**CS673 Software Engineering** 

**Team 2 - Focused Study**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Siddhesh Dighe | Team Leader | *Siddhesh* | 10/19/2023 |
| Pushkar | Design Implementation Leader | *Pushkar* | 10/19/2023 |
| Sai Ramya Devineni | QA Leader | *Ramya* | 10/19/2023 |
| Ayush Bhaliya | Security Leader | *Ayush* | 10/19/2023 |
| Paridhi Talwar | Requirement Leader | *Paridhi* | 10/19/2023 |
| Deepali Chawla | Configuration Leader | *Deepali* | 10/19/2023 |
| Supriya | Backup QA Leader | *Supriya* | 10/19/2023 |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

[Introduction](#_87t9hln2vjz0)

[Software Architecture](#_3ipvmjgn6clp)

[Class Diagram](#_ky60nv8suxxm)

[UI Design (if applicable)](#_7ucksmkf6rzx)

[Database Design (if applicable)](#_tcmuor4nl1kz)

[Security Design](#_x18fj36s1121)

[Business Logic and/or Key Algorithms](#_mtfbusfb0eq3)

[Design Patterns](#_9zvwkmc4luo5)

[Any Additional Topics you would like to include.](#_15tmymhipvdv)

[References](#_50ojo9i46ytq)

[Glossary](#_8n34lvocupub)

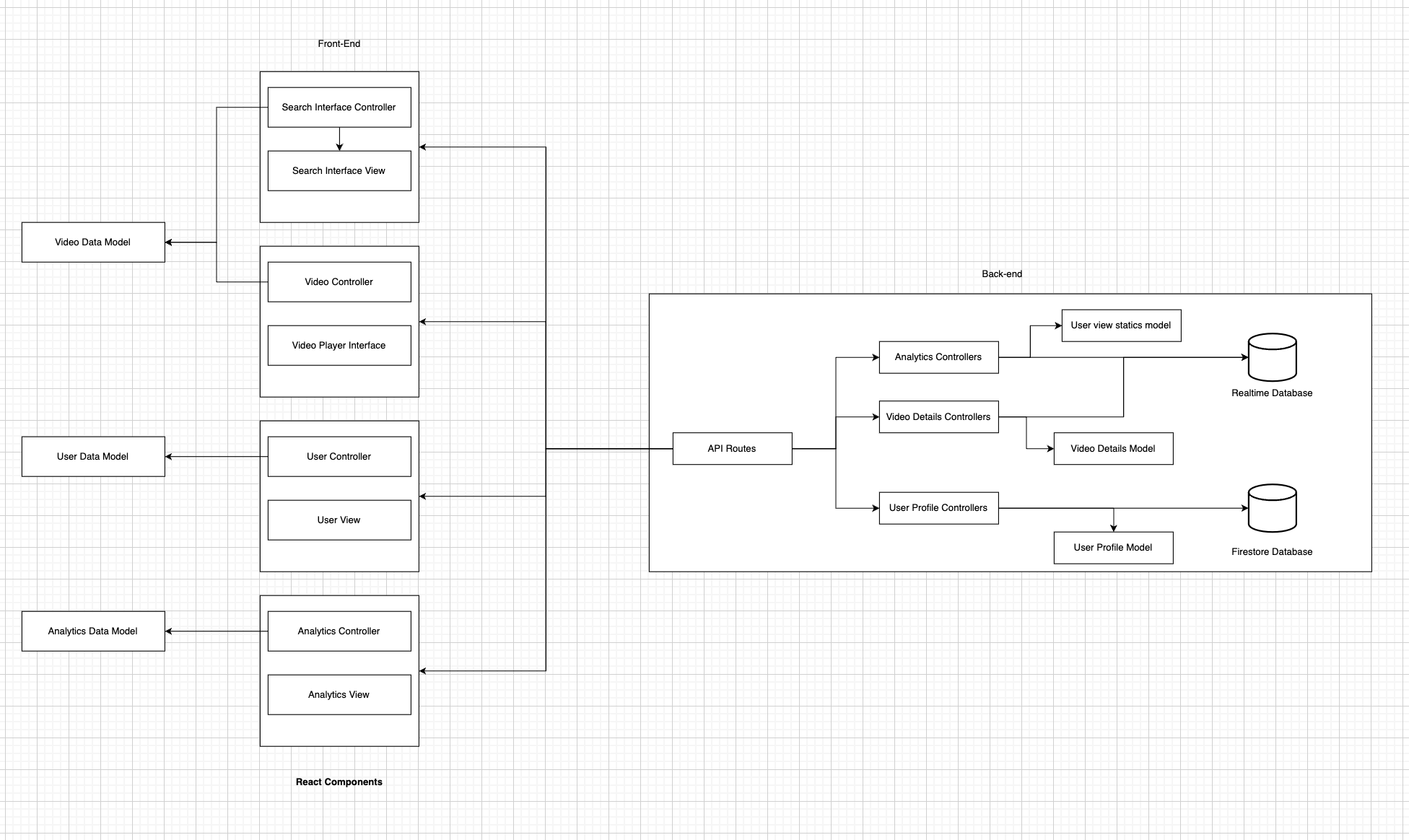
# Introduction

We introduced FocusedStudy in the [CS673 Team 2 - SPPP](https://docs.google.com/document/d/1_ksxY6Lx5jmB0yWBAFynKryJhSajHyqCK84IfrErR8I/edit) document. Through this document, we aim to describe the design choices we’ve made to meet the requirements set for Focused Study. The document details the Software Architecture, and the components of the system using a UML Class Diagram, and Lofi Wireframes to highlight the UI Design, Database design, and design patterns.

The top goals of the FocusedStudy system are divided into the following two categories:

1. External Quality Characteristics - efficiency, reliability, robustness, usability
2. Internal Quality Characteristics - readability, maintainability.

# Software Architecture

  
  
We are planning to use the **Model-View-Controller(MVC)** architecture to decompose our software system into multiple subsystems.

**Intend:** Separate the application’s concerns into three interconnected components; Model(data), View(UI), and Controller(business logic).

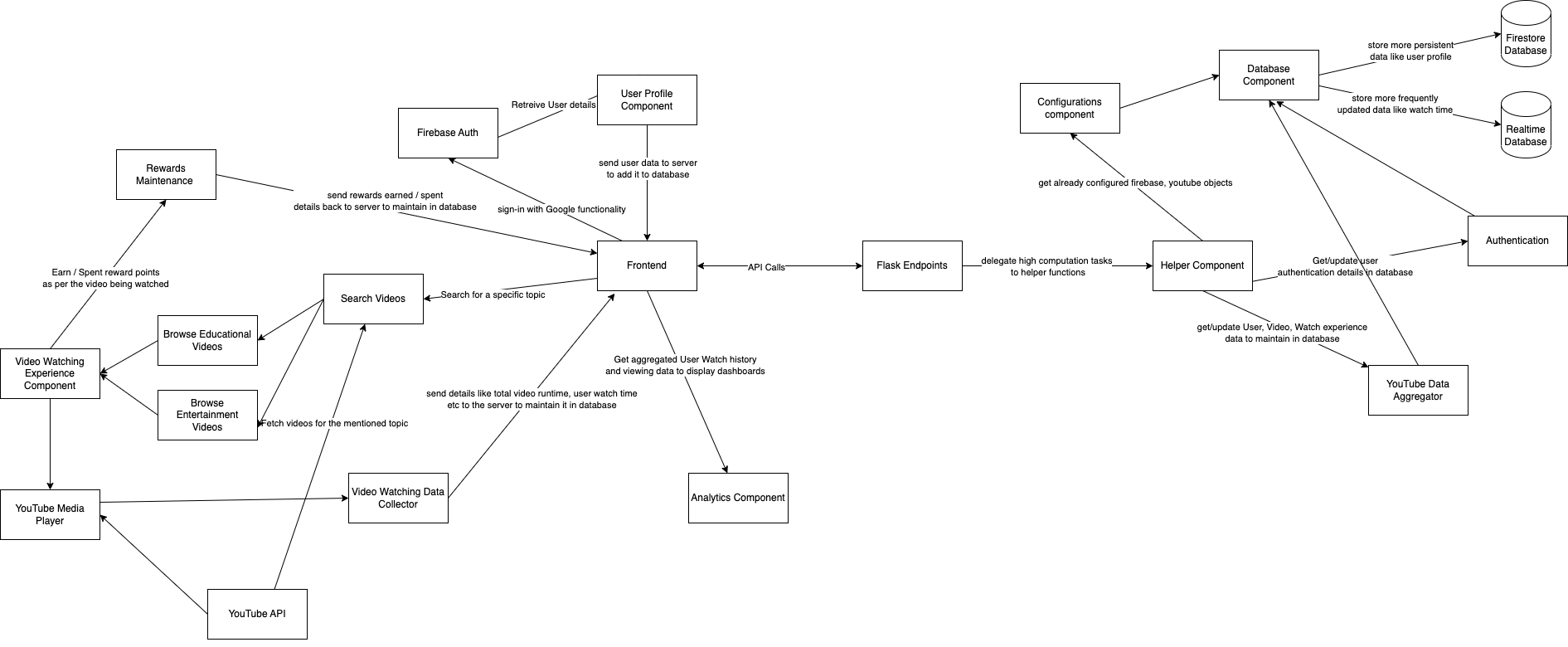
**Reason for Choosing:** MVC Architecture is the best fit for the desirable functional requirements from our project.

**Components:**

1. **Model:** Represents the data and the business rules of the application.
   * **Video Data:** Information about the curated educational/entertainment videos fetched from Youtube.
   * **User Data:** Information about the user, their watch history, earned points, etc.
   * **Analytics Data:** Data related to user behavior, attention span, and learning habits.
2. **View:**
   * **Search Interface:** Allows users to search for educational videos.
   * **Video Player Interface:** Provides a distraction free video watching experience.
   * **Analytics Dashboard:** Displays insights into the user’s learning habits and attention span.
   * **Gamification Interface:** Displays earned points and rewards.
3. **Controller:** Acts as an interface between Model and View. It takes the user’s input from the view, processes it and returns the display output to the view.
   * **Search Controller:** Handles search queries and fetches relevant videos.
   * **Video Controller:** Manages video playback, tracking watch time, and other related functionalities.
   * **User Controller:** Manages user authentication, profiles updates, and gamification aspects.
   * **Analytics Controller:** Processes and presents data related to user behavior.

Benefits:

* Separation of Concerns: Each component has a distinct responsibility, making the application modular and easier to manage.
* Flexibility: Changes in one component have minimal impact on others.
* Scalability: New features or functionalities can be added with minimum disruption to existing code.



**Component Architecture:**

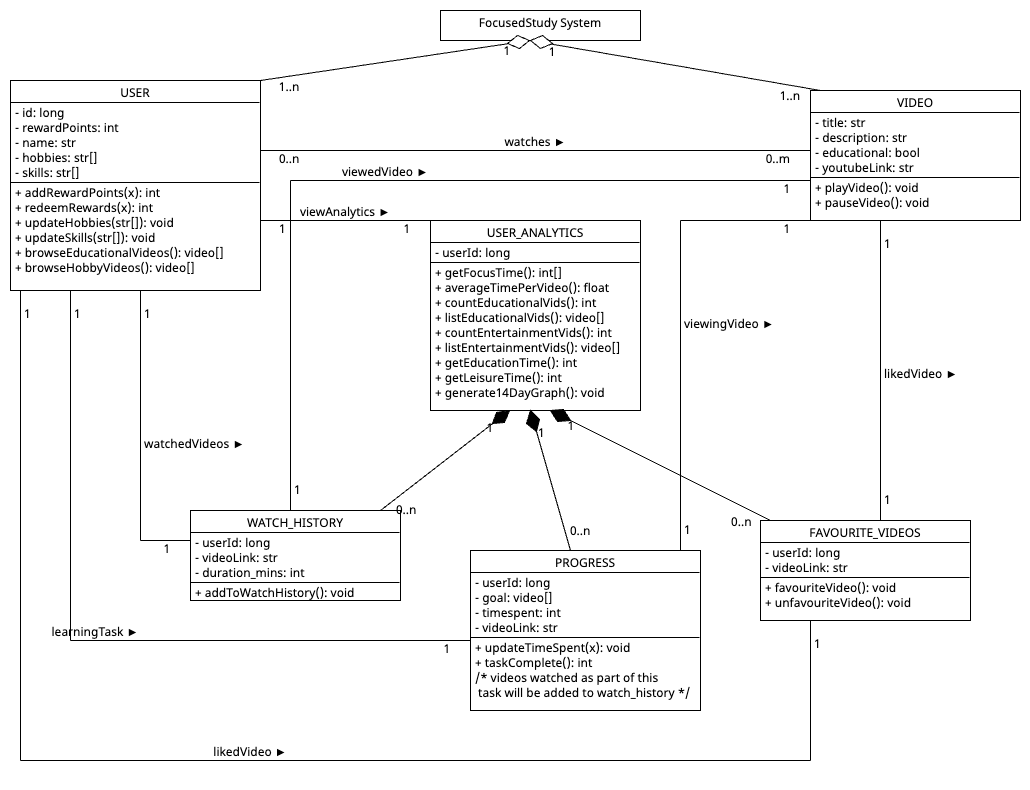
**Backend:**

1. **Flask endpoints:** This is the entry point for the flask server.Every request (GET , POST) is sent to this component. The only responsibility of the flask end point is to delegate high computation tasks to helper functions.
2. **Helper Component:** It takes care of high computation tasks. It connects to other components like the youtube data aggregator or the database component to complete the request and send a JSON response back to the client.
3. **Configuration Component:** This component is responsible for setting up FirebaseAuth , Firestore and the realtime database.
4. **Database Component:** This component uses the Firebase configuration done by the configuration component to read and write data to/ from Firestore and the real time database.
5. **Firestore Database:** It is used to keep persistent data like the user profile. It will be used to store data which would be read less frequently hence won't need much read and write operations. In Spite of this it can store a lot of data without it being expensive , for example it can store data for thousands of users but each user needs to fetch that data only once.
6. **Realtime Database:** It is used to store analytical data. It will be used for data which requires less storage but is very cost effective to read and write constantly in real time , for example the user's watch time will be stored in a real time database since it needs to be constantly updated.
7. **Authentication:** This component is responsible for checking if the user is authorized or not. Before granting the user permission to read and write from a real time database and Firestore database.
8. **Youtube Data Aggregator:** This component is responsible for accessing the youtube search API and youtube data API. It is also responsible for aggregating data received from both the API’s.

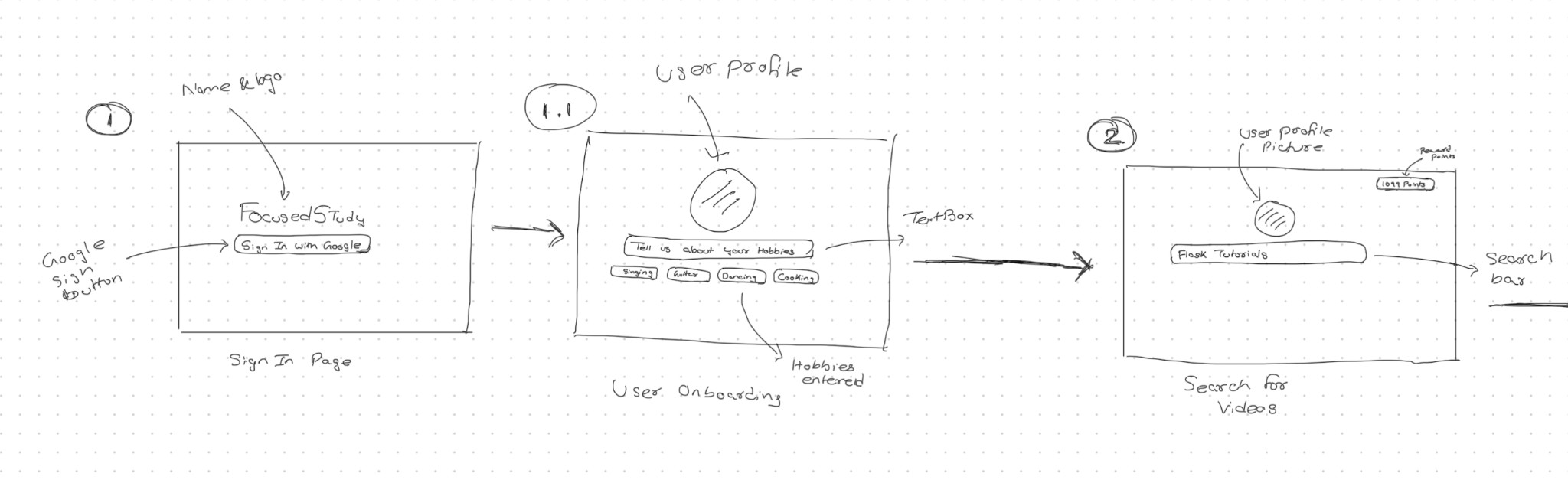
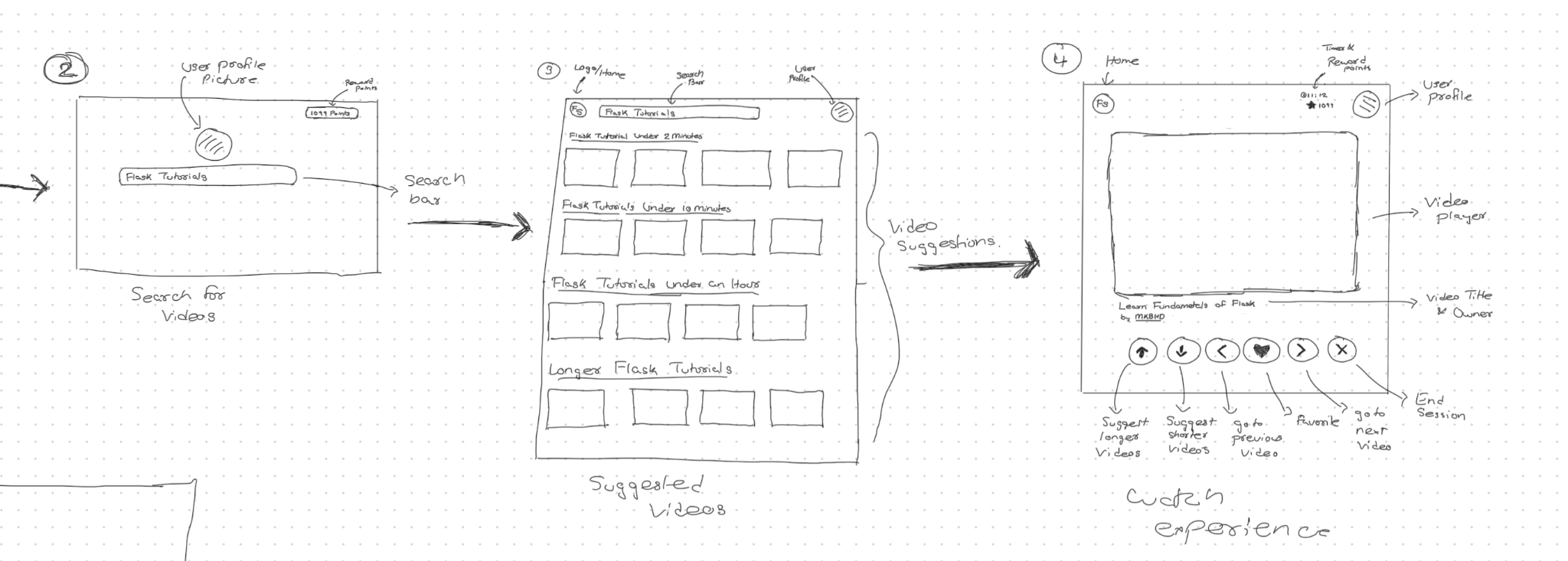
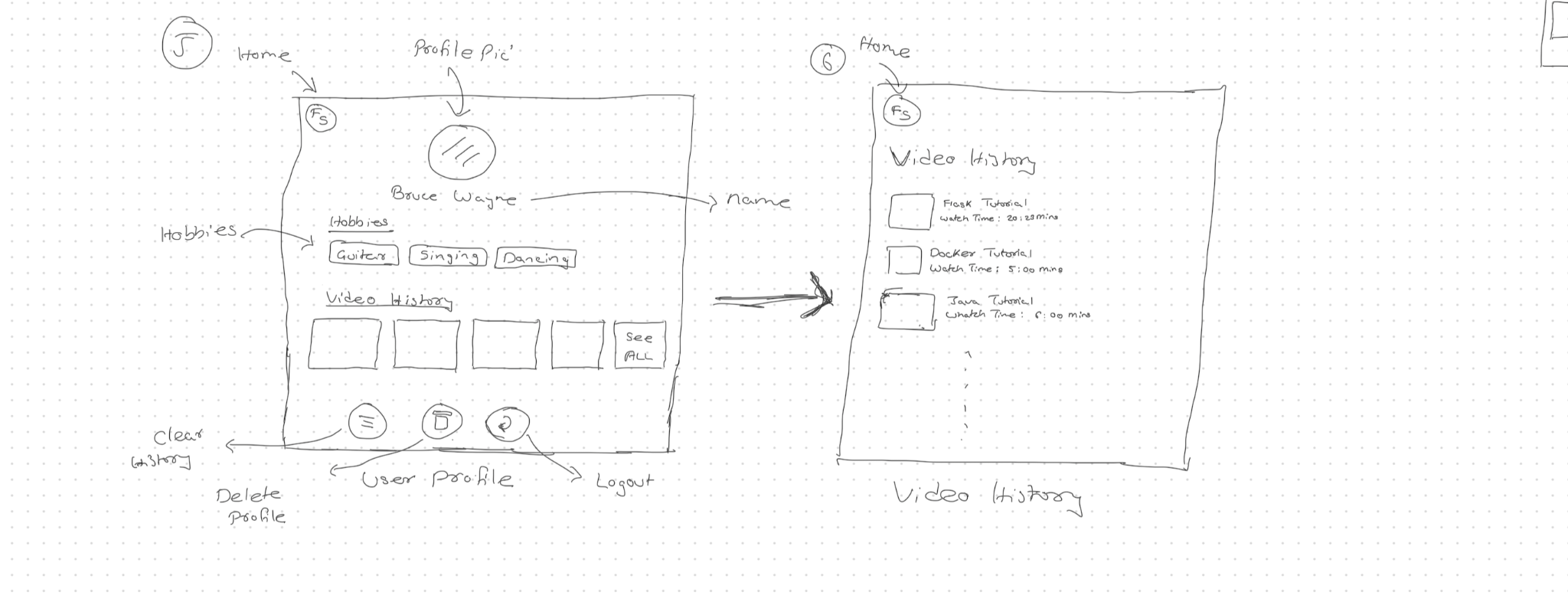
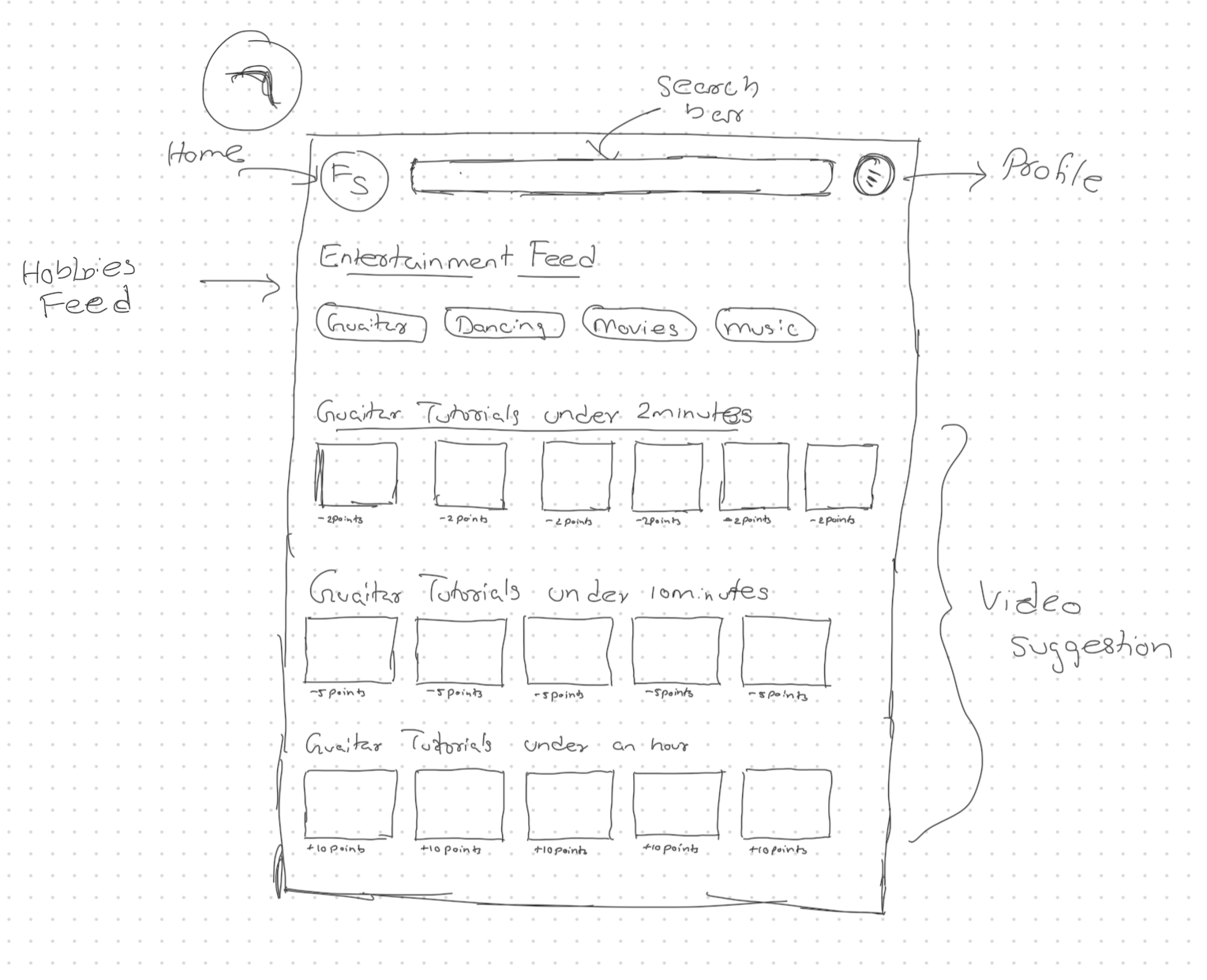
**Frontend:**

1. **User Profile Component:** This component contains design and logic for the user profile page. User profile would have basic user information like name , email , profile picture followed by user id which is unique. This component will also show users watch history , hobbies and favorite videos.
2. **Firebase Auth:** It will be used to authenticate users into the app. It will also keep a check on if the user is logged in or not.
3. **Search Videos:** This component contains design and logic for the search bar functionality. It sends a GET request to the server with parameters like search query , video duration and total number of videos to be fetched.
4. **Browse Educational Videos:** This component contains design and logic for the video that's being searched. It divides the browse feed into 3 sections one for videos under 4 minutes , two for videos under 30 minutes and the last one for in depth videos.
5. **Browser Entertainment Videos:** This component contains design and logic for entertainment based videos which relate more to the user’s hobby. It shows a list of all the hobbies mentioned by the user and provides 3 sections for the selected hobby. 3 sections being videos under 4 minutes, videos under 30 minutes and the last one for in depth videos
6. **Video Watching Experience Component:** This component is responsible for the overall design and functionality of the video being played. It provides buttons to start the next video, or the previous video. It also provides the ability to like a video to add it to favorites.
7. **YouTube Media Player:** This is the native media player provided by youtube which is used to track user actions (what duration was the video paused or stopped at)
8. **Video Watching Data Collector:** This component is responsible for tracking the user's video watching activity and a snapshot of iit to the server/database.
9. **Analytics Component:** This component is responsible for displaying charts & graphs for dashboard visualizing user activity & watch-time over the period of time.
10. **Reward Maintenance:** This component is responsible for keeping track of the rewards spent/earned by the user.

# Class Diagram

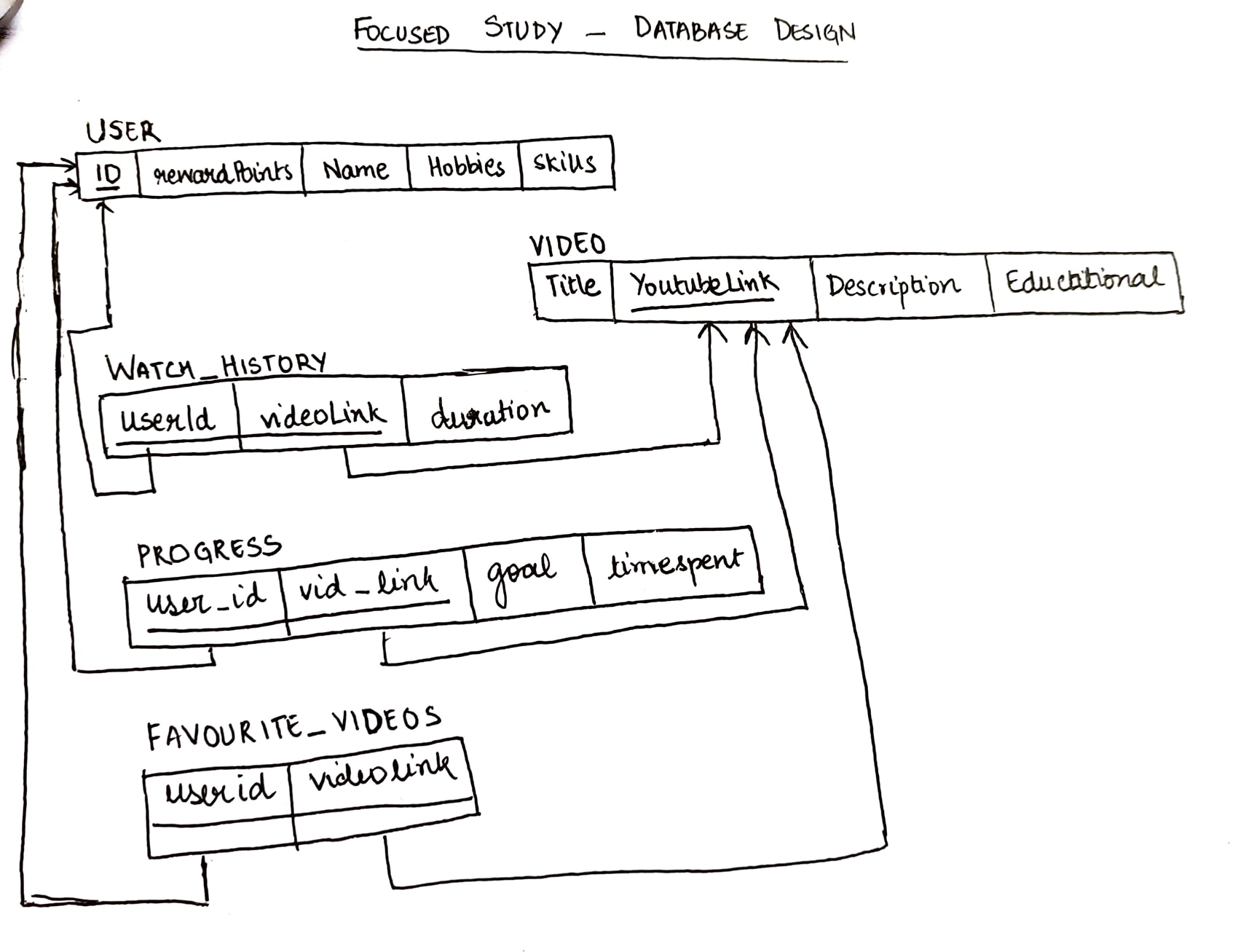


# UI Design (if applicable)

In this section, you can describe your UI design  
  
  
  
  
  
   
Note: We haven’t mentioned analytics dashboard in the wireframes yet because we are still brainstorming on how the screen should look like and what information it should provide.

# Database Design (if applicable)

For Database design we have used Relational DBM to demonstrate.



Note: This is our initial database design. It is subject to slight changes as we implement our project.

# Security Design

1. Every user is authenticated using their google accounts. That is, our system will make use of Google Authentication to allow users to access our system. Therefore, we’re not storing passwords in our database.
2. Any interaction that a user has with our system post login, would require a JSON Web Token (JWT) to ensure that they’re authenticated.
3. All communication between the front end and back end will happen through HTTPS enabled REST APIs.

# Design Patterns

**Observer Pattern:**

The Observer pattern is implemented to models when a state change should lead to updating relevant objects.

In our application, everytime a user wraps up a learning session they are awarded points, similarly their reward points must be deducted after an entertainment video watching session.

**Composite Pattern:**

The Analytics Dashboard will be implemented using the Composite Design pattern. The analytics dashboard consists of three customised playlists with videos they’ve watched, the time they’ve spent in each learning session, their collection of liked videos. Each of these playlists will be a separate component using this design pattern.   
**Singleton Pattern:**

Singleton pattern would be mostly used as an interface between the database and the backend server where the backend server would only use a single database object to both read and update the database.

# Any Additional Topics you would like to include.

# References

# Glossary