SummarAIze - Milestone 3 Final Report

NUS Orbital 2025
Team Name: SummarAIze
Level of Achievement: Apollo 11

Foreword

Greetings Orbital Evaluators, Advisors, and fellow Orbitees.

This document presents the full technical report and reflective documentation for SummarAIze, an AI-powered learning aid designed to assist students with revision through summarization, mind mapping, and test simulation features.

As Orbital 2025 participants, we set out to solve the challenge faced by many Singaporean students: navigating a mountain of revision content before exams. We realized that existing tools—while helpful—rarely integrated summarization, mind maps, and test prep into a cohesive platform. This inspired us to develop SummarAIze.

In this README, you'll find detailed insights into our motivation, design decisions, technical architecture, challenges faced, and the milestones we've achieved. Each major feature is explored in depth, including user authentication, AI integration, OCR processing, Firebase database design, and interactive UI logic. We have also included unit, integration, and user testing results, as well as comprehensive diagrams.

This project has been a transformative learning experience, introducing us to full-stack software engineering, CI/CD, and collaborative agile development. We hope this documentation serves as a clear, comprehensive record of what we've built and why.

Posters

Milestone 1 Poster: [Insert Image Placeholder]

Milestone 2 Poster: [Insert Image Placeholder]

Milestone 3 Poster: [Insert Image Placeholder]

Proof-of-Concept

Demo Video: [Insert YouTube/Drive Link Placeholder]
Test Account:
Email: test@summarAIze.app
Password: 12345678

Deployment

SummarAIze is deployed via [Insert Hosting Platform]. The frontend is served as a React SPA, with the backend hosted via FastAPI and Firebase functions.

Proposed Level of Achievement

Apollo 11

Aim

To streamline student revision by providing a single platform for summarization, mind mapping, and exam simulation using AI.

Motivation

Singaporean students face a major burden in revision due to fragmented tools and overwhelming materials. SummarAIze combines summarization, mind maps, and test practice into one cohesive interface.

Vision

SummarAIze will become the default revision companion for students preparing for exams, reducing friction in content digestion and encouraging meaningful, adaptive learning.

User Stories

Emma (University student)...
James (High schooler)...
Sophia (Medical student)...

Features

1. AI-Powered Summarization
2. Interactive Mind Map
3. Test Mode
4. Export/Storage Options
5. Summary Customization

AI-Powered Summarization

Description:
SummarAIze allows students to upload study materials in PDF, DOCX, or image format (OCR-supported) and choose from three summarization formats: bullet points, short paragraphs, or long paragraphs. The AI (OpenAI GPT) processes the text, extracts key points, and highlights important content.

Implementation Philosophy:
We designed the summarizer for speed, clarity, and adaptability. After file upload, content is passed through preprocessing (text extraction, cleaning, deduplication), then summarized using system prompts customized to summary length and tone.

Implementation Challenges:
- Prompt tuning to produce structured output
- Handling noisy OCR text from low-resolution images
- Ensuring output is streamable in chunks to avoid long wait times for users

Mind Map Generator

Description:
The mind map generator takes AI-extracted key concepts and visualizes them as an editable mind map.

Implementation Philosophy:
We convert hierarchical key points into a graph structure using a recursive tree walker to determine parent-child relationships. The initial visualization is rendered with a JS library (e.g. react-flow), and users can drag, rename, and add/delete nodes interactively.

Implementation Challenges:
- Mapping linear AI output to a tree/graph
- Preventing overlapping nodes in UI
- Syncing real-time edits back to Firestore efficiently

Test Mode

Description:
Users upload test papers and manually input questions. Supports MCQ, open-ended, and 'other' formats. Simulates a test environment with timer, feedback, and attempt history.

Implementation Philosophy:
We built a flexible question model supporting custom question types and tied submissions to a Firebase collection with grading logic embedded in the frontend and backend.

Implementation Challenges:
- Parsing PDF content for usable formatting
- Handling grading for free-text answers
- Accurately tracking and saving time/attempt data for each question set

Customizable Summary Preferences

Users can select summary detail level and toggle between simplified or technical vocabulary modes. This is reflected in prompt design and preview mode toggles.

Challenges:
- Mapping user intent to AI prompt variants
- Allowing re-generation without losing user progress

Export and Storage

Exported formats include PDF, DOCX, and TXT. html2pdf.js and python-docx are used on frontend/backend respectively.

Challenges:
- Generating consistent layout across formats
- Avoiding XSS when sanitizing exported HTML