Computer Science and Artificial Intelligence Department
College of Computer Science and Engineering
Jeddah University, SAUDI ARABIA

Senior Project

Hasanah

By

Bylsan Alsulami 2110091
Shadan Almuzil 2114431
Jude Alaki 2112084
Lyan Baamer 2111469
Dalia Alharthi 2116684

Supervisor: Dr.Samia Snoussi

Submission date: 29 November 2024

Dedication

This project is dedicated to all Muslims seeking to perfect their prayer and deepen their connection with their faith. To the new reverts embracing Islam, to the children taking their first steps in prayer, and to every individual striving for excellence in worship, this work is for you. It is also dedicated to our families, whose constant love and encouragement have been our foundation, and to everyone who supported us in turning this vision into reality.

Acknowledgment

First and foremost, we thank Allah (SWT) for His endless blessings and guidance throughout our journey.

We extend our deepest gratitude to our supervisor, Dr. Samia Snoussi, whose invaluable support, expert guidance, and belief in our abilities have been instrumental in the success of this project. Your mentorship has inspired and motivated us at every step, and for that, we are forever grateful.

To our families, thank you for your patience, encouragement, and unwavering faith in us during this journey. Your support has been our constant source of strength.

Finally, we express heartfelt appreciation to our team members for their collaboration, dedication, and hard work. Together, we transformed an idea into a meaningful and impactful project.

Table of Contents

List of Figures

Chapter 1

1.1 Introduction

In this first chapter, we're describing our project that mixes new technology with the Islamic practice of Salah, which is a fundamental pillar of Islam. Salah is not just about spirituality; it also involves specific physical movements. Doing these movements correctly is very important for the prayer to be spiritually meaningful and for keeping the person praying physically healthy. However, mastering these poses can be hard without proper guidance. Our project will use deep-learning technology to help people improve their prayers.

1.2 General Context

The system utilizes deep learning to analyze users' praying postures, pinpointing inaccuracies, or areas for improvement. once the deep learning has identified an incorrect posture the system then guides the user to instructional videos that are part of our dataset. These videos, carefully selected and included in the system's extensive library, demonstrate the correct way to perform prayer postures. Creating a personalized learning experience. This system aims to help people on their spiritual journey by using new technology to improve praying

1.3 Problem definition

The main problem among prayers is the uncertainty regarding the correctness of their prayer performance like bowing down, rising after bowing down, sujud, and mistakes of the hands during prayer and while raising hands at the beginning of prayer, leading to a lack of confidence in their practice. There exists a pressing need for a sophisticated solution to provide users with a reliable means of verifying the accuracy and adherence to established principles in their prayers.

1.4 Aims

1. We aim to develop a smart system, to address the issue of mistakes in prayer, ensuring adherence to Islamic principles and rituals and accommodating variations in individual

practices and prayer styles.

2. Our goal is to provide interactive notes and feedback to guide prayers in correcting specific mistakes and improving their prayer performance, promoting a deeper sense of connection and humility in worship.

3. we aim to ensure that the system enhances the understanding of prayer movements and fosters mindfulness in worship, thereby improving the overall quality and spiritual experience of prayer sessions.

4. we aim to develop a system that caters to the needs of young Muslims and r to Islam, providing them with a comprehensive tool for learning and practicing prayer. This system will offer personalized guidance, interactive feedback, and educational resources tailored to their level of understanding.

1.5 Objectives

- Provide a platform for young Muslims or new converts to Islam to learn prayer rules.
- Automatically detect prayer posture-related errors by mobile application.
- Help people correct their prayer posture errors by themselves.
- Help people improve the performance of their prayer.
- Provide a user's history section to allow users to check their previous sessions.

1.6 Proposed Solution

The system offers the finest possible prayer experience for Muslims and new reverts to Islam. The system uses computer vision to detect and correct errors in the user's prayer positions. It also utilizes CNNs and deep learning algorithms to analyze the videos and detect movements. The system comprehends and analyzes the visual data from the videos by training a model on a large dataset that comprises correct and incorrect positions. First, the movements in the video are tracked to identify all incorrect movements. Next, correction videos are displayed that demonstrate the proper form for each position during prayer and address typical mistakes made during prayer. Furthermore, it is intended that the system will have a user interface (UI) that is clear, concise, and easy to navigate through

1.7 Report outline

The content will be organized into four chapters. Chapter 1 will provide a comprehensive overview, covering the General Presentation. In Chapter 2, we will conduct a comparative

analysis of Related Work. Chapter 3 will delve into a General Analysis, focusing on the requirements necessary for this project. Lastly, Chapter 4 will be dedicated to design Development, exploring its various aspects.

1.8 Project plan

General Presentation

- Day 1-2: Introduction
- Day 1-2: General Context
- Day 1-2: Problem Definition
- Day 1-2: Aims
- Day 1-4: Objectives
- Day 1-4: Proposed Solution
- Day 1-4: Report Outline
- Day 1-3: Project Plan

Related Works

- Day 1-2: Introduction
- Day 1-2: Background
- Day 1-4: Existing System
- Day 1-4: Comparison

General Analysis

- Day 1-2: Introduction
- Day 1-3: Requirements Gathering
- Day 1-2: Functional Requirements
- Day 1-2: Non-Functional Requirements
- Day 1-2: Conclusion

total days: 25

Figure 1.1: Project plan

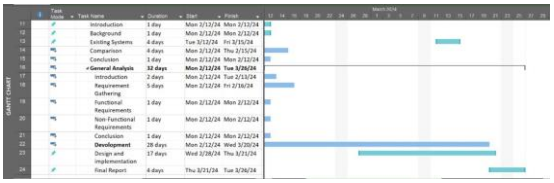


Figure 1.2: Project plan

Chapter 2

2.1 Introduction

This chapter explores five distinct systems, each developed with its unique technological approach and offering a range of services. We will extensively examine the functionalities provided by each system, followed by a comprehensive overview. Toward the end, we will compare these systems with our own, emphasizing the distinctive features and diversity they offer. The chapter sequence includes Background, Overview of Current Systems, and Comparative Analysis.

2.2 Background

One of the common issues not addressed, given the rapid advancement of technology and the variety of technologies employed in the religious sphere, is elaborating the proper form of prayer straightforwardly to both new Muslims and Muslims seeking to enhance their prayer experience. We provide the Hasanah system as a solution to this issue and to enhance the prayer experience. Our system provides several features to improve the prayer experience and mitigate the anxiety that comes with praying improperly. These features include the following:

- Computer vision: Object tracking and detection are one of the main uses of computer vision in video analysis. This involves locating and monitoring particular prayer positions inside a video frame. Computer vision uses scene understanding and parsing to precisely identify and track positions to interpret and comprehend the content of a video scene.

- Convolutional Neural Networks (CNNs): The use of CNNs to identify and analyze user prayers is a crucial aspect of deep learning for video analysis. The visual data in the videos is processed by CNNs. CNNs can be used to extract features from individual video frames in the context of video analysis, which aids in the identification of patterns and movements.

- Feedback Display: By playing a correction video showing how to perform the position correctly.

- Dataset Creation: The system will be based on a dataset of videos created especially for the system to establish the accurate criteria for each correct position in prayer. This will help to check prayer position errors and give video correction feedback.
- User-friendly system: The success of this project depends on creating a user-friendly system. This covers the design of the user interface (UI). A system's user interface (UI) design greatly influences user experience and engagement. A simple, clear, and easy-to-navigate interface can make it easier for users to learn how to use a system and motivate them to continue using it.
- Data Security and Privacy: The system records videos and sensitive user data, therefore, one of the objectives of this system is to guarantee data security and privacy via applying user consent protocols. The system makes sure that authorization and authentication procedures are in place to protect user data by requiring email and password. As well as ensuring the privacy of the users by keeping the users' videos private and not using them for any other purpose without consent.

2.3 Existing Systems

In the following subsections, we will provide a detailed description of five competing systems. We will delve into their capabilities, performance, and limitations, drawing comparisons to our suggested system:

2.3.1 First system: Salat

Salat project aims to develop an artificial intelligence (AI)-based assistive framework to help Muslims perform Salat (Islamic prayer) with accurate postures, the project leverages Convolutional Neural Networks (CNNs), specifically the YOLOv3 algorithm, to identify and evaluate the correctness of the prayer's postures. By creating a dedicated dataset that captures the three fundamental postures of Salat—Standing (Qiyam), Bowing Down (Ruku), and Prostration (Sujud). This technology promises to guide worshippers, especially beginners and children, in performing their prayers correctly, ensuring they meet the spiritual and formal requirements of Salat.

Pros:

1. Educational Tool: Salat offers a way for newcomers and children to learn the correct postures required in Salat, potentially speeding up the learning process.

2. Accuracy: Deep learning models specifically Convolutional Neural Networks (CNN) and YOLOv3 neural network, particularly with a dataset specifically designed for Salat postures, can achieve high accuracy in recognizing and differentiating between the various postures

3. Accessibility: Provides a way for individuals who may not have access to a physical instructor to learn and correct their Salat postures.

Cons:

1. Privacy concerns: The use of cameras to capture and analyze prayer postures may raise privacy issues, especially if the system is not fully localized or if data is stored improperly.

2. limited correction: The system only corrects three types of prayer postures. It might not cover all the wrong ways people might pray. If someone's prayer mistakes are in different postures, they might not find the help they get useful or related.

3. No feedback: lack of corrective video feedback which is particularly beneficial for beginners and children who learn better through seeing exact demonstrations of the correct postures.

2.3.2 second system: Istaqim

Istaqim is a mobile AI assistant application that utilizes deep learning techniques. The primary objective of this mobile app is to guide worshippers by accurately identifying incorrect postures in their prayers, assessing the corresponding errors, and offering just three corrections for ruku posture, raising posture, and sujud posture. Specifically, the project focuses on employing Convolutional Neural Networks (CNN) and the state-of-the-art YOLOv5 neural network for posture detection.

Pros:

1. Accurate Posture Identification: utilizes deep learning techniques, specifically Convolutional Neural Networks (CNN) and YOLOv5 neural network, to accurately identify incorrect postures in prayers. This ensures that users receive reliable feedback on their postures.

2. Error Assessment and Corrections: The system not only detects incorrect postures but also assesses the corresponding errors. It offers three specific corrections for ruku posture, raising posture, and sujud posture. This targeted approach assists users in improving their prayer postures effectively.

3. Utilization of Deep Learning: By employing deep learning techniques, the system leverages the power of AI to analyze visual data and provide real-time guidance. The use of CNN and YOLOv5 enhances the accuracy and efficiency of posture detection.

Cons:

1. Limited Corrections: The system offers only three corrections for specific postures. It may not address the full range of potential errors in prayer postures. Users with different posture mistakes may find the suggestions insufficient or irrelevant

2. Reliance on Visual Data: Since the system relies on visual data, it may face challenges in scenarios with poor lighting conditions or unfavorable camera angles. This could result in inaccurate posture detection and subsequent guidance.

3. Privacy concerns: privacy concerns may arise due to the recording of users' video footage during prayer sessions. Users might be uneasy about the potential storage and handling of these video recordings.

2.3.3 Third system:Kemtai

Kemtai is an exercise platform that uses AI to act as a personal trainer. It uses computer vision and complex neural network algorithms to provide real-time analysis and feedback. There is a multitude of exercises to choose from. There are 111 points in your body the software analyzes to check the form. It doesn't require any additional hardware, the only hardware that's used is the camera. Kemtai analyzes the 111 points in the body and compares them to the form of the exercise chosen, it provides a real-time percentage of the correctness and gives relevant commentary feedback. The main concern of many users is privacy, the software doesn't record the session. Rather, it records a mockup of the 111 points in the body.

Pros:

1. Versatility: the app has a large dataset of physical movements, ranging from physical therapy, elderly care, high-intensity training, and more.

2. Real-time feedback: it uses neural network algorithms to give feedback in the form of percentages and correcting commentary.

3. Privacy: Kemtai doesn't record the screen, rather it creates and saves a mockup of the movements using relevant points in the body.

4. Accessibility: the app doesn't require additional hardware; it uses only the camera on the phone.

Cons:

1. It needs a lot of space to analyze the full body.

2. The app is suffering from occasional bugs.

3. Inconvenient to move the device mid-workout to see different parts in different movements.

2.3.4 Fourth system: Vay Sport

VAY Sports is a comprehensive mobile application designed specifically for athletes, allowing them to meticulously monitor the correctness of their movements, postures, and the duration of each sporting activity. The technological framework of VAY Sports, leverages advanced artificial intelligence techniques such as image and pattern recognition, along with deep learning technology, to achieve goals and objectives. In VAY Sports, the primary focus lies in detecting and correcting errors in the sports movement and analyzing and optimizing athletes' movements during various sporting activities. The application utilizes sophisticated image recognition-based tracking through the device's camera to monitor athletes' movements in real time. This allows for precise tracking and analysis of movements such as running, jumping, or weightlifting, providing athletes with valuable insights into their technique and performance. Furthermore, the VAY Sports application evaluates the accuracy and duration of movements to provide users with indicators of their performance. Where it focuses on analyzing athletic performance, ensuring precise tracking, and improving overall application efficiency and accuracy.

Pros:

1. The app uses human motion analysis to track your movements and provide you with feedback and guidance from professional trainers.

2. The app has an extensive and growing library of exercises that it can analyze.

3. The app is hardware-independent and markerless, meaning you can use any camera device and train without any sensors or markers.

4. The app provides real-time audio feedback from your coach on your exercise execution.

5. The app has statistics to show your progress and compare your performance with previous sessions.

6. The app detects the mistake in your movement and gives you the right one.

Cons:

1. The app may not work well in low-light or crowded environments.

2. The app may not recognize some complex or uncommon movements.

3. The app may have some bugs or errors that affect the user experience.

4. privacy concern: Analyzing videos to correct sports postures may inadvertently capture intimate details of an individual's body mechanics and postures, potentially making users uncomfortable about the level of scrutiny.

2.3.5 fifth system: Kaia Health

Kaia Health is a mobile application that users can access through using a smartphone or tablet. It works on both IOS and android operating systems. It is a health App that targets patients with chronic pain. It's objective is to make effective therapy accessible at anytime anywhere. It takes an integrative, evidence-based approach to pain, based on multimodal rehabilitation (MMR) which combines guided physical exercises, relaxation techniques, and pain education. It offers a self-management program for MSK (Musculoskeletal) care. The application dynamically adjusts and customizes itself to the demands and activities of the user. In order to assist patients in achieving MSK health, Kaia also helps users to get in touch with certified health coaches. Kaia Health's AI algorithms can assess performance and direct users through training by analyzing movements in real time using a smartphone's selfie camera. The pocket-sized computer vision technology from Kaia Health analyzes motion. Computer vision technology can monitor a user's motions in real time to evaluate exercise performance and deliver corrective real time voice feedback pinpointing the wrong forms and correcting them during the training session. The App works as a personal coach at anytime and anyplace. Kaia health can this by tracking body movements, recognizing body landmarks in a picture or video to get a sense of the posture and movement of the body, making sure that the user is exercising with the correct form.

Pros:

1. Kaia health computer vision can work more accurately than a depth sensing camera.

2. Wearable technology is not necessary for Kaia Health.

3. Kaia motion coach can track the full body instead of one joint angle.

Cons:

1. The application is only available in the US and EU.

2. Privacy concerns: The application gathers personal information, video, and biometric information. The processing of this data may give rise to privacy issues; users may not consent to the storage of their personal data or the handling of these data.

3. The application has inside purchased.

4. The Kaia app requires a stable connection to the internet.

2.4 System comparison with existing systems

The following are the criteria we have chosen and why we deemed them important, to compare our system with five other systems alike:

1. Prioritizing user interface: an easy-on-the-eye user interface is crucial for the user experience.

2. Video dataset: since our system detects errors from videos, the dataset too must match.

3. Detect more than three: to have a broader spectrum of errors to detect.

4. Video feedback: a more elaborative explanation of how to correctly fix the error in the form.

5. Privacy: important to ensure the privacy of the users to ease their minds.

CRITERIA	PRIORITIZE UI	VIDEO DATASET	DETECT MORE THAN 3	VIDEO FEEDBACK	PRIVACY
SALAT	×	×	×	×	×
ISTAQIM	×	✓	×	✓	×
KEMTAI	✓	✓	✓	✓	✓
VAY SPORT	×	✓	✓	✓	×
Kaia Health	✓	✓	✓	×	×
HASANAH	✓	✓	✓	✓	✓

Figure 2.1: Comparison of related work and our system

During our research and comparisons, we have found that most of the systems lack one or more of the criteria we have specified, except for our system and Kemtai. The differential factor between the two systems is that Kemtai focuses on the fitness domain whereas our system focuses on the religion domain.

2.5 Conclusion

In conclusion of this chapter, we have thoroughly covered the scope of the proposed solution, delving into the key studies and related systems in the form-correcting domain. We have provided insightful details on their functionality and features. Additionally, we have included a comprehensive comparison table that juxtaposes our current system with others in the field. Moving forward, the subsequent chapter will focus on the requirements-gathering process. We will outline the methodology employed for gathering both functional and non-functional requirements for the system. This will provide a comprehensive understanding of the necessary specifications and objectives that will guide the development and implementation of the solution.

Chapter 3

3.1 Introduction

This chapter aims to define the functional and non-functional requirements of our system, as well as the appropriate techniques to evaluate its limitations. Additionally, we will outline all the proposed objectives for the system.

3.2 Requirements gathering

We have launched an online survey, we got 33 responses, and it was open for three weeks to gather input from our target audience, Muslims, to study our problem space.

The questions are:

1. What is your age group?

- **Reason for asking:** We asked this question to understand the relation between the effect age has on the answers.
- **Summary of responses:** The majority of participants, scoring 74.2%, were between the age 18 to 24.

Figure 3.1: Question 1

2. Have you ever wondered if your prayer is correct?

- **Reason for asking:** We have asked this question to gauge the reoccurrence of this problem. To find a solution to potentially aid in the ease of this monumental act of worship.
- **Summary of responses:** 97%have answered yes.

Figure 3.2: Question 2

3. If you want to learn how to correctly perform a prayer position what's your usual way to do so?

- **Reason for asking:** We have asked this question to choose a method in which we will correct the prayer form, in a way that is close to how people already learn.
- **Summary of responses:** Almost 50% of the answers learn by YouTube videos, and 27% from friends or peers.

Figure 3.3: Question 3

4. Would you use a system that would check your prayer form?

- **Reason for asking:** We have asked this question to gauge the demand for our proposed system within the community and to know beforehand that it will be used.
- **Summary of responses:** Nearly 85% have answered yes.

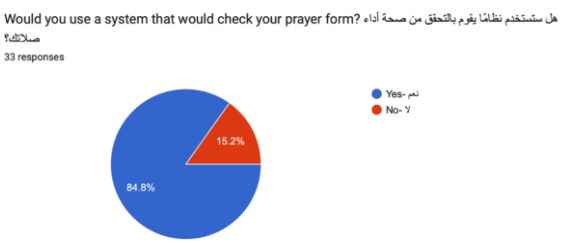


Figure 3.4: Question 4

5. Which prayer position is important for the system to include?

- **Reason for asking:** We have asked this question to see which prayer position people need more help in, so it would initially be our main focus.
- **Summary of responses:** 33% care more about takbirat al-ihram, following with 27% for ruku, then sujud and lastly raising from ruku.

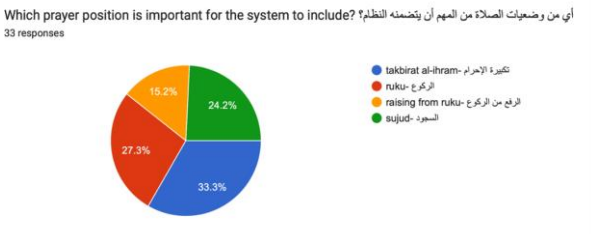


Figure 3.5: Question 5

6. Do you think it's useful to have a system that evaluates your prayer form and gives feedback on how to correct it?

- **Reason for asking:** We have asked this question to gauge the public opinion of whether our system is important generally.
- **Summary of responses:** 97% have answered yes.

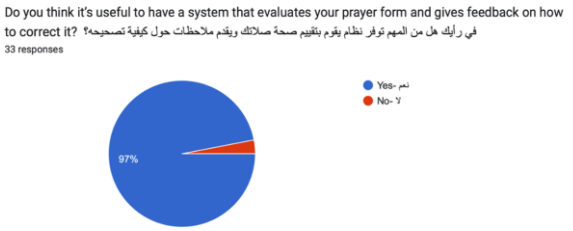


Figure 3.6: Question 6

7. How would you prefer the feedback to be in the form of?

- **Reason for asking:** We have asked this question to know how best our feedback should be for our intended users.
- **Summary of responses:** The majority prefers the feedback to be in video form.

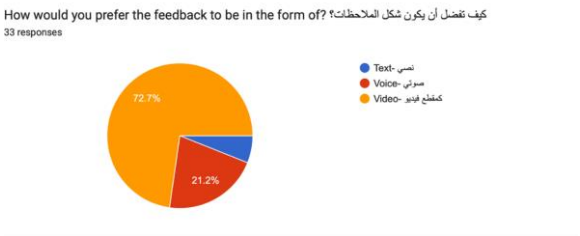


Figure 3.7: Question 7

8. Would you be reluctant if you had to upload your video for evaluation?

- **Reason for asking:** We have asked this question to know if uploading a video for the evaluation would be a big concern and if so, prioritize privacy.
- **Summary of responses:** 58% would be reluctant to post themselves.

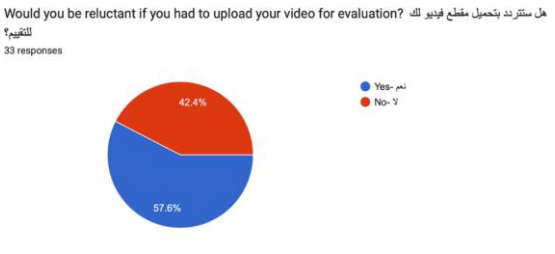


Figure 3.8: Question 8

9. In your opinion, what aspect is the most important to have in the system?

- **Reason for asking:** We have asked this question to understand where our resources should lean more towards.
- **Summary of responses:** Both security and privacy have gained the same importance with 24 votes and an inviting interface had 15 votes.

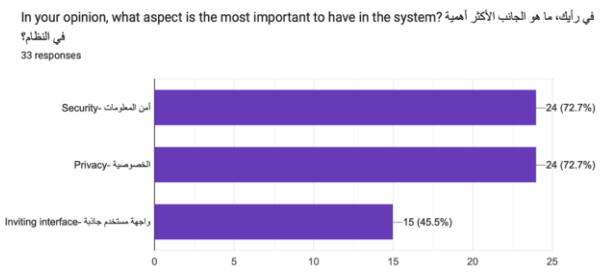


Figure 3.9: Question 9

10. What services would you like to see in our system?

- **Reason for asking:** We have asked this question to take feedback to improve our system based on our users' preferences and suggestions.
- **Summary of responses:** The most common are the following:
  - To include different languages.
  - To be inclusive to different age groups such as elderly people or kids.
  - To have an easy and inviting interface.

3.3 Functional requirements

- Learning prayer posture rules

  - Identifying the prayer position.
  - Testing if there is an existing error.

- Error detection

  - Analysis of the video.
  - Detect kind of error.

- Error representation

  - Provide the position of the error.

- Correction of the incorrect position

- History section presentation

3.4 Non-functional requirements

- Privacy: prioritize user privacy and process video data securely on the device.

- Performance: provide real-time feedback and efficient video frame processing

- Usability: The app should have an intuitive user interface and be accessible to users of all ages and abilities

- Reliability: The app should be stable and reliable, with minimal downtime or errors.

- Security: Implementing robust security in our app is crucial to protect user data and ensure trustworthiness

3.5 Use-Case diagram

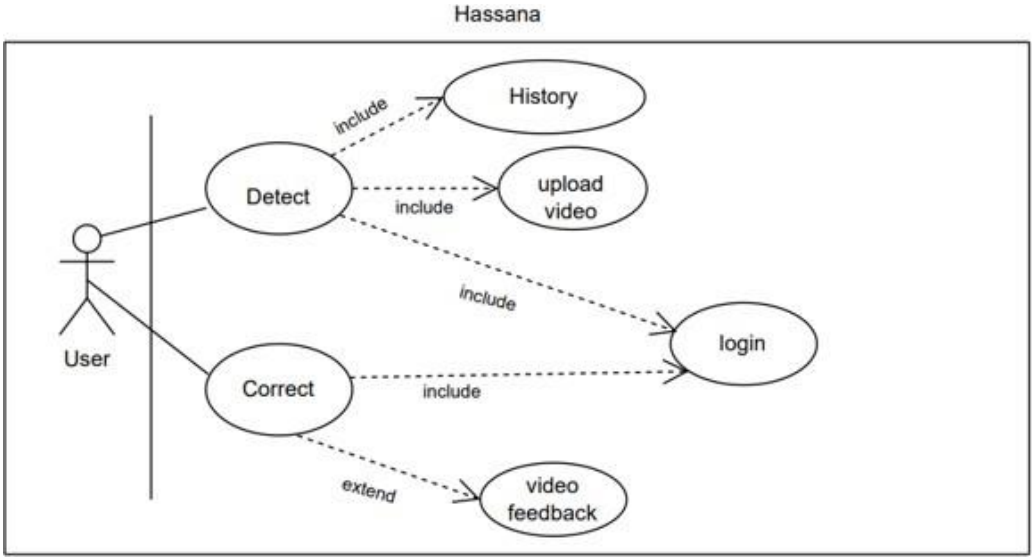


Figure 3.10: Use-Case diagram

Element	Description
User	Represents an individual who interacts with the software application.
Login	Represents the process of entering the user's login credentials (username and password).
Detection	Represents the state where the application checks for errors in prayers postures.
Correction	Represents the state where the application provides videos feedback to the user of their incorrect posture.
Upload Video	Represents the feature or functionality within the application that allows users to upload videos and to transfers the video file from the user's device to the application's server.
History	Represents the feature or functionality within the application that allows users to view profile and see their previous postures attempts and check their progress.
video feedback	Represent the process of providing for user the correction video

Table 3.1: Description of key elements depicted in the use case diagram.

3.6 Sequence diagram

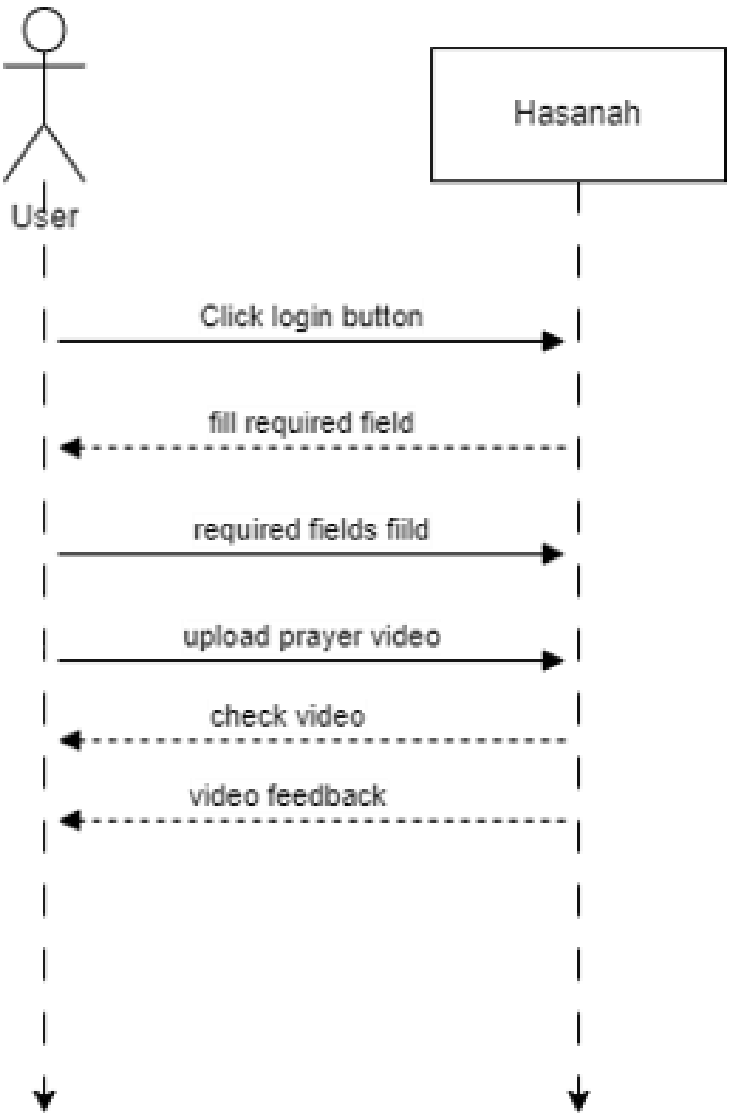


Figure 3.11: Sequence diagram

This diagram appears to be a dynamic diagram that illustrates the interaction between a User our system Hasanah. The key steps depicted are:the User clicks the login button then the User fills in the required fields then the User uploads a prayer video, then the system checks the video lastly the User receives video feedback.

3.7 Conclusion

In this chapter, we discussed the methodology employed for requirements gathering and analysis, as well as the functional and non-functional requirements of the system. A concise description of each requirement was provided. Moving forward, the next chapter will offer an overview of the dashboard's fundamental functionalities, use cases, design, and accompanying diagrams.

Chapter 4

4.1 Introduction

In this chapter, we will cover four key aspects of system design: static design, architectural design, dynamic design, and user interface design.

4.2 Design of the Static Aspect

This class diagram represents the key components of our project, which focuses on detecting errors in prayer postures and providing feedback. The diagram includes the following classes:

- **User**: This class represents the user of our system and includes functions for logging in and uploading videos.
- **Video**: The Video class is responsible for managing video-related operations, such as retrieving videos.
- **Interface**:The Interface class encapsulates the functionalities related to user interaction, including login, registration, uploading videos, and retrieving feedback
- **Compare**: The Compare class handles the comparison of prayer postures and provides the correct posture feedback.

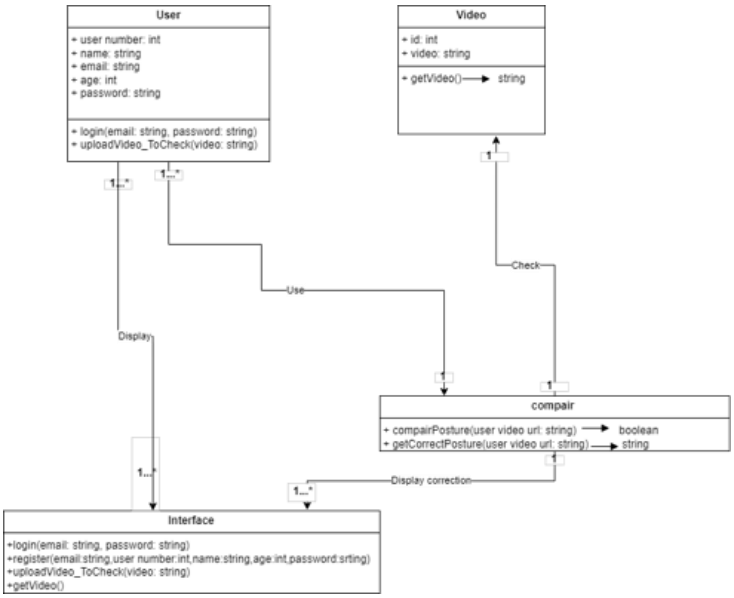


Figure 4.1: class diagram

4.3 Design of the Architectural Aspect

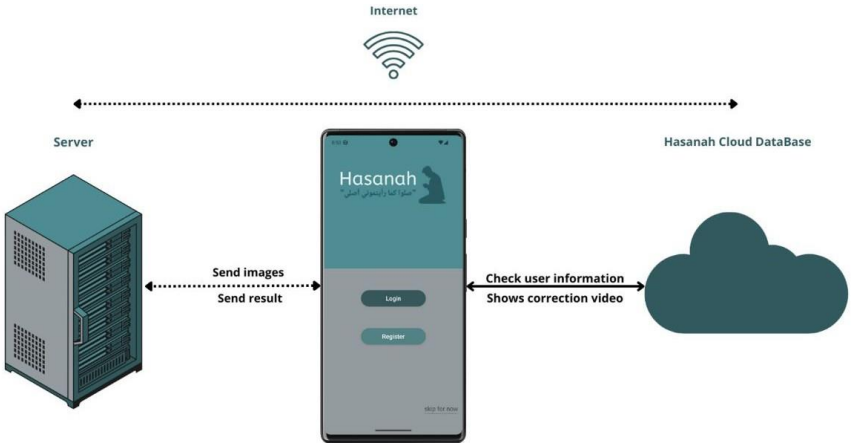


Figure 4.2: Architectural Design

As shown in Figure 4.4, This architecture design represents the ”Hasanah” system, which is a mobile application for enhancing prayer experiences by analyzing and correcting prayer postures. Here’s a detailed explanation of the components and their interactions:

i. **Mobile Application**: The user interacts with the ”Hasanah” mobile application, which serves as the interface for:

- Logging in or registering new users.
- Uploading images of prayer postures for analysis.
- Receiving results and accessing corrective videos for prayer postures.

ii. **Internet**:

- The internet acts as the communication bridge between the mobile application, server, and database.
- Data such as images, results, user information, and correction videos are transmitted over the internet.

iii. **Server**: The server processes the images uploaded by the user. Its roles include:

- **Receiving Images**:Uploaded images of prayer postures are sent from the app to the server.
- **Analyzing Postures**:The server runs computer vision algorithms (e.g., CNN-based models) to detect errors in the prayer posture.
- **Returning Results**:Once the analysis is complete, the server sends the results back to the app, indicating whether the posture is correct or not.

iv. **Hasanah Cloud Database**: The cloud database stores critical information needed for the system to function:

- **User Information**:Details about registered users, such as login credentials.
- **Correction Videos**:A repository of instructional videos to guide users in correcting their prayer postures.
- **System Logs**:Information about analyzed images and their corresponding results.

This architecture ensures a smooth flow of information from the user’s input to the final feedback, enabling the system to provide accurate and helpful guidance on proper prayer postures.

4.4 Design of the Dynamic Aspect

The sequence diagram illustrates the interaction between an actor and three objects: **Interface**, **Hasanah System**, and **Video**.

- **Actor**: Represents the user interacting with the system.
- **Interface**: The user interface through which the actor interacts with the system.
- **Hasanah System**: The core system responsible for processing and analyzing data.
- **Video**: Represents the video data being handled within the system.

The sequence diagram details the flow of actions starting from the user input through the interface, followed by processing within the Hasanah System, and interaction with the video object. This includes capturing video, sending it for analysis, receiving feedback, and displaying results to the user.

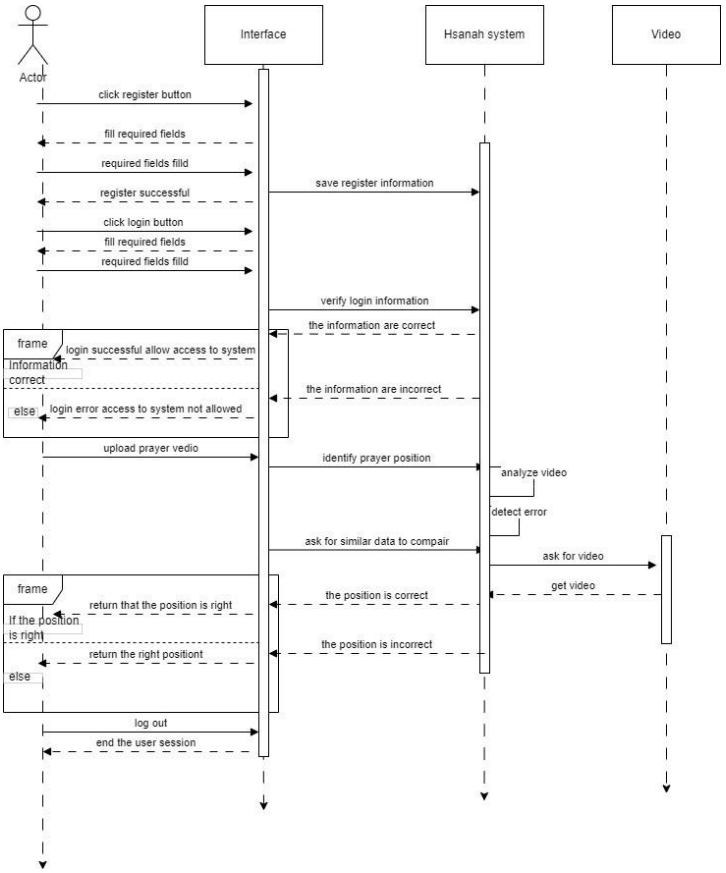


Figure 4.3: Detailed Sequence Diagram

4.5 User interface design

The following image showcases the user interface design created using Figma. This design emphasizes a user-friendly and intuitive layout, ensuring that users can easily navigate and interact with the Hasanah System. Key features of the interface include clear navigation menus, accessible controls for video capture and playback, and informative feedback displays. The design aims to provide a seamless experience for users, enhancing their interaction with the system and ensuring efficient completion of tasks.

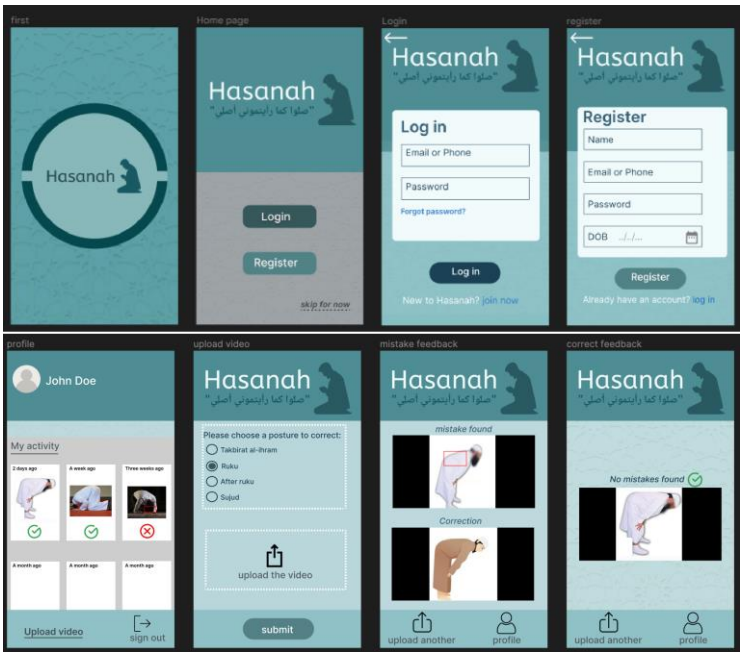


Figure 4.4: User interface design

4.6 Conclusion

In this chapter, we have comprehensively explored four critical aspects of system design: static design, architectural design, dynamic design, and user interface design. Each of these components plays a vital role in the overall functionality and user experience of the Hasanah System. These design aspects ensure that the Hasanah System is robust, efficient, and user-centric, laying a strong foundation for implementation and future enhancements.

Chapter 5

5.1 Introduction

This chapter provides an overview of the project's main components. It starts by describing the software tools used, followed by the hardware setup and significant parts of the code. Key aspects of the implementation are highlighted to give a clear understanding of how the system was built and operates.

5.2 Software Description

The development of this project relied on several essential software tools, each playing a critical role in different stages of the process:

- **Android Studio**: integrated development environment (IDE), was used to design and implement the user interface (UI). It provided a platform to create, test, and optimize the app, ensuring a user-friendly and seamless experience.
- **Firebase**: a Backend-as-a-Service (BaaS) platform built on Google's infrastructure, provides tools for app development, user authentication, and real-time data synchronization. In this project, Firebase is used for:
  - Storing user data in real-time.
  - Authenticating users through email.
  - Synchronizing data across devices to ensure availability, even offline.
- **Roboflow**: Roboflow was employed for dataset collection and preprocessing. It enabled efficient data management, augmentations, and the creation of a well-structured dataset for training the machine learning model.
- **Google Colab**: The machine learning model was developed and trained using Google Colab. This platform facilitated efficient model training without the need for extensive local hardware resources.
- **Flask**: Flask was utilized as a lightweight and flexible web framework to create the API for the project. It enabled seamless communication between the detection model and the mobile application by handling HTTP requests and responses efficiently.
- **Ngrok**: Ngrok was used to establish a secure tunnel for the API, making the locally hosted service accessible over the internet. internet.
- **Yolov8n:** Used for object detection tasks, providing accurate and efficient analysis of user-uploaded images for prayer posture detection

5.3 Hardware Description

Our system ”Hasanah” integrates hardware components to support its functionality of analyzing and correcting prayer postures.

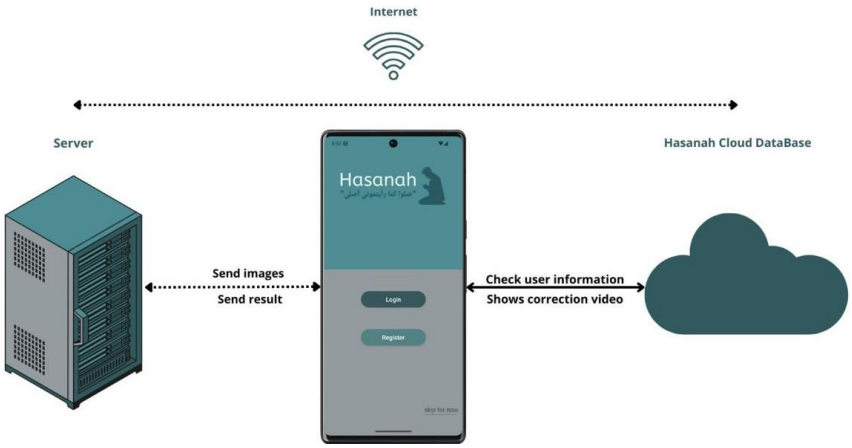


Figure 5.1: Hardware Design

- **The user’s phone** is used to capture images of prayer postures via the camera and upload the captured prayer movements.
- **The mobile application** serves as the primary interface. The user selects a pose and uploads an image through the application. The application then sends the image to the server via an API built with Flask. The server hosts the model, which analyzes the prayer posture, detects any errors, and returns the result to the application to be displayed to the user.
- Once the result is displayed, it is saved in the Hasanah Cloud Database. When the user logs into their account, the application retrieves their stored data from the **Hasanah Cloud Database**, allowing the user to access their recorded history at any time for future reference or review.
- To ensure smooth functionality, a reliable Wi-Fi connection is essential. Wi-Fi facilitates quick and efficient uploading of the user’s prayer posture, preventing delays or interruptions, and ensures that backup data is securely transmitted.

5.4 Details

5.4.1 Frontend Implementation

:The user interface for our project was developed using Java to ensure compatibility with Android systems. Android Studio was used for development, offering the necessary tools for efficient and seamless app creation.

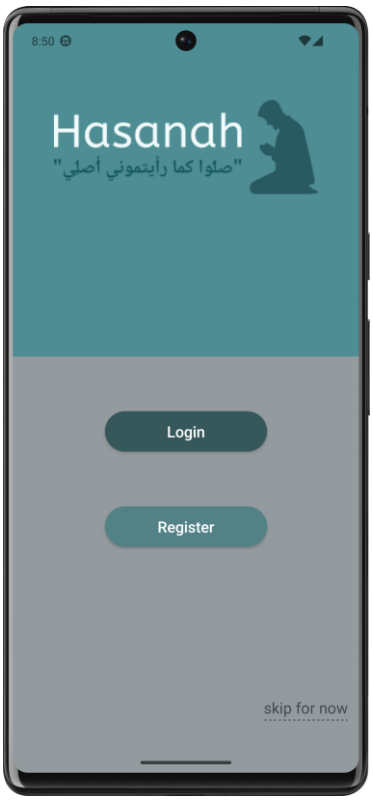


Figure 5.2: Splash Screen

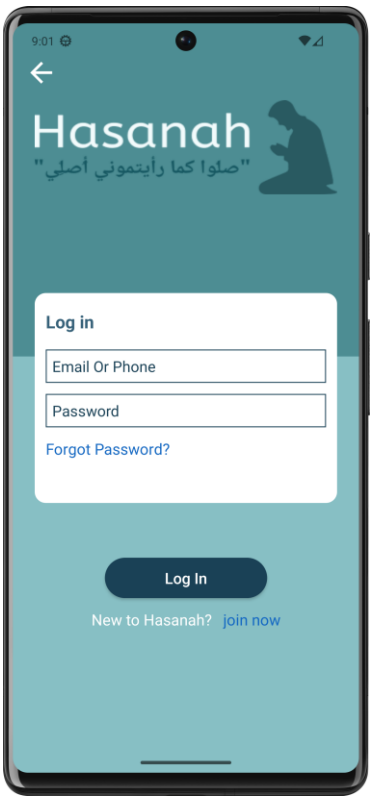
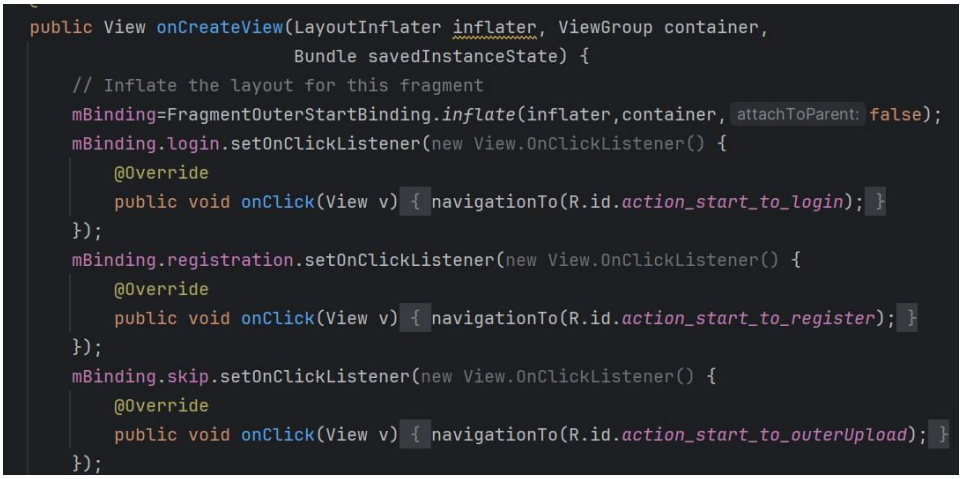


Figure 5.3: Login Screen

Figure 5.4: Register Screen

- **Splash Screen:** Serves as the entry point to the Hasanah application, providing users with three main options:
  - **Login Button:** Registered users can access their accounts by entering their email and password. Once the user submits valid credentials and presses the ”Login” button, a request is sent to Firebase for authentication. Upon successful login, the app retrieves the user’s stored data (e.g., prayer posture analysis history) from the Hasanah Cloud Database, ensuring a personalized experience.
  - **Register Button:** New users can create an account by selecting this option. After entering the required information (e.g., email, password, Date of Birth, and Name), the app sends the data to Firebase. Firebase handles user authentication and securely stores the account information.
  - **Skip for Now Option:** This option allows users to explore the app without logging in or registering. However, certain features, such as saving results or accessing historical data, will not be available. Users can later return to the Splash screen to log in or register.



```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    mBinding=FragmentOuterStartBinding.inflate(inflater,container, attachToParent: false);
    mBinding.login.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { navigationTo(R.id.action_start_to_login); }
    });
    mBinding.registration.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { navigationTo(R.id.action_start_to_register); }
    });
    mBinding.skip.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { navigationTo(R.id.action_start_to_outerUpload); }
    });
```

Figure 5.5: Splash Code

Description: The landing code that guides the user to either log in or register or skip. And **onClickListeners** enable seamless navigation based on user button interactions.

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    mBinding = FragmentLoginBinding.inflate(inflater, container, attachToParent: false);

    mBinding.arrowBack.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { navigationTo(R.id.action_login_to_start); }
    });
    mBinding.register.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { navigationTo(R.id.action_login_to_register); }
    });
    mBinding.forgotPassword.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { navigationTo(R.id.action_login_to_forgotPassword); }
    });
    login();

    return mBinding.getRoot();
}
```

Figure 5.6: Login Code

Description: The code for logging into an existing account. **onClickListeners** enable seamless navigation based on user button interactions.

```
public void signIn(String email, String password) {  1 usage
    auth = FirebaseAuth.getInstance();
    auth.signInWithEmailAndPassword(email, password)
            .addOnSuccessListener(authResult -> {
                Common.password =password;
                loading.dismiss();
                funLoginSuccessfully();

            })
            .addOnFailureListener(e -> {
                loading.dismiss();
                funLoginField("Sorry, we couldn't find your account. Please check your pass
            });

}

private void funLoginSuccessfully() {  1 usage
    SweetAlertDialog success = SweetDialog.success(getContext(),  title: "You have successfull
    success.show();
    startActivity(new Intent(getActivity(), UserMainActivity.class));
    getActivity().finish();
}
```

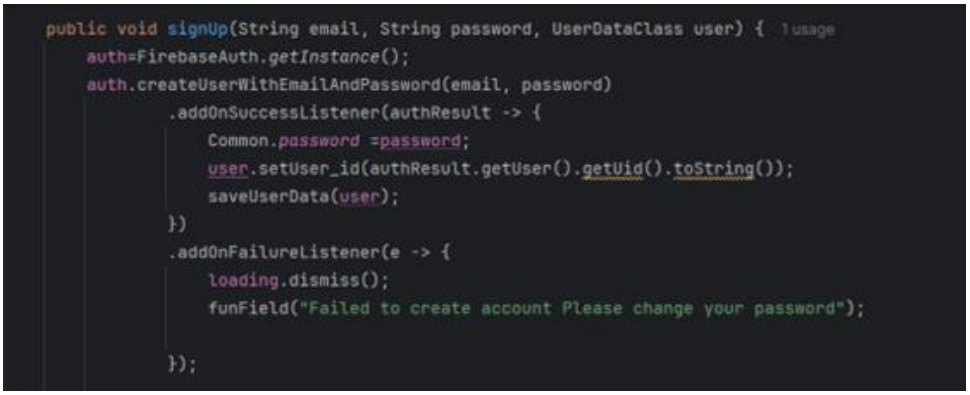
Figure 5.7: Login Code

Description: The **signIn()** function checks whether the provided email and password match an existing user in the Firebase authentication database. Upon successful login, the **funLoginSuccessfully()** function is triggered. This function displays a success message and navigates the user to the home interface, where they can view their user history. In the case where the login process was unsuccessful, a message will appear stating the error.

```
public void checkEnteredData() {  1 usage
    String name = mBinding.name.getText().toString();
    String email = mBinding.email.getText().toString();
    String password = mBinding.password.getText().toString();
    String date = mBinding.date.getText().toString();
    if (TextUtils.isEmpty(name)) {
        mBinding.name.setError("Please enter your name");
    } else if (email.isEmpty()) {
        mBinding.email.setError("Please enter your password");
    } else if (!EmailValidator.isValidEmail(email)) {
        mBinding.email.setError("Enter the password correctly 'user@gmail.com'");
    } else if (password.isEmpty()) {
        mBinding.password.setError("Please enter your password");
    } else if (!PasswordValidation.validatePassword(password)) {
        mBinding.password.setError("Please enter a strong password that contains uppercase letters, " +
                "lowercase letters, numbers and symbols (at least 8 characters)");
    } else if (date.isEmpty()) {
        mBinding.date.setError("Please enter your DOB");
    } else {
        funLoading();
        UserDataClass user = new UserDataClass(name, email, date,  photo: "",  user_id: "");
        signUp(email, password, user);
    }
}
```

Figure 5.8: Register Code

Description: The code to create a new account. The **checkEnteredData()** function checks the validity of the input data. It verifies if the user has filled all the boxes and checks the format.



```
public void signUp(String email, String password, UserDataClass user) {  1 usage
    auth=FirebaseAuth.getInstance();
    auth.createUserWithEmailAndPassword(email, password)
            .addOnSuccessListener(authResult -> {
                Common.password =password;
                user.setUser_id(authResult.getUser().getUid().toString());
                saveUserData(user);
            })
            .addOnFailureListener(e -> {
                loading.dismiss();
                funField("Failed to create account Please change your password");

            });
```

Figure 5.9: Register Code

Description: In the **signup** function, user data is validated to ensure inputs like email and password meet the required criteria (e.g., valid email format, strong password). Firebase Authentication's **createUserWithEmailAndPassword(email, password)** method is then called to create a new user. On success, Firebase generates a unique user ID (uid) and securely stores the credentials. This ID can be used to associate the user with additional details in the Firebase database, such as their profile or preferences. If the process fails (e.g., email already exists or weak password), an error message is shown to the user. Successful signups may redirect users to a dashboard, while errors prompt corrections.

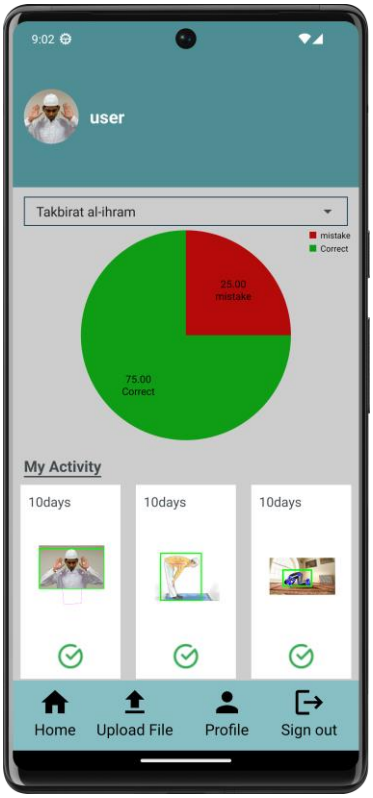


Figure 5.10: Home Screen

Figure 5.11: Profile Screen

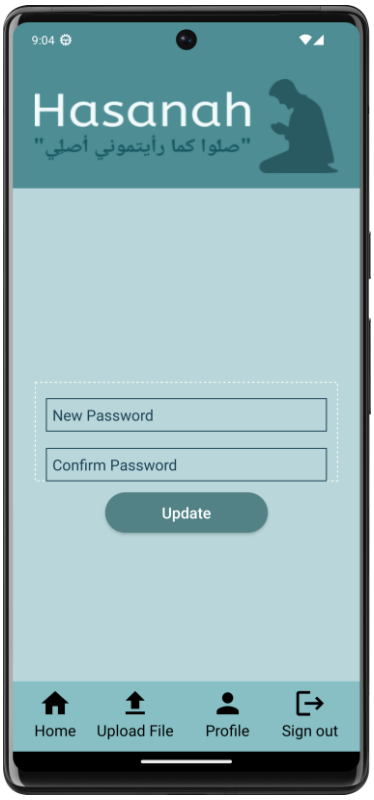


Figure 5.12: Updated Password Screen

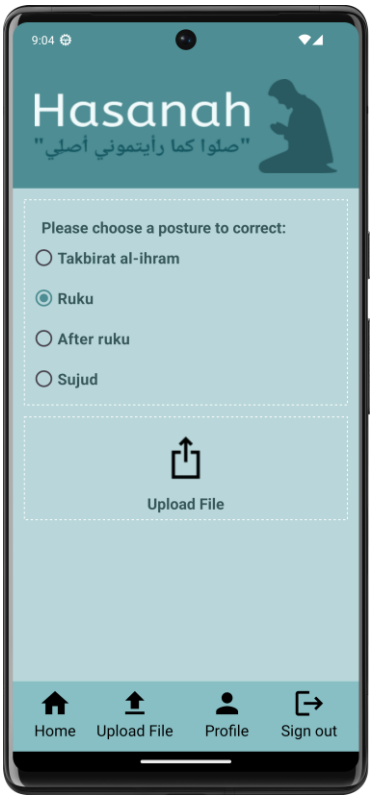


Figure 5.13: Upload File Screen

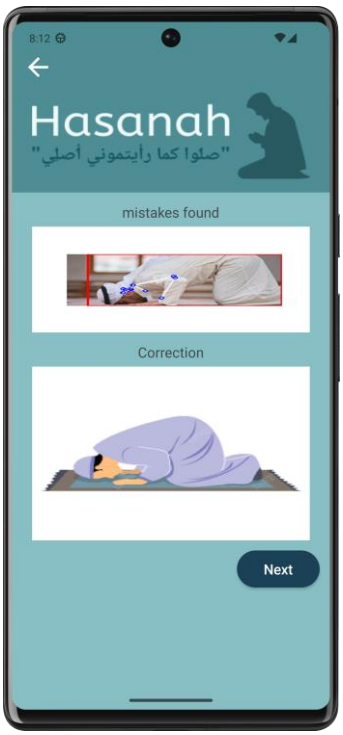


Figure 5.14: Result Screen

Figure 5.15: Video Screen

```
private void defineRecyclerview() {  1 usage
    statuses = new ArrayList<>();
    statuses.add(new Status( correct: 0,  mistake: 0));
    statuses.add(new Status( correct: 0,  mistake: 0));
    statuses.add(new Status( correct: 0,  mistake: 0));
    statuses.add(new Status( correct: 0,  mistake: 0));

    activities = new ArrayList<>();
    adapter = new ActivitiesAdapter(activities, getActivity());
    mBinding.activities.setLayoutManager(new GridLayoutManager(getActivity(),  spanCount: 3));
    mBinding.activities.setAdapter(adapter);
}
```

Figure 5.16: Home Code

Setting up the **RecyclerView** and **ActivitiesAdapter** to display activities is important for showing the history of the user's interactions.

```
private void getUserActivities() {  1 usage
    database.child(userId).addValueEventListener(new ValueEventListener() {
        @Override  2 usages
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if (snapshot.exists()) {
                for (DataSnapshot data : snapshot.getChildren()) {
                    ActivityDataClass activity = data.getValue(ActivityDataClass.class);
                    if (activity.getStatus() == 0) {
                        int i = statuses.get(activity.getType()).getCorrect() + 1;
                        statuses.get(activity.getType()).setCorrect(i);
                    }
                    if (activity.getStatus() == 1) {
                        int i = statuses.get(activity.getType()).getMistake() + 1;
                        statuses.get(activity.getType()).setMistake(i);
                    }
                    activities.add(activity);
                }
            } else {
                mBinding.nodata.setVisibility(View.VISIBLE);
            }
            loading.dismiss();
            adapter.notifyDataSetChanged();
            viewBarCharStatics( i: 0);
```

Figure 5.17: Home Code

The **getUserActivities()** method that listens for and retrieves the user's activity data from Firebase Realtime Database is key to fetching and displaying user history.

```
private void getUserData() {  1 usage
    database.child(userId).addValueEventListener(new ValueEventListener() {
        @Override  2 usages
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            loading.dismiss();
            if (snapshot.exists()) {
                String userName = snapshot.child( path: "userName").getValue().toString();
                String email = snapshot.child( path: "email").getValue().toString();
                String dob = snapshot.child( path: "dob").getValue().toString();
                String user_id = snapshot.child( path: "user_id").getValue().toString();
                image = snapshot.child( path: "photo").getValue().toString();
                mBinding.userName.setText(userName);
                mBinding.date.setText(dob);
                mBinding.email.setText(email);
                if (!image.isEmpty()) {
                    Glide.with(getActivity()).load(image).into(mBinding.imageProfile);
                }
                user = new UserDataClass(userName, email, dob, image, user_id);
            }
        }
```

Figure 5.18: Profile Code

to check and update their profile.And Fetching User Data **getUserData()** Retrieves user data from Firebase, including name, email, date of birth, and profile picture.

```
private void clickUpdate() {  1 usage
    mBinding.update.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String userName = mBinding.userName.getText().toString();
            String dob = mBinding.date.getText().toString();

            if (userName.isEmpty()) {
                mBinding.userName.setError("Please enter your name");
            } else if (dob.isEmpty()) {
                mBinding.date.setError("Please enter your DOB");
            } else {
                alertDialog();
            }
        }
    });
}
```

Figure 5.19: Profile Code

Updating User Profile **clickUpdate()** and **saveData()**

Figure 5.20: Update Password Code

Description: The Code for resetting the password.And **checkData()**: This method validates the new password and checks if the passwords match. If validation passes, it proceeds to the **changePassword()** method.

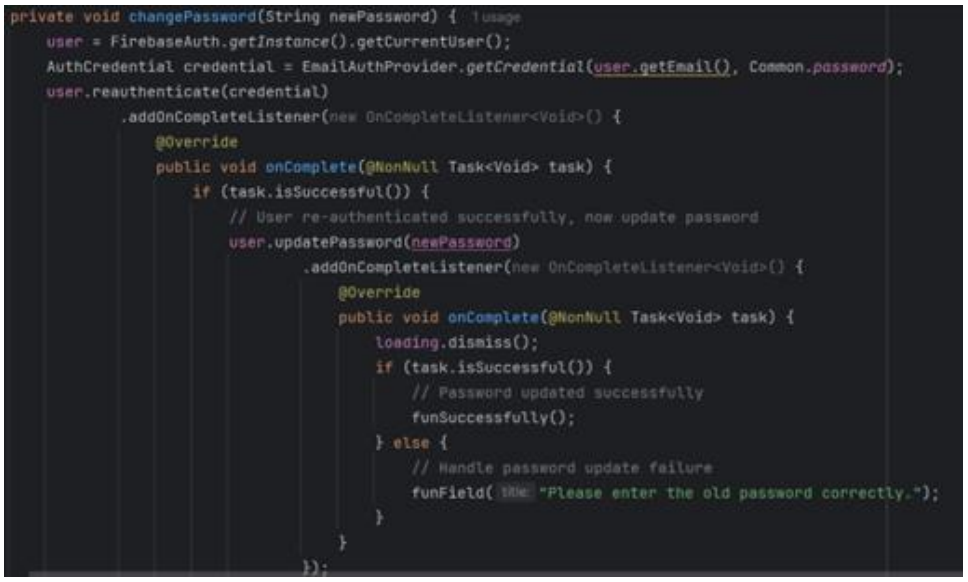


Figure 5.21: Update Password Code

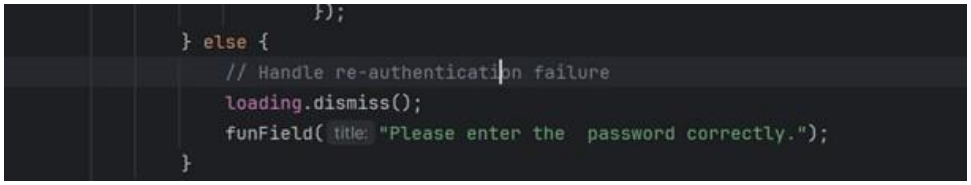


Figure 5.22: Update Password Screen

Description: **changePassword()**: This method takes care of reauthenticating the user and updating the password if reauthentication is successful.

```
public void getAiResponseTakbeer(MultipartBody.Part body) { 1 usage
    apiInterface.UploadFileTakbeer(body).enqueue(new Callback<RespnseName>() {
        public void onResponse(Call<RespnseName> call, Response<RespnseName> response) {
            loading.dismiss();
            if (response.isSuccessful()) {
                Log.e( tag: "succ", response.body().toString());
                Common.respnseName = response.body();
                funLoginSuccessfully();
                saveToActivities(response.body());
            } else {
                Log.e( tag: "failed",  msg: "failed " + response.message() + " " + response.code());
                funField( title: "There was an error connecting to the server.");

            }
        }

        @Override
        public void onFailure(Call<RespnseName> call, Throwable t) {
            loading.dismiss();
            Log.e( tag: "error", t.getMessage());

            Toast.makeText(getContext(),  text: "error : " + t.getMessage(), Toast.LENGTH_SHORT).show();
        }
```

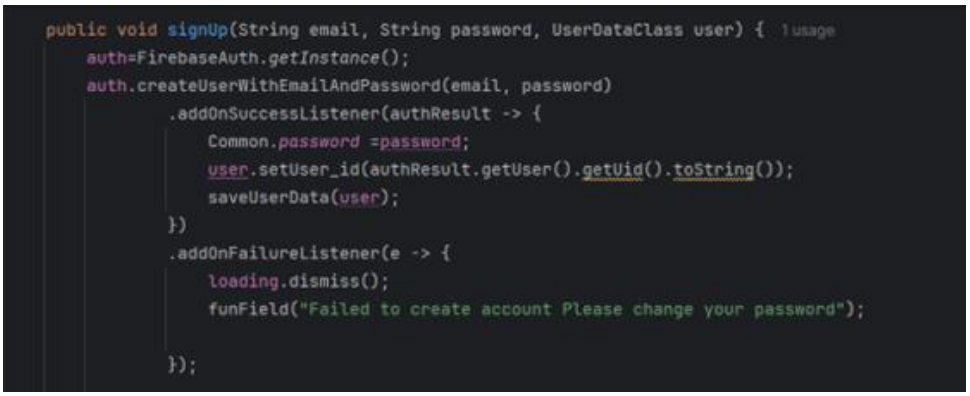
Figure 5.23: Uploading File Code

Description:Code to let the user check the posture that he/she wants to detect and upload image.The various **getAiResponse*()** methods upload the image to the server, based on the selected position. Each **getAiResponse*()** method is designed to handle the upload for a specific type of image (corresponding to a certain position type like ”Takbeer,” ”Ruku,” ”Sujud,” etc.)

```
private void viewData()  1 usage
{
    byte[] decodedByte = Base64.decode(Common.respnseName.getImage_base64(), Base64.DEFAULT);
    bitmap = BitmapFactory.decodeByteArray(decodedByte,  offset: 0, decodedByte.length);
    mBinding.image.setImageBitmap(bitmap);
}
```

Figure 5.24: Result Code

shows the analyzed image after decoding the base64 string.

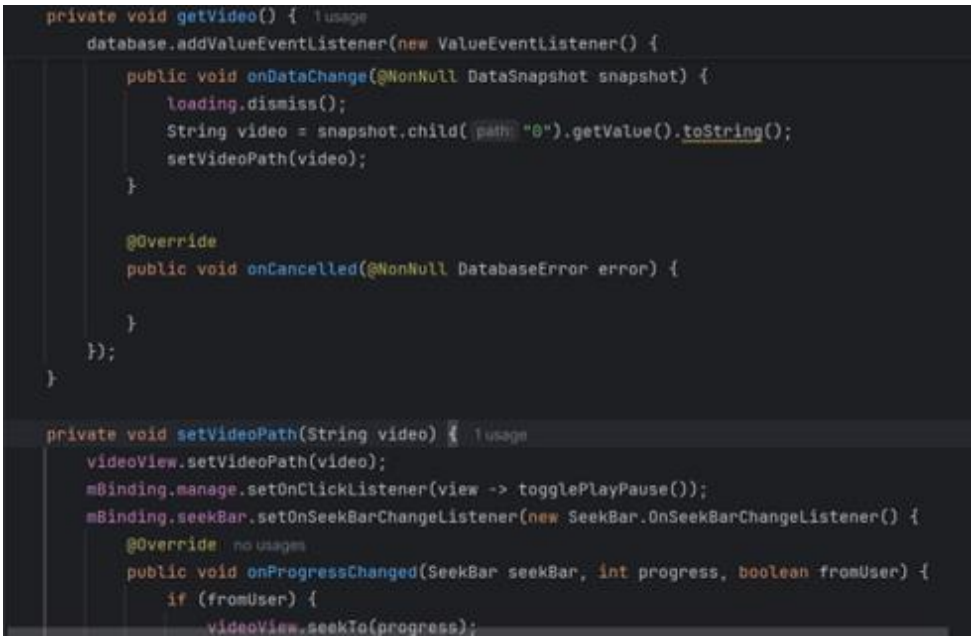


```
public void signUp(String email, String password, UserDataClass user) { 1 usage
    auth=FirebaseAuth.getInstance();
    auth.createUserWithEmailAndPassword(email, password)
            .addOnSuccessListener(authResult -> {
                Common.password =password;
                user.setUser_id(authResult.getUser().getUid().toString());
                saveUserData(user);
            })
            .addOnFailureListener(e -> {
                loading.dismiss();
                funField("Failed to create account Please change your password");

            });
```

Figure 5.25: Result Code

This part checks whether the user’s posture has any mistakes (based on the AI response) and adjusts the UI accordingly.



```
private void getVideo() {  1 usage
    database.addValueEventListener(new ValueEventListener() {
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            loading.dismiss();
            String video = snapshot.child( path: "0").getValue().toString();
            setVideoPath(video);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}

private void setVideoPath(String video) {  1 usage
    videoView.setVideoPath(video);
    mBinding.manage.setOnClickListener(view -> togglePlayPause());
    mBinding.seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override  no usages
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
            if (fromUser) {
                videoView.seekTo(progress);
```

Figure 5.26: Video Code

The code retrieves a video URL from Firebase Realtime Database using **getVideo()** and sets it in the **VideoView** with the **setVideoPath()** method

5.4.2 Backend Implementation

Model

The detection model for this project was developed using advanced deep learning techniques to accurately analyze and correct prayer postures. Here's a detailed breakdown of its components and methodologies:

- **Convolutional Neural Networks (CNNs)**: We employed CNNs as the core architecture for image analysis and feature extraction. CNNs are particularly effective for tasks involving spatial hierarchies in images, making them ideal for identifying and understanding the intricate details of prayer postures. The layers of the CNN were trained to recognize key body parts and movements relevant to prayer positions, ensuring high accuracy in detecting correct and incorrect forms.
- **YOLO (You Only Look Once)**: For object detection, we integrated the YOLOv8 algorithm, a state-of-the-art real-time object detection framework. YOLOv8 is highly efficient in detecting objects in images with minimal latency, making it well-suited for analyzing images. It was used to:
  - Identify body parts such as the head, hands, knees, and feet.
- **Computer Vision (CV) Techniques**: In addition to CNNs and YOLO, several computer vision techniques were incorporated to enhance the model's performance:
  - **Pose Estimation**: Used to detect key landmarks on the human body (e.g., joints and limbs) and ensure proper alignment during prayer postures.
- **Dataset** The model was trained using a dataset consisting of 3,225 images, divided as follows:
  - **Training Set**: 2,726 images used to train the model and enable it to learn the required patterns.
  - **Validation Set**: 349 images used to evaluate the model's performance during training and fine-tune its parameters.
  - **Testing Set**: 150 images used to assess the model's accuracy and robustness after training.

  This distribution ensured that the model could generalize effectively while maintaining high accuracy and reliability for real-world usage.

Setting Up YOLO and OpenCV:

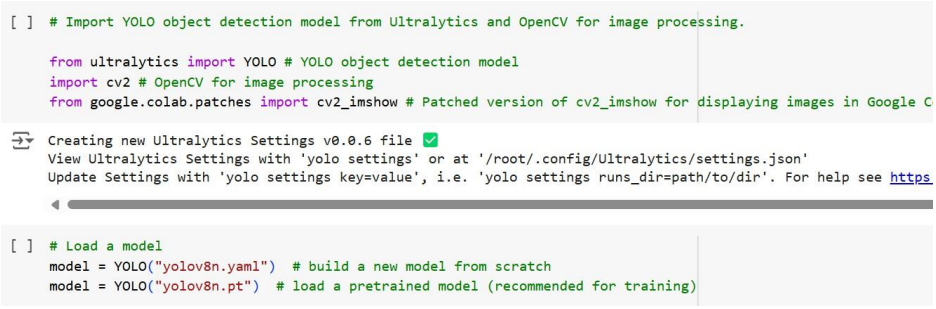


Figure 5.27: Setting up

YOLO is used for detecting objects (e.g., body parts or prayer positions) in uploaded files. OpenCV allows processing uploaded file frames efficiently.

Stages of Training the YOLOv8 Model:

- **First Stage: Initial Setup (Testing Phase):**
  - **Subset Used:** In the initial phase, we used a small subset of the dataset, consisting of 246 images for training and 23 images for validation.
  - **Objective:** The primary goal was to evaluate whether YOLOv8 was suitable for our application and capable of accurately detecting the key postures (Ruku, Sujud, Takbeer, and After Ruku (Qiam)).
  - **Reason:** This preliminary test was conducted to save time and resources. In case YOLOv8 did not perform adequately, it would allow us to switch to an alternative algorithm without investing significant effort into full training.
  - **Result:** YOLOv8 performed well during this testing phase, providing satisfactory results for posture detection. Based on this success, we decided to proceed with YOLOv8 for further training.
- **Second Stage: Intermediate Training Phase:**
  - **Subset Used:** In the second stage, we expanded the dataset to 1,400 images for training, 349 images for validation, and 150 images for testing.
  - **Objective:** The aim was to enhance the model's accuracy by training on a larger portion of the dataset and generating results for evaluation.
  - **Application:** At this stage, we assumed it was the final phase of training and based our poster presentation on the results from this setup.
  - **Result:** The results were satisfactory, but upon further analysis, we concluded that there was still potential for improvement in the model's performance.
- **Third Stage: Final Training with the Full Dataset:**
  - **Subset Used:** After reviewing the results from the intermediate phase, we decided to train the model using the full dataset, which included 2,726 images for training, 349 for validation, and 150 for testing.
  - **Objective:** The goal was to maximize the model's accuracy by utilizing the entire dataset and providing it with the most comprehensive training possible.
  - **Result:** The final training phase yielded significantly improved results, confirming that using the full dataset led to a noticeable enhancement in the model's accuracy and performance.

Training YOLOv8 Model:

- Epochs: 40 epochs for training.
- Initial Learning Rate: 0.1
- Final Learning Rate: 0.0001
- Cosine Annealing: True (learning rate decay is used to optimize training).
- Batch Size: 64 images per batch.
- Layers: The model uses 249 layers.
- Training Dataset: 2709 images in the training set.

Each epoch processes 249 layers, with each layer passing through 64 images.

Figure 5.28: Training YOLOv8

Train the YOLOv8 model with the specified data and hyperparameters.

```
                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22        [15, 18, 21]  1    431452  ultralytics.nn.modules.head.Detect           [4, [64, 128, 256]]
Model summary: 249 layers, 2,690,988 parameters, 2,690,972 gradients, 6.9 GFLOPs
```

Figure 5.29: First Training Output

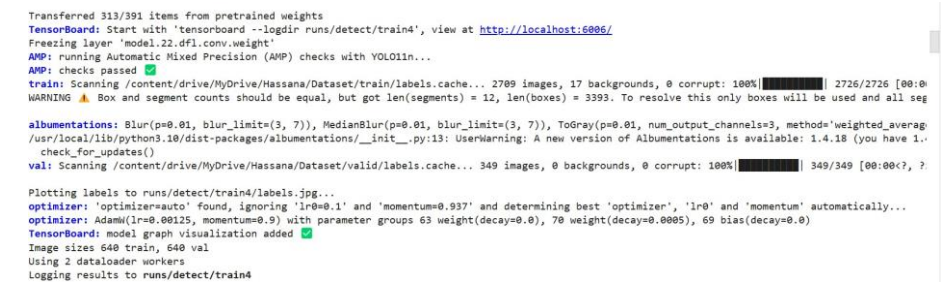


Figure 5.30: Second Training Output

YOLOv8 Model Validation:

- Validation Dataset: 349 images.
- Validation Layers: 186 layers process the validation dataset.

```
[ ]  model.val(data="/content/drive/MyDrive/Hassana/Dataset/data.yaml")
```

Figure 5.31: Validation YOLOv8

```
Ultralytics 8.3.28 🚀 Python-3.10.12 torch-2.5.0+cu121 CPU (Intel Xeon 2.20GHz)
Model summary (fused): 186 layers, 2,685,148 parameters, 0 gradients, 6.8 GFLOPs
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100%|██████████| 755k/755k [00:00<00:00, 15.2MB/s]
val: Scanning /content/drive/MyDrive/Hassana/Dataset/valid/labels.cache... 349 images, 0 backgrounds, 0 corrupt: 100%|██████████| 349/349 [00:00<?,

                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 22/22 [01:55<00:00,  5.23s/it]
                   all        349        409      0.931      0.918      0.957      0.765
               raising         53         61      0.937      0.934      0.969       0.76
                  ruku         82         87      0.939      0.886      0.929       0.72
                 sujud        147        179      0.887      0.918      0.948      0.749
               takbeer         74         82      0.962      0.934      0.984      0.829
Speed: 5.6ms preprocess, 292.5ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to runs/detect/val
ultralytics.utils.metrics.DetMetrics object with attributes:

ap_class_index: array([0, 1, 2, 3])
box: ultralytics.utils.metrics.Metric object
confusion_matrix: <ultralytics.utils.metrics.ConfusionMatrix object at 0x7dfb20891090>
curves: ['Precision-Recall(B)', 'F1-Confidence(B)', 'Precision-Confidence(B)', 'Recall-Confidence(B)']
curves_results: [[array([          0,    0.001001,    0.002002,    0.003003,    0.004004,    0.005005,    0.006006,    0.007007,    0.008008,
0.009009,     0.01001,    0.011011,    0.012012,    0.013013,    0.014014,    0.015015,    0.016016,    0.017017,    0.018018,    0.019019,
0.02002,    0.021021,    0.022022,    0.023023,
          0.024024,    0.025025,    0.026026,    0.027027,    0.028028,    0.029029,     0.03003,    0.031031,    0.032032,    0.033033,
0.034034,    0.035035,    0.036036,    0.037037,    0.038038,    0.039039,     0.04004,    0.041041,    0.042042,    0.043043,    0.044044,
0.045045,    0.046046,    0.047047,
```

Figure 5.32: Validation Output

Model Performance:

- Final Accuracy: 93% (0.93)
- Recall: 91% (0.91)
- mAP50: Mean Average Precision at 50%, showing how well the model detects objects at a 50% confidence threshold (95% mAP50).
- P50-95: mAP calculated over the range of 50% to 95% confidence, showing more stringent object detection (76% mAP50-95).

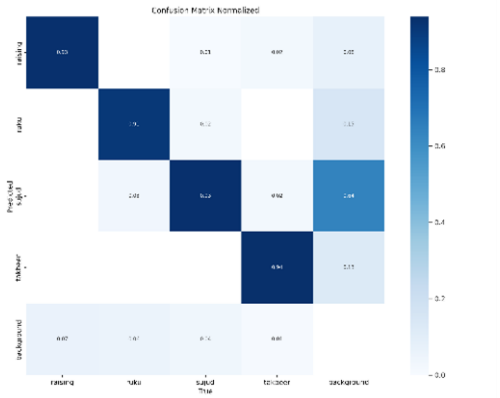


Figure 5.33: Normalized Confusion Matrix

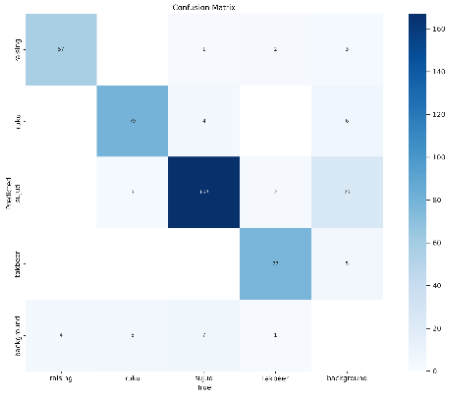


Figure 5.34: Confusion Matrix

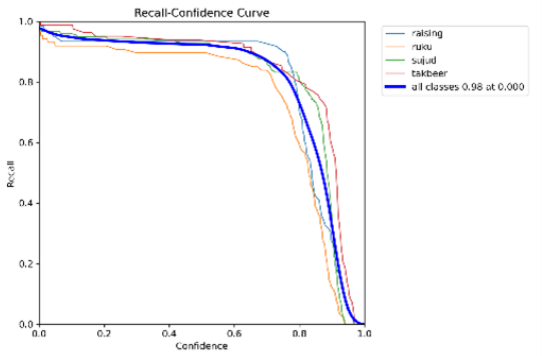


Figure 5.35: Recall

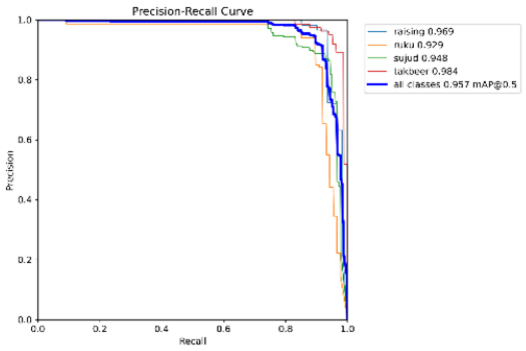


Figure 5.36: Precision and Recall

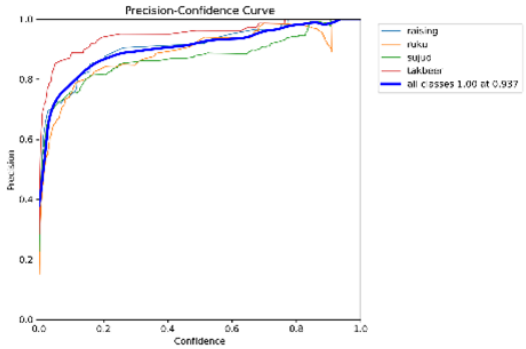


Figure 5.37: Precision

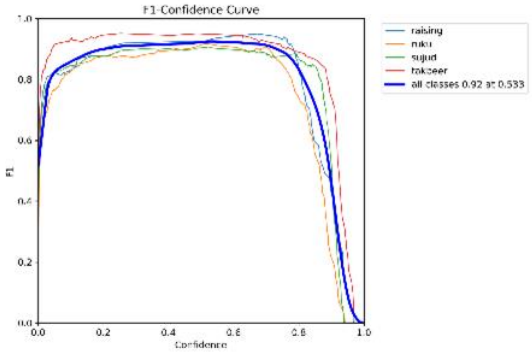


Figure 5.38: F1 Score

Testing YOLO V8 Model:

- **Ruku:** To detect the Ruku (bowing) position in Salat, the code calculates the angle at the hip by analyzing the relationship between three key points: the knee, hip, and shoulder. The process involves measuring the angles formed by the lines connecting the knee to the hip and the hip to the shoulder on both the left and right sides. These measured angles are compared to the ideal angle of 90 degrees, representing the correct bowing posture. The absolute difference between the measured angle and 90 degrees is calculated for each side. If the difference is within a threshold of 30 degrees on either side, the pose is considered correct. This method ensures accurate detection of the bowing position and evaluates how closely the posture aligns with the ideal form.

To detect the Ruku position accurately, the photo should be taken from the right angle:

  - **Side Angle (Preferred):** The best angle is from the side, showing the knees below the hips and shoulders.
  - **Slight Tilt (Acceptable):** A slight tilt from the front is also fine, as long as the posture is clear.
  - **Front Angle (Avoid):** Avoid taking the photo from the front as it doesn't show the proper alignment.

Validation Logic:

  - **Angle at the Hip: Calculate the angle formed by:**
    * Line from **Shoulder to Hip.**
    * Line from Hip to Knee. The angle should be close to 90°

$$\text{Deviation} = |\text{calculated angle} - 90^\circ| < 30^\circ$$

Figure 5.39: Ruku Equation

  - **Shoulders Alignment:**Ensure the y-coordinates of shoulders differ by less than a threshold.

```
# Initialize Mediapipe Pose model
mp_pose = mp.solutions.pose
pose = mp_pose.Pose()
```

Figure 5.40: Pose Model

Mediapipe is used to detect body landmarks, while YOLO detects individuals in the input frame.

```
    # Function to calculate angle between lines formed by three points
    def calculate_angle(a, b, c):
        angle_rad1 = math.atan2(c[1]-b[1], c[0]-b[0]) - math.atan2(a[1]-b[1], a[0]-b[0])
        angle_rad1 = angle_rad1 % (2 * math.pi)
        angle_rad2 = math.atan2(a[1]-b[1], a[0]-b[0]) - math.atan2(c[1]-b[1], c[0]-b[0])
        angle_rad2 = angle_rad2 % (2 * math.pi)
        return math.degrees(min(angle_rad1, angle_rad2))
```

Figure 5.41: Calculate angle

This function calculates the angles between three points, which is key for detecting the body position.

```
# Function to check if the pose resembles the ruku position in Salat
def is_ruku_pose(landmarks):
    # Get relevant landmarks (e.g., shoulders, hips)
    left_hip = (landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x, landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].y)
    right_hip = (landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].x, landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].y)
    left_shoulder = (landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x, landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y)
    right_shoulder = (landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x, landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].y)
    left_knee = (landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x, landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].y)
    right_knee = (landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].x, landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].y)

    # Calculate angles between lines formed by hip to shoulder and hip to knee
    angle_left = abs(90 - calculate_angle(left_knee, left_hip, left_shoulder))
    angle_right = abs(90 - calculate_angle(right_knee, right_hip, right_shoulder))
    # Define condition for ruku pose (e.g., angles within a range around 90 degrees)
    ruku_angle_threshold = 30
    if angle_left <= ruku_angle_threshold or angle_right <= ruku_angle_threshold:
        return True, min(abs(angle_left), abs(angle_right)) / 100 * 100  # Calculate error percentage
    else:
        return False, 0
```

Figure 5.42: Detect body marks

This function checks the detected body landmarks to determine whether the Ruku' position is present.

```
def process_image(image_path):
    # Load Model
    results = model.predict(image_path)

    for r in results:
        print(r.boxes)
        # Extract bounding box coordinates
        bounding_box = x1, y1, x2, y2 = r.boxes.xyxy[0].tolist()
        print("Bounding box coordinates (x1, y1, x2, y2):", x1, y1, x2, y2)
        bounding_box_tuple = tuple(bounding_box)

    bounding_box = tuple(int(value) for value in bounding_box_tuple)

    # Load image
    image = cv2.imread(image_path)

    # Draw bounding box
    x1, y1, x2, y2 = bounding_box
    cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 6)

    # Convert image to RGB
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

Figure 5.43: Draw a bounding box

This shows how YOLO is used to detect a person and draw a bounding box.

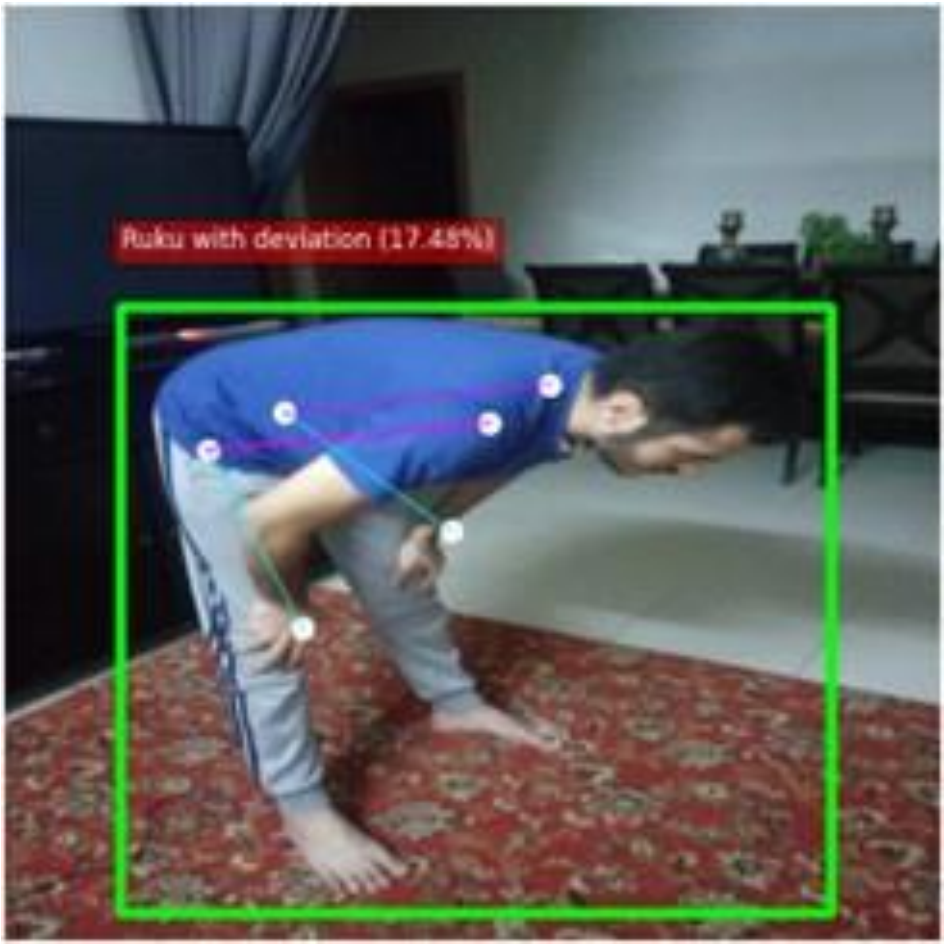


Figure 5.44: Result

Ruku pose detected with an error of 17.48%.

- **Takbeer:** To detects the Takbeer position in Salat, several key body landmarks are used to check if the person's posture matches the correct Takbeer position. The first condition is that the shoulders should be roughly level with each other, meaning the difference in the y values of the left and right shoulders must be small (less than 0.05). If the left and right shoulders are not aligned horizontally, it can affect the rest of the checks. For example, if the left shoulder is higher than the right, it will result in an uneven distance between the shoulder and the hip on that side, which will violate the condition of parallel shoulders. Additionally, the distance between the shoulder and the hip is checked to ensure that the shoulder is positioned above the hip (more than 0.3 units apart). The wrists and index fingers are also checked to ensure they are above the shoulders. If all these conditions are met, the posture is considered to be in the correct Takbeer position. However, if the shoulder alignment condition is not satisfied, the entire posture will be considered incorrect, regardless of the position of the hands or fingers. This ensures the Takbeer position is accurately detected, with special attention to the alignment of the shoulders.

To detect the "Takbeer" position in Salat, the photo should be taken from the correct angle:

- **Front Angle (Preferred):** The best angle is directly from the front to clearly show the alignment of the shoulders, elbows, and body.
- **Slight Tilt (Acceptable):** A slight tilt from the side is okay if the posture is still visible.
- **Steep Side Angle (Avoid):** Avoid a steep side angle as it makes it hard to see the proper alignment of the shoulders and body.

```python
def is_takbeer_pose(landmarks):
    left_shoulder = landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value]
    right_shoulder = landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value]
    left_hip = landmarks[mp_pose.PoseLandmark.LEFT_HIP.value]
    right_hip = landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value]
    left_wrist = landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]
    right_wrist = landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value]
    left_index = landmarks[mp_pose.PoseLandmark.LEFT_INDEX.value]
    right_index = landmarks[mp_pose.PoseLandmark.RIGHT_INDEX.value]

    # Check if left and right shoulders are on the same line
    if abs(left_shoulder.y - right_shoulder.y) < 0.05:
        # Check if distance between right hip and right shoulder in y-direction is big enough
        if right_hip.y - right_shoulder.y > 0.3:
            # Check if wrists are above the shoulders
            if left_wrist.y < left_shoulder.y and right_wrist.y < right_shoulder.y:
                # Check if index fingers are above the shoulders
                if left_index.y < left_shoulder.y and right_index.y < right_shoulder.y:
                    return True, abs(abs(left_shoulder.y)-abs(right_shoulder.y)) / 0.2 * 100
    return False, 0
```

Figure 5.45: Detect body marks

This function detects the Takbeer pose by comparing the alignment of body parts.

```python
def process_image(image_path):

    model_results = model.predict(image_path)

    for r in model_results:
        print(r.boxes)
        # Extract bounding box coordinates
        bounding_box = x1, y1, x2, y2 = r.boxes.xyxy[0].tolist()
        print("Bounding box coordinates (x1, y1, x2, y2):", x1, y1, x2, y2)
        bounding_box_tuple = tuple(bounding_box)

    bounding_box = tuple(int(value) for value in bounding_box_tuple)

    # Load image
    image = cv2.imread(image_path)

    # Convert image to RGB
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Detect pose landmarks
    results = pose.process(image_rgb)
    if results.pose_landmarks:
```

Figure 5.46: Draw a bounding box

This part loads the image, processes it, and displays the results

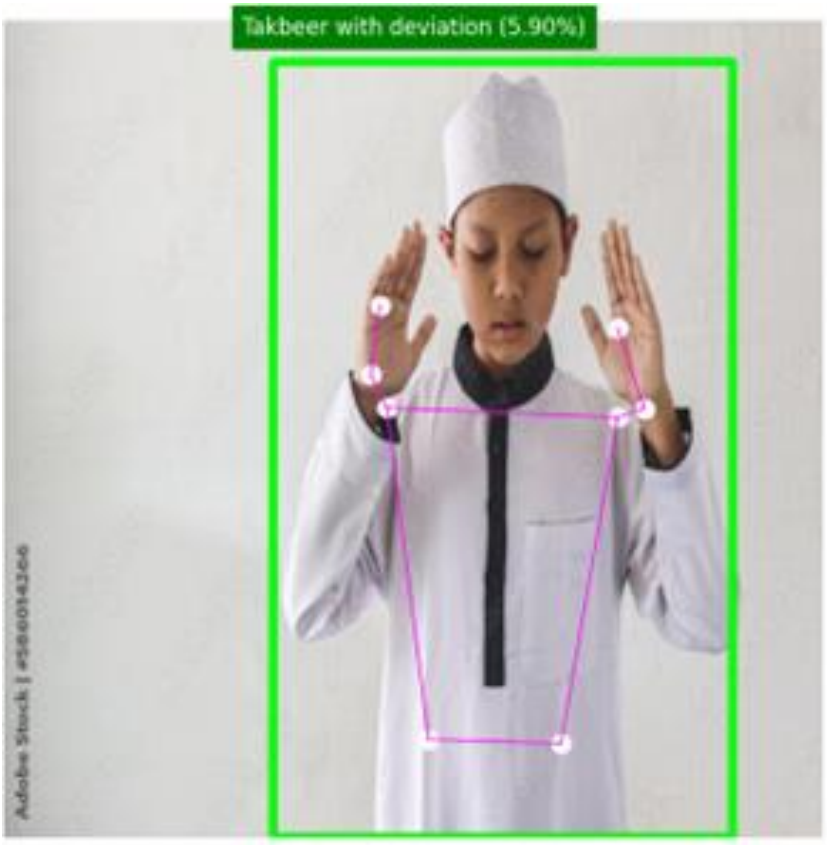


Figure 5.47: Result

Takbeer with deviation (5.90%)

- **Sujud** The code detects the Sujud (prostration) position in Salat by checking several key conditions in the person's posture using body landmarks.
  - **Knees Below Hips:** The first condition checks if both knees are positioned below the hips. This indicates that the person is kneeling, which is a fundamental part of the Sujud position.
  - **Wrists Close to the Nose:** The code then checks if both wrists are positioned close to the vertical line of the nose. Specifically, the difference in the y values between the wrists and the nose must be less than 0.1, ensuring that the wrists are near the face, which is typical in Sujud.
  - **Nose Below Hips:**The code verifies that the nose is positioned lower than the hips, indicating that the person's head is close to the ground, which is characteristic of the Sujud position.

  If all these conditions are met, the code considers the person to be in the correct Sujud position. If any condition fails, the posture is deemed incorrect. This method ensures an accurate detection of the Sujud position during Salat.

To detect the Sujud position accurately, the photo should be taken from the right angle:

- **Side Angle (Preferred):** The best angle is from the side, showing the knees below the hips and the wrists close to the nose.
- **Slight Tilt (Acceptable):** A slight tilt from the front is also fine, as long as the posture is clear.
- **Front Angle (Avoid):** Avoid taking the photo from the front as it doesn't show the proper alignment of the knees and wrists.

```
def is_sujud_pose(landmarks):
    left_knee = landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value]
    right_knee = landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value]
    left_ankle = landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value]
    right_ankle = landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value]
    left_wrist = landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]
    right_wrist = landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value]
    nose = landmarks[mp_pose.PoseLandmark.NOSE.value]
    left_hip = landmarks[mp_pose.PoseLandmark.LEFT_HIP.value]
    right_hip = landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value]
```

Figure 5.48: LandMarks

This function to put landmarks.

```
# 1. Knees should be below hips (indicating kneeling)
if left_knee.y > left_hip.y and right_knee.y > right_hip.y:
    print(1)
    print(abs(left_wrist.y - nose.y),abs(right_wrist.y - nose.y) )


    # 2. Check if wrists are close to the nose's vertical position (y-coordinate difference is close to zero)
    if abs(left_wrist.y - nose.y) < 0.1 and abs(right_wrist.y - nose.y) < 0.1:
        print(abs(left_wrist.y - nose.y),abs(right_wrist.y - nose.y) )
        print(2)

        # 3. Nose should be lower than hips (indicating the head is close to the ground)
        if nose.y > left_hip.y and nose.y > right_hip.y:
            print(3)
            return True, abs(left_wrist.y - nose.y)*2 * 100

return False, 0
```

Figure 5.49: Check body part

Function to Check for Sujud Pose.

```
def process_image(image_path):

    model_results = model.predict(image_path)

    for r in model_results:
        print(r.boxes)
        # Extract bounding box coordinates
        bounding_box = x1, y1, x2, y2 = r.boxes.xyxy[0].tolist()
        print("Bounding box coordinates (x1, y1, x2, y2):", x1, y1, x2, y2)
        bounding_box_tuple = tuple(bounding_box)

    bounding_box = tuple(int(value) for value in bounding_box_tuple)

    # Load image
    image = cv2.imread(image_path)

    # Convert image to RGB
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Detect pose landmarks
    results = pose.process(image_rgb)
    if results.pose_landmarks:
```

Figure 5.50: Process the Image

Function to Process the Image and Visualize Results.

Figure 5.51: Result

Sujud with deviation (0.60%).

- **After Ruku(Qiam)** The code detects the ”Raising” position in Salat by checking several key conditions in the person’s posture using body landmarks.

  – **Shoulders Level:**The first condition checks if both shoulders are on the same horizontal level, ensuring that the upper body is upright. The difference in the y values of the left and right shoulders should be minimal (less than 0.1).

  – **Elbows Close to the Body:** The code then checks if both elbows are close to the body, indicating that the arms are raised but not too far apart. This ensures that the posture resembles the typical ”raising” position.

  – **Hips Above Knees:** The code verifies that the hips are positioned above the knees, indicating that the person is in an upward posture after bowing.

  – **Head Alignment:** The code checks if the nose and chin are aligned horizontally, ensuring the head is straight and upright.

  – **Straight Body:** Finally, the code ensures that the body is straight by checking that the difference between the y values of the shoulders and hips is small (less than 0.3), indicating the body is upright.

  If all these conditions are met, the person is considered to be in the correct ”Raising” position. The code also calculates the error percentage based on the difference between the shoulder and hip positions.

To detect the ”Raising” position in Salat, the photo should be taken from the correct angle:

– **Front Angle (Preferred):** The best angle is directly from the front to clearly show the alignment of the shoulders, elbows, and body.

– **Slight Tilt (Acceptable):** A slight tilt from the side is okay if the posture is still visible.

– **Steep Side Angle (Avoid):** Avoid a steep side angle as it makes it hard to see the proper alignment of the shoulders and body.

Validation Logic:

– **Shoulders Alignment:**

$$\text{Difference in Shoulder Y-Coordinates} = |y_{\text{left\_shoulder}} - y_{\text{right\_shoulder}}| < \text{threshold}$$

Figure 5.52: Shoulder Equation

If the shoulders’ y-coordinates differ by less than 10% of the image height, they are considered aligned.

– **Hips Above Knees:** Compare the y-coordinates of the hips and knees:

$$y_{\text{left\_hip}} < y_{\text{left\_knee}} \quad \text{and} \quad y_{\text{right\_hip}} < y_{\text{right\_knee}}$$

Figure 5.53: Hips Above Knees

– **Head Alignment:** Ensure the x-coordinates of the nose and chin are close, confirming the head is upright:

$$|x_{\text{nose}} - x_{\text{chin}}| < \text{threshold}$$

Figure 5.54: Head Equation

– **Torso Straightness:** Check the difference between the vertical positions of shoulders and hips:

$$|y_{\text{left\_shoulder}} - y_{\text{left\_hip}}| \quad \text{and} \quad |y_{\text{right\_shoulder}} - y_{\text{right\_hip}}|$$

Figure 5.55: Torso Equation

```
def is_raising_pose(landmarks):
    left_shoulder = landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value]
    right_shoulder = landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value]
    left_elbow = landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value]
    right_elbow = landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value]
    left_knee = landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value]
    right_knee = landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value]
    left_ankle = landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value]
    right_ankle = landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value]
    left_hip = landmarks[mp_pose.PoseLandmark.LEFT_HIP.value]
    right_hip = landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value]
    nose = landmarks[mp_pose.PoseLandmark.NOSE.value]
    chin = landmarks[mp_pose.PoseLandmark.LEFT_EAR.value]  # Using left ear as an approximation for the chin
```

Figure 5.56: Landmarks

```
# 1. Shoulders should be level (indicating upright posture)
if abs(left_shoulder.y - right_shoulder.y) < 0.1:
    print("Shoulders level")

    # 2. Elbows should be close to the body (raised hands condition)
    if abs(left_elbow.x - left_shoulder.x) < 0.1 and abs(right_elbow.x - right_shoulder.x) < 0.1:
        print("Elbows close to body")

        # 3. Hips should be above the knees (indicating upward movement)
        if left_hip.y < left_knee.y and right_hip.y < right_knee.y:
            print("Hips above knees")

            # 4. Head should be aligned (nose and chin are aligned)
            if abs(nose.x - chin.x) < 0.05:
                print("Head aligned")

                # 5. Body should be straight (upper body is upright, not bent)
                if abs(left_hip.y - left_shoulder.y) < 0.3 and abs(right_hip.y - right_shoulder.y) < 0.3:
                    print("Body straightness achieved")
                    return True, abs(left_shoulder.y - left_hip.y)/5 * 100

return False, 0
```

Figure 5.57: Checking

```python
def process_image(image_path):

    model_results = model.predict(image_path)

    for r in model_results:
        print(r.boxes)
        # Extract bounding box coordinates
        bounding_box = x1, y1, x2, y2 = r.boxes.xyxy[0].tolist()
        print("Bounding box coordinates (x1, y1, x2, y2):", x1, y1, x2, y2)
        bounding_box_tuple = tuple(bounding_box)

    bounding_box = tuple(int(value) for value in bounding_box_tuple)

    # Load image
    image = cv2.imread(image_path)

    # Convert image to RGB
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Detect pose landmarks
    results = pose.process(image_rgb)
    if results.pose_landmarks:
```

Figure 5.58: Image processing

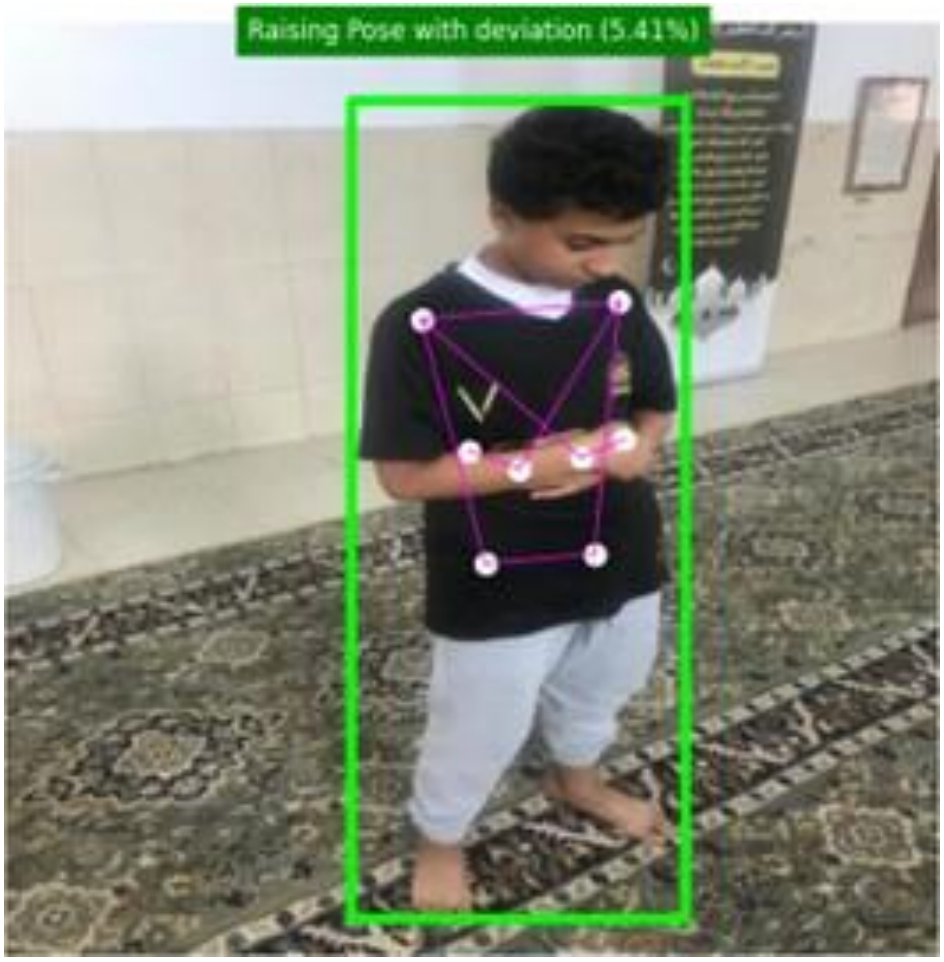


Figure 5.59: Result

Rasing with deviation (5.41%)

Server (API)

The Application Programming Interface is used to connect our interface to the model.

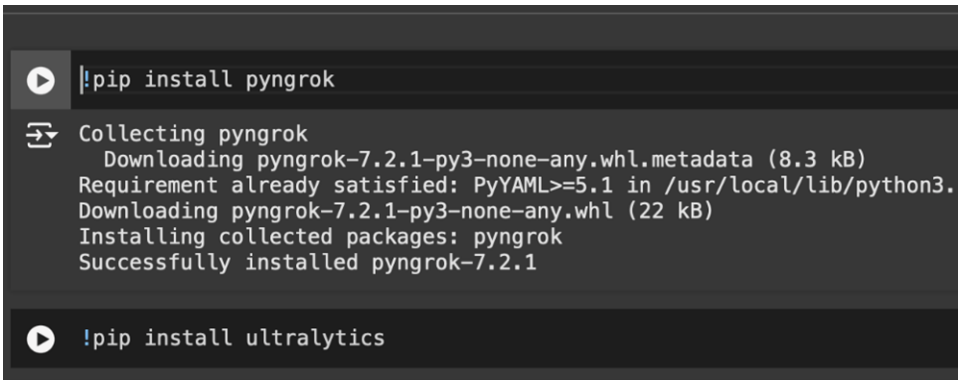


Figure 5.60: ngrok installed

We used ngrok API to provide global access to our application. And we installed ultralytics to have access to YOLO model.

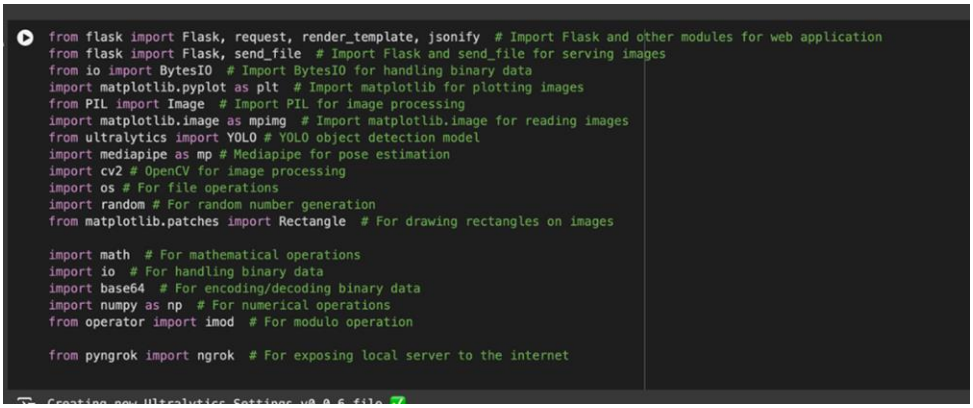


Figure 5.61: Flask

Flask is the web framework that serves as the backbone of communication between the user interface and the backend, which processes the posture analysis. It is lightweight, simple to use, and ideal for handling HTTP requests

How Flask is used:

- Handling User Requests: Users select the posture they want to check (e.g., Sujud, Ruku) and upload an image of themselves performing the posture. Flask handles the incoming HTTP requests from the front-end, receiving the image data and passing it to the backend for analysis.
- Error Detection and Communication: Once the image is received, Flask sends it to the machine learning models (YOLOv8 for object detection and Mediapipe for pose estimation). The model analyzes the image to detect any posture errors. If an error is detected, Flask returns a response indicating the type of error, serves a corrective video from the database.
- Exposing Local Server (Testing and Deployment): Flask, in combination with ngrok, allows the local server to be exposed to the internet. This is useful during development or testing phases, enabling access from any location and making it easy to deploy the application for broader use.

```
app = Flask(__name__)
ngrok.set_auth_token("2oq6itLuYMRRLskGyq4t7jnTwkw_3FGnQttFhNCNYQuzCrTFb")
public_url =  ngrok.connect(5000).public_url
print(public_url)

# Load a trained model
model = YOLO('/content/drive/MyDrive/Hassana/Run/runs/detect/train/weights/best.pt')

@app.route('/Ruku', methods=['POST', 'GET'])
def Detect_Ruku():
    imagefile = request.files['imagefile']
    image_path = imagefile.filename
    imagefile.save(image_path)

    results = model.predict(image_path)


    for r in results:
        # Extract bounding box coordinates
        bounding_box = x1, y1, x2, y2 = r.boxes.xyxy[0].tolist()
        print("Bounding box coordinates (x1, y1, x2, y2):", x1, y1, x2, y2)
        bounding_box_tuple = tuple(bounding_box)


    bounding_box_int = tuple(int(value) for value in bounding_box_tuple)

    print('The predicted class is ' + str(bounding_box_int))
```

Figure 5.62: API code

The code of the API is similar to the model training code with a few adjustments. We first gave ngrok API our authorization token. We got a public url using the port 5000, that gives the interface access to the api.We loaded the Trained model done by the model training code.Depending on what the user has chosen in the “Image uploading’ page, a “post/get” request will be sent to the API which will give access to the relative code.

Firebase

- **Realtime Database**:
  - **Activities**: This section stores data related to user uploads and posture detection processes. The primary key is the user ID, which branches out into unique activity IDs. Each activity record includes:
    - **Date**:The upload date, displayed in the history section.
    - **Image URL**:A link to the uploaded image.
    - **Status**:Indicates whether the posture was correct (0) or incorrect (1).
    - **Type**:Specifies the prayer posture (e.g., Ruku, Sujud, etc.).

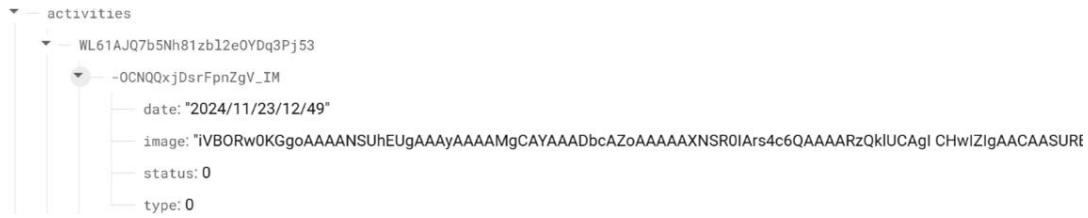


Figure 5.63: Activities

  - **Users**: Saves user information that is sent by the application interface like, date of birth, email, username and a unique id that is instantiated.by the firebase.

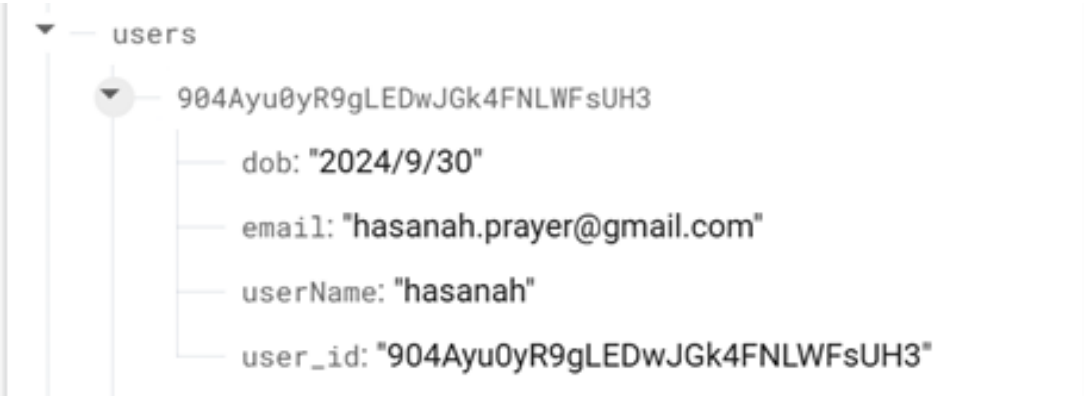


Figure 5.64: Users

  - **Videos**: It has correction videos that would appear to the user to explain the correct way to perform the posture. They are retrieved from the storage.

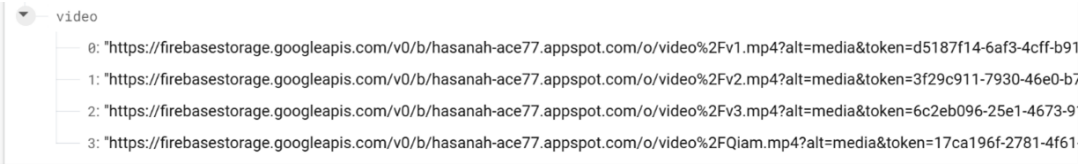


Figure 5.65: Videos

- **Authentication**:To authenticate the login attempt.
- **Storage**:Has profile images of the users. And the correction videos that are then retrieved by the Realtime database.

5.5 Conclusion

This chapter concludes with a thorough description of the project's architecture, including how hardware, software tools, and the underlying code that powers the application are all integrated. While Firebase provided effective user authentication and real-time data management, Android Studio made it easier to create an intuitive user experience. Through methodical training steps with different dataset sizes, the YOLOv8 model's training on Google Collab showed notable gains in posture detection accuracy. These components are effectively combined in the project to produce a reliable system that not only identifies and verifies prayer postures but also improves user experience by providing access to historical data and tailored feedback. All things considered, the effective deployment of these technologies highlights the project's potential to offer users insightful feedback and corrections, encouraging improved practice and comprehension of prayer postures.

Chapter 6

6.1 Introduction

In this chapter, we will discuss the various testing approaches used to ensure the functionality, integration, and usability of our project. We will cover Unit Testing, focusing on individual components, and Integration Testing, which checks the interaction between system modules. Additionally, we will outline Validation and System Testing to verify that the system meets all project requirements. Finally, we will explore Usability and Compatibility Testing, ensuring the system works well across different devices and provides a seamless user experience.

6.2 Scenarios for Unit Testing

One method of software testing is unit testing, which involves testing separate software application units or com- ponents separately. These units, which are usually functions or methods, are the smallest pieces of code and guarantee that they work as intended. Unit testing improves code quality, finds bugs early in the development cycle, and lowers the cost of later bug fixes. In our project we have 3 main test cases:

- manage user data (registration, logging and store data)
- upload image (detect image and verify correctness)
- view data (activity history)

Test case	Test scenario	Test data	Expected results	Actual results
1	store the registratio n informatio n	Correct personal information	The application stores personal information in the database	As expected
2	store the registratio n informatio n	Invalid formats or incorrect personal information	The application reject registration displaying error message	As expected
3	store the registratio n informatio n	Already registered emailin the database	The application reject registration displaying error message	As expected
4	Check the log ininformation	Lon in information founded in thedatabase	The application accept log in information usercan log in	As expected
5	Check the log ininformation	Log in information notfounded on the database	The application reject log in information displaying errormessage	As expected

Manage user data (registration, logging and store data)

Test case	Test scenario	Test data	Expected results	Actual results
1	The user choose Takbeer position to upload	Takbeer position image uploading	The application display a message (the image is uploaded successfully)and goes to the next step (detection)	As expected
2	The user clicks ok button to get the resulted image	The uploaded Takbeer position image	The application detect the Takbeer position in the image by checking shoulders alignment shoulders mustbe at same level horizontally	As expected
3	The user can verify whether or not the image of the Takbeer position he uploadedis correct	The detected Takbeer position image	The application verify the position ifit meets the conditions of Takbeer position	As expected

Upload Takbeer image (detect image and verify correctness)

Test case	Test scenario	Test data	Expected results	Actual results
1	The user choose Ruku position to upload	Ruku position image uploading	The application display a message (the image is uploaded successfully)and goes to the next step (detection)	As expected
2	The user clicks ok button to get the resulted image	The uploaded Ruku position image	The application detect the Ruku position in the image by measuringthe angels formed by the lines the measured angels are compared to the correct angel 90 degrees	As expected
3	The user can verify whether or not the image of the Ruku position he uploaded is correct	The detected Ruku position image	The application verify the position ifit meets the conditions of Ruku position	As expected

Upload Ruku image (detect image and verify correctness)

Test case	Test scenario	Test data	Expected results	Actual results
1	The user choose Sujud position to upload	Sujud position image uploading	The application display a message (the image is uploaded successfully)and goes to the next step (detection)	As expected
2	The user clicks ok button to get the resulted image	The uploaded Sujud position image	The application detect the Sujud position in the image by checking thecondition that both knees are positioned below the hips means the person is kneeling	As expected
3	The user can verify whether or not the image of the Sujud position he uploaded is correct	The detected Sujud position image	The application verify the position ifit meets the conditions of Sujud position	As expected

Upload Sujud image (detect image and verify correctness)

Test case	Test scenario	Test data	Expected results	Actual results
1	The user choose Raising position to upload	Raising position image uploading	The application display a message (the image is uploaded successfully)and goes to the next step (detection)	As expected
2	The user clicks ok button to get the resulted image	The uploaded Raising position image	The application detect the Raising position in the image by checking that the elbows is close to the bodymeans the arms are raised ensuringRaising position	As expected
3	The user can verify whether or not the image of the Raising position he uploadedis correct	The detected Raising position image	The application verify the position ifit meets the conditions of Raising position	As expected

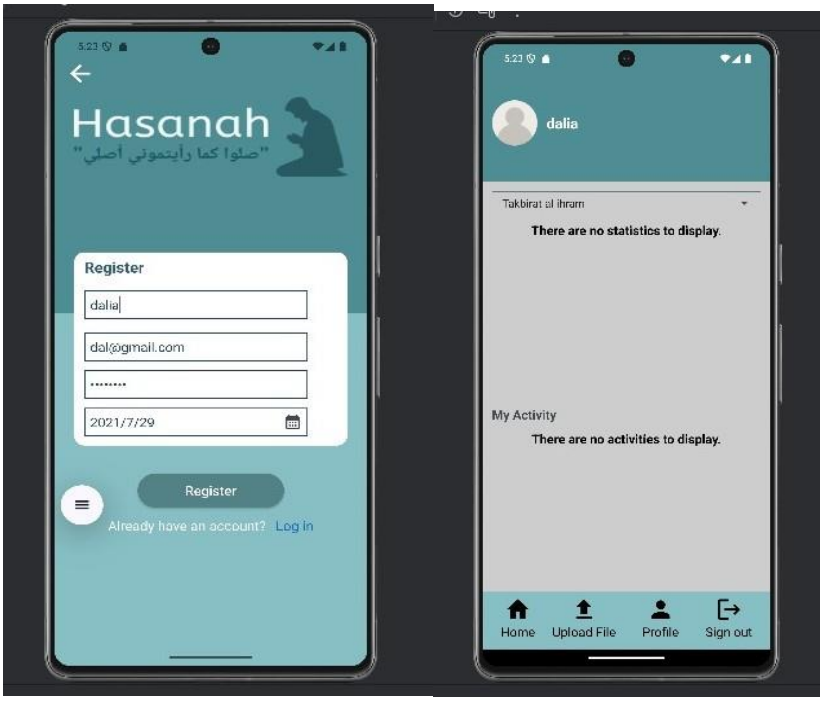
Upload Raising image (detect image and verify correctness)

Test case	Test scenario	Test data	Expected results	Actual results
1	The user view activity history	Recent activities	The application view all recent sessions and statistics of the sessions	As expected
2	The user view activity history	No recent activities	The application inform the user with a message there are no activities to display	As expected

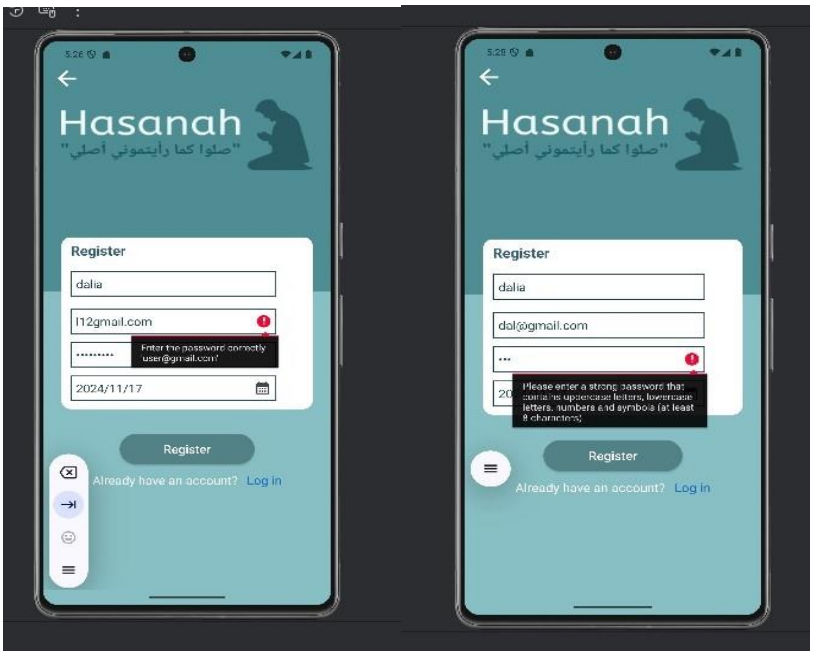
View data (activity history)

6.3 Unit Testing

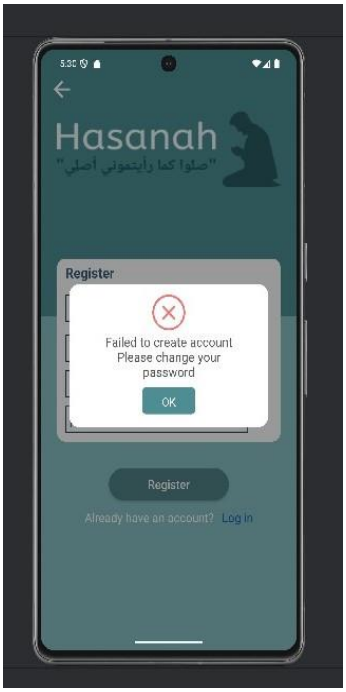
1- The user enter personal information the application stores the information in the database



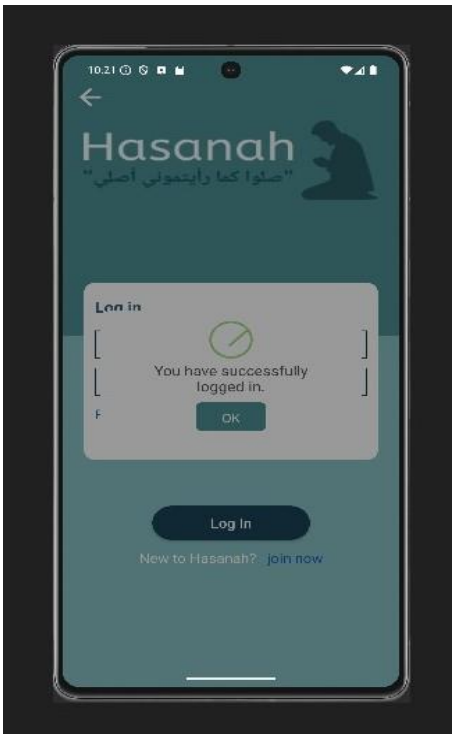
2- The user enters invalid format



3- Already registered email in the database

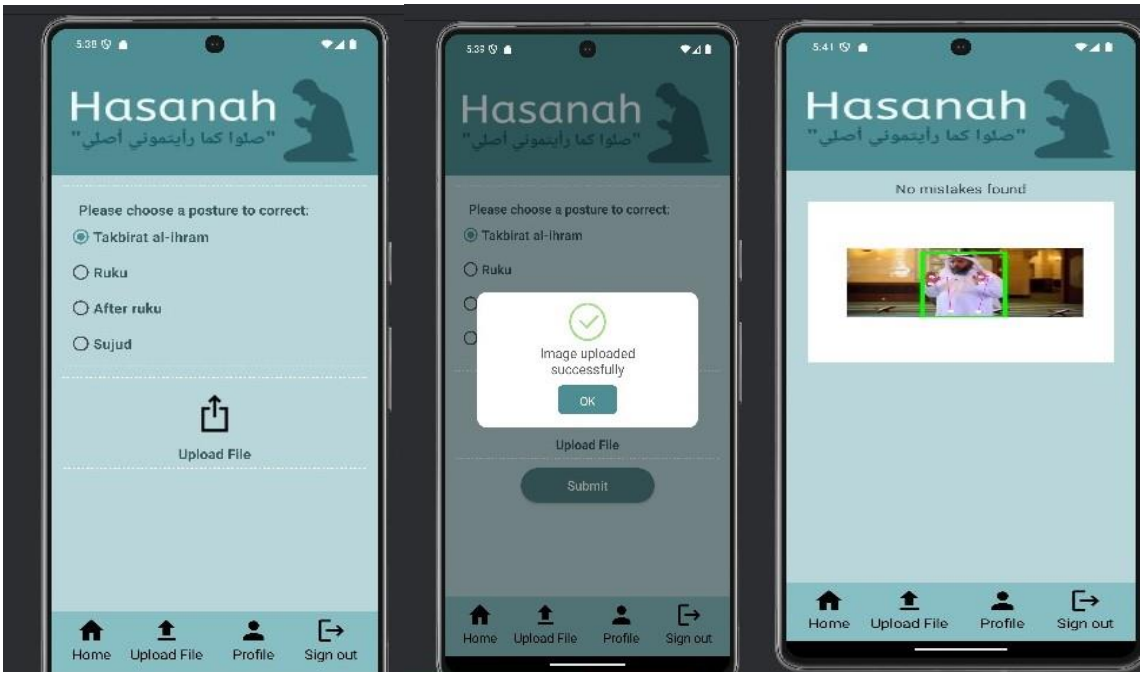


4- Logged in successfully log in information founded in the database

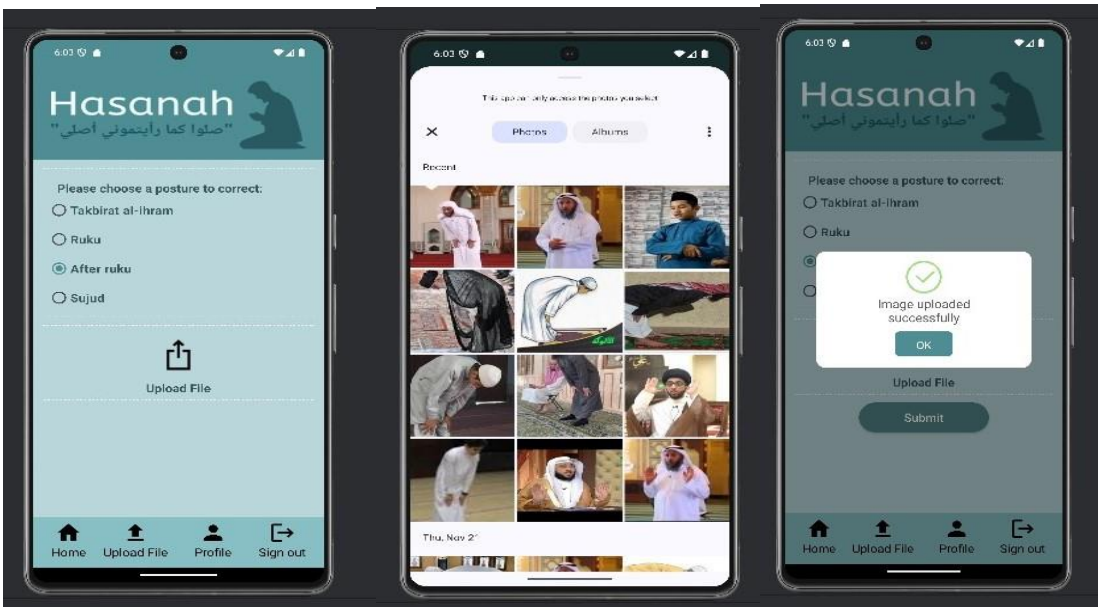


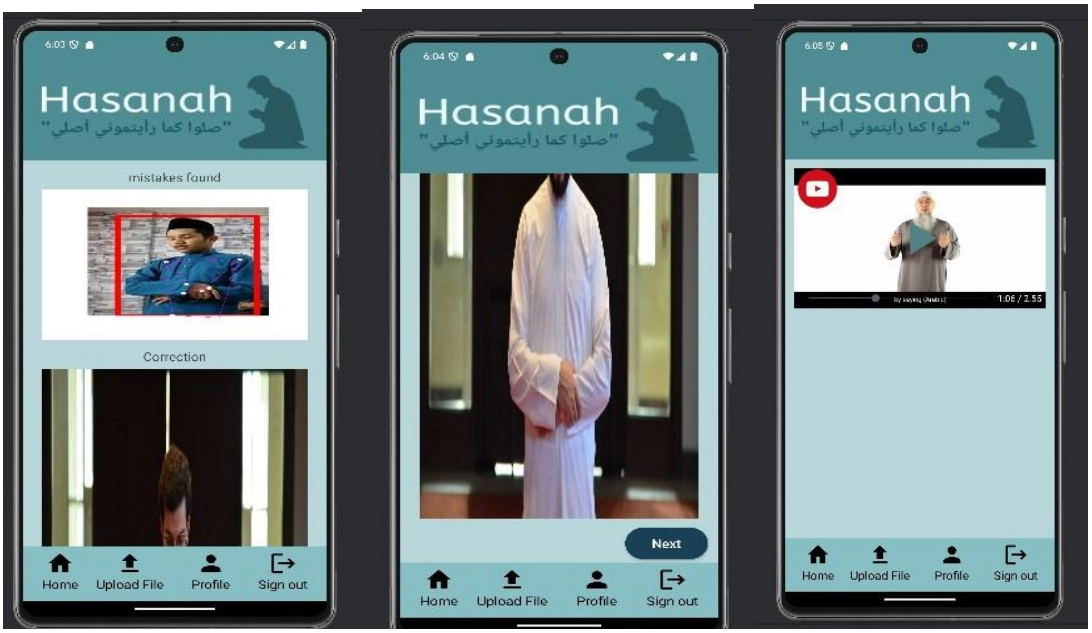
5- Log in error log in information not founded in the database

6- upload Takbeer image (detect image and verify correctness)

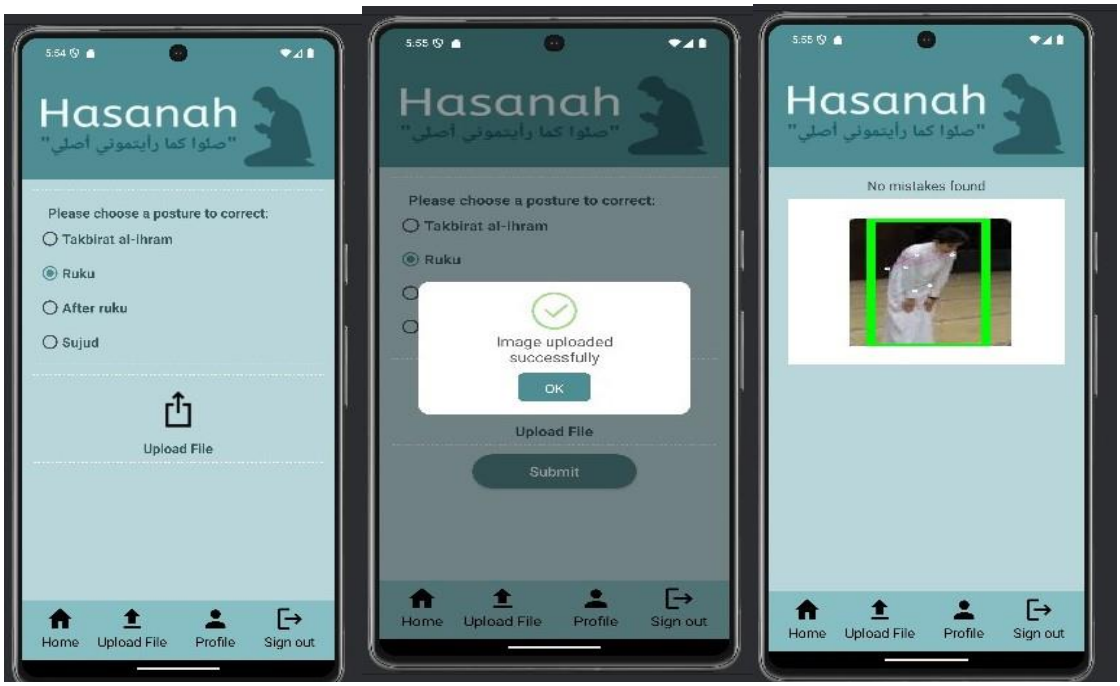


7- upload After ruku image (detect image and verify correctness)

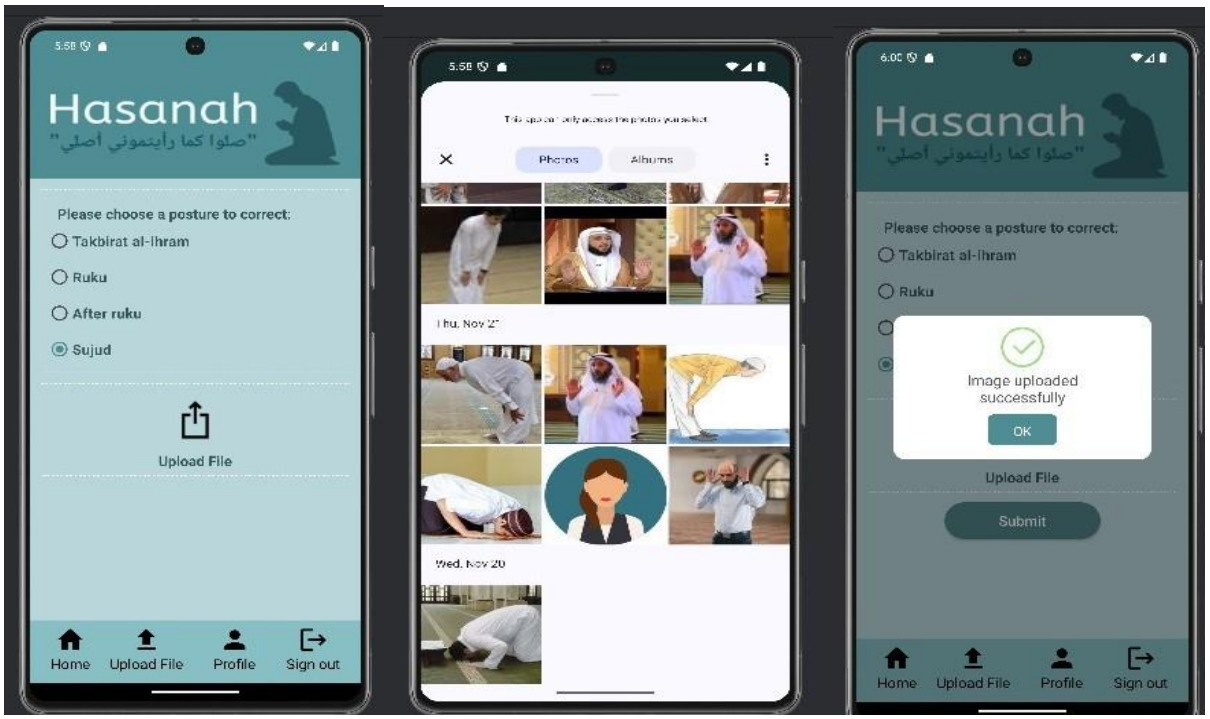




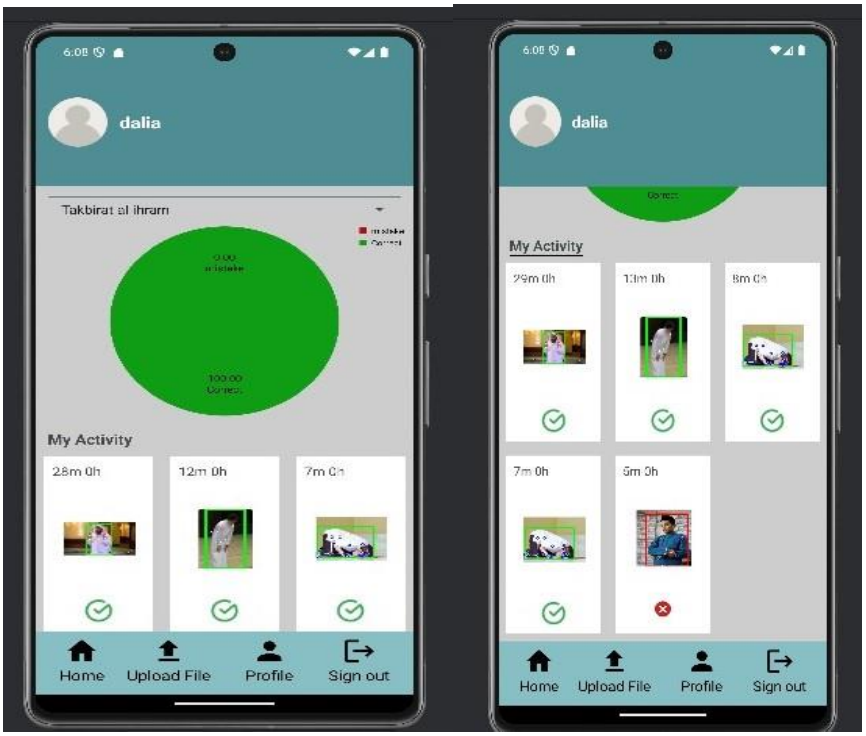
8- upload Ruku image (detect image and verify correctness)



9- upload Sujud image (detect image and verify correctness)



10- View activity history (display recent activities)



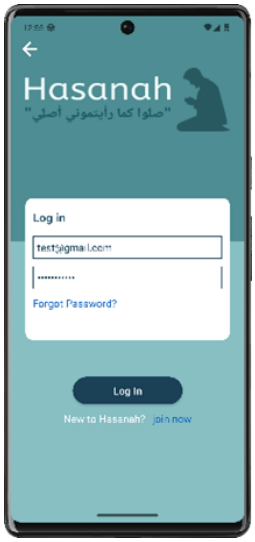
11- View activity history (no recent activities)

6.4 Integration Scenarios

Scenario	Expected Result	Actual Result
Register and log in to the app	1.1 The user registers or logs in to the app by entering a valid email and password. 1.2 If incorrect credentials are provided, the system displays an error message and allows retrying or password reset.	Pass
Upload a correct posture and view history	2.1 The app confirms the posture is correct, logs the result in the history section, and allows the user to view the updated log in the history section.	Pass
Upload an incorrect posture and provide feedback	3.1 The app identifies the posture as incorrect and provides corrective feedback (visual reference and video) while logging the result in the history section.	Pass
Select a posture from a radio button, upload a different one, and provide an error message	4.1 The model detects that the uploaded posture does not match the selected posture and provides a ”mistake is found” message.	Pass
Update profile information and view changes	6.1 The app saves the updated profile information and displays the updated details in the user’s profile section.	Pass
Change the password and log in with the new password	7.1 The app saves the updated password, invalidates the old one, and allows the user to log in with the new password.	Pass

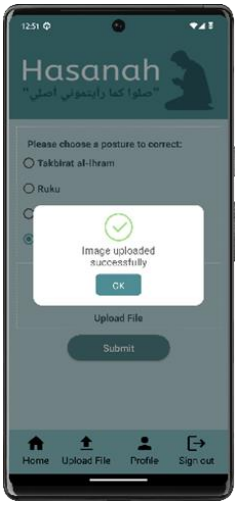
6.5 Integration Testing

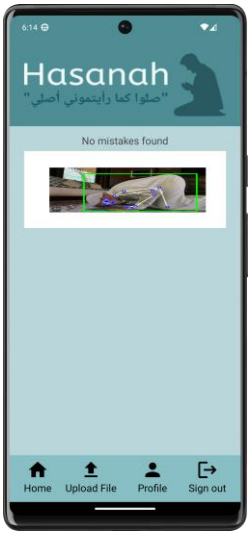
1. **Register and log in to the app:**

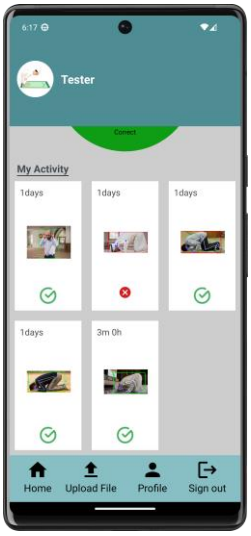


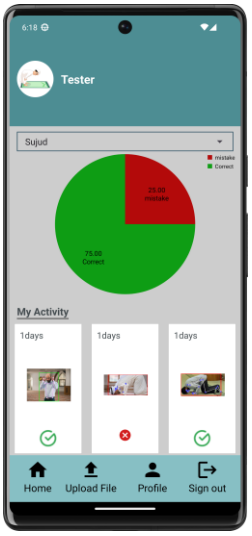
2. **Upload a correct posture and view history:**

   - **Sujud:**

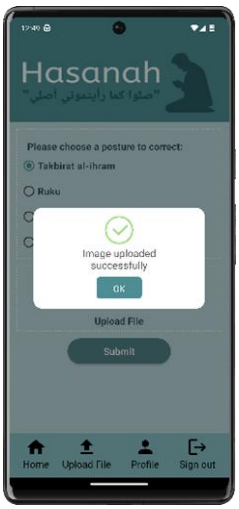


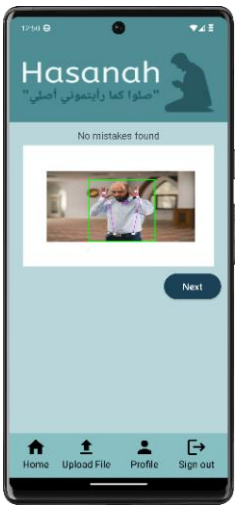


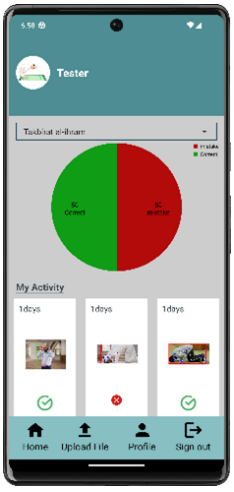




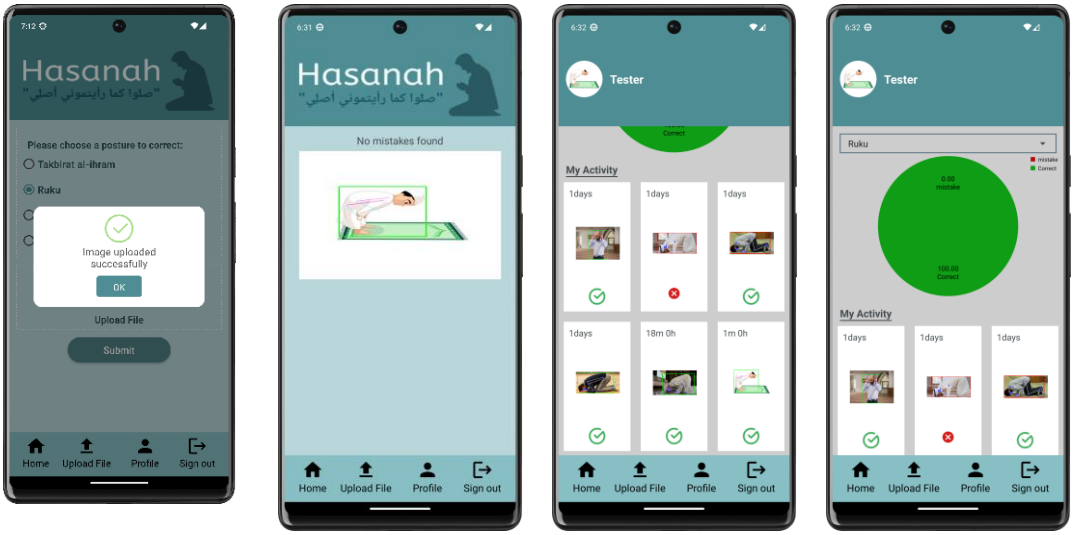
   - **Takbirat al-ihram:**



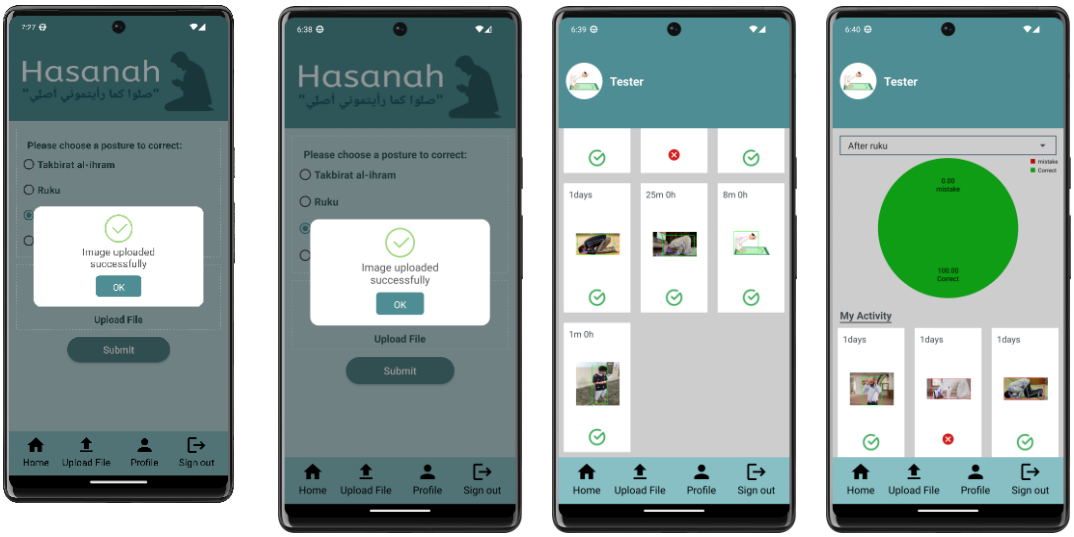




- **Ruku:**

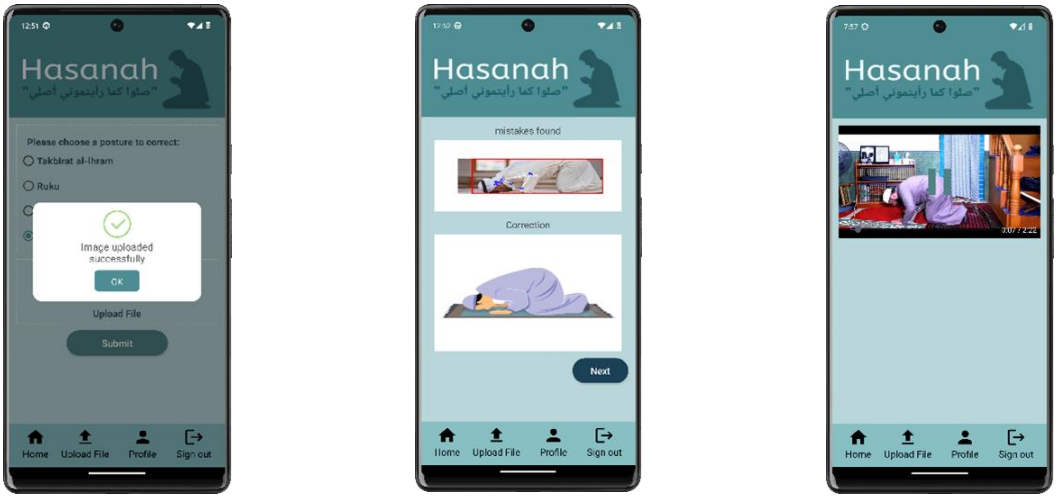


- **After ruku (Qiam):**

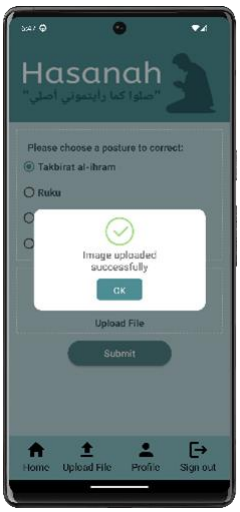


3. **Upload an incorrect posture and provide feedback:**

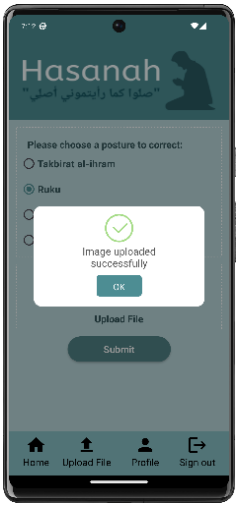
- **Sujud:**

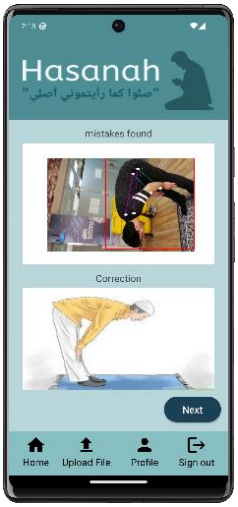


- **Takbirat al-ihram:**

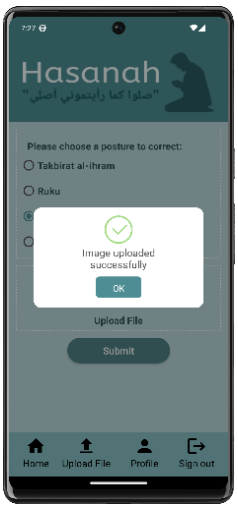


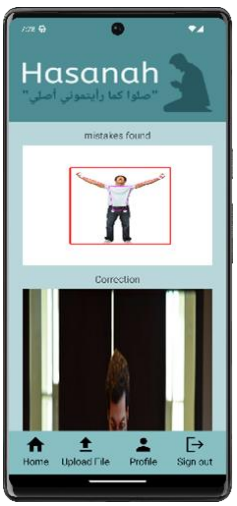
- **Ruku:**





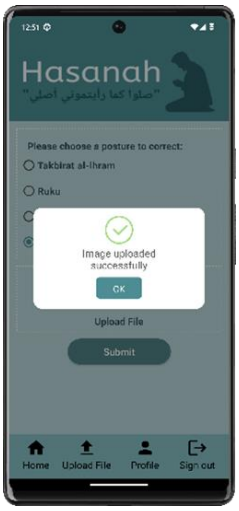
- **After ruku (Qiam):**

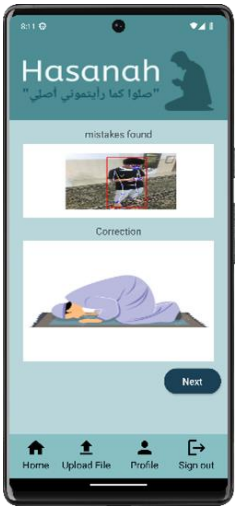




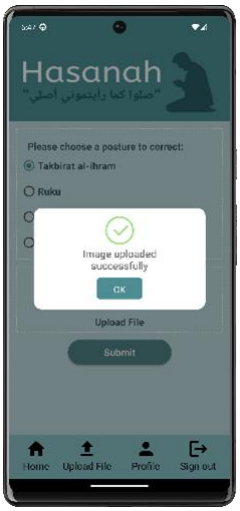
4. **Select a posture from a radio button upload a different one and provide error message:**

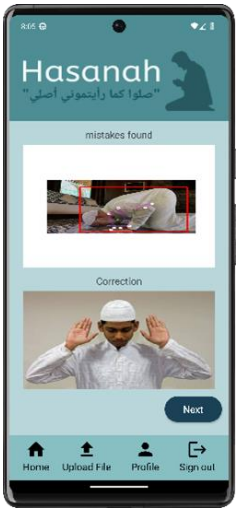
   - **Sujud:**



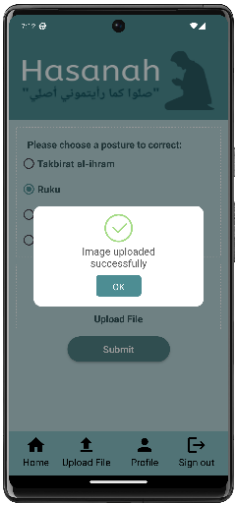


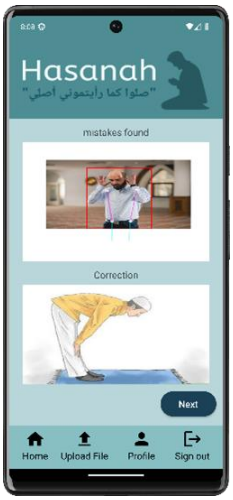
   - **Takbirat al-ihram:**



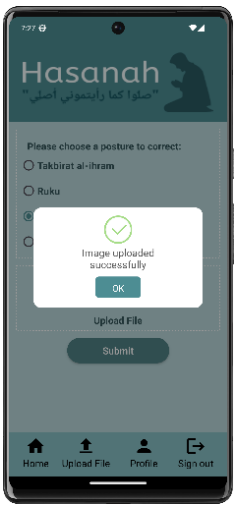


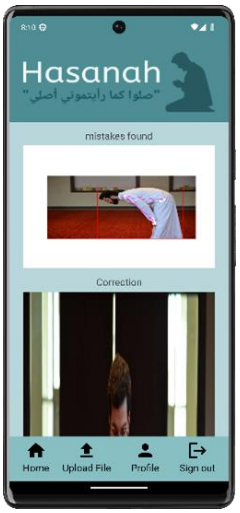
   - **Ruku:**



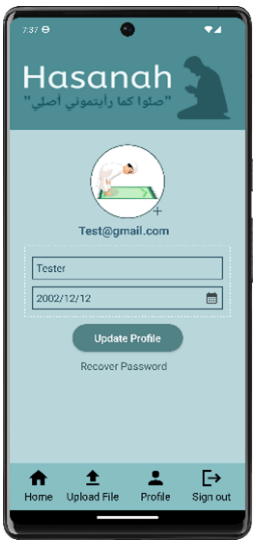


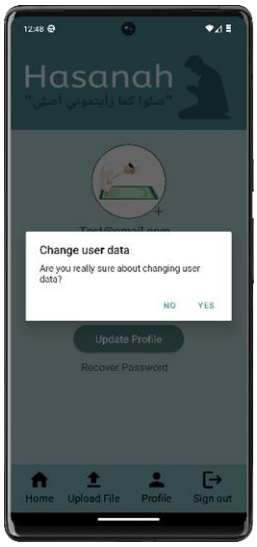
- **After ruku (Qiam):**

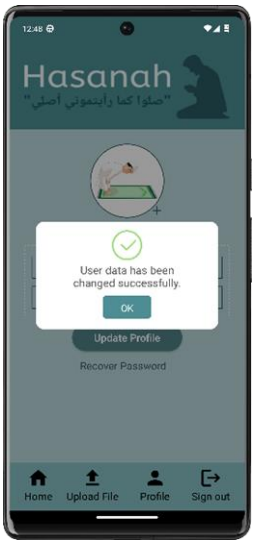


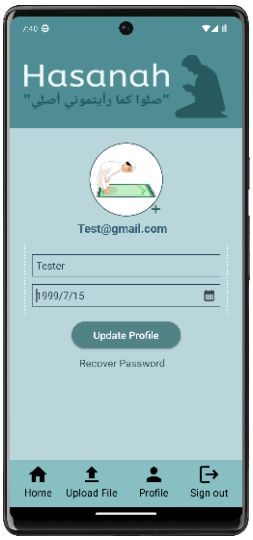


5. **Update profile information:**

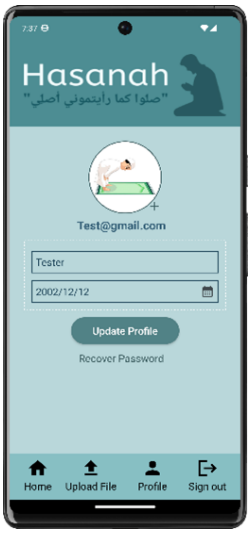


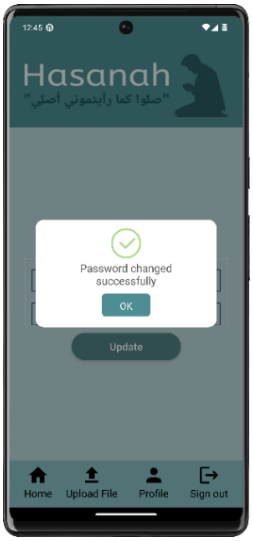






6. **Change password:**





6.6 Scenarios for Validation and System Testing

System testing is done after integration testing. It is a Black-Box testing that takes integration testing as input and creates output. It evaluates functional requirements. We will test successful registration and login followed by correct error detection and finally feedback and guidance.

6.6.1 Sign up and login:

We will evaluate the log-in and registering functionalities in table 6.3

functionality	**Expected Result**	**Actual Result**
Successful user registration	The user enters valid information for registration. The application will take user to homepage.	Pass
Successful user login	The user will enter correct email and password. The application will take user to homepage.	Pass

Table 6.1: Registration and login functions results

6.6.2 Error Detection:

We will evaluate the error detection functionality in table 6.4

functionality	**Expected Result**	**Actual Result**
Error Detection	The user clicks on the desired prayer posture, is able to upload a picture, the system recognizes the object and gives correct error detection points on the user's image.	Pass

Table 6.2: Error Detection function result

6.6.3 Feedback and Guidance

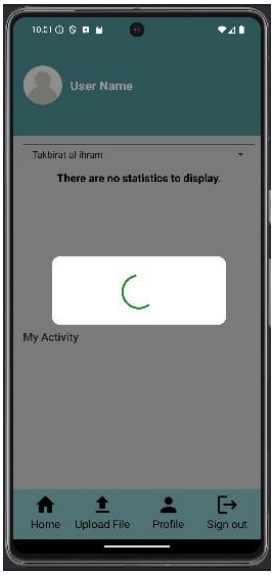
We will evaluate the feedback and guidance functionalities in table 6.5

functionality	**Expected Result**	**Actual Result**
Feedback and Guidance	The user chooses the desired posture and uploads image, if and only if the user has an error correction will be displayed.	Pass
Feedback and Guidance	The user chooses the desired posture and uploads image, if the user has an error the right correction video will be displayed.	Pass

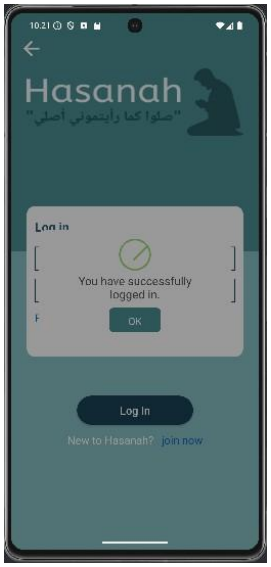
Table 6.3: Feedback and Guidance functions results

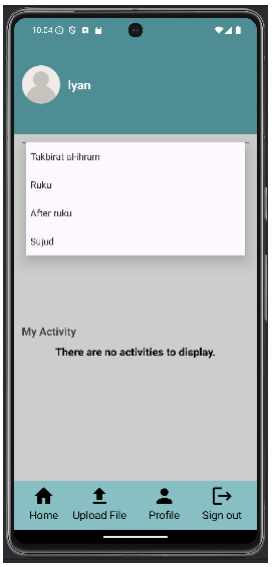
6.7 Validation and System Testing

6.7.1 Scenario 1: After successful registration, the application will take you to the home page.

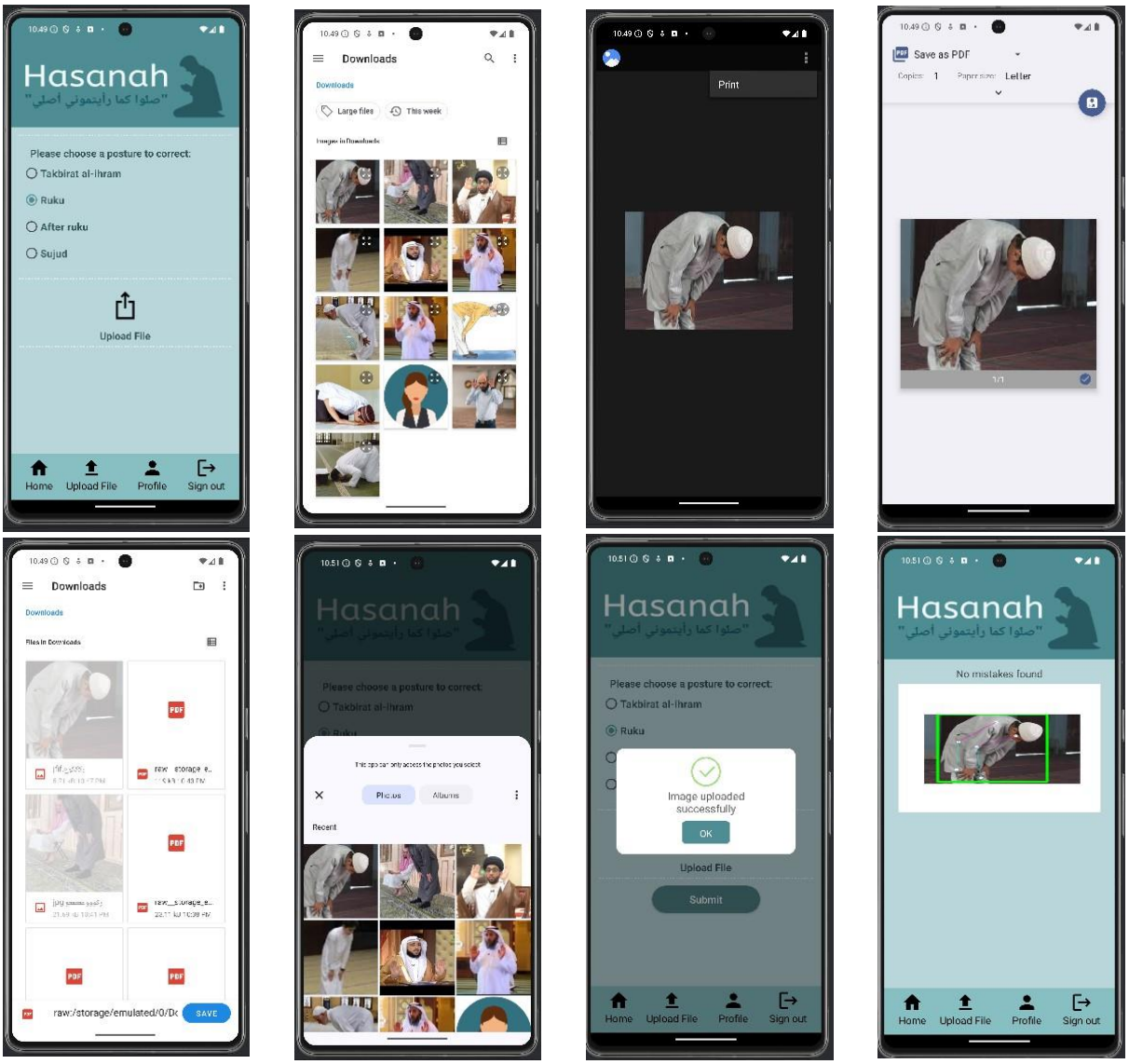


6.7.2 Scenario 2:After successfully logging in, the application will take you to the home page.

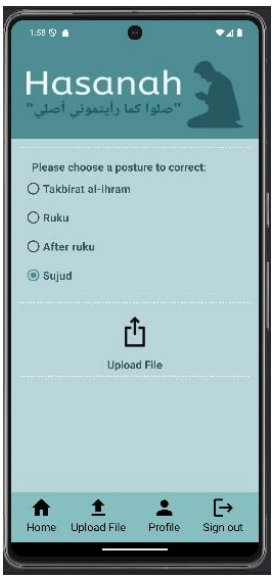


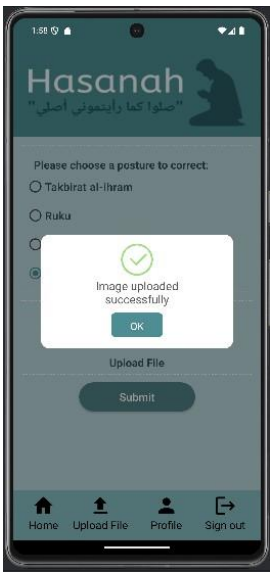


6.7.3 Scenario 3:Ruku

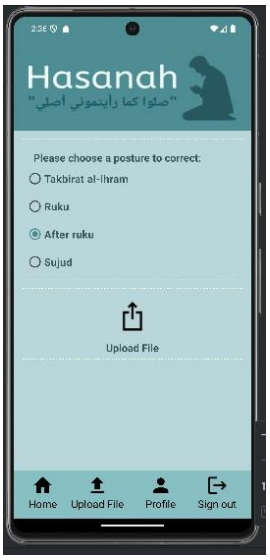


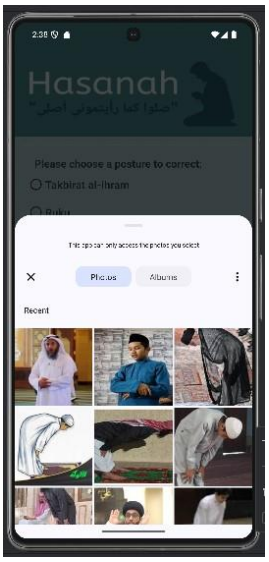
6.7.4 Scenario 3:Sujud

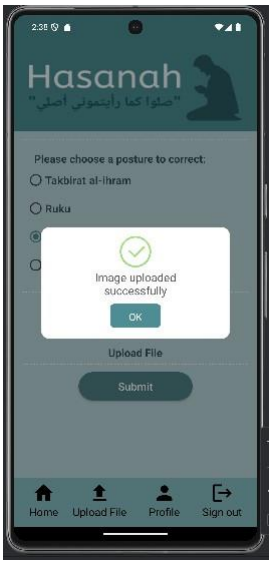


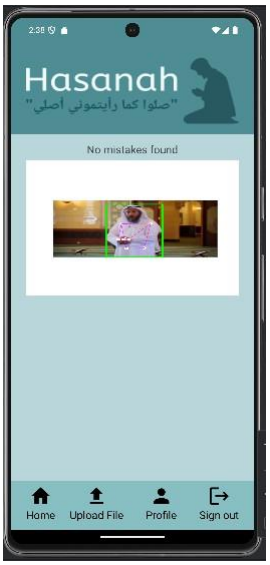


6.7.5 Scenario 3:after ruku

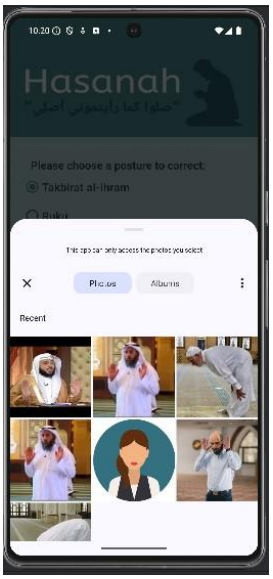


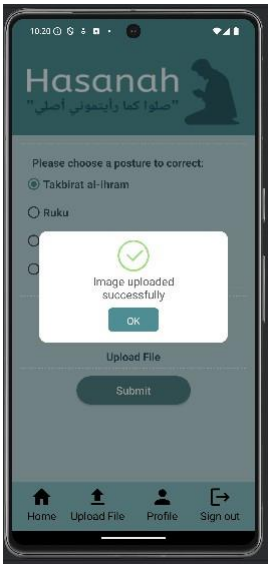


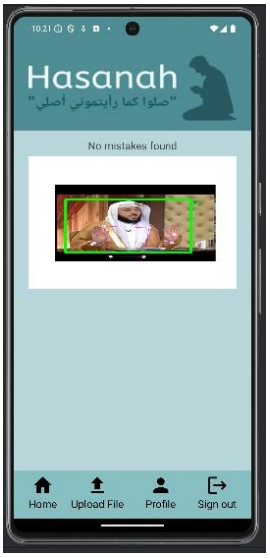




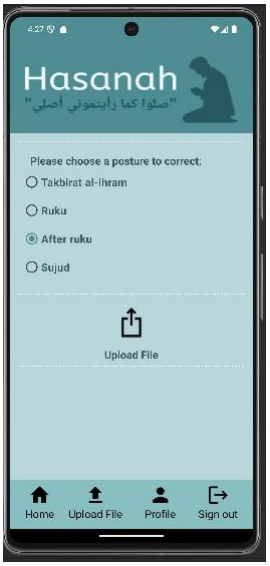
6.7.6 Scenario 3:takbirat al-ihram

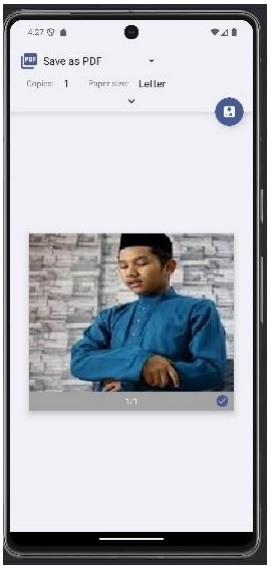




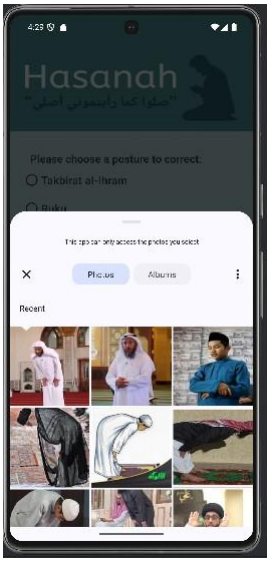


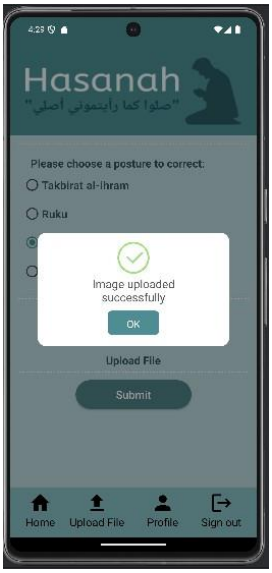
6.7.7 Scenario 4:after ruku

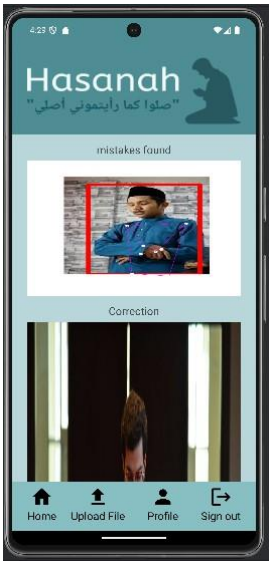





Downloads
Downloads
Files in Downloads
SAVE

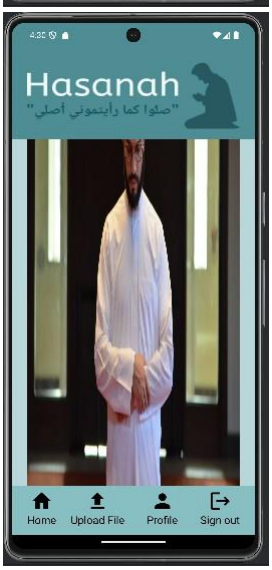

Hasanah
"صلوا كما رأيتموني أصلي"
Please choose a posture to correct:
Takbirat al-Ihram
Ruku
Recent


Hasanah
"صلوا كما رأيتموني أصلي"
Please choose a posture to correct:
Takbirat al-Ihram
Ruku
Image uploaded successfully
OK
Upload File
Submit
Home
Upload File
Profile
Sign out


Hasanah
"صلوا كما رأيتموني أصلي"
mistakes found
Correction
Home
Upload File
Profile
Sign out

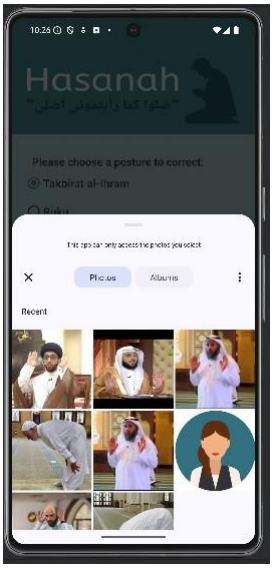

Hasanah
"صلوا كما رأيتموني أصلي"
Correction
Home
Upload File
Profile
Sign out

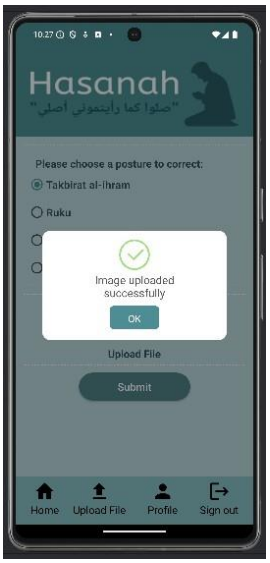

Hasanah
"صلوا كما رأيتموني أصلي"
Home
Upload File
Profile
Sign out

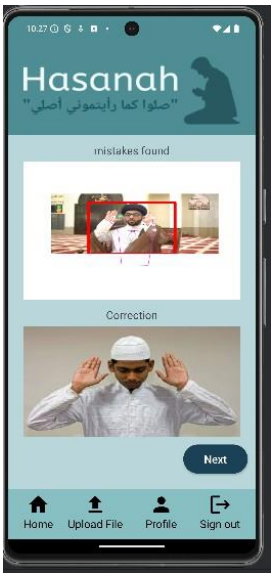

Hasanah
"صلوا كما رأيتموني أصلي"
Home
Upload File
Profile
Sign out


Hasanah
"صلوا كما رأيتموني أصلي"
Next
Home
Upload File
Profile
Sign out

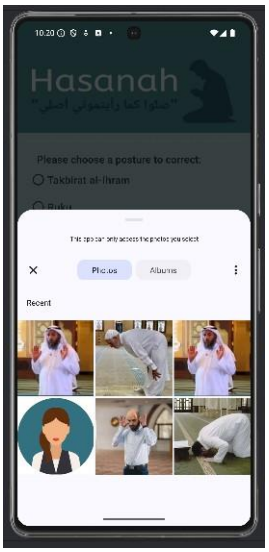
6.7.8 Scenario 4:takbirat al-ihram

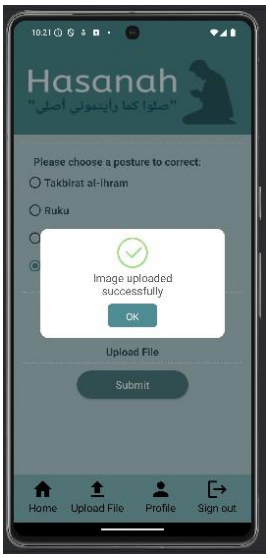


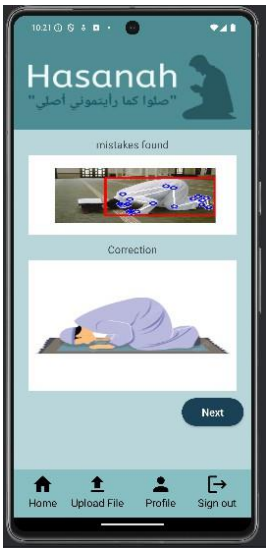




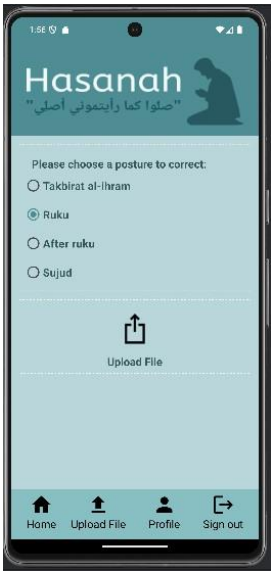
6.7.9 Scenario 4:Sujud

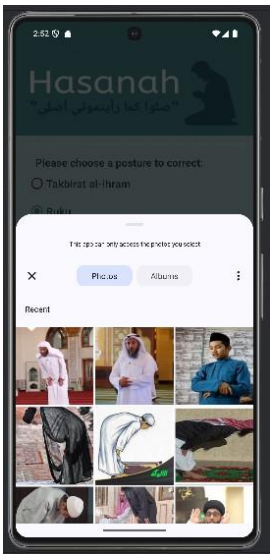


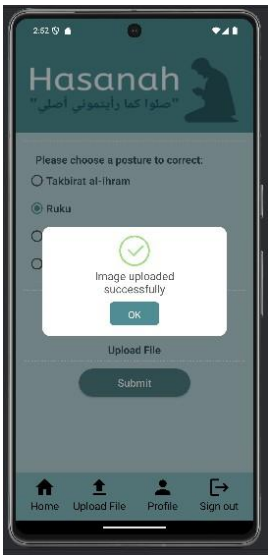


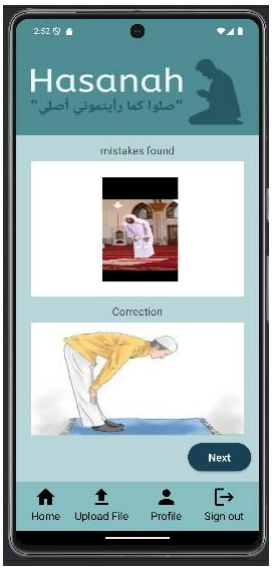


6.7.10 Scenario 4:Ruku



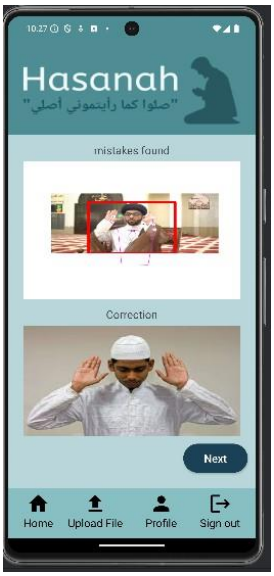




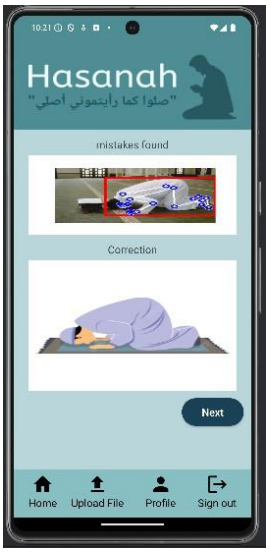


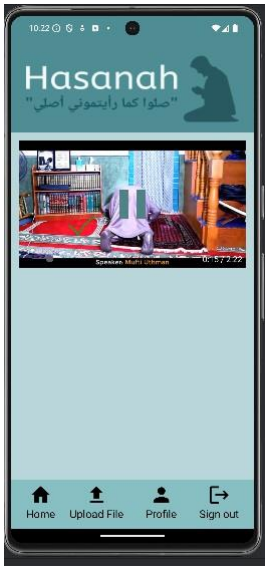
6.7.11 Scenario 5:After clicking on next, the video of the correction of after ruku will appear:

6.7.12 After clicking on next, the video of the correction of takbirat al-ihram will appear:

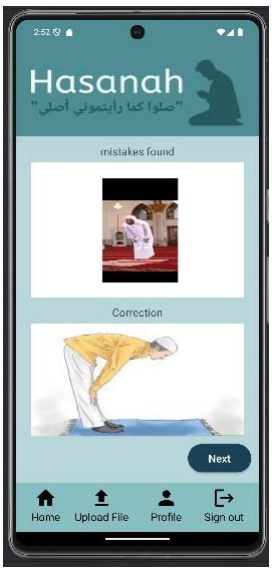


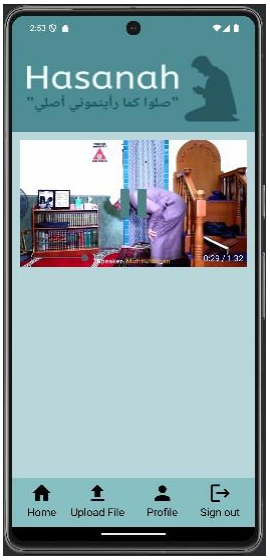
6.7.13 After clicking on next, the video of the correction of sujud will appear:





6.7.14 After clicking on next, the video of the correction of ruku will appear:





6.8 Conclusion

Ultimately, we evaluated our system using three primary types of testing: unit testing, integration testing, and system testing. Each type was conducted with specific scenarios and corresponding results. This testing phase aimed to confirm that the results align with the requirements and to verify that the majority of the system's functionalities are operating as intended.

Conclusion

In conclusion, “Hasenh” represents an innovative step in leveraging artificial intelligence technologies to enhance the process of learning and performing prayer. It addresses the needs of a wide range of users, including new Muslims, children, and individuals seeking to verify their prayer postures. By offering an accessible and educational platform, the project contributes to simplifying and improving the understanding of prayer practices.

Throughout this project, we integrated state-of-the-art AI techniques such as YOLOv8 for object detection and MediaPipe for body landmark identification, enabling precise analysis of prayer postures. Despite the technical challenges faced during development, our team successfully overcame them through collaboration and perseverance, achieving high accuracy and efficiency in the system.

Looking ahead, we aim to expand the project’s scope to include full prayer analysis, support for video uploads, and multi-language capabilities to cater to a broader audience. We believe “Hasenh” has the potential to become an essential tool for educating and assisting Muslims worldwide in perfecting their prayers.

This project stands not only as an academic achievement but also as a foundation for future advancements in utilizing AI technologies to serve the Islamic community globally.

Bibliography

[1] Assimalhakeem. ”Where to raise the hand when saying Allahu Akbar.” *YouTube*, 11 April 2020, https://youtu.be/lDKLzcTqdKs?si=PGPhbxYX11lwJPVA.

[2] Uthman Ibn Farooq. ”How to do Ruku.” *YouTube*, 1 Aug. 2021, https://www.youtube.com/watch?v=ToBJhhiUgoA.

[3] Assimalhakeem. ”What to say after raising up from Ruku.” *YouTube*, 22 Aug. 2022, https://youtu.be/Da5cJIloRz4?si=09zbFq-EgklmULQd.

[4] Uthman Ibn Farooq. ”Correct way of doing Sujud.” *YouTube*, 3 Aug. 2021, https://youtu.be/izmtVT_B_48?si=UhEUgQr-KYR4ZRz4.

[5] Huda Aziz, Hala Qassa, Badriah, Rahaf. ”Istaqim: An Assistant Application to Correct Prayer for Arab Muslims.” *ResearchGate*, 2022, https://www.researchgate.net/publication/369410589_Istaqim_An_Assistant_Application_to_Correct_Prayer_for_Arab_Muslims.