Student 1: Liel Biton 207000241
Student 2: Coral Engel 208455659

Software Design Description (SDD) - PlantKeeper App

1. Introduction

Purpose:
The purpose of "Plant Keeper" is to assist plant enthusiasts in managing their plants effectively by offering tools for plant identification, personalized care instructions, and progress tracking. It aims to simplify plant care for users while fostering a stronger connection between individuals and their plants.

Overview:
This document outlines the system's detailed design, focusing on components, interactions, data flow, and design considerations. It serves as a guide for developers and stakeholders to understand the software's architecture and implementation.

2. System Overview

"Plant Keeper" operates as a comprehensive plant care assistant that integrates advanced AI-powered plant recognition, real-time notifications, and growth tracking capabilities. The system consists of three main layers: a user-friendly mobile app interface, a robust backend powered by Firebase for data management, and an AI recognition model for plant identification. Users can upload plant images to receive instant identification results along with care guidelines. The app also helps users build a personal plant collection, set care reminders, and visualize their plants' growth over time through side-by-side photo comparisons.

The core focus is to empower users to proactively care for their plants, ensuring their health and longevity.

Key Components:

1. **Mobile App:** The mobile app serves as the user interface for all interactions. Users can upload plant images for identification, manage their plant collections, and receive notifications for care schedules.
2. **Server Side:** Backend for processing requests, managing data, and interfacing with APIs.
3. **Recognition Model API:** This AI-powered API analyzes user-uploaded images to identify plant species. It retrieves care instructions from the database and provides fallback options for manual plant entry if the image cannot be identified.
4. **Database:** Firebase is used to store textual data: user accounts, plant collections, care schedules, and plant journals.
5. **Cloud Storage:**
   Firebase Cloud Storage manages high-resolution images uploaded by users and training datasets for the recognition model.

Interaction: Users interact with the mobile app to upload plant images, view identification results, and manage plant collections. The app communicates with the backend, which processes requests and interfaces with the recognition API. Identified plant data and user collections are stored in the database, while images are handled in the cloud storage.

3. Design Considerations

The design of "Plant Keeper" prioritizes user experience and performance. Firebase is chosen for its real-time data capabilities, ensuring seamless synchronization between users' devices and the cloud.

Assumptions:

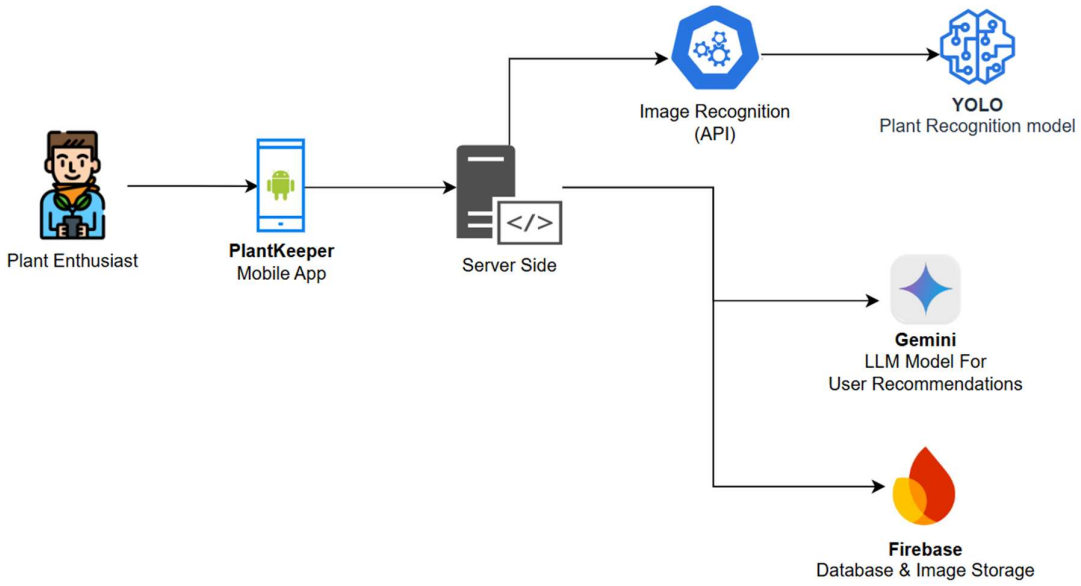
- Users have stable internet access.
- Plant recognition model is trained with common plant species.
- Users will access the app via Android devices running version 8.0 or higher.

Constraints:

- Initial scalability is limited to 50 users with 10 plants each.
- Prototype image storage capped at 15 GB.
- Training images for the AI model will be 720px or higher.

4. System Architecture

"Plant Keeper" employs a client-server architecture integrated with Firebase. The architecture ensures real-time data updates and seamless interactions between the mobile app and backend services.



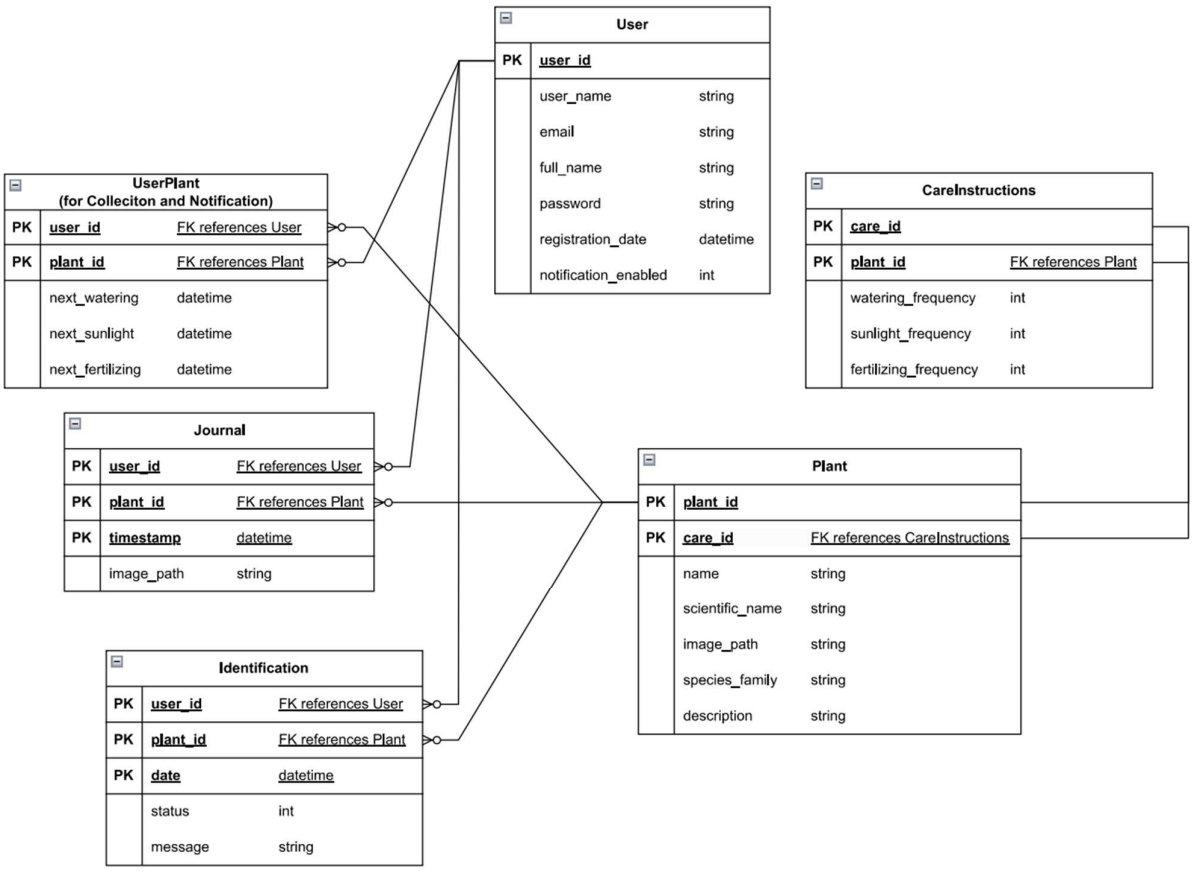
Technological Requirements:

- Development Environment: Android Studio.
- Firebase integration for database, notifications, and geolocation.
- Programming Languages: Kotlin for app development, Python for backend algorithms.
- Database Technology: Firebase Realtime Database for storing user data and plant information.
- Frameworks and Tools: YOLO model for image recognition, Firebase storage.
- Deployment via Google Play Store.

5. Data Design

This section outlines the **textual** and **image** data storage requirements for the *PlantKeeper* app. Key entities include users, plants, user-plant relationships, plant journals, and associated images. The database schema is structured to support efficient storage and retrieval of both textual data and image metadata.

Database Schema



Key Entities:

1. **User**: Stores user account details such as username, email, and password.
2. **Plant**: Contains information about plant types, care instructions, and attributes like watering frequency and sunlight needs.
3. **UserPlant**: Manages relationships between users and their plants, including personalized care schedules and notification dates.
4. **Journal**: Logs user-uploaded photos and descriptions of plant growth for tracking over time.
5. **CareInstructions**: Represents an extension of the **Plant** entity, containing all relevant care details
6. **Identification**: Records of identification trials.

Image Data Storage

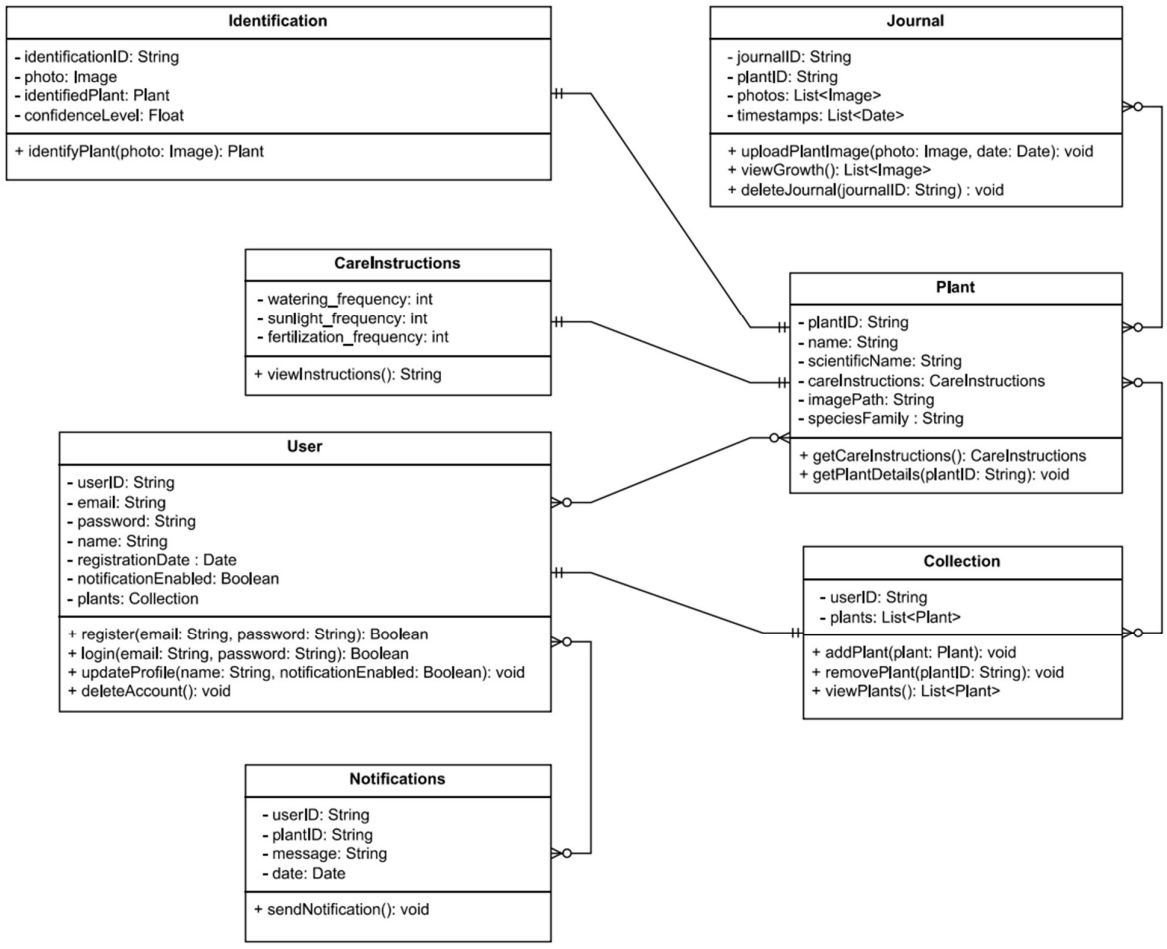
All images will be securely stored in Google Cloud using Firebase Storage, ensuring efficient retrieval. Types of images stored:

1. Plant Growth Journal Images - These images are uploaded by users to document and track the growth of their plants over time.
2. Plant Identification and Recommendation Images - These are **formal images** of plants that serve as references within the app.s

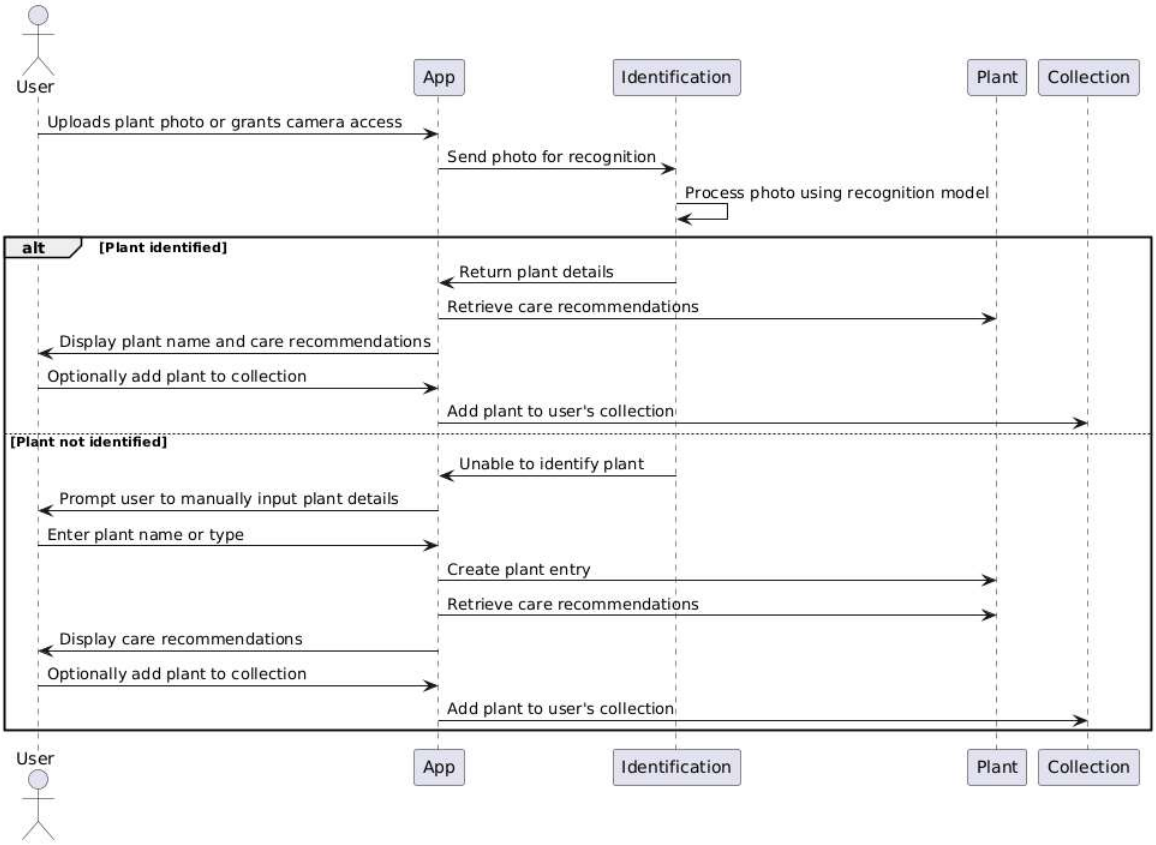
Data Flow for Plant Identification:

1. The user uploads an image through the app (from the camera or gallery).
2. The image is sent to the AI recognition model and it identifies the plant.
3. The AI model links the image to an entry in the **Plant** table.
4. **Database Interaction**:
    - Based on the identified plant, care details are pulled from the **CareInstructions** table.
    - Users can save the identified plant to their collection, which updates the **UserPlant** table.
5. Care instructions are displayed to the user.

6. Detailed Class Design (UML)



Sequence diagram - “Add new plant” Use-Case:



7. Functions and Optimizations

Performance Considerations:

- Plant recognition completes within five seconds for a smooth user experience.
- Firebase’s real-time database ensures instant updates to user data and app entities.

Scalability:

- The future product will include a microservices architecture to handle a growing user base.
- Firebase Cloud Storage supports increased storage needs as more users and images are added.

8. Algorithms:

PlantKeeper uses two different kinds of AI Models:

- Image Recognition Model
  For the Plant Recognition task, we chose to work with YOLO architecture, since this pre-trained model is efficient with resources and replay fast answers when identifying trained objects.
- LLM model
  For the Plant Recommendation task, we chose to use a trained LLM model which is free and accessible. The one that suited us the most was Google's model - Gemini.