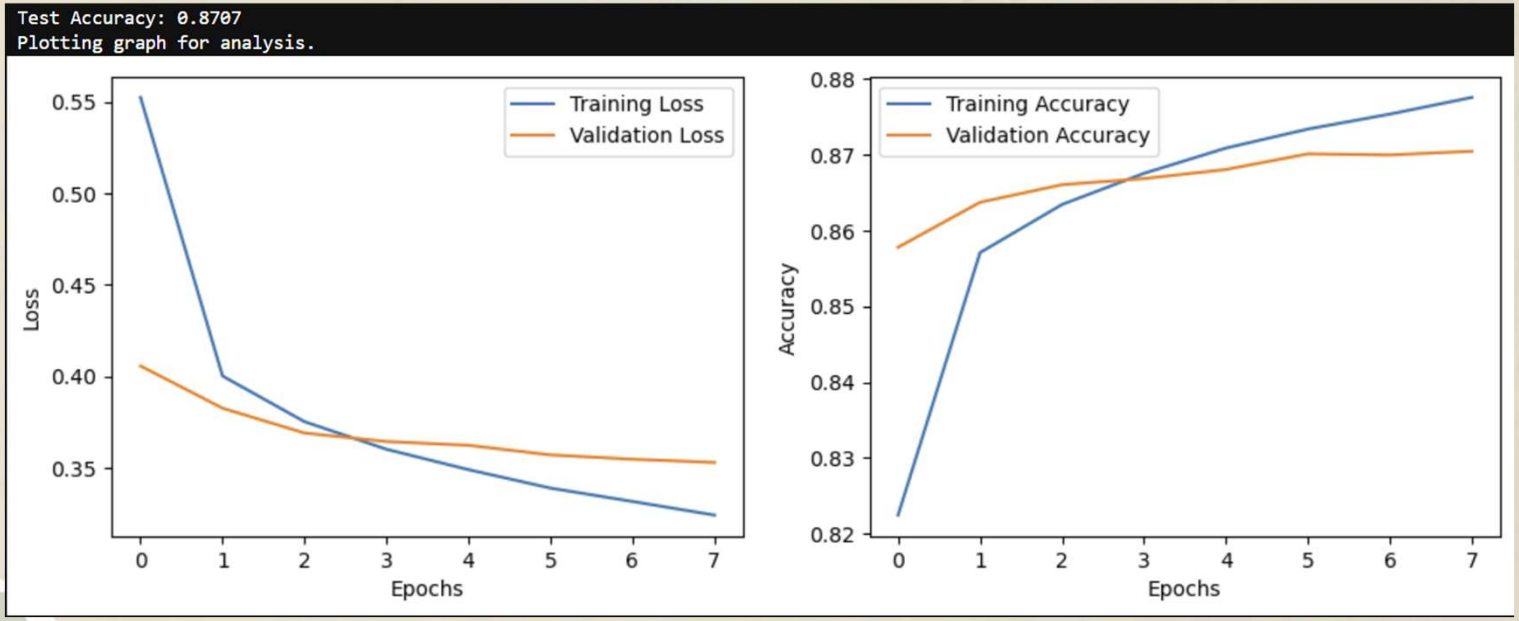
IPD – interim
Progression
Demonstration

VOXSCRIBE – A HANDWRITING RECOGNITION APP

STUDENT ID : W1947458
NAME : ABHINAVA SAI BUGUDI
SUPERVISOR : Dr. Dimitris Dracopoulos
Date : 6/02/2025

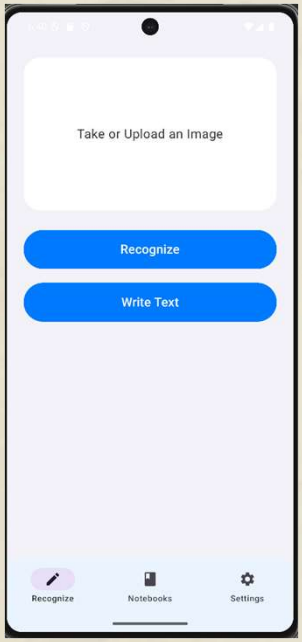
Project Progress & Achievement

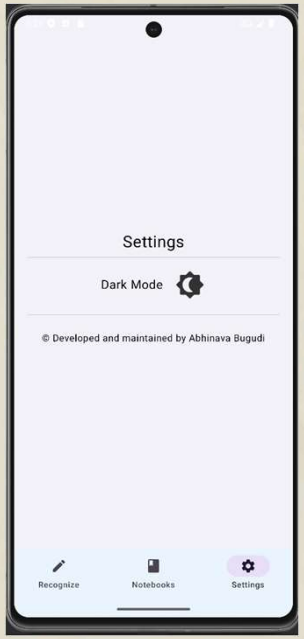
- Review of Achievements:
  - Successfully implemented **CNN-based handwriting recognition, i**mproved **model accuracy** (from ~85% to aiming for 90%+).
  - Developed a functional Jetpack Compose UI for an intuitive experience with basic notebook functionality, allowing users to save and view converted text.



Adaptations Made

- STORAGE :
- By eliminating reliance on the cloud, local storage for notebooks guarantees improved privacy, quicker access, and less complexity.
- Notes can be safely stored, arranged, and retrieved with ease by using Android's internal storage (Jetpack datastore).
- UI :
- Built a navigation system with three tabs for a smooth user experience.





Problems Overcame

Problem 01

Cluttered UI

Result : **Minimal UI,** with appealing **colours** and **animations**

Problem 02

Struggled Accuracy

Result : **Accuracy improved from 85% → 88%.**

Problem 03

Storing Data

Quicker access, **offline** support, and **simpler** implementation**.**

Upcoming Developments

Avg of 131 ms

Real-Time Handwriting Recognition

Advanced notebook features

Add, formatting option and the ability to export it into pdf, word and txt files

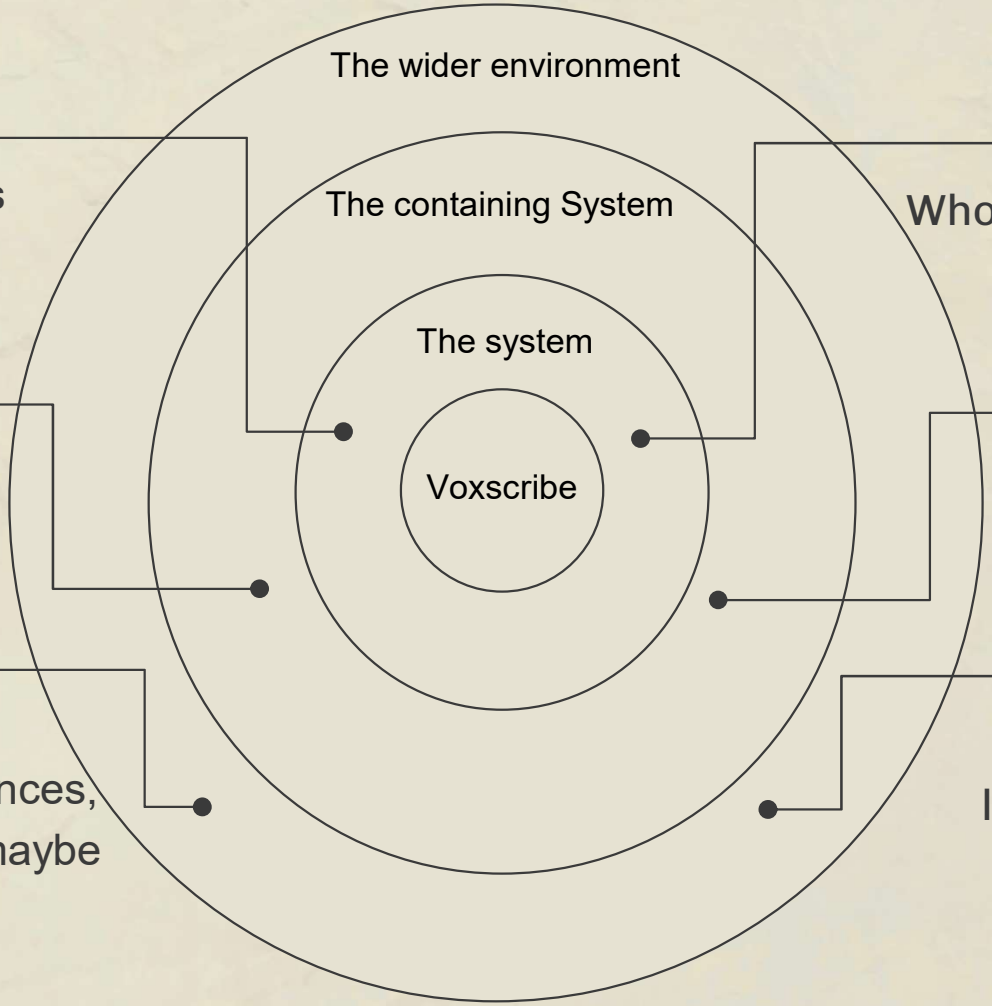
88% Accuracy

Improve **accuracy to 90%** or above

Offline Mode

Making sure user can access notebooks while offline

Identification Of Project Stakeholders



Primary Users
Students, researchers and Office Workers

Supervisor
offers academic advice and technological evaluation.

Advertising Agencies
advertising to specific audiences, boosting user uptake, and maybe generating revenue through collaborations

Developer
Who is reponsible for developing the UI, Model and the API

University
Provides a potential User base and supports the development

Competitors
Negative stakeholders like Google Keep, Apple Notes and Good notes

Functional Requirements

Implemented

01 The model recognises text 70% in time for testing, with a reported **accuracy of 89%**

02 Users can **create, edit, and delete notebooks** to organize recognized text.

03 Users can **add** and **edit** pages within notebooks dynamically

04 Notebooks and note are securely stored with **jetpack Datastore**, and users can toggle between **light and dark** mode

Pending

01 Connecting the model with the App using **REST API**

02 Implementing a **whiteboard based drawing** input for recognition

03 Enabling **real-time text conversion** through seamless model communication.

04 Adding **search and export** functionality for accessiblity and sharing

Non Functional Requirements

Implemented

01 **Simple** and **minimalist UI**

02 **Smooth animations**

03 **Lightweight,** easy to use and **easy to run**

04 **Offline** Functionality

Pending

01 Performance **optimization** and **improve recognition speed**

02 **Larger font options**, **voice input** (optional)

03 Handling **low quality inputs**

04 **Real time** autocorrection

SYSTEM ARCHITECTURE

01

User Interaction

Users upload/take an image or write notes.

02

Front End

Kotlin and Jetpack Compose

03

Back End

Rest API, Python (Machine learning), Datastore

04

Machine Learning Model

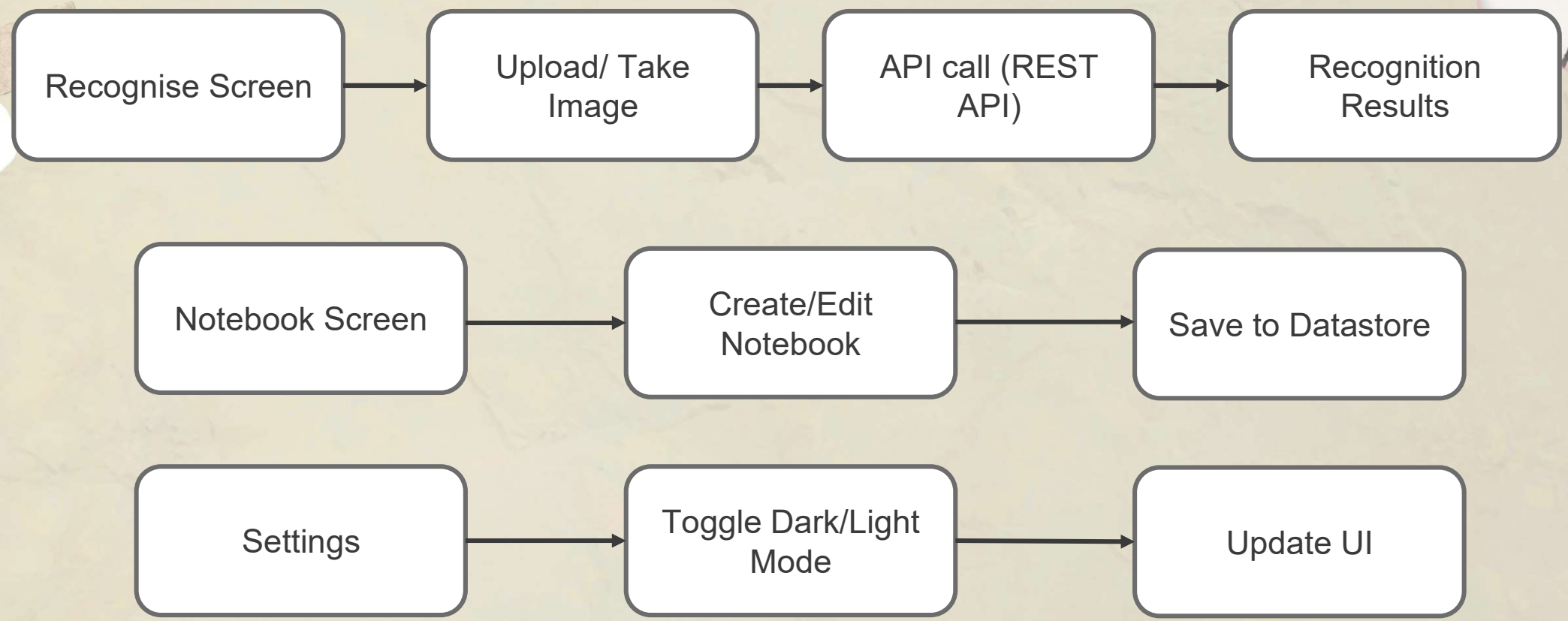
CNN based, trained with Tensorflow

05

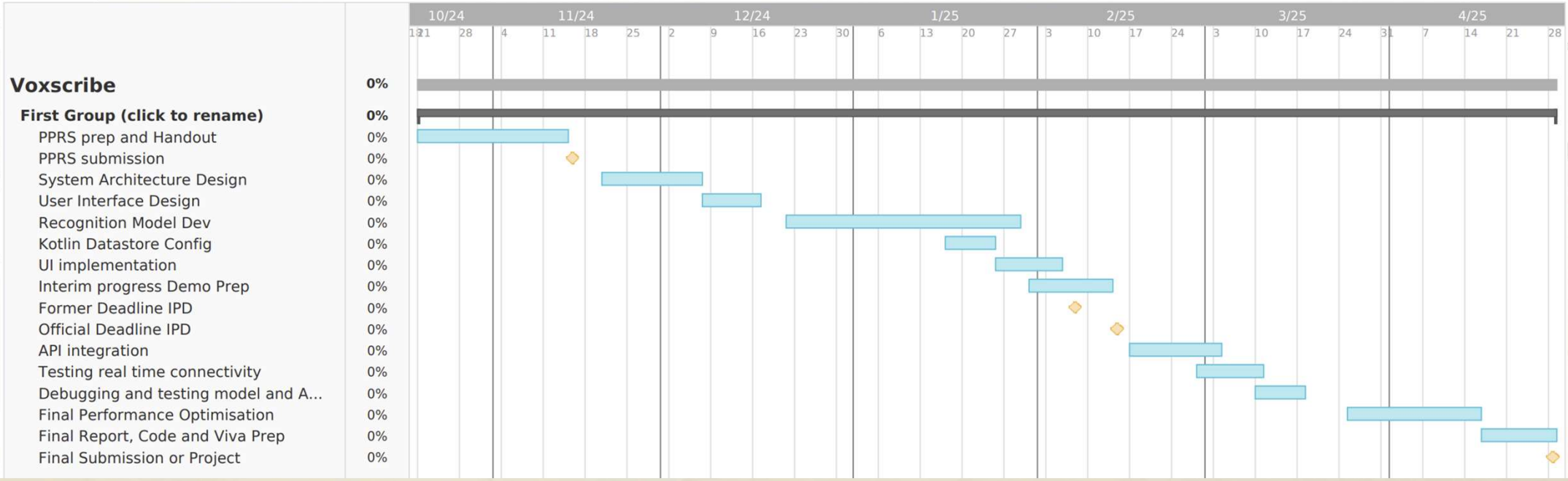
Integration

REST API for real time updates

DETAILED SYSTEM ARCHITECTURE



GANTT CHART



- **Data Persistence Completion** – The Kotlin DataStore configuration is now fully implemented, allowing a shift in focus towards API integration.
- **Extended API Integration Phase** – More time is allocated to ensure seamless communication between the app and the handwriting recognition model.
- **UI Refinements & Performance Optimization Moved Later** – Additional UI improvements and performance optimizations are planned closer to submission for a polished final product.
- **Buffer Time for Debugging & Final Testing** – Extra time is set aside to thoroughly test real-time connectivity and model accuracy before the final submission.

CONCLUSION

COMPLETED	ONGOING	YET TO BE DONE
Requirements Gathering	**Final UI polishing**	**API Integration**
UI design and Implementation	**Performance optimization**	**Real-Time Connectivity testing**
Kotlin Datastore	**Debugging and Refinements**	**Final Documentation**
Initial Model Development	**Improve Model Development**	**Submission Report**