cnic` varchar(50) DEFAULT NULL,  
  `candidate_phone_no` varchar(20) DEFAULT NULL,  
  `candidate_name` varchar(50) DEFAULT NULL,  
  `date` datetime NOT NULL,  
  `candidate_email` varchar(50) NOT NULL,  
  `candidate_experience` varchar(50) NOT NULL,  
  `timestamp` timestamp NULL DEFAULT NULL,  
  `interview_operation` tinyint(1) DEFAULT NULL,  
  `application_operation` tinyint(1) DEFAULT NULL,  
  PRIMARY KEY (`applicant_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

/*Table structure for table `location_details` */

```
CREATE TABLE `location_details` (  
  `location_id` int(11) NOT NULL AUTO_INCREMENT,  
  `location` text DEFAULT NULL,  
  `latitude` double DEFAULT NULL,  
  `longitude` double DEFAULT NULL,  
  PRIMARY KEY (`location_id`)
```

CHAPTER – 5

```
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
/*Table structure for table `salary_details` */
```

```
CREATE TABLE `salary_details` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `guard_id` int(11) DEFAULT NULL,  
  `salary_generated` double DEFAULT NULL,  
  `hours_worked` double DEFAULT NULL,  
  `deducted_amount` double DEFAULT NULL,  
  `month_year` date DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `guard_id` (`guard_id`),  
  CONSTRAINT `salary_details_ibfk_1` FOREIGN KEY (`guard_id`) REFERENCES `guard` (`guard_id`) ON DELETE  
  CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
/*Table structure for table `supervisor` */
```

```
CREATE TABLE `supervisor` (  
  `supervisor_id` int(11) NOT NULL,  
  `supervisor_name` varchar(50) DEFAULT NULL,  
  `supervisor_cnic` varchar(50) DEFAULT NULL,  
  `supervisor_phone_no` varchar(20) DEFAULT NULL,  
  `supervisor_password` varchar(50) DEFAULT NULL,  
  `supervisor_address` varchar(50) DEFAULT NULL,  
  PRIMARY KEY (`supervisor_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

CHAPTER – 5

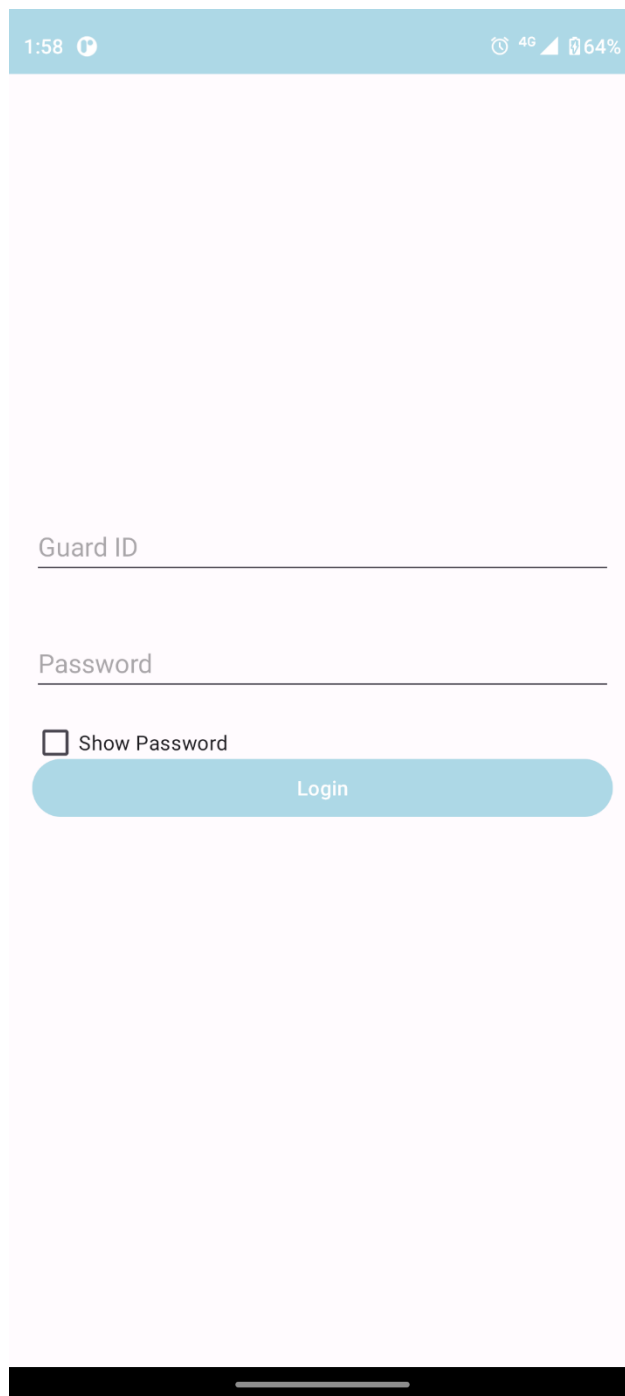
5.3.1 Screenshots:

5.3.2 Android:



Figure 19: Splash Screen

CHAPTER – 5



A mobile application login screen mockup. The screen has a light pink background. At the top is a teal status bar with the time '1:58', a location icon, '4G' network, and '64%' battery. Below the status bar is a large teal header area. The main content area is light pink and contains two text input fields labeled 'Guard ID' and 'Password'. Below the 'Password' field is a checkbox labeled 'Show Password'. At the bottom of the form is a teal rounded rectangular button labeled 'Login'. A black home indicator bar is at the very bottom of the screen.

1:58

Guard ID

Password

☐ Show Password

Login

Figure 20: login Screen

CHAPTER – 5



Figure 21: logged in Screen

CHAPTER – 5

5.3.3 Web Application:

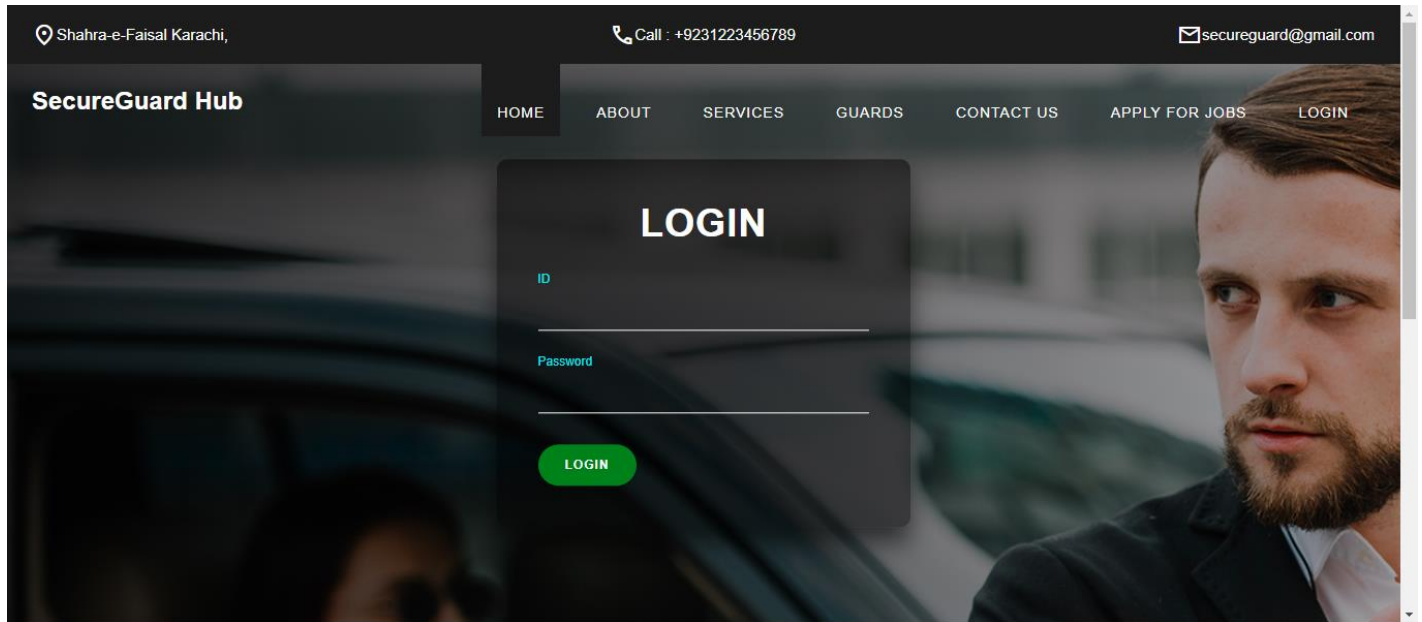


Figure 22: Web login

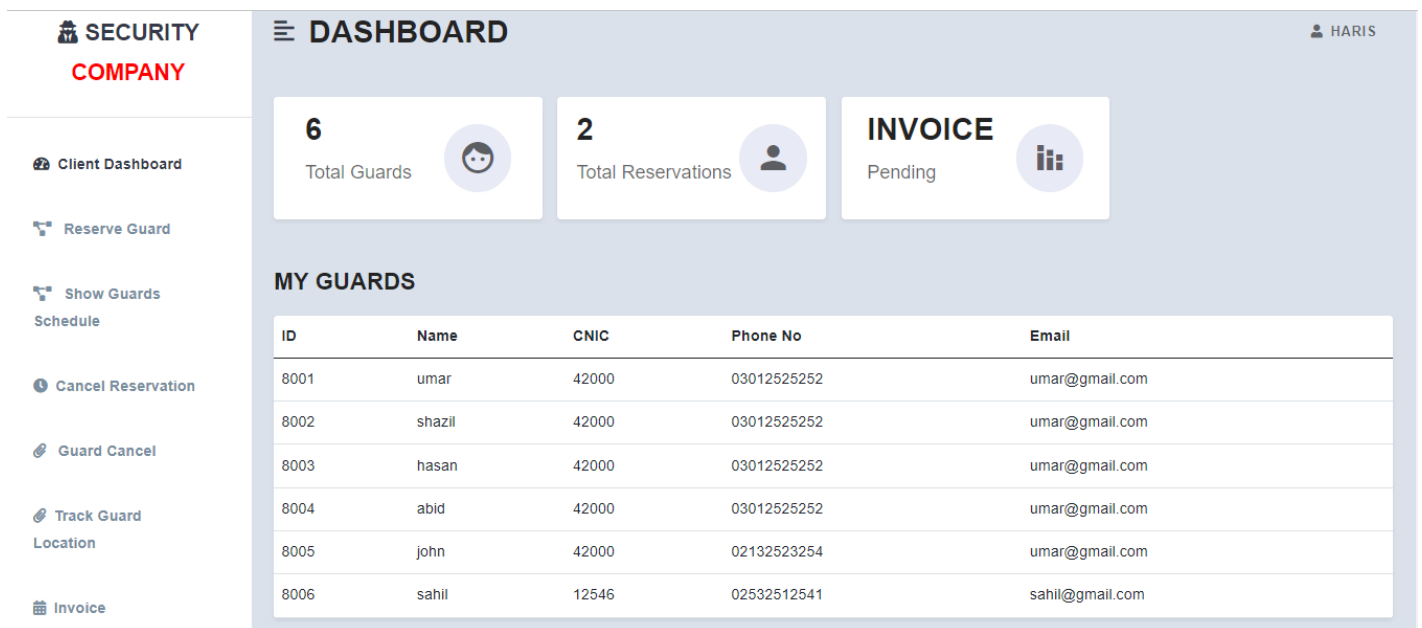


Figure 23: Client Dashboard

CHAPTER – 5

SECURITY
COMPANY

Client Dashboard

Reserve Guard

Show Guards
Schedule

Cancel Reservation

Guard Cancel

Track Guard
Location

Invoice

DASHBOARD

Search by Guard ID:

SEARCH

Guard ID	Guard Name	Duty Shift	Start Time	End Time	Start Date	End Date	Location
8001	umar	morning	08:30:00	20:30:00	2024-03-01	2024-03-30	Eiffel Tower
8002	shazil	morning	08:30:00	20:30:00	2024-03-01	2024-03-30	Eiffel Tower
8003	hasan	morning	08:30:00	20:30:00	2024-03-01	2024-03-30	Eiffel Tower
8004	abid	morning	08:30:00	20:30:00	2024-03-01	2024-03-30	Eiffel Tower
8005	john	morning	11:10:00	21:10:00	2024-03-01	2024-03-09	Eiffel Tower
8006	sahil	morning	11:10:00	21:10:00	2024-03-01	2024-03-09	Eiffel Tower

Figure 24: Guard Schedule

SECURITY
COMPANY

Client Dashboard

Reserve Guard

Show Guards
Schedule

Cancel Reservation

Guard Cancel

Track Guard
Location

Invoice

DASHBOARD

CANCEL RESERVATION

Reservation ID	Client ID	Client Name	Guard IDs	Guard Names	Reservation DateTime	Operations
1	1001	haris	8001,8002,8003,8004	umar,shazil,hasan,abid	2024-03-10 10:47:27	
2	1001	haris	8005,8006	john,sahil	2024-03-10 10:47:31	

Figure 25: Cancel Reservation

CHAPTER – 5

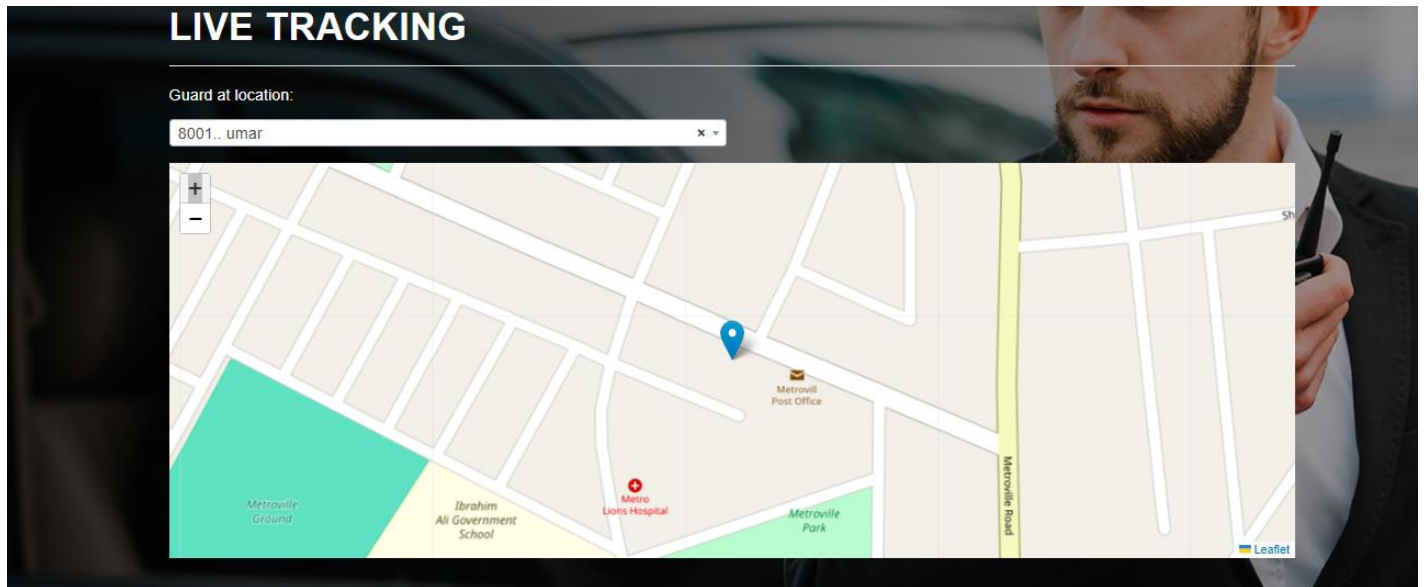


Figure 26: Guard Live Tracking

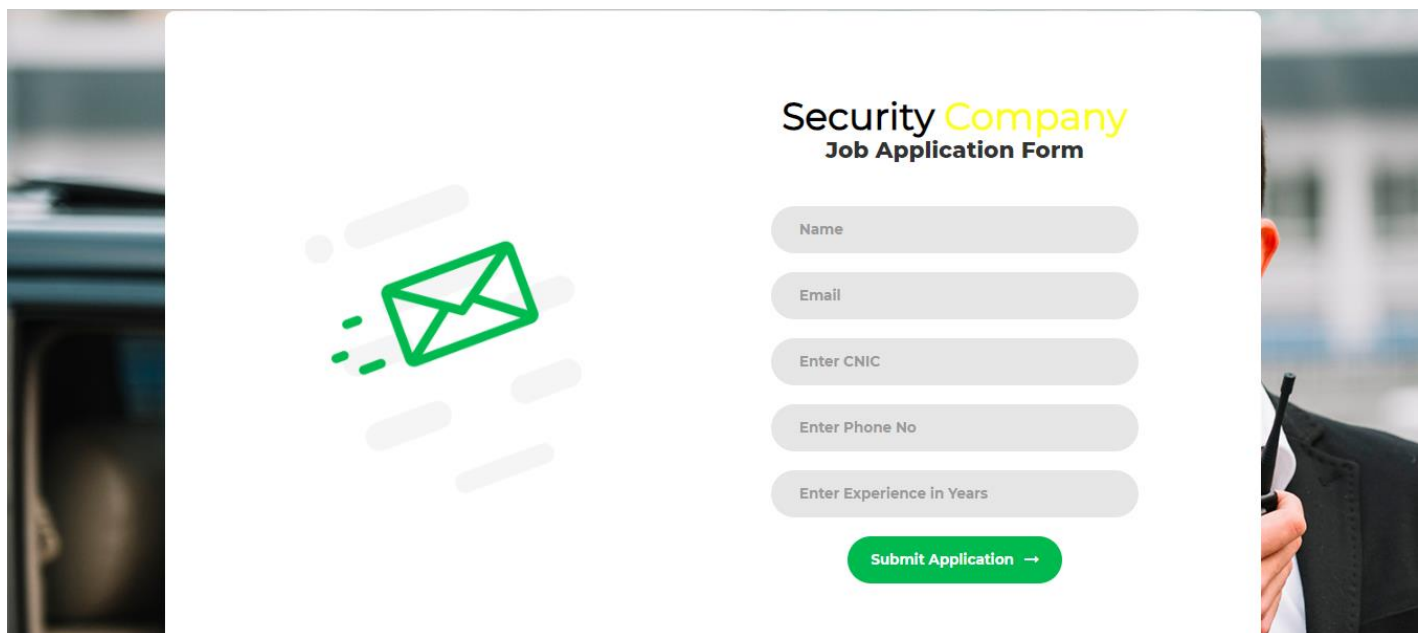
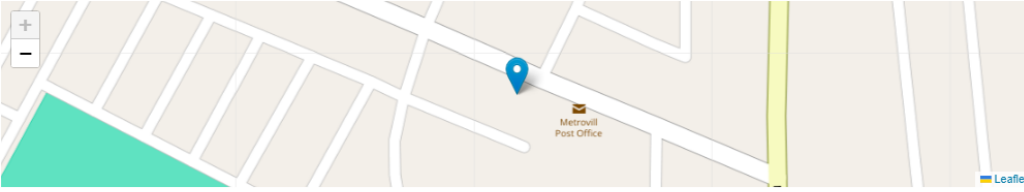


Figure 27: Apply for job

CHAPTER – 5

Guard ID:
8001.umar



Duty Location: Iqra University

Check-in Time:
02/12/2024 04:58 PM

Check-out Time:
02/12/2024 04:58 PM

[Check-in](#) [Check-out](#) [Absent](#) [Late](#)

Figure 28: Attendance Marking

SECURITY
COMPANY

- Supervisor Dashboard
- Call for Interview
- Guards Attendance
- Guards Schedule
- Reservations
- Show attendance
- Report Incident

SEARCH ATTENDANCE

DASHBOARD

Guard ID:
8001..umar

Start Date:
mm/dd/yyyy

End Date:
mm/dd/yyyy

SEARCH

Figure 29: Search Attendance

CHAPTER – 5

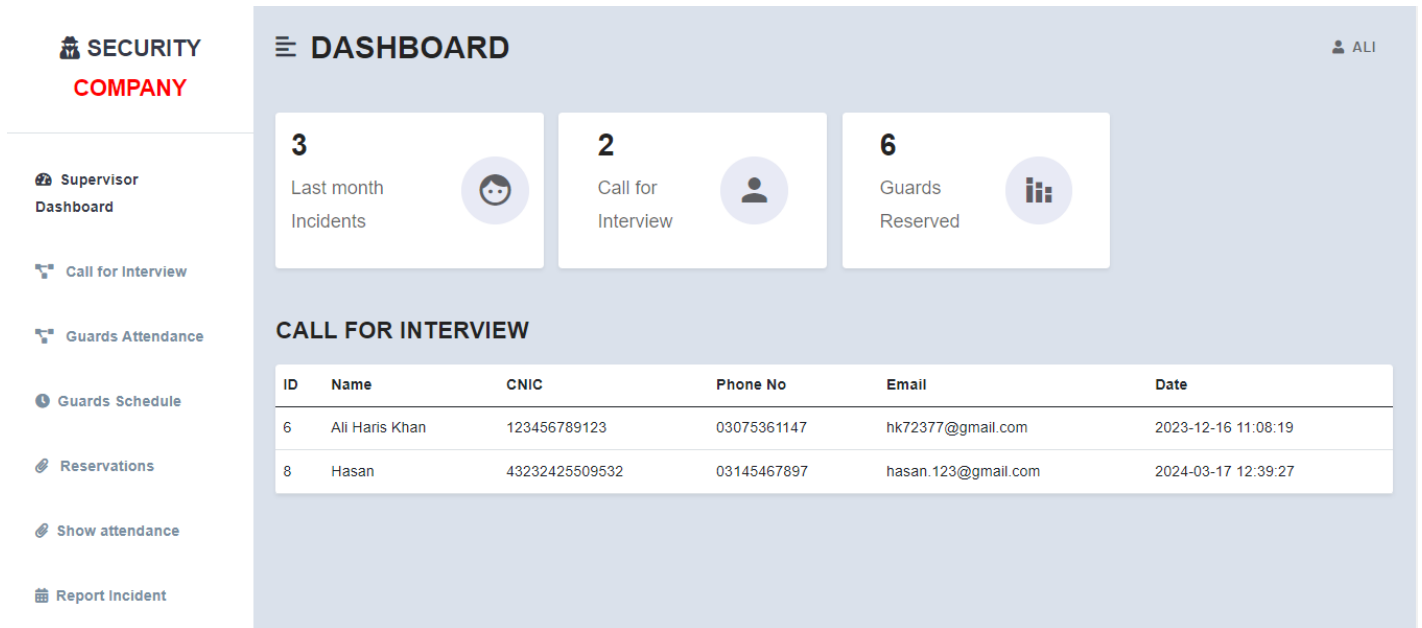


Figure 30: Supervisor Dashboard

The Incident Report form is overlaid on a background image of a security guard. It includes a **DASHBOARD** button in the top left. The form fields are: **Incident Date** (mm/dd/yyyy), **Incident Time** (HH:MM), **Guard at location:** (Select guards), **Incident location:** (Location), and **Description** (Enter Description). A **SUBMIT** button is located at the bottom left.

Figure 31: Insert Incident Report

CHAPTER – 5



SECURITY COMPANY

DASHBOARD

RESERVATIONS

Reservation ID	Client ID	Client Name	Guard IDs	Guard Names	Reservation DateTime	Operations
1	1001	haris	8001,8002,8003,8004	umar,shazil,hasan,abid	2024-03-17 13:04:36	
2	1001	haris	8005,8006	john,sahil	2024-03-17 13:04:33	

Supervisor Dashboard

- Call for Interview
- Guards Attendance
- Guards Schedule
- Reservations
- Show attendance
- Report Incident

Figure 32: Reservation Report

5.3 Summary:

This Chapter consist of test cases and development in which we explained about the frontend, backend design and database queries and some screenshot of our application. In this chapter we briefly give some explanation about the type of error happen on the system, source code of validation and etc. So from the next part some information flows are given to know how to system assignation works. The database queries are also provided to show how the system retrieve the data while functioning. As the system is very complex multiple external library were required to fulfill the requirement of the user, all the external libraries are mentioned below to give a detail description and screenshot of the application.

6.1 Introduction:

In this chapter, we have discussed about the test cases to determine whether the Web-Application is working the way it should and producing the expected results. We also test cases and usability test cases to test our software. This will help the readers to know about all the minor as well as the major working options of the web-application. After the completion of the implementation phase testing plays a vital role for making sure that the system works properly. Once the Web-application was developed the testing phase started. Each and every screen and button were tested according to the requirements and functionalities. Even though the web-application is complex it does use I treated functionality as one window is being used multiple times for different kinds of functionalities. For instance, the same data is being utilized for the Scheduling objectives as well as the live-tracking section. do to the re usability of our code the total test cases of the web-application turned out to be . Each test case has the requirement reference, project name, Web-application name while providing all the details of the test case. in detail attributes of the test case, such as test case ID, Test case description, test steps, expected result, pass or fail status, preparation date, running date end the date at which it was tested.

CHAPTER – 6

6.2 Test Cases:

Test Case ID	1.1	Test Case Description	Test the Login Functionality by User.		
Prepared By	Hamza Muzaffar	Project Name	SecureGuard Hub		
Tester's Name	Ali Haris Khan	Date Prepared	11 th Jan 2024	Date Run	21 th Jan 2024
S #	Prerequisites:				
1	User should be registered before login.				
<u>Test Scenario</u>	Verify on entering valid Admin id and password, the User can login				
Step #	Step Details	Expected Results	Actual Results	Status	
1	Open Site.	Site should open	As Expected	Pass	
2	Enter User id & Password	Credential can be entered	As Expected	Pass	
3	Click Login	User is logged in	As Expected	Pass	

Table 14: Test case 1

CHAPTER – 6

Test Case ID	1.2	Test Case Description	Test the Logout Functionality by User.		
Prepared By	Hamza Muzaffar	Project Name	SecureGuard Hub		
Tester's Name	Ali Haris Khan	Date Prepared	11 th Jan 2024	Date Run	21 th Jan 2024
S #	Prerequisites:				
1	User should be logged in.				
Test Scenario	Verify by clicking logout button in client dashboard				
Step #	Step Details	Expected Results	Actual Results	Status	
1	Open Client dashboard.	Dashboard should open	As Expected	Pass	
2	Click logout button	Client is logged in	As Expected	Pass	

Table 15: Test case 2

CHAPTER – 6

Test Case ID	1.3	Test Case Description	Test the live location tracking Functionality of Guard by Client.		
Prepared By	Hamza Muzaffar	Project Name	SecureGuard Hub		
Tester's Name	Ali Haris Khan	Date Prepared	11 th Jan 2024	Date Run	21 th Jan 2024
S #	Prerequisites:				
1	Guard latitude and longitude data should exist in database table.				
2	Guard should be reserved by Client logged in.				
<u>Test Scenario</u>	Verify on selecting any guard in live location tracking page.				
Step #	Step Details	Expected Results	Actual Results	Status	
1	Open Client dashboard.	Dashboard should open	As Expected	Pass	
2	Click on Track Guard Location	Track Guard Location page should open	As Expected	Pass	
3	Select on dropdown	Reserved guard list should show	As Expected	Pass	
4	Select any guard from list	Location should show on map	As Expected	Pass	

Table 16: Test case 3

CHAPTER – 6

Test Case ID	1.4	Test Case Description	Test the Guard reservation Functionality by User
Prepared By	Hamza Muzaffar	Project Name	SecureGuard Hub

Tester's Name	Ali Haris Khan	Date Prepared	11 th Jan 2024	Date Run	21 th Jan 2024
----------------------	----------------	----------------------	---------------------------	-----------------	---------------------------

S #	Prerequisites:
1	User should be logged in.

<u>Test Scenario</u>	Verify by reserving guard
-----------------------------	---------------------------

Step #	Step Details	Expected Results	Actual Results	Status
1	Open Client dashboard.	Dashboard should open	As Expected	Pass
2	Click on Reserve guard	Guards list should show	As Expected	Pass
3	Select guards	Guards should select	As Expected	Pass
4	Enter location, timing, shift and other details	Details should entered	As Expected	Pass
5	Click on reserve	Guard should reserved	As Expected	Pass

Table 17: Test case 4

CHAPTER – 6

Test Case ID	1.5	Test Case Description	Test the download Invoice Functionality by User
Prepared By	Hamza Muzaffar	Project Name	SecureGuard Hub

Tester's Name	Ali Haris Khan	Date Prepared	11 th Jan 2024	Date Run	21 th Jan 2024
----------------------	----------------	----------------------	---------------------------	-----------------	---------------------------

S #	Prerequisites:
1	User should be logged in.

<u>Test Scenario</u>	Verify on clicking the download button.
-----------------------------	---

Step #	Step Details	Expected Results	Actual Results	Status
1	Open Client dashboard.	Dashboard should open	As Expected	Pass
2	Enter Click on generate in invoice.	Invoice page should open	As Expected	Pass
3	Click on download invoice	PDF file of invoice should download	As Expected	Pass

Table 18: Test case 5

CHAPTER – 6

Test Case ID	1.6	Test Case Description	Test the Guard Schedule Functionality
Prepared By	Hamza Muzaffar	Project Name	SecureGuard Hub

Tester's Name	Ali Haris Khan	Date Prepared	11 th Jan 2024	Date Run	21 th Jan 2024
----------------------	----------------	----------------------	---------------------------	-----------------	---------------------------

S #	Prerequisites:
1	User should be logged in.

<u>Test Scenario</u>	Verify by reserving guard and then go to guard schedule page.
-----------------------------	---

Step #	Step Details	Expected Results	Actual Results	Status
1	Open Client dashboard.	Dashboard should open	As Expected	Pass
2	Click on Reserve guard	Guards list should show	As Expected	Pass
3	Select guards	Guards should select	As Expected	Pass
4	Enter location, timing, shift and other details	Details should entered	As Expected	Pass
5	Click on reserve	Guard should reserved	As Expected	Pass
6	Go to Client dashboard	Dashboard should open	As Expected	Pass
7	Click on Guard Schedule	Guard Schedule page should open and schedule should show.	As Expected	Pass

Table 19: Test case 6

CHAPTER – 6

Test Case ID	1.7	Test Case Description	Test the marking guard attendance Functionality
Prepared By	Hamza Muzaffar	Project Name	SecureGuard Hub

Tester's Name	Ali Haris Khan	Date Prepared	11 th Jan 2024	Date Run	21 th Jan 2024
----------------------	----------------	----------------------	---------------------------	-----------------	---------------------------

S #	Prerequisites:
1	To show on map, latitude and longitude data should exist in gps_tracking table.
2	For check in and check out, duty start time and duty stop time should exist in gps_tracking table (can also inserted manually).

Test Scenario	Verify on entering valid Supervisor id and password, the Supervisor can login
----------------------	---

Step #	Step Details	Expected Results	Actual Results	Status
1	Open Supervisor dashboard.	dashboard should open	As Expected	Pass
2	Click on guard attendance	Guard attendance page should open	As Expected	Pass
3	Click on dropdown	Guard list should show	As Expected	Pass
4	Select a guard	Guard location should show on map and check in and checkout fields show fill with start and stop time from gps_tracking table for selected guard	As Expected	Pass
5	Click on check in, checkout	The attendance should marked in database	As Expected	Pass
6	Click on late	The selected guard should marked late	As Expected	Pass
7	Click on absent	The selected guard should mark absent and check in and checkout should be cleared in guard_attendance table	As Expected	Pass

Table 20: Test case 7

CHAPTER – 6

Test Case ID	1.8	Test Case Description	Test the Login Functionality by Guard in mobile app.
Prepared By	Hamza Muzaffar	Project Name	SecureGuard Hub

Tester's Name	Ali Haris Khan	Date Prepared	11 th Jan 2024	Date Run	21 th Jan 2024
----------------------	----------------	----------------------	---------------------------	-----------------	---------------------------

S #	Prerequisites:
1	Guard should be registered before login.

<u>Test Scenario</u>	Verify on entering valid Guard id and password, the Guard can login
-----------------------------	---

Step #	Step Details	Expected Results	Actual Results	Status
1	Open App.	App should open	As Expected	Pass
2	Enter Guard id & Password	Credential can be entered	As Expected	Pass
3	Click Login	Guard is logged in	As Expected	Pass

Table 21: Test case 8

CHAPTER – 6

Test Case ID	1.9	Test Case Description	Test the GPS data sending Functionality by Guard in mobile app.
Prepared By	Hamza Muzaffar	Project Name	SecureGuard Hub

Tester's Name	Ali Haris Khan	Date Prepared	11 th Jan 2024	Date Run	21 th Jan 2024
----------------------	----------------	----------------------	---------------------------	-----------------	---------------------------

S #	Prerequisites:
1	Guard should be logged in in mobile app.

<u>Test Scenario</u>	Verify on clicking start location service in mobile app
-----------------------------	---

Step #	Step Details	Expected Results	Actual Results	Status
1	Open App.	App should open	1	Open App.
2	Enter Guard id & Password	Credential can be entered	2	Enter Guard id & Password
3	Click Login	Guard is logged in	3	Click Login
4	Click on start location service	Notification should show in notification bar and data should start sending to database	4	Click on start location service
5	Click on stop location service	Notification should stop to show in notification bar and data should stop sending to database	5	Click on stop location service

Table 22: Test case 9

6.3 Summary:

We test our software to get our expected results of our Web-application or whether a system under test satisfies requirements or works correctly. After test cases, we get satisfied results the usability test case.

7.1 Introduction:

This chapter will summarize all of the work completed during the final year of the project, as well as the challenges, limitations, and future work for this project. In this chapter all the major and minor work will be discussed. We have included the limitations of the system in order to help the users to understand the system better. The future work section will provide an overview to the enhancement that can be made with this software. With the passing time the software has a lot vacancies for better additions. This will make the application more effective, and useful. In this section, I will be talking about how future work will enhance the Kinder Learning app. The Security Guard service management system is a web-application that has been designed to provide security guard on easy way.

The app is available in both Client & candidate, and it variety of features of that are perfect for user which need security guard. The live- tracking design reflects the latest research in early guard monitoring., and it is specifically tailored to engage security guard in their duty. The security guard service management web-app already includes some of the best features for reserve guard, monitoring and do clearing invoices - but there are still many new features that can be added to improve the user experience. Future work on this project will allow us to add new activities for user. The future work will be focused on improving the web-app usability and enhancing its features. The app is already equipped with need sources and features to be used by many people to apply for jobs, and the future work will make it even more useful for those who are trying to doing job. We aim to make the web-app more interactive with future updates. We want to create a more personalized experience for each user by adding features.

7.2 System Limitations and Challenges:

SecureGuard Hub is a revolutionary and effective way to reserve guard. It has been designed to be interactive and personalized, which makes it a great tool for monitoring the guard. However, as with every other technology, has its limitations and challenges. In this section, we will talk about the limitations and challenges of security guard service management web-app.

- i. The web-app cannot be used properly on in, which means that if there are no internet connections available the client will not be able to use it at all.

CHAPTER – 7

- ii. The main limitation of SecureGuard Hub is its lack of understanding the access. This means that guard cannot use it when if they are not familiar with basic technology and phone usage as many rural area guard would need 2 3 days to understand this web-app.

The challenges we faced during the development of the project were:

- i. Live Guard tracking was the main challenge we faced during the development.
- ii. One of the major challenges of SecureGuard Hub is that it no such web-app has been made before.

7.3 Future Work:

SecureGuard Hub will be a Web-app that will provide a personalized Security guard for safety of all clients. The web-app will offer experienced security guard, live-guard tracking and display invoices on portal that are tailored to each client needs. It will also include a guard dashboard where guard can track their attendance and schedules. The future of SecureGuard Hub is to make it a more inclusive and diverse environment for all clients. SecureGuard Hub is an web-app that is designed to reserve teir desire-able security guard. The developers are constantly working on new features, updates, and improvements to make the web-app even better for both client and guard. The future work that can be done on SecureGuard Hub is to make it more interactive.

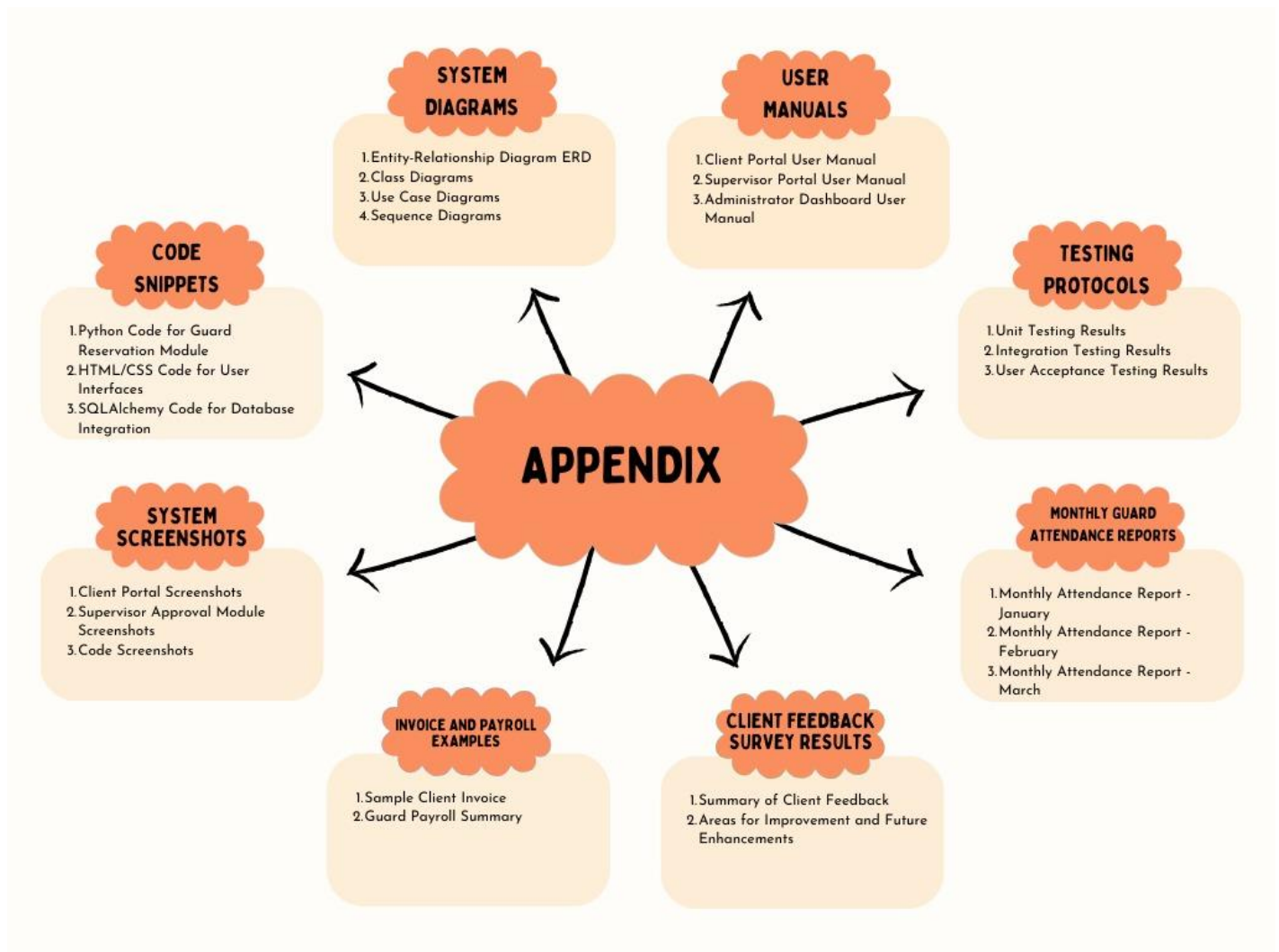
7.4 Conclusion:

We can conclude that client use mobile phones just only for other works but we can use it for reservation purpose too. So we have developed an web-application that will help clients at their safe environment for individuals, businesses which is especially crucial in areas prone to crime or during events with large crowds. With this web-app, clients can monitor their guards and relax.By using SecureGuard Hub will also provide candidate with the opportunity to be apply for job in easy way. This is a great web-app that will benefit client and candidate on the long run.

REFERENCES

- i. "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin
- ii. "Code Complete: A Practical Handbook of Software Construction" by Steve McConnell
- iii. "The Pragmatic Programmer: Your Journey to Mastery" by Andrew Hunt and David Thomas
- iv. "Effective Java" by Joshua Bloch
- v. "Refactoring: Improving the Design of Existing Code" by Martin Fowler
- vi. "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
- vii. "Test-Driven Development: By Example" by Kent Beck
- viii. "Clean Architecture: A Craftsman's Guide to Software Structure and Design" by Robert C. Martin
- ix. "JavaScript: The Good Parts" by Douglas Crockford
- x. "Head First Design Patterns" by Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra
- xi. "Flask Framework Cookbook" by Shalabh Aggarwal
- xii. "Flask Framework Cookbook" by Steve Westgarth

APPENDIX



These appendices provide additional details, documentation, and supporting materials related to the Guard Management System project.