Engineering Study Hub

YOUR PATH TO COMPUTER SCIENCE MASTERY

Team ESH

Software Requirements Specification (SRS)

Engineering Study Hub

1. Introduction

1.1 Purpose

This document defines the functional and non-functional requirements for the Engineering Study Hub, a web platform that enables universities to upload and categorize study materials (videos, PDFs) by semester and year and provides quizzes for students to test their knowledge. It outlines each function of the system, including inputs, processes, and outputs.

1.2 Scope

The Engineering Study Hub will allow university administrators to upload study materials, and students can access these resources based on their semester. The platform will also feature quizzes to test student knowledge, and both students and university admins can register, log in, and manage their profiles.

2. Overall Description

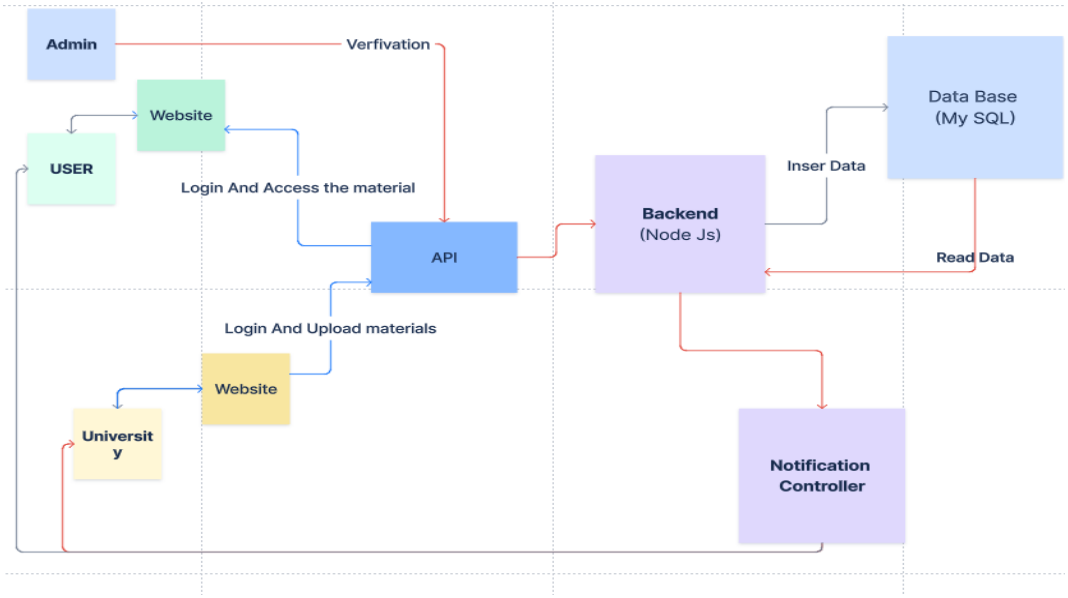
2.1 Product Features

- User Registration and Login
- Study Material Upload (Videos, PDFs)
- Study Material Access (Search by semester, year, etc.)
- Quiz Management (Create, Attempt, View Results)
- User Profile Management (Admin and Student)

2.2 Technologies Used

- Frontend: HTML, CSS, JavaScript
- Backend: Node.js, Express.js
- Database: MongoDB

2.3 System Architecture



---

3. Functional Requirements

Each functional requirement is described below, along with inputs, processes, and outputs.

3.0 Website Pages :

1. Login Page

Purpose: To authenticate users (universities and students) before granting access to the platform.

Process:

- **User Selection**:
  - Two buttons for user types: **University** and **Student**.
  - On selection, the page can either refresh or redirect to a login form tailored to the selected user type.
- **Login Form**:
  - **For Students**:
    - Input fields:
      - **Email/Username**: For student identification.

        - **Password**: For authentication.
    - Button: **Login** - validates credentials against the database.
- **For Universities**:
    - Input fields:
        - **University ID/Email**: Unique identifier for the university.
        - **Password**: For authentication.
    - Button: **Login** - validates credentials against the database.

- **Error Handling**:
    - Display error messages if the login fails (e.g., incorrect credentials).
- **Navigation Links**:
    - Links to **Forgot Password?** and **Sign Up** (if applicable).

2. Home Page

Purpose: To display a collection of videos organized by semester and subjects.

Process:

- **Video Library Section**:
    - Display a grid or list of videos with thumbnails, titles, and brief descriptions.
    - Each video can be categorized by semester or subject for easy navigation.
- **Search Functionality**:
    - A search bar to allow users to quickly find specific videos.
- **Filters**:
    - Option to filter videos by semester, subject, or type (e.g., lectures, tutorials).
- **Video Playback**:
    - Clicking on a video thumbnail opens a modal or redirects to a dedicated video player page.
- **User Interactions**:
    - Option for students to bookmark or save videos for later viewing.
    - Buttons to share videos via social media or direct links.

3. Notes Page

Purpose: To provide access to PDF notes and resources related to the videos.

Process:

- **Notes List**:
    - Display a list of available notes categorized by semester and subject.
    - Each entry should include a title, brief description, and download link.
- **Search and Filter Options**:
    - Implement a search bar and filters similar to the home page for easy navigation.
- **Download Functionality**:
    - Clicking on a note link should prompt the user to download the PDF file.
- **Preview Option**:
    - Consider implementing a preview feature that allows users to view the notes in the browser before downloading.

4. Quiz Page

Purpose: To engage students with interactive quizzes to reinforce learning.

Process:

- **Quiz List**:
    - Display a list of available quizzes categorized by subject and difficulty.
- **Quiz Start Button**:
    - Each quiz entry has a button to start the quiz.
    - Clicking the button redirects the user to the quiz interface.
- **Quiz Interface**:
    - Display questions one at a time or all at once, depending on your design.
    - Options for multiple-choice answers or short answers.
    - A timer to track the time taken for each quiz.

- **Submission**:

    - A **Submit Quiz** button that checks answers against correct solutions and provides instant feedback.
    - Option to review answers after submission, displaying correct answers and explanations.
- **Score Tracking**:
    - Save the user's score and progress in their profile for future reference.

5. University Upload Page

Purpose: To allow universities to upload and manage their study materials.

Process:

- **Upload Form**:
    - Input fields for university administrators to enter the title, description, and category of the study material.
    - File upload option for PDFs, videos, or other resources.
- **Category Selection**:
    - Dropdowns or checkboxes to select the relevant semester and subject for the uploaded materials.
- **Submit Button**:
    - A button to submit the form, which will store the information in the database.
    - Upon successful upload, display a confirmation message or redirect to a materials management page.
- **Material Management**:
    - A section to view previously uploaded materials, with options to edit or delete entries.
    - Filters to search and sort materials based on title, date, or category.

General Considerations

- **User Authentication**: Ensure secure handling of user data during login and uploads.
- **Database Structure**: Organize your database to efficiently manage users, videos, notes, quizzes, and uploads.

- **Responsive Design**: Make sure all pages are responsive for various devices (desktop, tablet, mobile).
- **User Experience**: Keep the interface clean and intuitive, making it easy for both students and universities to navigate the platform.

3.1: Input-Output Process :

3.1 User Registration

- **Function**: Register new users (students and admins).
- **Input**:
    - Username
    - Email
    - Password
    - Role (student or university admin)
- **Process**:

1. The user submits the registration form.
2. The backend validates the inputs.
3. Password is hashed using bcrypt.js.
4. User details are stored in MongoDB.

- **Output**:
    - Success message if registration is successful.
    - Error message if email already exists or validation fails.

3.2 User Login

- **Function**: Authenticate users using email and password.
- **Input**:
    - Email

- Password

- **Process**:

1. The user submits the login form.
2. The backend checks the database for the user's email.
3. On success, a JWT token is generated and sent back to the user.

- **Output**:
  - JWT token for successful login.
  - Error message for invalid credentials.

3.3 Upload Study Materials (Videos, PDFs)

- **Function**: University admins upload study materials categorized by semester and year.
- **Input**:
  - Title
  - Type (PDF or Video)
  - Semester
  - Year
  - File (Video or PDF)
- **Process**:

1. Admin submits the upload form.
2. Multer handles the file upload.
3. Metadata (title, type, semester, year, file URL) is saved to MongoDB.

- **Output**:
  - Success message and confirmation of file upload.
  - Error message if file upload fails.

3.4 View Study Materials

- **Function**: Allow students to search and view study materials by semester and year.
- **Input**:
    - Semester
    - Year
- **Process**:

1. User selects semester and year or enters a search keyword.
2. Backend queries the MongoDB database for matching materials.
3. The list of matching materials (with URLs) is returned to the front end.

- **Output**:
    - List of study materials (PDFs, videos) for the selected semester and year.
    - Error message if no materials are found.

3.5 Create Quiz

- **Function**: University admins create quizzes with questions for students.
- **Input**:
    - Title
    - Semester
    - List of Questions (with multiple choice options and correct answers)
- **Process**:

1. Admin enters quiz details and submits.
2. Quiz details are stored in the MongoDB Quiz collection.

- **Output**:
    - Success message when quiz is successfully created.
    - Error message for validation errors.

3.6 Attempt Quiz

- **Function**: Students attempt quizzes related to their semester.
- **Input**:
    - Quiz ID
    - Student's selected answers for each question
- **Process**:

1. Student selects a quiz and submits their answers.
2. Backend calculates the score based on correct answers.
3. The score is stored in the database with the student's ID.

- **Output**:
    - Score after quiz completion.
    - Feedback on correct/wrong answers.
    - Error message if the quiz is invalid or expired.

3.7 View Quiz Results

- **Function**: Students can view their past quiz results.
- **Input**:
    - User ID
- **Process**:

1. Backend fetches the user's quiz results from the Quiz Results collection.
2. Results are returned to the frontend.

- **Output**:
    - List of past quiz attempts and scores.

3.8 Profile Management

- **Function**: Users can view and update their profile information.

- **Input**:
    - Updated Username, Email, or Password
- **Process**:

1. User submits updated information.
2. Backend validates inputs and updates the MongoDB User collection.
3. Password is re-hashed if changed.

- **Output**:
    - Success message if the update is successful.
    - Error message if validation fails.

---

4. Non-Functional Requirements

4.1 Security

- Passwords will be stored using bcrypt.js for hashing.
- JWT will be used for user authentication and authorization.

4.2 Performance

- The system should handle concurrent uploads and access by multiple students.
- MongoDB indexing will be used to optimize search and retrieval performance.

4.3 Usability

- The user interface should be simple and intuitive, with clear navigation for uploading and accessing study materials and quizzes.

4.4 Scalability

- The platform should be scalable to accommodate multiple universities and a growing number of students and study materials.

---

5. Database Design

The project will use **MongoDB** for storing user data, study materials, and quiz-related information.

5.1 Collections

1. **User Collection**:

- User ID
- username
- email
- password
- role (admin or student)

2. **Study Material Collection**:
    - Material ID
    - title
    - type (PDF or Video)
    - semester
    - year
    - file URL
3. **Quiz Collection**:
    - Quiz ID
    - title
    - semester
    - questions (Array of questions and correct answers)

4. **Quiz Results Collection**:
    - Result ID
    - User ID
    - Quiz ID
    - score
    - date Attempted

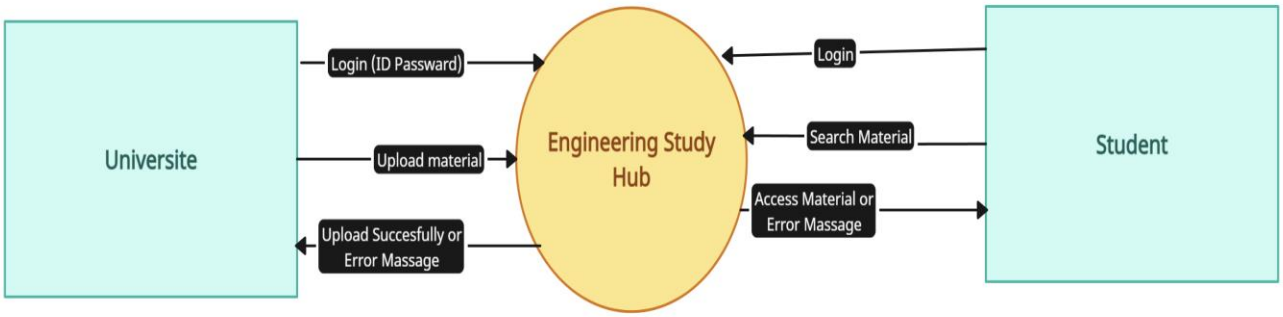
---

6. Conclusion

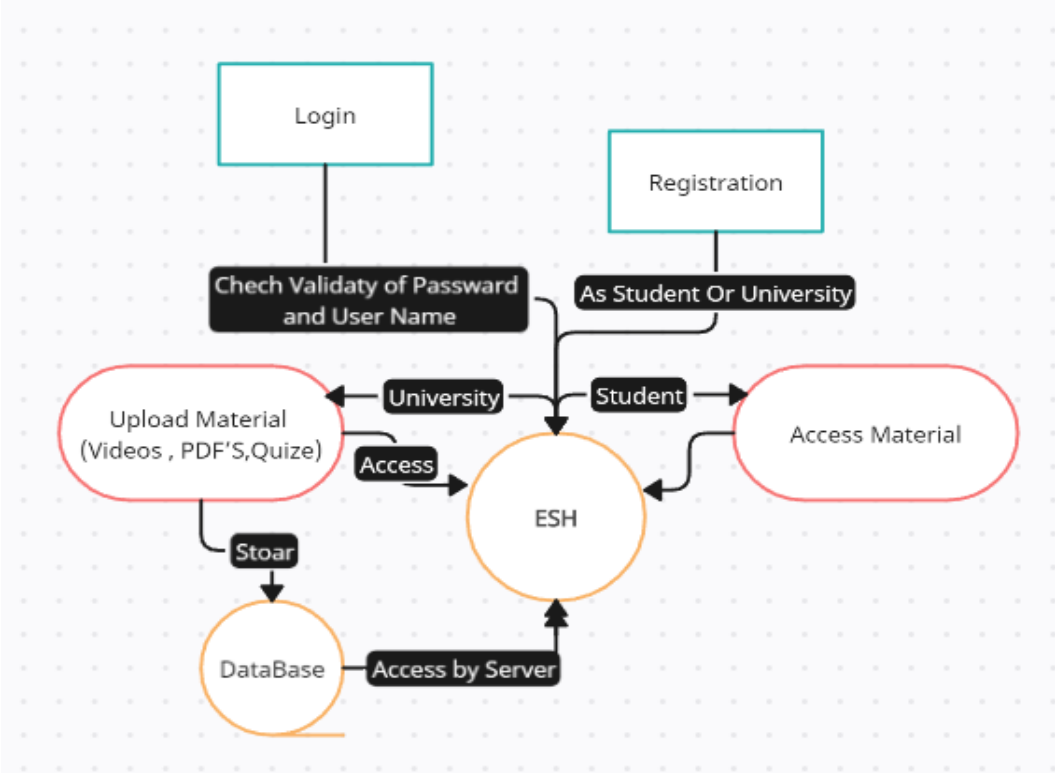
This SRS document outlines all the required functionalities, processes, and inputs/outputs for the Engineering Study Hub. By following this specification, you'll be able to develop a fully functional platform that allows students and university administrators to manage study materials and quizzes effectively.

Data Flow Diagram (DFD):

1) Data Flow Diagram (DFD) – Level 0 (Context Level) :



2) Data Flow Diagram (DFD) – Level 1



Data Flow Diagram (DFD) – Level 2 :