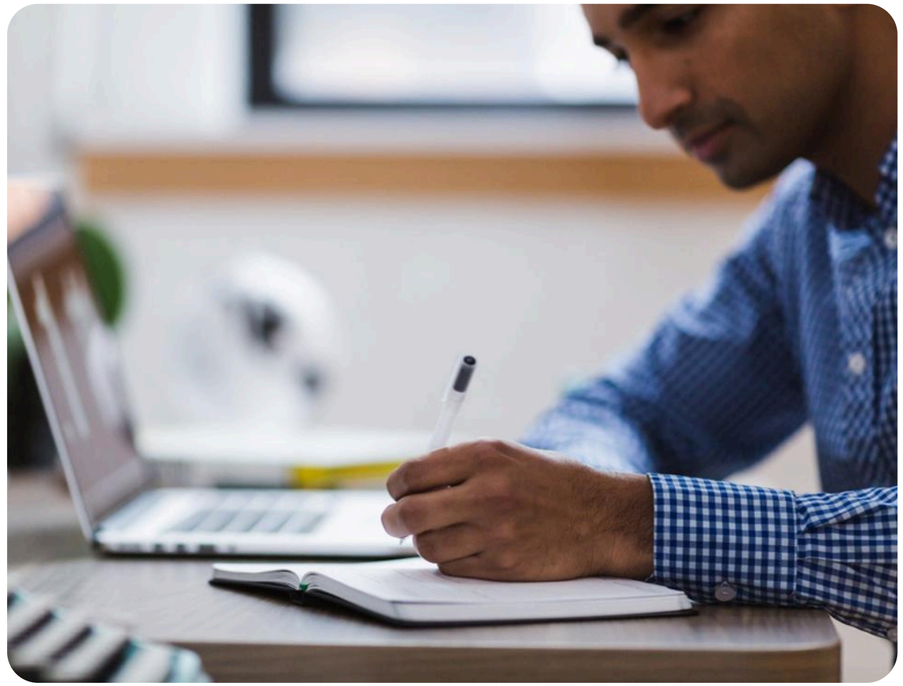
2025/05/28

CSE416: Final Presentation

SummarAIze

MINHYEOK IM & SEMIN BAE



Overview

01 Project Description

- WHAT IS SUMMARAIZE?
- WHO ARE THE USERS?
- EXISTING ALTERNATIVES
- SCOPE

02 Requirements

- FUNCTIONAL REQUIREMENTS
- NON-FUNCTIONAL REQUIREMENTS
- USES CASE

03 Design

- USER INTERFACE (EARLY SKETCHES, HIGH FIDELITY)
- DATA DESIGN

04 Implementation

- SW ARCHITECTURES
- FRONT/BACK-END ARCHITECTURES
- CODE CONVENTION
- GENTT CHART
- MILESTONE 1~4 & UPDATED PROGRESS
- MAIN ROLE

05 Takeaways

- WHAT WE’VE GOT FROM THIS PROJECT?

01: Project Description

What is the SummarAIze?

SummarAIze

- AI-powered platform
- Efficiently Read, Summarize, Manage Academic Paper
- Generating Concise Summaries with Page References
- Extracting Citation in Various Academic Styles
- Offering Personal Library
- Make Save Time, Academic Resarch more Accessible and Managable for Academic Papers

01: Project Description

Who are the Users?

1

Undergraduate Student

- Undergraduate are new to reading Academic Paper
- Don’t have much of Time
- SummarAIze helps by Creating Short, Clear summaries with page number, understand quickly
- Saving Time read easier

2

Graduate student & Researchers

- Have to read lots of papers
- Need to Find Useful Information
- SummarAIze Offers smart Summaries, Personal Library
- Pulls out Citation

3

Non-Native Language Users

- Most of Papers are based on English
- When doesn’t match the user’s native Language, It lead difficulties in understanding
- Make better Accessibility for Non-Native language

01: Project Description

Scope

1 What the product will do?

- Process and analyze academic PDF documents (up to ~30 pages)
- Generate concise summaries highlighting important information
- Attempt to provide page references for key information when possible
- Extract and format basic citations in common academic styles (MLA and APA)
- Provide basic paper recommendations based on keywords
- Store and organize summarized papers in a personal library

2 What the product will Not do?

- Replace comprehensive reading for critical academic analysis
- Write original content or essays based on papers
- Edit or modify original PDF documents
- Translate papers between languages
- Perform advanced semantic analysis of paper relationships
- Support real-time collaborative analysis
- Process extremely large papers (>50MB) due to API limitations

01: Project Description

Scope

3 Platforms and Device Support

- Web Applications Accessible via Modern Browsers (Chrome, Firefox)
- Primary focus on Desktop Experience
- No native mobile application

01: Project Description

Existing Alternatives

ResearchGate/Google Scholar

Strengths: Extensive paper database, citation tracking, author networks
Weaknesses: Limited summarization features, no personalized content extraction

Mendeley

Strengths: Excellent citation management, library organization
Weaknesses: No AI-powered summarization, limited content extraction

ChatGPT

Strengths: Can generate summaries when provided with paper content
Weaknesses: No direct PDF processing, requires manual input, no citation extraction

02: Requirements

Functional Requirements

User Authentication and Management

- Users shall be able to create accounts and log in securely through Google OAuth.
- Users shall be able to view and manage their account settings.
- Users shall be able to log out of the application.
- Users shall be able to change user’s profile image.
- Users shall be able to change account passwords.

Document Upload and Processing

- Users shall be able to upload PDF files through drag and drop.
- Users shall be able to upload PDF files through the system file explorer.
- The system shall process and extract text content from uploaded PDFs.
- The system shall analyze the document structure to identify key sections.

02: Requirements

Functional Requirements

AI-Powered Summarization

- The system shall generate concise summaries of uploaded research papers using OpenAI API.
- The system shall attempt to identify key information with page references when possible.
- The system shall support the user’s preference language.

Citation Management

- The system shall extract basic citation information from papers automatically if it assesses from PDF meta datas(title, authors, year, journal).
- The system shall generate citations in various common formats.

02: Requirements

Functional Requirements

Research Recommendation

- The system shall suggest related papers based on AI-powered matching systems.

Library Management

- Users shall be able to save summarized papers to their library.
- The system shall display a simple list of previously processed papers.

User Interface

- The interface shall be functional on desktop browsers.
- The interface shall display paper information and summary in a readable format.
- The interface shall have a consistent and user-friendly design.

02: Requirements

Non-Functional Requirement

Performance Requirements

- The system shall process and summarize a typical research paper (20-30 pages) within 2-3 minutes, as AI processing may take time.
- The web application shall load initial content within 5 seconds on standard broadband connections.
- performance degradation, appropriate for a university project scale.
- Search functionality within the user's library shall return results within 3 seconds.

Security Requirements

- All user data shall be encrypted both in transit and at rest.
- The system shall implement OAuth 2.0 for secure authentication.
- The system shall maintain separation between user accounts, ensuring users cannot access others' libraries.
- The system shall implement rate limiting to prevent abuse of the API.

02: Requirements

Non-Functional Requirement

Reliability Requirement

- The system shall function reliably during demonstration periods.
- The system shall implement basic error handling for common failure scenarios.
- If summarization fails, the system shall provide basic error messages.
- The system shall maintain user data throughout the course project.

Usability Requirement

- New users shall be able to upload and summarize their first paper within 5 minutes without requiring documentation.
- The interface shall be intuitive for typical college students.
- The system shall function correctly on Chrome and Firefox browsers.
- The system shall provide basic tooltips for the main features

Scalability Requirement

- The database shall support efficient storage and retrieval of at least 1,000 processed papers, suitable for demonstration purposes.
- The system architecture shall be designed with future scalability in mind, even if not immediately implemented.

02: Requirements

Uses Case

SignUp

Primary Actor: Customer
Priority: High
Scenario

1. The user enters registration details or uses Google OAuth.
2. The system validates and creates an account.
3. The user is redirected to the main dashboard.

Extension: If the email is already in use, the system notifies the user and prompts for another email.

Login

Primary Actor: Customer
Priority: High
Scenario

1. The user enters their login credentials or uses Google OAuth.
2. The system verifies the credentials and grants access to the platform.

Extension: If the credentials are incorrect, the system displays an error message and prompts for re-entry.

02: Requirements

Uses Case

Logout

Primary Actor: Customer

Priority: High

Scenario

1. The user clicks the "Logout" button.
2. The system logs out the user and redirects to the main page.

Extension: If the session expires, the system automatically logs out the user and prompts for re-login.

Drag & Drop - Summarization, Recommendation, and Citation

Primary Actor: Customer

Priority: High

Scenario

1. The user uploads a research paper by dragging and dropping it.
2. The system processes the file, extracts key information, and generates a summary.
3. The system displays extracted citations and recommendations for related papers.

Extension:

- If the uploaded file is not a valid PDF, the system displays an error message.
- Users are also able to upload the PDF file with the system file explorer.

02: Requirements

Uses Case

Multi-Langauage Summarization

Primary Actor: Customer

Priority: Medium

Scenario

1. The user selects a preferred language for summarization.
2. The system generates a summary in the selected language.
3. The summary is displayed in the preferred language.

Extension: If the language is not supported, the user can not select that language from the language list.

Profile Management

Primary Actor: Customer

Priority: Medium

Scenario

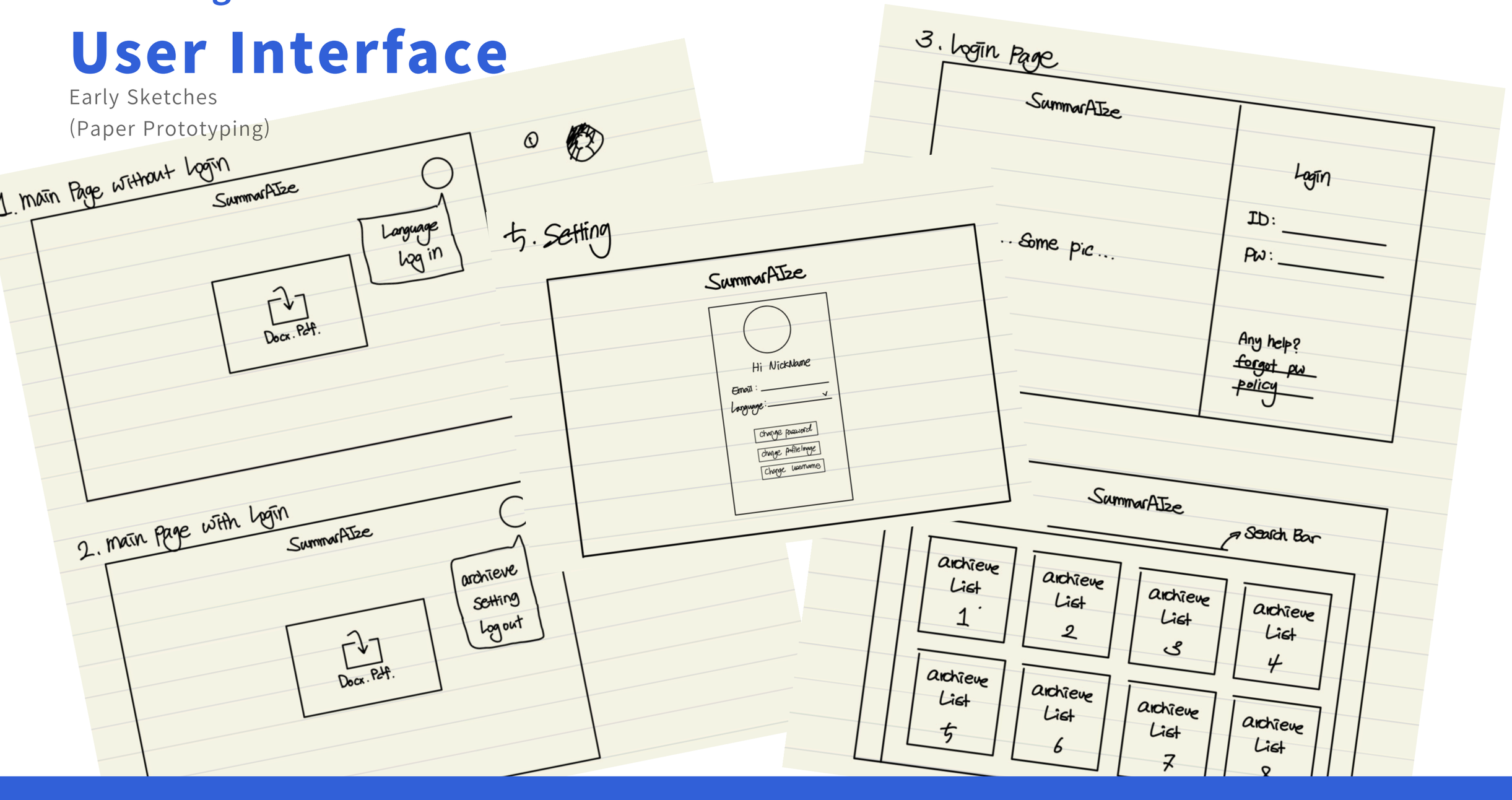
1. The user accesses the profile settings.
2. The user updates their profile picture, name, or password.
3. The system saves and updates the user information.

Extension: Users can not select the unsupported format picture file. If the password update fails (e.g., weak password), the system prompts the user to enter a stronger password.

03: Design

User Interface

Early Sketches
(Paper Prototyping)

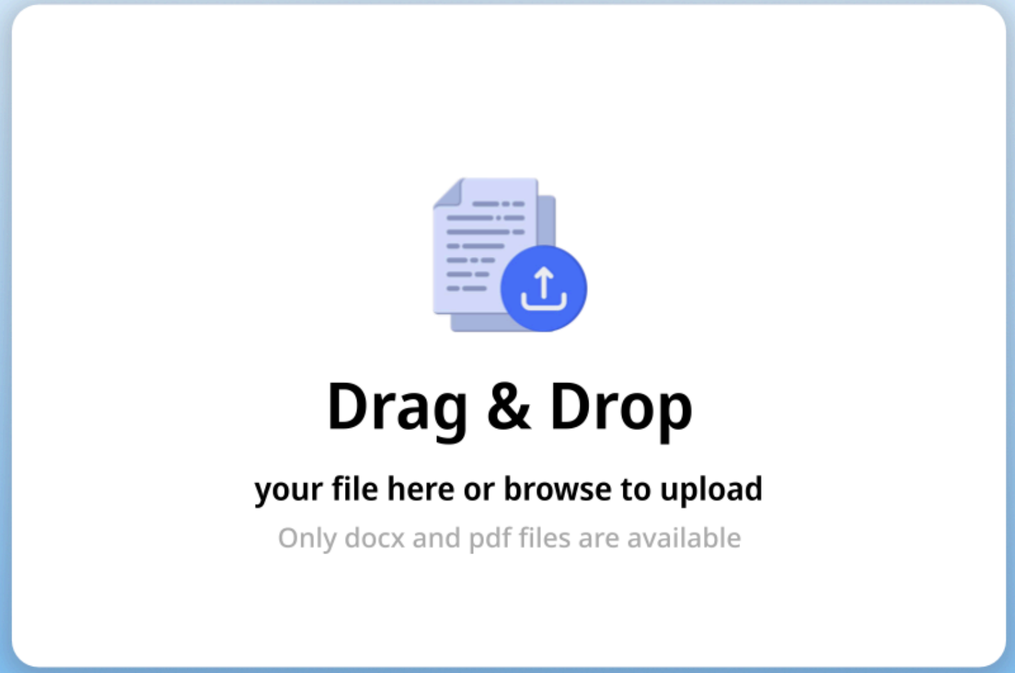


03: Design

User Interface

Higher Fidelity Mockups

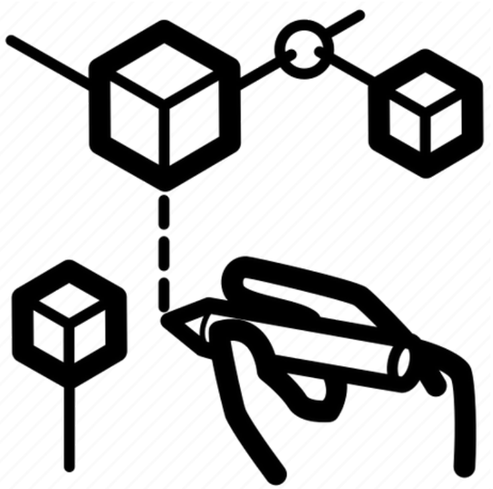
Link: Figma



03: Design

Data Design

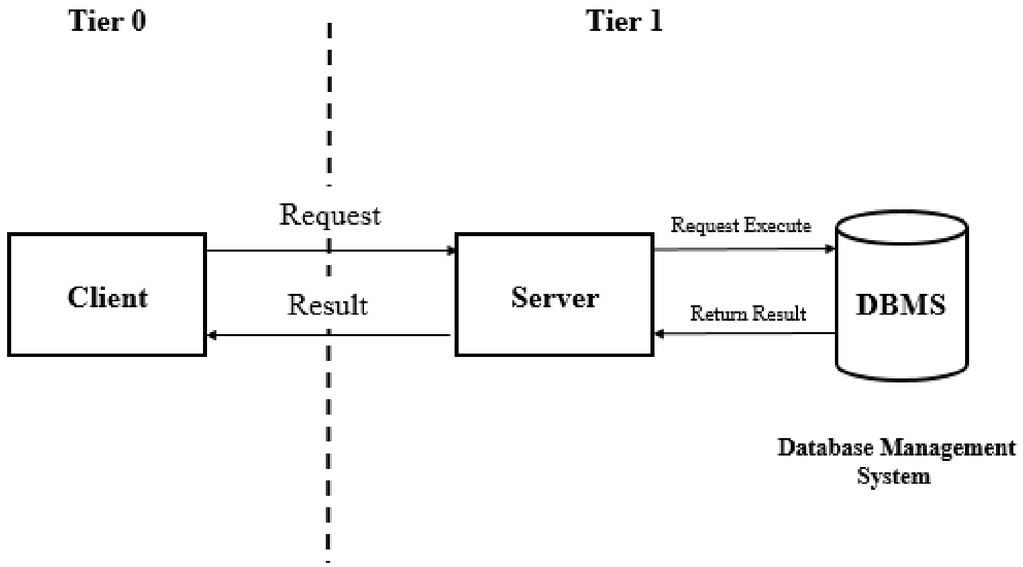
Link: Github



Name	HTTP Method	URL	Input	Output	Additional Notes
Sign Up	POST	`/api/auth/signup`	`json { "email": "string", "password": "string", "username": "string" }`	`json { "success": boolean, "message": "string", "user": User }`	Creates new account with default profile image
Login	POST	`/api/auth/login`	`json { "email": "string", "password": "string" }`	`json { "success": boolean, "message": "string", "user": User, "accessToken": "string", "refreshToken": "string", "expiresIn": number }`	Returns JWT tokens for session management
Google OAuth	GET	`/api/auth/google`	Query params from Google OAuth	Redirects to frontend with tokens	Handles Google OAuth callback
Refresh Token	POST	`/api/auth/refresh`	`json { "refreshToken": "string" }`	`json { "success": boolean, "message": "string", "user": User, "accessToken": "string", "expiresIn": number }`	Generates new access token

04: Implementation

SW Architectures



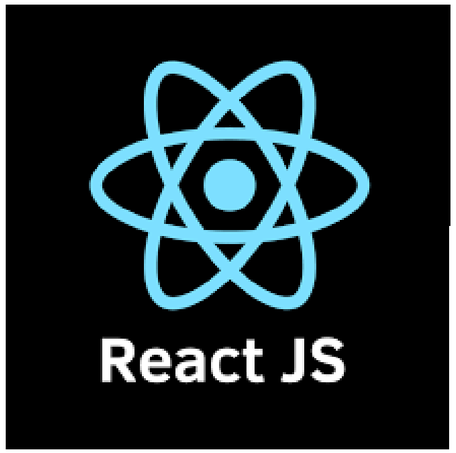
Client-Server Architecture

Case

- Client uploads PDF documents to the server for processing and storage
- Server validates and processes incoming PDF files from client requests
- Database stores PDF metadata and file references with indexing system
- Server returns processing results and status updates to the client

04: Implementation

Frontend Architectures



Google Pdf Viewer

- React (v19.0.0) - For building the user interface components
- React Router (v7.4.1) - For client-side routing and navigation
- Tailwind CSS (v3.4.17) - For styling and responsive design
- Google PDF(v2.x) - For rendering PDF documents in the browser
- React Markdown (v10.1.0) - For rendering markdown-formatted summaries

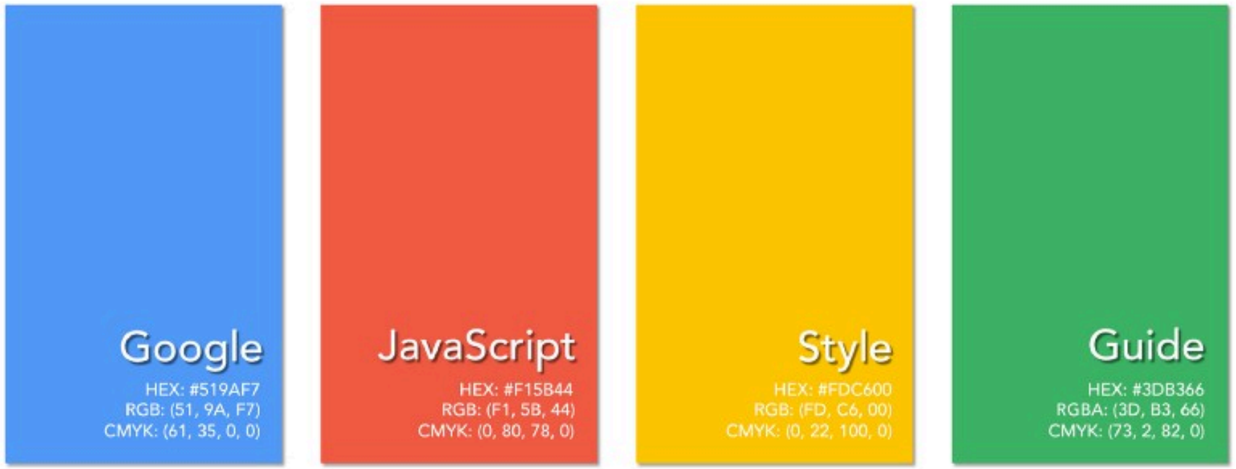
04: Implementation

Backend Architectures

- Node.js (v18. x): Server-side JavaScript runtime
- AWS Lambda: For serverless execution of PDF processing and AI summarization
- aws-sdk(v2.1531.0), aws-lambda-ws-server(v0.1.8)
- serverless-websockets-plugin(v1.0.0),
- socket.io(4.7.2): for the web socket connection with the client
- DynamoDB (NoSQL): For storing user data, paper metadata, summaries, and library management

04: Implementation

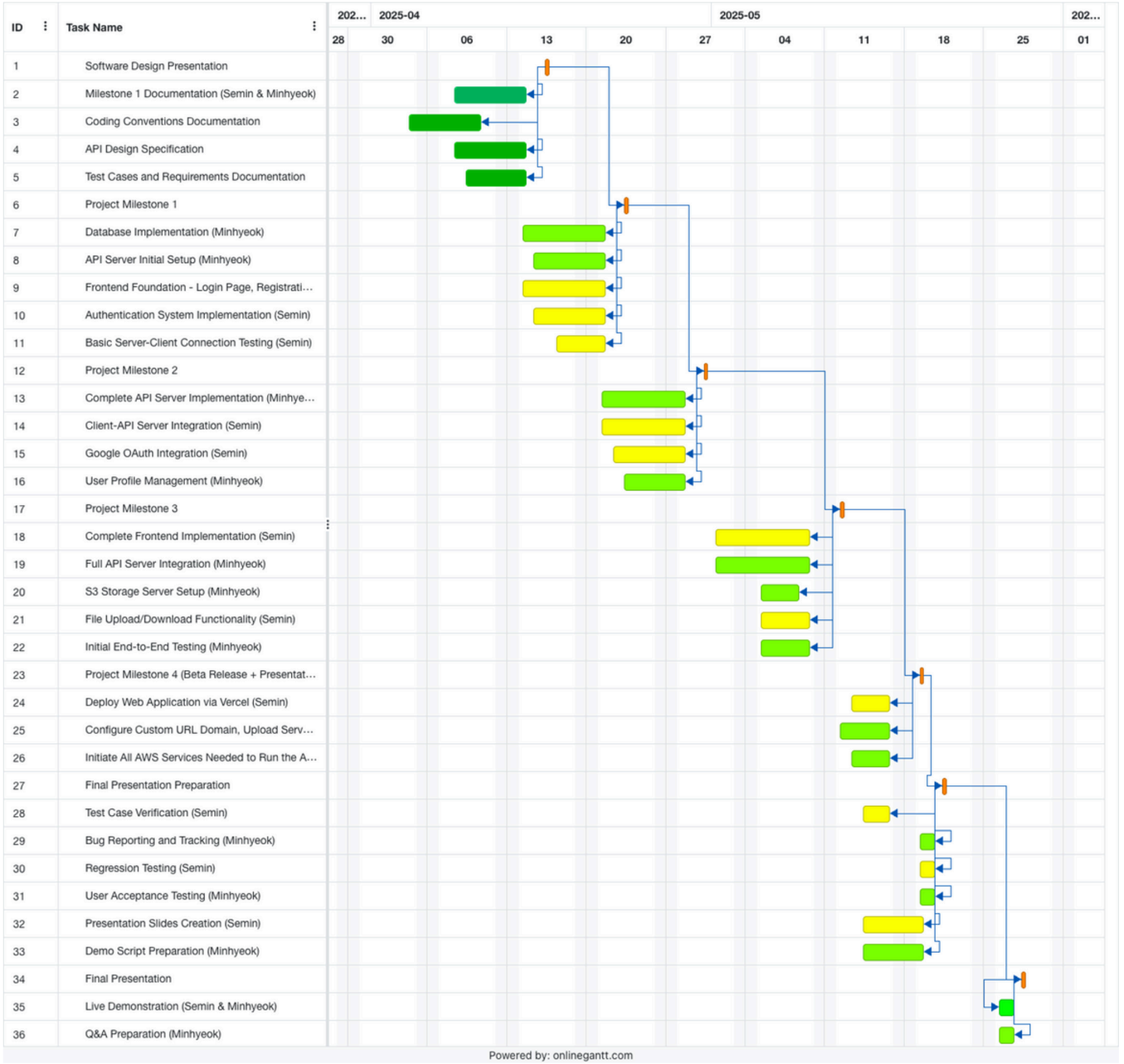
Code Convention



- JavaScript/React (Frontend): Airbnb JavaScript Style Guide, React-specific conventions
- Node.js (Backend): Google JavaScript Style Guide
- ESLint for static code analysis, JSDoc for code documentation, Prettier for code formatting
- Camel case for variables and functions, Pascal case for component names

04: Implementation

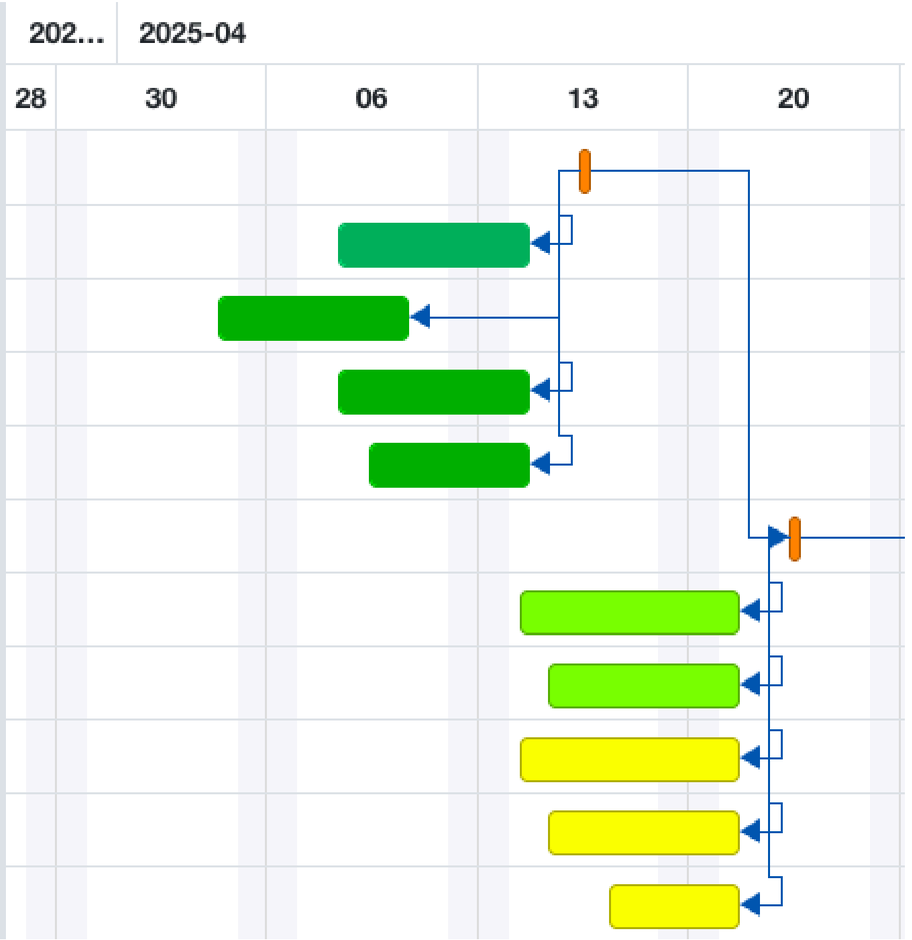
Gentt Chart



04: Implementation

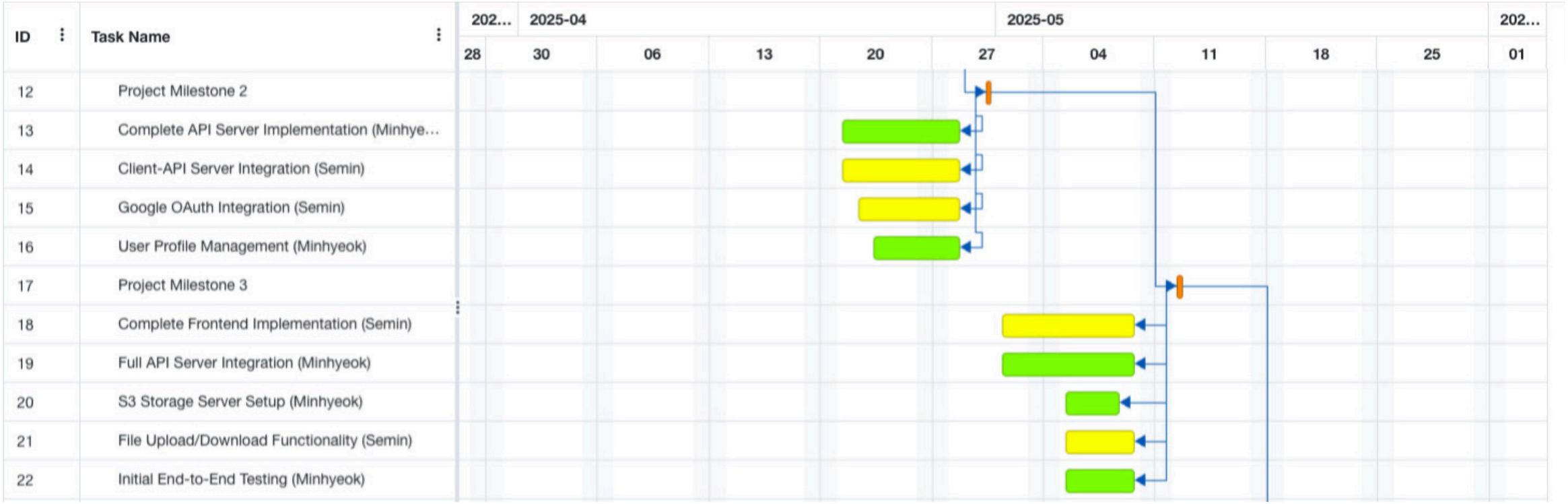
Gentt Chart

ID	Task Name
1	Software Design Presentation
2	Milestone 1 Documentation (Semin & Minhyeok)
3	Coding Conventions Documentation
4	API Design Specification
5	Test Cases and Requirements Documentation
6	Project Milestone 1
7	Database Implementation (Minhyeok)
8	API Server Initial Setup (Minhyeok)
9	Frontend Foundation - Login Page, Registrati…
10	Authentication System Implementation (Semin)
11	Basic Server-Client Connection Testing (Semin)



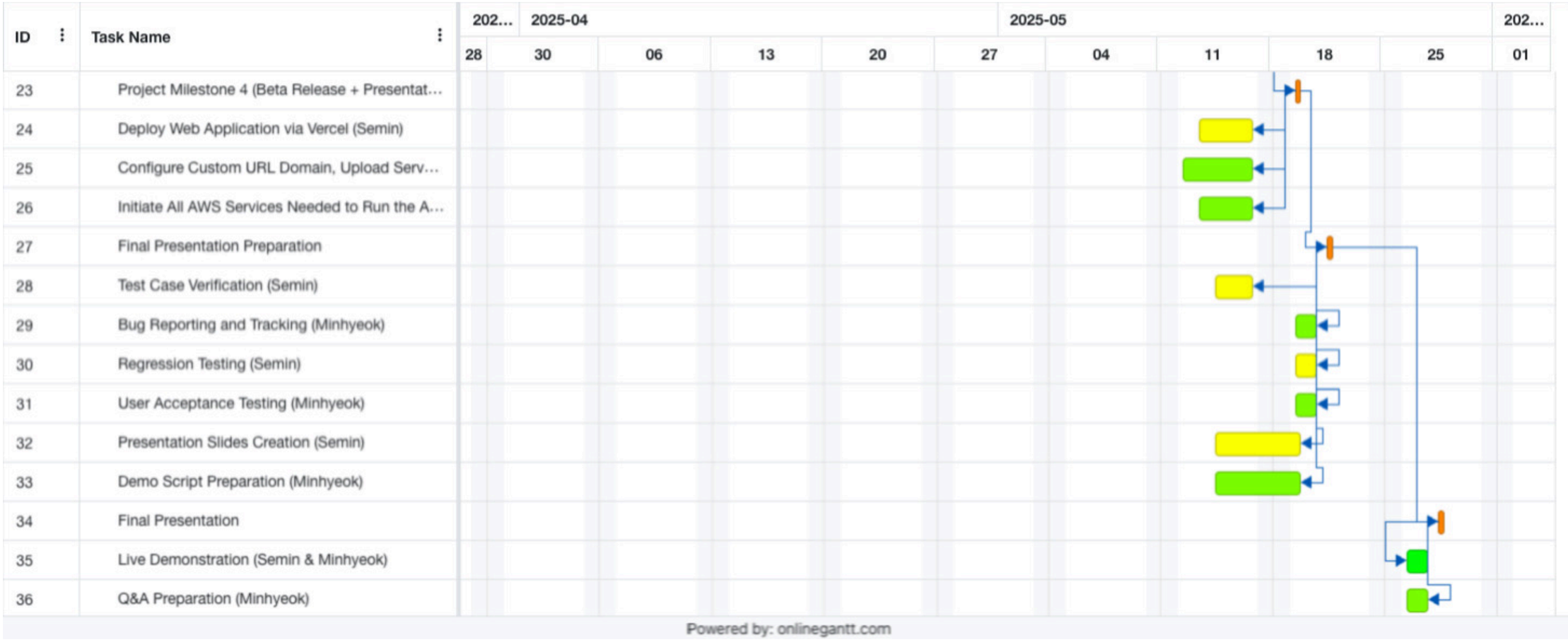
04: Implementation

Gentt Chart



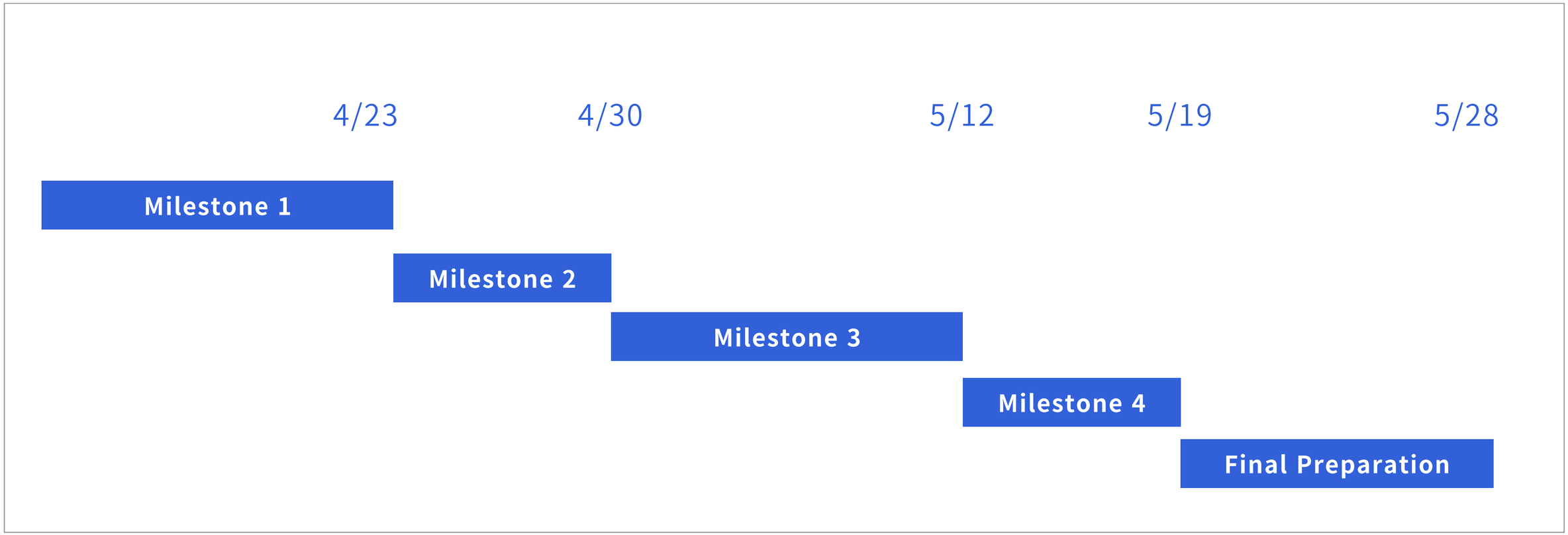
04: Implementation

Gentt Chart



04: Implementation

DueDate Chart



4/23
Milestone 1 Deadline

4/30
Milestone 2 Deadline

5/12
Milestone 3 Deadline

5/19
Milestone 4 Deadline

5/28
Final Presentation

MILESTONE 1

Due Date: April 23

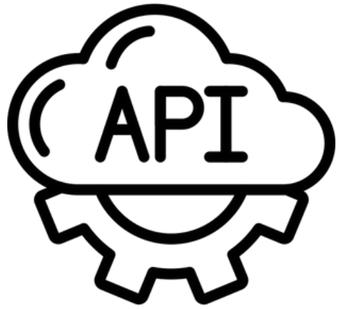
Key Implementation

01

DATABASE IMPLEMENTATION

02

API-SERVER INITIAL SETUP



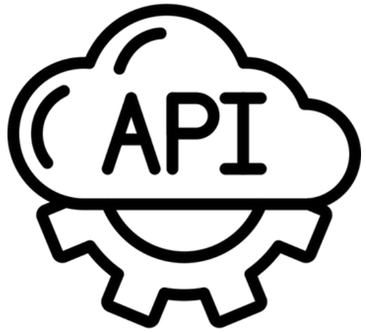
03

FRONTEND FOUNDATION & SERVER-CLIENT CONNECTION

04: Implementation

MILESTONE 1 - MINHYEOK

Database Implementation & API Server Initial Setup



1. Database Implementation
   - Set-Up DynamoDB Tables
     - User Table
     - Paper Table
     - User Archieve Table
     - Connection Table (for Websocket)
   - Create basic database CRUD operation
   - Implment database connection utilities

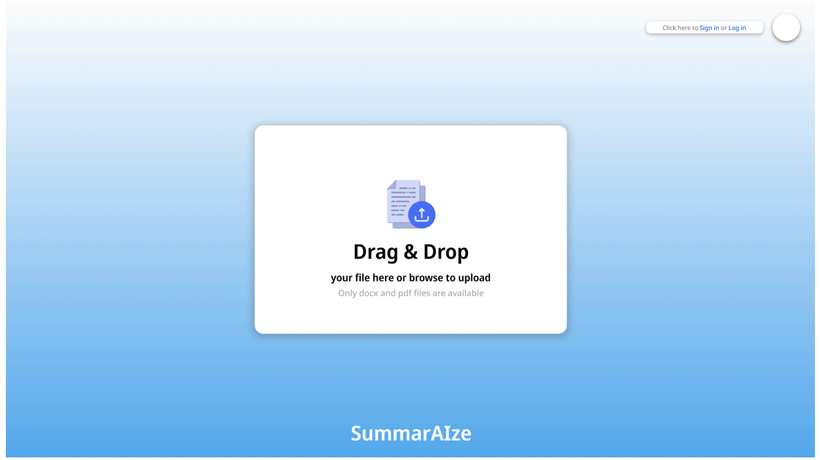
2. API server Initial Setup
   - Configure AWS SDK for local development
   - Set up Node.js project with necessary dependencies
   - Set-up basic log-system and error handling

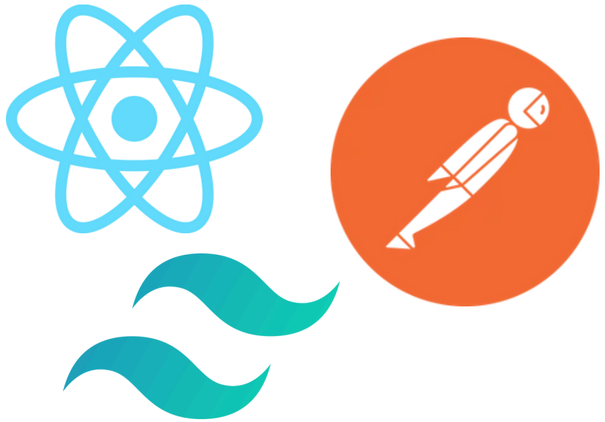
3. S3 Storage Server Setup
   - Configure S3 Bucket
     - Paper Upload, Summaries, User Profile
   - Implement File versioning and Encryption

04: Implementation

MILESTONE 1 - SEMIN

Front-end Foundation & API Server - Client Test





1. Front-end Foundation
   - Create React project with necessary dependencies
   - Configure routing with React Router
   - Set up Tailwind CSS for styling
   - Create login page UI
     - Login form
     - Error handling (Delayed to Milestone 3)
     - Redirection logic (Delayed to Milestone 3)
   - Implement Homepage with File Upload Component UI

2. API Server-Client Connection Testing
   - Connecting Test with Postman API service
   - Validate Request and Response format
   - Verify that the data returns the correct value according with Client parameters

MILESTONE 2

Due Date: April 30

Key Implementation

01

COMPLETE API SERVER IMPLEMENTATION

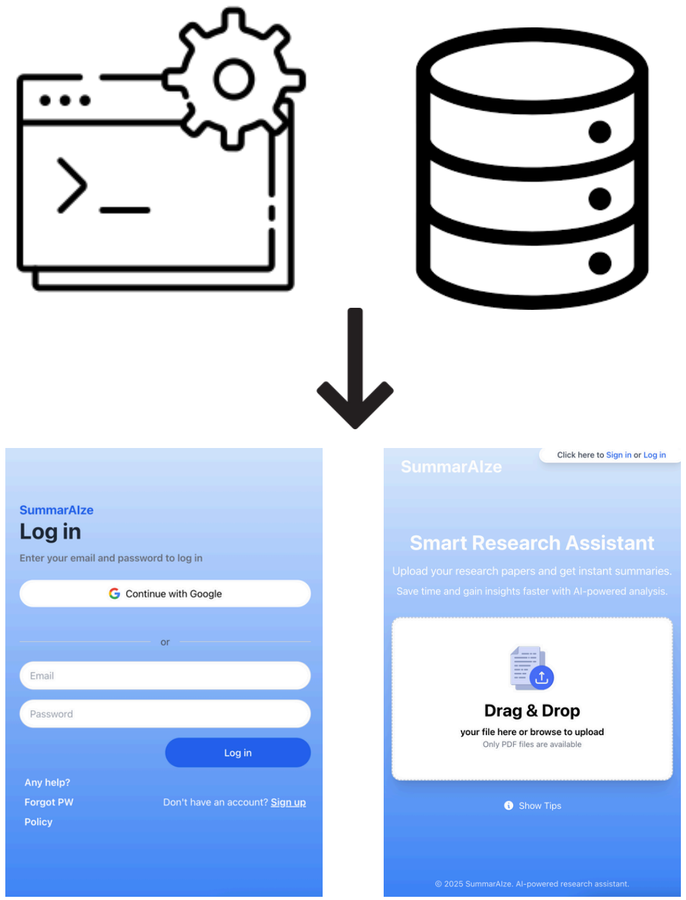
02

FRONTEND DEVELEOPMENT & CLIENT-API SERVER INTEGRATION

04: Implementation

MILESTONE 2 - MINHYEOK

Complete API Server Implementation



1. Complete API Endpoints for user Authentication
   - Login Endpoint
   - Signup Endpoint
   - Logout Endpoint

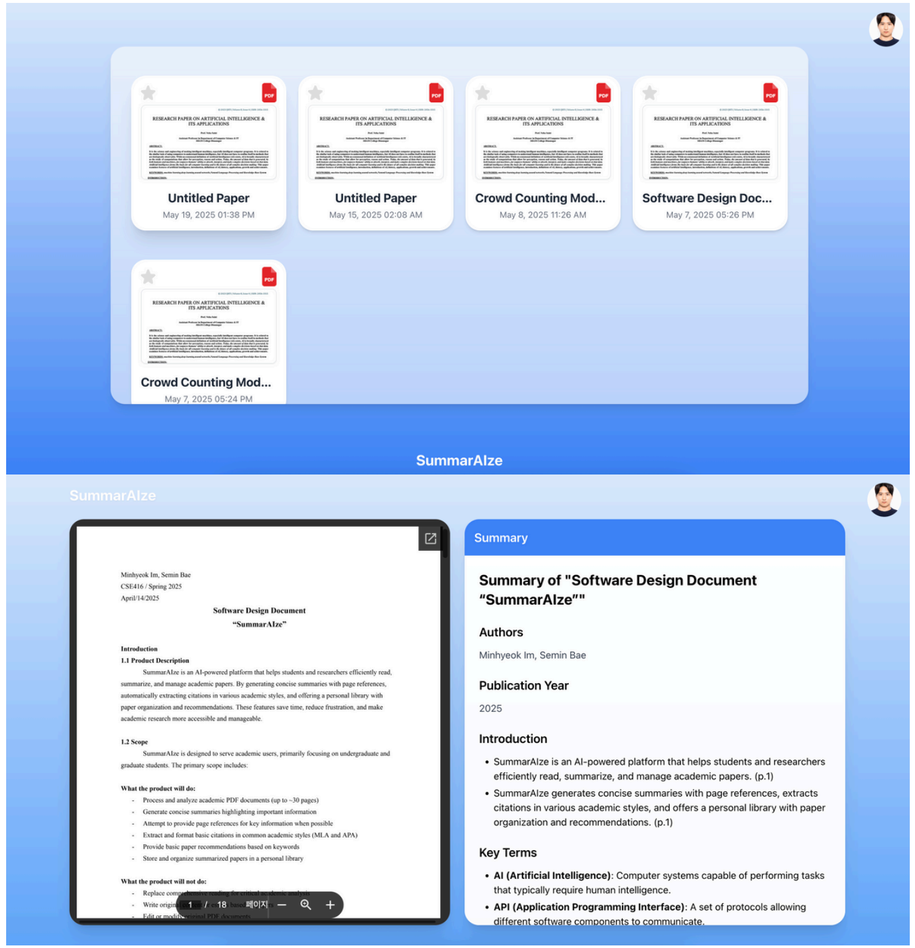
2. Complete API for Paper Management
   - Upload Request endpoint
   - Upload Confirmation endpoint
   - Paper Processing endpoint

3. Complete API Endpoints for Library Management
   - Load Library
   - Get paper Details Endpoints
   - Get Content URL Endpoints

04: Implementation

MILESTONE 2 - SEMIN

Frontend Development & Client-API server Integration



1. Implement API Client-Server in React
   - Authentication Service
   - Paper Upload Service
   - Login Service

2. Complete Full Developement of Frontend Webpage
   - Complete Login and Signup Page
   - Complete Library Client Page
   - Complete BookStand Client Page
   - Complete Redirection after Signin/Signup Processing
   - Complete All the Routing Service each pages

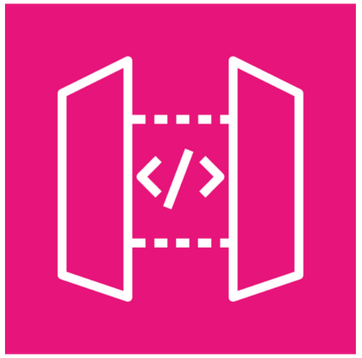
MILESTONE 3

Due Date: MAY 12

Key Implementation

01

WEB SOCKET CONNECTION & COMPLETE API SERVER INTEGRATION



02

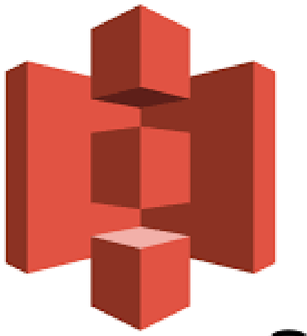
OPEN-AI SERVER CONNECTION

04: Implementation

MILESTONE 3 - MINHYEOK

Web Socket Connection & Complete API Server Integration

WebSocket



Amazon S3

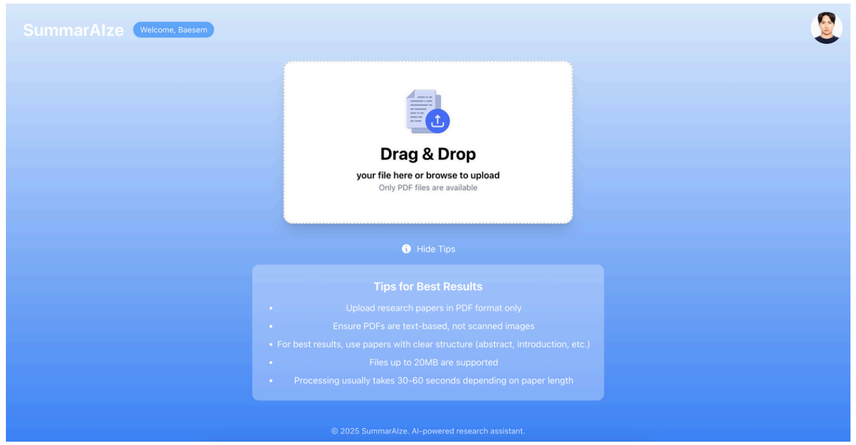
1. Full API Server Integration
   - Optimize API Responses
   - Complete Error Handling for all Edge cases
   - Complete Remaining API Endpoints
   - Complete to load Pdf file from S3 bucket in Server
   - Complete to load the Generated Summarize in Server

2. Implement Websocket Connection
   - Create a unique websocket connection between the user and the server
   - Server consistently sends paper progress status through the web socket connection
   - Disconnect the connection when the summarization is done

04: Implementation

MILESTONE 3 - SEMIN & MINHYEOK

Open-AI Server Connection & Client-Server API Connection



1. Open-AI Server Connection
   - Get the API Keys from OpenAI Developer Portal
   - Set Up Development Environment
   - Initialize the API Client
   - API connection with server and openAI

2, Client-Server API connection
   - Complete to connect BookStore page
   - Complete User Page
   - Addes Some Front-end Features
     - Color Transition when Icon clicked
     - Greeting UI in HomePage
     - Dynamic Instruction to use SummarAIze

MILESTONE 4

Due Date: MAY 19

Key Implementation

01

FRONT-END DEPLOYMENT

02

DEPLOYMENT AND INTEGRATION OF BACK-END SERVICE

MILESTONE 4 - MINHYEOK

Web Application Deployment

1. Finished End-to-End Backend Server
   - Completed to Change the Summarization Language
   - Completed access to all Summarized Paper from User Library
   - Completed All the Library End-to-End Connection
   - Complete All the BookStand End-to-End Connection
   - Implement Google OAuth SignUp function on the backend.

2. Deployment all the updated Back-end Deployment
   - Setup AWS Environment
   - Configure Serverless Framework
     - Set up yml file
   - Prepare Lambda functions
   - Deploy backend services

04: Implementation

MILESTONE 4 - SEMIN

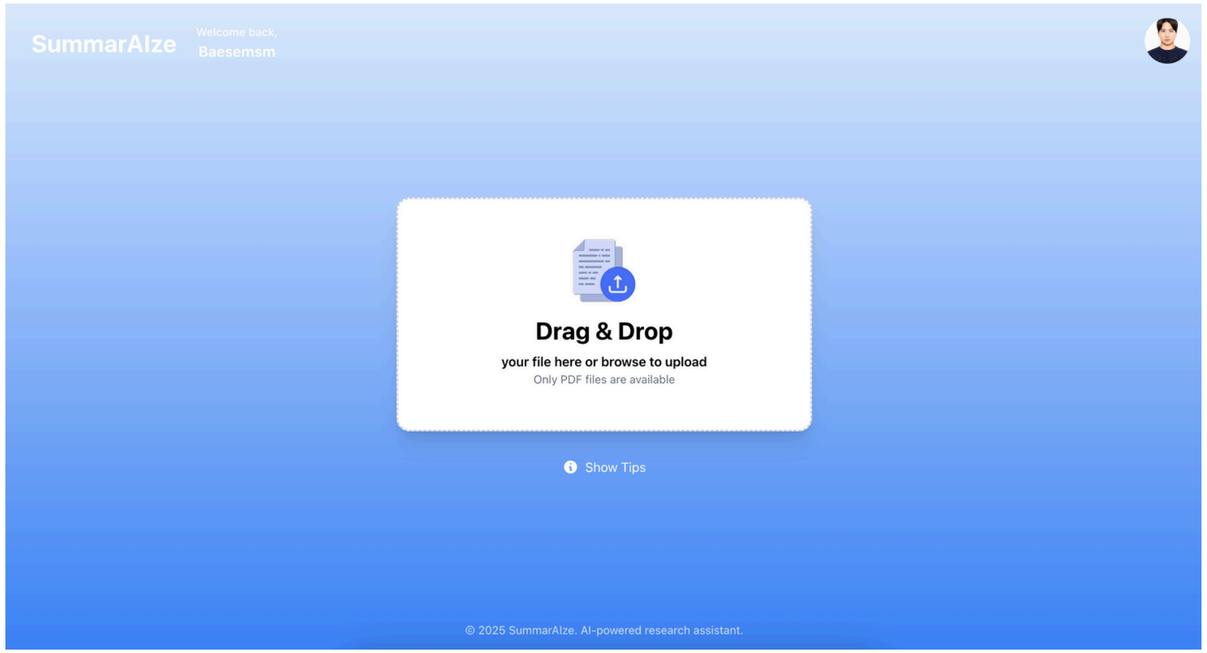
Web Application Deployment

1. Web Application Deployment
- Deploy Frontend Application via Vercel
  - Configure build setting
  - Set up Environmnet variables

04: Implementation

NEW UPDATE: SEMIN & MINHYEOK

Complete All the Delayed Progress

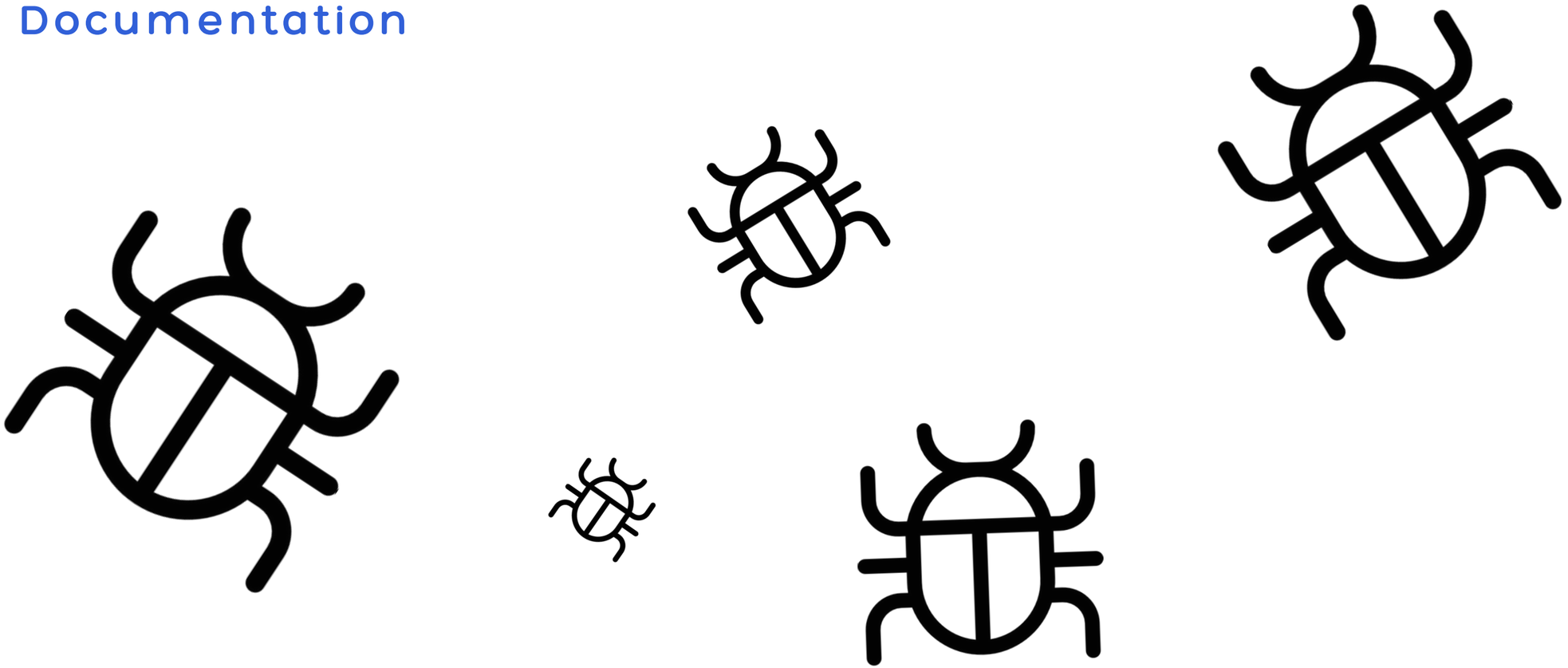


- Complete to Change the UserProfile
  - User Profile Image
  - User Name
- Enhance the User Experience
- Google OAuth
- Presentation Preparation
- Documentation
- Tests

04: Implementation

BUG REPORT

Bug Documentation



Link: Google Docs

04: Implementation

MAIN ROLE

MINHYEOK IM

- Database (DynamoDB) Setup & Implementation
- API server (AWS Lambda) Setup & implementation
- S3 Storage Setup & integration
- Client-Server interaction integration
- Google OAuth integration (backend)
- Complete backend deployment

SEMIN BAE

- Frontend foundation & implementation
- Google OAuth integration (frontend)
- Client-Server connection integration
- File Upload & Download implementation
- Complete frontend deployment

05: Takeaways

WHAT WE'VE GOT FROM THIS PROJECT

Full-Stack Development Experience with Team

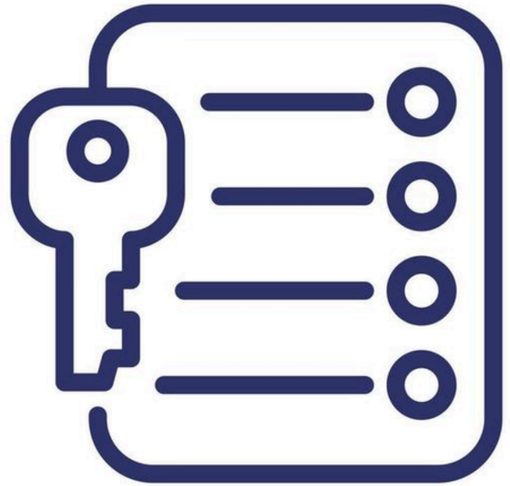
- React + AWS Lambda Webservice
- Dsign RESTful API and WebSocket connection
- DynamoDB NoSQL DB service
- Deploy both Frontend & Backend

Third-Party Integration Mastery

- OpenAI API
- Google OAuth
- AWS Services

Project(Team-work) Experience

- Bug Report
- Milestone (Agile Developement Process)
- Documentation



CSE416 FINAL PRESENTATION

THANK YOU

CSE416: Final Presentation

SummarAIze

MINHYEOK IM & SEMIN BAE

Service Link:

React App

Web site created using create-react-app

vercel.app