**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).

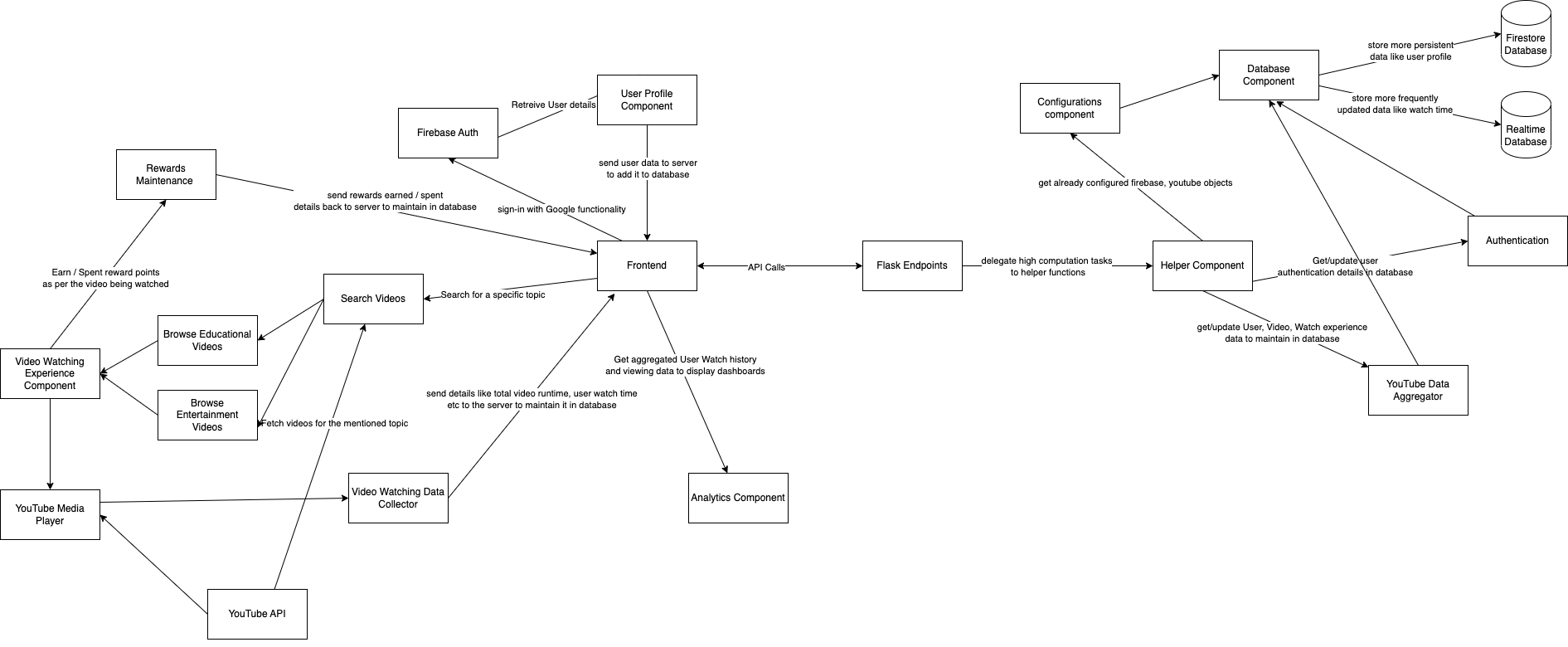
**Reason for Choosing:** MVC Architecture is the best fit for the desirable functional requirements from our project.

**Components:**

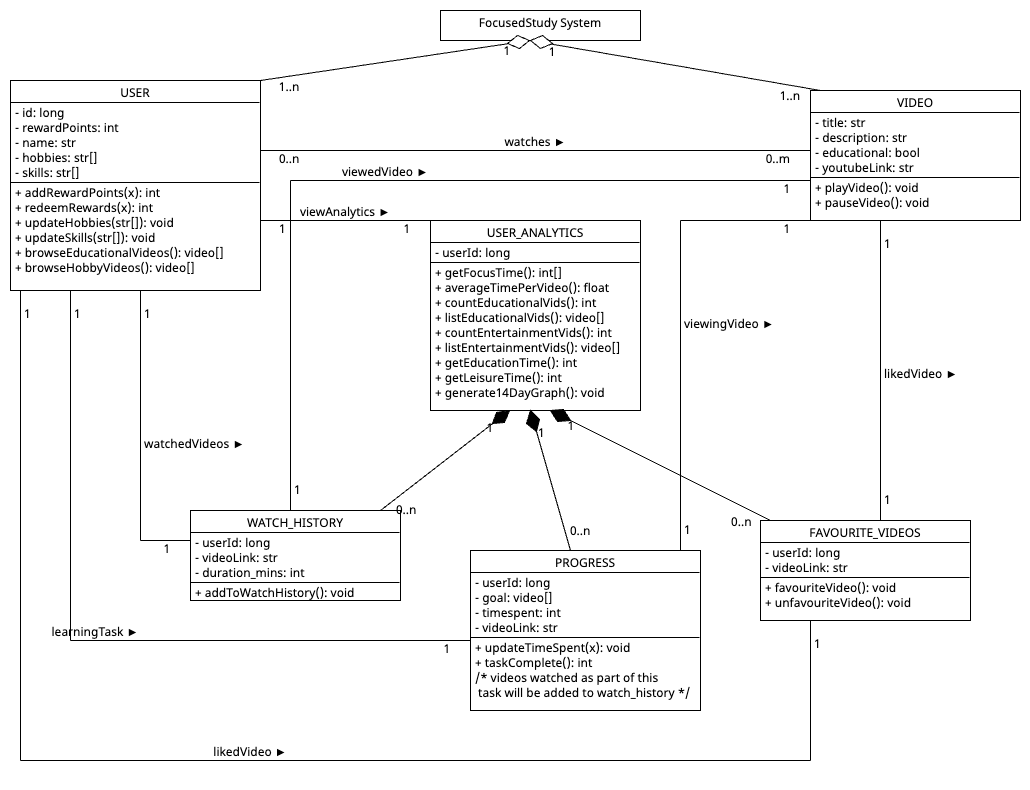
1. **Model:** Represents the data and the business rules of the application.
   * **Video Data:** Information about the curated educational/entertainment videos fetched from Youtube.
   * **User Data:** Information about the user, their watch history, earned points, etc.
   * **Analytics Data:** Data 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 vidoes.
   * **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:

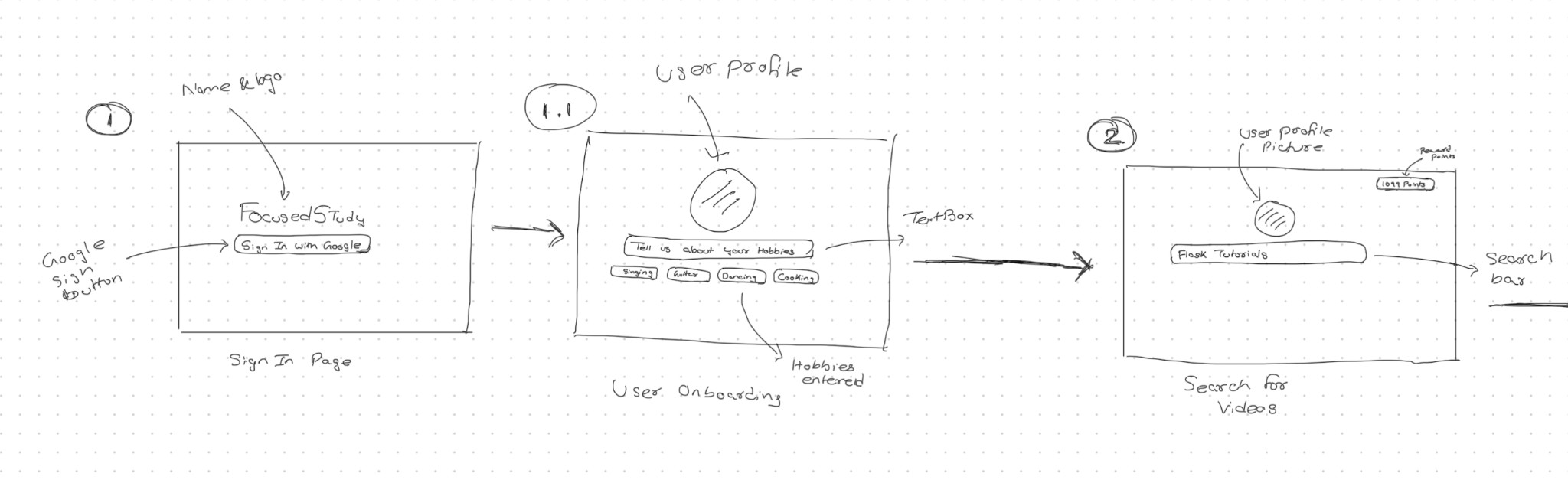
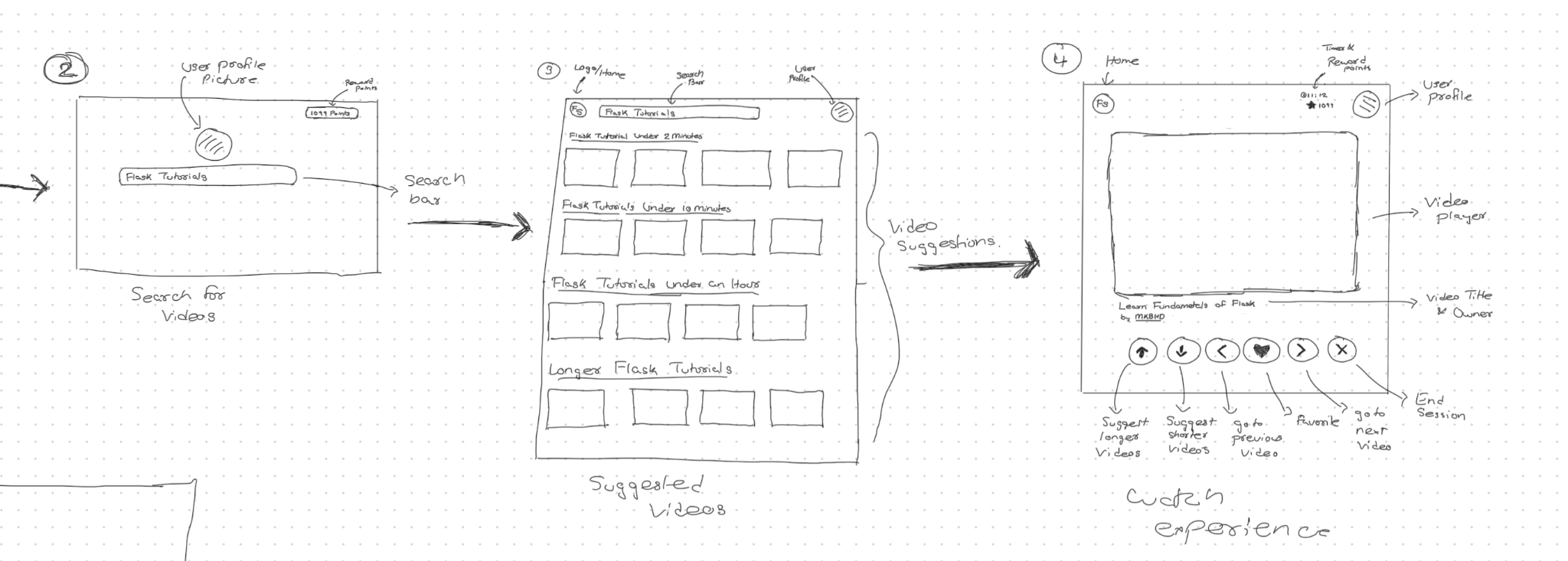
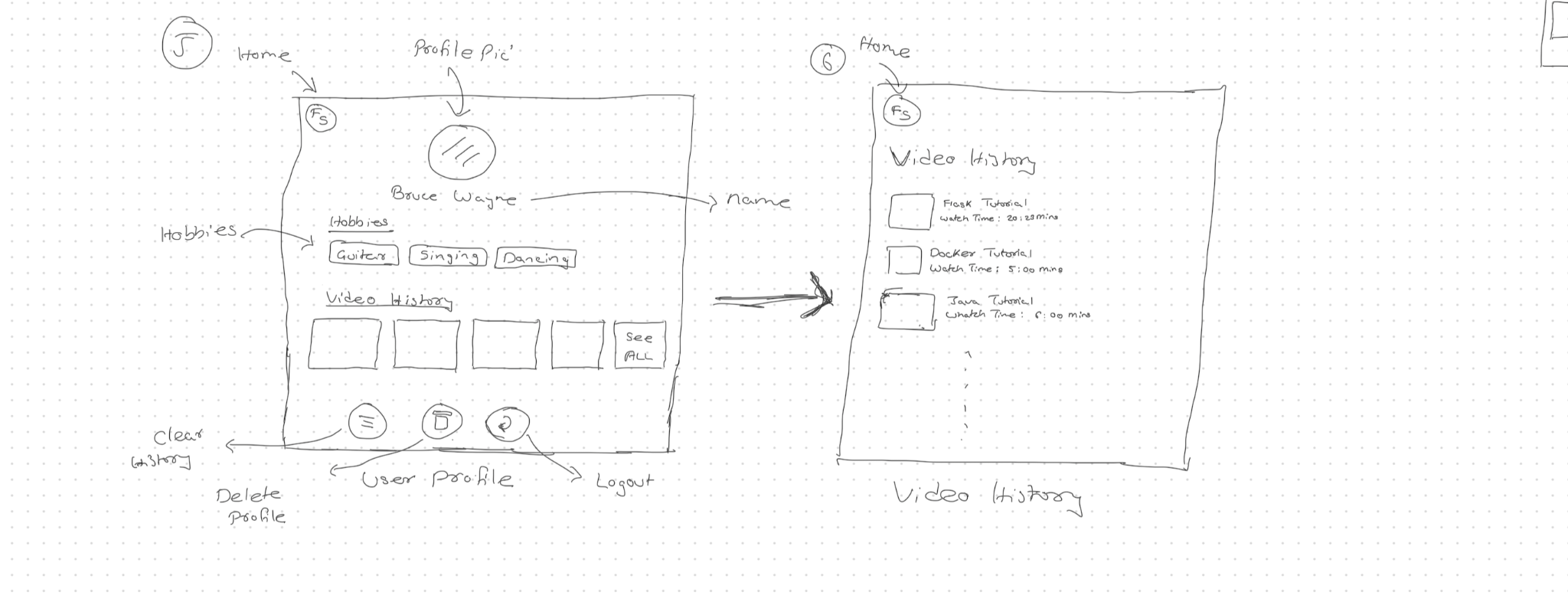
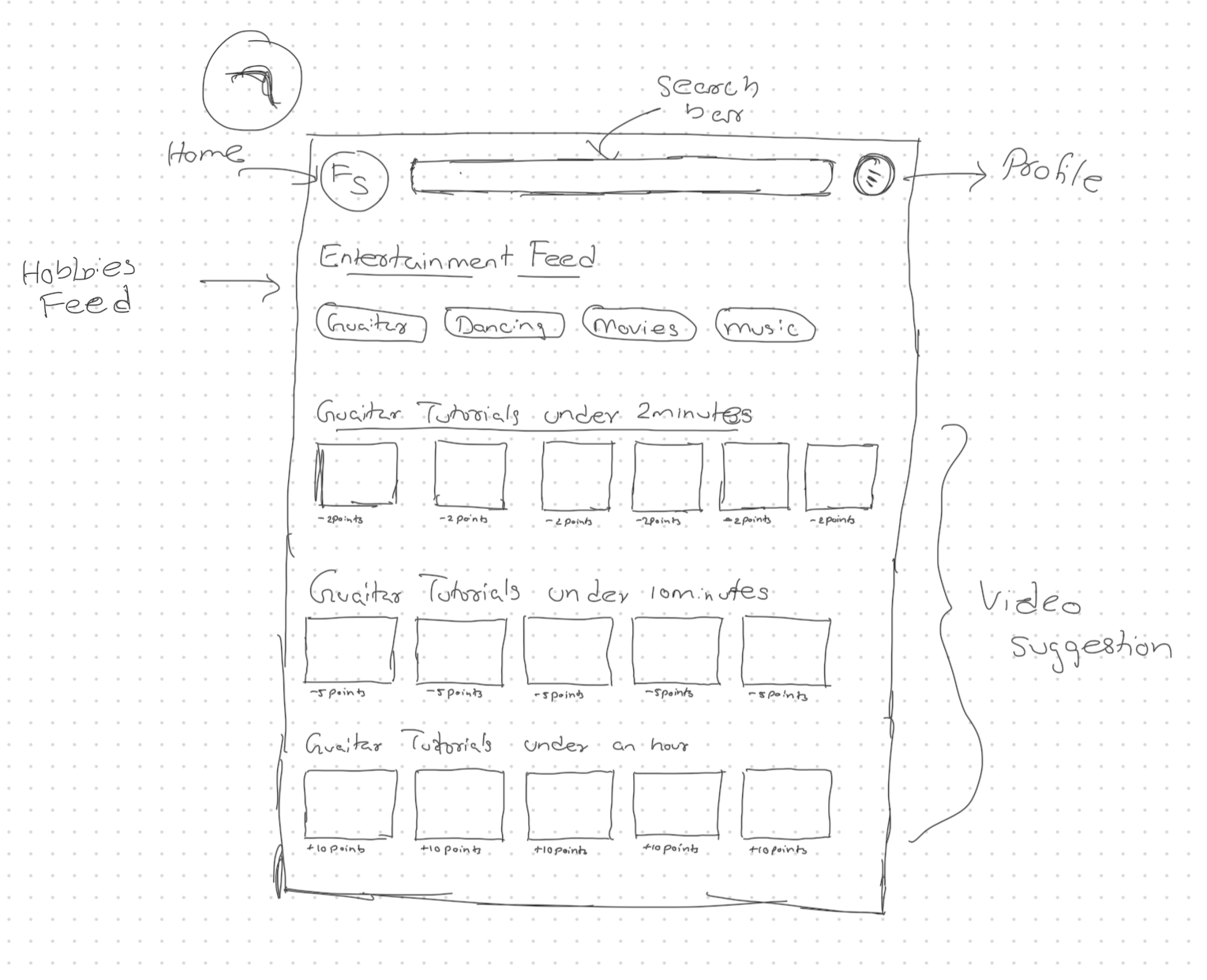
* Separation of Concerns: Each component has a distinct responsibility, making the application modular and easier to manage.
* Flexibility: Changes in one component have minimal impact on others.
* Scalability: New features or functionalities can be added with minimum disruption to existing code.



# Class Diagram

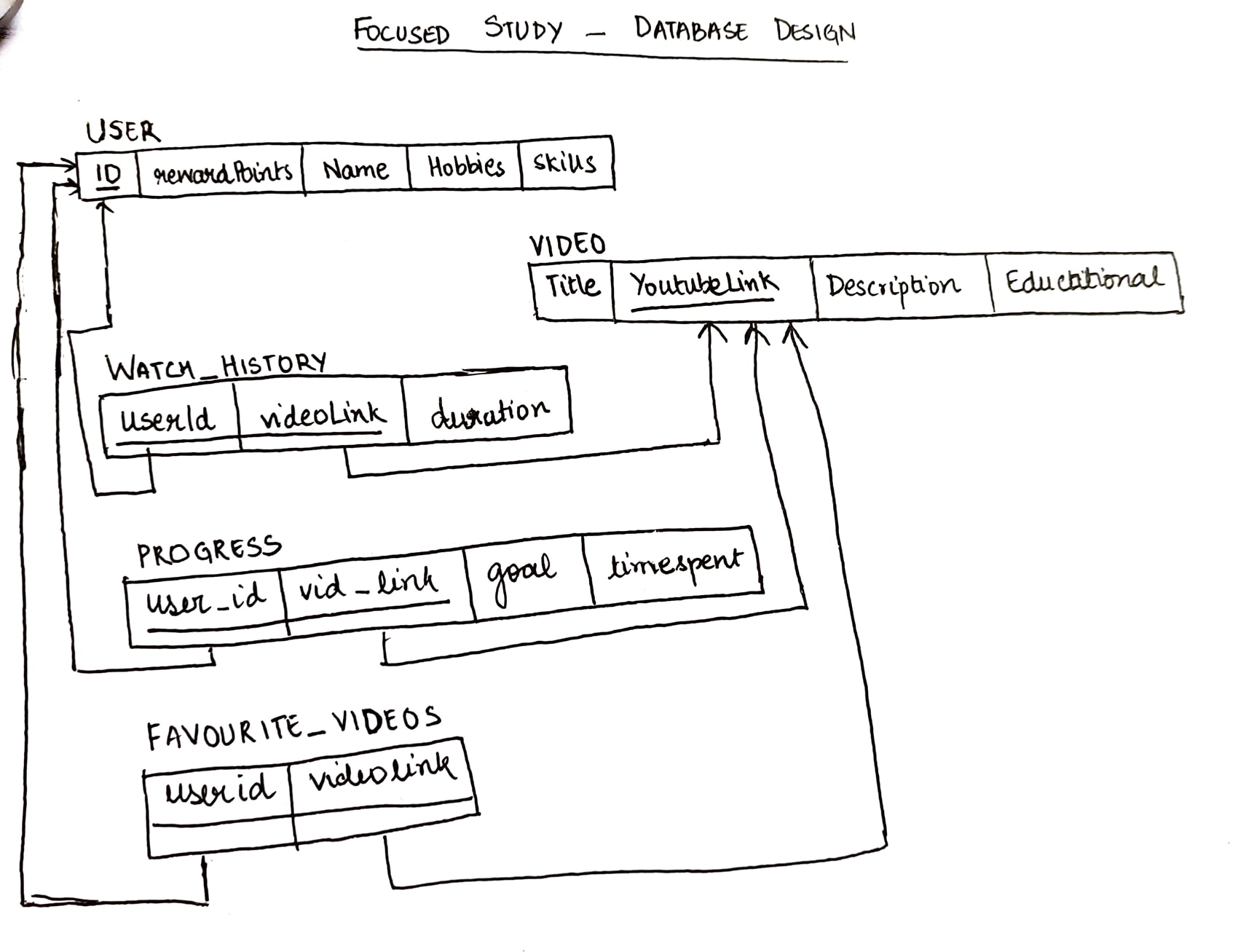


# UI Design (if applicable)

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