**SOFTWARE REQUIREMENT SPECIFICATION**

|  |  |
| --- | --- |
| **Name** | SERVESHWAR R A |
| **Roll no** | 7376221CS306 |
| **Seat no** | 262 |
| **Project ID** | 22 |
| **Problem Statement** | WIKI PAGE GENERATION |

**1**. **Introduction**

**1.1. Purpose:**

The purpose of this document is to present a detailed description of the Wiki Page Generator. It will explain the purpose and features of the system, the UI/UX of the website, what it will do, the constraints under which it must operate and how the website will react to errors or invalid inputs.

**1.2. Scope of Project:**

● This website will solve many issues for Office Academics. They have been using Excel sheets to keep records of the faculties who are meant to upload lecture materials and lecture videos for the wiki page.

* This project will solve these issues for the Admin in Office Academics, The website will also enable faculties to submit their material for approval and receive remarks if rejected. The admin will get the code for our college’s wiki page in MediaWiki markup format so that, it can be directly pasted into the source code of the website.
* Also, the Admin will also get comprehensive reports regarding submissions made by faculties
* Faculties on the other hand will receive remainders regarding due dates ,will be able to see records of their submissions and overview of the points awarded for their submissions.

**2. System Overview:**

**2.1. Users:**

1. **Faculties**:

They can submit materials for approval and receive remarks if their materials get rejected. Get remainders regarding due dates for their submissions and get a report of the points they have obtained for their submissions.

**2. Admins:**

Review submitted lecture videos and materials, approve or reject lecture videos (with remarks), assign points, get generated source code for wiki pages and access analytical dashboards for submission records.

**2.2. Features:**

**1. Login and registration:**

Faculties can login with their existing bitsathy account.

**2. Subjects Assigned:**

Faculties can see the subjects assigned to them and due dates for the final submission. They can submit lecture videos and materials for approval. Upon submission, the uploaded materials are submitted to the admin interface for review and further processing

**3. Submission Status:**

Faculties can view the status of their submissions and see the submission history in the Report Tab.

**4. Report of their submissions:**

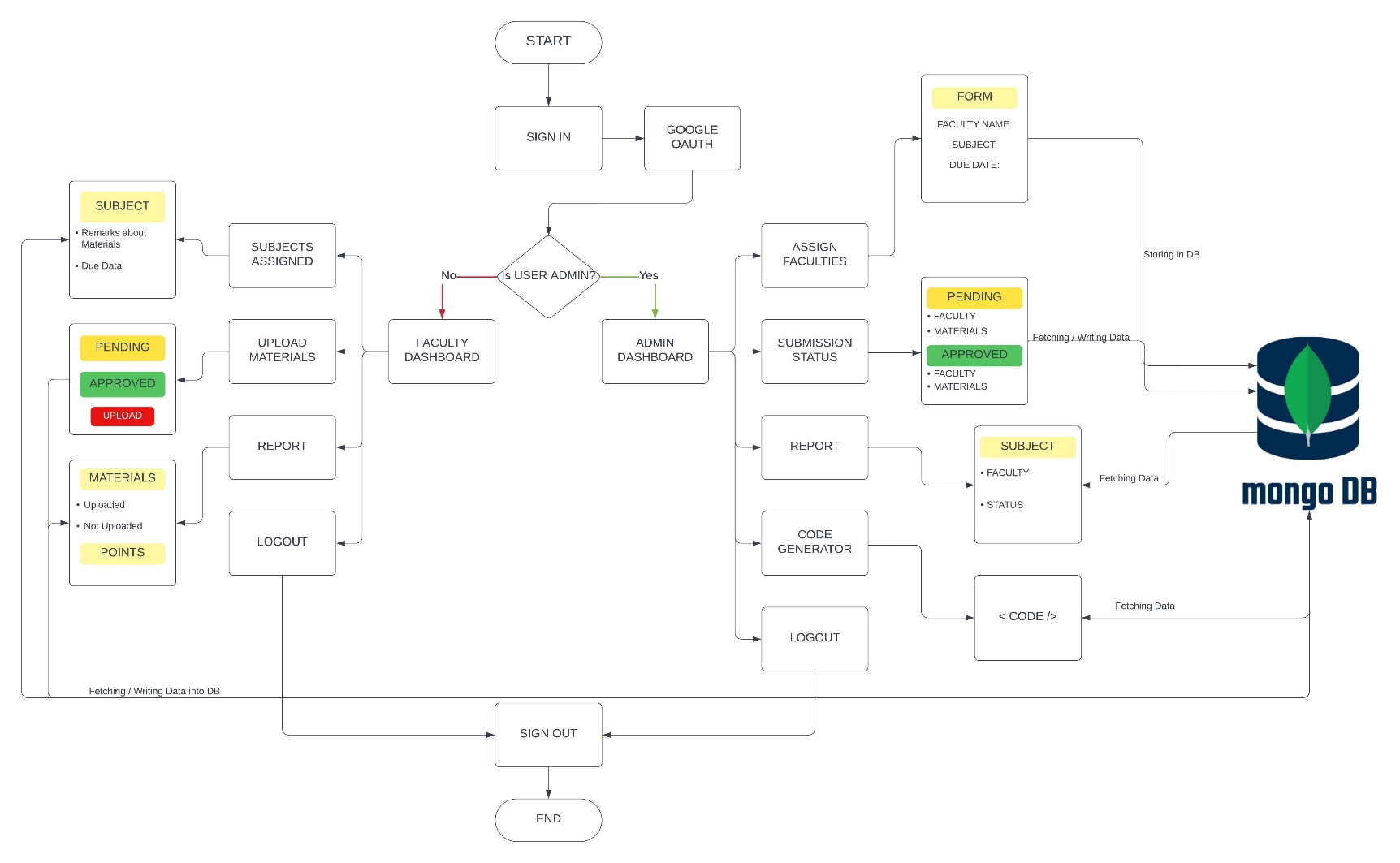
Faculties can see the report of their submissions and points allocated for their timely submissions.

**5. Admin Access:**

Admin can view all submitted materials, view the materials’ details, approve or reject the submission with suitable remarks and award points for that.

**6. Admin’s Analytical Dashboard:**

Admin can view the number of submissions by category, pending approvals and also see the reports of faculties.



**3. Workflow:**

**3.1 Functional Requirements:**

**● User Management:**

○ Faculties can login and submit the materials assigned to them .

○ Admins have access control with an analytical dashboard and dedicated features .

**●Submission of materials:**

○ Faculties can see their submissions along with the details

○ Submission contains:

■ Lecture Video

■ Lecture Material

**● Submission Status:**

○ Faculties can view the current status of their submission

○ If the submission is rejected then the remarks is shown

○ Faculties can also see the points awarded to them

**● Admin Dashboard:**

* Admin can assign subjects to faculties.

○ Admins can view a list of all submitted materials.

* Admins can approve or reject materials with remarks
* Admins can award points for submissions and negative points for submissions that are submitted beyond the due date
* Admin gets access to generated code after all the materials are approved and uploaded.

**3.2. Non-Functional Requirements:**

● **Performance**: The system must respond to user actions within 2 seconds to ensure efficient usability and must handle a concurrent user load of at least 100 users without significant performance degradation.

● **Security**: User data must be encrypted during transmission and storage, and access to sensitive functionalities should be restricted to authorized admin users through secure authentication mechanisms.

● **Usability**: The user interface should be intuitive and user-friendly, with clear and concise error messages provided to guide users in case of input errors or system failures.

● **Reliability**: The system should be available 24/7 with minimal downtime and should have a backup and recovery mechanism in place to prevent data loss in case of system failures or crashes.

● **Scalability**: The system should be designed to accommodate an increasing number of users and data volume over time, and it should be scalable to support additional features and functionalities as per future requirements.

**4.Backend:**

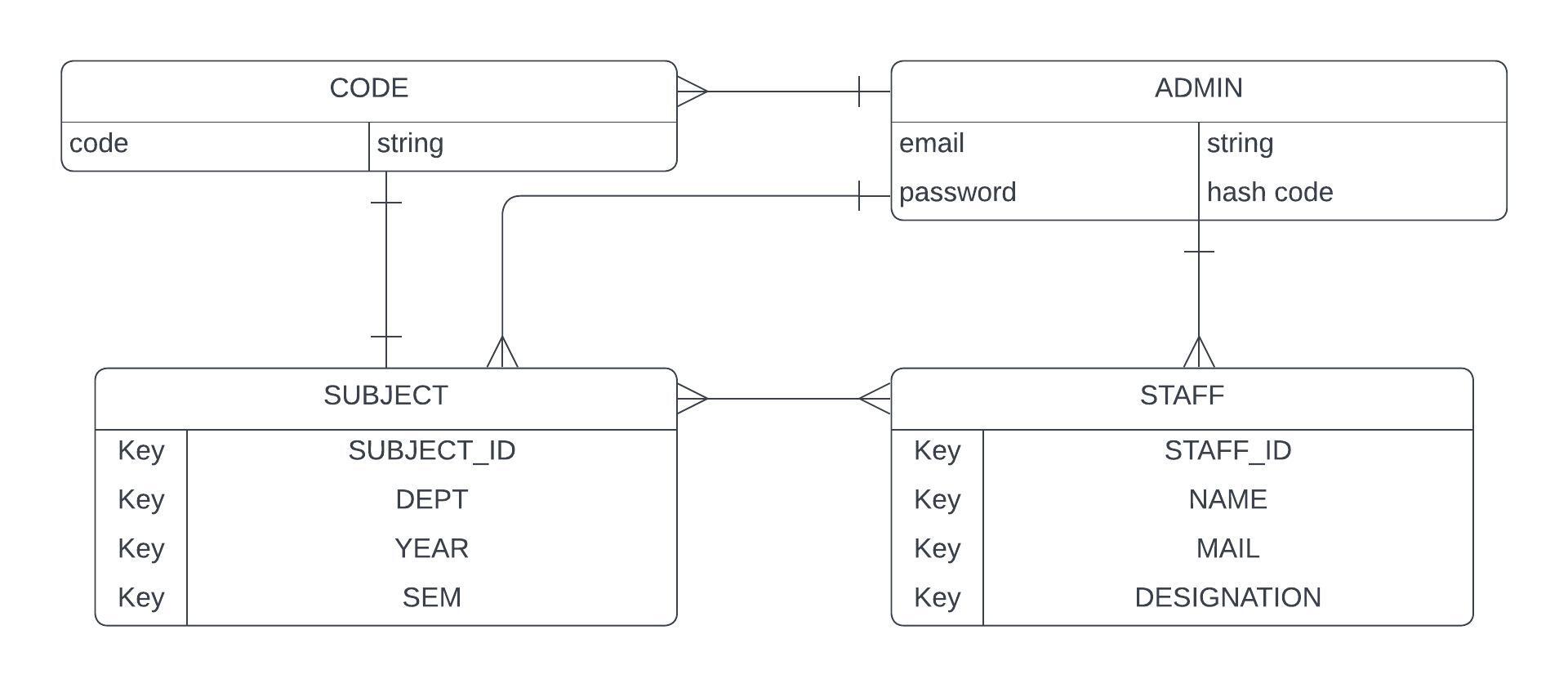
**4.1. Subject entity**

|  |  |
| --- | --- |
| Subject\_ID | INTEGER |
| Dept | STRING |
| Year | INTEGER |
| Sem | STRING |

4.**2. Staff entity**

|  |  |
| --- | --- |
| Staff\_ID | STRING |
| Name | STRING |
| Mail | STRING |
| Designation | STRING |

**4.3. ER DIAGRAM:**



**5. Stack:**

|  |  |
| --- | --- |
| Front End | Vue Js, Bootstrap CSS |
| Backend | Node Js, Express |
| Data Base | MongoDB |
| Authentication | Google OAuth 2.0 |