Software Requirements Specification

for

Sahara

Version 1.2

Prepared by

Group: 8 **Group Name: Ravi and Friends**

Pothuganti Nikhil	230760	nikhilp23@iitk.ac.in
Divi Pothukuchi	230378	divip23@iitk.ac.in
Aarav Oswal	230012	aaravoswal23@iitk.ac.in
Ravi Arora	230846	raviarora23@iitk.ac.in
Achyuth Warrier	230052	achyuthw23@iitk.ac.in
Ashik Stenny	230224	ashiks23@iitk.ac.in
Harshpreet Kaur	230464	harshpreet23@iitk.ac.in
R.Charan	230819	rcharan23@iitk.ac.in
Spandan Pati	231031	spandanp23@iitk.ac.in
Vedhanth Balasubramaniam	231135	vedhanthb23@iitk.ac.in

Course: CS253

Mentor TA: Prof. Indranil Saha

Date: 24/01/2025

Table of Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Ravi and Friends	The initial draft was prepared.	21/01/25
1.1	Ravi and Friends	The initial draft was modified based on input from Prof. Indranil Saha.	23/01/25
1.2	Ravi and Friends	The draft was reviewed and final changes were made.	24/01/25

1 Introduction

1.1 Product Scope

Our product aims to reduce the workload of professors and course instructors, by providing a single platform for conduction of the course CS253 and other group-project centric courses, unleashing the power of cutting-edge AI tools to assist with assessment, and providing relevant features to streamline the logistics for the course. The major benefits are:

- **Streamlined Course Management**: By integrating all essential tools in one platform, instructors can easily manage announcements, schedules and deadlines without juggling multiple systems.
- **Efficient Assessment and Grading**: Leveraging advanced AI tools, the platform simplifies the grading process, including evaluating project reports, and even requirement documents using LLM-based assessments, saving valuable time for instructors.
- **Enhanced Collaboration**: Students benefit from features like group and project registration, team-only discussion forums, and git-based project tracking, which promote effective collaboration and monitor contributions seamlessly.
- **Real-Time Progress Monitoring**: Progress heatmaps and milestone management tools provide instructors and students with clear insights into team activities and deadlines, ensuring that projects stay on track.
- **Management of TAs:** Instructors can leverage pre-designed form templates to gather and organize information about the diverse skill sets of TAs. This information is then utilized to efficiently allocate TAs to project groups, aligning their expertise with the specific requirements of each group.

By centralizing these features, our product transforms the way instructors and students approach group-project courses, making the entire process more efficient, collaborative, and impactful.

1.2 Intended Audience and Document Overview

1.2.1. Document Overview

This document consists of 4 Sections:

- **Introduction** - Provision of basic terminologies and information used that will help the reader understand the document better. Can be skipped if the reader is familiar with the terminology.
- **Overall Description** - Offers precise and concise information regarding the functionality, interface, assumptions and dependencies of the software. Recommended to be viewed by all readers.
- **Specific Requirements** - Offers a detailed description of the software, its functionality, interfaces and working by dividing the entire user experience into multiple use cases.

- **Non-Functional Requirements** - Describes the performance, security and other qualities that are essential for the safety and smooth running of the software.

1.2.2. Intended Audience

This document is designed for:

- **Software Development Team** that will design and construct this software as per the requirements of the client i.e. professors, teaching assistants, and students.
  Important Sections: Product Overview (2.1) → Product Functionality (2.2) → Functional Requirements (3.2) → Use Case Model (3.3) → Non - Functional Requirements (4.1, 4.2)

- **Project Managers** i.e. the Teaching Assistants and Course Instructor, to help them overlook the planning, development and execution of the software at every step.
  Important Sections: Sections 2, 3 and 4.

- **Testers**, who perform product quality checks. In this case, beta-testing will be carried out by another group.
  Important Sections: Design and Implementation Constraints (2.2) → Assumptions and Dependencies (2.3) → External Interface Requirements (3.1) → Functional Requirements (3.2) → Complete Section 4

- **End Users** i.e. students, professors and TAs who will be finally using the product and giving feedback.
  Important Sections: Product Overview (2.1) → Product Functionality (2.2) → Design and Implementation Constraints (2.3)

1.3 Definitions, Acronyms and Abbreviations

- **AI** - Artificial Intelligence
- **CPU** - Central Processing Unit
- **HTTPS** - Hypertext Transfer Protocol Secure
- **ID** - Identification/ Identity
- **LLM** - Large Language Model
- **MBPS** - Megabytes per second
- **MFA** - Multi-Factor Authentication
- **MTTF** - Mean Time To Failure
- **OTP** - One Time Password
- **PII**- Personally Identifiable Information
- **Prof**- Professor
- **RAM** - Random Access Memory
- **RBAC** - Role Based Access Control
- **SRS** - Software Requirement Specifications
- **TA** - Teaching Assistant
- **UI** - User Interface

1.4 Document Conventions

The document follows the IEEE formatting requirements

- The heading*s* of all the sections are written in **bold** *and* underlined using Arial font size 14.
- The headings of subsections are **bold** and written in the same font as that of content using font size 12.
- The content of the section is written in Arial font size 11.
- Document is single - spaced.
- 1” margins are maintained.
- Bullet point ordering has been used as a list - setting type tool.

1.5 References and Acknowledgments

References:

Style Guide (for this document) : IEEE Software Requirements Specification Template.
Moqups and Canva were used for designing the UI and Use Case Models.
Lucidchart was used for designing the flowcharts.

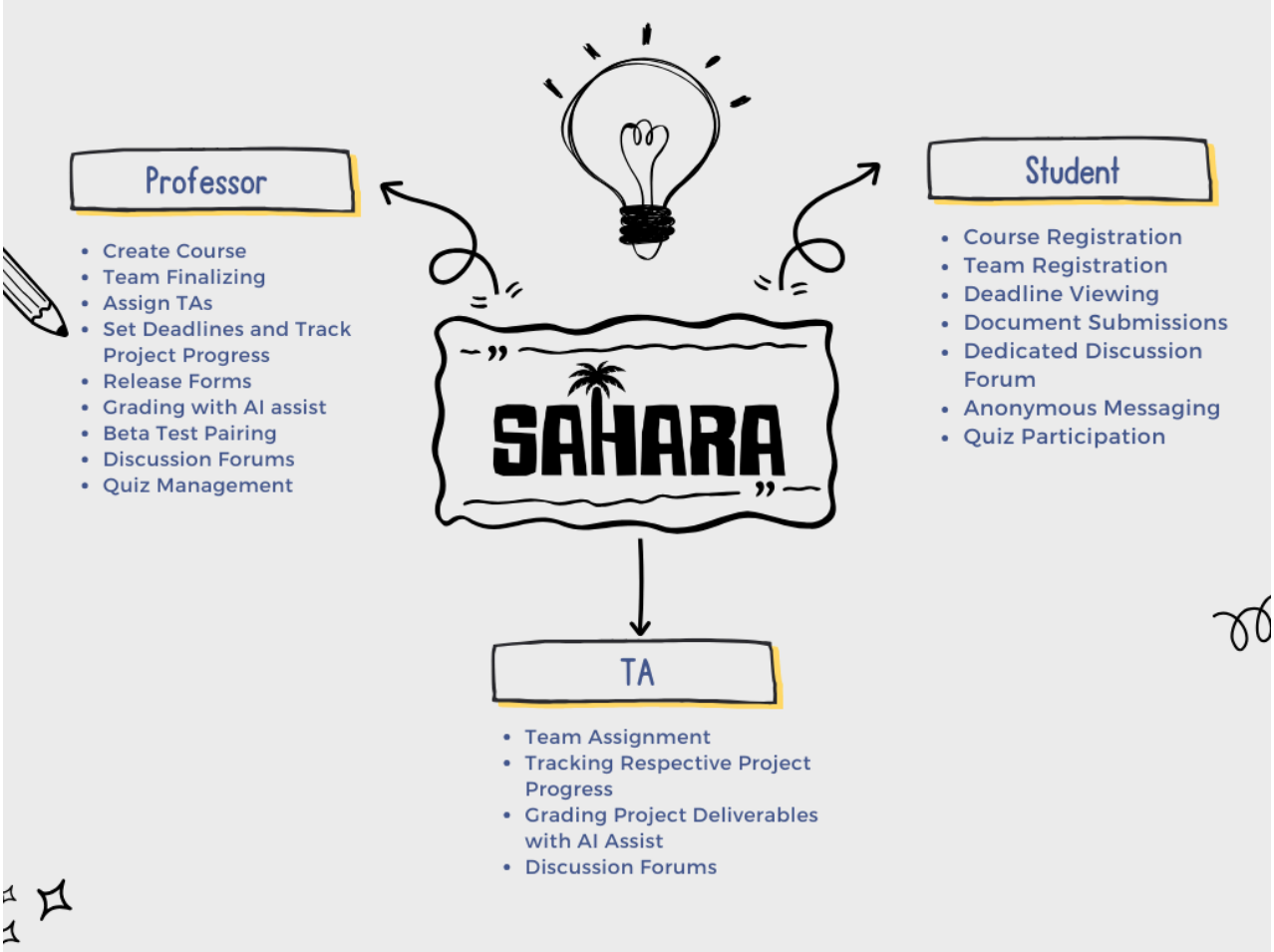
Acknowledgements:

The fundamental principles of our system design were established through the lectures and mentorship provided by Professor Dr. Indranil Saha.

2 Overall Description

2.1 Product Overview

The product aims to be a new self-contained product replacing the tedious tasks of managing a course conducting a group project (intended more specifically for CS253), which currently requires the professor to manage the course via inefficient methods like using Google Forms for creation of teams, managing submissions, etc., having to resort to mail for team-specific threads. This becomes an incredibly tedious task for both the instructor as well as the student, due to the lack of an organised and streamlined platform dedicated for this process. As a standalone product, this product aims to address the need for a solution dedicated to conducting such group-project centric courses. The architecture and design of the software allow it to serve as a centralised portal for a hassle-free and smooth course conduction process.



2.2 Product Functionality

1. Professor Interface Functionalities:

    - Adding teams and assigning TAs to teams
    - Track Team Progress on Projects
    - Grading Project related documents with AI Assist
    - Beta test pairing assist
    - Access to discussion forum with team and TA
    - Provides section for announcements
    - TA-team pairing assistance
    - The Professor would be able to release customizable forms

2. Student Interface Functionalities:

    - Receive Announcements from TA/Professor
    - Provides an interface for participating in quizzes
    - Event Calendar Showing Project Deadlines
    - Provides a portal for submission of deliverables
    - Displays the grades for the submissions
    - Access to Multiple Discussion Forum Functionality

3. TA Interface Functionalities:

    - Variable functionality access depending on Professor discretion
        - Tracking Project Progress
        - Grading Project docs
    - Access to Discussion Portal with Team and Professor

2.3 Design and Implementation Constraints

Hardware Requirements:

- A functional computer or mobile device with a stable internet connection would be sufficient to access the application.
- Mobile devices must have a minimum of **2 GB of available RAM** and an internet speed of at least **0.1 Mbps** to ensure the application opens smoothly.

Software Requirements:

- The application is compatible with any devices and operating systems supporting modern browsers such as **Google Chrome**, **Mozilla Firefox**, **Microsoft Edge**, **Internet Explorer, Safari**, or **Opera**, and works better on latest versions of these browsers.

2.4 Assumptions and Dependencies

Assumptions:

- The maximum users registered on the platform will be at least 1000.
- The course instructor account cannot be de-registered from the platform.
- The application is not used by more than 50 concurrent users.
- The project size on Git is under 2GB.

Dependencies:

- ReactJS - Javascript based framework for web-app development
- NextJS - React based framework for better React applications
- NodeJS - Javascript support engine for server-side and client-side development
- Git - using Git and Git server for developing project management tools
- TailwindCSS - CSS framework for developing modern responsive user interfaces
- Material UI - Ready to use component provider for creating user interfaces
- MongoDB - Database for usage in the backend

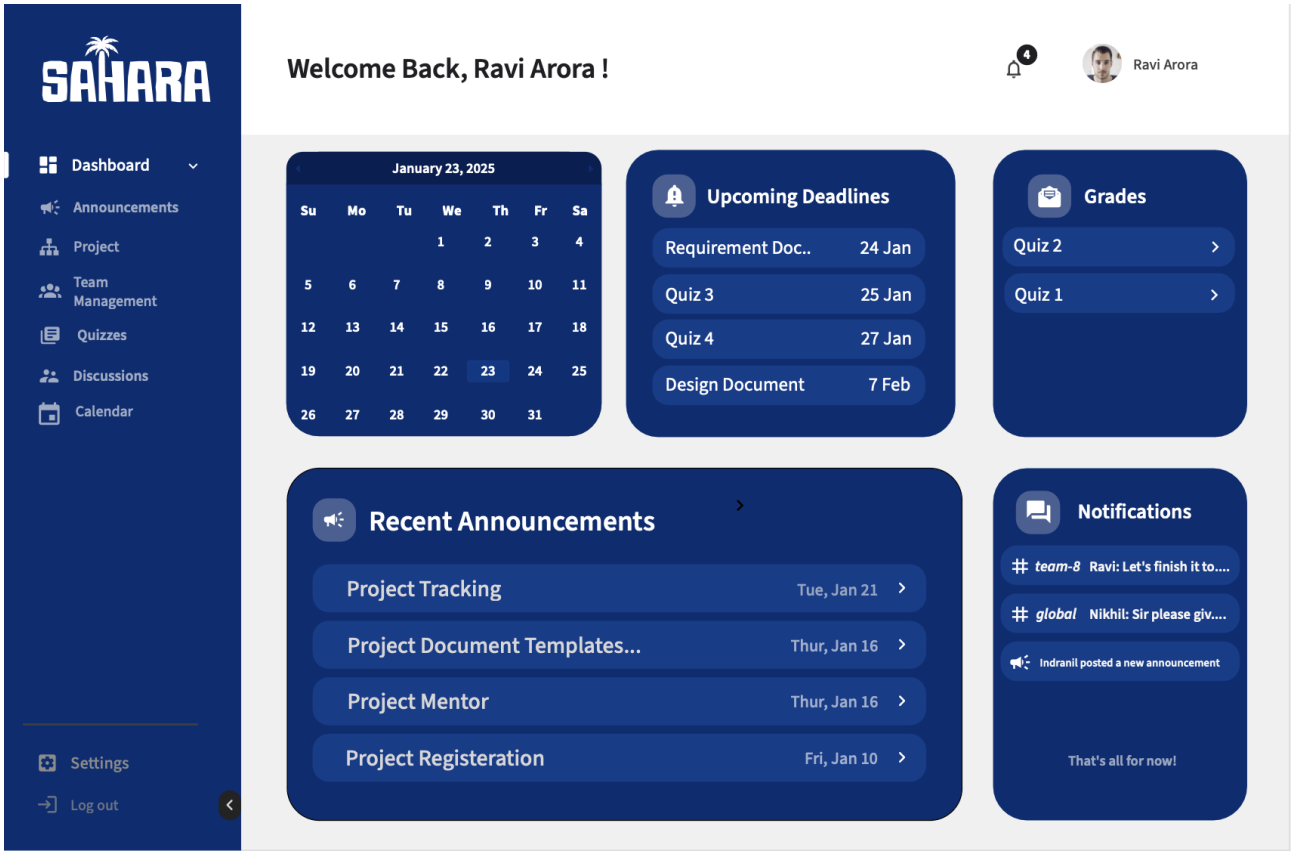
3 Specific Requirements

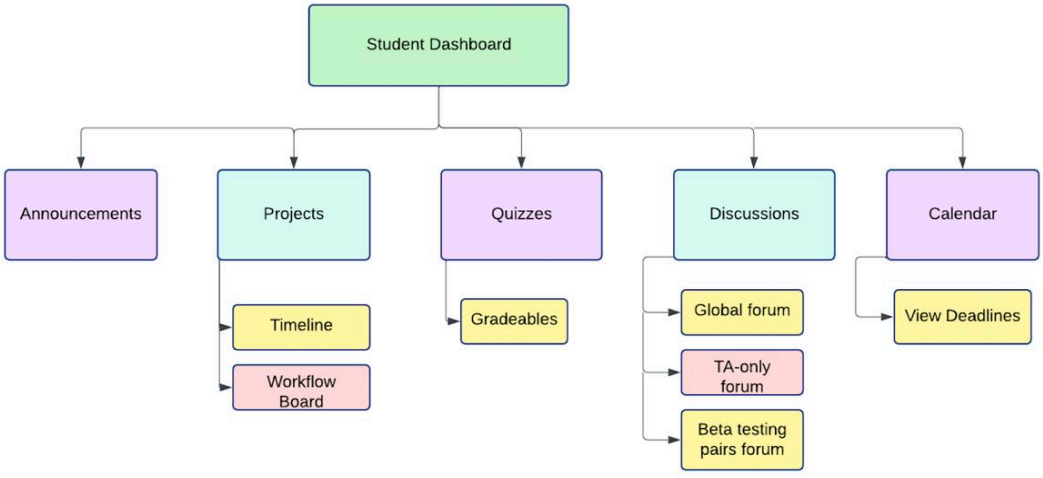
3.1 External Interface Requirements

3.1.1 User Interfaces

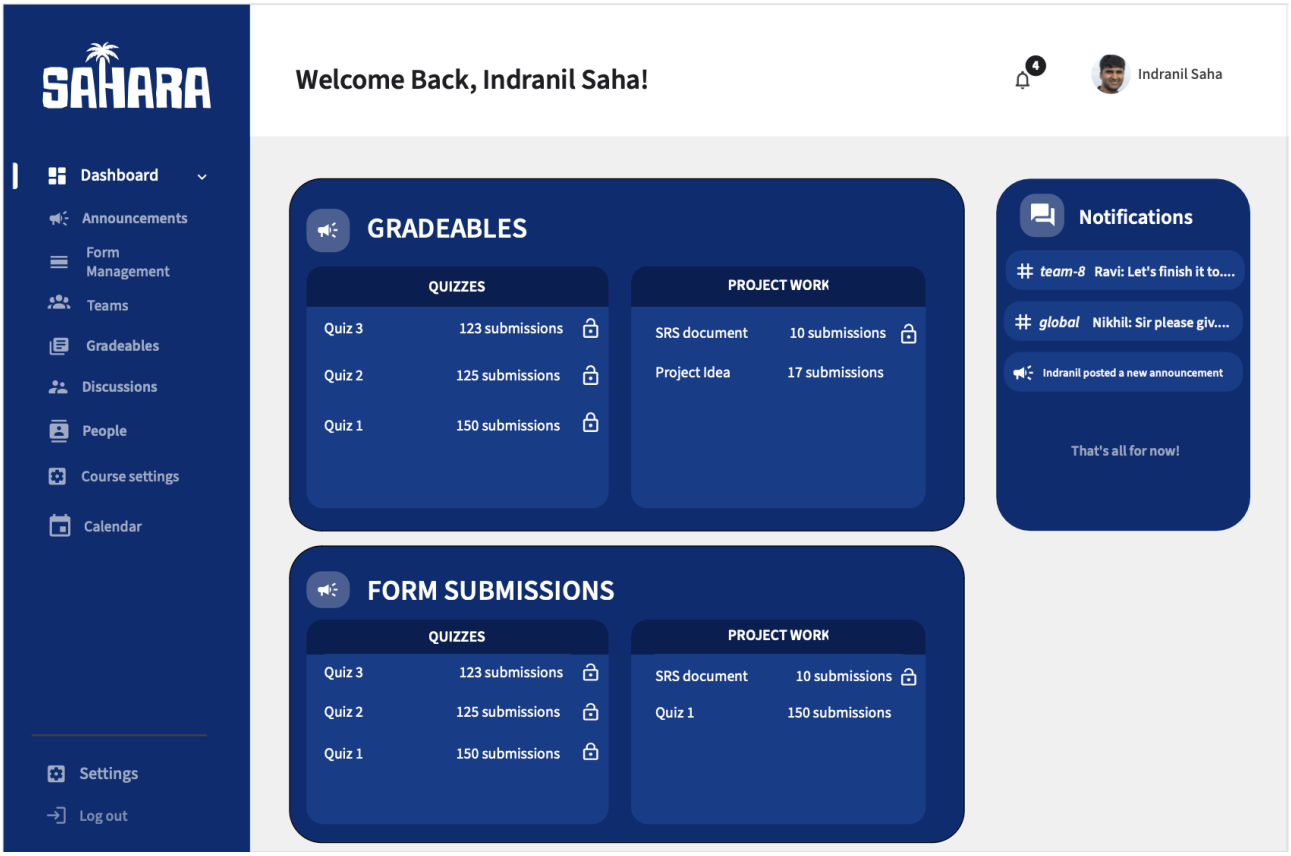
The application will be used by **three** different categories of users.

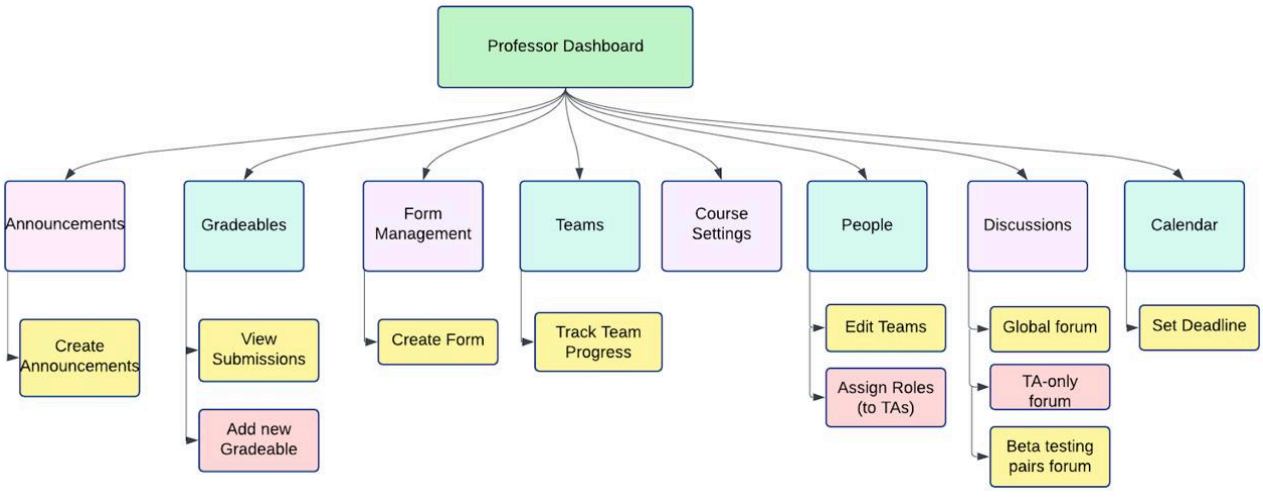
Student Dashboard



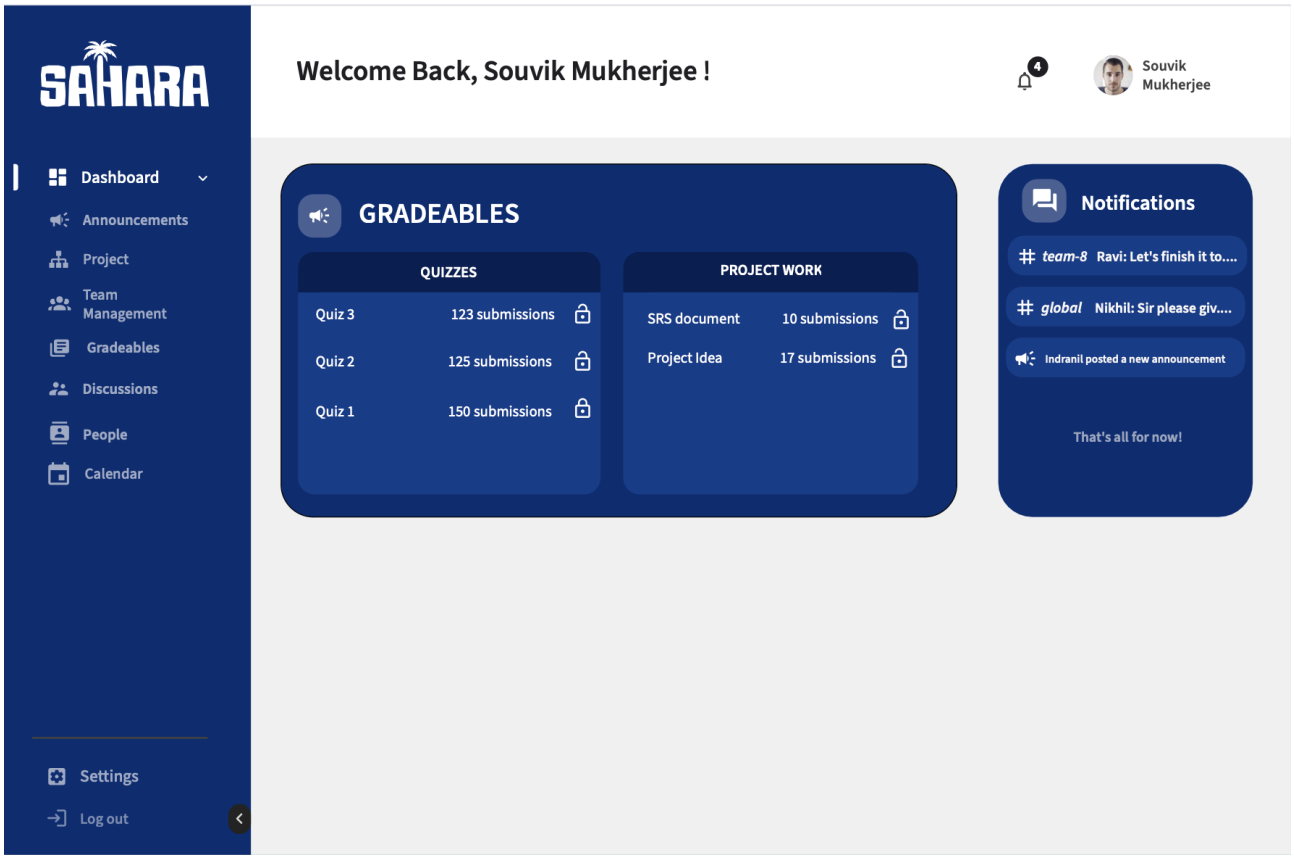


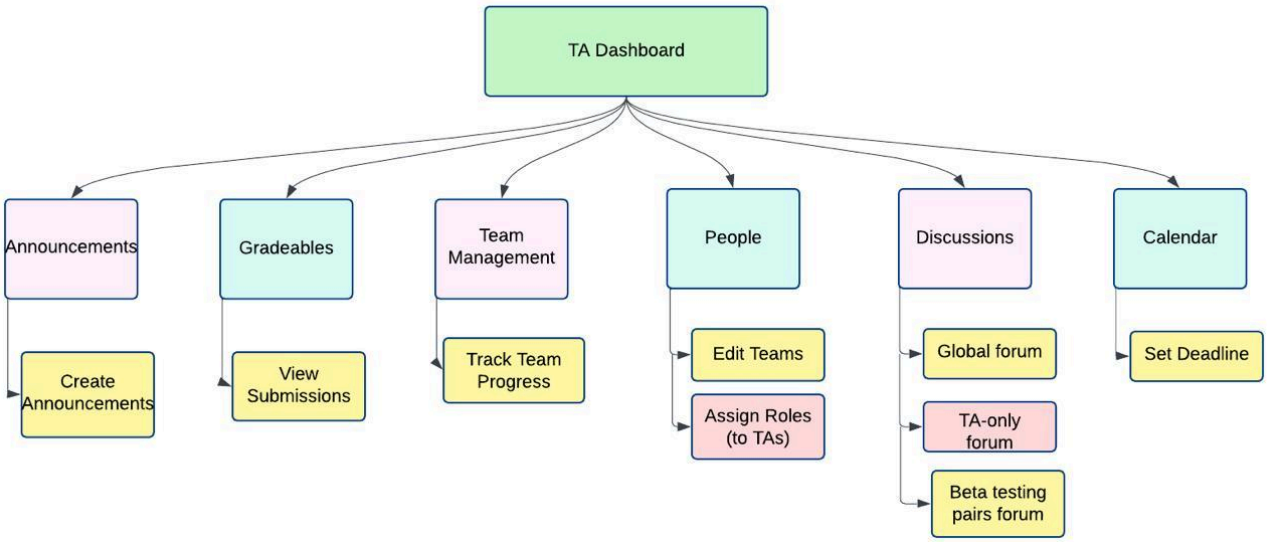
Professor Dashboard





TA Dashboard





3.1.2 Hardware Interfaces

Client Devices:

- **Supported Device Types:**
    - Desktop Computers (Windows, macOS, Linux)
    - Laptops (Windows, macOS, Linux)
    - Smartphones and Tablets (iOS, Android)
- **Nature of Interaction:**
    - The website will send and receive data via standard internet protocols (e.g., HTTPS).
    - Responsive design ensures the website adapts to varying screen sizes and resolutions.

Web Servers:

- **Logical Characteristics:** The website will interface with the web server to host content and handle HTTP/HTTPS requests.
- **Physical Characteristics:** Compatible with servers running modern operating systems (Linux, Windows) equipped with sufficient CPU, RAM, and storage to handle expected traffic loads.

3.1.3 Software Interfaces

- The client-side components of the software must be able to operate seamlessly with various modern web browsers like Apple Safari 7+, Google Chrome 44+, Microsoft Edge 90+, Mozilla Firefox 40+.
- For ensuring authenticity of users the software shall verify credentials with the institute database.
- We will use databases for storing the following: -
    - User login credentials
    - Team details
    - Details of the forms published by the professor.
    - Quiz results and statistics and any other data.

MongoDB will be used as the database used in the backend.

3.2 Functional Requirements

3.2.1 Team Registration

- The system will automate the team registration process, managed by the administrator. The administrator can predefine team specifics, such as size and member demographics.
- Team leaders will be able to create teams and invite members via invite codes or on-platform notifications, which members can accept or decline. Members also have the option to be auto-assigned into a team.
- At the end of the team registration period, the system would show all the teams to the administrator, also indicating the teams with less than required members and the users with no assigned team. The administrator would then be able to assign teams to the unassigned users.

3.2.2 Team TA assignment

- Teams can tag their profiles with keywords reflecting their tech stack or project domain (e.g., "AI," "Blockchain," or "Mobile Development"). TAs can similarly tag their areas of expertise.
- The system will use these tags, along with administrator-defined prompts, to generate AI-based suggestions for matching TAs to teams. The administrator can review and manually finalize these pairings to ensure the best alignment between TAs and teams.

3.2.3 Project Timeline Management

- The administrator will be able to define deadlines for various project deliverables, such as presentations, or final reports. These deadlines will be displayed on a shared platform calendar accessible to all users.
- Users can customize their calendars to add personal deadlines or events, helping them manage both team and individual schedules. Professors will be able to monitor team progress and track deliverables in real time.

3.2.4 Git-based Project Tracking

- Each team will have access to an integrated Git-based version control system on the platform. This feature will be mandatory for all project work, ensuring centralized tracking of updates and changes.
- Key functionalities include:
    - **Progress Monitoring**: The system will allow administrators and professors to track project activity and ensure teams are making steady progress.
    - **Heatmaps**: Visual heatmaps will highlight contributions at both the team and individual levels, enabling better visibility into participation and workload distribution.
    - **File Sharing**: During the beta testing phase, project files will be shared automatically with paired teams, streamlining collaboration.
- This system ensures transparency, promotes accountability, and simplifies evaluation.

3.2.5 Assisted Grading

- The system will utilize AI to analyze project documents submitted by students. These documents will be assessed using predefined criteria set by the instructor, such as clarity, technical depth, completeness, adherence to guidelines, and alignment with project objectives.
- The AI will provide detailed, actionable feedback for each document, identifying areas of strength and improvement. For example:
    - Highlighting incomplete sections or missing deliverables.
    - Identifying errors in technical content or formatting.
    - Offering suggestions for better structuring or presentation.
- This feedback will be presented to the administrator to assist in assigning grades. While the final grading remains the administrator's responsibility, the system ensures consistency and reduces manual effort.

3.2.6 Customizable Forms

- The administrator will have the capability to create and release forms tailored to specific needs, such as collecting data from students and TAs. These forms can be used for:
    - Skill assessments to evaluate technical proficiencies.
    - Availability schedules for better planning of team activities.
    - Feedback collection for mid-project reviews or overall course evaluation.
- The forms can include a variety of input types, such as multiple-choice questions, text boxes, or file uploads. Data collected through these forms will be automatically linked to relevant team or TA profiles, making it easy to access and integrate with other platform features.
- This functionality streamlines the data collection process, ensuring the administrator can gather and manage information efficiently.

3.2.7 Discussion forums

- The platform will feature a robust system of discussion forums to facilitate communication:
    - **Global Forum**: Accessible to all users for announcements, general queries, and cross-team discussions.
    - **Team-Specific Forums**: Automatically created after team registration, these forums will provide a private space for team members to collaborate. Assigned TAs will also have access to these forums to provide guidance.
    - **Paired-Team Forums**: Created during the beta testing phase, these forums will enable communication between paired teams, promoting smoother collaboration.
    - **Anonymous Doubt Clearance Forum**: Students will be able to ask doubts anonymously.

- These forums will foster teamwork, allow TAs to provide timely input, and help paired teams coordinate effectively during beta testing.

3.2.8 Beta Testing Pairing

- The system will assist the administrator during the beta testing phase by generating automated team pairing recommendations. These recommendations will consider project domains, technical compatibility, and any specific prompts provided by the administrator to ensure logical pairings.
- Once the administrator finalizes the pairings, the system will automatically:
    - **Create Paired-Team Forums**: Each paired team will have a dedicated discussion forum to facilitate communication and collaboration.
    - **Share Project Files**: All necessary project files and documents will be shared between the paired teams, enabling them to review and test each other's work without additional overhead.
- This pairing and collaboration process ensures effective beta testing, fostering cross-team feedback and smoother coordination.

3.3 Use Case Model

3.3.1 Use Case #1 (Team Creation and Registration)

Author - Achyuth Warrier, Aarav Oswal

Purpose - To create teams for the project

Requirements Traceability -

- The system shall allow students to create a team and share a code for joining the team. Alternatively, they can send invites via the platform directly.
- The system shall allow the instructor to choose the maximum and minimum number of students in each team.
- The system shall allow the teams to claim relevant technical tags to specify the tech stack required for their project.
- The system shall allow the instructor to lock the teams, allowing no further changes.

Priority - High

Preconditions -

- All students enrolled in the course must be registered on the platform.

Post conditions -

- Creation of team roles which allow instructors to create forms for team submissions.
- Creation of forums specific to teams.

Actors - Professor, Students, System

Exceptions -

- For students not assigned teams or teams with less than minimum members, the prof must manually assign these members, with the option for randomized allocation.
- For duplicate team names, the user must be notified and be prompted for a unique name.

3.3.2 Use Case #2 (Project Management)

Author - Harshpreet Kaur, Ravi Arora

Purpose - To allow the students and professor to keep track of the various project deliverables and to allow the students to maintain their projects on the system.

Requirements Traceability -

- The system shall allow the professor to add deadlines for the project deliverables that shall be visible on the integrated Calendar of both the students and the professor.
- The system shall provide for a git-like version control system that will be used by the students to work on their project.
- The system shall allow the students as well as the professor to see various progress indicators for the progress of respective teams, such as contribution heatmaps, tech stack, etc.

Priority - High

Preconditions -

- The team creation and registration process must be completed. Users must be able to login to the website.

Post conditions -

- The professor shall be able to add deadlines for the project documents.
- The students and the professor shall be able to see the deadlines on their calendar.
- The students shall be able to see a version control system tab on the system where they can work on their project.
- The professor shall be able to access the projects of all the teams.
- The students as well as the professor shall be able to see the heatmaps of the team as well as the students.

Actors - Professor, Students, System

Exceptions - None

3.3.3 Use Case #3 (TA allocation based on project relevant skills)

Author - Nikhil Pothuganti, Ashik Stenny

Purpose - The primary objective of this use case is to automate the assignment of teaching assistants (TAs) to project teams based on their skill sets and team requirements. This ensures optimal support for each team and efficient utilization of TA expertise.

Requirements Traceability -

- **TA Skill Requirements -** Identification of the specific skills and expertise that each of the TA's have in various languages to support the project groups effectively.
- **Project Requirements -** A clear understanding of the projects technical requirements and the specific skills of each of the members of the team
- **Implementation and monitoring -** Trace the implementation of the allocation and monitor the progress of each of the teams, keeping track of how well the TA's are able to work with the team members.

Priority - High

Preconditions -

- Teams have registered their projects and provided information about the skills they require.
- A form for TAs has been floated and completed with details such as their skill sets, preferred project types, and availability.
- The system has access to an updated database of TAs and team requirements.

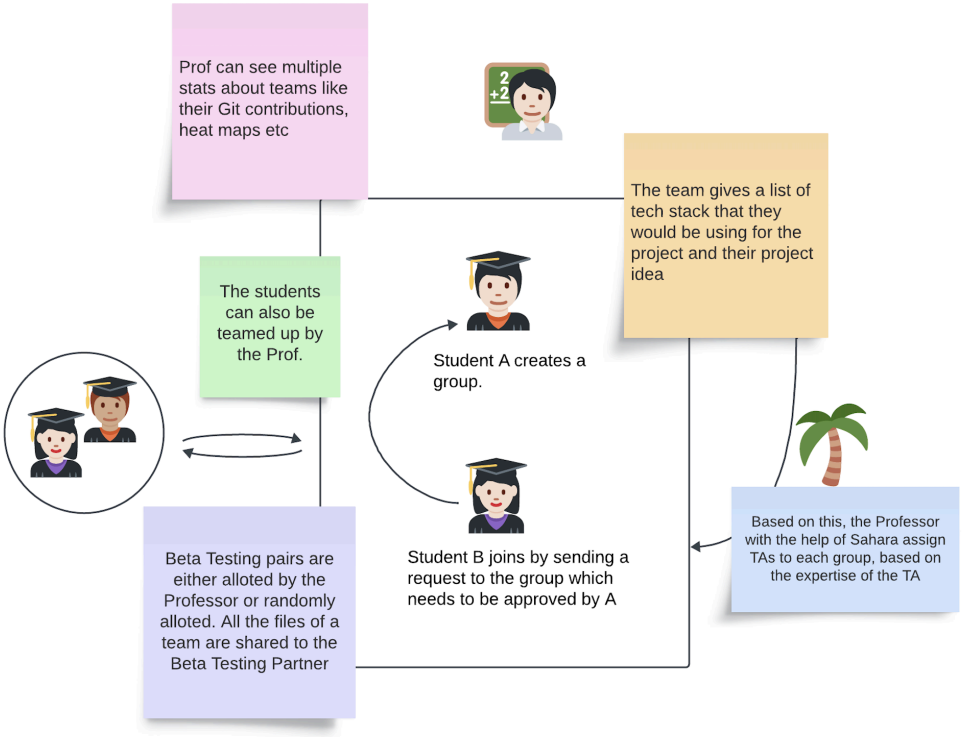
Post conditions -

- The system pairs each team with a TA whose skill set aligns with the team's requirements.

Actors - Students, TAs, Professor and System.

Exceptions -

- TA availability and scheduling constraints - A TA with the ideal skill set might not be available due to scheduling conflicts which might not be taken into account in our software.
- Incomplete TA profiles: Request completion of missing details and defer allocation.



Overview of Use Cases #1, #2 and #3

3.3.4 Use Case #4(Grading Assignments)

Author - Spandan Pati, R.Charan

Purpose - The primary objective of this use case is to automate the grading process for documents submitted by students. The AI tool evaluates submissions based on predefined criteria and generates a concise analysis.

Requirements Traceability -

- The system shall allow the AI tool to provide feedback on each submitted document based on the assessment criteria provided by the instructor.

Priority - High

Preconditions -

- Students have submitted their documents via the platform.
- The AI tool has been trained on the relevant data and grading criteria.

Post conditions -

- The AI tool provides a detailed qualitative analysis of each submission.
- The analysis is stored in the database and accessible to instructors.
- Instructors use the analysis to inform their grading decisions.

Actors - Instructors, Students, AI Grading Tool, Database.

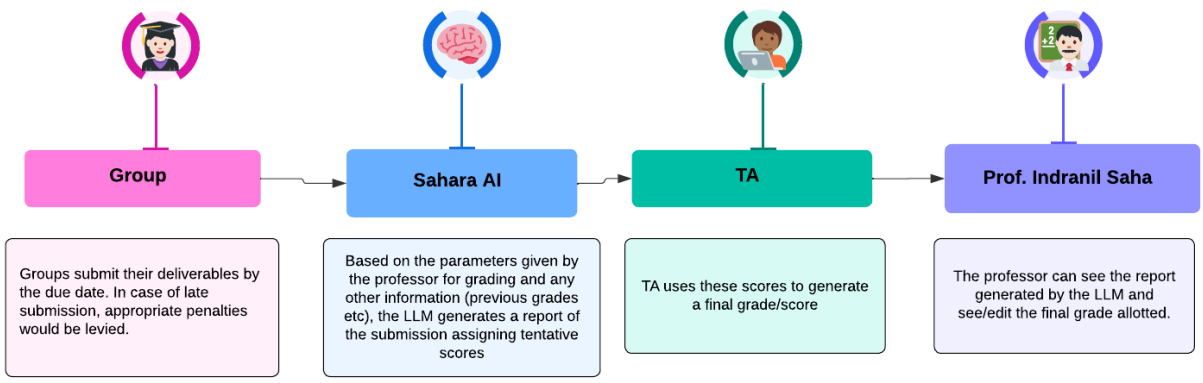
Exceptions -

- Invalid document format: Notify the student and request resubmission of document.
- AI grading tool failure: Revert to manual grading and notify the TA about the issue.

Includes-

- Upload Analysis Criteria.
- Submit Assignment.

Overview of Use Case # 4



3.3.5 Use Case #5 (Dedicated Discussion Forums)

Author - Vedhanth Balasubramaniam, Divi Pothukuchi

Purpose - To manage all discussions related to the course/project in a single place, such as a global forum, forums with teams and respective TA's, etc.

Requirements Traceability -

- The system shall allow for the creation of global discussion forums accessible to all students and instructors.
- The system shall automatically create team-specific discussion forums after team registration. (Non-functional)
- The system shall enable TAs to participate in forums associated with their assigned teams.
- The system shall support notifications for new messages and updates in forums.

Priority - Medium

Preconditions -

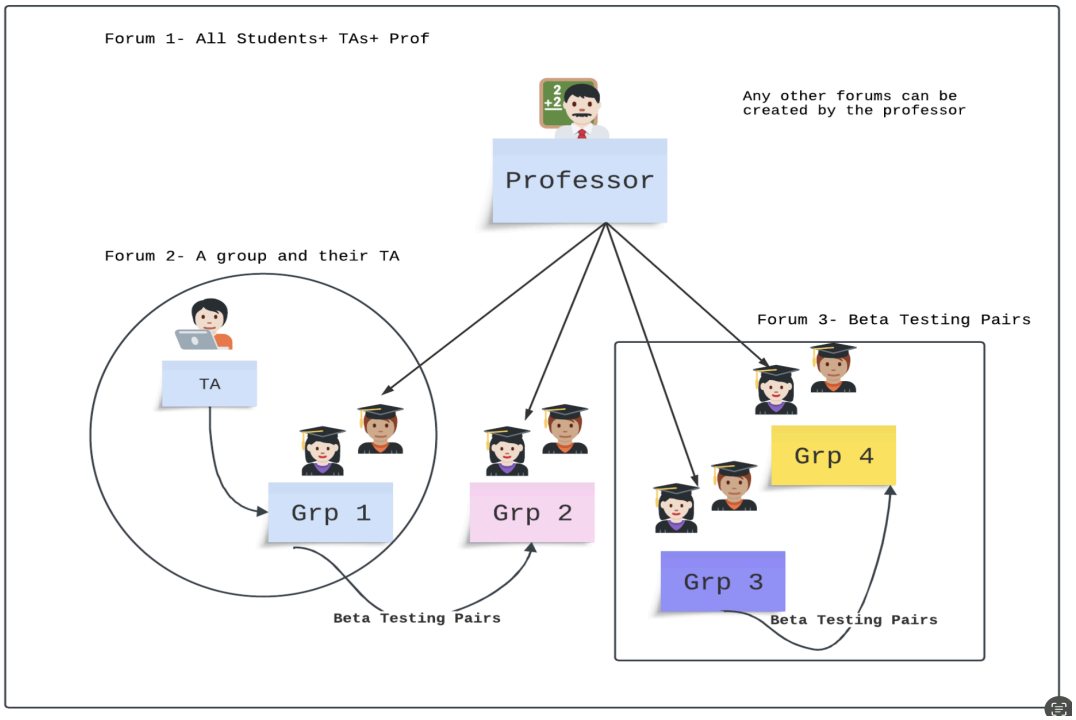
- Teams should be registered and TA's should be assigned, for creation of automated discussion forums.

Post conditions -

- Dedicated forums are created and accessible based on user roles (e.g., team forums for team members, TA forums for TAs and their teams).
- Users receive notifications for relevant updates in their associated forums.

Actors - Professor, Students, System

Exceptions - None



Overview of Use Case # 5

4 Other Non-functional Requirements

4.1 Performance Requirements

- **Concurrency** – The site should be able to able to handle a load of at least 250 users (including the course instructor, teaching assistants and students)
- **Responsiveness** – Average response time should be below 1 second, with no considerable latency for the users.
- **Compatibility** – It should perform optimally with proper layout and performance for various devices and web browsers.
- **Availability** - System should have an availability of 99.9% over a 24 hour period especially during critical times such as document/assignment submissions.
- **Throughput** - The site should be able to handle 500 API requests per second.

4.2 Safety and Security Requirements

- **Login Mechanism:**
  The system must implement robust and secure login mechanisms to prevent unauthorized access or hacking attempts. Multi-factor authentication (MFA) will be included as an additional layer of security for critical operations.
- **Data Backup:**
  Periodic, automated, and secure backups of the entire database must be taken to ensure data integrity and minimize the risk of data loss. Backups will be stored in encrypted form at multiple secure locations.
- **Secure Transmission of Information:**
  All data transmitted between the client and server must use secure protocols such as HTTPS with TLS 1.3 or higher to ensure confidentiality and integrity.
- **Password Security:**
  System administrators will not have access to the plaintext passwords; they will only be able to verify the correctness of the passwords during authentication.
  The system will enforce a password policy to prevent the use of weak passwords. Weak passwords are defined as follows:

  - Length of less than 8 characters.
  - Containing only lowercase or uppercase letters.
  - Using commonly used phrases (e.g., "password123") or patterns.

  Users will receive prompts for password updates if their credentials are deemed weak or compromised.

- **Browser Compatibility:**
  The system will only support browsers that adhere to modern security standards, including:
  - Apple Safari 7+
  - Google Chrome 44+
  - Microsoft Edge 90+
  - Mozilla Firefox 40+

- **Forgot Password Feature:**
  A secure "Forgot Password" feature will be implemented with the following mechanisms:
  Users must answer pre-configured security questions or verify their identity via an alternative method (e.g., email or OTP).
  Reset links or temporary codes will expire after a short duration to minimize misuse.

- **Encryption of Sensitive Data:**
  All sensitive user data, such as personally identifiable information (PII), must be encrypted in storage and transit to prevent data breaches and unauthorized access.

- **Access Control:**
  Role-based access control (RBAC) will be implemented to ensure that users have access only to the features and data necessary for their role. Administrative functions will require additional authentication.

4.3 Software Quality Attributes

4.3.1 Maintainability

- The architecture, design, implementation, and documentation of the website must minimize maintenance overhead to support regular academic updates, such as syllabus changes, grading, and announcements.

- Fixing a security defect, including updating documentation and testing, must not take more than **two working days**, achieved by designing the website to be modular and easy to debug.

- The average work time required to add a minor feature, such as adding a new type of assignment or a notification system, including documentation and unit testing, should be less than **one person-week**.

4.3.2 Availability

- In case of a server crash, the course website must be restored to its latest state within **two hours** to avoid disruption to grading, submissions, or student access to materials.

- The system availability should handle peak usage during exam preparation periods when students are actively accessing materials, submitting assignments, or checking grades.

- The system must be robust enough to support high activity during deadlines for assignments or projects and at the end of the semester when grades are released.

4.3.3 Reliability

- The **Mean Time to Failure (MTTF)** shall be more than **one week** under normal academic usage conditions.
- The system must undergo extensive feature testing (e.g., handling assignment submissions, grading systems), load testing for peak activity periods, and regression testing before each academic term or major release.
- The website should consistently provide accurate results, such as displaying correct grades, assignment statuses, and notifications to all users.

4.3.4 Portability

- The website will be designed to be portable, responsive, and compatible with any modern web browser. This ensures that students, TAs, and the professor can seamlessly access it on desktops, laptops, tablets, and smartphones without encountering compatibility issues.
- User data and project-related content can be exported in standard formats (e.g., CSV, JSON) to ensure compatibility with other tools or systems.
- APIs will be provided for easy integration with external software like grading systems, learning management systems (LMS), or team collaboration tools

By focusing on these aspects, the system ensures that users can access and utilize the platform efficiently, regardless of their chosen devices or operating environments.

Appendix A – Data Dictionary

Data Dictionary

- **Common Class Variables**

Name	Type	Description	States	Operations
name	string	Name of student	Any text	Obtained from email
email	string	Email id of institute	Any valid email id	Specified during registration
pwd	string	Password for login	Any text	Specified during registration

- **Common Class Methods**

Name	Description
register()	To register a new user of type student
login()	To verify login details
logout()	To logout
forgot_pwd()	To change password if needed
forum_comm()	To comment in a forum thread

- **Professor Class Variables**

Name	Type	Description	States	Operations/Requirement
course_ID	string	Course ID	Any valid course ID	Specified during registration
team_size	int	Size for student team for the course	Any valid integer	Specified during registration and can be changed later

- **Professor & TA Class Methods**

Name	Description
add_course()	Add a new course
reg_course	Register for new course
announce()	To make announcements
create_quiz()	To create a quiz
view_quiz()	To view quiz statistics
create_form()	To float a form (for TAs and students both)
view_form()	To view forms responses in an organized manner
set_deadline()	To create and change deadlines
view_team()	To check team progress and details
assign_role()	To assign roles to different users on the platform

***Rows in bold indicate professor specific methods**

- **Student Class**

Name	Type	Description	States	Operations
stu_name	string	Name of student	Any text	Obtained from email
stu_email	string	Student email id of institute	Any valid email id	Specified during registration
stu_pwd	string	Password for login	Any text	Specified during registration
roll_no	int	Roll Number of student	Any valid roll number	Specified during registration
team	int	Team number to which they belong	Any valid integer	Updated after team form is submitted

- **Student Class Methods**

Name	**Description**
form_submit()	To submit forms floated by professor/TAs
add_course()	To register for a new course
quiz_submit()	To answer and submit quiz
view_quiz_stu()	To view results of quizzes
team_submit()	To submit team related forms
team_join()	To join team once forms are floatedMissing notifications for forum updates: Enable a fallback notification mechanism, such as email alerts.

Data Relationships and Constraints

- **Relationships**:
    - Professors manage courses, which are linked to teams.
    - Teams consist of students who belong to a specific course.
    - TAs are assigned to teams based on skills and availability.
- **Constraints**:
    - Email addresses must be unique across all user roles.
    - Passwords must adhere to security policies (minimum 8 characters, alphanumeric).
    - Team sizes must not exceed the limit specified by the professor.
    - Deadlines cannot overlap for the same course deliverables.

Appendix B - Group Log

- A whatsapp group was created for communication between members.
- Offline meetings were hosted whenever necessary to take inputs from members on the project progress.

Date and Time duration of the Meet	**Meet Type**	**Ongoing discussions in the meet**
10/01/2025 9:00 am-12:00 am	Offline Meet	Meet to discuss the initial project idea and elaborate on real world relevance.
14/01/2025 9:00 am-12:00 am	Offline Meet	Discussion over what to include in software functionalities. Initial ideas on the requirement document were shared. Tasks associated with the initial draft (SRS doc) were equally distributed to each team member.
17/01/2025 9:00pm-2:00am	Offline Meet	A meeting was scheduled to discuss and review the initial draft. It was decided to arrange a meeting with the client to assess how well the draft aligns with their requirements
21/01/2025 11:00am-12:00pm	Offline Meet	Discussion over updates after meeting with the client. The changes required on the initial draft was decided and associated tasks were assigned to each member.
22/01/2025 9:00pm-1:00am	Offline Meet	A meeting was held to make changes to the initial draft, ensuring that each team member contributed equally to its completion.
23/01/2025 9:00pm-1:00am	Offline Meet	A meeting was held to work on UI for the draft.
24/01/2025 8:00pm-11:00pm	Offline Meet	A meeting was held to prepare the final draft, ensuring that each team member contributed equally to its completion.