The Egyptian E-Learning University

Faculty of computer and information technology

Currency detection app

Graduation Project Documentation

Team Members

- Mohamed Hosny Hassanien Dessouky (2001176)
- Ahmed Mohamed Salah Gabr (2000029)
- Hassan Refaet Abd ElRahem Hassan (2000780)
- Mohamed Hussein Ahmed Abd Elkareem (2000565)
- Shady Shawky Shaker Basel (2001311)
- Abdallah Mohamoud Ibrahim Abd Elkereem (2000819)
- Ahmed Sayed Moahmed Abd elaziz (2000697)

Supervised by:
Dr.Khaled Fathy Hussien
Eng. Nourhan Khashaba

EELU
الجامعة المصرية للتعلم الإلكتروني الأهلية
THE EGYPTIAN E-LEARNING UNIVERSITY

Contents

Acknowledgement

First of all, we would like to thank and praise God for his countless blessings

We extend our sincere thanks to Dr. Khaled Fathi for his help and effort and the advice he gave us that helped us provide the best model for project.

Let us not forget the indescribable efforts of the assistant lecturer Engineer Nourhan Khasba, and we thank her for the effort she made throughout the

Project period.

Let us not forget the efforts of the college of doctors and engineers Provide the best for us

Abstract

This project aims to develop an application that helps visually impaired and blind individuals easily and accurately identify Egyptian currency using advanced technologies such as MediaPipe and Kotlin. This application contributes to improving the quality of life and increasing the independence of user group.

Project Objectives

1. **Develop a reliable and user-friendly application** that can identify Egyptian currency.
2. **Utilize modern technologies** in image processing and video analysis to enhance the accuracy of currency recognition.
3. **Improve the independence of visually impaired and blind individuals** by providing a mobile-based tool.

Tools and Technologies Used

1. **MediaPipe**: An open-source framework for building machine learning solutions in image and video processing. It was used for tracking and detecting hand and eye features, aiding in the identification and differentiation of currency.
2. **Kotlin**: A modern programming language designed for Android app development. We chose Kotlin for its ease of integration with Android libraries and its high performance.
3. **OpenCV**: An open-source library used for image processing, utilized to enhance the quality of input images and analyze them before applying MediaPipe models.

Project Development Phases

1. **Data Collection**: Gathering images of various Egyptian currencies under different lighting conditions and backgrounds.
2. **Data Processing**: Using OpenCV to enhance the images and ensure the clarity of currency details.
3. **Model Development**: Employing MediaPipe to train a model to recognize currency based on its visual features.
4. **App Development**: Programming the application using Kotlin and integrating the MediaPipe model.
5. **Testing and Evaluation**: Conducting extensive tests to ensure the accuracy and speed of currency recognition, and obtaining feedback from target users to improve the application.

Chapter 1 : Introduction

1.1 Introduction :

Blind and visually impaired people face many problems in their daily lives, and one of these problems is determining the currencies when paying for something or when taking the rest, so the primary aim is to design and implement an innovative system that caters to the needs of visually impaired individuals by facilitating the identification of various denominations of Egyptian currencys. With advancements in technology and a growing emphasis on inclusivity, the project endeavors to bridge the gap between accessibility and currency recognition for the visually impaired community.

The significance of this project lies in its potential to empower individuals with visual impairments, enabling them to navigate financial transactions independently and with confidence. By creating a solution that offers an accessible and intuitive method for distinguishing Egyptian currencys, this project aligns with broader efforts aimed at fostering inclusivity and equal participation in economic activities

The successful implementation of this project promises to serve as a stepping stone towards a more accessible and inclusive society, where individuals of all abilities can engage confidently in everyday activities, including financial transactions involving Egyptian currency.

1.2. Motivation

The primary motivation for this project is to assist visually impaired individuals and enhance the accuracy of financial transactions involving cash. Identifying banknotes can be particularly challenging for those with visual impairments, who often rely on touch or size differences to distinguish between notes. This method is not always reliable, especially with similar-sized notes or notes that are worn out.

For visually impaired individuals, handling cash independently is crucial for day-to-day transactions and maintaining personal autonomy. By developing a mobile app that can accurately detect and identify different denominations of Egyptian currency, we aim to provide a tool that empowers these individuals, enabling them to manage their finances more effectively and with greater confidence.

Additionally, accurate currency detection plays a significant role in reducing fraud. With the proliferation of smartphones, leveraging mobile technology for currency detection presents an accessible and efficient solution. Most people carry smartphones, which are equipped with advanced cameras and processing capabilities, making them ideal platforms for deploying machine learning models for real-time image recognition. By developing a mobile

app for currency detection, we can harness these capabilities to provide a user-friendly tool that offers quick and accurate identification of banknotes.

This mobile app can have significant social benefits:

Empowering Visually Impaired Users: The app can serve as an invaluable tool for visually impaired individuals, helping them identify currency notes independently and securely.

Enhancing Transaction Accuracy: For all users, the app can ensure that they are receiving the correct amount of money in transactions, thereby reducing errors and misunderstandings.

Facilitating Inclusion: Overall, the app promotes financial inclusion by making currency handling more accessible to everyone, regardless of their visual abilities.

1.3 Problem Background

This section provides an in-depth exploration of the difficulties encountered by visually impaired individuals in accurately distinguishing and identifying various denominations of Egyptian currencys. This section aims to shed light on the specific challenges that form the basis for the project's focus.

Challenges Faced by Visually Impaired Individuals:
Lack of Accessible Identification Methods: Currently, there is a scarcity of reliable and accessible methods or tools tailored to assist visually impaired individuals in identifying Egyptian currencys. Traditional methods often involve reliance on assistance from others or memorization, which can be cumbersome and impractical in daily transactions.

Complexity in Currency Differentiation:

Egyptian currencys, with their varied sizes, edges, and embossed features, pose a significant challenge for blind individuals in distinguishing between different denominations. The lack of tactile cues or accessible features exacerbates the difficulty of accurate identification.

Impact on Independence and Financial Autonomy:

The inability to discern between currency values independently restricts the autonomy of visually impaired individuals in handling their finances. This reliance on others for currency identification undermines their ability to engage fully in economic activities.

1.4 The Problem Statement

The problem we aim to address is the need for a reliable method to detect and classify Egyptian currency notes using a mobile application. The complexity of this task stems from several factors that need to be addressed to ensure the solution is both effective and practical for everyday use.

Recognizing Different Denominations:

The app must accurately identify and distinguish between the eight types of Egyptian banknotes in circulation: 5 pounds, 10 new pounds, 10 old pounds, 20 new pounds, 20 old pounds, 50 pounds, 100 pounds, and 200 pounds. Each denomination has unique features such as color, size, and design elements, which the app must recognize.

Robustness to Environmental Variations:

In real-world scenarios, users will capture images of banknotes under various conditions. The app must perform reliably despite changes in lighting (e.g., bright sunlight, indoor lighting, low light), angles (e.g., tilted or rotated notes), and backgrounds (e.g., different surfaces or textures behind the note). These variations can affect image quality and clarity, challenging the accuracy of detection.

Physical Condition of Banknotes:

Banknotes often suffer wear and tear through regular use. They might be crumpled, folded, torn, or dirty. The app needs to be resilient to such imperfections, maintaining high accuracy even when the physical condition of the notes is less than ideal.

Real-Time Processing:

For practical usability, the app should process images and provide results in real-time or near real-time. Users need swift feedback to make quick decisions during transactions.

User-Friendly Interface:

The application should be easy to use, especially for visually impaired individuals. This includes intuitive navigation, clear instructions, and audible feedback to guide users through the detection process.

Scalability and Maintenance:

The solution should be scalable to accommodate future updates, such as the introduction of new banknotes or changes in existing ones. It should also be maintainable, allowing for regular updates and improvements based on user feedback and advancements in technology.

1.5 Problem Solutions

To solve the problem of detecting and classifying Egyptian currency notes using a mobile application, we implemented a multifaceted approach leveraging advanced machine learning techniques, robust image processing algorithms, and practical mobile application development. Here is a detailed breakdown of our solution:

Convolutional Neural Networks (CNNs) for Image Recognition:

Model Selection: We chose Convolutional Neural Networks (CNNs) due to their proven effectiveness in image recognition tasks. CNNs can automatically learn and extract features from images, making them ideal for distinguishing between different banknotes.

MobileNet Algorithm: Initially, we implemented the MobileNet algorithm, known for its efficiency on mobile devices. MobileNet is a lightweight model designed for resource-constrained environments like smartphones, providing a good balance between accuracy and computational efficiency.

EfficientLite-2 Algorithm: To further enhance performance and accurecy, we adopted the EfficientLite-2 algorithm. This model combines the efficiency needed for mobile deployment with improved accuracy and robustness, making it well-suited for real-world applications.

Data Collection and Preparation:

Diverse Dataset: We collected a comprehensive dataset of images covering all eight types of Egyptian banknotes. The dataset includes images captured under various conditions to ensure the model can generalize well.

Preprocessing: Images were preprocessed to standardize their size and format. This step ensures consistency and helps the model learn more effectively.

Data Augmentation: Techniques such as rotation, scaling, brightness adjustment, and adding noise were applied to the dataset. Augmentation increases the variability of the training data, helping the model become more robust to different real-world conditions.

Model Training and Validation:

Training Process: The CNN models were trained on the prepared dataset using Google Colab, which provides the computational resources needed for intensive training processes.

Validation: We used a separate validation set to monitor the model's performance and prevent overfitting. Regular validation ensures that the model maintains high accuracy on unseen data.

Evaluation of Model :

Performance Metrics: Models were evaluated using metrics like accuracy and precision to ensure a comprehensive understanding of their performance. These metrics help identify strengths and weaknesses in the model's ability to classify banknotes accurately.

Mobile Application Development:

Integration with Kotlin : The trained models were integrated into a mobile application developed using Kotlin in Android Studio. Kotlin offers modern language features and seamless integration with Android's ecosystem, ensuring a smooth development process.

MediaPip Integration: We used MediaPip, an open-source framework for building multimodal machine learning pipelines, to facilitate the integration of machine learning models into the mobile app. MediaPip provides tools for real-time perception and facilitates efficient processing of visual data.

User Experience Design:

Intuitive Interface: The application was designed with a user-friendly interface, making it accessible for all users, including those with visual impairments. Features such as buttons, clear instructions, and voice feedback enhance usability.

Real-Time Processing: Ensuring the app provides real-time or near-real-time feedback was crucial for practical use. Optimizations were made to ensure the app processes images quickly and efficiently, providing immediate results to the user.

By combining these strategies, we developed a robust and efficient solution for detecting and classifying Egyptian currency notes using a mobile application. This approach not only addresses the technical challenges but also ensures the application is practical and beneficial for everyday use, especially for visually impaired individuals.

1.6 Project Phases

A. Project Planning and Goal Definition

Objective: Establish a clear roadmap for the project by defining goals, scope, deliverables, and timelines.

Define Project Scope: Identify the primary goals, such as creating an app to detect and classify Egyptian currency notes, and outline secondary objectives, such as enhancing accessibility for visually impaired users.

Stakeholder Identification: Determine the key stakeholders, including developers, end-users (e.g., visually impaired individuals).

Requirements Gathering: Collect and document the functional and non-functional requirements, including technical specifications, user interface preferences, and performance criteria.

Resource Allocation: Plan the necessary resources, including team members, hardware and software. Ensure the availability of tools like Android Studio, Google Colab, and MediaPip.

Timeline and Milestones: Develop a detailed project timeline with specific milestones for each phase. This includes deadlines for data collection, model development, training, testing, deployment, and maintenance.

B. Data Collection and Preparation

Objective: Assemble and preprocess a comprehensive dataset to train and validate the machine learning models.

Dataset Acquisition: Gather a diverse set of images representing all eight types of Egyptian banknotes. Ensure that images are captured under various conditions to simulate real-world scenarios.

Image Preprocessing: Standardize the images by resizing, normalizing, and formatting them to ensure consistency. This step helps improve the efficiency and accuracy of the training process.

Data Augmentation: Apply techniques such as rotation, scaling, cropping, brightness adjustment, and noise addition to artificially expand the dataset. This enhances the model's robustness and ability to generalize to new images.

Labeling: Ensure all images are accurately labeled with the correct denomination. This is critical for supervised learning algorithms like CNNs.

C. Model Selection and Development

Objective: Choose the most suitable machine learning models and develop initial versions to solve the currency detection problem.

Algorithm Evaluation: Assess various convolutional neural network (CNN) architectures to determine the best fit for the task. we Considered factors such as accuracy, speed, and compatibility with mobile devices.

Initial Model Development: Implement and code the selected models using platforms like Google Colab. Start with basic versions to establish a baseline performance.

D. Training and Validation

Objective: Train the models on the collected dataset and validate their performance to ensure accuracy and reliability.

Training Process: Use the preprocessed and augmented dataset to train the CNN models.

Validation: Split the dataset into training and validation sets. Use the validation set to monitor the model's performance during training, preventing overfitting and ensuring the model generalizes well to unseen data.

Iterative Training: Continuously refine the models based on validation feedback.

E. Evaluation and Model Tuning

Objective: Evaluate the trained models using comprehensive metrics and fine-tune them for optimal performance.

Performance Metrics: Assess the models using metrics such as accuracy and precision. These metrics provide insights into the model's ability to correctly classify the banknotes.

Model Optimization: Implement techniques such as model pruning or quantization to reduce the model's size and improve its efficiency, making it more suitable for deployment on mobile devices.

F. Deployment and Integration

Objective: Integrate the trained models into a mobile application and ensure the app is functional and user-friendly.

Mobile App Development: Use Kotlin and Android Studio to develop the mobile application. Design the user interface with accessibility in mind, providing clear instructions and audible feedback for visually impaired users.

Model Integration: Embed the trained models into the mobile app using frameworks like MediaPipe. Ensure the models interact seamlessly with the app's interface and functionalities.

Testing: Conduct extensive testing on various devices to verify the app's performance and usability under different conditions. Address any bugs or issues that arise during testing.

1.7 Project Schedule :

Phase 1: Planning and Research (Oct to Nov)

1.1 Preamble, Problem Background, and Statement

1.2 Significance of the Project and Objectives

1.3 Project Scope, Requirements, and Limitations

Phase 2: Development (Dec to Feb)

2.1 Data Collection and Preprocessing

2.2 Machine Learning Model Development and Training

2.3 Mobile App Interface Development

Phase 3: Testing and Refinement (Mar to May)

3.1 Integration and Testing of Image Recognition

3.2 Real-time Processing Implementation and Testing

3.3 Security Features

Phase 4: Documentation and Finalization (Jun)

4.1 Project Report Compilation

4.2 Final Testing and Quality Assurance

Overall Project Schedule:

Start Date: [Oct]

End Date: [Jun]

Total Duration: [9 months]

CHAPTER TWO : RELATED SYSTEM

2.1 Deep learning:

Deep learning plays a crucial role in the development of an app for currency detection aimed at helping blind people. Here's a summary of its importance in this project:

- **Accuracy and Precision**

  Deep learning models, especially Convolutional Neural Networks (CNNs), are highly effective at recognizing complex patterns in images. For currency detection, this means the app can accurately identify different denominations, even in varied lighting conditions, orientations, and wear states of the currency.

- **Robustness**

  Deep learning provides robustness against variations in the input data. It can handle different textures, colors, and sizes of currency notes, making it reliable in real-world scenarios where the appearance of currency may vary.

- Real-Time Processing

  Using frameworks like MediaPipe, which is optimized for real-time applications, deep learning models can process images quickly. This allows the app to provide immediate feedback to users, which is crucial for assistive technologies intended for the visually impaired.

- Automated Feature Extraction

  Deep learning automates the feature extraction process, eliminating the need for manual intervention. The models learn and extract relevant features from currency images during the training phase, ensuring that the system can adapt to new currencies or changes in existing ones with minimal adjustments.

- Adaptability and Learning

  Deep learning models can be continuously improved with additional training data. This adaptability is important for maintaining the accuracy and reliability of the app as new currencies are introduced or existing ones are updated.

- Enhanced User Experience

  By leveraging deep learning, the app can offer a seamless and intuitive user experience. Users do not need to perform complex actions; they can simply show the currency note to their device's camera, and the app will detect and announce the denomination.

- Integration with Assistive Technologies

  Deep learning models can be easily integrated with other assistive technologies, such as voice synthesis and haptic feedback, providing a comprehensive solution for blind users. This integration ensures that the detected currency information is conveyed effectively, enhancing the user's independence.

- Scalability

  Once trained, deep learning models can be deployed across various platforms (smartphones, tablets, etc.), making the app accessible to a wider audience. This scalability is essential for reaching and assisting as many users as possible.

- Improved Training Efficiency

  Advanced deep learning techniques and frameworks streamline the training process, allowing developers to efficiently create and refine models. This efficiency is critical for maintaining the app's relevance and accuracy over time.
- **summary**

  Deep learning is integral to the development of a currency detection app for blind users due to its accuracy, robustness, real-time processing capabilities, and adaptability. It enables the creation of a reliable, user-friendly, and scalable solution that significantly enhances the independence and quality of life for visually impaired individuals.

2.2 Existing Systems :

1. **Image Processing and Recognition System:**
   This system encompasses the core functionality of recognizing and processing images of Egyptian currency. It involves algorithms and models trained for currency recognition, typically based on machine learning and computer vision techniques.

2. **Machine Learning Model:**
   Central to the project, this model is trained on a dataset of images to classify and identify different denominations of Egyptian currency. Convolutional Neural Networks (CNNs) or other deep learning architectures are commonly used for this purpose.

Initial Model: MobileNet

Architecture:

- MobileNet is a streamlined architecture that uses depthwise separable convolutions to build light-weight deep neural networks. This makes it suitable for mobile and embedded vision applications.

Training Process:

- The model was trained on the collected dataset using transfer learning techniques. Hyperparameters such as learning rate, batch size, and number of epochs were tuned to optimize performance.

Challenges Faced:

- The model's accuracy was low during training.

- **Model: EfficientNet-Lite 2**

Architecture:

- EfficientNet-Lite 2 is part of the EfficientNet family, which scales the network width, depth, and resolution in a principled way. EfficientNet Lite models are optimized for mobile and edge devices, providing a balance between performance and efficiency.

Advantages:

- EfficientNet-Lite 2 offers better accuracy and faster inference times compared to MobileNet, making it more suitable for real-time applications.

Training Process:

- Similar to MobileNet, EfficientNet-Lite 2 was trained using transfer learning. Additional techniques such as dropout and data augmentation were used to prevent overfitting and improve generalization.

Performance Metrics:

- The model's performance is evaluated using metrics like accuracy, precision, recall, F1-score, and inference time. EfficientNet-Lite 2 showed significant improvements in these metrics compared to MobileNet.

3. MediaPipe Integration

MediaPipe is a cross-platform framework for building multimodal machine learning pipelines. It provides functionalities for real-time processing, making it ideal for integrating with mobile applications.

MediaPipe is used to handle image preprocessing, such as face and hand detection, which helps in accurately focusing on the currency note. It also facilitates real-time detection and feedback.

4. Mobile App Interface:

The mobile application interface allows users to interact with the image recognition system. It incorporates the camera functionality, image capture, and processing capabilities required to recognize and display information about the currency notes.

Mobile Application Development

Development Environment:

Android Studio is the integrated development environment (IDE) used for developing the app. Kotlin is the primary programming language due to its modern features and full compatibility with Java.

User Interface:

The UI is designed to be intuitive, with simple navigation and clear instructions. Features include a camera interface for capturing currency images and an output screen displaying the detected denomination and additional information.

Integration with ML Models:

The trained ML models are converted to TensorFlow Lite format for efficient execution on mobile devices. These models are integrated into the app using TensorFlow Lite libraries.

5. **Real-time Processing Module:**
   This system ensures that the app can process images captured through the device's camera in real-time, providing immediate feedback on the denomination and possibly the authenticity of the currency notes.

6. **Dataset or Data Repository:**
   It stores information related to the currency notes, such as their images, denominations.

7. **User Experience and Accessibility System:**
   Focuses on ensuring a user-friendly experience, including intuitive app design, compatibility across various mobile devices and operating systems, and accessibility features for different user needs.

8. **Testing and Quality Assurance System:**
   This system involves comprehensive testing methodologies to ensure the accuracy, reliability, and robustness of the image recognition and currency detection capabilities.

9. **Deployment and Maintenance Infrastructure:**
   Covers the processes and tools needed to deploy the app to app stores and manage updates, bug fixes, and improvements over time.

These systems work together to create a seamless and reliable mobile application capable of accurately detecting and providing information about Egyptian currency notes in real-time. Collaboration among these systems ensures a holistic and functional solution for users.

2.3 Data Collection :

We collected a comprehensive data set that includes images of all Egyptian currency denominations:

1. 5 pounds
2. 10 pounds
3. New10 pounds
4. 20 pounds
5. New20 pounds
6. 50 pounds
7. 100 pounds
8. 200 pounds

Data Set Adjustments

To ensure the data set was suitable for training a robust deep learning model, several adjustments and improvements were made:

Data Augmentation

We applied data augmentation techniques to generate new images from the original ones, increasing the diversity and size of the data set.

Techniques used included:

Rotation: Rotating images at various angles.

Lighting : Changing brightness.

Cropping and Scaling: Cropping parts of the images and resizing them.

Varied Photography Conditions

We captured images of the currency under different conditions to ensure robustness, including:
Different Lighting Conditions: Daylight, artificial light, and low light.
Multiple Angles and Directions: Images taken from various angles and orientations.
Diverse Backgrounds: Using different backgrounds to improve the model's ability to generalize.

Addressing Overfitting

We encountered overfitting, where the model performed well on training data but poorly on test data. To address this, we:

Carefully Split Data:
Ensured a balanced division of training and test sets.

Data Augmentation:
Further increased the data set size to enhance generalization.

Model Simplification:
Reduced model complexity to avoid learning noise and irrelevant details from the training data.

Methods and Techniques Used

Multi-Angle and Lighting Photography

To ensure data comprehensiveness, we took images from different angles and under various lighting conditions.

Performance Analysis and Iteration

After each modification to the data set, we trained the model and analyzed its performance to verify improvements and reduce overfitting.

Utilization of Modern Tools

We employed deep learning libraries such as TensorFlow and PyTorch for building and training the model.

MediaPipe was used to enhance real-time processing and performance in image recognition tasks.

Results and Recommendations

Improved Accuracy and Generalization

Through repeated adjustments and enhancements, the model's accuracy significantly improved, and overfitting was substantially reduced.

Continuous Data Collection

We recommend ongoing data collection from new environments and regular updates to the data set to maintain high model performance.

Types of data :

A. Five pounds

Examples :

B. Ten pounds

Examples:

C. New ten pounds

Examples :

D. Twenty pounds
Examples :

E. New Twenty pounds
Examples :

F. Fifty pounds

Examples :

G. Hundred

Examples :

Two hundred

Examples :

2.4 Overall Problems of Systems :

- **Accuracy and Reliability:**
  Some systems might struggle with accurately identifying different denominations of currency, especially when dealing with worn-out or damaged notes. Variations in lighting conditions or angles can also affect recognition accuracy. References for
  A Review of Currency Recognition System Using Image Processing and Machine Learning Techniques" - This paper discusses the various challenges faced by currency recognition systems, including dealing with worn-out or damaged notes, and the impact of lighting conditions and angles on recognition accuracy.
  Reference: M. R. Rashed, M. A. Razzaque, and M. S. R. Kabir, "A Review of Currency Recognition System Using Image Processing and Machine Learning Techniques," International Journal of Advanced Computer Science and Applications, vol. 11, no. 3, 2020.
- **Speed and Performance:**
  Real-time processing might face challenges in terms of speed, causing delays in accurately recognizing and providing information about currency notes, which can affect user experience.

- Compatibility and Accessibility:
  Compatibility issues across different mobile devices, operating systems, or camera qualities can hinder the widespread usability of these apps. Moreover, accessibility features for users with disabilities might be lacking in some applications.

- Maintenance and Updates:
  Regular maintenance, updates, and improvements are necessary to keep the app functional and aligned with evolving currency designs or security features.

- Resource Intensiveness: Some systems may be computationally intensive, consuming a significant amount of device resources like CPU and memory, which can affect the overall performance of the mobile device.

2.5 Overall Solution Approach :

1. **Enhanced Algorithms and Models:**
   Improving machine learning models by refining algorithms, leveraging larger and more diverse datasets, and employing advanced techniques like transfer learning to enhance accuracy and adaptability across different currencies and conditions.
2. **Robust Image Preprocessing:**
   Implementing advanced preprocessing techniques to handle variations in lighting, angles, and note conditions, ensuring better image quality before feeding them into recognition models.

3. **Real-time Optimization:**
   Optimizing algorithms and models for faster processing and recognition speed, possibly utilizing edge computing or more efficient algorithms to deliver near-instantaneous results.

4. **Compatibility and Accessibility Improvements:**
   Ensuring compatibility across a wide range of devices, operating systems, and camera qualities. Implementing accessibility features to cater to users with diverse needs, such as voice commands or screen readers.

5. **Regular Maintenance and Updates:**
   Establishing a structured maintenance schedule to provide regular updates, address bugs, and incorporate improvements aligned with evolving currency designs or security enhancements.

6. **Resource Optimization:**
   Developing more resource-efficient algorithms and optimizing code to reduce the app's computational intensity, enhancing performance without compromising accuracy.
7. **User Feedback and Testing:**
   Actively collecting user feedback and conducting extensive testing across various scenarios to continuously improve the system's performance, usability, and reliability.

By adopting this multifaceted approach, developers and researchers can systematically tackle the challenges faced by existing currency detection systems, ensuring a more accurate, efficient, secure, and user-friendly experience for individuals using these mobile apps.

2.6 Summary :

- **Project Overview :**

The project centers on creating a mobile app equipped with a sophisticated image recognition system. This system will leverage machine learning algorithms and computer vision techniques to enable users to identify and authenticate different denominations of Egyptian currency notes using their smartphone cameras. And therefore, helping people with visual impairment (blind) identify their money.

- **Objectives:**
  - Accuracy and Recognition: Develop a robust image recognition model capable of accurately identifying various denominations of Egyptian currency.

  - Real-time Processing: Enable real-time processing through the mobile app, allowing users to swiftly capture and analyze currency notes via their device cameras.

  - User-Friendly Interface: Design an intuitive and accessible mobile interface for seamless user interaction and experience.

- **Scope :**

  The project encompass Creation of machine learning models trained on diverse datasets of Egyptian currency notes. Development of a mobile application interface allowing users to capture, process, and receive instant information about detected currency notes.

  Project Deliverables :

  A functional mobile application capable of recognizing and providing information about various denominations of Egyptian currency in real-time.

  Documentation outlining the project's development process, methodologies, and technical aspects.

  Expected Outcomes :

  - Enhanced accessibility and efficiency in identifying Egyptian currency notes.

  - User-friendly and accessible mobile application contributing to financial convenience .

  - Through the successful execution of this project, users will benefit from a reliable, accessible, and efficient mobile tool for handling and Egyptian currency, fostering greater confidence in financial transactions, and helping people with visual impairment (blind).

Chapter three: Application

- **Introducation**:

We have developed the Currency Detection App using image recognition
technology that determines the name of the Currency by processing the image
of the Currency using the EfficientNet-Lite2 model.

We have also linked the Currency Detection App to a special dataset to
display detailed information about the Currency that helps the blind people to understand
and learn more about the Currency.

3.1 Product perspective

Users can simply upload a photo, and the EfficientNet-Lite2 model will

provide the identified Currency, including its common name and label

Classification and information about the Currency.

The user can insert the photo containing the image of the Currency into the

application by taking the photo with his camera phone or inserting it from his

phone's memory.

3.2 User characteristics

There are two types of users who interact with the system: customer and administrator, each of these types of users have different use of the system so that each of them has their own requirements.

The customer:

Opens the application, uses the application to choose an operation he/she wants to do and then the application will perform it.

The administrator:

Can improve the application for new updates. He can add verified captains to the system.

3.4 Use Cases

Nowadays, everyone usually uses mobile phone all the time.so we created an easy-to use mobile app to help blind people in thier lives.

He faces many situations in which he needs to identify the Currency he finds or wants information about the Currency to use in his professional life or simply to identify and explore it. Therefore, we developed Currency Detection App to identify Currency through the image .

User can use this mobile application to:

1. The user can download Currency Detection App easily.

2. User can take the photo with his phone or inserting it from his

phone's memory.

3. Users can simply upload a photo to the application.

4. The EfficientNet-Lite2 model will provide the identified Currency.

5. Identified Currency, its name and Listen to the pronunciation of the selected currency

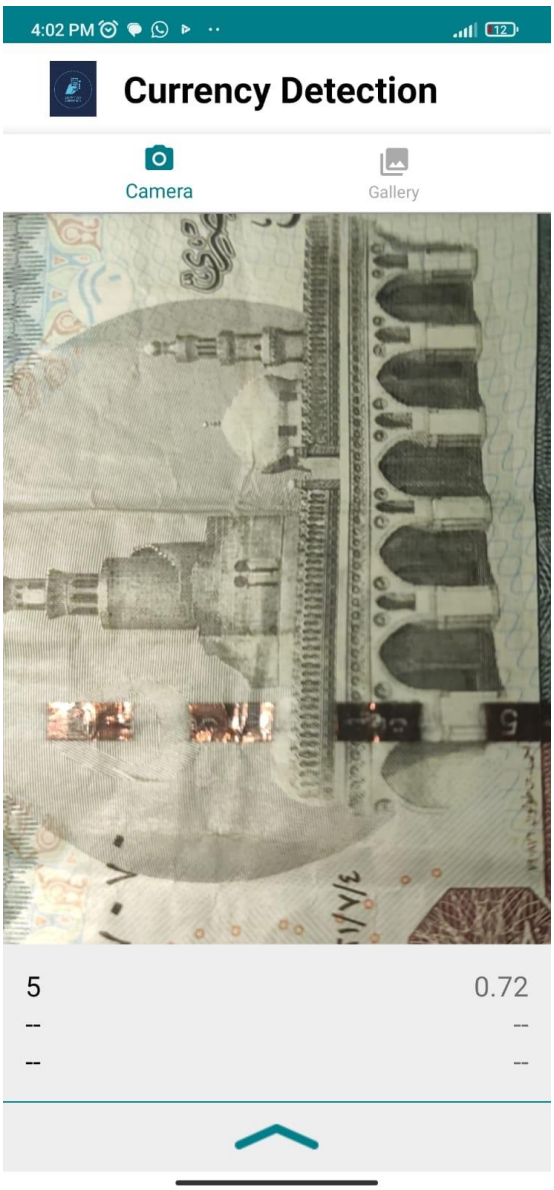
3.5 Design

- **Splash Screen page:**

  On this page, the logo for the Currency Detection application is displayed

- **Live camera:**

  When the Live Scan page is opened, the device's camera is turned on to enable the user to point it at the Currency he/she wants to identify. We have easy-to-use user interface



- Scan page

  This page is responsible for taking a picture or upload it from device memory of the Currency from the user,

  scanning the image and sending it to the EficientNet lite 2 model to analyze and process the image to identify the Currency and retrieve information about

  this Currency after identifying it.

  The user takes a picture of the unknown Currency with his phone's camera or uploads the picture of the unknown Currency from the user's phone's memory in an easy and simple way that saves time for the user.

  Currency Detection App, through the camera page, takes a photo from the user and scans the image to identify the Currency using a model

3.6 Kotlin

Kotlin is a modern, statically typed programming language that has been officially supported by Google for Android app development since 2017.

It's designed to be fully interoperable with Java, which means we can use both languages within the same project. With the constant development of Kotlin by each of JetBrains and Google working on improving the

language and its tooling. The Whether we're building a small app or a large-scale enterprise application, Kotlin has the features and support to make our Android development process smoother and more enjoyable.

The most important points that made us use the Kotlin language:

Conciseness: Kotlin reduces boilerplate code, which means we write less code for the same functionality compared to Java.

Null Safety: The language design incorporates null safety to prevent the common NullPointerExceptions that are often encountered in Java.

Extension Functions: These allow us to extend a class with new functionality without having to inherit from the class.

Smart Casts: The compiler tracks conditions inside if-expressions and automatically casts types if possible.

Data Classes: Simplify the creation of classes that just hold data and automatically generate such 'as getters', 'setters', 'equals()', 'hashCode()', and 'toString()' methods.

Safety: Kotlin’s type system is designed to eliminate the danger of null references from code.

3.7 Android Studio

Android Studio is the official Integrated Development Environment (IDE)

for Android app development, based on IntelliJ IDEA.

It's designed to provide a comprehensive set of tools to enhance productivity and ease the app development process and we used some features of Android Studio:

Project Structure:

Organized by modules for quick access to key source

files, including manifests, Kotlin source code, and resources.

Gradle Build System:

Allows customization of the build process, creation

of multiple APKs, and resource reuse across source sets.

Code Editor:

Intelligent code editor with code completion for Kotlin.

Emulator:

A fast and feature-rich emulator to test applications on a variety of Android devices.

Live Edit:

Updates composables in emulators and physical devices in realtime as we code.

Testing Tools:

Extensive testing tools and frameworks to ensure app Quality

3.8 MediaPipe Solutions

MediaPipe Solutions provides a suite of libraries and tools for we to quickly apply artificial intelligence (AI) and machine learning (ML) techniques in our applications.

We can plug these solutions into our applications immediately, customize them to our needs, and use them across multiple development platforms. MediaPipe Solutions is part of the MediaPipe open source project, so we can further customize the solutions code to meet our application needs. The MediaPipe Solutions suite includes the following:

These libraries and resources provide the core functionality for each MediaPipe Solution:

MediaPipe Tasks:

Cross-platform APIs and libraries for deploying solutions.

MediaPipe Models:

Preavious trained, ready-to-run models for use with each solution

These tools let us customize and evaluate solutions:

MediaPipe Model Maker:

Customize models for solutions with our data.

Image classification task

The MediaPipe Image Classifier task let us perform classification on images. We can use this task to identify what an image represents among a set of categories defined at training time. This task operates on image data with a machine learning (ML) model as static data or a continuous stream and outputs a list of potential categories ranked by descending probability score.

Task details

This section describes the capabilities, inputs, outputs, and configuration options of this task.

CHAPTER FOUR : SYSTEM REQIRMENTS ENGINEERING AND PLANNIING

4.1 Introduction

This chapter for our currency detection project aimed at assisting visually impaired individuals. In this chapter, we will delve into the critical aspects of gathering, analyzing, and documenting requirements essential for the successful development of our mobile application.

Context Establishment:

Our project focuses on developing a mobile application that empowers visually impaired individuals to identify different denominations of Egyptian currency using their smartphones. This application is designed to enhance accessibility and independence for this user group, addressing a significant need in our community.

Purpose Statement:

The purpose of this chapter is to outline a systematic approach to requirements engineering and planning, emphasizing their paramount importance in ensuring that our final product aligns with user needs and expectations. By establishing clear requirements early in the project lifecycle, we aim to mitigate risks, reduce rework, and deliver a solution that truly meets the needs of our stakeholders.

Outline of the Chapter:

This chapter will guide you through various stages of requirements engineering and planning, starting with a feasibility study to assess the viability of our project. We will explore techniques for eliciting requirements from stakeholders and users, define both functional and non-functional requirements, and conclude with a summary of key findings and insights.

Significance:

Requirements engineering and planning are fundamental to the success of any project. By understanding the needs of our users and defining system functionalities upfront, we can minimize misunderstandings, optimize resource utilization, and ultimately deliver a product that adds tangible value to our target audience.

Audience Consideration:

This chapter is intended for project stakeholders, developers, and anyone involved in the requirements gathering and planning process. Whether you're a seasoned professional or new to the field, our goal is to provide you with practical insights and strategies to navigate this critical phase of the project effectively.

As we embark on this journey of requirements engineering and planning, let us remain focused on our shared vision of creating an inclusive and empowering solution for visually impaired individuals. Through collaboration and dedication, we can turn our vision into reality.

4.2 Feasibility

The Feasibility Study section is a crucial phase in our project's lifecycle, aimed at assessing the practicality and viability of our currency detection application from various angles. This assessment ensures that our resources are allocated efficiently and that the project stands a realistic chance of success.

Technical Feasibility: This aspect evaluates whether our project can be technically implemented given the available resources and technology. We will assess the availability of suitable machine learning algorithms for currency recognition, the compatibility of Kotlin and Android Studio for our development needs, and any technical challenges that may arise during implementation.

Economic Feasibility: Economic feasibility examines whether our project is financially viable within the allocated budget and whether the potential benefits outweigh the costs. We'll estimate the development costs, including expenses for software development, hardware requirements, personnel, and any other associated costs. Additionally, we'll conduct a cost-benefit analysis to determine whether the anticipated benefits, such as enhanced accessibility for visually impaired individuals, justify the investment.

Operational Feasibility: Operational feasibility evaluates whether our project can be seamlessly integrated into existing processes and systems without causing disruption. We'll

consider factors such as the compatibility of our mobile application with various smartphone models and operating systems, the ease of use for both visually impaired users and developers, and any potential logistical challenges related to deployment and maintenance.

Risk Assessment: As part of the feasibility study, we'll identify and evaluate potential risks that could impact the success of our project. This includes technical risks such as algorithm accuracy and performance issues, economic risks such as budget overruns and funding constraints, and operational risks such as user adoption challenges. We'll develop risk mitigation strategies to address these potential threats proactively.

Feasibility Study Report: This report will summarize the results of the technical, economic, and operational assessments, as well as any identified risks and their mitigation strategies. It will serve as a valuable reference for project stakeholders and decision-makers, informing their decision on whether to proceed with the project or explore alternative solutions.

In summary, the Feasibility Study section provides a thorough assessment of our currency detection project's feasibility, considering technical, economic, and operational factors, as well as potential risks. This evaluation is essential for making informed decisions and ensuring the success of our project.

4.3 Requirements Elicitation Techniques

In this section, we explore various methods used to gather requirements from stakeholders and users, crucial for the development of our currency detection application.

Interviews: Interviews involve direct conversations with stakeholders and users to gather insights into their needs, preferences, and challenges. Structured interviews follow a predefined set of questions, while unstructured interviews allow for more open-ended discussions. Both types offer valuable insights into user requirements.

Prototyping: Prototyping involves creating early versions of the application to gather feedback. Prototypes can range from sketches to interactive mockups. Testing prototypes with users validates requirements and identifies areas for improvement.

Document Analysis: Document analysis reviews existing documentation, such as user manuals or system specifications. This provides context and insights into current processes, aiding in identifying areas for improvement or integration.

Stakeholder Collaboration: Collaboration with stakeholders throughout the process ensures their needs are met. Involving stakeholders in discussions, decision-making, and validation activities builds trust and ensures the final product aligns with their requirements.

By leveraging these techniques in collaboration with faculty members and friends, we can ensure that the currency detection application meets the needs and expectations of its users while benefiting from diverse perspectives and expertise.

4.4 Targeted Users

In this section, we identify the primary individuals who will benefit from and interact with our currency detection application. Our targeted users are visually impaired individuals who rely on the application to identify different denominations of Egyptian currency. Here's a breakdown of this section:

Primary Users: Visually impaired individuals are the primary users of our currency detection application. They face challenges in identifying currency denominations due to their visual impairments, which can impact their independence and financial autonomy. Our application aims to address these challenges by providing a convenient and accessible solution for currency recognition using smartphones.

Characteristics of Targeted Users: Understanding the characteristics and needs of visually impaired individuals is essential for designing an effective and user-friendly application. Factors such as varying degrees of visual impairment, familiarity with smartphone technology, and preferences for auditory or tactile feedback should be considered during the application's development.

Challenges Faced by Targeted Users: Visually impaired individuals encounter various challenges in their daily lives, including difficulties in identifying currency. This can lead to dependence on others for assistance and may hinder their ability to manage finances independently. Our application seeks to mitigate these challenges by providing a reliable and intuitive solution for currency recognition.

User-Centered Design Approach: We adopt a user-centered design approach to ensure that our currency detection application meets the needs and preferences of visually impaired individuals. This involves actively involving users in the design and development process, soliciting feedback, and incorporating their input to create a solution that is truly user-centric.

Accessibility Considerations: Accessibility is a fundamental aspect of our currency detection application, given its target audience of visually impaired individuals. The application should adhere to accessibility standards and guidelines to ensure that it is usable by individuals with various types and degrees of visual impairment. This includes features such as screen reader compatibility, high contrast interfaces, and intuitive navigation.
By identifying visually impaired individuals as the targeted users of our currency detection application and understanding their characteristics, challenges, and needs, we can ensure that the application is designed and developed with their specific requirements in mind. Adopting a user-centered design approach and prioritizing accessibility considerations will enable us to create a solution that enhances the independence, accessibility, and quality of life for visually impaired individuals in their daily interactions with currency.

4.5 Functional Requirements Definition

In this section, we delineate the specific functionalities that our currency detection application must encompass to effectively serve visually impaired users. These functional requirements lay down the groundwork for the application's development, ensuring it aligns with the users' needs and expectations. Here's a comprehensive overview:

Identification of Core Functions: Our primary objective is to identify the core functions essential for the currency detection application's functionality. These functions are pivotal in enabling visually impaired users to accurately identify various denominations of Egyptian currency.

Image Capture: The application should facilitate users in capturing images of currency notes using their smartphone cameras. This feature enables users to input currency images into the application for identification purposes.

Currency Recognition: A fundamental aspect of the application is its ability to robustly recognize and differentiate between different denominations of Egyptian currency based on the provided images. This core functionality forms the backbone of the application's purpose.

Auditory Feedback: Following the currency recognition process, the application must deliver auditory feedback to the user. This feedback mechanism can take the form of verbal announcements or auditory cues, ensuring users receive immediate and accessible information regarding the identified currency denomination.

User Interface: The application should boast an intuitive and accessible user interface tailored explicitly for visually impaired users. This includes design elements such as high contrast visuals, large buttons, and options for tactile feedback, all geared towards enhancing usability and accessibility.

Real-time Processing: It is imperative that the currency recognition process occurs in real-time, offering users instantaneous feedback regarding the identified denomination. Real-time processing minimizes delays, ensuring a seamless user experience.

Offline Functionality: The application should be equipped to perform currency recognition tasks offline, without necessitating a continuous internet connection. This offline functionality ensures users can utilize the application irrespective of their location or internet connectivity status.

Error Handling: Robust error handling mechanisms must be integrated into the application to address any potential issues encountered during the currency recognition process. These mechanisms should encompass scenarios such as image quality inadequacies, unrecognized denominations, or processing errors, guaranteeing users receive accurate and dependable feedback consistently.

By meticulously defining these functional requirements, we establish a comprehensive framework outlining the core features and capabilities our currency detection application must embody. These requirements serve as guiding the development, implementation, and testing phases, thereby ensuring the final product aligns seamlessly with the needs and preferences of visually impaired users.

4.6 Functional Requirements Specification

In this section, we provide a detailed breakdown of the functional requirements outlined earlier, elucidating how each requirement will be implemented within the currency detection application. This specification serves as a comprehensive guide for our team, offering clarity on the features and functionalities to be incorporated. Let's delve into the specifics:

Image Capture:

- Users should be able to initiate the image capture process through a designated button or gesture within the application's interface.
- Upon activation, the smartphone's camera module will be launched, allowing users to align and capture images of currency notes effectively.

Currency Recognition:

- Robust algorithms leveraging machine learning and image processing techniques will be integrated to recognize and distinguish between different denominations of Egyptian currency based on the captured images.
- Trained models specific to each currency denomination will be utilized to facilitate accurate recognition.
- The recognition process will be executed locally on the user's device to ensure real-time processing.

Auditory Feedback:

- Upon successful currency recognition, the application will provide auditory feedback to the user, conveying the identified denomination.
- Auditory feedback will be delivered in a clear and concise manner, utilizing synthesized speech or pre-recorded audio prompts.

User Interface:

- The application's user interface will prioritize accessibility, featuring high contrast visuals, large and easily distinguishable buttons, and intuitive navigation options.
- Text labels, buttons, and icons will be appropriately sized and spaced to facilitate easy interaction for visually impaired users.
- Tactile feedback mechanisms, such as vibration patterns or haptic feedback, will be incorporated to provide additional cues and enhance usability.

Real-time Processing:

- The currency recognition process will be optimized for real-time performance, minimizing latency between image capture and feedback delivery.
- Efficient algorithms and processing techniques will be employed to reduce computational overhead and ensure consistent real-time processing across various smartphone hardware configurations.

Offline Functionality:

- The application will support offline currency recognition, eliminating the dependency on a continuous internet connection.
- Trained machine learning models and currency databases will be packaged within the application's installation package, enabling local processing without external data sources.
- Users will receive notifications of currency data updates or model improvements when an internet connection is available.

Error Handling:

- Robust error handling mechanisms will be implemented to address various scenarios encountered during the currency recognition process.
- Common error conditions, such as blurry images or unrecognized denominations, will be detected and communicated to the user.
- Clear and informative error messages will guide users on corrective actions or suggest alternative approaches to improve recognition accuracy.

By meticulously specifying these functional requirements, we provide a comprehensive roadmap for the development team, ensuring clarity, consistency, and alignment with the project's objectives. This detailed specification facilitates efficient implementation and contributes to the successful realization of the currency detection application.

4.7 Non-Functional Requirements

This section outlines the non-functional requirements for the currency detection application. Unlike functional requirements that specify what the system should do, non-functional requirements define the attributes and characteristics that the system should possess. These requirements are crucial for ensuring the overall quality, performance, and usability of the application. Let's dive into each aspect:

Usability:

- The application should be designed with user-friendliness in mind, ensuring it is intuitive and easy to navigate for visually impaired users.
- Clear and concise instructions should be provided throughout the application to guide users effectively.
- Accessibility features such as screen reader compatibility, voice commands, and customizable user preferences should be incorporated to enhance usability for users with visual impairments.

Performance:

- The currency recognition process should be swift and responsive, with minimal latency between image capture and feedback delivery.
- The application should be optimized to run efficiently on various smartphone devices, ensuring smooth performance and minimal resource consumption.
- Regular performance monitoring and optimization should be conducted to maintain high levels of responsiveness and efficiency.

Reliability:

- The application should be reliable and stable, with robust error handling mechanisms in place to handle unexpected errors or interruptions gracefully.
- Currency recognition accuracy should be high, minimizing false positives or false negatives to ensure reliable identification of currency denominations.
- Thorough testing, including unit tests, integration tests, and user acceptance testing, should be performed to validate reliability and identify any potential issues.

Scalability:

- The application should be designed to accommodate a growing user base and increasing usage volumes over time.
- Scalability features such as cloud-based infrastructure, load balancing, and auto-scaling should be implemented to handle spikes in traffic or demand effectively.
- Regular performance testing and capacity planning should be performed to ensure that the application can scale seamlessly as needed.

Compatibility:

- The application should be compatible with a wide range of smartphone devices, operating systems, and screen sizes to maximize accessibility for users.
- Compatibility testing should be conducted across different device configurations and platforms to verify consistent performance and user experience.
- Updates and optimizations should be made as necessary to maintain compatibility with evolving technology standards.

Regulatory Compliance:

- The application should comply with relevant regulations and standards, including accessibility standards for visually impaired users and data privacy regulations such as GDPR.
- Compliance documentation should be maintained and updated regularly to demonstrate adherence to regulatory requirements and mitigate legal risks.

By addressing these non-functional requirements, we ensure that our currency detection application not only meets the functional needs of visually impaired users but also delivers a high-quality, reliable, and secure user experience.
Adhering to these requirements is essential for building trust with users, maintaining compliance with regulations, and achieving long-term success in the market.

4.8 Summary

This section serves as a concise recapitulation of the essential aspects covered throughout the system requirements engineering and planning phase for our currency detection application project. It encapsulates the key points discussed in the preceding chapters, offering stakeholders a quick reference and overview of the project's scope and objectives. Let's highlight the main components covered:

Introduction:

emphasizing the significance of meticulous planning and requirement gathering in laying the groundwork for the successful development of our currency detection application.

Feasibility Study:

A thorough feasibility study evaluated the project's technical, economic, and operational viability, identifying potential challenges and opportunities.

Requirements Elicitation Techniques:

Techniques such as interviews, Prototyping , Document Analysis , Stakeholder Collaboration were used to gather comprehensive requirements from stakeholders, ensuring the application meets user needs effectively.

Targeted Users:

Visually impaired individuals were identified as the primary users. Understanding their unique needs helps us design an application that effectively addresses their challenges.

Functional Requirements Definition:

We defined essential functions like image capture, currency recognition, auditory feedback, user interface design, and error handling.

Functional Requirements Specification:
Detailed descriptions of each functional requirement guide the development team, ensuring successful implementation of the application's features.

Non-Functional Requirements:
Non-functional requirements like usability, performance, reliability, scalability, compatibility, and regulatory compliance were identified to ensure overall quality and effectiveness.

Summary:
The system requirements engineering and planning phase provided a solid foundation for the currency detection application. This stage sets the groundwork for development, implementation, and testing, ensuring a high-quality, user-friendly application for visually impaired individuals.

CHAPTER FIVE: Explaination the code

Code

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import os
import cv2
from tqdm import tqdm
import tensorflow as tf
from tensorflow import keras

from google.colab import files
import os
import tensorflow as tf
assert tf.__version__.startswith('2')

from mediapipe_model_maker import image_classifier

import matplotlib.pyplot as plt



import os

# Example usage in Colab
folder = "/content/drive/MyDrive/data_set" # change path
```

```
# Joining paths in Colab
image_path = os.path.join("/content", "drive", "MyDrive",
folder,)


print(image_path)
labels = []
for i in os.listdir(image_path)
  if os.path.isdir(os.path.join(image_path, i))
    labels.append(i)
print(labels)


NUM_EXAMPLES = 5

for label in labels
  label_dir = os.path.join(image_path, label)
  example_filenames = os.listdir(label_dir)[NUM_EXAMPLES]
  fig, axs = plt.subplots(1, NUM_EXAMPLES, figsize=(10,2))
  for i in range(NUM_EXAMPLES)
    axs[i].imshow(plt.imread(os.path.join(label_dir,
example_filenames[i])))
    axs[i].get_xaxis().set_visible(False)
    axs[i].get_yaxis().set_visible(False)
  fig.suptitle(f'Showing {NUM_EXAMPLES} examples for {label}')

plt.show()
```

```
data = image_classifier.Dataset.from_folder(image_path)
train_data, remaining_data = data.split(0.8)
test_data, validation_data = remaining_data.split(0.5)


spec = image_classifier.SupportedModels.EFFICIENTNET_LITE2
hparams =
image_classifier.HParams(learning_rate=0.3,epochs=15,export_
dir="exported_model")
options =
image_classifier.ImageClassifierOptions(supported_model=spec
, hparams=hparams)
options.model_options =
image_classifier.ModelOptions(dropout_rate = 0.07)


model = image_classifier.ImageClassifier.create(
   train_data = train_data,
   validation_data = validation_data,
   options=options,
)


loss, acc = model.evaluate(test_data)
print(f'Test loss{loss}, Test accuracy{acc}')


model.export_model()
```

```
!ls exported_model
files.download('exported_model/model.tflite')
```

B. Explanation of the code

This script organizes files from a main folder into training, testing, and validation folders based on specified ratios. Here's a summary of what it does

1. It imports the necessary libraries 'os' for interacting with the file system.
2. It sets the paths for the main folder and the folders where the data will be organized for training, testing, and validation.
3. It defines the ratios for splitting the data into training, testing, and validation sets.

This script is useful for organizing data into separate sets for machine learning tasks such as training and evaluating models. It ensures that the data distribution follows the specified ratios, facilitating fair evaluation and validation of the models.

import numpy as np

- Imports the numpy library, which is essential for numerical operations in . It provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

import matplotlib.pyplot as plt

- Imports the pyplot module from matplotlib, a plotting library used for visualizing data. It provides functions for creating a wide variety of static, animated, and interactive plots.

%matplotlib inline

- A magic command specific to Jupyter notebooks (and Google Colab) that ensures plots are displayed directly within the notebook.

import os

- Imports the os module, which provides a way to interact with the operating system. It allows for tasks such as reading and writing to the file system, navigating directories, and obtaining environment variables.

import cv2

- Imports OpenCV, a powerful library for computer vision tasks. It includes functions for image processing, video capture, and analysis, such as object detection and face recognition.

from tqdm import tqdm

- Imports the tqdm library, which is used for creating progress bars. This is useful for tracking the progress of loops, especially in long-running tasks like data loading or model training.

import tensorflow as tf

- Imports TensorFlow, a comprehensive open-source platform for machine learning. It provides a wide range of tools for building and training neural networks.

from tensorflow import keras

- Imports the keras module from TensorFlow, which offers a high-level API for building and training deep learning models. It simplifies the process of defining and training neural networks.

from google.colab import files

- Imports the files module from Google Colab, which provides functions to handle file operations in Google Colab, such as uploading and downloading files.

from mediapipe_model_maker import image_classifier

- Imports the image_classifier module from MediaPipe Model Maker, which is used for creating image classification models. MediaPipe is a framework for building multimodal (e.g., video, audio) machine learning pipelines.

Setting Up Paths and Labels

folder = "/content/drive/MyDrive/data_set" # change path

- Specifies the path to your dataset stored in Google Drive. This is the directory where the images of the banknotes are stored.

image_path = os.path.join("/content", "drive", "MyDrive", folder)

- Constructs the full path to the dataset directory by joining parts of the file path using os.path.join, which ensures the correct format regardless of the operating system.

print(image_path)

- Prints the path to the dataset directory to verify that it has been correctly set.

labels = []

- Initializes an empty list called labels that will store the names of subdirectories within the dataset directory. Each subdirectory corresponds to a class label (e.g., different denominations of banknotes).

for i in os.listdir(image_path)

- Iterates over the contents of the dataset directory using os.listdir, which returns a list of all files and directories in the specified path.

If os.path.isdir(os.path.join(image_path, i))

- Checks if the current item is a directory using os.path.isdir. This ensures that only directories (which represent different classes) are considered.

labels.append(i)

- Adds the directory name (which corresponds to a class label) to the labels list if it is a directory.

print(labels)

- Prints the list of labels (subdirectory names) to verify that the correct labels have been collected.

Displaying Sample Images

NUM_EXAMPLES = 5

- Defines the number of example images to display for each label. This helps in visually inspecting a sample of the dataset.

for label in labels

- Iterates over each label in the labels list to process images for each class.

label_dir = os.path.join(image_path, label)

- Constructs the path to the directory containing images for the current label by joining the base dataset path with the label name.

example_filenames = os.listdir(label_dir)[NUM_EXAMPLES]

- Lists the filenames in the label directory and selects the first NUM_EXAMPLES files. This provides a subset of example images for display.

fig, axs = plt.subplots(1, NUM_EXAMPLES, figsize=(10, 2))

- Creates a grid of subplots for displaying multiple images. Here, it creates a row of NUM_EXAMPLES subplots with a specified figure size.

for i in range(NUM_EXAMPLES)

- Iterates over the number of example images to be displayed.

axs[i].imshow(plt.imread(os.path.join(label_dir, example_filenames[i])))

- **plt.imread**: Reads an image from a file.
- **imshow**: Displays the image in the subplot axs[i].

```
    axs[i].get_xaxis().set_visible(False)
```

- **get_xaxis().set_visible(False)**: Hides the x-axis ticks for the subplot.

```
    axs[i].get_yaxis().set_visible(False)
```

- **get_yaxis().set_visible(False)**: Hides the y-axis ticks for the subplot.

```
  fig.suptitle(f'Showing {NUM_EXAMPLES} examples for {label}')
```

- **fig.suptitle**: Sets a title for the entire figure.

```
plt.show()
```

- **plt.show**: Displays the figure with the subplots.

Creating and Splitting the Dataset

```
data = image_classifier.Dataset.from_folder(image_path)
```

- **image_classifier.Dataset.from_folder**: Creates a dataset object from the images in the specified folder. The folder should be organized such that each subfolder corresponds to a class label.

```
train_data, remaining_data = data.split(0.8)
```

- **data.split(0.8)**: Splits the dataset into training (80%) and remaining (20%) subsets.

```
test_data, validation_data = remaining_data.split(0.5)
```

- **remaining_data.split(0.5)**: Further splits the remaining data into testing (10% of the original data) and validation (10% of the original data) subsets.

Model Specification and Training

```
spec = image_classifier.SupportedModels.EFFICIENTNET_LITE2
```

- **spec**: Specifies the EfficientNet-Lite2 model, a lightweight CNN architecture suitable for mobile devices.

```
hparams = image_classifier.HParams(learning_rate=0.3, epochs=15, export_dir="exported_model")
```

- **image_classifier.HParams**: Sets hyperparameters for model training.
  - **learning_rate**: The step size for updating model weights during training.
  - **epochs**: The number of times the training algorithm will iterate over the entire training dataset.
  - **export_dir**: The directory where the trained model will be saved.

```
options = image_classifier.ImageClassifierOptions(supported_model=spec, hparams=hparams)
```

- **image_classifier.ImageClassifierOptions**: Configures options for the image classifier, including the selected model and hyperparameters.

```
options.model_options = image_classifier.ModelOptions(dropout_rate=0.07)
```

- **image_classifier.ModelOptions**: Sets additional model options.
  - **dropout_rate**: The fraction of the input units to drop during training to prevent overfitting.

```
model = image_classifier.ImageClassifier.create(
    train_data=train_data,
    validation_data=validation_data,
    options=options,
)
```

- **image_classifier.ImageClassifier.create**: Trains the image classification model using the specified training and validation data and options.
  - **train_data**: The training dataset.
  - **validation_data**: The validation dataset.
  - **options**: The configuration options for the model, including model type, hyperparameters, and model-specific options.

Model Evaluation

```
loss, acc = model.evaluate(test_data)
```

- **model.evaluate**: Evaluates the trained model on the test dataset.
  - **test_data**: The test dataset.
  - **loss**: The value of the loss function on the test dataset.
  - **acc**: The accuracy of the model on the test dataset.

```
print(f'Test loss: {loss}, Test accuracy: {acc}')
```

- **print**: Outputs the test loss and accuracy to the console.

Model Export and Download

```
model.export_model()
```

- **model.export_model**: Exports the trained model to the specified directory (exported_model).

```
!ls exported_model
```

- **!ls exported_model**: A shell command that lists the contents of the exported_model directory to verify that the model has been exported.

```
files.download('exported_model/model.tflite')
```

- **files.download**: Downloads the exported model (model.tflite) to the local machine.

This project involves creating an image classification model to detect and classify Egyptian currency notes. The process includes:

1. **Data Preparation**: Collecting and preprocessing images of banknotes.
2. **Model Selection**: Choosing and configuring the EfficientNet-Lite2 model for image classification.
3. **Training and Validation**: Training the model on the dataset and validating its performance.
4. **Evaluation**: Testing the model on unseen data to assess its accuracy and robustness.
5. **Deployment**: Exporting and downloading the trained model for integration into a mobile application.

This approach ensures that the developed model is both efficient and accurate, making it suitable for deployment in a mobile app.

CHAPTER SIX : SYSTEM DESIGN

6.1 Introduction

Delineating its role and importance in the development process of our currency detection application. It serves as a prelude, providing context and setting the stage for the subsequent sections. Let's delve into the key aspects of this section:

Purpose:

This underscores the pivotal role of design in shaping the architecture, functionality, and user experience of the currency detection application. By outlining the objectives and goals of the design phase, it establishes a clear direction for the ensuing discussions.

Significance:

This section emphasizes the criticality of employing structured design methodologies to ensure the success of the currency detection application. It highlights the importance of systematic planning, analysis, and visualization in transforming abstract requirements into tangible system components and interfaces. By adhering to rigorous design practices, the project team can mitigate risks, enhance productivity, and deliver a high-quality solution.

Overview:

It outlines the key components, methodologies, and techniques that will be explored in subsequent sections, offering readers a roadmap for navigating through the design process. This

overview sets expectations and prepares readers for the in-depth discussions to follow.

Context:

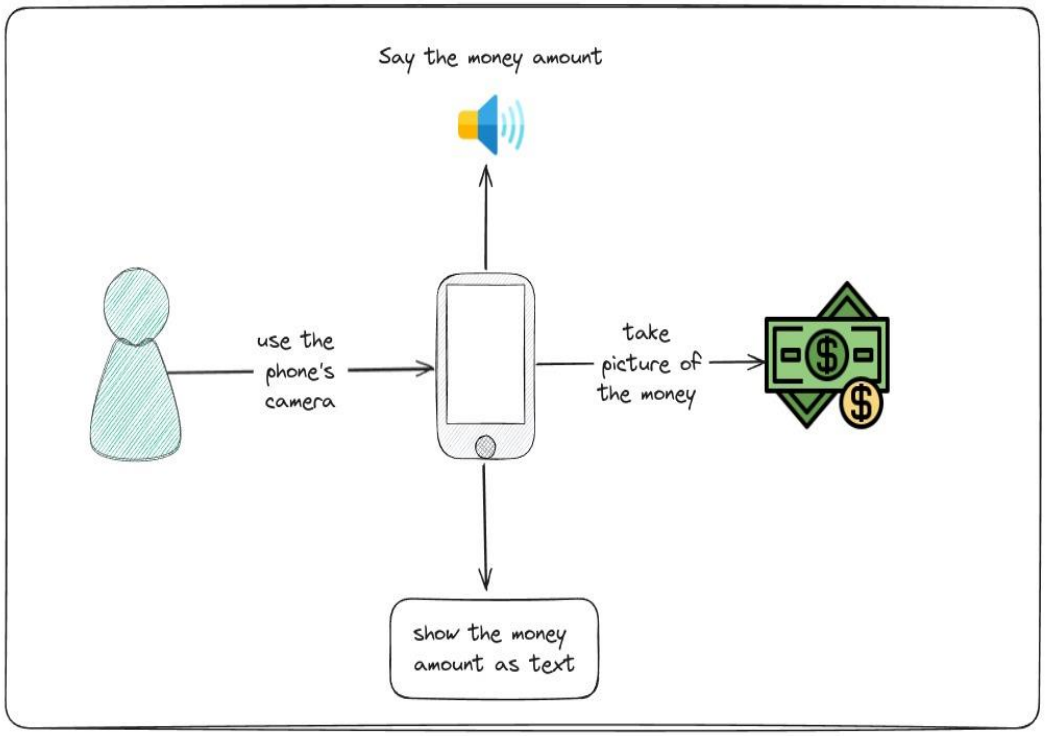
This could include background details about the project's inception, stakeholders involved, technological landscape, and any external factors influencing design decisions. Contextual insights help readers grasp the broader context within which the design process unfolds.

Audience:

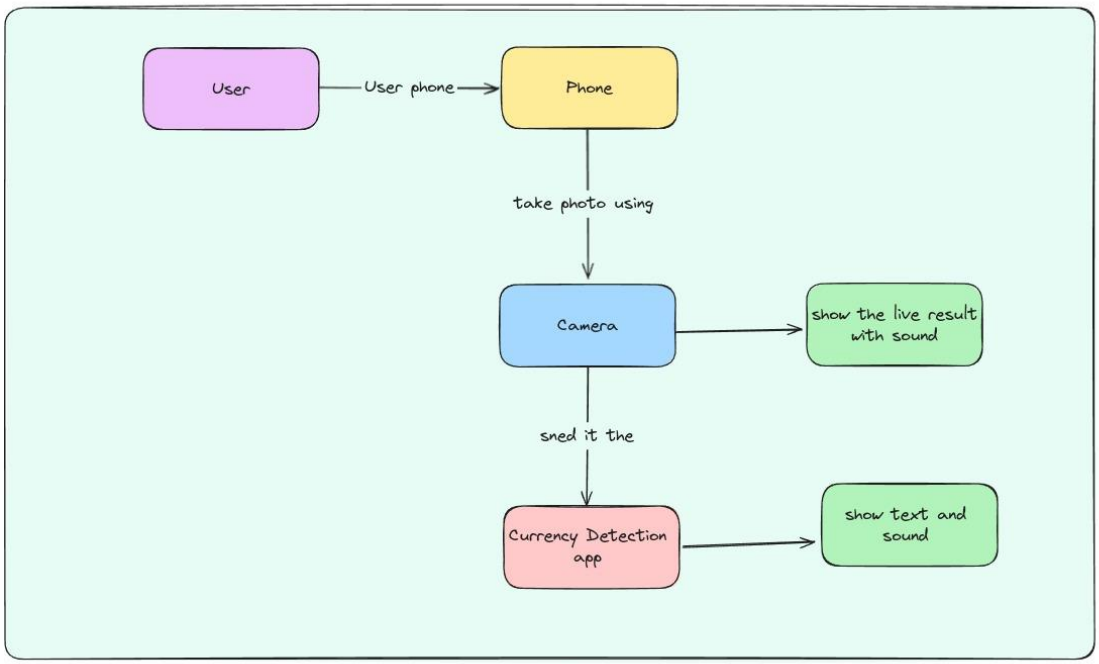
This could include developers, designers, project managers, clients, and other stakeholders invested in the project's success. By tailoring the content to the needs and expectations of the target audience, the Introduction ensures that readers derive maximum value from the subsequent discussions on design methodologies and components.

providing readers with a clear understanding of its purpose, significance, and scope. By elucidating the objectives, setting expectations, and offering contextual insights, it paves the way for an informed and structured exploration of design principles and practices in the context of the currency detection application.

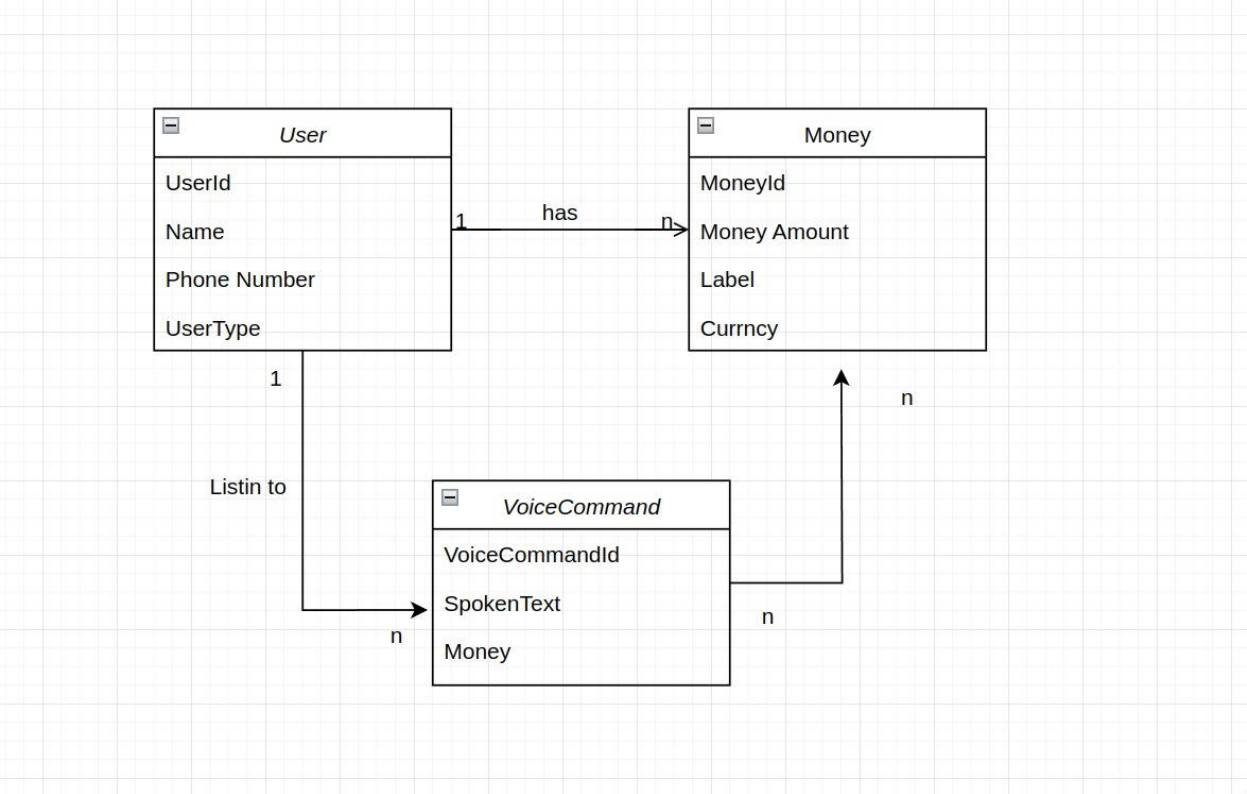
6.2 Context Diagram



6.3 Data Flow Diagram (DFD)



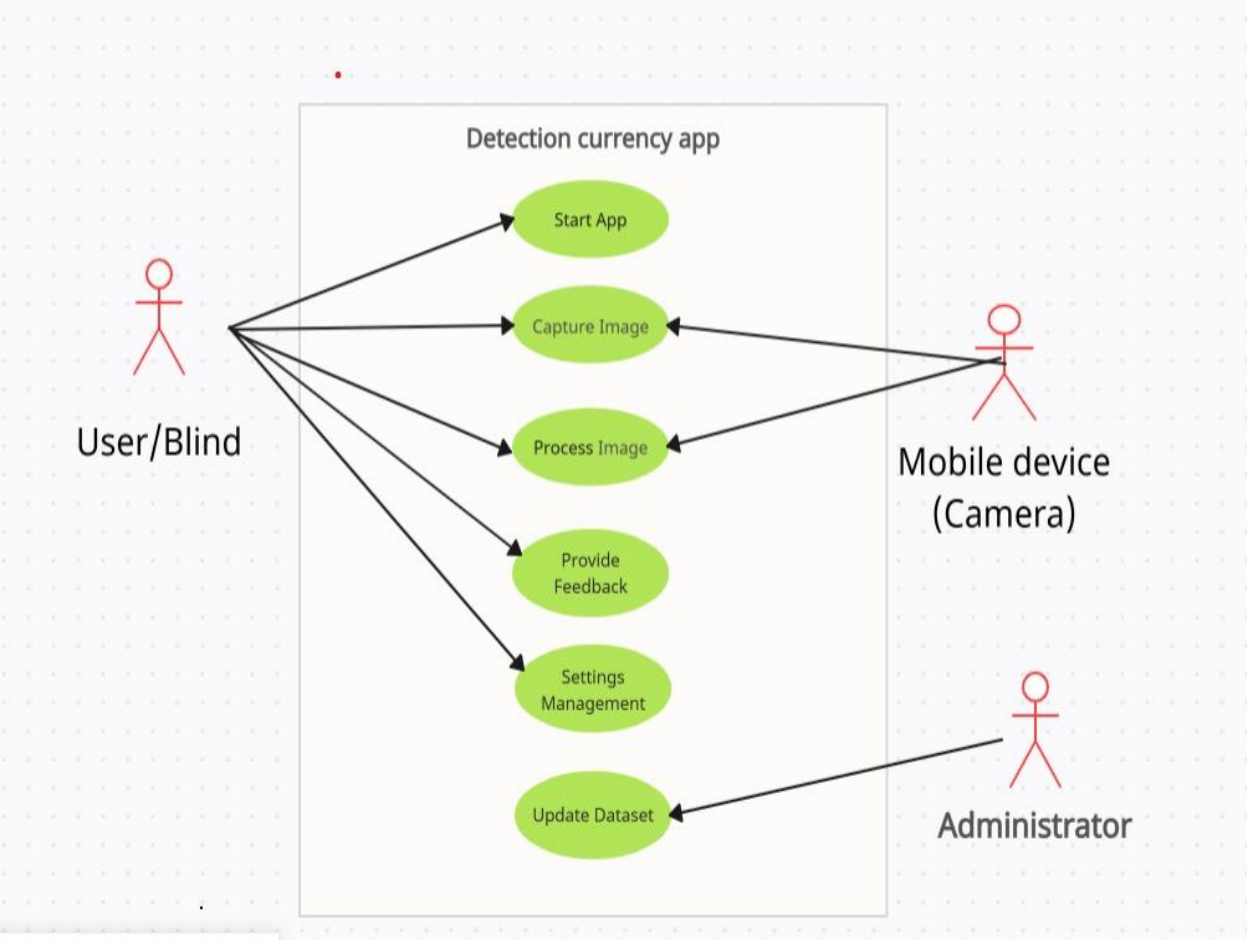
6.4 Entity Relationship Diagram(ERD)



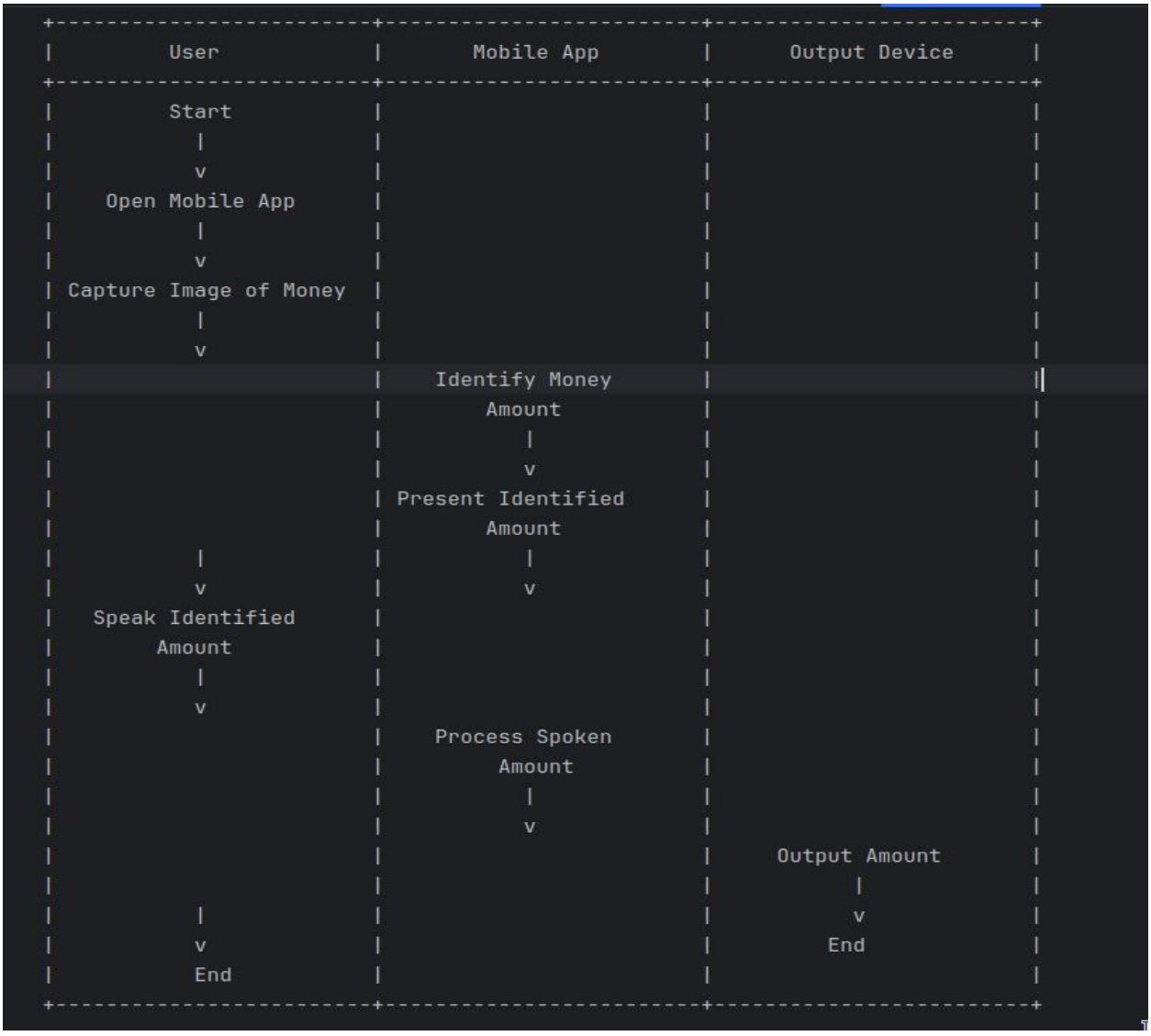
Object-Oriented Design: focuses on the objects that make up the system and how they interact. The key diagrams used here include:

Object-Oriented Design Requirements

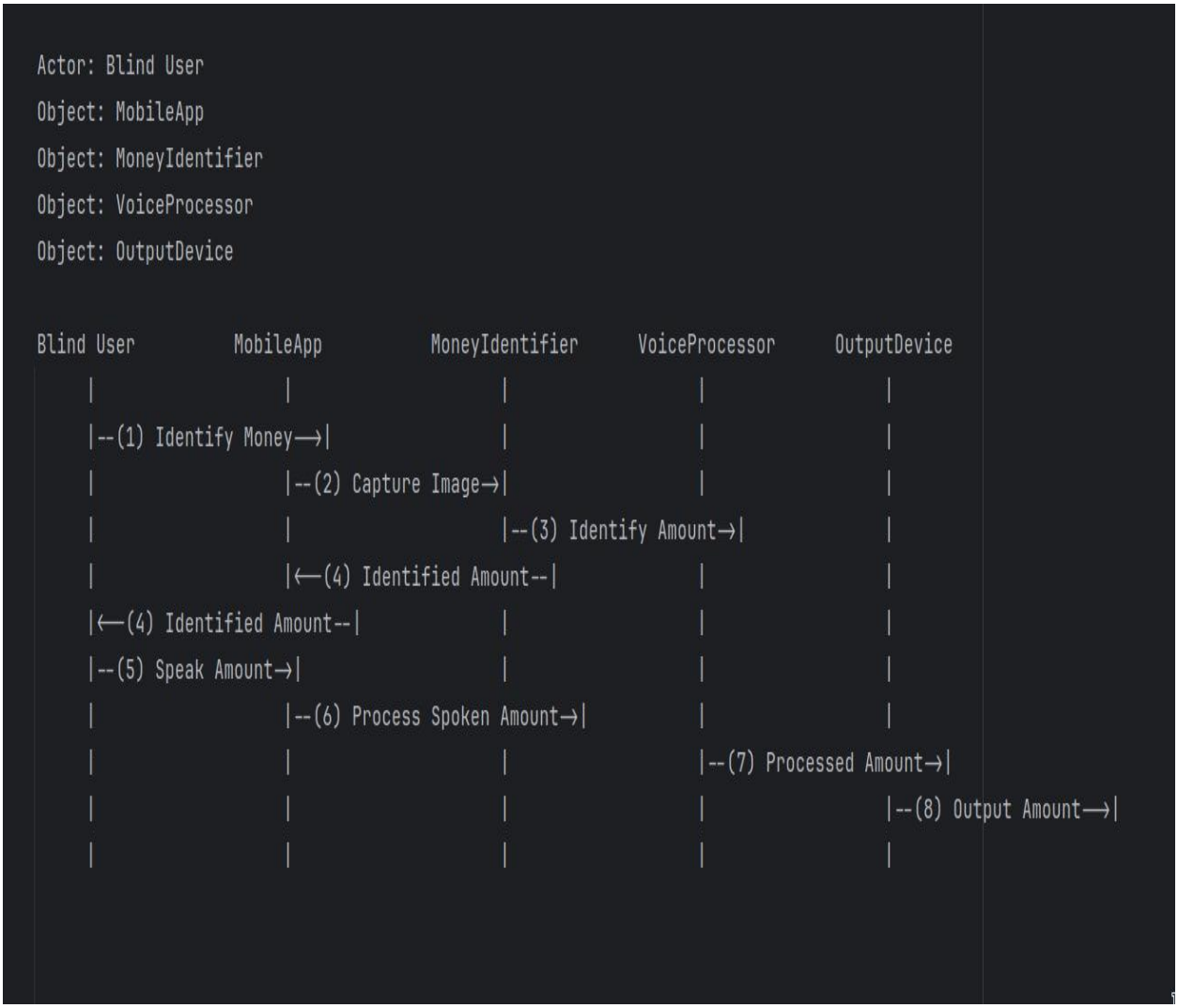
6.2 UML Use Case Diagram



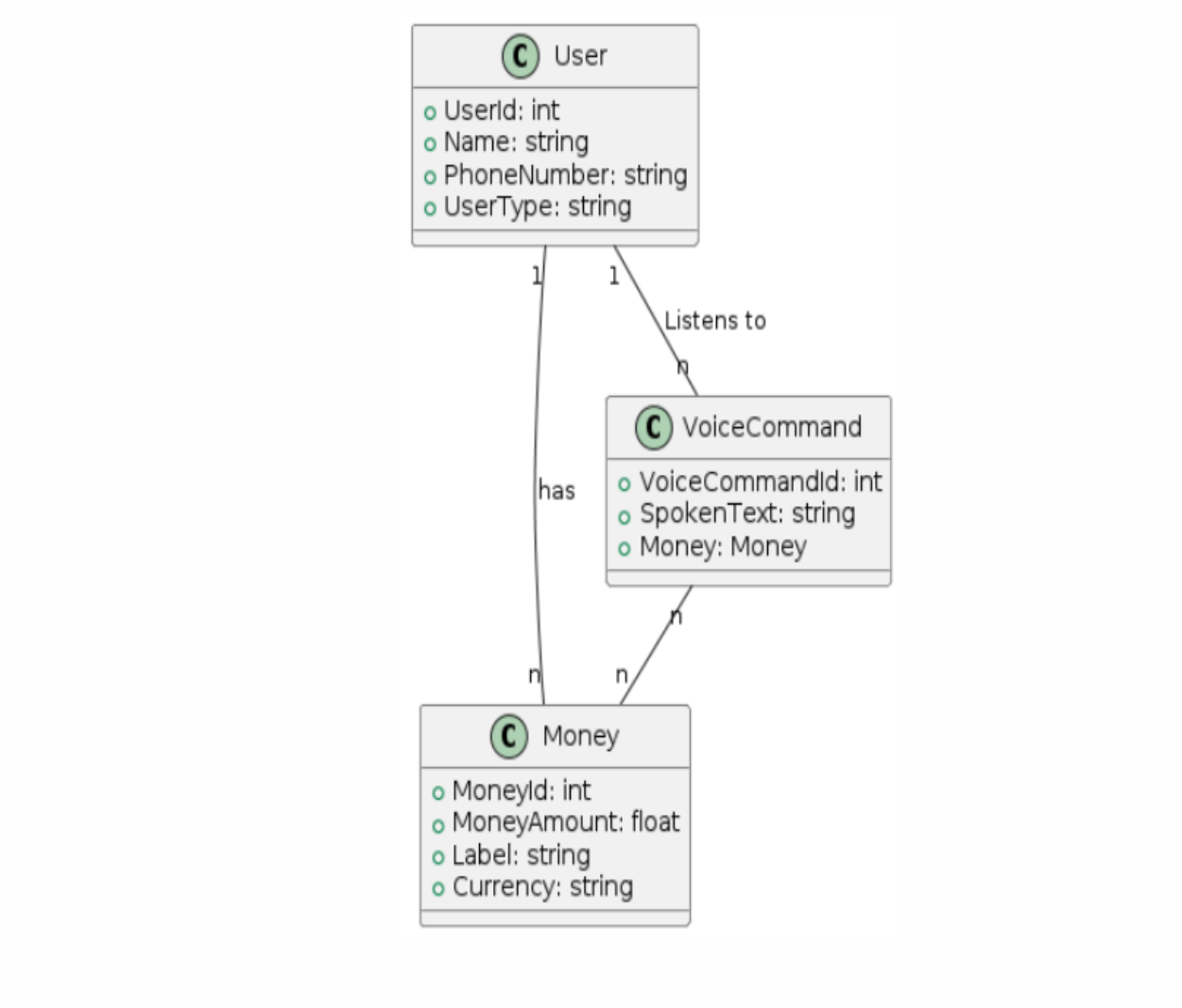
6.3 UML Activity Diagram



6.4 UML Sequence Diagram

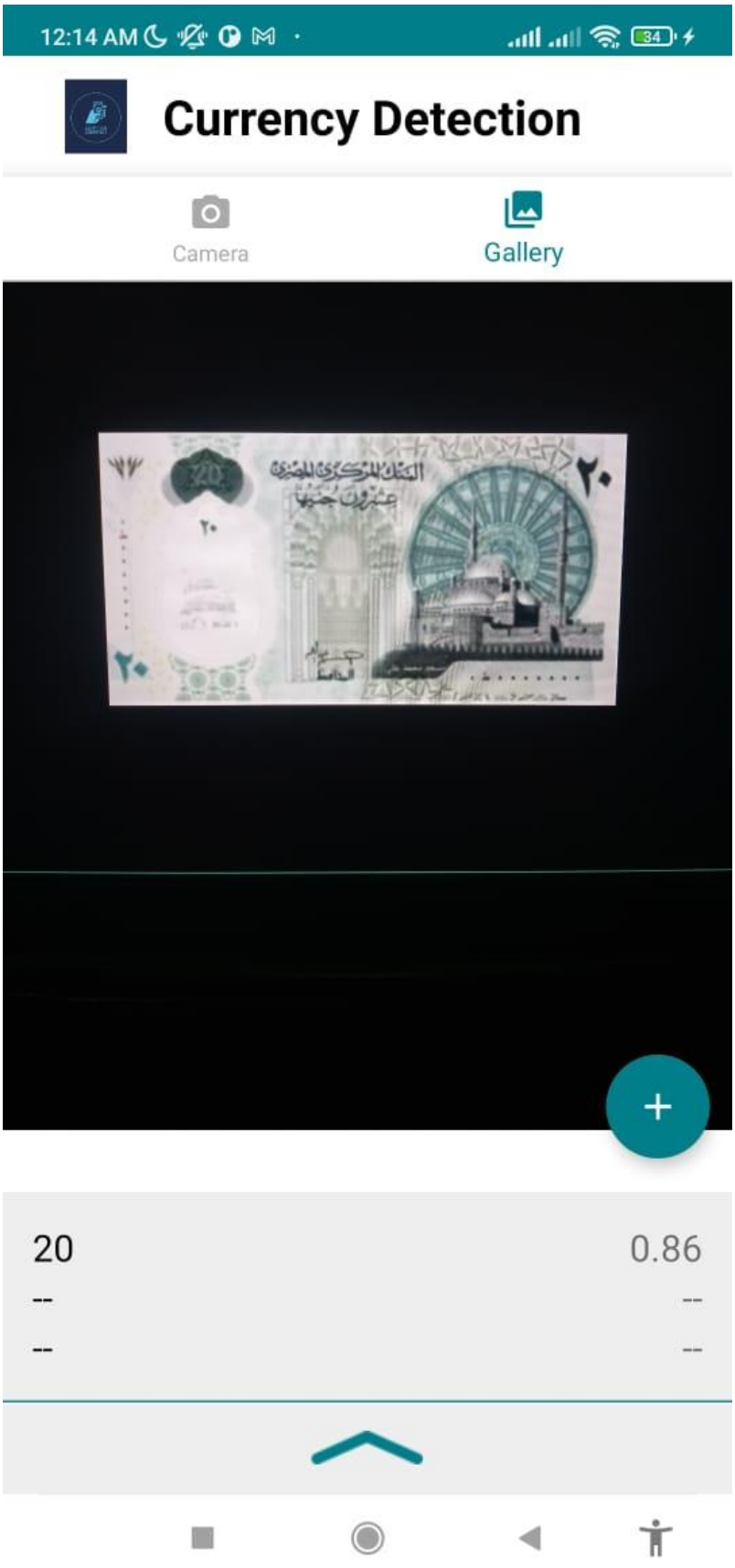


6.5 UML Class Diagram



6.6 Graphical User Interface (GUI) Design


12:09 AM
Currency Detection
Camera
Gallery
100
0.64
--
--
--
--


12:14 AM
Currency Detection
Camera
Gallery
20
0.86
--
--
--
--

6.7 Summary

This summary provides an overview of the various diagrams and designs required for both the high-level system overview and detailed object-oriented design of the application. Each diagram serves a specific purpose and together they ensure a well-rounded understanding and development of the system.

Summary of Design Requirements for the Application

6.2 Context Diagram

A context diagram provides a high-level overview of the system, depicting the system as a single process and showing its interactions with external entities (actors).

6.3 Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) illustrates how data flows through the system. It breaks down the system's processes, showing data sources, data storage, and how data moves between these components.

6.4 Entity Relationship Diagram (ERD)

An Entity Relationship Diagram (ERD) models the system's data entities, their attributes, and the relationships between them. This helps in understanding the database structure.

Object-Oriented Design Requirements

6.2 UML Use Case Diagram

A UML Use Case Diagram depicts the functional requirements of the system by illustrating the interactions between users (actors) and the system (use cases). It shows what the system does from the user's perspective.

6.3 UML Activity Diagram

A UML Activity Diagram represents the workflow or the business process within the system. It shows the sequence of activities, decision points, and parallel processes.

6.4 UML Sequence Diagram

A UML Sequence Diagram illustrates how objects interact with each other over time. It shows the sequence of messages exchanged between objects to carry out a specific functionality.

6.5 UML Class Diagram

A UML Class Diagram provides a detailed view of the system's structure. It shows the system's classes, their attributes, methods, and the relationships between the classes.

6.6 Graphical User Interface (GUI) Design

Graphical User Interface (GUI) Design outlines the visual and interactive aspects of the system. It includes the layout of screens, user input mechanisms, and the overall user experience design.

Chapter Seven: Conclusion

7.1. Summary of the Project

This project involved developing a mobile application for detecting and classifying Egyptian currency notes. We addressed a critical need for a reliable and accessible tool to assist visually impaired individuals and improve the accuracy of financial transactions. The application leverages advanced machine learning techniques, specifically convolutional neural networks (CNNs), integrated into a mobile-friendly framework. The project was divided into several phases, each meticulously planned and executed to ensure the success of the final product.

7.2. Achievements

Successful Integration of EfficientLite-2: We transitioned from MobileNet to EfficientLite-2, achieving higher accuracy and better performance suitable for mobile devices.

Robust Dataset Preparation: A comprehensive dataset was collected, preprocessed, and augmented to train the models effectively.

Model Training and Validation: The models were trained and validated rigorously, ensuring high accuracy in detecting and classifying different denominations of Egyptian banknotes.

User-Friendly Mobile Application: Developed using Kotlin in Android Studio, the app offers an intuitive interface and integrates seamlessly with MediaPipe for real-time image processing.

Enhanced Accessibility: The application includes features specifically designed to assist visually impaired users, such as auditory feedback and easy navigation.

7.3. Challenges and Solutions

Variation in Lighting and Angles: Ensuring accurate detection under varying lighting conditions and angles was challenging. We addressed this through extensive data augmentation and robust model training.

Wear and Tear of Banknotes: Handling worn-out and damaged banknotes required the model to be resilient. We included images of banknotes in different conditions to train the model effectively.

Real-Time Processing: Achieving real-time processing on mobile devices was crucial. By optimizing the models and integrating with MediaPipe, we ensured efficient and quick detections.

7.4. Future Work

Expanding Currency Detection: Future iterations of the app could include detection for currencys and additional currencies, expanding its usability beyond Egyptian banknotes.

Enhanced User Feedback: Implementing more sophisticated feedback mechanisms, such as vibration patterns and customizable auditory feedback, could further improve accessibility.

Continuous Learning: Incorporating a feedback loop where the app learns from user interactions could help in refining the model's accuracy and adapting to new currency designs.

Security Enhancements: Adding features to detect counterfeit banknotes can further increase the app's utility and reliability**.**

7.5. Conclusion

The successful development of the Egyptian currency detection mobile app represents a significant step towards leveraging technology for social good. By combining advanced machine learning models with practical mobile application development, we have created a tool that not only aids visually impaired individuals but also enhances the accuracy of everyday financial transactions. This project showcases the potential of integrating AI and mobile technology to solve real-world problems, setting the stage for future innovations in this field.

CHAPTER EIGHT: References:

1. https://mediapipe-studio.webapps.google.com/home
2. https://www.kaggle.com/datasets
3. https://arxiv.org/pdf/2009.07409
4. https://kotlinlang.org/docs/kotlin-reference.pdf
5. https://developer.android.com/
6. https://colab.google/
7. https://www.tensorflow.org/tutorials/images/classification?hl=ar
8. https://paperswithcode.com/task/image-classification
9. https://byjus.com/maths/data-sets
10. https://paperswithcode.com/method/efficientnetv2
11. https://www.analyticsvidhya.com/
12. remove.bg/upload
13. https://www.camscanner.com/
14. https://ai.google.dev/edge/mediapipe/solutions/vision/image_classifier
15. https://www.tensorflow.org/lite