System Design Document

Project: Syllabally

Sprint: Sprint 2

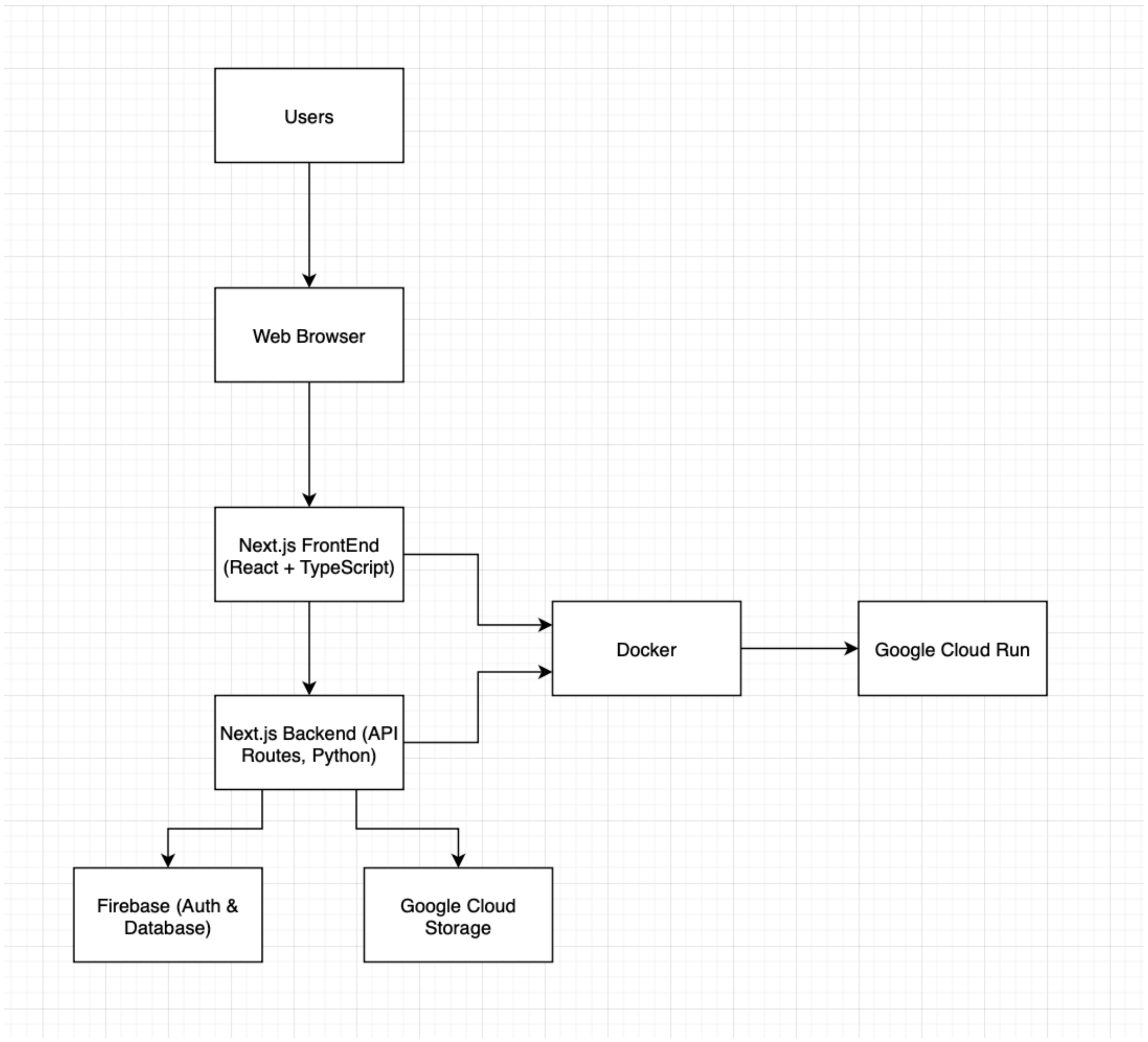
Team Members: Hien Le, Maya Shamir, Thomas Aziz

Table of Contents

1. Overview

This document provides a high level of the Syllabally system, including its architectural components, interaction with the environment, system decomposition, and detailed class descriptions. The platform’s goal is to allow students to register, search, and manage the credit system.

2. System Architecture


Users
Web Browser
Next.js FrontEnd (React + TypeScript)
Docker
Google Cloud Run
Next.js Backend (API Routes, Python)
Firebase (Auth & Database)
Google Cloud Storage

3. Environment Dependencies and Assumptions

- **OS**: Compatible with Windows, MacOS, and Linux for development and deployment.
- **Programming Languages**: JavaScript (Node.js for backend and Next.js for frontend).
- **Databases**: Firebase for credit system and Google Cloud storage for the uploaded syllabuses
- **Authentication**: Firebase Authentication for handling user sign-up, login, and access control.

4. System Decomposition

The system is divided into components that map onto the architectural view:

- **Frontend**: Implements UI for account management, syllabus browsing, and homepage intefaces.
- **Backend**: Handles API requests, processes business logic, and manages interactions with Firebase Authentication.
- **Database**: Controlling user credits and manage the upload files in the database.
- **Authentication**: Firebase Authentication provides secure access control and user session management.

Each component is responsible for interacting with other components to fulfil specific roles:

- **Frontend** requests data and displays it to the user.
- **Backend** receives requests, validates, processes, and retrieves data from the database, returning responses to the frontend.
- **Database** holds all user, syllabus, and credit data.
- **Storage** keeps syllabus files accessible through secure links.

5. Class Descriptions (CRC Cards)

Class Name: User Class	
● **Parent Class**: None ● **Subclasses**: None	
● **Responsibilities**: ○ Create and manage user accounts. ○ Track user credits and account status. ○ Store bookmarks and recommendations.	● **Collaborators**: AuthController, CreditManager, DatabaseManager

Class Name: AuthController Class	
● **Parent Class**: None ● **Subclasses**: None	
● **Responsibilities**: ○ Handle user authentication (registration, login, logout). ○ Validate and maintain user sessions.	● **Collaborators**: FirebaseAuth, User, ErrorHandler

Class Name: SearchController Class	
● **Parent Class**: None ● **Subclasses**: None	
● **Responsibilities**: ○ Provide search functionality for syllabi. ○ Handle search queries and filtering options.	● **Collaborators**: SyllabusManager, User

Class Name: SyllabusManager Class	
● **Parent Class**: None ● **Subclasses**: None	
● **Responsibilities**: ○ Upload and categorize syllabi. ○ Search syllabi based on filters. ○ Store syllabus metadata in the database.	● **Collaborators**: DatabaseManager, CreditManager, SearchController

Class Name: CreditManager Class	
● **Parent Class**: None ● **Subclasses**: None	
● **Responsibilities**: ○ Calculate and update user credits. ○ Track transactions related to credit changes.	● **Collaborators**: User, SyllabusManager, DatabaseManager

Class Name: ErrorHandler Class	
● **Parent Class**: None ● **Subclasses**: None	
● **Responsibilities**: ○ Handle system errors and exceptions. ○ Manage error logging and provide feedback to users.	● **Collaborators**: AuthController, SyllabusManager, CreditManager

Class Name: DatabaseManager Class	
● **Parent Class**: None ● **Subclasses**: None	
● **Responsibilities**: ○ Handle all database interactions. ○ Store and retrieve data related to users, syllabi, and credits.	● **Collaborators**: User, SyllabusManager, CreditManager

6. Error Handling Strategy

The platform includes error-handling strategies for anticipated errors and exceptions. The **ErrorHandler** class is responsible for managing exceptions and providing user feedback in the following cases:

- **Invalid User Input**: Display specific error messages for missing or incorrect form fields.
- **Authentication Failures**: Notify users of failed login or registration attempts, suggest recovery steps.
- **File Upload Errors**: Alert users if a syllabus upload fails, with suggestions to retry or check the file format.
- **Database Connection Failures**: Implement retry logic on failed database operations, with fallback responses for prolonged outages.
- **Network Failures**: Show user-friendly messages and retry options for network issues during critical operations.