

Capstone Project Phase B

**24-1-D-9**

**Earth Keepers**

[**https://github.com/raeedAtaria1/EarthKeepers**](https://github.com/raeedAtaria1/EarthKeepers)

**Supervisor: Dr. Reuven Cohen**

**Students:**

**Suhel Khoury 212103527**

**&**

**Raeed Ataria 212154629**

[**Suhel.Khoury@e.braude.ac.il**](mailto:Suhel.Khoury@e.braude.ac.il)

[**Raeed.Ataria@e.braude.ac.il**](mailto:Raeed.Ataria@e.braude.ac.il)

[Abstract 3](#_Toc177856370)

[1. Introduction 4](#_Toc177856371)

[2. The Product 5](#_Toc177856372)

[2.1 The Product Architecture 5](#_Toc177856373)

[2.2 The Interaction with the Application 7](#_Toc177856374)

[2.3 Activity Diagram 8](#_Toc177856375)

[2.4 Which technologies we used 9](#_Toc177856376)

[2.4.1 YOLO: 9](#_Toc177856377)

[2.4.2 MediaPipe 10](#_Toc177856378)

[2.5 Environments 11](#_Toc177856379)

[2.6 Dataset 11](#_Toc177856380)

[2.6.1 YOLO Dataset 11](#_Toc177856381)

[2.6.2 MediaPipe Dataset 13](#_Toc177856382)

[2.7 Results 14](#_Toc177856383)

[2.7.1 Results of Training YOLO 14](#_Toc177856384)

[2.7.2 Results of Training MediaPipe 19](#_Toc177856385)

[2.8 Testing 20](#_Toc177856386)

[2.9 The process of crafting software 21](#_Toc177856387)

[3. Challenges 22](#_Toc177856388)

[3.1 Description of Challenges Faced 22](#_Toc177856389)

[3.2 Conclusions: 23](#_Toc177856390)

[3.3 Decision-Making Process 23](#_Toc177856391)

[3.4 Overall Effectiveness of the Work 24](#_Toc177856392)

[3.5 Meeting the Project’s Criteria: Explanation and Justification 24](#_Toc177856393)

[4. User Guide 25](#_Toc177856394)

[5. Maintenance Guide 33](#_Toc177856395)

[6. References 34](#_Toc177856396)

# Abstract

Our app “Earth Keepers” addresses the growing issue of uncollected trash in public spaces by developing a gamified Android app that detects and categorizes trash in real-time using the phone’s camera. The app rewards users with points for collecting detected items, encouraging active participation in environmental cleanup efforts.

We trained the YOLOv8 object detection model on a customized dataset containing over 15,000 images, including augmented data from 5,500+ original pictures of various trash classes, such as paper, glass, plastic, organic, and metal. This tailored approach ensured precise and efficient real-time detection. MediaPipe, a hand tracking tool, was also used to accurately recognize gestures related to grabbing trash items.

Key challenges included optimizing the app’s user interface and ensuring efficient real-time performance on Android devices. By focusing efforts on these challenges, we achieved a detection accuracy of over 97% for trash items and above 90% for hand recognition, while maintaining smooth and responsive app operation.

# 1. Introduction

Our Android app "Earth Keepers," is an innovative mobile application designed to address the increasing problem of uncollected trash in public spaces. The app leverages advanced computer vision technologies, specifically YOLOv8 for real-time object detection and MediaPipe for hand tracking, to identify and categorize trash items as users collect them.

To enhance user engagement, our app incorporates several gamification elements. Users earn points for every piece of trash they collect, with higher points awarded for items that have a greater environmental impact. The app also fosters a competitive spirit by allowing users to compete with others. Additionally, each time an item is collected, the app provides immediate audio feedback, notifying users of the detected item and the points they've earned. These features work together to make the cleanup process more engaging and enjoyable.

The primary audience for this application includes environmentally conscious individuals and community groups committed to reducing litter and enhancing public spaces. By making the process enjoyable and interactive, the app aims to engage a broader audience, including young people and those who may not have previously participated in environmental activities.

**Key Features:**

**Real-Time Detection and Classification:** The app uses the smartphone's camera to detect trash items in real-time, providing immediate feedback to users through auditory signals when trash is correctly identified and collected.

**User Engagement through Gamification:** Users earn points for each piece of trash they collect, which fosters a sense of competition and engagement. The app tracks these points and displays them on leaderboards, encouraging users to improve their scores and contribute more to the community's cleanup efforts.

**User-Friendly Interface:** The app's design prioritizes ease of use, with intuitive navigation and clear visual cues to guide users through the trash collection process. The interface displays the types of trash detected, the points earned, and other relevant information in real-time.

# 2. The Product

## 2.1 The Product Architecture

**Client App:**

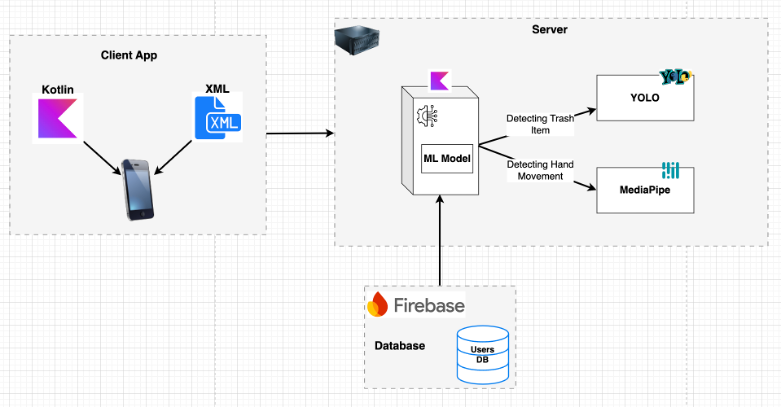
* The client app is developed using Kotlin for the application logic and XML for defining the user interface layouts. This combination is used for building a responsive and interactive Android application that runs on users' mobile devices.
* Kotlin handles the app's functionality, interactions, and real-time data processing, while XML defines the visual components and layouts of the app.

**Server:**

* The server component includes a Machine Learning (ML) model responsible for processing incoming data from the client app.
* YOLO (You Only Look Once): This object detection algorithm is utilized by the ML model for identifying and categorizing trash items detected by the phone’s camera in real-time.
* MediaPipe: This framework is integrated into the ML model to detect and process hand movements, specifically recognizing gestures related to grabbing trash items.
* The server processes the detection tasks and sends back the results to the client app, enabling real-time interaction and feedback.

**Database:**

* The app uses Firebase as its backend database service, specifically to manage and store user-related information.
* The Users DB component within Firebase handles data related to user accounts, sessions, and points scored during trash collection activities.



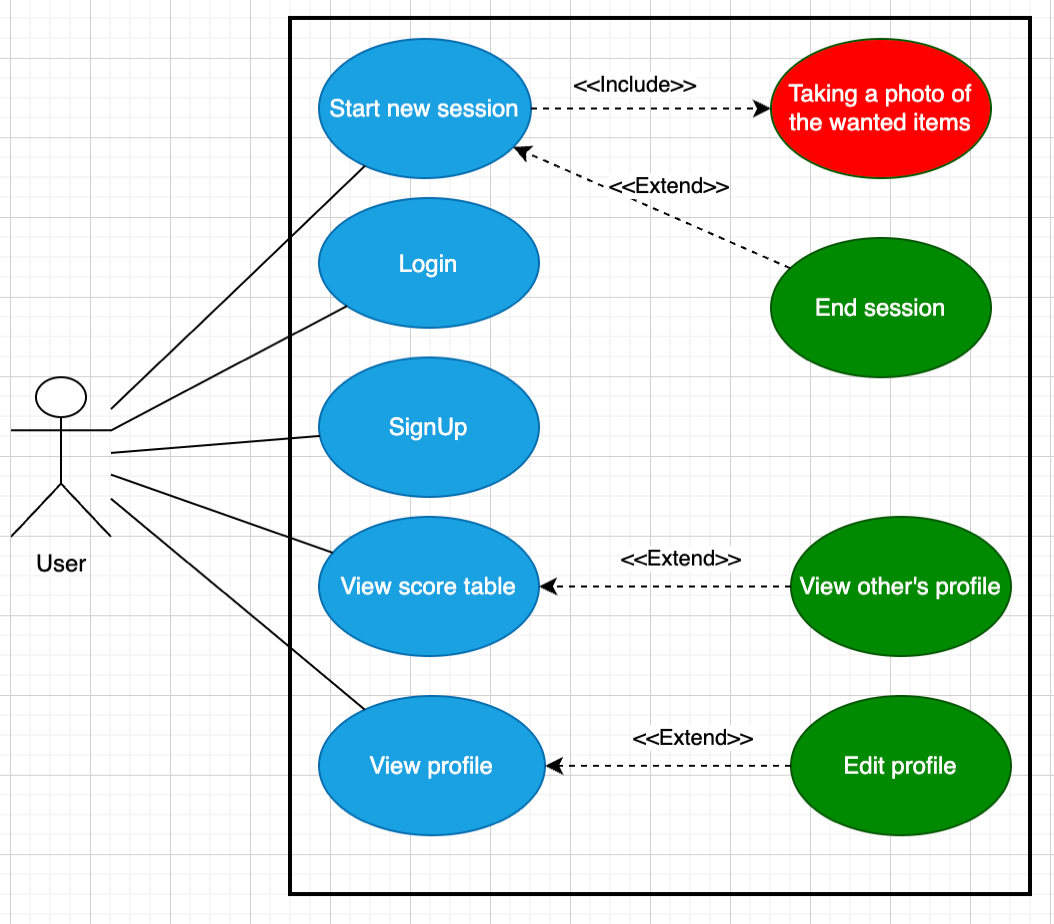
*Figure 1: Software Architecture Diagram*

## 2.2 The Interaction with the Application

The interaction with the application is described by the Use Case diagram, that represents the functionalities or use cases of a system from a user's perspective. It depicts the interactions between the system and its actors.

The use cases are as follows:

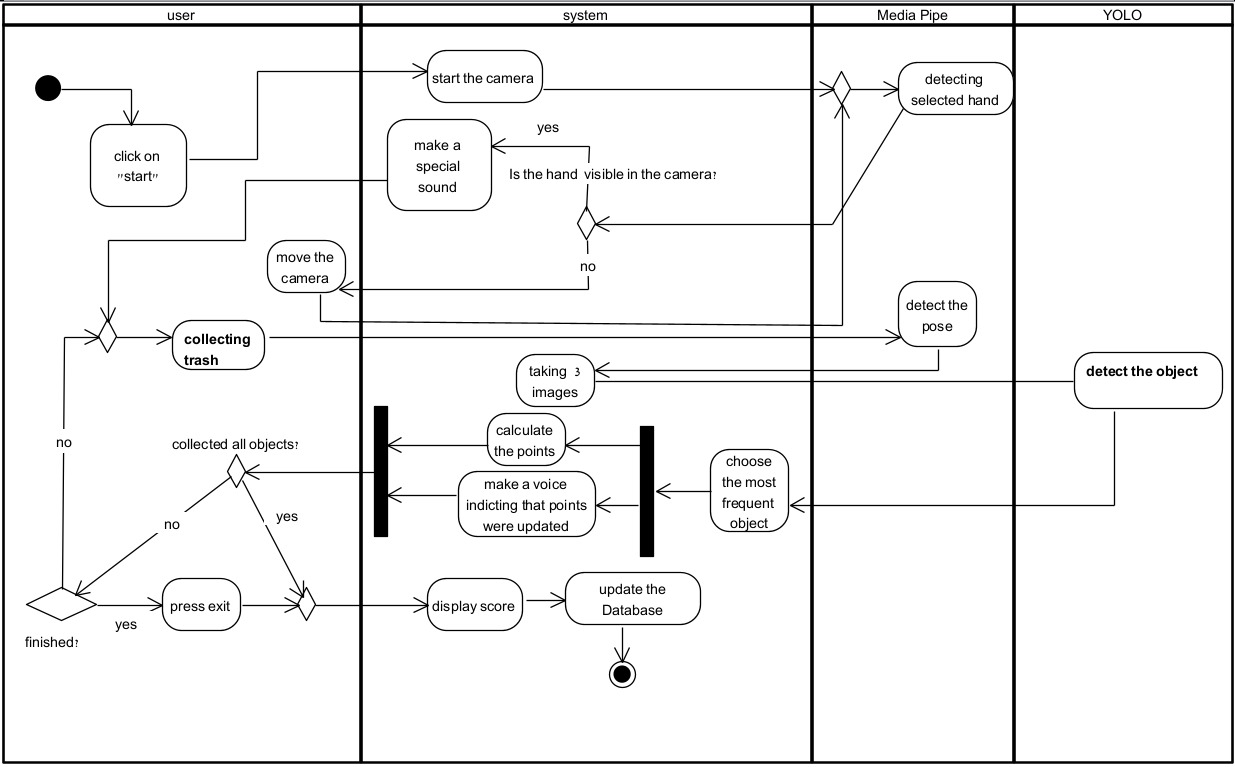
1. **Sign Up:** create an account and register with the system.
2. **Login:** user authenticate and log in to the system.
3. **Start new session:** The app allows users to initiate a new session for trash collection. First, the user is prompted to take a photo of the area containing the items they intend to collect. After capturing the photo, they can proceed to the real-time collection process. During the collection phase, the user has the flexibility to end the session at any time.
4. **View score table**: view a table or scoreboard, which displays scores or rankings of other users. The user can also **“view other’s profile”**.
5. "**View profile**": view the personal profile within the system. The user can also **“edit profile”** in order to modify his own profile’s information.



*Figure 2: use case diagram*

## 2.3 Activity Diagram

This activity diagram illustrates the primary actions users take within the system and outlines how the system responds to these actions.

*Figure 3 Activity Diagram*

Before starting the collection session, the user takes a photo of all the items they intend to collect. The program then calculates the maximum number of points available for the session based on the identified items.

The user opens a session for collecting trash, the camera starts and Media-pipe starts working in the background. Media-pipe follows the preferred hand of the user (the hand that the user will use to pick up the trash), and it recognizes when the user picks up the trash, If the camera is directed on the selected hand the user will hear a special voice (indicating that he can start collecting the trash). The user will hear : “You lifted [item] and earned [number of points] points!” when the user picck the item.

The program uses YOLO to identify the object that was picked and the program gives the user the points according to the class of trash that was collected. The user can select to finish whenever he wants, and the score will be presented.

\*The camera will take 3 pictures when the object is lifted so the classification of the trash item would be more accurate, if the camera identified that the object is “x” in 2 or more pictures then the user will get the points according to the identified item.

## 2.4 Which technologies we used

### 2.4.1 YOLO:

In our project, we employed YOLOv8. YOLO (You Only Look Once) is a popular object detection algorithm due to its efficiency and effectiveness. Predicting bounding boxes and class probabilities directly from the full image in one evaluation.

The key idea behind YOLO is the partition of the input image into grid cell that are processed by a series of CNN convolution and pooling layers with output of a very small matrix. Lastly, the output matrix is processed by a fully connected layer that predicts the bounding boxes and provides the class probabilities for the objects located within each grid cell.

This approach enables YOLO to achieve a real-time performance, as it only requires a single pass through the neural network to make predictions for the entire image [3][4].

YOLOv8 offers multiple variants, including YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x, each differing in size, speed, and accuracy. **We chose YOLOv8s** because it balances performance and computational efficiency, making it suitable for deployment on mobile devices. Specifically, YOLOv8s has a mean Average Precision (mAP) of 44.9, a moderate parameter count (11.2M), and a reasonable inference speed on both CPU (128.4 ms) and GPU (1.20 ms), which is optimal for real-time applications like our app.

To achieve accurate trash detection, we built our own custom data set. The training involved iterating through this dataset, adjusting the model’s parameters to minimize a predefined loss function, and optimizing the model’s capability to detect and classify trash objects. Post-training, the YOLOv8 model effectively identifies various types of trash with high precision in images and video frames.

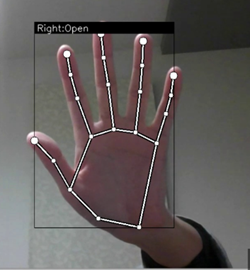


*Figure 4: YOLO-identified image featuring various trash items.*

### 2.4.2 MediaPipe

MediaPipe is an efficient and usable framework designed for constructing machine learning pipelines tailored for processing time-series data such as video and audio. It relies on OpenCV for video processing and FFMPEG for audio handling, and more other tools like OpenGL TensorFlow libraries. It is compatible with various platforms including Desktop, Android, and iOS. In fact, Nowadays, it has been utilized by Google, like Google Lens, Google Photos, and Gmail, and many other projects [1][4].

We integrated MediaPipe to facilitate the identification of hand gestures related to object manipulation. We trained MediaPipe to recognize specific hand movements (closed fists or open palms), which are indicative of grabbing actions. Our system uses MediaPipe to count how many times an object is grabbed and employs YOLO to determine the nature of the item.



*Figure 5: Detecting that the hand is open*

## 2.5 Environments

Our project was developed using Android Studio, a robust IDE tailored for Android development. Android Studio provides a suite of tools including an intelligent code editor, debugging environment, and flexible build system.

We chose Kotlin as our programming language for its modern features and seamless compatibility with Java. Kotlin’s ease of use and similarity to Java, a language we are well-acquainted with, made it an ideal choice for development.

XML (Extensible Markup Language) was used for defining user interface layouts and managing resources. XML’s standardized approach ensures compatibility and integration with other tools and systems, making it a foundational technology for building effective and maintainable Android applications.

## 2.6 Dataset

### 2.6.1 YOLO Dataset

**1. Selection of Waste Types:** We chose to focus on several waste types including paper, glass, plastic, and organic materials because these are the most commonly found types of litter in public spaces and have significant environmental impacts:

* **Paper:** Though biodegradable, paper waste contributes to deforestation and landfill overflow when not properly recycled.
* **Glass:** Non-biodegradable and potentially hazardous, glass waste can cause physical harm and environmental contamination.
* **Plastic:** Known for its long decomposition time and harmful effects on wildlife, plastic is a major environmental pollutant.
* **Metal:** Though recyclable, metal waste like cans can take hundreds of years to decompose in landfills. Improper disposal contributes to environmental degradation, soil, and water pollution, and misses the opportunity to save energy through recycling.
* **Organic Waste:** While organic materials like food peels decompose, they can attract pests and release methane, a potent greenhouse gas.

**2. Selected Classes:** The dataset includes 9 specific classes: Can, Glass Bottle, Paper Cup, Peel, Plastic Bag, Plastic Bottle, Plastic Cup, Snack Wrapper, and Tissue. These classes represent a broad spectrum of common trash types, each with distinct environmental implications and disposal needs.

**3. Dataset Composition:** The dataset comprises over 5,500 original images, capturing various instances of each class in different conditions and environments. This diverse collection helps improve the model’s ability to generalize and accurately detect trash in real-world settings.

**4. Points Allocation:** Points for each class were determined based on the environmental impact and difficulty of recycling or disposal:

**Higher Points (Incentivize Collection):**

* **Plastic Bottle (10 points)**
* **Plastic Bag (15 points)**
* **Glass Bottle (12 points)**
* **Can (10 points)**

**Medium Points:**

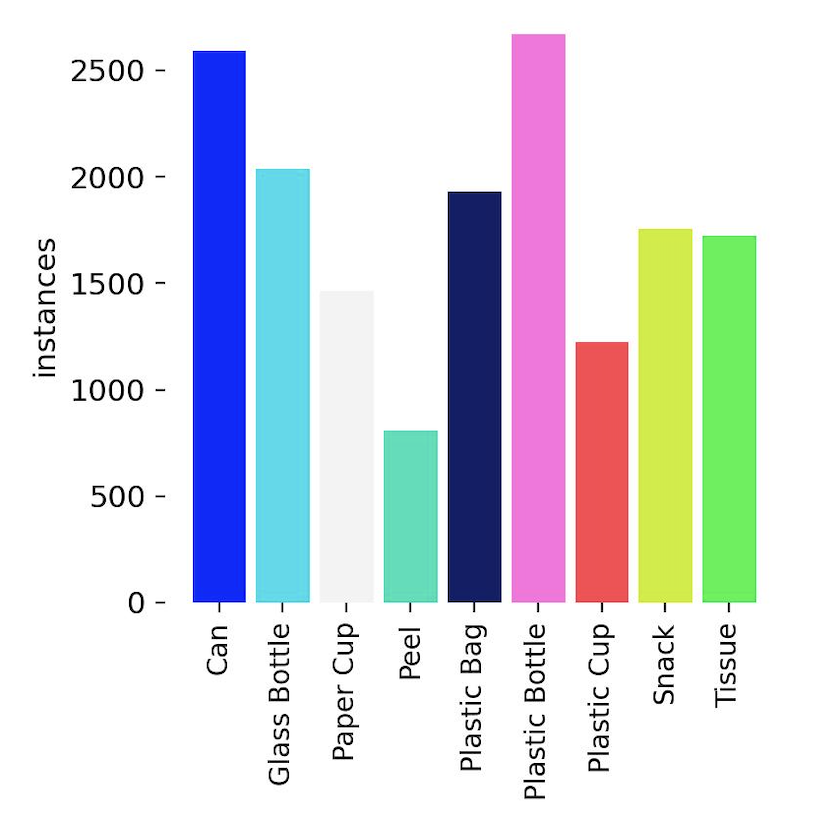
* **Plastic Cup (8 points)**
* **Snack Wrapper (7 points)**

**Lower Points (Easier to Collect/Lower Impact):**

* **Paper Cup (5 points)**
* **Tissue (2 points)**
* **Peel (3 points)**

This point allocation strategy encourages users to prioritize collecting items with higher environmental impacts, such as plastics and metals, while still rewarding the collection of less harmful items. This approach helps maximize the positive environmental impact of the app by guiding user behavior toward the most beneficial actions.

**5. Dataset Splitting and Distribution:** The dataset was split into 80% training, 10% validation, and 10% testing. This split ensures that the model has a sufficient amount of data to learn from (training), a set to validate against to tune parameters (validation), and a separate set to evaluate final performance (testing). The 80/10/10 split is a standard practice that balances the need for robust training data with reliable validation and testing sets.



*Figure 6: Number of instances for each label*

**6. Importance of Augmentation:** Augmentation is crucial for enhancing the dataset and improving the model’s robustness. By artificially expanding the dataset through augmentation, we address challenges such as class imbalance, variability in object presentation, and overfitting. We used six types of augmentation:

* **Flip (Horizontal, Vertical):** Helps the model recognize items from different orientations.
* **Crop:** Adjusts image size and framing, simulating different distances and perspectives.
* **Rotation:** Provides robustness against rotated objects in real-world scenarios.
* **Saturation:** Adjusts color intensity to handle various lighting conditions.
* **Blur:** Mimics out-of-focus conditions, improving detection in less ideal settings.
* **Noise:** Adds random variations, enhancing the model's resilience to visual imperfections.

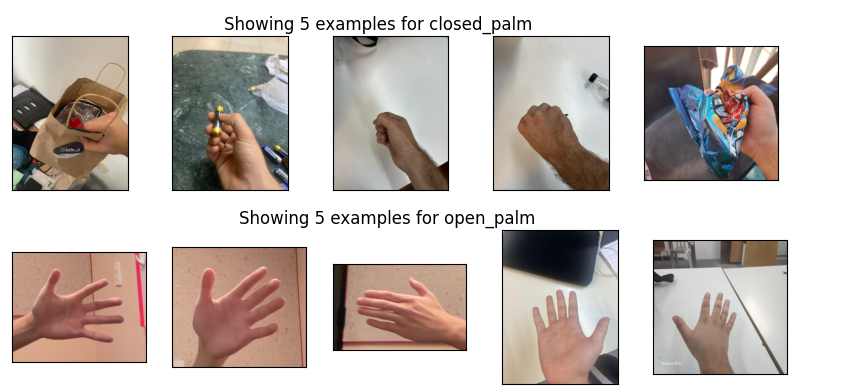
**7. Augmented Dataset:** After applying these augmentations, the dataset expanded to over 15,000 images, providing a richer and more diverse training set. This augmentation ensures that the model is exposed to a wider array of visual scenarios, improving its accuracy and reliability in real-world applications.

### 2.6.2 MediaPipe Dataset

To improve hand-tracking accuracy, we trained MediaPipe using a custom dataset specifically tailored for hand gesture detection. The dataset consisted of:

* **550 images of open hands**
* **1000 images of closed hands**

This dataset allowed us to fine-tune the MediaPipe's hand-tracking model, resulting in enhanced detection of both open and closed hand gestures.



*Figure 7: Images from the data set for MediaPipe*

## 2.7 Results

### 2.7.1 Results of Training YOLO

**Confusion Matrix:** The confusion matrix provides an overview of the model's performance in classifying the different classes of trash items. Here's an explanation of the results:

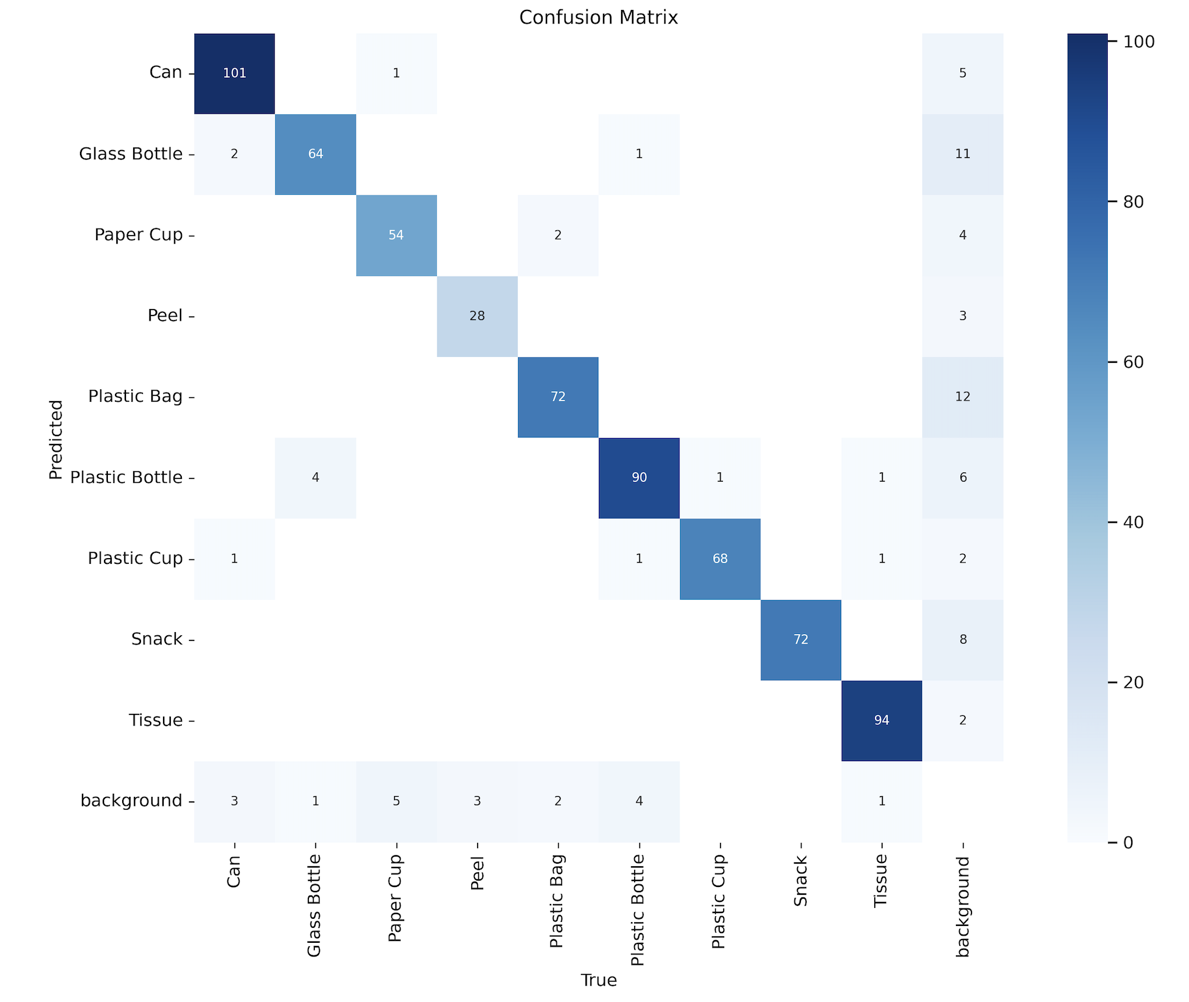
Diagonal Elements: The diagonal values represent the correct predictions made by the model for each class. For example, the model correctly predicted 101 instances of "Can" and 64 instances of "Glass Bottle." Higher values on the diagonal indicate better performance for those specific classes.

Off-Diagonal Elements: These values indicate misclassifications where the model predicted one class as another. For instance:

* 2 instances of "Glass Bottle" were misclassified as "Can."
* 4 instances of "Plastic Bottle" were mistaken for "Glass Bottle."

Overall Assessment:

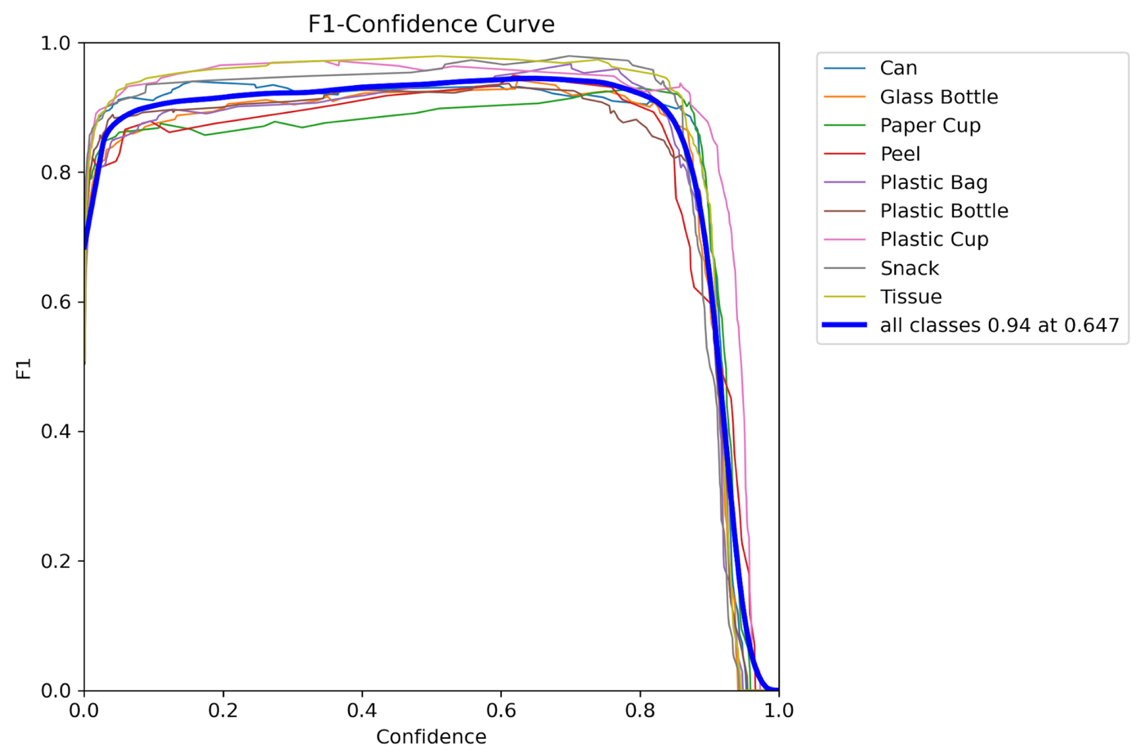
The model demonstrates good overall performance, especially for the major classes with a high number of correct predictions.

.

*Figure 8: confusion matrix for the trash items*

**F1-Confidence Curve:** The F1-Confidence Curve provides a measure of the model's precision and recall at various confidence levels, effectively illustrating the trade-off between these metrics as the confidence threshold changes. Here's an analysis of the F1-Confidence Curve:

High F1 Score Across Confidence Levels: The overall F1 score for all classes peaks around 0.94 at a confidence level of 0.647, indicating that the model maintains a good balance between precision and recall across most confidence thresholds. This is a strong result, suggesting the model is well-calibrated and performs effectively at distinguishing between classes.

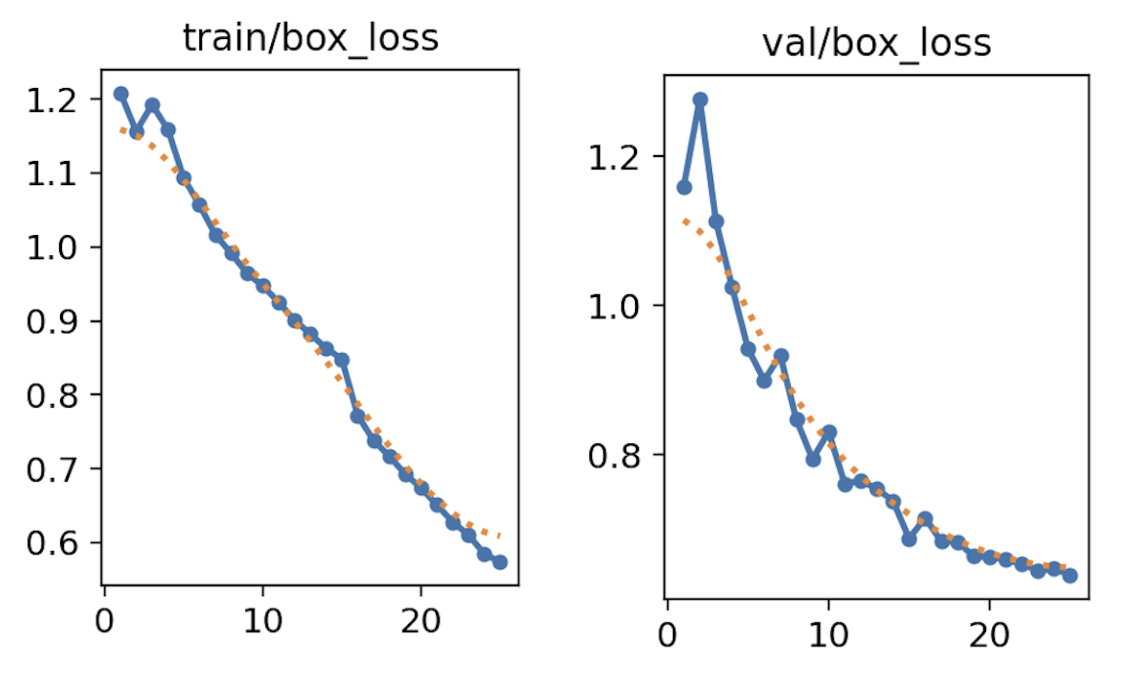


*Figure 9: confidence curve*

**Box Loss**

The following graphs represent the loss associated with bounding box predictions for both the training and validation datasets. They measure how well the model is predicting the correct box surrounding each object and its location.

Both losses show a consistent downward trend, indicating that the model is improving its ability to predict object locations accurately.

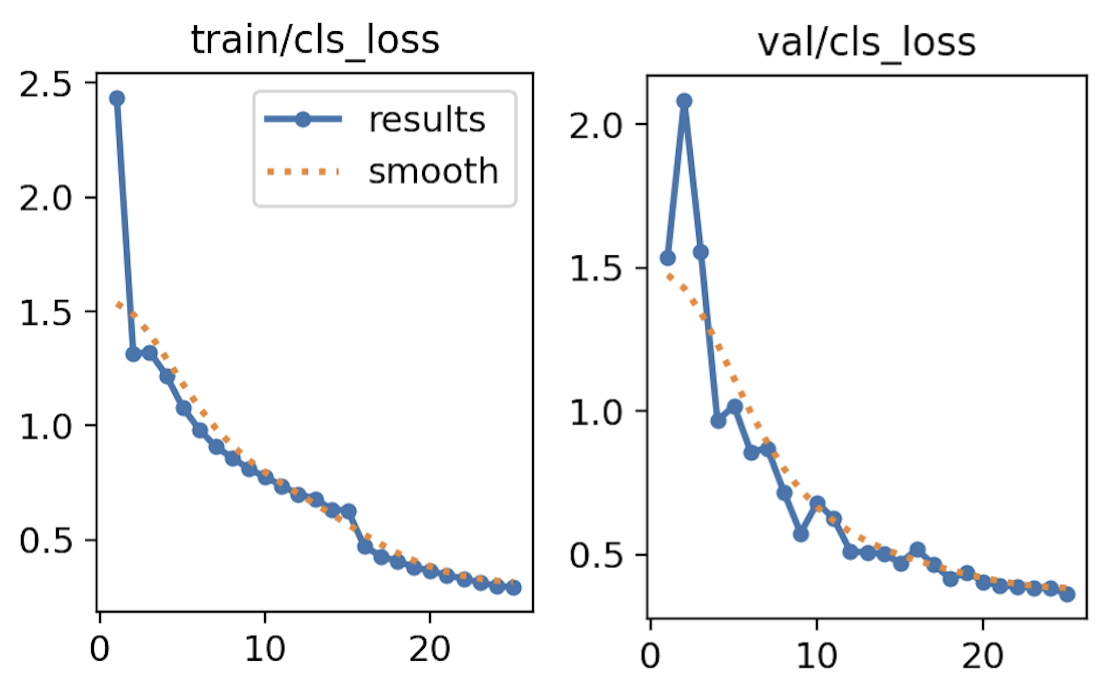


*Figure 10: train and validation box loss*

**Classification Loss**

These graphs track the classification loss during training and validation, which measures how well the model is predicting the correct class for each object.

Both training and validation classification losses decrease steadily over epochs, demonstrating that the model's classification performance is improving consistently.



*Figure 11: train and validation class loss*

Overall Model Performance:

**Good Performance**: The consistent decrease in both box and classification losses for training and validation suggests that the model is learning effectively.

**No Overfitting**: The close alignment between training and validation loss curves, without a widening gap, indicates that there is no significant overfitting. Both training and validation losses decrease at a similar rate, showing that the model maintains good generalization to unseen data.



*Figure 12: examples of YOLO with our data set*

**Optimal training epochs:**

Training the model for 25 epochs has proven to be sufficient, as both training and validation losses have reached a steady state, indicating that the model has learned effectively from the data without overfitting. The validation losses are closely aligned with the training losses, suggesting a good generalization to unseen data.

### 2.7.2 Results of Training MediaPipe



*Figure 13: model Loss and validation per epoch*

**Good Performance**: The consistent decrease in loss for training and validation suggests that the model is learning effectively.

**No Overfitting**: The close alignment between training and validation loss curves, without a widening gap, indicates that there is no significant overfitting. Both training and validation losses decrease at a similar rate, showing that the model maintains good generalization to unseen data.

## 2.8 Testing

|  |  |  |  |
| --- | --- | --- | --- |
| NUM**​** | Test**​** | Expected Result**​** | Achieved Result**​** |
| 1​ | Test the accuracy of trash item recognition ​  ​ | The system succeeds with 95% accuracy​  ​ | The system succeeds with 97% accuracy​ |
| 2​ | Test hand detection with different skin tones and hand sizes ​ | The system is not biased towards specific hand’s data in 85% of the cases​  ​ | The system is not biased towards specific hand’s data in 87% of the cases​ |
| 3​ | Test the accuracy of the counting mechanism in counting the earned points​ | The system accurately counts the earned points​  ​ | The system accurately counts the earned points​ |
| 4 | Test the storing of the earned points in the DB​ | The system successfully saves the points in DB​ | The system successfully saves the points in DB​ |
| 5 | Test the full process of collecting the trash and getting the points​ | The system succeeds in the integration part with 85% accuracy (worst case)​ | The system succeeds in the integration part with 87% accuracy ​ (worst case)​ |

## 2.9 The process of crafting software

1. We customized the YOLOv8 code to detect various types of trash items using custom dataset. YOLO is responsible for identifying objects in the camera feed and classifying them as different types of trash.
2. MediaPipe was trained to detect hand gestures, specifically the actions of opening and closing the hand, to track the user’s hand movement when grabbing trash items. We developed a custom dataset of hand poses (1000 closed-hand images and 600 open-hand images) to improve detection accuracy.
3. YOLO and MediaPipe were integrated into a single codebase to work together. MediaPipe detects when a user’s hand is in a grabbing position, and YOLO simultaneously identifies the object being picked up by the hand. This combined process provides real-time feedback to users by recognizing the object and making a beep sound when an item is successfully picked up.
4. We built several UI components for the app, including:
   1. **Login and Sign-up Pages**: Allow users to create an account and log in with their credentials.
   2. **Scores Page**: Displays the total points a user has earned based on the number and types of trash items collected.
   3. **Profile Page**: Shows the user's personal information, collected points, and allows for editing profile details such as name, hand preference, and other settings.
   4. **Help Page**: Provides users with information about how to use the app and the trash collection process.
5. **Real-Time Interaction**:
   1. The app captures video in real time from the phone’s camera and processes the feed using YOLO and MediaPipe. When the user picks up an item, the app recognizes it and updates the score, with different trash types (paper, plastic, glass, etc.) earning different points.
   2. A beep sound serves as feedback to indicate that an item has been successfully collected, enhancing the interactivity and user experience.
6. **Error Management and Feedback**:
   1. Throughout the development process, we encountered various bugs, such as misidentifying hand gestures or failing to classify objects. We addressed these issues by refining our datasets and code logic.
   2. We also gathered feedback from users to make the app more user-friendly and efficient.

# 3. Challenges

## 3.1 Description of Challenges Faced

During the implementation of the app, we encountered several challenges that required innovative solutions across different technical areas:

1. Real-Time Object Detection and Classification

* **Challenge:** Implementing real-time trash detection and classification on Android devices posed challenges due to the need for high accuracy and low latency. The app needed to process video frames quickly on mobile hardware, which often has limited computational power.
* **Solution:** We integrated the YOLOv8 algorithm, optimized for speed and accuracy, and trained it on a customized dataset of 9 specific trash classes. By focusing the model on only the relevant classes, we minimized computational overhead, allowing the app to run efficiently on Android devices.

2. Customized Dataset for Specific Trash Classes

* **Challenge:** The initial datasets (TACO and COCO) included numerous irrelevant object categories, which could reduce detection accuracy and increase processing time on Android devices.
* **Solution:** We created a customized dataset containing only 9 relevant trash classes by filtering out unnecessary categories. This tailored dataset was used to train the YOLOv8 model, improving detection speed and accuracy.

3. Hand Tracking and Gesture Recognition

* **Challenge:** Accurate hand tracking and gesture recognition were crucial for the detection of the users picking up trash. Ensuring reliability across diverse hand shapes, sizes, and skin tones was challenging, especially under varying lighting conditions typical of outdoor environments.
* **Solution:** We used MediaPipe for real-time hand tracking, customizing its model with a diverse dataset to handle different hand characteristics. The solution included optimizing the model to work effectively even in less-than-ideal lighting and background conditions on mobile devices.

4. Imbalance and variability in the dataset

* **Challenge:** We noticed that the dataset had a significant class imbalance and variability in object presentation, which could lead to poor model performance and overfitting.
* **Solution:** Applying data augmentation techniques, which artificially expand the dataset and make it more robust. Specifically, we used six types of augmentation: flipping, cropping, rotating, adjusting saturation, blurring, and adding noise. These techniques helped the model become more resilient to different real-world scenarios and improve overall accuracy.

5. User Interface Responsiveness

* **Challenge:** Developing a responsive and user-friendly interface that provided real-time feedback was critical for user engagement. The UI needed to be intuitive and quick, displaying trash types and points without delay.
* **Solution:** The frontend was developed using lightweight Android components, with careful optimization to reduce rendering times and enhance responsiveness. The UI was designed to be intuitive and accessible, making it easy for users to navigate and interact with the app.

## 3.2 Conclusions:

**Achievement of Project Goals:** The "Earth Keepers" app successfully met its primary objectives. We created a user-friendly Android application that encourages environmental cleanup through real-time trash detection and gamification. The app's core functionalities, including accurate trash detection using a customized dataset, real-time feedback, and user engagement through a points system, were fully implemented and function as intended. The project goals were achieved within the constraints of mobile device performance, ensuring the app is both effective and efficient.

3.3 Decision-Making Process**:** Our decision-making was guided by a focus on user experience and performance optimization. When faced with challenges, we prioritized solutions that balanced technical feasibility with user satisfaction. For example, the decision to create a customized dataset was driven by the need for precision in trash detection, which directly impacts the app's effectiveness.

In conclusion, by carefully addressing each challenge and making informed decisions, we were able to deliver a successful project that met all of its goals, providing a valuable tool for environmental stewardship.

## 3.4 Overall Effectiveness of the Work

The project was largely successful, achieving its objectives with the effective use of YOLOv8 for object detection and MediaPipe for hand gesture recognition. However, several areas could have been improved upon in hindsight:

* **Earlier Focus on Custom Dataset**: Initially, time was spent on pre-existing datasets that included irrelevant categories. Creating a custom dataset earlier would have saved time and improved detection accuracy.
* **Enhanced Hand Gesture Recognition**: The initial dataset for hand gestures was limited. Expanding this earlier, along with optimizing recognition under different lighting conditions, could have further improved the app’s performance.
* **Improved Real-Time Performance Testing**: Testing in various environmental conditions could have been done earlier to ensure better optimization for real-time performance.
* **Task Division and Agile Workflow**: A more agile approach with frequent feedback loops could have streamlined the process and addressed issues sooner.
* **Earlier User Feedback Integration**: Involving users earlier in the development phase would have identified usability issues sooner, improving the overall user experience.
* **Cross-Platform Framework**: In hindsight, using a **cross-platform framework** like Flutter or React Native would have allowed the app to be developed for both iOS and Android simultaneously. This would have broadened the app’s reach and ensured compatibility across a wider range of devices, ultimately making the app more versatile and accessible.

## 3.5 Meeting the Project’s Criteria: Explanation and Justification

The project met the success criteria that were set:

* **Accuracy**: Object detection accuracy surpassed the goal of 90%, with YOLOv8 achieving over 97% accuracy for trash detection. Hand gesture recognition also performed well, meeting the requirement for detecting hand grabs.
* **Efficiency**: While the app processed trash items in real-time, the identification took approximately 2-3 seconds per item instead of the intended 1 second. Although this exceeded the initial standard, the system still performed efficiently enough for real-time usage and provided users with a smooth experience.
* **User Interface**: A user-friendly interface was successfully implemented, making it easy for users to navigate the app and receive real-time feedback during trash collection.
* **User Feedback**: Positive feedback was received from users, confirming that the app was intuitive and engaging, fulfilling the final success criterion

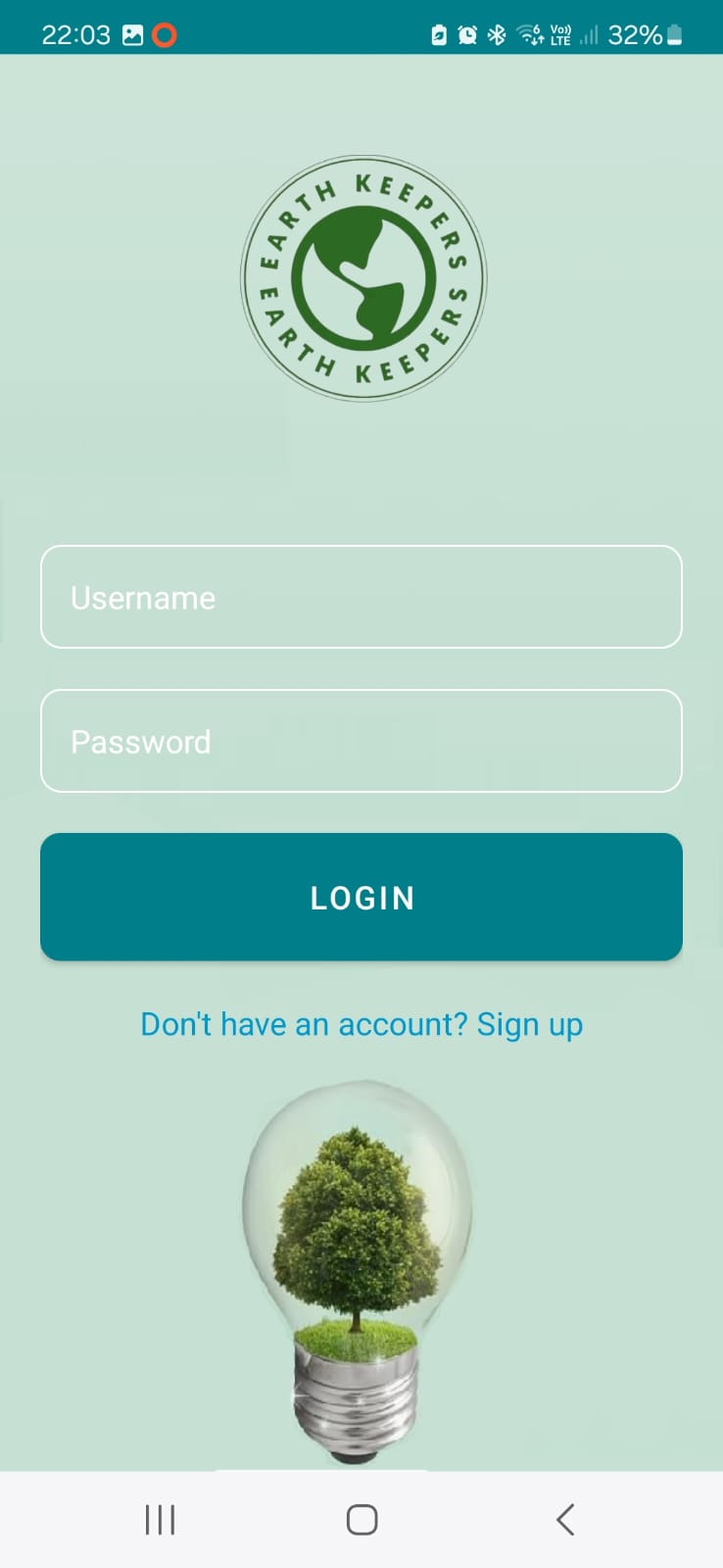
# 4. User Guide

The User Guide provides clear instructions to help users navigate the app and its features, ensuring a smooth experience. It serves as a quick reference for common tasks.

Creating a user-friendly interface for an app is crucial and essential for enhancing User Experience. Therefore, we are planning to build the following interface for our app.

**Login Page**  
This page allows users to log in or create a new account to access the app.

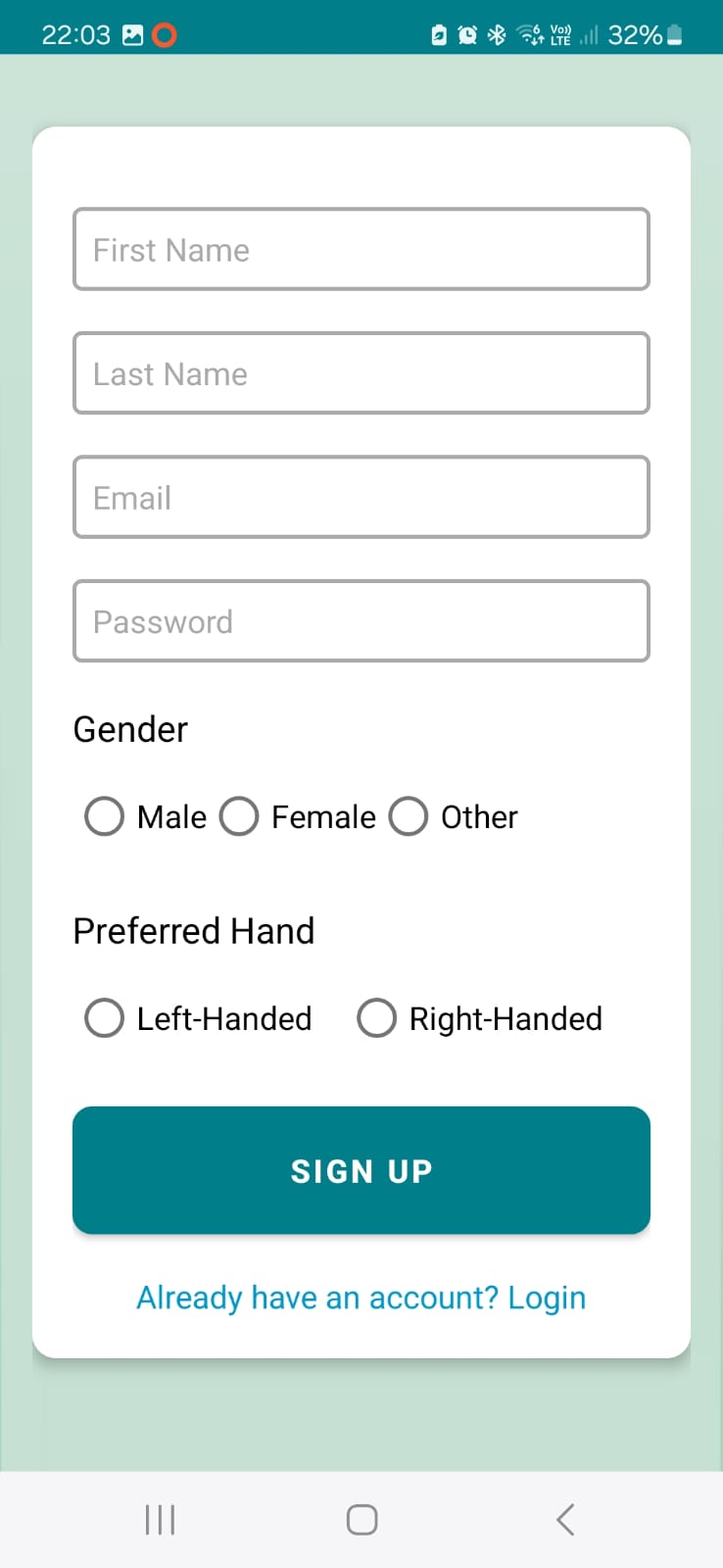
* **Login Form:** Users can enter their **Username** and **Password** to log in.
* **Sign-Up Prompt:** If users don't have an account, they can tap on the **Sign Up** link to create a new account.



*Figure 14: Login page*

**Sign-Up page:**

* **Personal Information:** Users need to enter their **First Name**, **Last Name**, **Email**, and set a **Password**.
* **Gender Selection:** Options include **Male**, **Female**, and **Other**.
* **Preferred Hand:** Users can select either **Left-Handed** or **Right-Handed** for customization during the app’s usage.

**

*Figure 15: sign-up page*

**Home Page**

The Home Page is the main hub for navigating the app. It includes several key buttons and information:

* **Profile Button:** Access this to view your profile and personal information, including your name and stats.
* **Start Button:** Tap this to begin a new session of waste detection.
* **Player’s High Scores Button:** View the top scores of other players and their collected points.
* **Help Button:** Get explanations on how the app works and how to use its features.



*Figure 16: Home page for the app*

**Pre-Session Page**

In this page, you should take one photo of all the trash items you want to collect during this session. Afterward, the app will calculate the maximum possible points you can earn and display the detected trash items along with their corresponding points.

Clicking the Start Button on the Home Page opens the Pre-Session Page, which includes:

* **Photo Capture Area**: This page allows the user to take a photo of the trash he wants to collect.
* **Retake Photo Button**: Use this button to retake the photo.
* **Start Collecting Button**: Use this button to proceed to the Real-Time Waste Detection Page and start the trash collection session.



*Figure 17:* Pre-Session Page

**Real-Time Waste Detection Page**

Clicking the “**Start Collecting” Button** on the Pre-Session Page opens the Real-Time Waste Detection Page, which includes:

* **END Button:** Use this button to end the session and save your earned points.
* **Video Capture Area:** This section displays the live video feed for detecting and collecting waste.

On this page, you begin collecting trash items, one item at a time. The app must first detect that your hand is open, and once you hold an item, it will detect that your hand is closed. After holding the item, you should wait for 2-3 seconds until you hear a sound that indicates what item you're holding and the points you've earned. If you don't hear anything (though this is unlikely), you should drop the item and try again.

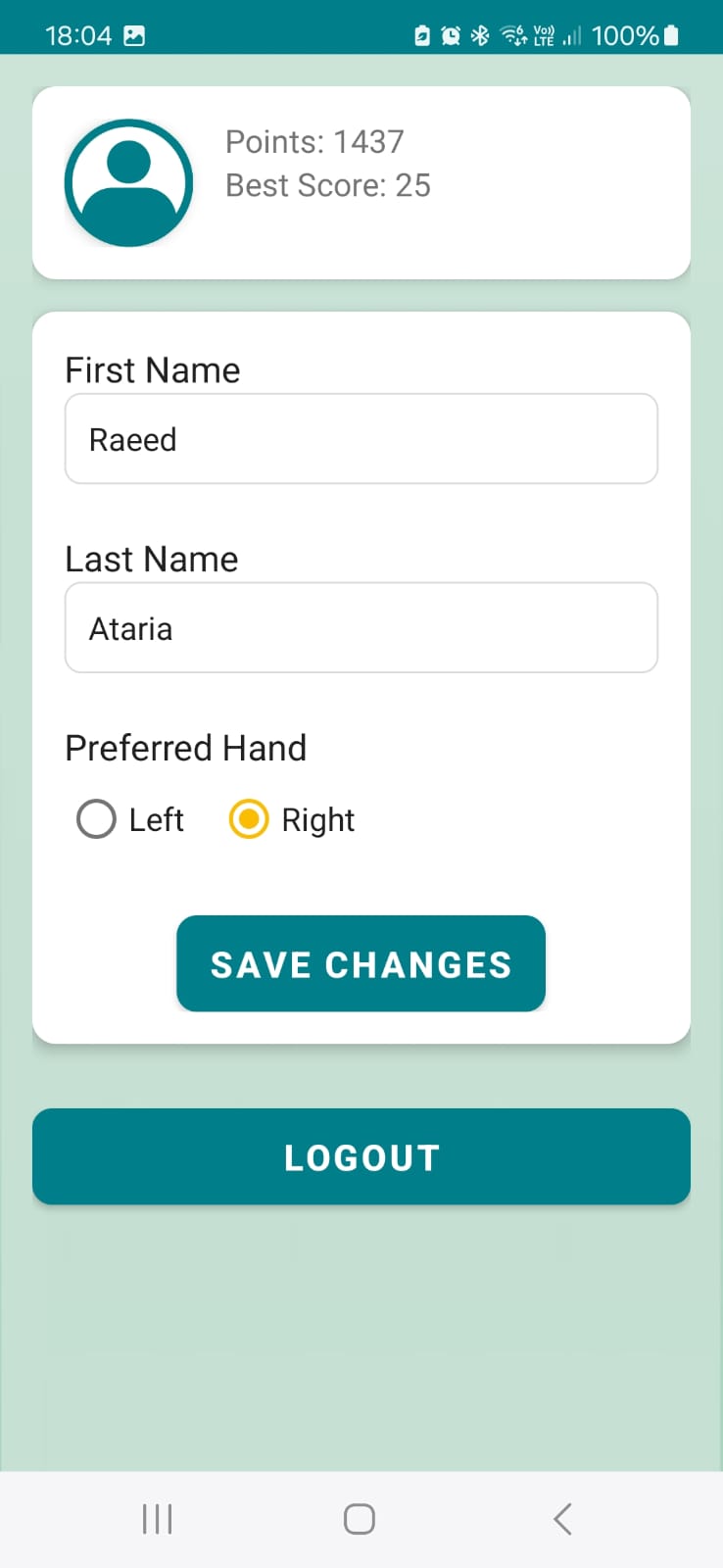


*Figure 18: Snapshot of a trash collection session in progress*

**Personal Info Page**

Access the Personal Info Page by clicking the **Profile Button** on the Home Page. This page provides:

* **Personal Information:** Displays your first and last names, along with your current collected points.
* **Total Points:** Shows the total number of points you have collected since joining the app.
* **Best Session Score:** Indicates your highest score achieved in a single session.
* **Hand Preference:** Displays your preferred hand for collecting trash. You can change this preference by clicking on the hand icon or button.
* **Logout Button:** Click this to log out of the app.

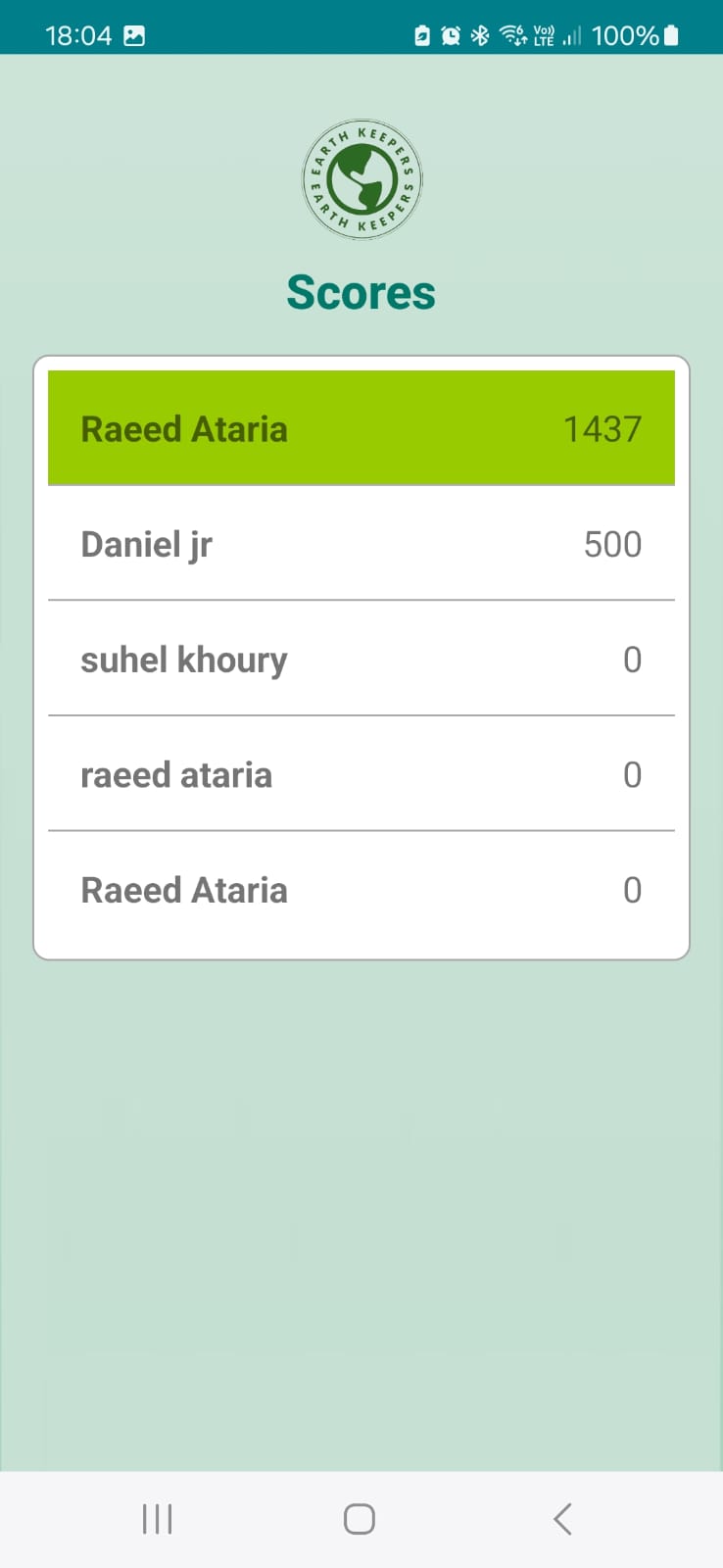


*Figure 19: Personal Info Page*

**Best Scores Page**

Access the Best Scores Page by clicking the **Player’s High Scores Button** on the Home Page. This page features:

* **Top Scores Table:** Displays a table of the top scores achieved by various players.
* **Player Rankings:** Shows the rank of each player, including yourself.
* **Highlighted User:** Your own ranking is highlighted in green.



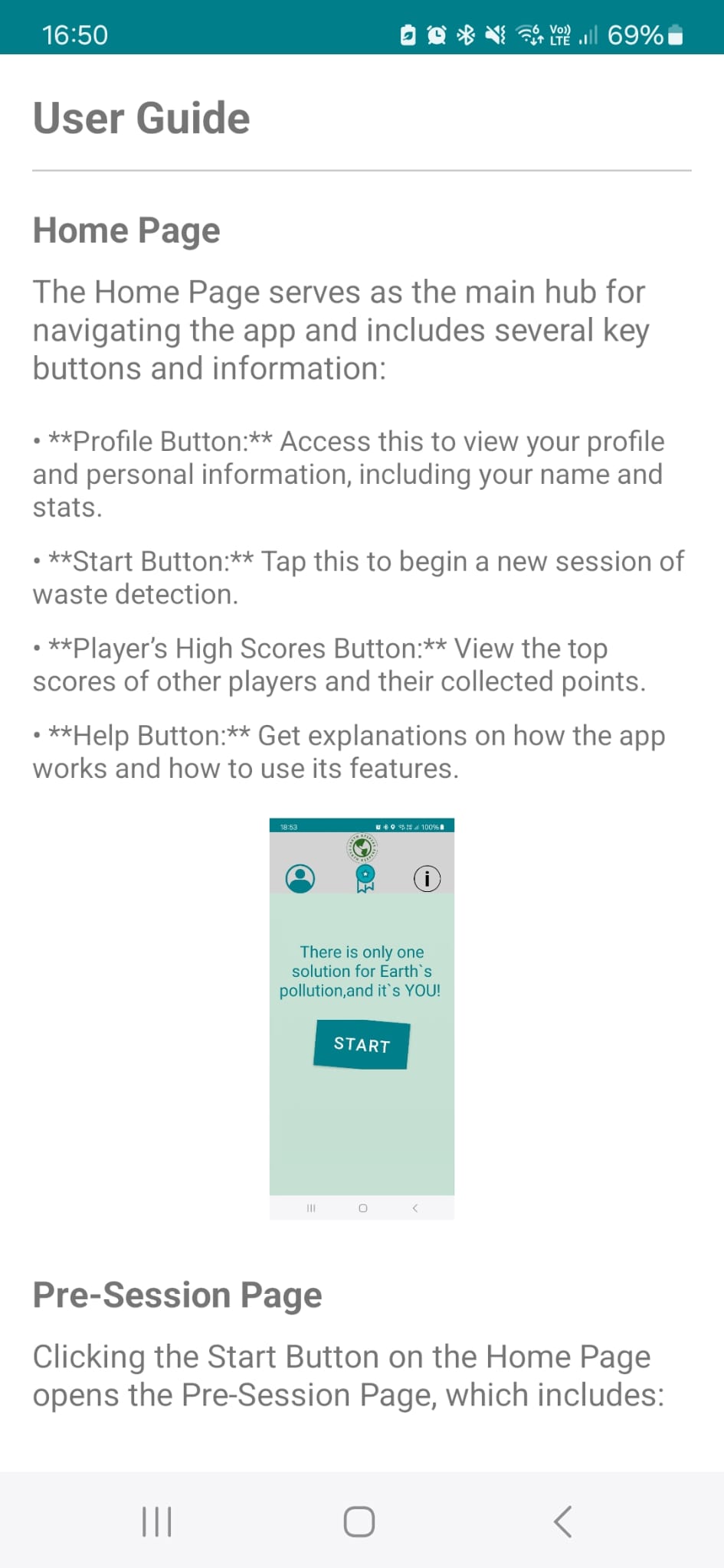
*Figure 20: Best Scores Page*

**User Help page:**



By clicking on the help icon (located on the right), you can access the user help page, which provides assistance for any feature or issue you encounter while using the app.

The following figure shows a portion of the User Guide within the app. You can access the complete User Guide directly in the app.



*Figure 21: User help Page*

# 5. Maintenance Guide

**Operating Environment:** Android Studio

* **Required Software and Hardware:**
  + **Operating System:** Windows, macOS, or Linux.
  + **Android Studio Version:** It is recommended to use the latest version.
  + **Android SDK:** Ensure the installation of the appropriate SDK for the target version of the application.
  + **Java Development Kit (JDK):** Install a compatible version of JDK (recommended JDK 11 or above).
* **Additional Requirements:**
  + **Git:** For sharing and updating source code from the repository on GitHub.
  + **Android Device:** For testing and running on a physical device; a device with a compatible version (Android 8.0 or above) is recommended.

**Installation and Operation Instructions:**

1. **Clone the Code from GitHub Repository:**
   * Open Android Studio and select **File > New > Project from Version Control**.
   * Enter the GitHub repository URL and click **Clone**.
2. **Setting Up a Physical Device for Testing (Alternatively, use Android Studio Emulator):**
   * Connect the Android device to your computer using a USB cable.
   * Enable Developer Mode on the device:
     + Go to **Settings > About phone**.
     + Tap **Build number** 7-10 times until you see a message that Developer Mode is enabled.
   * Enable USB Debugging:
     + Go to **Settings > Developer options** and enable **USB Debugging**.
     + Set **Default USB configuration** to **Charging phone only**.
3. **Running the Application:**
   * Ensure the device is connected and appears as an option for running in Android Studio.
   * Click the **Run** button and select the physical device or emulator.

# 6. References

1. Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg and Matthias Grundmann (2019). MediaPipe: A Framework for Building Perception Pipelines. Google Research.   
[[1906.08172] MediaPipe: A Framework for Building Perception Pipelines (arxiv.org)](https://arxiv.org/abs/1906.08172)

2. Chien-Yao Wang1,2, I-Hau Yeh2 , and Hong-Yuan Mark Liao (2024). YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. Institute of Information Science, Academia Sinica, Taiwan National Taipei University of Technology, Taiwan Department of Information and Computer Engineering, Chung Yuan Christian University, Taiwan. <https://arxiv.org/pdf/2402.13616>

3. Dr.Kundu, R. (2023). YOLO: Algorithm for Object Detection.

<https://www.v7labs.com/blog/yolo-object-detection>

4. Guillermo Sánchez‑Brizuela, Ana Cisnal, Eusebio de la Fuente‑López, Juan‑Carlos Fraile, Javier Pérez‑Turiel (2023). Lightweight real‑time hand segmentation leveraging MediaPipe landmark detection. Springer Link  
[Lightweight real-time hand segmentation leveraging MediaPipe landmark detection | Virtual Reality (springer.com)](https://link.springer.com/article/10.1007/s10055-023-00858-0)