Secure Mobile Attendance System with Face Recognition and Edge Computing

ENSURING QUALITY IN A SECURE MOBILE ATTENDANCE SYSTEM WITH FACE RECOGNITION: A QA APPROACH

THESIS

FIELD

PROJECT

by

Bently Edyson *2301894590*

BINUS INTERNATIONAL

BINUS UNIVERSITY

JAKARTA

2023

Secure Mobile Attendance System with Face Recognition and Edge Computing

ENSURING QUALITY IN A SECURE MOBILE ATTENDANCE SYSTEM WITH FACE RECOGNITION: A QA APPROACH

THESIS

Proposed as a requirement for obtaining
Sarjana degree at
Program Computer Science
Education Level Strata-1 (Sarjana/Bachelor)

by

Bently Edyson *2301894590*

BINUS INTERNATIONAL
BINUS UNIVERSITY
JAKARTA
2023

Secure Mobile Attendance System with Face Recognition and Edge Computing

ENSURING QUALITY IN A SECURE MOBILE ATTENDANCE SYSTEM WITH FACE RECOGNITION: A QA APPROACH

THESIS

Prepared By:

[illegible]

Bently Edyson

2301894590

Approved by:

Supervisor	Co-Supervisor
[illegible]	
Jude Joseph Lamug Martinez	**Ardimas Andi Purwita**
D4017	**D6405**

BINUS UNIVERSITY

Jakarta

2023

PERNYATAAN

STATEMENT

Dengan ini, saya/kami,

With this, I/We,

Nama (*Name*):	Bently Edyson
NIM (*Student ID*):	2301894590
Judul Tesis (*Thesis Title*):	ENSURING QUALITY IN A SECURE MOBILE ATTENDANCE SYSTEM WITH FACE RECOGNITION: A QA APPROACH

Memberikan kepada Universitas Bina Nusantara hak non-eksklusif untuk menyimpan,memperbanyak, dan menyebarluaskan tesis saya/kami, secara keseluruhan atau hanya sebagian atau hanya ringkasannya saja, dalam bentuk format tercetak atau elektronik.
Hereby grant to my/our school, Bina Nusantara University, the non-exclusive right to archive, reproduce, and distribute my/our thesis, in whole or in part, whether in the form of a printed or electronic format.

Menyatakan bahwa saya/kami, akan mempertahankan hak exclusive saya/kami, untuk menggunakan seluruh atau sebagian isi tesis saya/kami, guna pengembangkan karya di masa depan, misalnya dalam bentuk artikel, buku, perangkat lunak, ataupun sistem informasi.
I/We acknowledge that I/we retain exclusive rights of my/our thesis by using all or part of it in a future work or output, such as an article, a book, software, or information system.

Catatan: Pernyataan ini dibuat dalam 2 (dua) bahasa, Indonesia dan Inggris, dan apabila terdapat perbedaan penafsiran, maka yang berlaku adalah versi Bahasa Indonesia.
Note: This Statement is made in 2 (two) languages, Indonesian and English, and in the case of a different interpretation, the Indonesian version shall prevail.

Jakarta, 30/06/2023

[illegible]

Bently Edyson
2301894590

BINUS INTERNATIONAL

BINUS UNIVERSITY

Major Computer Science

Bachelor of Science Computer Thesis

Semester 8 - Year 2023

Secure Mobile Attendance System with Face Recognition and Edge Computing

ENSURING QUALITY IN A SECURE MOBILE ATTENDANCE SYSTEM WITH FACE RECOGNITION: A QA APPROACH

Bently Edyson 2301894590

Abstract

The Secure Mobile Attendance System with Face Recognition and Edge Computing was created in response to the growing need for reliable and secure attendance solutions. This solution enhances overall easy to use and good user experience while offering a dependable and practical method of tracking attendance. As Quality Assurance (QA), it was the author's responsibility to use a variety of testing techniques, including UI testing, usability testing, unit testing, integration testing and beta testing, in this project to assure the quality and dependability of the system. With a focus on the testing and quality assurance process, the author cover the development and implementation of the Secure Mobile Attendance System Test in this thesis. The outcomes of this study emphasize the advantages of employing facial recognition and edge computing technologies in attendance systems and highlight the significance of applying Quality Assurance (QA) techniques in the creation of safe mobile applications.

Keywords
Easy to use, user experience, Quality Assurance (QA), UI testing, usability testing, unit testing, integration testing, beta testing, quality assurance, test

ACKNOWLEDGEMENT

This thesis is developed by the author and the author’s team to complete their undergraduate study in Binus International University. The development of this thesis would not have been possible without the help of those who supported the author throughout this journey.

The author would like to express their gratitude to those have helped and supported the author:

- To the author’s family who provided the author with love and support,
- To the author’s supervisors, Sir Jude joseph Lamug Martinez and Sir Ardimas Andi Purwita for their constant guidance and meaningful advice,
- To the author’s team members and best friends, Yowen and Bryan Putra, who have worked together with the author since the start of this thesis and stayed together through it all,
- To the author’s close confidant, Darlene Calista, Josephine Nicole, Wilona Anastasia, Jennifer Lee, Michele Liang, Monique Senjaya, Jason Christian, who has been there since day 1 and provided never-ending emotional support, and,
- To all Computer Science lecturers in Binus International University who have taught the author with patience

Table of Content

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Background

Around the world, attendance systems that used paper-based records were very common, which made them spread across the world. These are used to keep track of a student's or employee's daily attendance. As time passed, new discoveries and advancements in technology kept growing. They used to keep track of everyone's attendance in a thick book. For example, a time clock that tracked attendance used a paper-based record where users were required to insert the timecard so that the machine could input the user's clock time. Nowadays, modern attendance systems make sure that the use of technology and the internet will make them more efficient and easier to use. Fingerprint scanners are a form of modern attendance system because each living human has their own unique code, such as their fingerprint.

The use of RFID card scanners is another popular method for maintaining attendance. The company uses this as a daily attendance tool to keep track of everyone in the company. By giving everyone their own RFID card, they are required to come and scan their card at the RFID scanner. Moreover, the use of mobile applications as an attendance system is also popular. Each user is required to make their own account in the application that they need to install. Each of these methods has its own issues that will be covered in the chapter that follows.

The author's team have decided to concentrate on mobile attendance applications with facial recognition among the different mobile attendance monitoring options available for several reasons. First, face recognition technology is an appealing replacement for conventional attendance systems since it has recently gotten more advanced and accurate. Second, facial recognition eliminates the need for physical devices like cards or readers, which can be misplaced or stolen, making attendance tracking more convenient and secure. Finally, face-recognition mobile attendance applications are more user-friendly and enable employees or students to check in and out swiftly and easily without interfering with their job or studies. Finally, facial recognition provides a more hygienic option, which is crucial in environments where individuals are close to one another, and the risk of virus transmission is considerable. Mobile attendance applications with face recognition help to reduce the requirement for shared devices or surfaces and promote a safer working or learning environment for everyone concerned.

1.2 Scope

In this section, the author outlines both the collective scope of their group and their own individual scope pertaining to this project.

1.2.1 Group Scope

The project at hand involved both the author and their team, each with distinct responsibilities that are shown in table 1.1. The author’s team primary objective is to create a mobile application that uses facial recognition to make sure that users are physically present, such as, in the office. This application will consist of several features, such as:

- Visualization of face recognition application
- Secure data as images will not be stored in database.
- Face Recognition feature

Table 1.1 Scope of Activities

Name	**Role**
Bently Edyson	- Design and implement usability testing on people's behavior on application. - Conduct unit testing on each important function and class. - Conduct integration testing on ai face recognition. - Conduct UI testing on the components of the application - Conduct beta testing to assess the system's performance and identify any bugs
Yowen	- Choosing a suitable face detection algorithm to be used. - Testing different pixel normalization methods and evaluating their performance of different face recognition models. - Integrating chosen model into an android application. - handle backend database system for the application.
Bryan Putra Suandi	- Designing the user interface of the application. - Developing the front end of the applica-

	tion. - Developing a live detection feature in the application for image preprocessing.

1.2.2 Individual Scope of Work

The Author’s responsibility was to:

- Usability testing will be performed to observe how users interact with the application.
- UI testing will be conducted to ensure that the components and design run smoothly.
- Unit testing will be conducted for critical functions, classes, or models within the application.
- Integration testing for data communication with Firebase
- Conduct technical beta testing to assess the system's performance and identify any bugs or errors encountered during real-world usage.

1.3 Aim and Benefits

In this section, the author’s team will discuss the objectives and advantages of this project.

1.3.1 Aims

This subsection will outline both the group-wide and individual goals of this project.

1.3.1.1 Group Aim

The aim of the group is to create a mobile application that uses facial recognition with edge computing to make sure that the user is physically present. With this application, the user's images will not be stored or transmitted during the use of the application, which enhances the security of the data. The application's key features include:

- Minimum of 70% score in usability testing
- 1 second or lower for the model’s inference time
- 25MB or less for the model size

1.3.1.2 Individual Aim

The aims of the author for this project can be broken down into these bullet points:

- Observe people's interactions with the application through usability testing analysis.
- Test crucial functions and classes through unit testing.
- Test Firebase for data communication through integration testing.
- Conduct beta testing, including User Acceptance Testing (UAT), to identify any bugs or errors that may occur during user interaction.

1.3.2 Benefits

In this section, the author’s team will explore the primary benefits of the Mobile Attendance System and the advantages of the author's objective.

1.3.2.1 Main Benefits

The main benefits of developing this Mobile Attendance System are:

- Helping businesses avoid the need to buy additional hardware.
- Protecting user privacy by avoiding the storage or transmission of user images to the server.
- Promoting hygiene by eliminating the need for users to repeatedly perform fingerprint scans.

1.3.2.2 Benefits of Personal Aim

As outlined in Section 1.3.1.2, the author has specific aims for this project, which offer several benefits, including:

- Analyzing user behavior during the use of the mobile attendance system application to enhance its design and user experience.
- Conducting integration and unit testing to identify and resolve any bugs or errors that may arise during user interaction with the application, ensuring its smooth operation when used by the intended users.

1.4 Structure

This thesis is composed of seven chapters, which will be briefly outlined in this section.

1.4.1 Introduction

In Chapter 1, the author outlined the project along with its scope, objectives, aims, vision, and mission.

1.4.2 Theoretical Foundation

In Chapter 2, the author will cover the various types of tests used in Quality Assurance (QA), including usability testing, UI testing, unit testing, integration testing, and beta testing.

1.4.3 Problem Analysis

Chapter 3, the author will delve into further detail about the problem at hand. The author will also explain how these tests can be utilized to address the issue.

1.4.4 Solution Design

In Chapter 4, the author will be focusing on the design of the solution, which will be informed by data gathered from the previously mentioned tests. Additionally, the chapter will explore the impact of these tests on other aspects of the project, including the front-end and back-end site, as well as the user experience.

1.4.5 Implementation

In Chapter 5, the author will focus on the results of the tests previously conducted. The author will analyze the data and present the findings in this chapter.

1.4.6 Discussion

In Chapter 6, the author will discuss the key results of the thesis and will connect the analytical findings with the author's original aim.

1.4.7 Conclusion and Recommendation

In the final chapter of the project, Chapter 7, the author will synthesize and present all the critical information that has been obtained throughout the work. Additionally, the author will provide recommendations for potential avenues of future research or development.

CHAPTER 2

THEORETICAL FOUNDATION

This Chapter will explore the theoretical foundation and the methods used to develop the mobile attendance application with face recognition, from the perspective of a QA. Usability, integration, and beta testing are crucial elements of the software development process, and they will all be covered in depth in this chapter.

2.1 Introduction

These are applications created to use facial recognition technology to track attendance. This technology makes tracking attendance simple and effective by allowing users to mark attendance by scanning their faces with a mobile device. Due to their accuracy and usability, mobile attendance applications using facial recognition are becoming more and more used in a variety of sectors, including business and education. Additionally, as they are unable to complete their attendance without connecting to dedicated wi-fi or internet in their office, this also covers the users' locations.

2.1.1 Overview

The development and testing of mobile attendance applications using face recognition technologies heavily relies on the theoretical background. To assure the application's usability, integration, and beta performance, a Quality Assurance (QA), I must possess a thorough understanding of the theoretical concepts and principles that guide its design and functionality.

2.1.2 Face Recognition Technology

A key element of the mobile attendance system application is a face recognition technology. This section explores the practical aspects of face recognition and its role within the application. It emphasizes the face recognition system's primary attributes and capabilities while concentrating on how it works with the mobile attendance application to reliably identify and authenticate people based on their facial traits. It is essential to comprehend how facial recognition technology is implemented and functions to ensure that it is used effectively within the mobile attendance system.

2.1.3 Quality Assurance Practices

The development of the mobile attendance system application involves Quality Assurance (QA) procedures on a regular basis [1]. To guarantee the application's reliability, functionality, and user happiness, this section focuses on the QA procedures, techniques, and processes that are used. It covers a range of QA topics, including test case development, test execution, defect management, and quality control [2]. The author may identify and fix software bugs, improve system performance, and create a user-satisfactory mobile attendance system through thorough QA procedures.

2.1.4 Methodology

In the context of developing a mobile attendance application with face recognition, a systematic and well-defined methodology is essential for ensuring the quality, reliability, and effectiveness of the application. This section presents the methodology employed by the Quality Assurance (QA) role in the project to achieve these objectives.

2.1.4.1 Requirements Gathering

It is important to work closely with all parties involved, including the development team and end users, to understand and collect the requirements for the mobile attendance application. The QA job takes an active involvement in efforts to gather requirements, working with stakeholders to create precise and quantifiable goals, functional needs, and user expectations. A mobile attendance system that satisfies user expectations and is in line with the project's objectives can be developed with the help of QA.

2.1.4.2 Test Planning

A thorough test strategy is created, detailing the testing methodology, goals, and scope for the mobile attendance application. UI testing, usability testing, unit testing, integration testing, and beta testing are all covered by the test plan. The plan is created after identifying and including the test environment, test data, and required tools and resources.

2.1.4.3 Test Case Development

All the mobile attendance application's features and functionalities are carefully covered by test cases. Each test case is documented with clear instructions, including the necessary conditions, steps to follow, and expected results. During attendance monitoring, particular emphasis is paid to including edge instances, boundary conditions, and scenarios that reflect real-world usage.

2.1.4.4 Test Execution and Defect Management

The established test plan is followed when executing the developed test cases. Any faults or flaws discovered during testing are recorded using a defect tracking tool, and test results are monitored and tracked. To collect, track, and prioritize reported concerns, a formal defect management process is put in place. Defect

Management is a method for identifying and resolving defects [3]. The QA job works closely with the development team to quickly identify, correct, and confirm faults.

2.1.4.5 Continuous Improvement

Continuous improvement is an integral part of the testing process for the mobile attendance application. Feedback from end-users and stakeholders is actively sought and incorporated, and regular retrospectives are conducted to reflect on the testing process and make improvements. The goal of this iterative process is to improve the application's usability and quality. The most used model is PDCA: Plan, Do, Check, Act [4]. These model breakdowns into these details:

- **Step 1: Problem Definition and Approach Formulation**

  During this initial stage, the primary objective is to precisely define the problem and devise an appropriate approach for its resolution. This in-volves addressing questions such as:

  - What is the scope of the problem?
  - What is the desired target or outcome?
  - What approach will be most effective in achieving the desired out-come?

  Furthermore, it entails assembling a team and establishing a timeline for the project.

- **Step 2: Execution (Do)**

  In this phase, the author can execute a comprehensive plan or implement a pilot solution on a smaller scale. Either way, the key is to experiment and test new approaches to evaluate their effectiveness. It is crucial to document the steps taken throughout the process and gather data and feedback along the way.

- **Step 3: Evaluation (Check)**

  This step involves evaluating the chosen approach and comparing the results to the initial expectations outlined during the planning stage. Ask questions such as:

  - Was the approach successful and effective?
  - Did it perform as anticipated? If not, what were the reasons?
  - What aspects worked well and what aspects did not?

  If the approach proved unsuccessful, the author could return to the first step (problem definition and approach formulation) and consider the lessons learned and the reasons behind the lack of intended results. If it was successful, you can proceed to the next step.

- **Step 4: Implementation and Improvement (Act)**

  Drawing from the insights and feedback gained from the previous steps, it is now time to fully implement the new solution. However, it is important to note that this does not imply the final solution or the only approach. Instead, it establishes a new baseline against which future improvements can be measured.

The QA job ensures that the mobile attendance application with face recognition complies with high standards, satisfies user requirements, and offers a reliable and user-friendly solution by using this methodology.

In summary, the methodology used in this project involves requirements gathering, comprehensive test planning, test case development, effective test execution, defect management, and a commitment to continuous improvement. This methodology allows the QA role to contribute effectively to the quality assurance process and ensure the success of the mobile attendance application.

2.1.5 Usability and User Experience

Usability is a critical factor that determines the success of any software application, including mobile attendance apps with face recognition. Designing user-friendly interfaces and boosting user experience can be aided by using theoretical frameworks like User-Centered Design (UCD) and Human-Computer Interaction (HCI).

Additionally, usability testing can also point out product flaws and inform iterative design, leading to better user experiences and long-term success. In general, taking usability into account is crucial for developing user-centered solutions that satisfy the requirements and expectations of the intended consumers.

UX (User Experience) is an approach that allows your users to navigate your website or app without confusion and with ease, providing a smooth experience of your brand [5]. Meanwhile, User-friendly software is exactly what it sounds like: it's a technical solution that is easy for at least most people to use to get their jobs done[6].

The importance of UX is that it can lead to increased user satisfaction, improved productivity, and ultimately, increased usage and adoption of the product or service. UX design involves designing products and services that are user-centered, intuitive, and enjoyable to use, and involves techniques such as user research, prototyping, and usability testing.

For mobile attendance applications that use facial recognition to be successful, usability and user experience are critical factors. Users must be able to easily navigate the app and do their tasks without error. Users can more easily accomplish their goals with the aid of a well-designed user interface that prioritizes usability and user experience, which may lead to increased usage and user satisfaction. Furthermore, poor usability and user experience can cause confusion, dissatisfaction, and app abandonment. Because of this, integrating usability and user experience testing techniques into the design and testing of mobile attendance apps helps guarantee that the app satisfies the needs and expectations of the end users and, ultimately, help the application succeed.

Methods user experience testing includes:

- Surveys: Surveys involve collecting data from a large group of users through questionnaires or online forms. This method is useful for gathering quantitative data about user preferences, satisfaction, and opinions about a product.
- Interviews: Interviews involve one-on-one conversations with users to gain in-depth qualitative insights about their experiences with a product. This method is useful for uncovering the reasoning behind user behavior and preferences.
- Task analysis: Task analysis involves breaking down a user's interactions with a product into smaller tasks to identify any usability issues or areas for improvement. This method is useful for evaluating the overall usability of a product.
- Click tracking: Click tracking involves analyzing user clicks and move-

ments on a website or app to identify any usability issues or areas for improvement. This method is useful for understanding user behavior and identifying areas of a product that may need further optimization.

- Heat mapping: Heat mapping involves using a visual representation to show where users interact the most with a product. This method is useful for identifying where users spend the most time and which areas of a product may need further attention.

2.1.5.1 User-Centered Design (UCD)

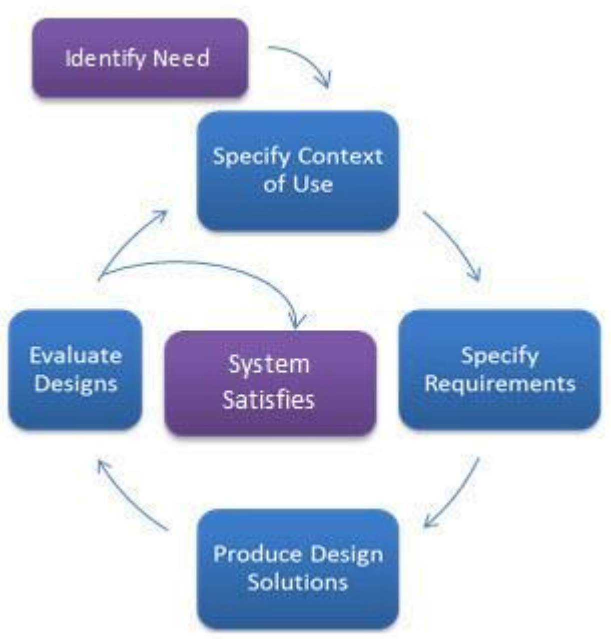


Figure 2.1 User-Centered Design Basics image

User-Centered Design (UCD) is an iterative design methodology that focuses on comprehending and accommodating end users' wants, objectives, and constraints. User research, prototyping, usability testing, and iteration are common components of the UCD process. UCD aims to provide products that not only meet the needs and expectations of the target customers but are useful [7].

The methods to be able to do UCD are:

- Describe the context of use by identifying the users of the product, the purposes for which they will use it, and the circumstances in which they will use it.

- Determine any business needs or user objectives that must be satisfied for the product to succeed.
- Develop design solutions: This step of the process can be carried out in stages, progressing from a basic idea to a finished design.
- Design evaluation is as essential to excellent software development as quality testing is, ideally through usability testing with real users.

2.1.5.2 Human Computer Interaction (HCI)

HCI (human-computer interaction) is the study of how people and technology interact. The design, assessment, and implementation of interactive computing systems for human use are all part of this interdisciplinary discipline. HCI focuses on the creation of user interfaces, system usability, and the investigation of human-technology interaction. By creating systems that are simple, effective, and enjoyable to use, HCI aims to improve how humans engage with technology [8].

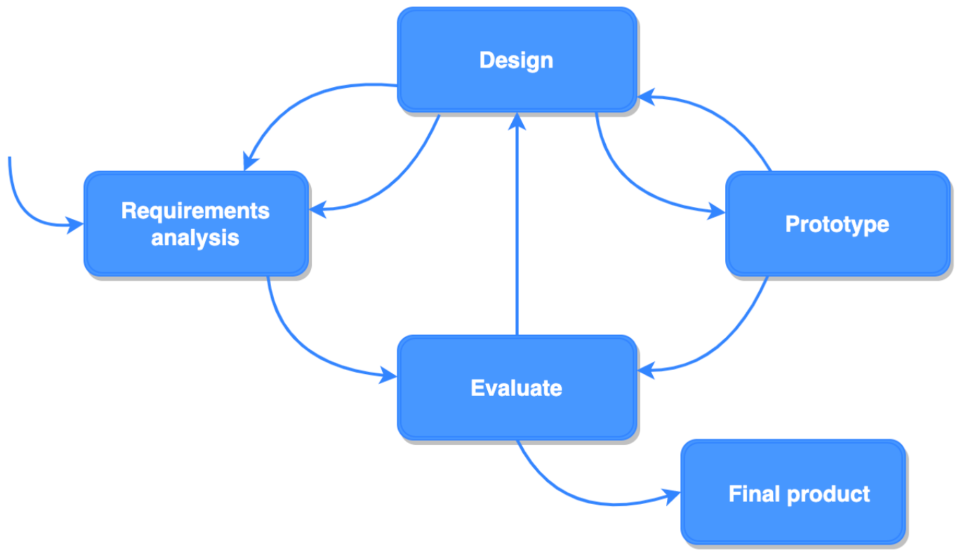


Figure 2.2 Introduction to Human-Computer Interaction & Design

The process of HCI are divided into 3 process (exclude: Final Product):

- User research and requirements analysis: Prior to creating, it is important to understand our target audience and the issues that the author can help them with through the system. Whatever technology, the author wish to create will have a certain context, which ought to direct the design procedure.

- Designs and prototyping: Designing and prototyping are steps in the second step. In essence, the author are translating the user needs into potential answers. This will include both physical design and conceptual design (how will a system operate) (colors, interaction styles). After that, the author prototype their many design concepts.
- Evaluating the designs: After the finished product is constructed, the author must assess its design. To ensure that it complies with specifications and HCI design principles, evaluation should involve both experts and actual users. From there, the author may adjust the design and make changes for the better.

2.1.6 UI Testing

UI testing is a vital aspect of software quality assurance that focuses on evaluating the user interface's correctness, functionality, and visual aspects. Understanding the principles and techniques of UI testing is crucial for ensuring the mobile attendance system's usability and effectiveness [9].

UI testing involves assessing the user interface elements to ensure they adhere to design specifications and provide a seamless user experience. It encompasses principles such as correctness, responsiveness, visual consistency, error handling, and accessibility.

To conduct UI testing effectively, various techniques can be employed, including functional testing, usability testing and visual testing. These techniques help evaluate different facets of the user interface, ensuring its functionality, ease of use, visual appeal, cross-platform compatibility, and localization.

2.1.7 Usability Testing

Usability testing is a fundamental component of the QA role in developing mobile attendance applications with face recognition. Theoretical frameworks such as User Testing and Cognitive Walkthrough can assist in conducting effective usability testing to identify and address usability issues.

Usability testing is the practice of testing how easy a design is to use with a group of representative users. It usually involves observing users as they attempt to complete tasks and can be done for different types of designs. It is often conducted repeatedly, from early development until a product's release [10].

Usability is important because it determines whether users can engage with a system or product in a way that allows them to achieve their goals successfully, efficiently, and satisfactorily. A usable product can increase productivity, decrease errors and user annoyance, boost user satisfaction, and eventually raise adoption.

Methods that can be used will be usability testing because on the mobile attendance app with face recognition, actual users are requested to carry out particular tasks while their interactions are watched and recorded. This offers information on how simple or complex the app is to use and what tweaks can be made to improve the user experience.

2.1.8 Integration Testing

Software testing called "integration testing" involves combining and evaluating various software modules or components all at once to make sure they work together as intended [11].

To make sure that all the system's different components, including the face recognition technology, function as expected, integration testing is crucial for mobile attendance applications with face recognition. A user interface, a database, and face recognition technologies are just a few of the complicated software systems that are frequently used in mobile attendance applications. Integration testing is a useful tool for ensuring that the system is operating appropriately and that all these components are correctly talking with one another. Without integration testing, problems like data inconsistencies or communication problems among several components could go undetected and result in poor user experience. To guarantee the overall dependability and efficacy of mobile attendance applications with face recognition, thorough integration testing is essential.

2.1.9 Beta Testing

Beta testing, also known as user acceptance testing, focuses on evaluating the software application's performance and usability in a real-world environment. It involves gathering feedback from a group of end-users who actively engage with the application before its official release [12]. This testing approach allows for the identification of potential issues, bugs, or usability concerns that users may encounter during their interaction with the application.

By conducting beta testing, the development team can obtain valuable insights into how the mobile attendance application with facial recognition performs under normal usage conditions. It provides an opportunity to assess the system's overall performance, usability, and user satisfaction, helping to validate that the application meets the intended user requirements. The feedback collected during beta testing serves as valuable input for further refining the application and addressing any identified issues or areas for improvement.

The inclusion of beta testing in the software development process ensures that the mobile attendance application is thoroughly evaluated in a real-world context, leading to a more robust and user-friendly final product. It enhances the overall quality and reliability of the application, contributing to a smooth and satisfying user experience.

2.1.9.1 User Acceptance Testing

UAT adheres to essential principles, including user involvement, comprehensive test planning, test execution, and feedback analysis. Among the various techniques used in UAT, prototype testing stands out as a notable approach. This technique involves providing potential users with a representative version of the application, known as a prototype, to assess its usability and acceptance.

2.1.9.2 Prototype Testing

Prototype testing enables users to interact with a working model of the application that closely resembles the final product. Users are assigned specific tasks and scenarios that simulate real-world usage. They provide feedback on the prototype's functionality, ease of use, and overall user experience. Prototype testing allows for early identification of usability issues and serves as an opportunity to incorporate user feedback into subsequent iterations of the software.

By embracing prototype testing within UAT, researchers and practitioners can obtain valuable insights into the effectiveness of the application's design and functionality. This technique provides an opportunity to gather user feedback at an early stage, facilitating iterative improvements and ensuring that the final product meets user expectations.

Understanding the principles and techniques of UAT, including prototype testing, lays the groundwork for effective evaluation of software systems' usability and acceptance. This theoretical foundation serves as a framework for the subsequent chapters, where UAT will be implemented and evaluated within the research study.

2.1.10 Maze Testing

A technique used by QA to evaluate the usability and effectiveness of the mobile attendance system application is maze testing. It entails building a controlled environment within the application's interface that resembles a maze-like structure that users must navigate through to complete tasks and interact with different functionalities. Through maze testing, QA can assess user behavior, navigational patterns, and the application's general usability.

2.1.11 Maze Testing for Usability Assessment

A technique used by QA to evaluate the usability and effectiveness of the mobile attendance system application is maze testing. It entails building a controlled environment within the application's interface that resembles a maze-like structure that users must navigate through to complete tasks and interact with different functionalities. Through maze testing, QA can assess user behavior, navigational patterns, and the application's general usability.

2.1.12 Test Scenarios and Tasks

It is crucial to specify specific test scenarios and activities that are in line with the system's functionality and user goals in order to carry out maze testing for the mobile attendance system application. When using the application, users may encounter real-world situations like recording attendance, accessing reports, or managing user profiles. These scenarios should accurately portray those situations. These goals should be reflected in the accompanying maze tasks to make sure that the testing process covers important facets of the application's usability and functionality.

Maze testing enables QA to assess how users interact with the mobile attendance system application, expose any usability issues, and improve the user experience by creating pertinent test scenarios and objectives.

2.1.13 Test Participants and Data Collection

The target audience of the application should be represented in the maze testing for the mobile attendance system project. This may apply to staff members, students, or any users of the application for tracking attendance. To provide a complete picture of usability and user experience, it's crucial to involve people with different levels of exposure to mobile attendance systems.

Data gathering methods like observation and post-task questionnaires should be used during maze testing. Participants' navigational routes, completion times, mistakes, and any feedback or remarks should all be recorded by observers. This approach of gathering data aids QA in gaining quantitative and qualitative understandings of user interactions, providing a thorough assessment of the mobile attendance system application.

2.1.14 Metrics and Analysis

In evaluating the performance and usability of the mobile attendance system application, several metrics can be employed. These metrics include:

- Success Rate: The percentage of participants who successfully complete attendance-related tasks within the maze.
- Completion Time: The average time it took participants to find their way

out of the maze and complete attendance-related tasks.

- Errors: The amount and kind of mistakes made by participants, such as trouble using certain features or running into problems when taking attendance.
- Path Efficiency: Evaluating participants' navigational paths for efficiency and directness while taking criteria like usability, intuitiveness, and instruction clarity into account.

These metrics can be analyzed by QA to find patterns and trends that point out areas where the mobile attendance system application needs work. This study helps iterative design processes and evidence-based decision-making to improve usability, functionality, and user satisfaction.

2.1.15 Benchmarking

In the maze testing process for the mobile attendance system application, benchmarking is essential. While specialized criteria for testing attendance systems in a maze might not exist, more general usability benchmarks or industry standards for mobile applications can be utilized as a guide. When compared to established criteria, these benchmarks offer useful information about how the mobile attendance system application performs in terms of usability, efficiency, and user satisfaction.

Industry standards for mobile applications would include recommendations and best practices for usability, effectiveness, and overall user experience in the context of the mobile attendance system application. In addition to other requirements that are seen to be crucial for producing a high-quality mobile application, these standards may include design concepts, interface rules, performance benchmarks, and accessibility requirements.

QA can find the application's strong points and its opportunities for improvement by comparing it to industry standards. With the help of this knowledge, the development team can make intelligent choices and enhance the user experience of the mobile attendance system application during the iterative design phase.

CHAPTER 3

PROBLEM ANALYSIS

This chapter will discuss the problem statement for this project and the suggested method to solve the problem will be discussed in this chapter. There will also be a review of prior studies in this field.

3.1 Problem Statement

Any software application's usability and user experience are important aspects. Poor usability and user experience in the context of mobile attendance systems with facial recognition may lower user interest and adoption rates. To ensure efficient and successful attendance monitoring, these systems must be simple to use. Nonetheless, it can be difficult to design and construct a system that is user-friendly for everyone given the growing complexity of technology and the wide range of user backgrounds. The system's usability and user experience may also be impacted by problems like technological faults and security worries.

The usability and user experience issues in mobile attendance systems with facial recognition must therefore be recognized and resolved. This chapter explores usability and user experience problems that can prevent the adoption of such systems and suggests fixes to improve both in general. The author will identify the typical usability and user experience difficulties faced by users and developers through a thorough analysis of the available literature and case studies. The author will also examine how bad usability and user experience affect adoption rates and make suggestions for how to build and create a user-friendly face-recognition mobile attendance system.

3.2 Related Works

[13] In the context of mobile attendance systems with facial recognition, the conventional attendance marking method has been found to be vulnerable, inaccurate, and time-consuming, particularly in large classrooms. It fails to effectively identify absentees and proxy attendees. To address these challenges, a proposed solution is a web-based mobile attendance system that incorporates existing mobile devices with a camera and a facial recognition system. This system aims to automate and streamline attendance monitoring in classrooms with minimal implementation requirements. Initial testing of the system prototype received positive responses from volunteers, indicating its potential to replace the conventional attendance marking method.

A post-task questionnaire was conducted to analyze the respondents' perception of the Mobile Attendance System with Facial Recognition (MAS-FR). The questionnaire focused on measuring the system's usefulness, ease of use, and overall user satisfaction. The results indicated that the majority of respondents viewed the MAS-FR prototype as easy to understand and use, with positive impressions of its website interface. Users strongly agreed on the application's usefulness, particularly for supervisors, and expressed support for its development as a measure to prevent signature fraud. Some suggestions for improvement were also provided, such as incorporating the system within an application instead of relying solely on a website. Additionally, enhancements in user-friendliness and the provision of user guidance were recommended based on respondents who required assistance from the developer. This evaluation highlighted both the strengths and weaknesses of the system, providing valuable insights for further refinement.

The author can relate to the insights provided by post-task questionnaires during usability testing of the mobile attendance system prototype. Post-task questionnaires are a common tool used in usability testing to gather feedback from users and evaluate their perception of the system's usefulness, ease of use, and overall satisfaction. By administering such questionnaires, valuable insights can be obtained regarding the system's performance, user experience, and areas that require improvement.

In the context of the mobile attendance system with facial recognition, the post-task questionnaire allowed for the collection of user feedback and perceptions regarding the MAS-FR prototype. The questionnaire provided an opportunity to assess the system's usability, understand user perspectives, and identify areas where the system excelled or required enhancements. Analyzing the responses from the questionnaire helps in understanding the system's strengths and weaknesses from the user's point of view, enabling me to contribute to the improvement of the system's usability and user experience.

The inclusion of a post-task questionnaire during the usability testing phase reflects a QA-driven approach to ensuring the effectiveness and user-friendliness of the mobile attendance system. It allows for the identification of user preferences, pain points, and suggestions for improvement, ultimately guiding the iterative development process and contributing to the enhancement of the system's usability and user experience.

The author also can relate to the challenges of finding a comprehensive mobile app testing tool that addresses all desired quality factors. While many testing tools focus on usability, correctness, and robustness, there is a limited number of tools that support incremental development, post-deployment maintainability, flexibility, and other critical quality attributes.

[14] The increasing trend of mobile application usage necessitates the delivery of high-quality apps, similar to web or desktop applications. Simplifying the quality assurance process for mobile devices is achieved with automated testing tools. However, the evaluation and comparison of these tools based on the specific quality factors they enhance in the tested apps are still lacking. This research study aims to bridge that gap by evaluating different testing tools and identifying the quality factors they contribute to achieving in the apps under

test.

The findings of this study are valuable for both practitioners and researchers. Practitioners can benefit from the identification of specific tools that assist in assuring desired quality factors in their tested apps. Researchers can leverage the study's findings to propose solutions or enhance existing tools to address a broader range of critical quality attributes. The study reveals that while automated testing is strong in usability, correctness, and robustness, there is room for improvement in areas such as testability, performance, extensibility, maintainability, scalability, and platform compatibility.

The author resonates with the challenges of selecting suitable testing tools and emphasizes the importance of considering a broad range of quality factors in mobile app testing. It highlights the need for continuous improvement in testing practices to ensure the overall quality and success of mobile applications.

3.2.1 Usability and User Experience of Mobile Attendance Applications

Usability and user experience are essential aspects of any software application, including mobile attendance applications with face recognition. Poor usability and user experience can lead to decreased user interest and adoption rates (User Experience Is Just as Important as Technology, n.d.). As a QA, it is important to ensure that the mobile attendance application is user-friendly and easy to use.

3.2.2 Evaluation Methods for Usability and User Experience

The usability and user experience of mobile attendance applications can be assessed using several different techniques. User testing is a popular strategy that involves watching users as they use the program to complete tasks and obtaining feedback on their experience (What Is User Testing? Definition - Omniconvert, n.d.). Doing user testing as a QA can give you useful information about the application's potential weak points.

3.2.3 Design Improvements for Usability and User Experience

As a QA, it is important to work closely with the development team to implement design improvements that enhance the usability and user experience of the mobile attendance application (Samsukha, 2021). For example, simplifying the user interface, providing clear instructions, and ensuring consistency throughout the application can improve user satisfaction and adoption rates.

3.3 Proposed Solution

As a Quality Assurance (QA), the author will concentrate on making sure the system is simple to use and intuitive for users when developing the proposed solution for the mobile attendance system with face recognition. Following strategies will be used to accomplish this goal:

- A comprehensive set of unit tests, integration tests, and beta tests will be used to guide the system's development. By doing so, it will be possible to make sure that the system works as planned and that any problems are discovered early in the development process.
- Testing for Usability: To make sure the system is simple and straightforward for users, it will go through rigorous usability testing. Testing will be done with a sample user group to find any problems or areas of difficulty.
- Constant Improvement: User feedback and test results will be used to guide future system updates and improvements. This process will be continuing.
- Documentation: Users will be given thorough documentation to ensure that they can use the system to its full potential. To help users complete the attendance-taking process, this will feature simple instructions, visual

aids, and prompts.

In general, the proposed solution for the face-recognition mobile attendance system strives to give users a system that is simple to use and intuitive, while also making sure that the system works as intended and any difficulties are discovered early in the development process. The system will continue to be user-friendly and efficient throughout time thanks to the emphasis on testing, usability, and continual improvement.

3.4 Testing Approach

The author will use a variety of testing techniques, such as unit testing, integration testing, usability testing, and beta testing, to assure the quality of our system. Each system component will be tested using unit testing, while the system's interactions between various components will be tested using integration testing. The system will be put through beta testing as well as usability testing to see how user-friendly it is.

I've examined the issue that the mobile attendance system with facial recognition is meant to solve in this chapter. The author has researched the relevant works in face-recognition mobile attendance systems, given the author suggested solution, and discussed the testing strategy that the author would employ to assure its quality. The author will go over the specifics of the suggested solution's implementation in the following chapter.

CHAPTER 4

SOLUTION DESIGN

This chapter describes the facial recognition mobile attendance application's solution design is covered in this chapter. The program will be created to correspond to the specifications listed in Chapter 3. Ensuring the application is dependable, secure, and simple to use will be the author's responsibility as the Quality Assurance (QA) during the solution design phase.

4.1 Testing Strategy

In the development of the mobile application attendance system with face recognition and edge computing, a comprehensive testing strategy is essential to ensure the quality, reliability, and usability of the system. This section outlines the testing approach and methodologies employed to achieve these objectives.

4.1.1 Testing Methodologies

In the context of developing a mobile attendance application with face recognition, a systematic and well-defined methodology is essential for ensuring the quality, reliability, and effectiveness of the application. This section presents the methodology employed by the Quality Assurance (QA) role in the project to achieve these objectives.

4.2 UI Testing

UI testing focuses on validating the correctness, responsiveness, and visual aspects of the user interface. As a QA, the author will select appropriate UI testing techniques and tools, such as manual testing, automated testing frameworks, and visual validation tools, to ensure a seamless user experience.

For UI testing, the author will employ a combination of manual testing and automated testing frameworks. Manual testing involves manually interacting with the user interface to validate its functionality, responsiveness, and design consistency. In addition, the author will utilize the Espresso framework, an Android testing framework specifically designed for UI testing. Espresso provides APIs for interacting with and asserting the behavior of UI components, allowing the author to automate UI tests efficiently.

By incorporating the Espresso framework into UI testing strategy, the author can streamline the testing process, increase test coverage, and ensure that the user interface meets the desired functional and visual requirements.

4.3 Usability Testing

Usability testing evaluates the ease of use, intuitiveness, and overall user experience of the mobile application. The author's solution design includes the selection of usability testing techniques, such as user interviews, surveys, or observations, to gather feedback from end-users and enhance the application's usability.

Usability testing aims to assess how well the application meets user requirements, how easily users can navigate through the application, and how intuitive and efficient the user interface is. By gathering feedback from end-users, the author gains valuable insights into their perspectives, preferences, and pain points.

To conduct usability testing, the author will employ techniques such as user interviews, where the author will directly interact with users to understand their experiences and gather feedback. Surveys and questionnaires can also be utilized to collect quantitative data and assess user satisfaction. Additionally, the author may observe users interacting with the application to gain insights into their behaviors and identify usability issues.

In this process, the author will incorporate Maze, a usability testing platform, to create and run tests, collect user feedback, and analyze results. Maze allows me to simulate user interactions, create test scenarios, and analyze user behavior within the application. By leveraging Maze, I can obtain valuable usability insights and make informed decisions to improve the user experience.

To measure the usability of the mobile attendance application, the author will utilize the Maze usability testing platform. Maze offers various usability testing features and provides insights into user behavior and interactions. One of the key metrics used by Maze to evaluate usability is the System Usability Scale (SUS).

The System Usability Scale (SUS) is a standardized questionnaire that assesses the perceived usability of a system or application. It consists of a set of statements to which users respond, indicating their level of agreement or disagreement. The SUS score is calculated based on the users' responses and provides a quantitative measure of usability.

Additionally, Maze also utilizes the Maze Usability Score (MUS) framework, which includes three key metrics:

- **Scenario Completion Usability Score (SCUS): SCUS** measures the users' ability to successfully complete specific tasks or scenarios within the mobile attendance application. It evaluates the effectiveness of the application in enabling users to accomplish their goals and tasks efficiently.

- **Maze Interactions Usability Score (MIUS): MIUS** assesses the ease of use and intuitiveness of the application by analyzing user interactions, such as clicks, taps, swipes, and other actions. It provides insights into how users navigate through the application and interact with its various features and functionalities.
- **Maze Accessibility Usability Score (MAUS): MAUS** evaluates the accessibility and inclusivity of the mobile attendance application. It considers factors such as color contrast, font size, and other design elements that contribute to the application's accessibility for users with diverse needs.

By incorporating the Maze usability testing platform and the **SUS**, **SCUS**, **MIUS**, and **MAUS** metrics, the author can gather valuable insights into the application's usability and make data-driven improvements to enhance the user experience. Through user interviews, surveys, and observations, the author will gather feedback from end-users and incorporate it into the testing process, ensuring that the mobile attendance application meets high usability standards and provides a seamless and intuitive user experience.

Regarding the calculation of the usability score, it may be more straightforward to use a simplified approach that aligns with the report's methodology. For instance, the author can calculate the score as (100 - (0.5 * miss click rate) - (average time according to a specific range)). This method provides a direct and easy-to-understand measure of usability without involving additional calculations based on **SCUS**, **MIUS**, and **MAUS**. By adopting this approach, the author can focus on the key factors of miss click rate and average time, ensuring a more streamlined assessment of the application's usability. Through usability testing, the author ensures that the mobile attendance application is user-friendly, intuitive, and aligns with the needs and expectations of the end-users. It helps identify areas of improvement, refine the user interface, and deliver a highly usable and satisfying application.

4.4 Unit Testing

Unit testing is performed to ensure the reliability and correctness of individual system components. As part of solution design, the author will outline the unit testing approach, including the use of unit testing frameworks and tools.

For unit testing, the author will utilize the **JUnit** framework, which is commonly used for writing tests in Java and Kotlin. **JUnit** provides a robust and flexible testing framework that allows me to define test cases, execute tests, and verify the expected behavior of individual functions, modules, or classes.

In addition to **JUnit**, the author will leverage the **Mockito** framework, a mocking library for **Kotlin**. **Mockito** enables me to create mock objects and stub dependencies, facilitating the isolation of components during unit testing. By using **Mockito**, the author can focus on testing the specific functionality of individual components without being hindered by dependencies.

By combining **JUnit** and **Mockito**, the author can effectively perform unit testing, identify defects or issues at the component level, and ensure the reliability and correctness of the system's individual units.

4.5 Integration Testing

Integration testing verifies the proper functioning and data flow between different system components. The author will explain the integration testing strategy that will be employ, which includes testing the interactions between modules and ensuring seamless integration.

To facilitate integration testing, the author will utilize appropriate tools and techniques. While there are no specific integration testing frameworks mentioned in the previous sections, the author will rely on the selected unit testing framework, **JUnit**, to support integration testing as well. **JUnit** can be utilized for both unit testing and integration testing purposes, allowing the author to test the interactions between different components and validate their integration within the system.

The author will employ techniques such as test data generation, test harnesses, and simulators to simulate real-world scenarios and test the integration points between system modules. These techniques, coupled with the capabilities of **JUnit**, will help identify and resolve any integration issues, ensuring the smooth functioning and data consistency of the system.

The solution aims to ensure the correctness and reliability of the user creation functionality in the mobile attendance system. Integration testing plays a crucial role in verifying the proper functioning and data flow between different system components. In this solution design, a comprehensive integration testing strategy will be employed to ensure seamless integration and validate the interactions between modules.

4.5.1 Framework:

The solution utilizes the following frameworks and technologies:

4.5.1.1 Firebase Firestore:

Firebase Firestore is a **NoSQL** document-based database provided by Google Firebase. It serves as the storage and synchronization mechanism for the mobile attendance system, enabling efficient data handling and real-time updates.

4.5.1.2Kotlin Programming Language:

The solution is implemented using the **Kotlin** programming language, a modern language compatible with the Java Virtual Machine (**JVM**). Kotlin offers concise and expressive syntax, enhancing the efficiency and readability of the codebase.

4.5.1.3 Coroutines:

Coroutines, a feature of the **Kotlin** programming language, will be utilized for asynchronous programming. They provide a way to write asynchronous code sequentially, simplifying the handling of tasks such as network requests. By utilizing **Coroutines**, the solution achieves efficient and structured concurrency.

The integration testing strategy focuses on testing the interactions between different system components to ensure seamless integration. Techniques such as test data generation, test harnesses, and simulators will be employed to simulate real-world scenarios and test the integration points between system modules.

The selected unit testing framework, **JUnit**, will also support integration testing. **JUnit** allows the testing of interactions between different components, validating their integration within the system. By leveraging the capabilities of **JUnit** and the employed techniques, any integration issues can be identified and resolved, ensuring the smooth functioning and data consistency of the system.

By applying effective integration testing methodologies and utilizing the appropriate tools and techniques, the solution design aims to validate the interactions between system components, detect integration issues, and ensure seamless integration within the mobile attendance system.

4.6 Beta Testing

Technical beta testing aims to validate the system's behavior and assess its overall readiness for release. It allows for the identification of potential issues, bugs, and usability concerns specific to the system's technical aspects. This testing approach provides valuable insights into the system's performance, security, compatibility, and overall technical robustness.

While there are various methods of beta testing, such as traditional beta testing, public beta testing, focused beta testing, and post-release beta testing, the author specifically chose to employ the technical beta testing method for the mobile attendance system. Technical beta testing was deemed most suitable due to its focus on evaluating the system's technical aspects and involving internal beta testers who possess in-depth knowledge of the system and its intricacies.

Technical beta testing involved the following key elements:

- **Involvement of Internal Beta Testers:**

  A select group of internal beta testers, comprising employees from the developing company, actively engaged with the mobile attendance system. This allowed for comprehensive testing and evaluation of the system's technical aspects.

- **Evaluation of Performance and Compatibility:**

  The internal beta testers assessed the system's performance, scalability, security, and compatibility with different devices and platforms. This evaluation provided insights into potential technical issues and areas for improvement.

- **Feedback Collection and Analysis:**

  The feedback provided by the internal beta testers was collected, analyzed, and documented. This feedback served as a valuable resource for identifying technical issues, improving system performance, and refining the overall user experience.

By focusing on technical beta testing, the author aimed to thoroughly evaluate the mobile attendance system from a technical standpoint. This approach ensured that potential technical issues and challenges were identified and resolved prior to the system's official release.

Through technical beta testing, the development team gained valuable insights into the system's technical performance, usability, and user satisfaction. The feedback collected during this testing phase was utilized to refine the system, address identified issues, and optimize its overall technical integrity.

The implementation of technical beta testing, as the chosen method among various beta testing approaches, enabled the development team to enhance the mobile attendance system's technical robustness, reliability, and user experience.

4.6.1 User Acceptance Testing (UAT)

In addition, in beta testing, User Acceptance Testing (UAT) is also tested to get scoring of the application. To evaluate the usability and acceptance of the proposed solution, the author will implement UAT, focusing on the main features of the application. The UAT process will involve gathering feedback and ratings from potential users regarding the ease of use of these specific features.

Participants will be provided with a prototype version of the application and will be asked to perform predefined tasks that simulate real-world scenarios. They will provide feedback and rate the main features on a scale of 1 to 5, indicating the level of ease of use. The collected feedback will be used to assess the acceptance and user-friendliness of the application's main features.

By implementing this scoring system through prototype testing and gathering ratings from users, the author aims to gain valuable insights into the effectiveness of the main features. The collected feedback will inform necessary improvements and enhancements based on the user feedback received.

The UAT technique employed in this study combines prototype testing with a scoring system to evaluate user acceptance and usability. This approach allows for targeted evaluation of the main features and provides valuable data for enhancing the application's user experience.

The UAT results will provide valuable insights into the usability of the solution. By implementing scoring through prototype testing and gathering ratings from users, the author aims to understand the effectiveness of the main features and make necessary improvements based on the feedback received.

CHAPTER 5
IMPLEMENTATION

This chapter presents a comprehensive analysis of the testing results and the development process of the mobile attendance system with facial recognition. The author evaluates the effectiveness of implemented testing strategies in enhancing usability and user experience. Various testing method, including usability testing, UI testing, unit testing, integration testing, and end-to-end testing, are employed to assess the application's performance, functionality, and user interface, revealing valuable insights into strengths, weaknesses, and areas for improvement.

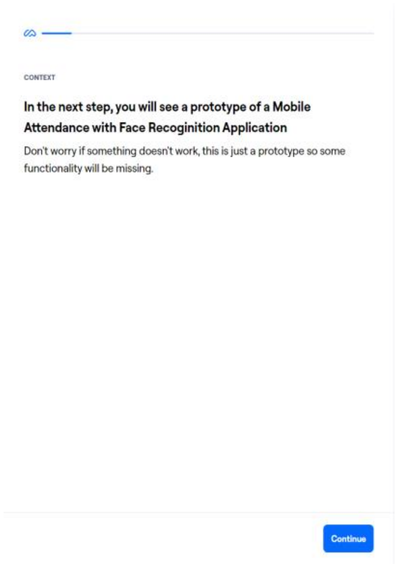
5.1 Usability Testing

Usability testing plays a critical role in evaluating the user experience and ensuring a seamless interaction with the mobile attendance system. In this subchapter, the author delve into the comprehensive usability testing process employed to assess the system's design, functionality, and overall user satisfaction. The Maze usability testing platform was instrumental in conducting these tests, offering valuable insights into user behavior, and providing a robust framework for analysis.

To begin, the design of the usability testing approach was carefully crafted to capture the key aspects of the mobile attendance system. To capture the key aspects of the mobile attendance system, two versions were designed and tested, as depicted in the following images. Version one is in English while 2nd version is in Indonesia.

(Note: Figures are screenshots through Computer, mobile testers will prompt with title and instructions or description then the prototype testing itself.)

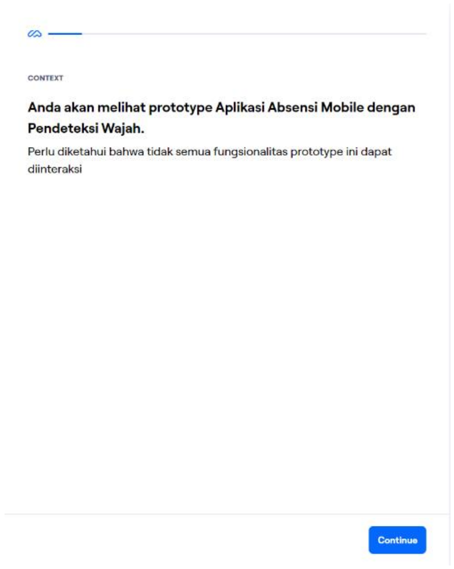
5.1.1 Introduction Page


CONTEXT

In the next step, you will see a prototype of a Mobile Attendance with Face Recoginition Application

Don't worry if something doesn't work, this is just a prototype so some functionality will be missing.

Figure 5.1 Version Introduction


CONTEXT

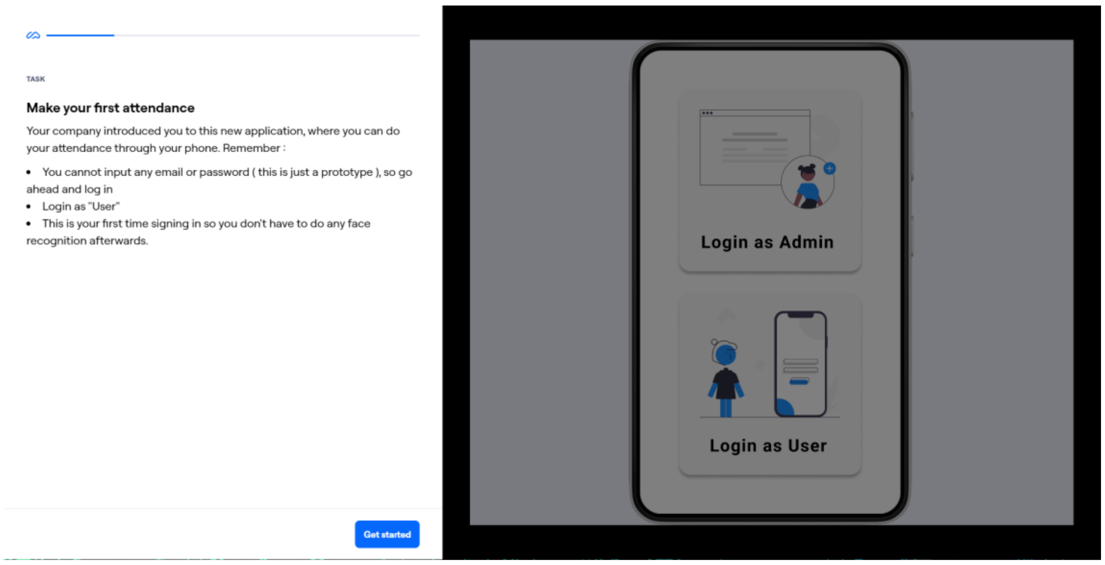
Anda akan melihat prototype Aplikasi Absensi Mobile dengan Pendeteksi Wajah.

Perlu diketahui bahwa tidak semua fungsionalitas prototype ini dapat diinteraksi

Figure 5.2 Version Introduction

This is just the introduction to the testers when they are first trying the author's team prototype through maze.

5.1.2 Tap In Page

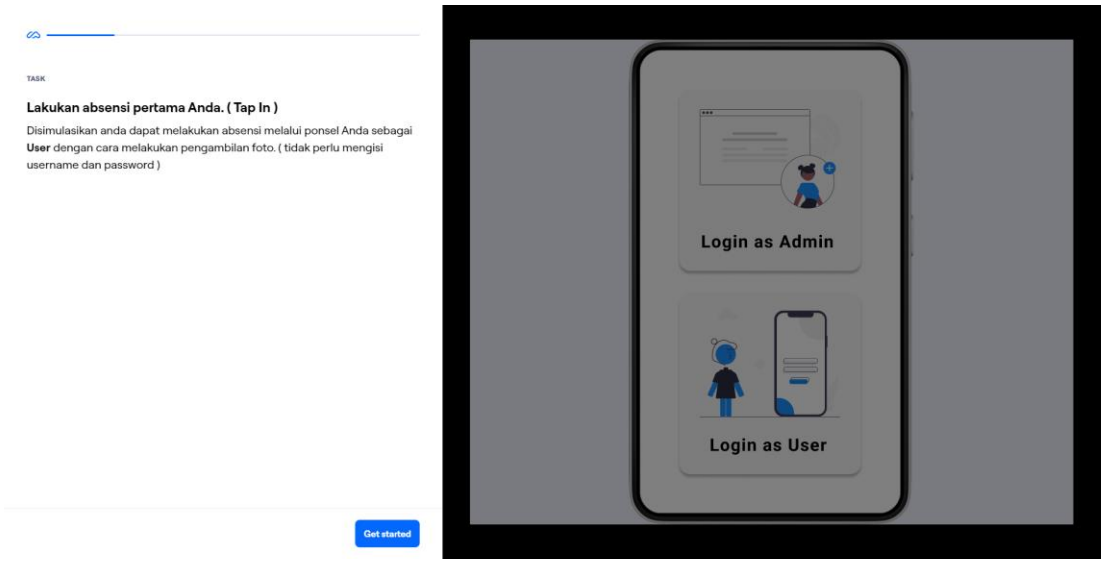

TASK

Make your first attendance

Your company introduced you to this new application, where you can do your attendance through your phone. Remember :

- You cannot input any email or password (this is just a prototype), so go ahead and log in
- Login as "User"
- This is your first time signing in so you don't have to do any face recognition afterwards.

Figure 5.3 Version 1 Tap in


TASK

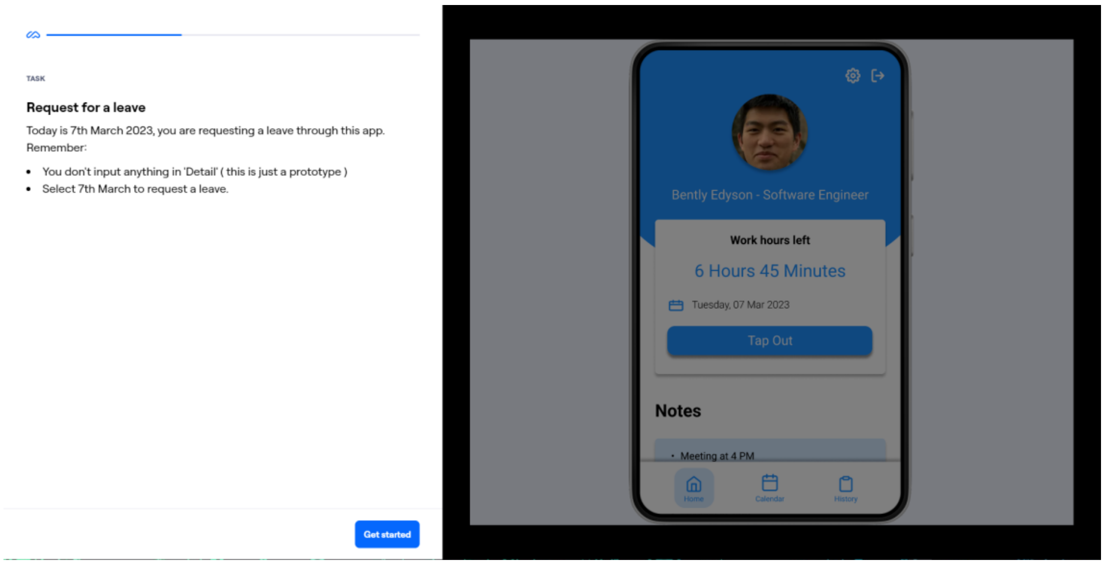
Lakukan absensi pertama Anda. (Tap In)

Disimulasikan anda dapat melakukan absensi melalui ponsel Anda sebagai **User** dengan cara melakukan pengambilan foto. (tidak perlu mengisi username dan password)

Figure 5.4 Version 2 Tap in

In this next page, users will conduct a process called "Tap In". Basically, testers will be guided with the given instruction or description.

5.1.3 Request Leave Page



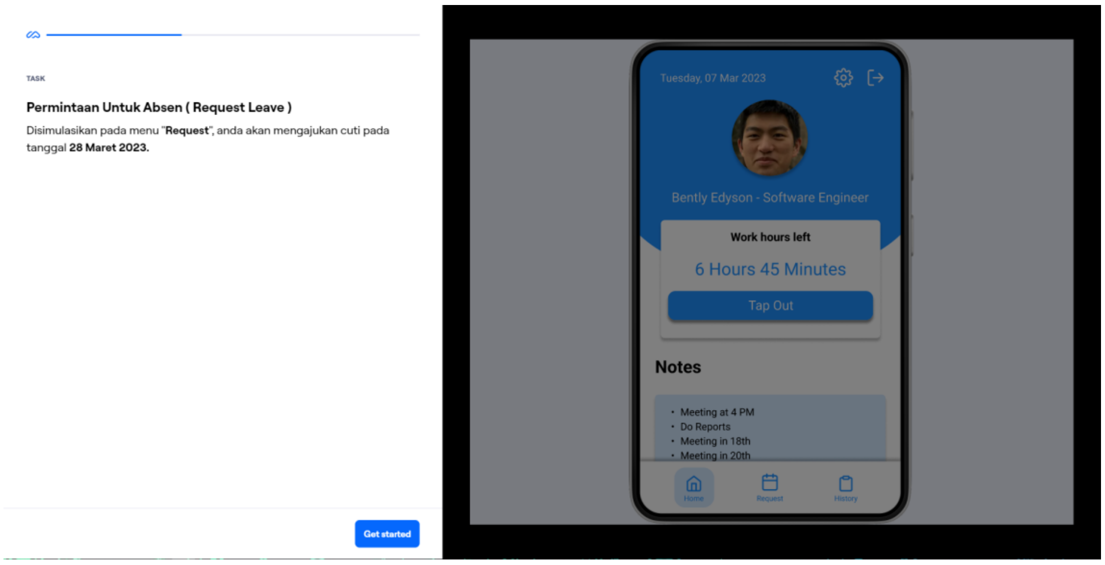
TASK

Request for a leave

Today is 7th March 2023, you are requesting a leave through this app.
Remember:

- You don't input anything in 'Detail' (this is just a prototype)
- Select 7th March to request a leave.

Figure 5.5 Version 1 Request Leave



TASK

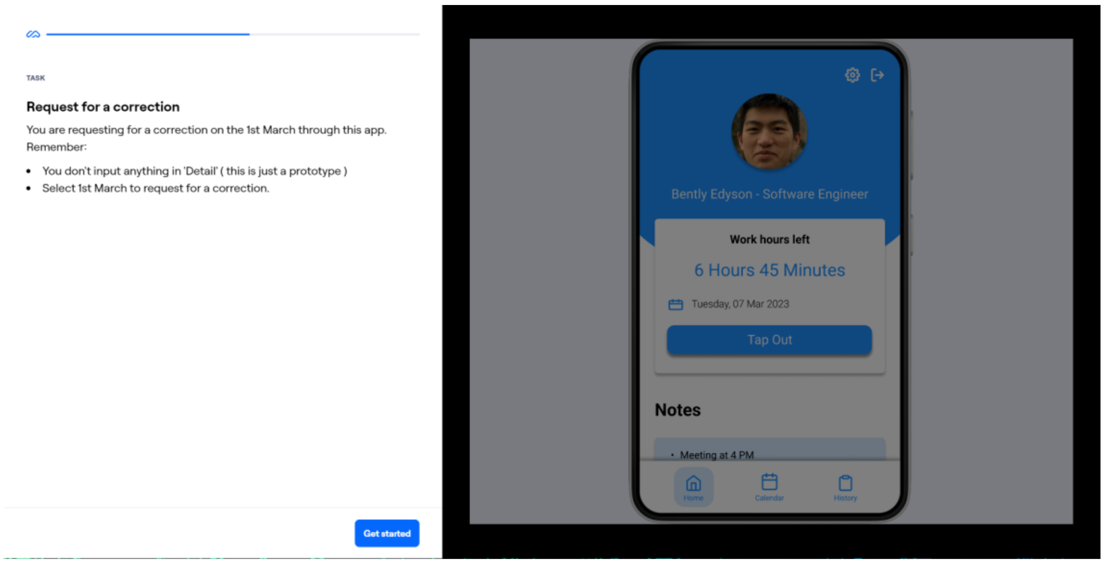
Permintaan Untuk Absen (Request Leave)

Disimulasikan pada menu "**Request**", anda akan mengajukan cuti pada tanggal **28 Maret 2023.**

Figure 5.6 Version 2 Request Leave

In this section, users will be prompted with instructions to request a leave through the prototype. A leave request can be initiated through the app by selecting a date that is at least a certain number of days in the future. Users are only allowed to submit leave requests for dates following the current date.

5.1.4 Request Correction Page

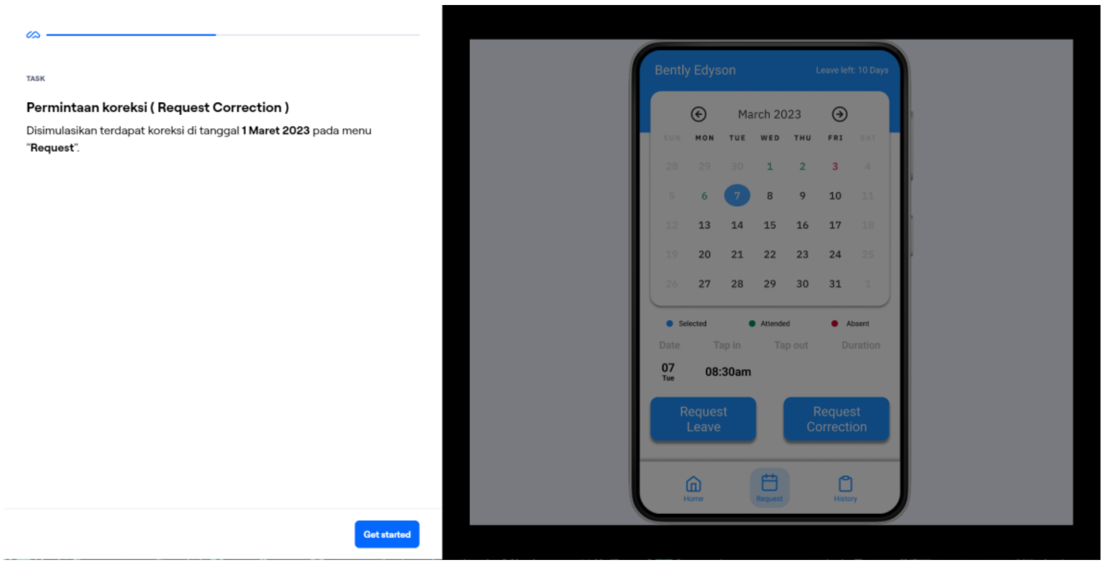

TASK

Request for a correction

You are requesting for a correction on the 1st March through this app.
Remember:

- You don't input anything in 'Detail' (this is just a prototype)
- Select 1st March to request for a correction.

Figure 5.7 Version 1 Request Correction


TASK

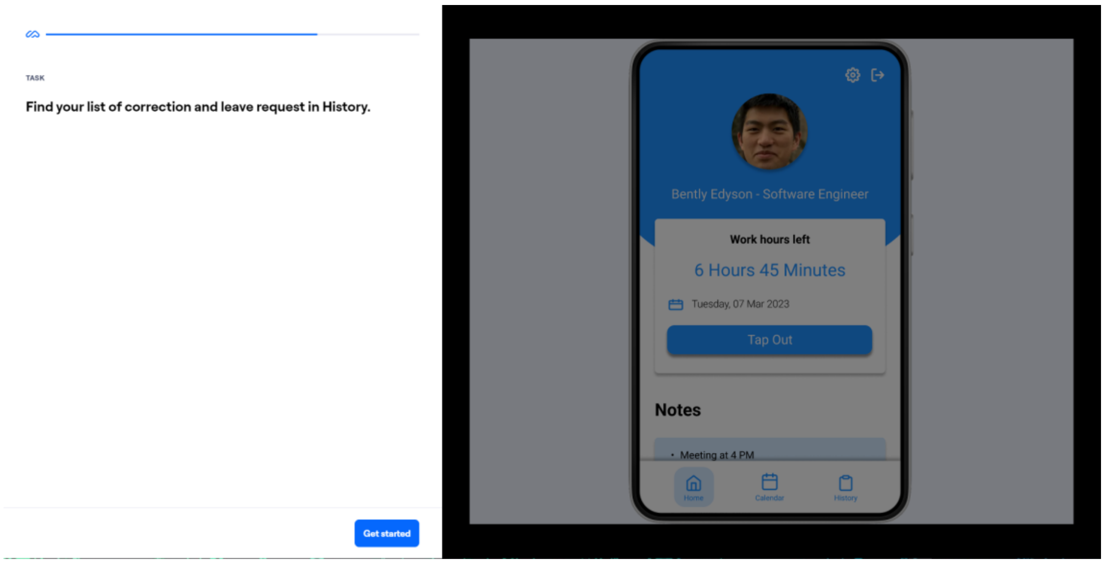
Permintaan koreksi (Request Correction)

Disimulasikan terdapat koreksi di tanggal **1 Maret 2023** pada menu "**Request**".

Figure 5.8 Version 2 Request Correction

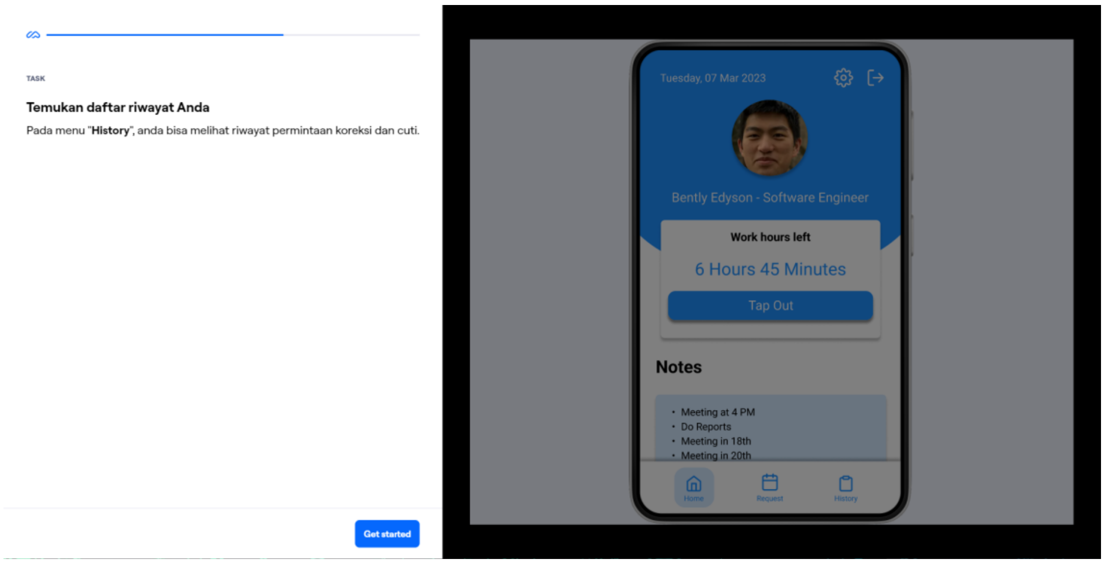
In this page, the users will be prompted with instruction that is like "Request Leave". The request correction functionality allows users to modify their requests, but it is limited to the current day and the days prior to it. A request correction means selecting a specific date according to the users and correcting the time of that day the *"Tap In"* or *"Tap Out.*

5.1.5 History Page


TASK

Find your list of correction and leave request in History.

Figure *5.9 Version 1 History Page*


TASK

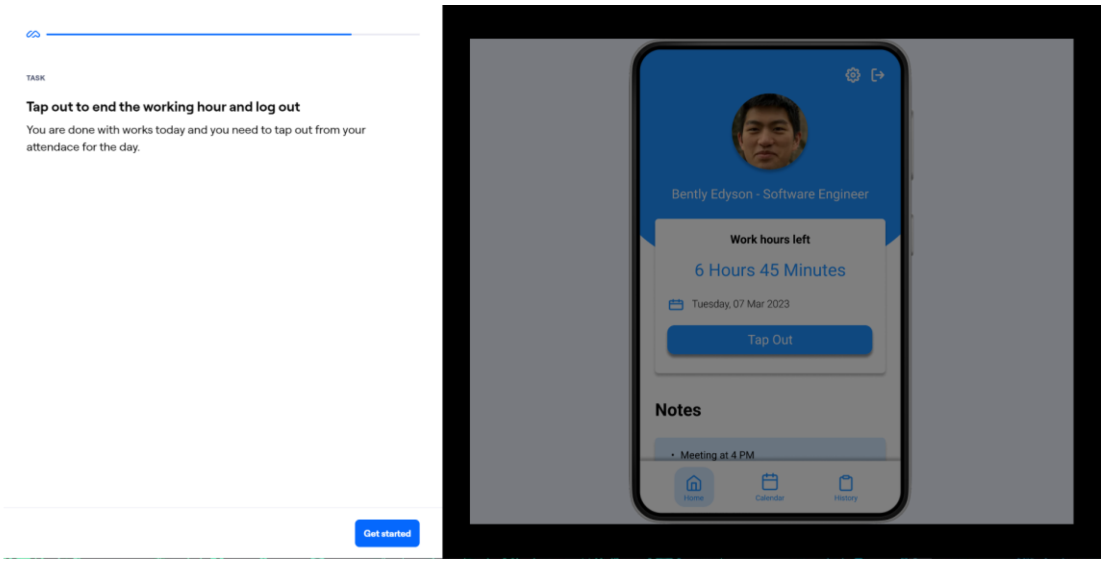
Temukan daftar riwayat Anda

Pada menu "**History**", anda bisa melihat riwayat permintaan koreksi dan cuti.

Figure 5.10 Version 2 History Page

In this section, the users will be given instructions to navigate themselves to "*History Page"*, which will contain the correction and leave requests from users. Each "*Request*" and "*Leave*" request has their own tab.

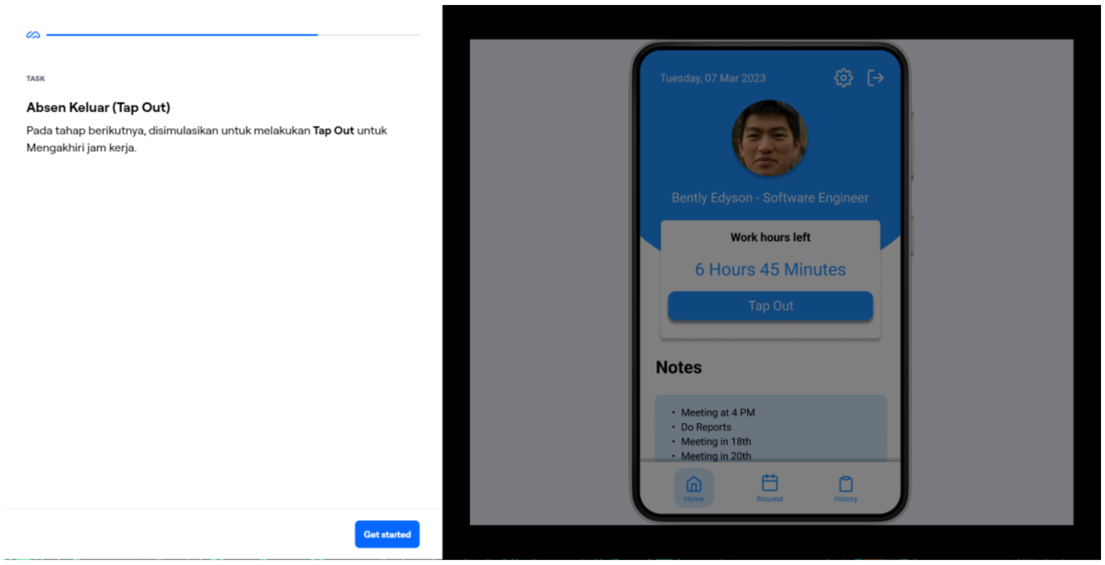
5.1.6 Tap Out Page


TASK

Tap out to end the working hour and log out

You are done with works today and you need to tap out from your attendace for the day.

Figure 5.11 Version 1 Tap Out


TASK

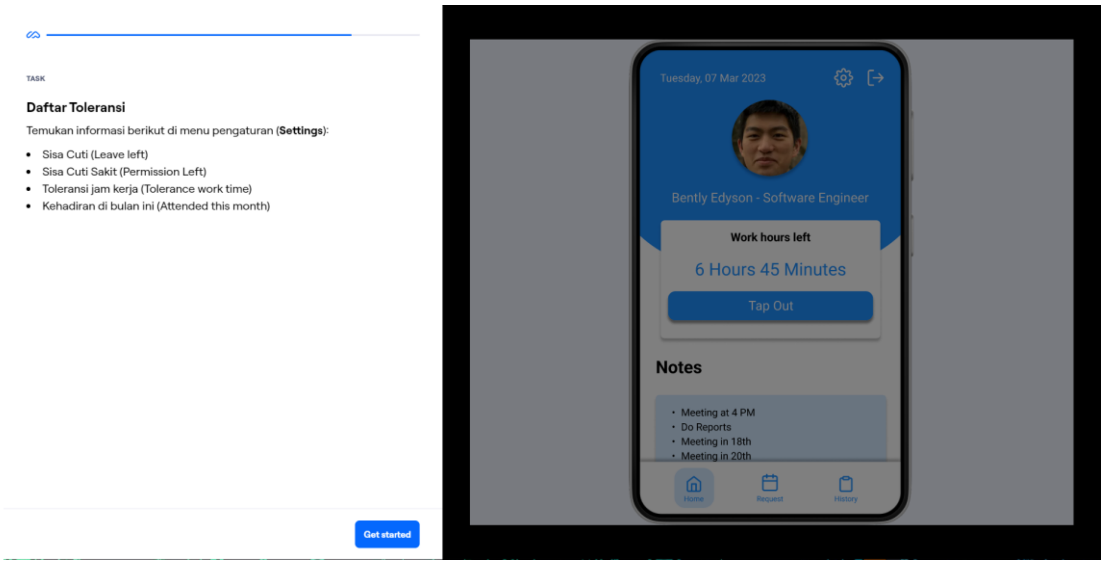
Absen Keluar (Tap Out)

Pada tahap berikutnya, disimulasikan untuk melakukan **Tap Out** untuk Mengakhiri jam kerja.

Figure 5.12 Version 2 Tap Out

In this Section, the users will be prompted with instructions to navigate and try to tap out from the application.

5.1.7 Information Page


TASK

Daftar Toleransi

Temukan informasi berikut di menu pengaturan (**Settings**):

- Sisa Cuti (Leave left)
- Sisa Cuti Sakit (Permission Left)
- Toleransi jam kerja (Tolerance work time)
- Kehadiran di bulan ini (Attended this month)

Figure 5.14 Version 2 Information

In this section, the users will be guided with instructions to navigate to the information page. In this page, users can see information regarding with certain uses. This information includes:

- Leave Left
- Permission Left
- Tolerance Work Time
- Attended This Month

However, in version 1 of maze testing, the author has not included any information tab, because after talking to their potential customers, they decide to add information page to follow their customers’ demands.

5.2 Usability Score

In the evaluation of the mobile attendance system's usability, a crucial aspect is the calculation of the Usability Score. The Usability Score provides a quantitative measure of the system's usability based on various factors and metrics. This subchapter explores the methodology and formulas used to calculate the Usability Score, including the System Completion and Usability Scale (**SCUS**), the Mean

Individual Usability Score (**MIUS**), and the Mean Average Usability Score (**MAUS**). In the following sections, the calculation and interpretation of the **SCUS**, **MIUS**, and **MAUS** will be discussed in detail.

5.2.1 System Completion and Usability Scale (SCUS)

The **SCUS** is a widely accepted and validated questionnaire-based method for assessing the overall usability of a system. It consists of a set of standardized questions that capture users' perceptions of system usability, effectiveness, and efficiency. By administering the **SCUS** questionnaire to a representative sample of users, valuable insights into the system's usability can be obtained.

To calculate the **SCUS**, the responses to each question are assigned a numerical value and then averaged. The resulting average score represents the overall system usability based on user feedback. The **SCUS** provides a valuable qualitative measure that complements the quantitative metrics used in the Usability Score calculation.

Formula

SCUS = **MAX** (0,100 - (**DOR*** **dW**) - (**MCR*mW**) - (**MIN** (10, **MAX** (0, (**AVGD** - 5)/2))))

Which has these variables:

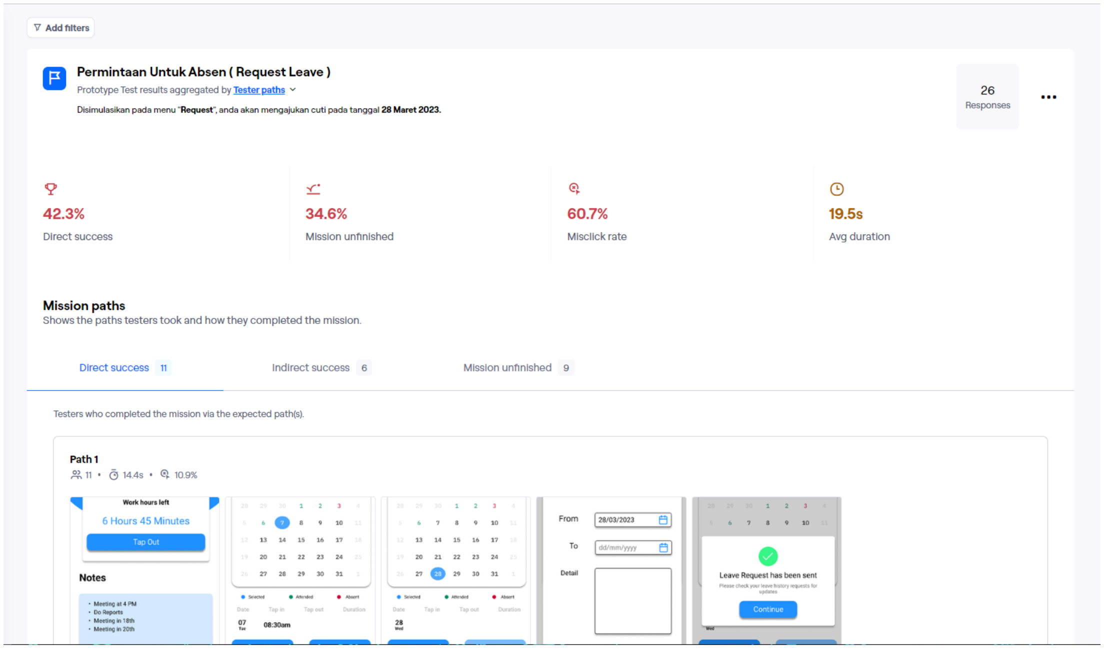
Table 1 5.1 Description of SCUS Formula

Variables	Descriptions
SCUS	Screen Usability Score
DOR	Exit rate (i.e., testers who abandoned the mission or stopped it before reaching the final screen)

dW	equals 1 point for every exit
DOR	For exit rate
MCR	Miss click rate
mW	equals 0.5 points for every miss click
AVGD	for Average Duration in seconds

Example

One of the examples is from page *“Permintaan Untuk Absen (Request Leave)”* *(Note: Maze Calculation does not round up to 1 even there are 0.5, unless above 0.6-0.9)*


Add filters

Permintaan Untuk Absen (Request Leave)
Prototype Test results aggregated by Tester paths
Disimulasikan pada menu "**Request**", anda akan mengajukan cuti pada tanggal **28 Maret 2023.**

26
Responses

42.3%
Direct success

34.6%
Mission unfinished

60.7%
Misclick rate

19.5s
Avg duration

Mission paths
Shows the paths testers took and how they completed the mission.

Direct success 11 | Indirect success 6 | Mission unfinished 9

Testers who completed the mission via the expected path(s).

Path 1
11 • 14.4s • 10.9%

Figure 5.15 Maze Request Leave Report

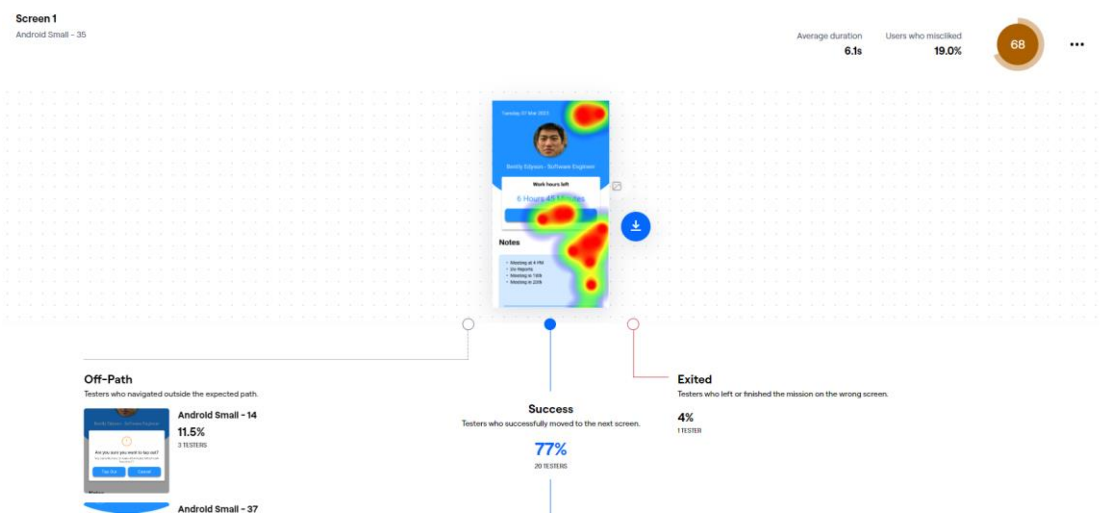
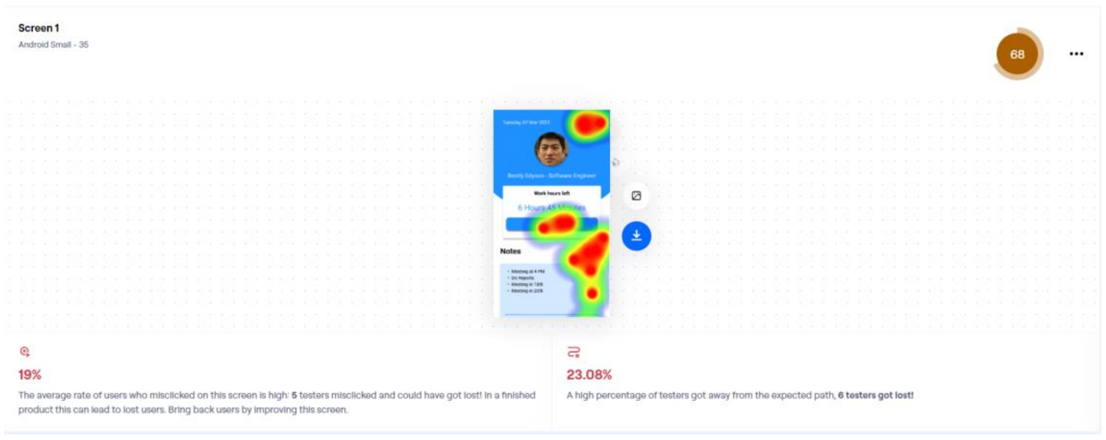


Figure 5.16 Screen 1 Maze Report



The average rate of users who misclicked on this screen is high: **5** testers misclicked and could have got lost! In a finished product this can lead to lost users. Bring back users by improving this screen.

A high percentage of testers got away from the expected path, **6 testers got lost!**

Figure 5.17 Screen 1 Maze Report

The expected 68 is counted as SCUS for screen one, for every screen they have their own SCUS score. However, the author only gives one example for **Screen 1**. The author labels all the variables such as:

- **DOR** = 23
- **Dw** = 1
- **MCR** = 19
- **Mw** = 0.5
- **AVGD** = 6

Substituting variables in the formula:

SCUS = **MAX** (0,100 – (**23*** **1**) – (**19*0.5**) – (**MIN** (10, **MAX** (0, (**6** – 5)/2))))

SCUS = **MAX** (0,100 – 23 – 9 – (**MIN** (10, **MAX** (0, 0.5))))

SCUS = **MAX** (0,100 – 23 – 9 – (**MIN** (10, 0.5))

SCUS = **MAX** (0,100 – 23 – 9 – 0)

SCUS = **MAX** (0, 68)

SCUS = 68

5.2.2 Mean Individual Usability Score (MIUS)

The **MIUS** is a quantitative metric that evaluates the usability of the mobile attendance system by considering factors such as direct success rate, indirect success rate, miss click penalties, and duration penalties. These metrics capture aspects related to user interaction, navigation, and task completion.

The **MIUS** formula combines the direct success rate (**DSR**), indirect success rate (**IDSR**), average miss clicks penalties (**MC_P**), and average duration penalties (**DU_P**). By assigning weights to each metric and summing them accordingly, the **MIUS** provides a numerical score that reflects the system's overall usability performance.

Formula

MIUS = DSR + (**IDSR** / 2) – **avg** (**MC_P**) – **avg**(**DU_P**)

Which has these variables:

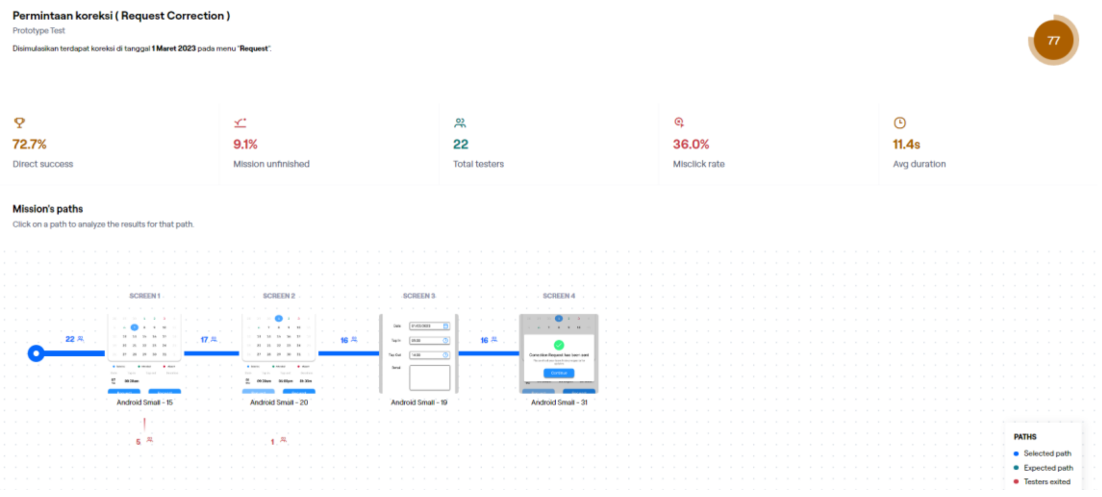
Table 5.2 Description of MIUS Formula

Variables	Descriptions
DSR	Direct Success Rate
IDSR	Indirect Success Rate
Avg	Average
MC_P	Miss clicks Penalty

	Formula: MCR * 0.5
DU_P	Penalty Duration **Formula:** ((MIN (10, MAX (0,(AVGD – 5)/2)))

Example

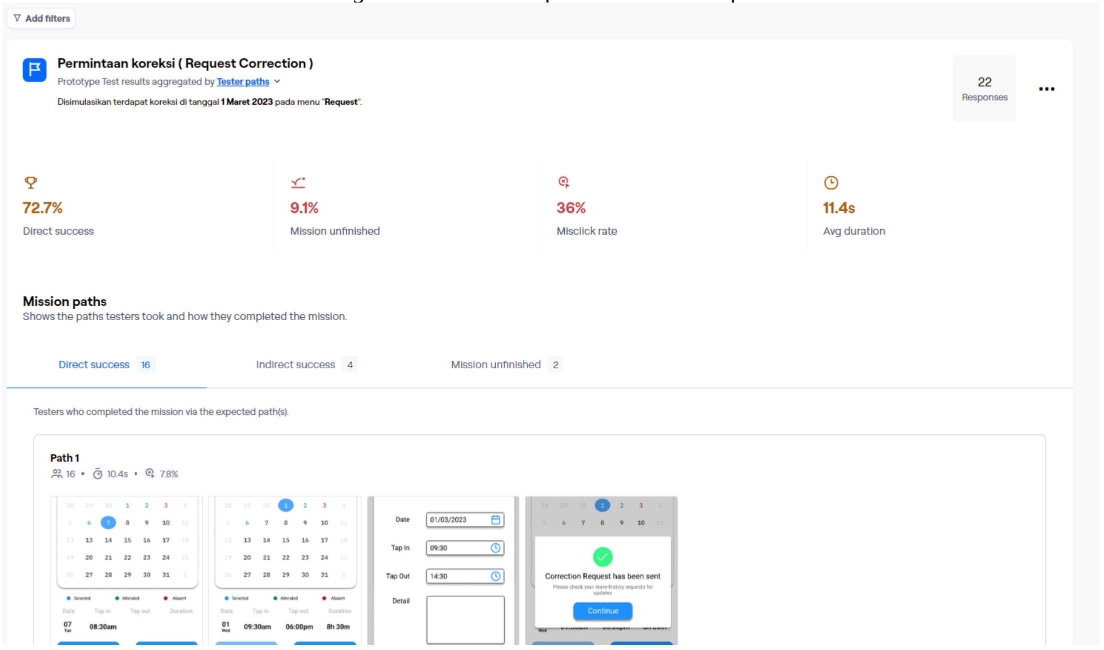
One of the examples is from page *"Permintaan Untuk Absen (Request Correction)"*:


Permintaan koreksi (Request Correction)
Prototype Test
Disimulasikan terdapat koreksi di tanggal **1 Maret 2023** pada menu "**Request**".

72.7% Direct success
9.1% Mission unfinished
22 Total testers
36.0% Misclick rate
11.4s Avg duration

Mission's paths
Click on a path to analyze the results for that path.

Figure 5.18 Maze Request Correction Report


Add filters

Permintaan koreksi (Request Correction)
Prototype Test results aggregated by Tester paths
Disimulasikan terdapat koreksi di tanggal **1 Maret 2023** pada menu "**Request**".

22 Responses

72.7% Direct success
9.1% Mission unfinished
36% Misclick rate
11.4s Avg duration

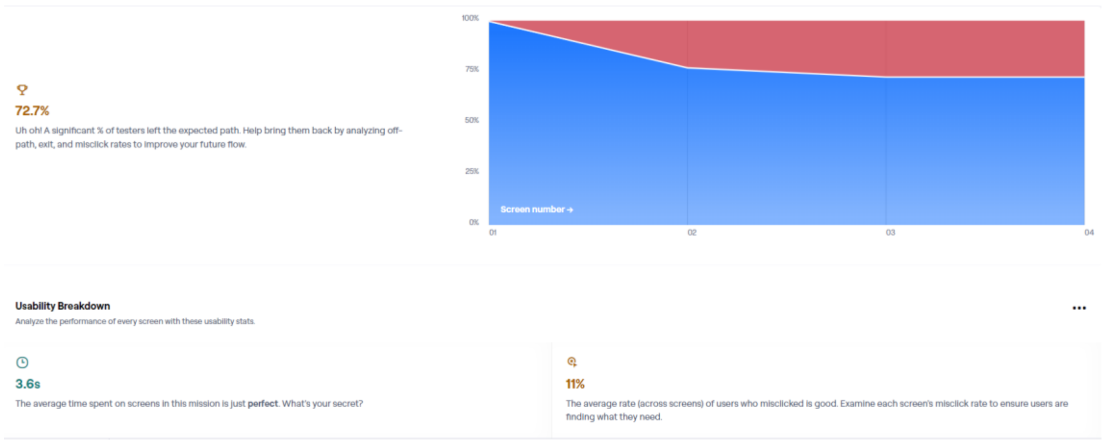
Mission paths
Shows the paths testers took and how they completed the mission.

Direct success 16 | Indirect success 4 | Mission unfinished 2

Testers who completed the mission via the expected path(s).

Path 1
16 • 10.4s • 7.8%

Figure 5.19 Maze Request Correction Report



72.7%

Uh oh! A significant % of testers left the expected path. Help bring them back by analyzing off-path, exit, and misclick rates to improve your future flow.

Usability Breakdown

Analyze the performance of every screen with these usability stats.

3.6s

The average time spent on screens in this mission is just **perfect**. What's your secret?

11%

The average rate (across screens) of users who misclicked is good. Examine each screen's misclick rate to ensure users are finding what they need.

Figure 5.20 Maze Request Correction Report

The expected 77 is counted as **MIUS** for all screens' missions, for every mission they have their own **MIUS** score. However, the author only gives one example. The author labels all the variables such as:

- **DSR** = 73
- **IDSR** = (4/22) * 100)) = 18
- **MC_P** = 11 * 0.5 = 5
- **DU_P** = ((**MIN** (10, **MAX** (0,(4 – 5)/2)))

Substituting variables in the formula:

MIUS = **DSR** + (**IDSR** / 2) – **avg** (**MC_P**) – **avg** (**DU_P**)

MIUS = 73 + (18 / 2) – **5** – ((**MIN** (10, **MAX** (0,(3.6 – 5)/2)))

MIUS = 73 + 9 – **5** – 0

MIUS = 82 - 5

MIUS = 77

5.2.3 Mean Average Usability Score (MAUS)

The **MAUS** is a derived metric that represents the average of all individual MIUS scores calculated for a specific set of usability tests. It provides an aggregated measure of usability performance across multiple testing scenarios and user interactions. By calculating the **MAUS**, the overall usability of the mobile attendance system can be summarized in a single score.

The **MAUS** formula considers the individual **MIUS** scores and averages them to obtain the final usability score. This metric enables a comprehensive evaluation of the system's usability performance across various testing conditions. By examining the results obtained through these metrics, practical recommendations can be derived to enhance the user experience and improve the overall usability of the system. These findings will serve as a guide for optimizing the mobile attendance system, ensuring a more user-friendly and efficient experience for the end-users.

Formula

MAUS=avg(MIUS)

For this calculation, each mission screen that the author conduct in maze will be added and divided by the total amount of mission screens that are being tested in the maze.

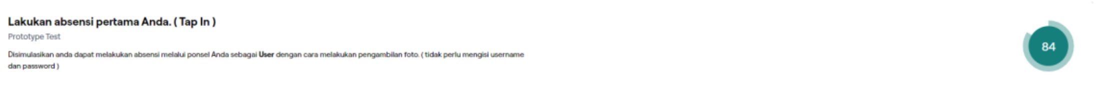

Lakukan absensi pertama Anda. (Tap In)
Prototype Test
Disimulasikan anda dapat melakukan absensi melalui ponsel Anda sebagai **User** dengan cara melakukan pengambilan foto. (tidak perlu mengisi username dan password)
84

Figure 5.21 Maze Tap In MIUS Score Report

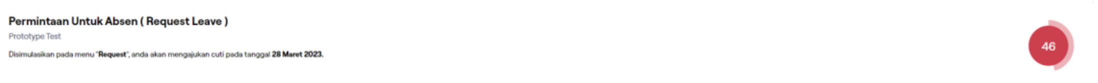

Permintaan Untuk Absen (Request Leave)
Prototype Test
Disimulasikan pada menu "**Request**", anda akan mengajukan cuti pada tanggal **28 Maret 2023.**
46

Figure 5.22 Maze Request Leave MIUS Score Report

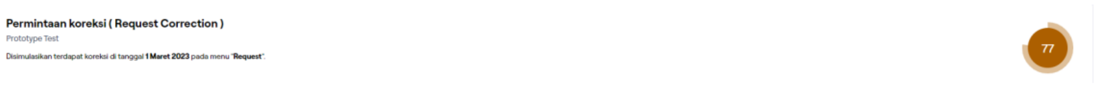

Permintaan koreksi (Request Correction)
Prototype Test
Disimulasikan terdapat koreksi di tanggal **1 Maret 2023** pada menu "**Request**".
77

Figure 5.23 Maze Request Correction MIUS Score Report

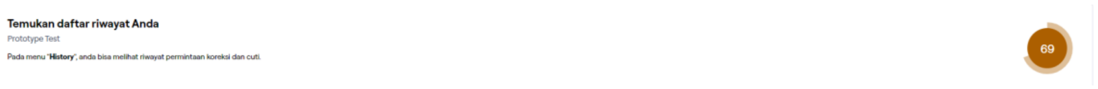

Temukan daftar riwayat Anda
Prototype Test
Pada menu "**History**", anda bisa melihat riwayat permintaan koreksi dan cuti.
69

Figure 5.24 Maze History Page MIUS Score Report

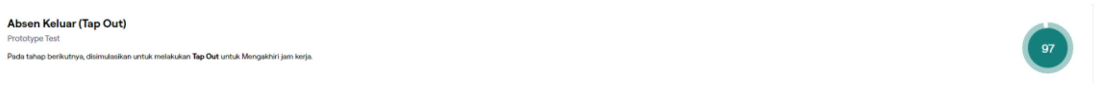

Absen Keluar (Tap Out)
Prototype Test
Pada tahap berikutnya, disimulasikan untuk melakukan **Tap Out** untuk Mengakhiri jam kerja.
97

Figure 5.25 Maze Tap Out MIUS Score Report

Daftar Toleransi
Prototype Test
Temukan informasi berikut di menu pengaturan (**Settings**):
- Sisa Cuti (Leave left)
- Sisa Cuti Sakit (Permission Left)
- Toleransi jam kerja (Tolerance work time)
- Kehadiran di bulan ini (Attended this month)

Figure 5.26 Maze Information Page MIUS Score Report

Listing all the variables for each mission screen score **(MIUS):**

- Tap In = 84
- Request Leave = 46
- Request Correction = 77
- History = 69
- Tap Out = 97
- Information Page = 78

Substituting the variables to the formula:

MAUS = avg (MIUS)

MAUS = avg (84 + 46 + 77 + 69 + 97 + 78**)**

MAUS = 451 / 6

MAUS = 75.166

MAUS ≈ 75

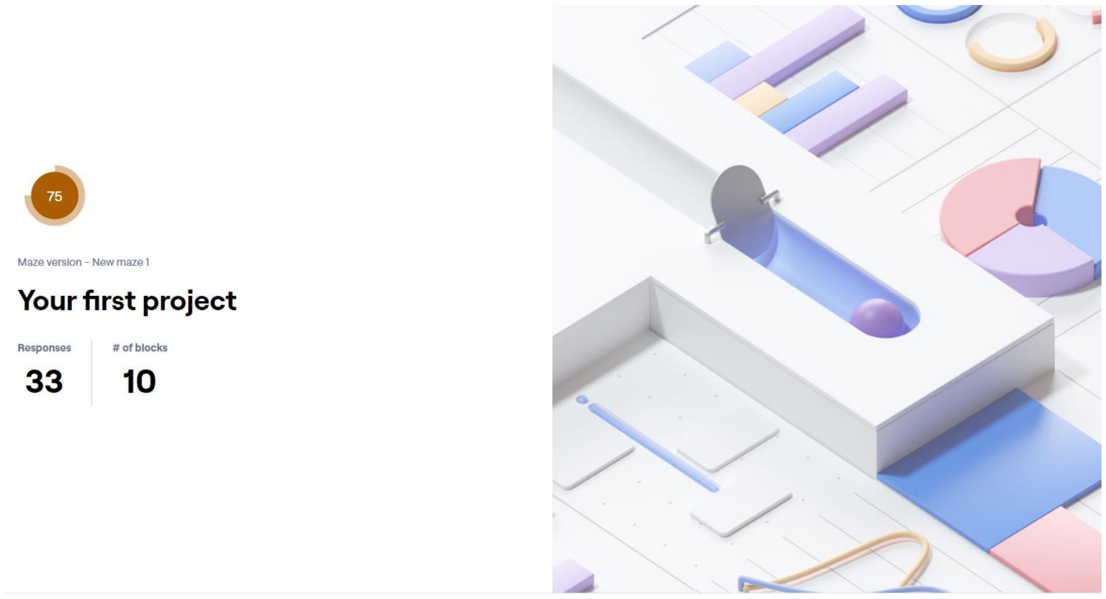


Figure 5.27 Overall Maze Report

MAUS, which stands for Maze Assessment and Utilization System, serves as the comprehensive evaluation metric representing the final score of the entire maze test. This robust scoring system, as highlighted by the author in the group aims section, has been achieved through meticulous design and meticulous implementation.

5.3 UI Testing

In Chapter 4, the author outlined a **UI** testing strategy that combines manual testing and automated testing frameworks to validate the functionality, responsiveness, and design consistency of the user interface. To streamline the **UI** testing process and increase test coverage, the author chose to incorporate the **Espresso** framework, an Android-specific testing framework designed specifically for **UI** testing.

Now, let's explore an example of how this strategy is implemented. The code snippet below demonstrates the **UI** testing of the 'ButtonHalfWidth' component, which is one of the files being tested in the application. This example showcases the utilization of the **Espresso** framework and its capabilities for efficiently automating **UI** tests.

```
import androidx.compose.ui.test.junit4.createComposeRule

import androidx.compose.ui.test.onNodeWithText

import androidx.compose.ui.test.performClick

import androidx.test.ext.junit.runners.AndroidJUnit4

import com.example.Thesis_Project.ui.components.ButtonHalfWidth

import org.junit.Assert.assertEquals

import org.junit.Rule

import org.junit.Test

import org.junit.runner.RunWith


@RunWith(AndroidJUnit4::class)

class ButtonHalfWidthUITest {


    @get:Rule

    val composeTestRule = createComposeRule()


    @Test

    fun buttonHalfWidth_shouldTriggerOnClick() {

        var isClicked = false


        composeTestRule.setContent {

            ButtonHalfWidth(onClick = { isClicked = true }, but-

tonText = "Click Me")

        }


        composeTestRule.onNodeWithText("Click

Me").performClick()


        assertEquals(true, isClicked)
        }
}
```

In this test file, the author focuses on ensuring that the 'ButtonHalfWidth' component behaves as expected when interacted with it. By leveraging the Espresso framework, the author can streamline the **UI** testing process, increase test coverage, and verify that the user interface meets the desired functional and visual requirements.

Within the 'ButtonHalfWidthUITest' class, which is annotated with '@RunWith (AndroidJUnit4::class)',the'buttonHalfWidth_shouldTriggerOnClick()' test method is defined. This method simulates a click on the 'ButtonHalfWidth' component and verifies that the expected behavior occurs by checking whether the 'isClicked' variable has been set to 'true'. The being conducted and shown in these figures below:

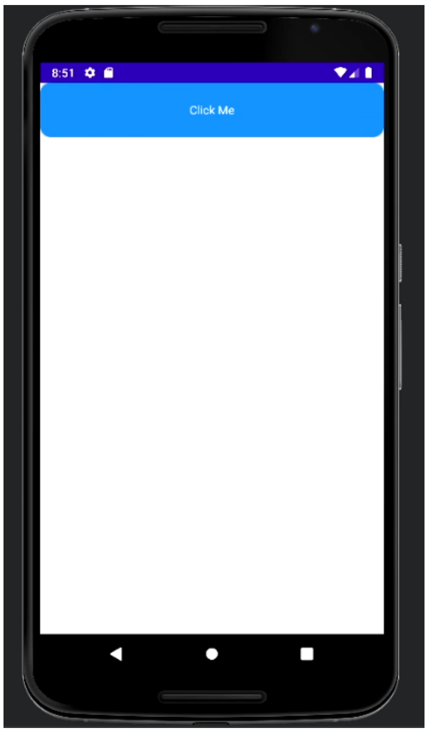


Figure *5.28 UI test running*

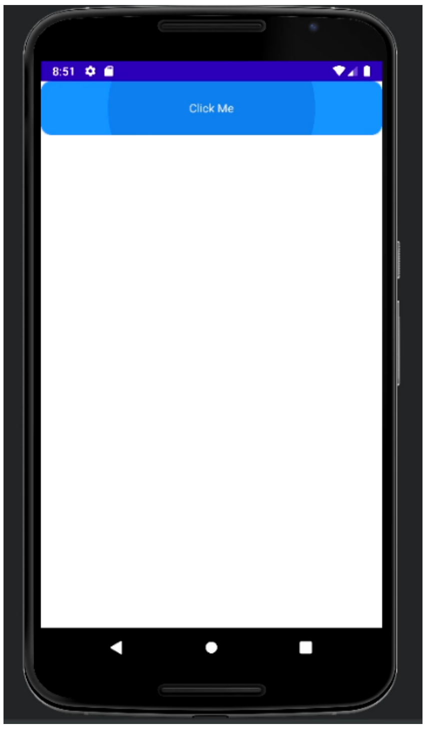


Figure 5.29 UI Test click Running

By employing this combination of manual testing and automated testing frame-works, specifically utilizing the Espresso framework, the author ensures the proper behavior of **UI** components and a seamless user experience. This approach not only streamlines the testing process but also enhances the overall quality and reliability of the application's user interface."

The UI testing strategy demonstrated in the given code snippet can be classified as a form of black box testing. Black box testing focuses on examining the functionality of a system without considering its internal implementation. In this case, the author interacts with the user interface of the 'ButtonHalfWidth' component, without any knowledge of its underlying code. By using the Espresso framework, which operates at the user interface level, the tester evaluates the expected behavior and ensures that the component functions correctly. The use of black box testing in this scenario allows for a thorough examination of the user interface's responsiveness and functionality without relying on the specifics of its internal implementation.

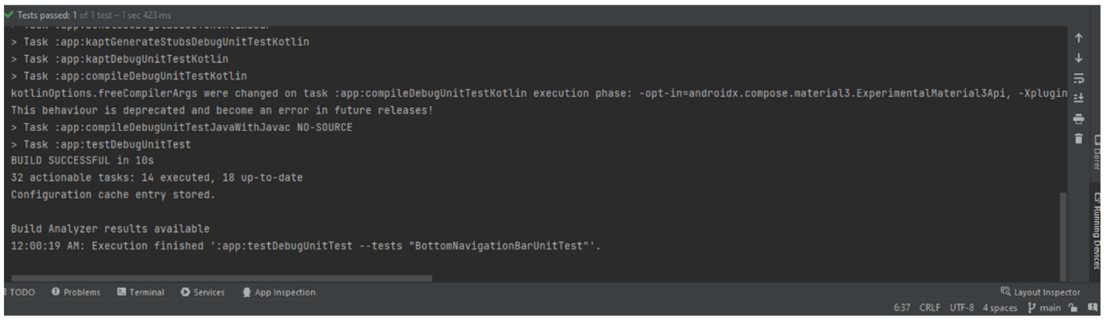
5.4 Unit Testing

This example showcases the utilization of the **JUnit** and **Mockito** frameworks to ensure the reliability and correctness of the individual component. The provided code snippet demonstrates the unit testing of the 'AdminBottomNavigationBar' functionality.

```
import androidx.compose.material.icons.Icons

import androidx.compose.material.icons.outlined.CalendarMonth

import androidx.compose.material.icons.outlined.Home

import androidx.compose.material.icons.outlined.Work

import androidx.compose.ui.graphics.vector.ImageVector

import androidx.navigation.NavController

import io.mockk.mockk
import io.mockk.verify

import org.junit.Test


data class BottomNavItemUnitTest(

    val title: String,

    val route: String,

    val icon: ImageVector

)


object AdminBottomNavBarRoutes {

    object AdminHomeScreen {

        const val route = "admin_home_screen"

    }


    object AdminUsersScreen {

        const val route = "admin_users_screen"

    }

}


val bottomNavItemsUnitTest = listOf(

    BottomNavItemUnitTest(

        title = "Home",

        route = AdminBottomNavBarRoutes.AdminHomeScreen.route,

        icon = Icons.Outlined.Home

    ),

    BottomNavItemUnitTest(

        title = "Users",

        route = AdminBottomNavBarRoutes.AdminUsersScreen.route,

        icon = Icons.Outlined.Work

    )
)


class AdminBottomNavigationBarUnitTest {


    @Test
    fun adminBottomNavigation-
Bar_OnItemClick_NavigateToCorrectScreen() {
        val navController = mockk<NavController>(relaxed = true)


        bottomNavItemsUnitTest.forEach { item: BottomNavItemU-
nitTest ->
            // Handle item click logic
            // Navigate to the corresponding route
            navController.navigate(item.route)


            verify { navController.navigate(item.route) }
        }
    }
}
```

The provided code snippet demonstrates the implementation of unit testing for the Admin Bottom Navigation Bar functionality. The AdminBottomNavigationBa-rUnitTest class contains a test method called adminBottomNavigation-Bar_OnItemClick_NavigateToCorrectScreen, which ensures that when an item in the bottom navigation bar is clicked, the navigation controller correctly navigates to the corresponding screen.

The bottomNavItemsUnitTest variable is a list of BottomNavItemUnitTest objects that represent the items in the bottom navigation bar. Each object contains properties such as the item's title, route, and associated icon. Within the adminBottomNavigationBar_OnItemClick_NavigateToCorrectScreen test method, a NavController is created using the mockk function from the Mockito framework. This mock object simulates the behavior of the navigation controller, allowing the unit test to focus on the specific functionality being tested.



```
Tests passed: 1 of 1 test – 1 sec 423 ms
> Task :app:kaptGenerateStubsDebugUnitTestKotlin
> Task :app:kaptDebugUnitTestKotlin
> Task :app:compileDebugUnitTestKotlin
kotlinOptions.freeCompilerArgs were changed on task :app:compileDebugUnitTestKotlin execution phase: -opt-in=androidx.compose.material3.ExperimentalMaterial3Api, -Xplugin
This behaviour is deprecated and become an error in future releases!
> Task :app:compileDebugUnitTestJavaWithJavac NO-SOURCE
> Task :app:testDebugUnitTest
BUILD SUCCESSFUL in 10s
32 actionable tasks: 14 executed, 18 up-to-date
Configuration cache entry stored.

Build Analyzer results available
12:00:19 AM: Execution finished ':app:testDebugUnitTest --tests "BottomNavigationBarUnitTest"'.
```

Figure 5.30 Unit Test Result

The test uses the "verify" function to assert that the navigation controller indeed receives the expected navigation calls for each item. This ensures that clicking on each item in the bottom.

The build process for the project is executed, and the test is successfully run without any failures. The build output indicates the success of the task and provides information on the executed tasks and their statuses.

The test method iterates over each item in the bottomNavItemsUnitTest list. It simulates the item click by navigating to the corresponding route using the navController.navigate method. Finally, the verify function from **Mockito** is used to verify that the navigation controller indeed navigates to the expected route.

By implementing this unit test, the author demonstrates the application of the unit testing approach outlined in Chapter 4. The test ensures the reliability and correctness of the AdminBottomNavigationBar component by validating its navigation behavior. By utilizing the **JUnit** and **Mockito** frameworks, the author effectively isolates the component being tested and verifies its expected functionality.

This unit testing methodology helps identify defects or issues at the individual component level, ensuring the reliability and correctness of the system's units. Through the systematic execution of unit tests, the author can increase the overall quality of the software and provide assurance that the Admin Bottom Navigation Bar functions as intended.

The code demonstrates the implementation of unit testing, which falls under the category of white box testing. White box testing involves examining the internal structure and logic of the code to evaluate its functionality and identify potential defects. In this case, the unit test focuses on verifying the navigation behavior of the Admin Bottom Navigation Bar by simulating item clicks and asserting that the navigation controller correctly navigates to the expected screens. The use of JUnit and Mockito frameworks aids in isolating the component being tested and validating its expected behavior.

5.5 Integration Testing

This section delves into an exemplary illustration of integration testing, further expanding on the concepts discussed in Chapter 4. The presented code snippet displays the integration testing of the 'AdminCreateUserDialog' functionality, showcasing the practical implementation of Firebase Firestore—a NoSQL document-based database provided by **Google Firebase**—and leveraging Kotlin **Coroutines** for asynchronous programming.

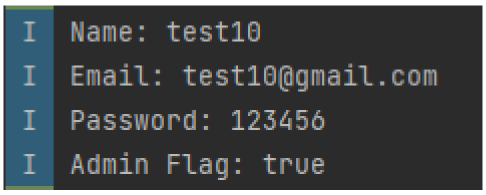
```
import com.google.firebase.firestore.FirebaseFirestore
import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.GlobalScope

import kotlinx.coroutines.launch


object AdminCreateUserDialogIntegrationTest {

    private val firestore: FirebaseFirestore = Fire-

baseFirestore.getInstance()


    fun logUserInputs(name: String, email: String, pass-

word: String, adminFlag: Boolean) {

        // Log user inputs

        println("Name: $name")

        println("Email: $email")

        println("Password: $password")

        println("Admin Flag: $adminFlag")


        // Perform integration test

        val user = hashMapOf(

            "name" to name,

            "email" to email,

            "password" to password,

            "adminFlag" to adminFlag

        )


        // Send data to Firebase and verify the result

        GlobalScope.launch(Dispatchers.IO) {

            firestore.collection("users")

                .add(user)
                .addOnSuccessListener {
                    println("Data added to Firebase suc-
cessfully")
                }
                .addOnFailureListener { exception ->
                    println("Failed to add data to Fire-
base: ${exception.message}")
                }
        }
    }}
```

The presented code serves as an example of the integration testing process for the 'AdminCreateUserDialog' functionality. Enclosed within the AdminCreateUser-DialogIntegrationTest object, the logUserInputs function captures and logs the user inputs while simultaneously performing the integration test. By constructing a user object with the provided input data, the function proceeds to send the object to **Firebase** Firestore—an integral component of the mobile attendance system, responsible for storage and data synchronization.

To achieve effective handling of asynchronous operations, **Kotlin Coroutines** are employed. Through the utilization of the GlobalScope.launch function, a **coroutine** is initiated on the IO dispatcher, enabling non-blocking execution of the operation.

The ultimate objective of this integration test is to validate the successful addition of data to **Firebase** Firestore. By utilizing the Firestore **API**, the code appends the user object to the 'users' collection. Success and failure listeners are registered to provide appropriate feedback messages based on the outcome of the operation.

In the AdminCreateUserDialogIntegration.kt file, there is a specific scenario where the AdminCreateUserDialogIntegrationTest.logUserInputs(name, email, password, adminFlag) function is called within the onSubmitClicked function. As a result, the following log entries are generated in the logcat:



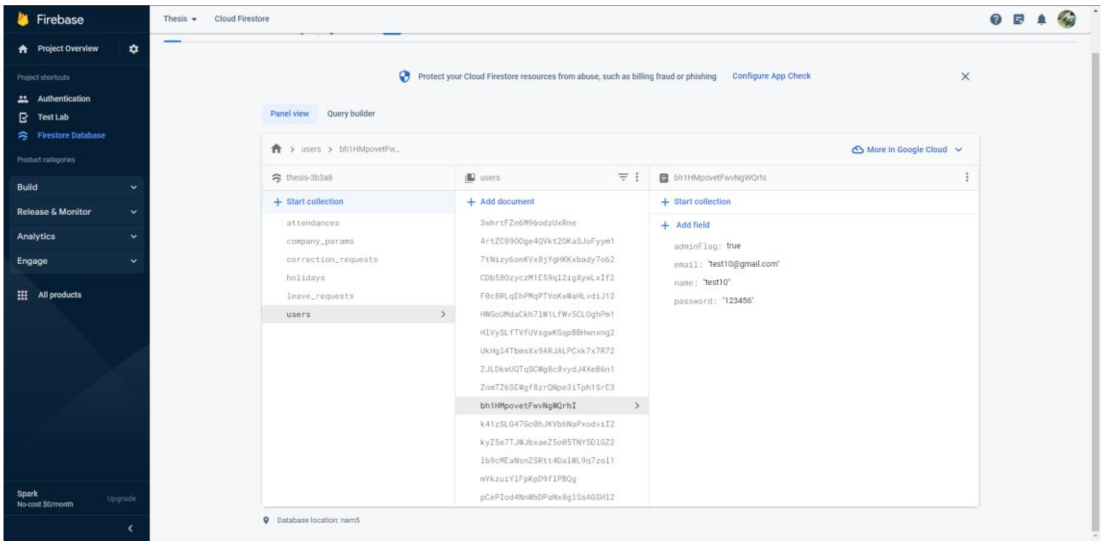
```
I  Name: test10
I  Email: test10@gmail.com
I  Password: 123456
I  Admin Flag: true
```

Figure 5.31 Integration Test Result

These log entries signify the successful execution of the integration test. The log entries include specific information about the user inputs that were logged, such as the name, email, password, and admin flag. In this example, the logged user inputs are as follows:

- Name: test10
- Email: test10@gmail.com
- Password: 123456
- Admin Flag: true

These log entries provide valuable insights into the data that was passed to the logUserInputs function during the integration testing process. By analyzing these log entries, developers can verify the correctness and accuracy of the user input handling within the AdminCreateUserDialog functionality.



users > bh1HMpovetFw_

thesis-3b3a8

- attendances
- company_params
- correction_requests
- holidays
- leave_requests
- users

users

- 3whrtFZn6M96odzUxRne
- 4rtZC890Oge4QVkt2OKaSJoFyym1
- 7tNizy6onKVx8jYgHKKxbady7o62
- CDb58OzyczM1E59ql2igXywLxIf2
- F0c8RLqEhPMqPTVoKwWaHLvdiJ12
- HWGoUMdaCkh71W1LfWv5CLOghPm1
- H1VySLfTVfUVsgwKSqpBBHwnxng2
- UkHgl4TbmsXx9ARJALPCxk7x7R72
- ZJLDkwUQTqSCWg8c8vydJ4XeB6n1
- ZnmTZ6SEWgf8zrQNpe3iTph1SrE3
- bh1HMpovetFwvNgWQrhI
- k41zSLG47Gc0hJKVb6NaPxodviI2
- kyZ5e7TJWJbxaeZ5o05TNY5D1GZ2
- 1b9cMEaNsnZSRtt4DalWL9q7zol1
- mYkzuiY1FpKpD9f1PBQg
- pCePIod4NnWbDPaNx8glSsAGSH12

bh1HMpovetFwvNgWQrhI

```
adminFlag: true
email: "test10@gmail.com"
name: "test10"
password: "123456"
```

Figure 5.32 Firestore Database Integration Testing

To further ensure the reliability of the tested functionality, a manual verification process was conducted to compare the generated log entries with the actual data inserted into the Firestore database.

This outcome aligns with the objectives of the integration testing strategy outlined in the thesis, as it ensures that the data input from the user is successfully captured and processed within the system. It validates the seamless integration between different system components, including the AdminCreateUserDialog functionality and Firebase Firestore, ensuring data consistency and reliability within the mobile attendance system.

By exemplifying this integration testing scenario, the author effectively demonstrates the practical implementation of the integration testing strategy elucidated in Chapter 4. This approach fosters the verification of component interactions, the identification of potential integration issues, and the achievement of seamless integration within the mobile attendance system.

The integration testing scenario for the 'AdminCreateUserDialog' functionality can be classified as black-box testing. This approach focuses on evaluating the system's external behavior and integration without considering internal implementation details. The test verifies the successful addition of data to Firebase Firestore, treating the 'AdminCreateUserDialog' as a black box by interacting with its defined inputs and outputs. By adopting this approach, the test ensures unbiased assessment of the system's behavior and its integration with external components, allowing for thorough testing and identification of potential integration issues while maintaining independence from internal implementation details.

5.6 Beta Testing

The author delves into the implementation of technical beta testing for the mobile attendance system. Technical beta testing aims to validate the system's behavior and assess its overall readiness for release. This testing approach focuses on evaluating the system's technical aspects, such as performance, security, compatibility, and overall robustness. The author, as an internal beta tester, actively engaged in the testing process, leveraging their expertise and familiarity with the system's functionalities.

5.6.1 Technical Beta Testing Process

During the implementation of technical beta testing, a structured process was followed to ensure a comprehensive evaluation of the system's technical aspects. The key elements of the technical beta testing process included:

5.6.1.1 Involvement of Internal Beta Testers

For the technical beta testing of the mobile attendance system, the author actively engaged as an internal beta tester. The author, possessing in-depth knowledge of the system's intricacies and functionality, played a crucial role in identifying potential technical issues and providing valuable feedback for system improvement.

5.6.1.2 Evaluation of Performance and Compatibility

The system's performance, scalability, security, and compatibility with different devices and platforms were evaluated during the testing phase. This assessment aimed to identify any technical issues that could impact the system's performance or user experience.

5.6.1.3 Feedback Collection and Analysis

Throughout the technical beta testing phase, feedback was collected, analyzed, and documented. The author, in their role as an internal beta tester, provided valuable insights into technical issues, system performance, and overall user experience.

5.6.2 Identification of Errors and UI Issues

To provide a comprehensive understanding of the technical beta testing process, screenshots and images were captured to showcase errors and UI issues encountered during testing. These visuals serve as representations of the identified issues, highlighting specific areas that require attention and improvement. It is important to note that all errors and UI issues mentioned in this section are part of the technical beta testing process and were addressed by the author and fixed by the development team. Let's look at some examples:

5.6.2.1 Error Example:

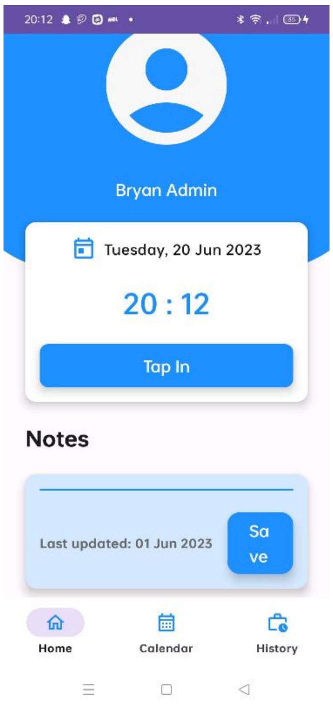


Figure 5.33 Note in Home Page

Description: The observation of this error during the technical beta testing phase highlights an important usability issue that needs to be addressed. It signifies that the feature for saving notes is not functioning properly, potentially causing inconvenience for users who rely on this functionality.

5.6.2.2 UI Issue Example:

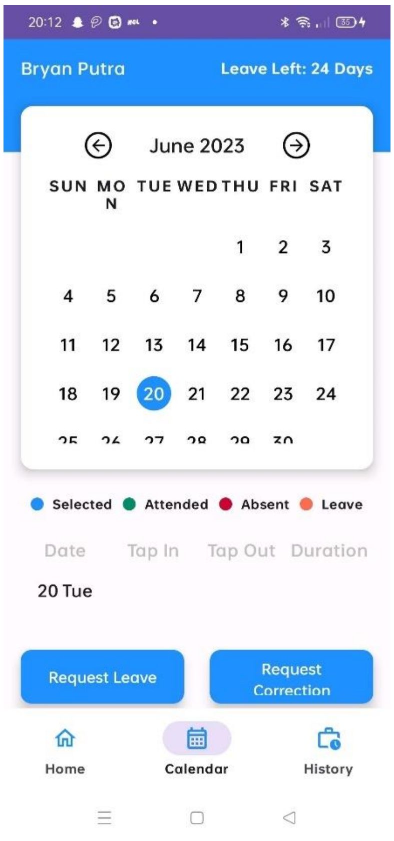


Figure 5.34 Month Title Error

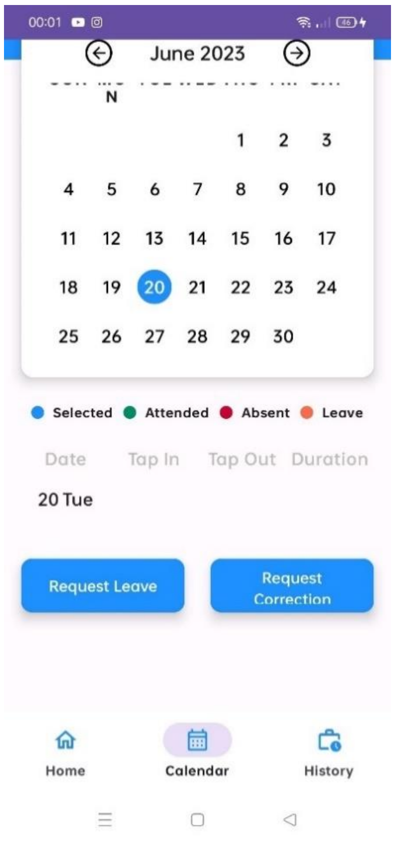


Figure 5.35 Month Scroll Bar Error

In figure 5.3.4, image showcases a **UI** issue where the text inside the button, "Request Correction", is not neat and tidy with the size of the button. It also shows the Month Text above the date of the calendar is not in the right size because the alphabet "**N**". Issues were identified and adjustments will be made.

In figure 5.35, images show that when the author scrolls down the "Calendar" page, all the other components will also follow on how the author scrolls. This issue was identified through technical beta testing, and necessary adjustments for scroll bar so that calendar will not move were made to enhance the visual layout and improve usability across different screen sizes.

In both figures, the page called "Calendar" was an issue regarding with their potential customers and then replaced with the word "Request".

These are the images of the fixed **UI**:

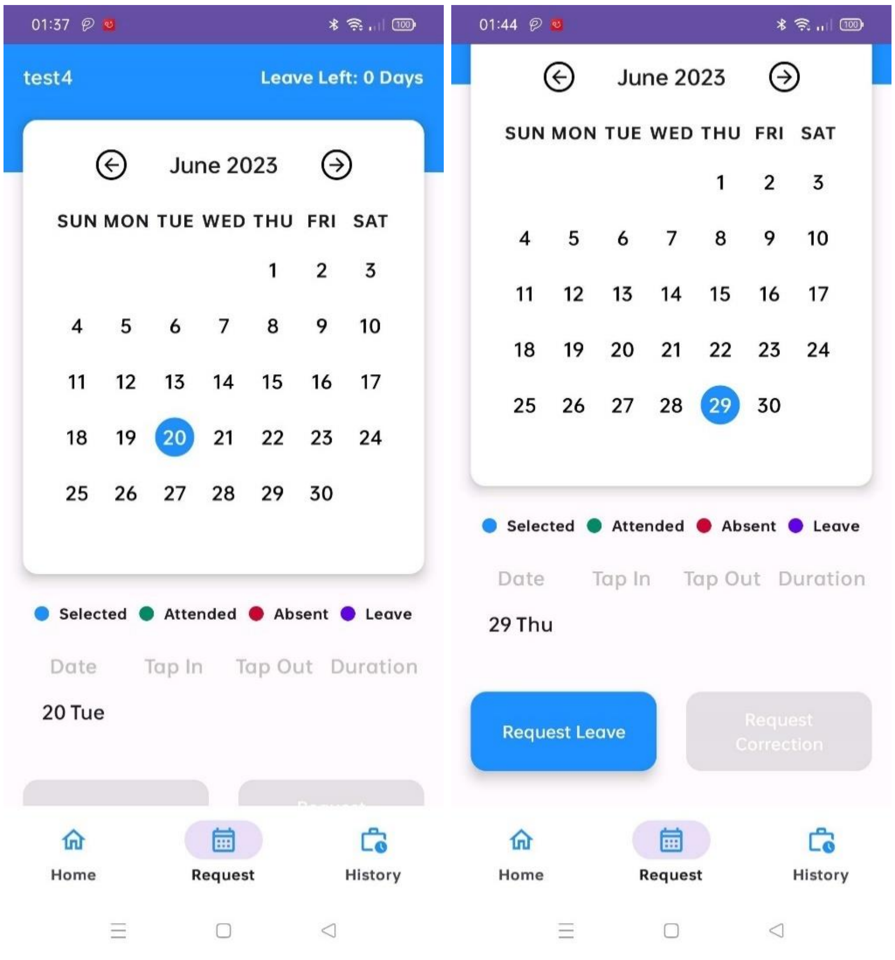
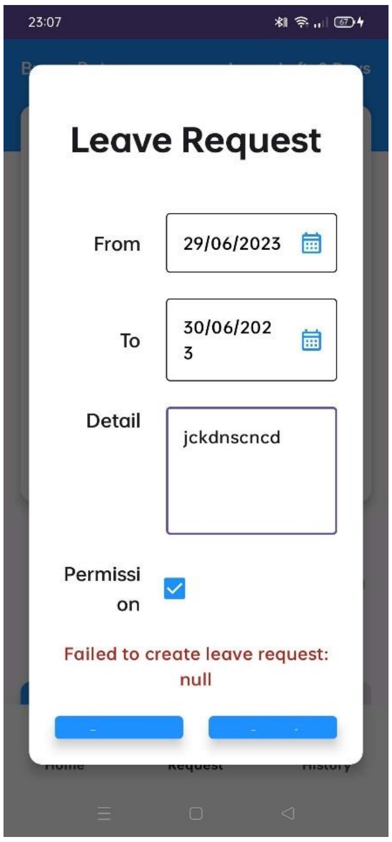

test4
Leave Left: 0 Days
June 2023
SUN MON TUE WED THU FRI SAT
● Selected ● Attended ● Absent ● Leave
Date Tap In Tap Out Duration
20 Tue
29 Thu
Request Leave
Request Correction
Home
Request
History

Figure 5.36 Fixed Request Page

Issues was resolved as the "**MON**" stands for Monday is within its size and box. With the fixes too regarding the box overlapping the "Request Correction" text has been fixed.



29/06/2023

30/06/202
3

jckdnscncd

✓

Failed to create leave request:
null

Figure 5.37 When Requesting a Leave

In figure 5.37, The image shows that the Text button “Request” on the left button and “Cancel” on the right button were not showing only after the author clicked “Request”. The image also show that when the author try to “Request” a leave, the text prompted “Failed to create leave request:null” indicates that there are errors in creating the leave request. However, the error was resolved with another error when author click “Request” for the 2nd time. The submission of the error is submitted. These issues were identified and addressed to the development team.

These are the images of Fixed **UI**:

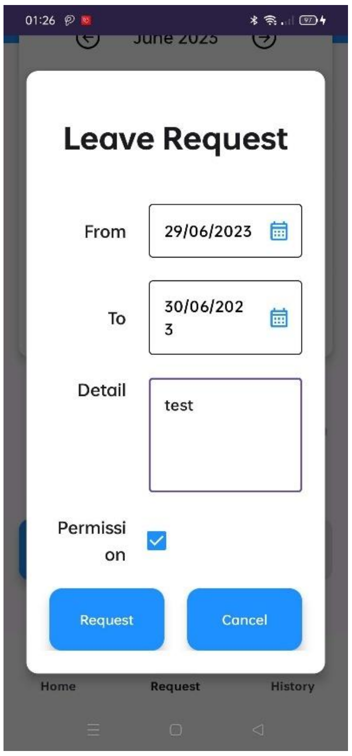
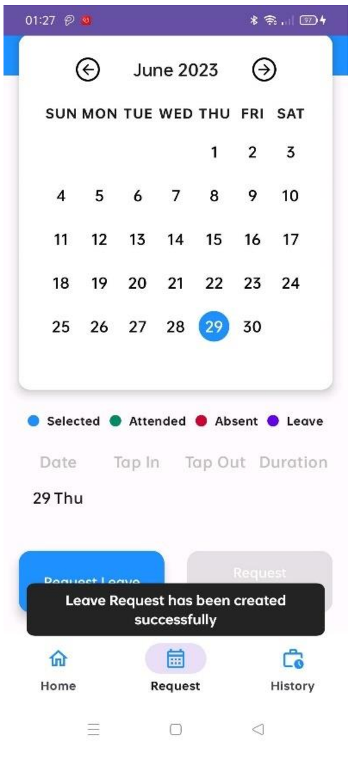


Figure 5.38 Requesting



Leave Request has been created successfully

Figure 5.39 Success Request

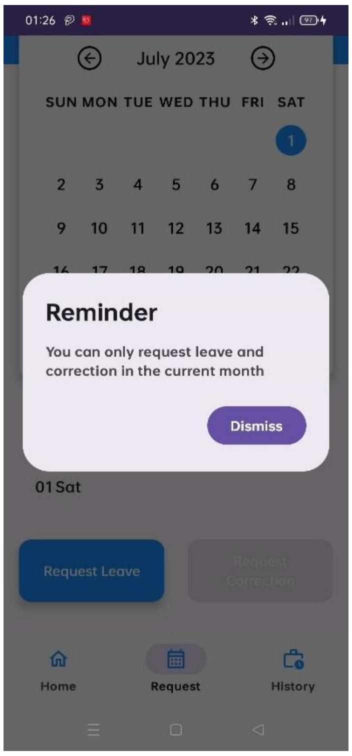


Figure 5.40 Request Validation

In figure 5.38, requesting on the day that they go to work will be allowed. For testing purposes, the author tested successfully and prompted with "Leave Request has been created successfully" in figure 5.39.

In figure 5.40, issues regarding author can request a leave on any date has been fixed. The author tested any date in July 2023 and was not able to make a leave request. The issues have been fixed and added.

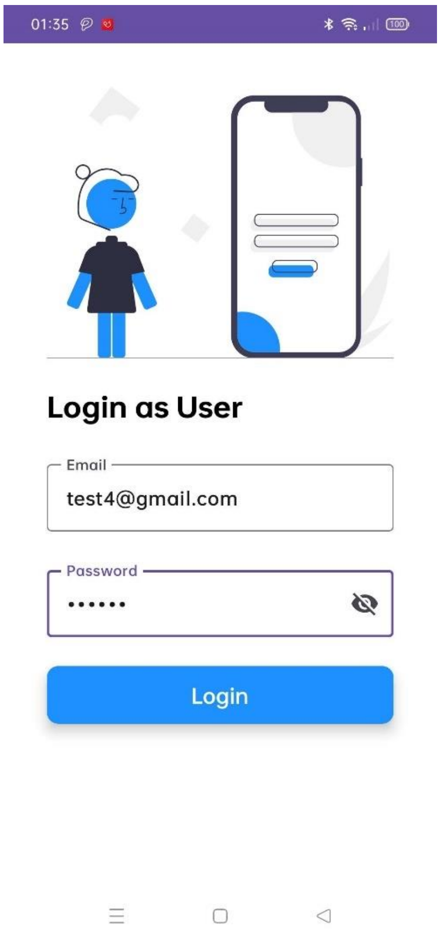


Figure 5.41 Login as test4

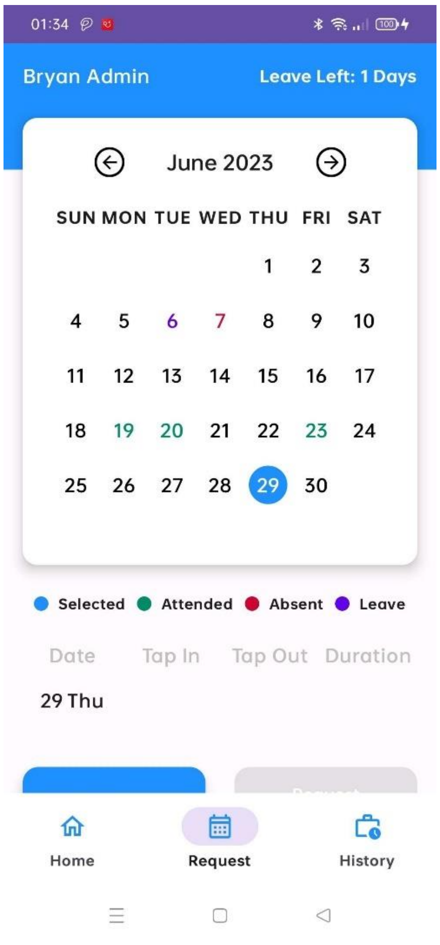

01:34
Bryan Admin
Leave Left: 1 Days
June 2023
SUN MON TUE WED THU FRI SAT
● Selected ● Attended ● Absent ● Leave
Date Tap In Tap Out Duration
29 Thu
Home
Request
History

Figure 5.42 test4 Request page

In figure 5.41, the author logged in as test4, that has been added through admin page and put test4 as their username too. However, the request page of user “test4” was named “Bryan Admin” on the top left during the test. This was identified by the author and addressed to the development team.

This is the Fixed **UI**:

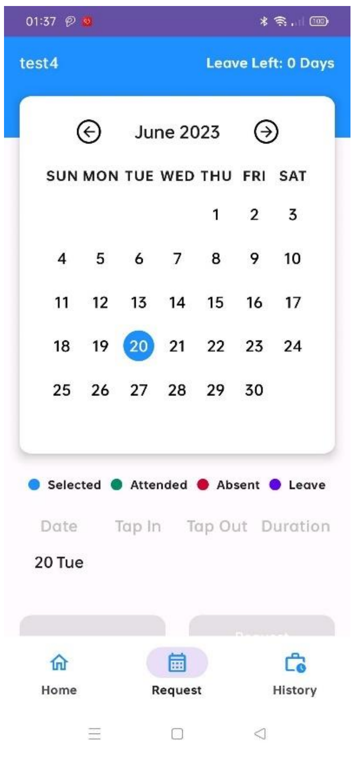


Figure 5.43 Fixed Request UI

The author has checked and logged in as “test4” and identified that the issue has been solved as the text on top left on “Request” page has follow the logged user’s username.

5.6.3 Leveraging Technical Beta Testing Insights

By implementing technical beta testing, the author aimed to thoroughly evaluate the mobile attendance system's technical performance and usability. The insights gained from this process played a crucial role in refining the system, addressing identified issues, and optimizing its overall technical integrity.

The systematic identification and resolution of technical issues during the implementation of technical beta testing contributed to the creation of a high-quality and reliable mobile attendance system.

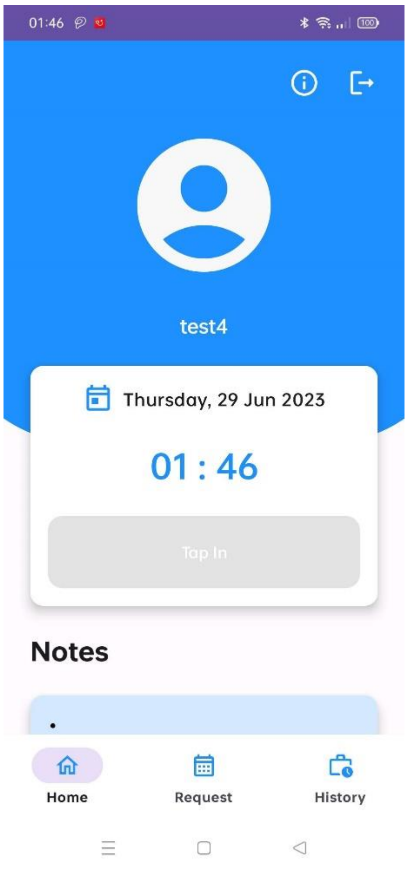
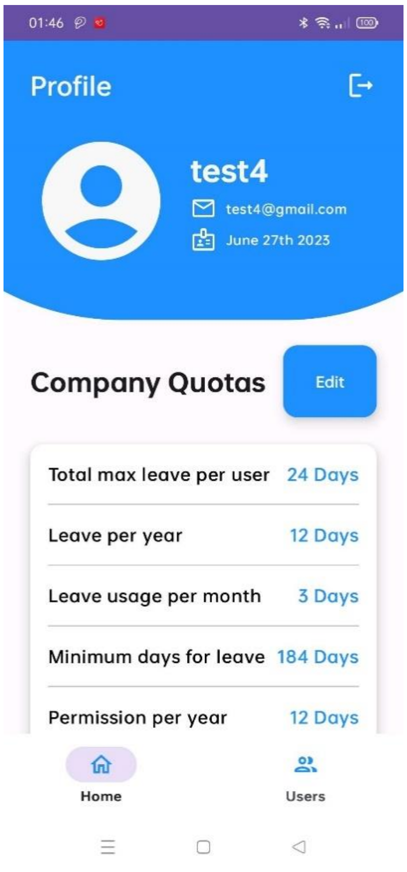


Figure 5.44 Login user as test4


01:46

Profile

test4

test4@gmail.com

June 27th 2023

Company Quotas

Edit

Total max leave per user	24 Days
Leave per year	12 Days
Leave usage per month	3 Days
Minimum days for leave	184 Days
Permission per year	12 Days

Home

Users

Figure 5.45 Login admin as test4

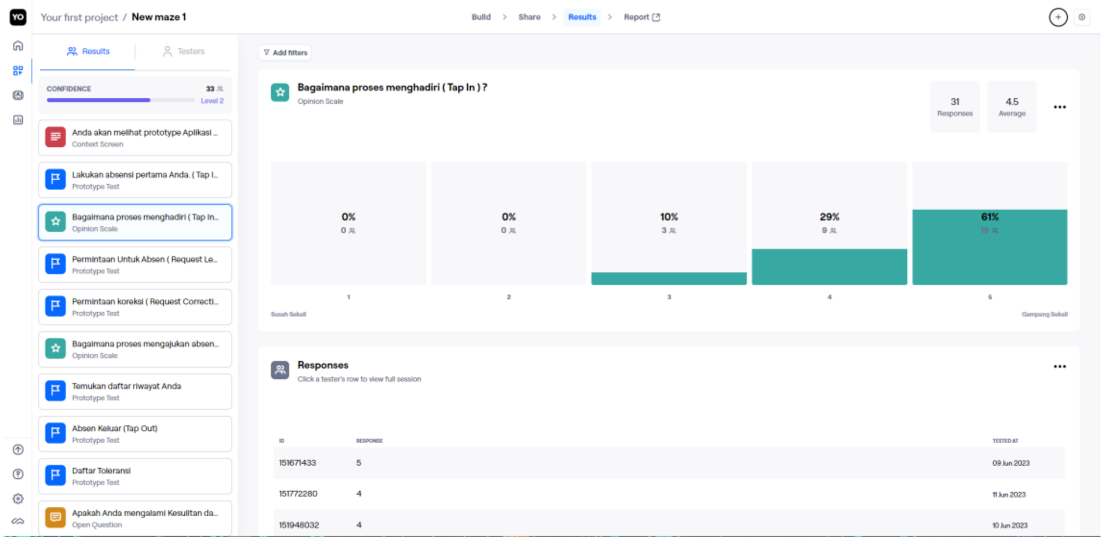
In Figure 5.44, the author tests to log in as user. The author then logged out and headed to log in as admin. Regarding what users or admin the author logged in. The test identifies when the author pressed the back button on mobile devices. The app will trace back on its navigation graph. So, when the author is on the admin page after logged in as users. The author finds out that pressing the back button will bring the author back to figure 5.44. This issue has been identified and fixed.

Overall, the author has indicated that technical beta testing is considered as a grey box testing because it involves having partial knowledge of the internal workings of the system while still focusing on external behaviors and inputs. In technical beta testing, the author has access to the system's design, architecture, and code to some extent. This knowledge enables them to conduct in-depth testing, including the identification of potential issues, bugs, or usability concerns, and providing feedback for further refinement. While having access to internal information, the testing team still evaluates the system based on its external behavior, making it a combination of both white box and black box testing approaches, hence falling under the grey box testing category.

5.6.4 User Acceptance Testing

User Acceptance Testing (UAT) was conducted to evaluate the acceptance and usability of the application among potential users. The testing focused on the main features of the application and utilized a rating scale of 1-5 to assess the ease of use. It is important to note that the ratings were collected specifically for individual features, rather than providing an overall rating of the application.

During the UAT process, a prototype version of the application was provided to potential users. They were given specific tasks to perform, and their interactions and feedback were observed and recorded. The tasks were designed to cover various scenarios and functionalities within the application. Participants were asked to rate each main feature on a scale of 1-5, where 1 represented "difficult to use" and 5 represented "very easy to use."



Anda akan melihat prototype Aplikasi ...
Context Screen

Lakukan absensi pertama Anda. (Tap I...
Prototype Test

Bagaimana proses menghadiri (Tap In...
Opinion Scale

Permintaan Untuk Absen (Request Le...
Prototype Test

Permintaan koreksi (Request Correcti...
Prototype Test

Bagaimana proses mengajukan absen...
Opinion Scale

Temukan daftar riwayat Anda
Prototype Test

Absen Keluar (Tap Out)
Prototype Test

Daftar Toleransi
Prototype Test

Apakah Anda mengalami Kesulitan da...
Open Question

Bagaimana proses menghadiri (Tap In) ?
Opinion Scale

31 Responses
4.5 Average

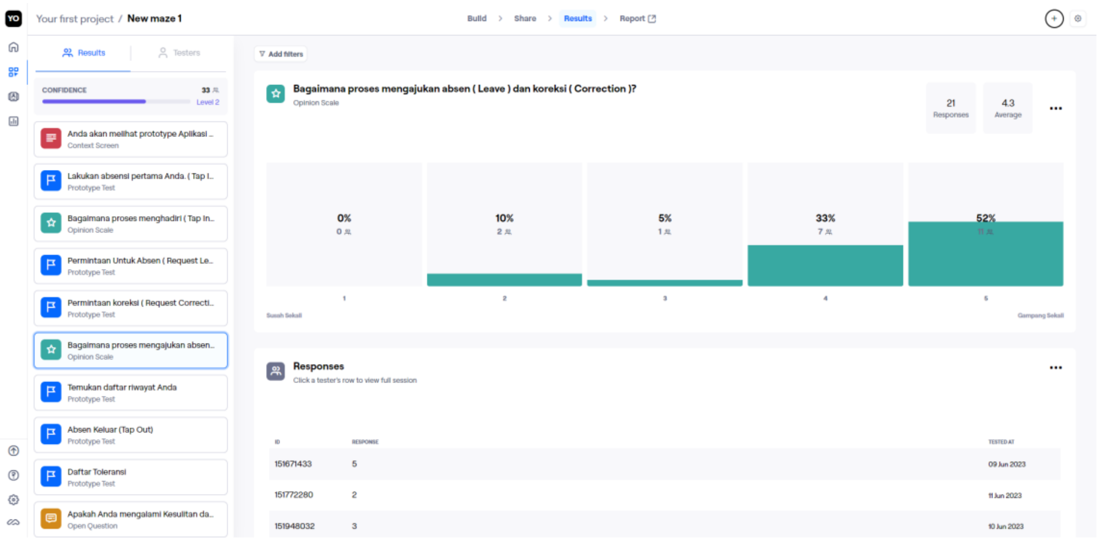
1	2	3	4	5
0%	0%	10%	29%	61%
0	0	3	9	19

Susah Sekali — Gampang Sekali

Responses
Click a tester's row to view full session

ID	RESPONSE	TESTED AT
151671433	5	09 Jun 2023
151772280	4	11 Jun 2023
151948032	4	10 Jun 2023

Figure 1 5.46 Tap In User Acceptance Score



Anda akan melihat prototype Aplikasi ... — Context Screen
Lakukan absensi pertama Anda. (Tap I... — Prototype Test
Bagaimana proses menghadiri (Tap In... — Opinion Scale
Permintaan Untuk Absen (Request Le... — Prototype Test
Permintaan koreksi (Request Correcti... — Prototype Test
Bagaimana proses mengajukan absen... — Opinion Scale
Temukan daftar riwayat Anda — Prototype Test
Absen Keluar (Tap Out) — Prototype Test
Daftar Toleransi — Prototype Test
Apakah Anda mengalami Kesulitan da... — Open Question

Bagaimana proses mengajukan absen (Leave) dan koreksi (Correction)?

21 Responses | 4.3 Average

1	2	3	4	5
0%	10%	5%	33%	52%
0	2	1	7	11

ID	RESPONSE	TESTED AT
151671433	5	09 Jun 2023
151772280	2	11 Jun 2023
151948032	3	10 Jun 2023

Figure 2 5.47 Request Page User Acceptance Score

The results of the UAT indicated that the application was well-received by the participants and result of the UAT is the main page of the author’s team application, when user interacting the “Request” page and “Tap In” feature. As a standard, the ratings for the main features ranged from 3 to 5, with an average score of 4.4 from the average of 4.5 from “Tap In” feature and 4.3 from “Request” Page. This indicates a generally positive response regarding the ease of use and user-friendliness of the application's main features.

Furthermore, in the appendix section, potential user provided additional feedback and comments about the application. They expressed satisfaction with the overall design and deployment of the application. It should be noted that the prototype used during the UAT closely resembled the final product, as it was developed by the author's team. Potential user appreciated the consistency between the prototype and the final product, which contributed to their positive perception of the application.

5.7 Analysis

During the usability testing sessions, it became evident that the second version, with its clear and concise instructions, resonated more successfully with the users. The streamlined and shorter instructions facilitated a smoother user experience, allowing participants to navigate the system effortlessly and accomplish tasks with greater efficiency. The insights gained from testing both versions provided valuable feedback for refining the design and enhancing user satisfaction.

It is important to note that the screenshots displayed in this subchapter represent a work in progress and do not reflect the final design of the mobile attendance system.

During the unit testing phase, the individual components and functions of the mobile attendance system were thoroughly examined. This testing approach aimed to verify the correctness of each unit in isolation and ensure that they functioned as expected. The unit testing process involved creating test cases for various scenarios and executing them to validate the behavior and functionality of the system's units.

The unit testing process played a crucial role in identifying and rectifying errors at the granular level. By isolating and testing each unit independently, the author gained confidence in the reliability and accuracy of the system's core functionality. The insights gained from unit testing were instrumental in identifying and addressing bugs, ensuring the overall stability and robustness of the mobile attendance system.

Through integration testing, the author aimed to identify any inconsistencies, communication failures, or compatibility issues between different modules. By simulating real-world scenarios and executing integration test cases, the author gained valuable insights into the system's behavior in a whole environment. Integration testing provided assurance that all components integrated properly, enabling smooth data flow and system functionality.

Finally, through technical beta testing, the author was provided with access to the mobile attendance system and encouraged to actively engage with its features and functionalities. The findings from technical beta testing were instrumental in refining the design, resolving errors, and enhancing the overall functionality and usability of the mobile attendance system. The identified issues and suggestions for improvement were carefully analyzed and implemented, resulting in significant enhancements to the system's performance and user experience.

CHAPTER 6

DISCUSSION

6.1 Discussion

The results obtained from this study indicate that the developed Secure Mobile Attendance System with Face Recognition and Edge Computing is effective in providing a reliable and practical method for tracking attendance. The system's usability and user experience were enhanced through the implementation of various testing techniques, including UI testing, usability testing, integration testing, and unit testing.

The integration of facial recognition and edge computing technologies in attendance systems was found to offer several advantages. It provides an advanced and accurate alternative to conventional attendance methods, eliminating the need for physical devices like cards or readers. Additionally, the user-friendly nature of face-recognition mobile attendance applications enables swift and easy check-in and check-out processes, without disrupting users' work or studies. Moreover, the use of facial recognition promotes a hygienic approach by reducing the need for shared devices or surfaces, especially in environments where individuals are in proximity and there is a significant risk of virus transmission.

It is crucial to address the usability and user experience issues in mobile attendance systems with facial recognition to ensure widespread adoption and usage. This chapter focused on identifying and resolving common usability and user experience challenges faced by users and developers. Through a thorough analysis of available literature and case studies, the study aimed to provide insights into the impact of poor usability and user experience on adoption rates and offer suggestions for building and creating a user-friendly face-recognition mobile attendance system.

The author’s role as Quality Assurance (QA) in the development of the Secure Mobile Attendance System was to ensure its simplicity, intuitiveness, and overall quality. Various strategies were employed to achieve this goal, including comprehensive unit tests, integration tests, and beta tests. Rigorous usability testing was conducted with a sample user group to identify any potential issues or areas of difficulty. User feedback and test results were continuously incorporated into system updates and improvements. Thorough documentation was provided to guide users and facilitate the attendance-taking process.

CHAPTER 7

CONCLUSION AND RECOMMENDATION

7.1 Conclusion

The Secure Mobile Attendance System with Face Recognition and Edge Computing was developed in response to the growing need for reliable and secure attendance solutions. This research study focused on ensuring quality in the system through a comprehensive QA approach. The author's role as a Quality Assurance (QA) professional played a vital part in ensuring the system's usability and user experience, which are critical factors affecting adoption rates.

Throughout the development and implementation of the Secure Mobile Attendance System, the QA approach encompassed various testing techniques, including UI testing, usability testing, integration testing, and unit testing. The aim was to guarantee a user-friendly and intuitive system while maintaining its security and dependability. The results of the QA process were measured against a target usability score of minimum 70%, indicating the system's user-friendliness.

The findings highlight the author's successful role as a QA professional in achieving the target usability score. The system's usability was evaluated by potential users, as suggested in Appendix Table A.1. The feedback received from users indicates that the app has successfully met the minimum usability score requirement, demonstrating its user-friendly nature. However, the author claims that the mobile attendances system application has its own flaws. For example, twins with similar features could be considered the same when scanning face is ongoing, users that use sunglasses that cover their faces will not be able to scan their face, and some features that was suggested by users to be able to have data convert to excel datasheets has not deployed.

7.2 Recommendations

Based on the insights gained from the development and evaluation of the Secure Mobile Attendance System, the following recommendations are proposed for further improvement and refinement:

- **Continuous User Feedback**

  To maintain and enhance the system's usability and user experience, continuous feedback from users, especially potential users or its stakeholders should be solicited and incorporated into future updates. Regular surveys, interviews, and usability testing sessions can provide valuable insights for identifying areas of improvement and addressing user needs and preferences. This also includes updated usability testing with its current new design to be tested again to a potential user that has not tested the app before.

- **Ongoing Performance Optimization**

  Continual monitoring of the system's performance, particularly during peak usage periods, will help identify and address any performance bottlenecks. This optimization ensures that the system operates smoothly and efficiently, providing a seamless experience for users.

- **Adding more features**

  Converts that have not been implemented or tested such as admins are able to process the data and convert it to excel datasheets. Additionally, the author’s team decided to add a salary system for future work, where according to the users’ absence, the admin itself can have each user’s salary data without needing them to calculate the amount manually.

By implementing these recommendations, the Secure Mobile Attendance System with Face Recognition and Edge Computing can further enhance its usability, user experience, and overall quality. The system's success in achieving the target usability score demonstrates the author's effective role as a QA professional in ensuring a user-friendly application that meets the needs of potential users.

In conclusion, it is successfully developed and implemented a Secure Mobile Attendance System with Face Recognition and Edge Computing, focusing on ensuring quality through a comprehensive QA approach. The author's role as a QA professional played a significant part in achieving the target usability score and confirming the system's user-friendliness. The recommendations provided offer guidance for future improvements, ensuring that the system remains user-centric, secure, and efficient.

References

[1] "Glossary - Attendance Management," www.peoplehum.com. https://www.peoplehum.com/glossary/attendance-management#:~:text=Attendance%20Management%20keeps%20track%20of [Accessed 10 April 2023].

[2] Simplilearn, "What Is Quality Control: Benefits, Examples, Techniques Explained," Simplilearn.com, Apr. 26, 2021. https://www.simplilearn.com/what-is-quality-control-article#:~:text=The%20four%20types%20of%20quality [Accessed 11 April 2023].

[3] V. Nanda, "Defect Management Process in Software Testing," www.tutorialspoint.com, Nov. 30, 2021. https://www.tutorialspoint.com/defect-management-process-in-software-testing#:~:text=Defect%20Management%20is%20a%20method [Accessed 15 April 2023].

[4] R. Lynn, "The Importance of Continuous Improvement," Planview, 2022. https://www.planview.com/resources/articles/lkdc-importance-continuous-improvement/ "User Experience Is Just as Important as Technology." CTI, www.cti.com/user-experience-is-just-as-important-as-technology/ [Accessed 15 April 2023].

[5] 99designs Team, "What is UX? And why is it important?," 99designs, Jan. 14, 2021. https://99designs.com/blog/web-digital/what-is-ux/ [Accessed 17 April 2023].

[6] J. Honig, "What Makes Software User-Friendly?," start.docuware.com, Mar. 10, 2022. https://start.docuware.com/blog/document-management/what-makes-software-user-friendly#:~:text=User%2Dfriendly%20software%20is%20exactly [Accessed 17 April 2023].

[7] Usability.gov, "User-Centered Design Basics | Usability.gov," Usability.gov, 2019. https://www.usability.gov/what-and-why/user-centered-design.html [Accessed 17 April 2023].

[8] A. Fawcett, "Introduction to Human-Computer Interaction & Design Principles," Educative: Interactive Courses for Software Developers, Feb. 12, 2021. https://www.educative.io/blog/intro-human-computer-interaction [Accessed 17 April 2023].

[9] P. Iyengar, "A Complete Guide for UI Testing," www.headspin.io, Dec. 28, 2021. https://www.headspin.io/blog/ui-testing-a-complete-guide-with-checklists-and-examples (accessed Jun. 15, 2023).

[10] INTERACTION DESIGN FOUNDATION, "What is Usability Testing?," The Interaction Design Foundation, 2019. https://www.interaction-design.org/literature/topics/usability-testing [Accessed 17 April 2023].

[11] R. Awati, "What is Integration Testing (I&T)?," SearchSoftwareQuality, Jan. 2022. https://www.techtarget.com/searchsoftwarequality/definition/integration-testing#:~:text=Integration%20testing%20%2D%2D%20also%20known [Accessed 18 April 2023].

[12] Kitakabee, "How to perform Beta Testing for Applications?," BrowserStack, Aug. 17, 2022. https://www.browserstack.com/guide/how-to-perform-beta-testing#:~:text=for%20Beta%20Testing- [Accessed Jun. 23, 2023].

[13] N. Nordin and N. H. Mohd Fauzi, "A Web-Based Mobile Attendance System with Facial Recognition Feature," International Journal of Interactive Mobile Technologies (iJIM), vol. 14, no. 05, p. 193, Apr. 2020, doi: https://doi.org/10.3991/ijim.v14i05.13311[Accessed

23 June 2023].

[14] H. Anjum et al., “A Comparative Analysis of Quality Assurance of Mobile Applications using Automated Testing Tools,” International Journal of Advanced Computer Science and Applications, vol. 8, no. 7, 2017, doi: https://doi.org/10.14569/ijacsa.2017.080733 [Accessed 24 June 2023].

Appendices

Client Interview

The author’s team conducted an interview with a potential client as part of their research and development process for the mobile attendance application system. This interview aimed to gain valuable insights and feedback from the client to better understand their needs, challenges, and expectations regarding attendance management. The following are the details regarding the interview:

- When: 26 May 2023
- Where: PT. Sentra Solusi Informatika
- Interviewer: Bryan Putra Suandi
- Interviewee: Lisna Winda (HRD Manager)

Summary

The interview was split into three main parts. The first part involves the introduction of both parties involved in the interview. Bryan introduces himself. The interviewee also introduces herself as Winda and her position as Human Resources staff (HRD) in PT. Sentra Solusi Informatika, a company engaged in IT software for Travel, Tour, and Hotel.

The next part of the interview involves the main bulk of the discussion. This part is further split into:

- **Understanding the current system**

Bryan asks Winda to briefly explain the workflow of the current attendance system, which involves using a fingerprint system to collect data in the form of name, date, tap-in time, and tap-out time on a weekly basis and manually inputting it into Excel. They discuss the tolerance for lateness and the penalties for exceeding the allowed lateness. Winda explains that there is a tolerance of 7 hours of lateness per month, and employees are allowed to determine their own tap-in and tap-out times, but there are limits to the arrival and departure times that must be followed.

- **Problems regarding the current system**

Bryan explores the issues faced by users and administrators of the fingerprint-based attendance system. They discuss challenges such as power outages leading to the inability to record attendance and the manual process of handling absences and finding missing records. Winda mentions that many users forget to tap in or tap out, and when employees are absent, the search and handling process is still done manually.

The final part of the interview involves demoing a prototype of the mobile attendance application being developed by the author and his team and receiving the interviewee's thoughts about the application. The following are the main points of this section:

- The application is beneficial as it helps to remind users to tap in/ tap out in the form of notifications.
- The application lacks several features such as the aforementioned tolerance work time feature as well as leave/ permission quota to facilitate their needs.
- The interviewee expresses concerns regarding the possibility of employees tapping in/out from outside of the working environment.

In conclusion, the interview revealed several client needs. The potential client faces challenges with the current fingerprint-based attendance system, including power outages affecting attendance recording and manual processes for handling absences. As a solution, the proposed mobile application can provide reminders to employees to tap in and tap out through notifications on their smartphones. This will facilitate timely and accurate attendance recording. Additionally, suggested additional features include monitoring lateness tolerance usage during the month and tracking leave, permissions, and sick days. These features will assist in more efficient administration and tracking of employee attendance.

Interview Transcript

The full interview transcript (in Indonesian) can be found below:

Table A.1: Interview transcript

Bryan	**Ka Winda**
Selamat siang kak, terima kasih telah menyepatkan waktunya ya untuk bisa wawancara dengan saya, untuk kepentingan skripsi group saya, Sebelum saya mulai, mungkin kaka bisa perkenalkan terlebih dahulu, nama kaka, dan posisi kaka di perus-ahaan ini	Perkenalkan nama saya Winda, saya sekarang di posisi staff HRD, kita ber-gerak di bidang IT software untuk Travel, Tour, Hotel.
Maaf kak, nama perusahaannya tadi apa ya?	PT. Sentra Solusi Informatika
Pertanyaan pertama, nah kak, kalo boleh tanya tuh, sekarang di perus-	Kita pake system fingerprint, jadi setiap seminggu sekali saya Tarik data. Terus

ahaan kaka ini data absensinya itu pake sistem kyk gimana ya kak? bisa dijelasin alurnya secara sekilas saja	di situ ada nama, tanggal, jam tap in dan tap outnya, terus saya input ke excel sendiri. Di situ, jadi, keliatan misal ada yang terlambat berapa menit, itu mengurangi toleransi jam yang satu bulan, kita kan ada toleransi keterlambatan 7 jam, jadi setiap 1 bulan gk boleh lebih dari 7 jam.
Berarti dari sini, kaka itu di sistem absensinya itu ada toleransi jam, berarti karyawan itu perbolehkan untuk pakai, dan tap in dan tap outnya terserah mau kapan ya?	Kita itu jam masuknya kan 8:30 dan pulangnya 17:30. Jadi setiap hari itu, kita hanya bisa membayar keterlambatan, 30 menit, walaupun kita misal dateng terlambat 1 jam, di hari itu, kita hanya bisa bayar 30 menit, dan 30 menitnya itu mengurangi toleransi yang 7 jam itu
Kalo misal masalah, kan kaka pake sistem sidik jari kan ya?	Iya
Nah, kalo kaka misal dari segi user itu, untuk masalah sistem absen pake mesin jari ini, apakah biasanya ada kendala atau keluh kesal yang sangat sering terjadi gitu ya, di sisi user dan admin tapi di sisi adminnya itu, sebut	Kita kebetulan nih sering mati lampu, nah kalo mati lampu, gk bisa absen. Dan dari sisi admin, kita gk bisa tarik data apapun, karena banyak yang gk bisa absen. Jadi admin harus manual input datanya.

saja yang bisa di kaitkan dengan usernya, tidak harus se-detail untuk admin saja ? Soalnya kita perspec-tivenya mau membantu experience user dan admin yang masih bisa di kaitkan dengan user gitu.	
Ada lagi gk selain itu?	Kebanyakan user sering lupa tap in tap out. Terus misal kalao mereka ada pu-lang cepet atau datang terlambat, mere-ka sering lupa tap in atau tap out. Misal ada yang gk masuk, contoh 3 hari. Tapi kan dari awal, saya input rekap absennya itu seminggu sekali. Misal dalam seminggu itu ada beberapa orang gk masuk. Itu saya harus cari manual, caranya dengan Tarik data dari fingerptrint itu, saya cariin satu2. User ini 3 hari gk ada, kemana? Saya paling, bikin keterangan manual, Contoh “User ini gk masuk, izin atau sakit “gitu
Tadi kan, sebelumnya kaka udah mencoba sekilas prototype aplikasi	Cukup membantu untuk yang sistem tap in dan tap out. Karena itu kan sistem

kita. Nah, menurut kaka itu, apakah aplikasi kami itu bisa menyelesaikan, masalah2 dan kendala yang kaka barusan sebut? Dari sisi admin dan dari sisi user	mobile, jadi misal ada yang lupa tap in / tap out kan. User bisa tau, ada muncul warning, di hp mereka masing2, dan mereka segera tap in / tap out nya gitu, mereka bisa pencet sendiri lah tanpa kita manualin dari sisi admin.
Mungkin dari sisi kayak, kalao misal, kan kalian pakai sistem mesin ya ?	Iya
Jadi kan mungkin kaka ngomong tadi seperti, mati lampu atau kendala yang lain, kan jadi tanggung jawab pemilik smartphone.	Iya itu mempermudah. Dan untuk waktunya mungkin bisa di buat general. Kan hp masing2 bisa beda waktunya. Ikutin jam Dunia
Apakah ada fitur2 lain yang mungkin kaka merasa perlu sebagai user dan sebagai admin yang bisa kita pakai yang belum di aplikasi kami? Yang bisa kita implementasikan	Fitur tambahannya mungkin bisa di tambah, yaitu yang kita kan ada toleranasi 1 bulan, 7 jam. Nah, mungkin bisa di tambah fitur itu, jadi kita dari sisi admin, bisa liat user ini, pemakaian selama beberapa hari sudah terpakai berapa jam. Terus ada fitur lagi tambahan mengenai cuti, sisa cuti, saldo cuti. Pemakaian selama bulan ini berapa, terus pemakaian selama satu tahunnya berapa.
Pertanyaan terakhir, kalao menurut kaka itu, kaka kebayang gk, kalao	-nya Mungkin bisa tap in atau tap out di sembarang tempat, di luar jam kerja.

aplikasi kami itu mungkin menonjol-kan masalah2 baru kayak contoh, misalkan pakai sistem sidik jari ada + -. Kalao misal datanya kan pasti, datanya gk bisa minta temennya tap innya, menurut kakak itu – apa saja yang bakal bsia muncul kalao kita pake aplikasi hpini, dengan datanya pakai muka, sebagai penanda karya-wan masuk?	Misal kita kan ada visit ke client, kalao misal absensinya by mobile kyk gitu, mereka bisa, abis dari client, nongkrong dulu di luar, terus nunggu waktu jam pulang. Tap in tapoutnya teserah mere-ka gitu, gk di dalam lingkungan kantor, tidak dalam aktivitas kerja gitu.
Ada lagi gk kak?	Terus kalo ini, registrasinya pakai apa ya?
Kalao kita sih untuk sekarang, jadi kita kebayangnya tuh, setiap kali ada karyawan baru, dari database perus-ahaannya itu, perusahaanmya nge-generate username sama password untuk masuk ke aplikasi kami. Soal-nya kalao kita open to public untuk akses register kan, nanti banyak orang bisa register sembarangan. Nah kita buat khusus untuk perusahaan nih, jadi username sama password-nya untuk loginnya itu, nanti setelah perusahaan nge hire karyawan, baru	Kalao misal login itu. Misalkan ada yang re-sign. Login itu bisa di rename ke orang lain? atau bikin user baru? Us-er ini yang bersangkutan udah resign bisa di timpah?

di kasih ke karyawan baru ini. Jadi karyawan baru ini tinggal pake username password itu untuk pake login ke aplikasi kami.	
Tergantung perusahaan, perusahaan itu mau mencatat karyawan ini pernah kerja di sini atau gk. Kalao perusahaan butuh data itu, kita gk timpah, kita buat baru. Cuman yang lama, statusnya resign, tp kalao perusahaan butuh di timpah, bisa	Kalao misal ada yang hari ini ada yang gk ada tap in atao tap out, bisa di kasih warning. Misal, pagi itu kan kita jam 8:30, jam 10 ud di kasih warning, jadi file re-portnya, hari ini yang gk masuk ataupun yang belum absen itu siapa aja. Dan tapoutnya di warningin berarti sekitar jam 18:00
Ini kasih warningnya ke usernya ya?	Kalao bisa, iya
Kalao misal telat masuk, maksimum 1 jam setengah ya, jam 10 berarti?	Iya
Kalao telat pulangnya harus jam 6 ya?	Iya
Itu saja yang kepikiran ya?	Baru itu aja
Mungkin kaka sebagai user nih, kaka merasa keberatan untuk memakai aplikasi ini tuh di sisi apa sih bi-asanya? Kalao mungkin kaka bisa bayangkan	Kalao dari sisi admin, program kami kan belum keliatan pemakaian per hari, dia, pemakaian toleransinya berapa la-ma, belum keliatan.

	Terus belum keliatan juga, untuk pemakaian cuti, izin, sakit, itu biasanya di bedain, belum ada fasilitas itu kan ya, secara detail, perusernya?
Beberapa ud ada sih, tp yang kaka mention belum ada	
Udah itu saja sih kak, pertanyaan wawancara dari saya, terima kasih ya sudah meluangkan waktu untuk menghadiri wawancara ini	Iya sama2, semoga membantu