|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Software Requirement Specifications**  **Fast Central Hub**  **Version: [1.6]**   |  |  | | --- | --- | | Project Code | F22-1206 | | Supervisor | Dr. Abdul Aziz | | Co Supervisor |  | |  |  | | Project Team | Amman Soomro (19K-1048)  Naba Zehra (19K-1118)  Mashood Nadeem (19k-1090) | | Submission Date | 25-November-2022 | |

[Instructions]

* No section of template should be deleted. You can write ‘Not applicable’ if a section is not applicable to your project. But all sections must exist in the final document.
* All comments/examples mentioned in square brackets ([]) are in the template for explanation purposes and must be replaced / removed in final document.
* This’ Instruction’ section should also be removed in final document.
* MS-Word Reviewing feature must be used to get the document reviewed by supervisors or co-supervisors.

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Name of Person** | **Date** | **Description of change** |
| 1.0 | Mashood Nadeem | 10 Nov 22 | Started Working on the SRS |
| 1.1 | Naba Jafri | 13 Nov 22 | Added Few Details |
| 1.2 | Naba Jafri | 15 Nov 22 | Added Functional Hierarchy |
| 1.3 | Amman Soomro | 20 Nov 22 | Added Scope, Functional and Non Functional Requirements |
| 1.4 | Mashood Nadeem | 22 Nov 22 | Added Use Cases |
| 1.5 | Mashood Nadeem | 24 Nov 22 | Reviewed Document |
| 1.6 | Amman Soomro, Mashood Nadeem | 25 Nov 22 | Final Review and Proof Reading |

Distribution List

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | |
| Dr. Abdul Aziz | | Supervisor |
|  | | Co- Supervisor |
|  | |  |

Document Sign-Off

|  |  |  |
| --- | --- | --- |
| **Version** | **Sign-off Authority** | **Sign-off Date** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[1.](#_30j0zll) Introduction 7

[1.1.](#_3znysh7) Purpose of Document 7

[1.2.](#_2et92p0) Intended Audience 7  
1.3 Abbreviations ………………………………………………………………………………………...7

[1.4.](#_tyjcwt) Document Convention 7

[2.](#_3dy6vkm) Overall System Description 8

[2.1.](#_1t3h5sf) Project Background 8

[2.2.](#_4d34og8) Project Scope 8

[2.3.](#_2s8eyo1) Not In Scope 8

[2.4.](#_17dp8vu) Project Objectives 8

[2.5.](#_3rdcrjn) Stakeholders 8

[2.6.](#_26in1rg) Operating Environment 8

[2.7.](#_35nkun2) System Constraints 8

[2.8.](#_1ksv4uv) Assumptions & Dependencies 8

[3.](#_2jxsxqh) External Interface Requirements 9

[3.1.](#_z337ya) Hardware Interfaces 9

[3.2.](#_3j2qqm3) Software Interfaces 9

[3.3.](#_1y810tw) Communications Interfaces 9

[4.](#_4i7ojhp) Functional Requirements 10

[4.1.](#_2xcytpi) Functional Hierarchy 10

[4.2.](#_1ci93xb) Use Cases 10

[4.2.1.](#_3whwml4) [Title of use case] 10

[5.](#_2bn6wsx) Non-functional Requirements 11

[5.1.](#_qsh70q) Performance Requirements 11

[5.2.](#_3as4poj) Safety Requirements 11

[5.3.](#_1pxezwc) Security Requirements 11

[5.4.](#_49x2ik5) User Documentation 11

[6.](#_147n2zr) References 12

[7.](#_3o7alnk) Appendices 13

1. **Introduction**

* 1. **Purpose of Document**

The purpose of this document is to specify all the requirements of our project in a detailed format, including its goals, scope and services. In a detailed format all the requirements of the stakeholders would be mentioned, such as system functionalities and its implementations.

* 1. **Intended Audience**

The targeted audience for this document will be our supervisor, members of the jury, university faculty and students.

**1.3 Abbreviations**

Not Applicable.

* 1. **Document Convention**

The font used for the headings and the body is Arial. The headings have a font size of 16 while the subheadings have a font size of 12. Following the paragraphs, it has a font size of 10. This format is maintained throughout the document.

1. **Overall System Description**
   1. **Project Background**

Students currently studying in university highly desire to be facilitated in terms of being up to date with whatever is going in the university or having availability of study materials. For course feedback and tentative study plan students require Flex. For announcement, study material and submission purpose students require SLATE or Google Classroom. For being updated regarding different societies and their events students require Facebook or Instragram. This raises an issue of high distribution of data over different applications. Apart from this, Final Year students require some extra study material, information and older FYPs’ for their Final Year Project idea, working and documentation, which turns out to be unavailable online at any student portal. Students new to courses require some introduction of that particular course before opting for it and some feedback of the teachers teaching that particular course.

* 1. **Project Scope**

Our project, Fast Central Hub will be facilitating students in different aspects over a single platform. All university students can log in using their credentials provided them by the university. On a single platform, study material of different courses and its introduction for students new to the course will be available on a single click. Students can keep themselves up to date regarding different societies, their events, any holiday, extra day at university or any other announcement by the university just by visiting Fast Central Hub. Rather than having data distributed over different platforms we bring everything onto Fast Central Hub, at a single platform. Final year students can easily access older final year projects for idea and reference purpose. A student if faces problem in understanding his teacher’s study material, he can easily go to Fast Central Hub and get access to study material of that same course of different teachers. Students not familiar with their respective teacher and get an overview and rating of that teacher following which course is being taught.

* 1. **Not In Scope**

You can not assign any work or sell your services over it. Any mode of communication between students over this platform is not supported. As there is no idea of submissions, attendance, course registration, professors marking anything as an example, or course results, our system is not entirely ties to any learning management system.

* 1. **Project Objectives**

The objective of our project is to facilitate the students of the university in important aspects and phases of their graduation. Students face lack of enough study material on any online platform associated with their university. With evolving time students have started to prefer soft copies over hard copies and want all material to be a available just a click away. Our objective is to make all the study material, teacher’s information/rating, university updates and other miscellaneous information to be available for student of every batch, just a click away. We aim 100% of the university students to get what it takes to graduate with a good score.

* 1. **Stakeholders**

Following are the stakeholder of this application:

* Users (Students, Teacher, Admin).
* Project Team Members
* The Project Supervisor
* Members of the Jury
* University
  1. **Operating Environment**

The project requires the users to only have a smartphone, computer or laptop without any constraint on the operating system as the application will have the ability to run on Android, Windows or iOS devices and any web browser. However, users must have access to the internet as the application will not run without a stable WIFI connection.

* 1. **System Constraints**

Since our application is being developed with input and feedback from our university and its audience it is strictly not allowed to copy the whole website or any part of it. The team is liable to take any legal action(s) if such a circumstance arises. Any changes made to the functional and non-functional requirements of the application have to be discussed with the university’s admin before the team can move ahead. This is a cross-platform website therefore users of any operating system or device can get access to it. As stated earlier WIFI is a very important requirement in order to access the applications services. Furthermore, the language used in this application is only English i.e. the interface and all its features will use English to communicate information to the users. This website is only liable to be used for university management purpose, not for any other industrial use.

* 1. **Assumptions & Dependencies**

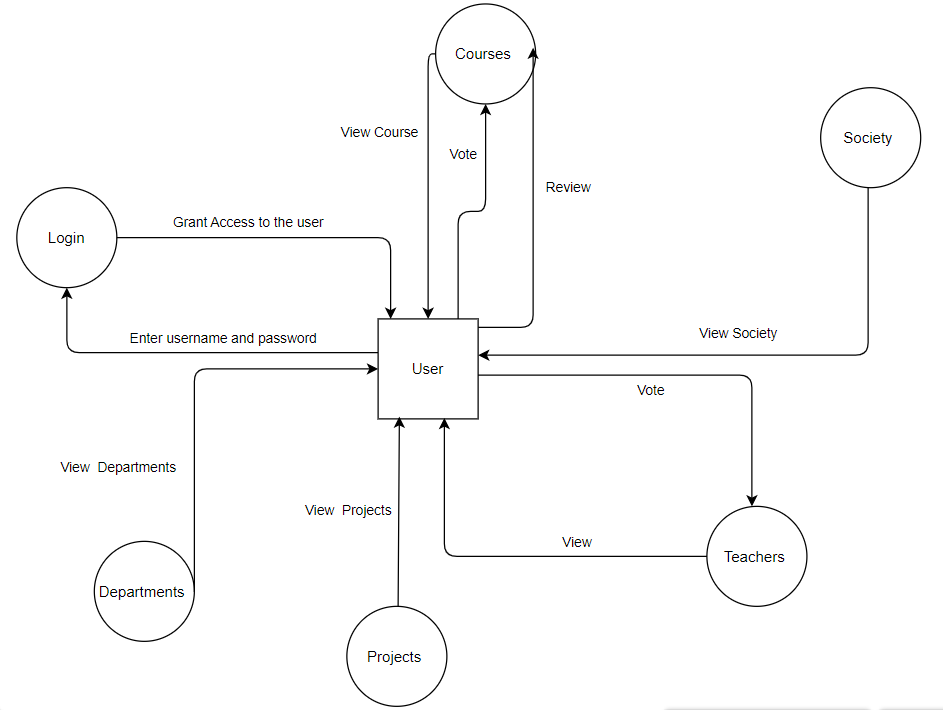
We assume that the users off our website:

* Have a stable internet connection
* Understand English
* Users for this website are; Students, Teachers or Admin

System Dependencies:

* Our project cannot be completely executed and tested until we’re provided with the necessary data required from the university.
* Our system and its functionalities must be obey the rules and laws of the university.
* If university desires, we must remove that particular functionality from our system.

1. **External Interface Requirements**



* 1. **Hardware Interfaces**

For this web application all you need is a computer or any other device with an internet connection to access it

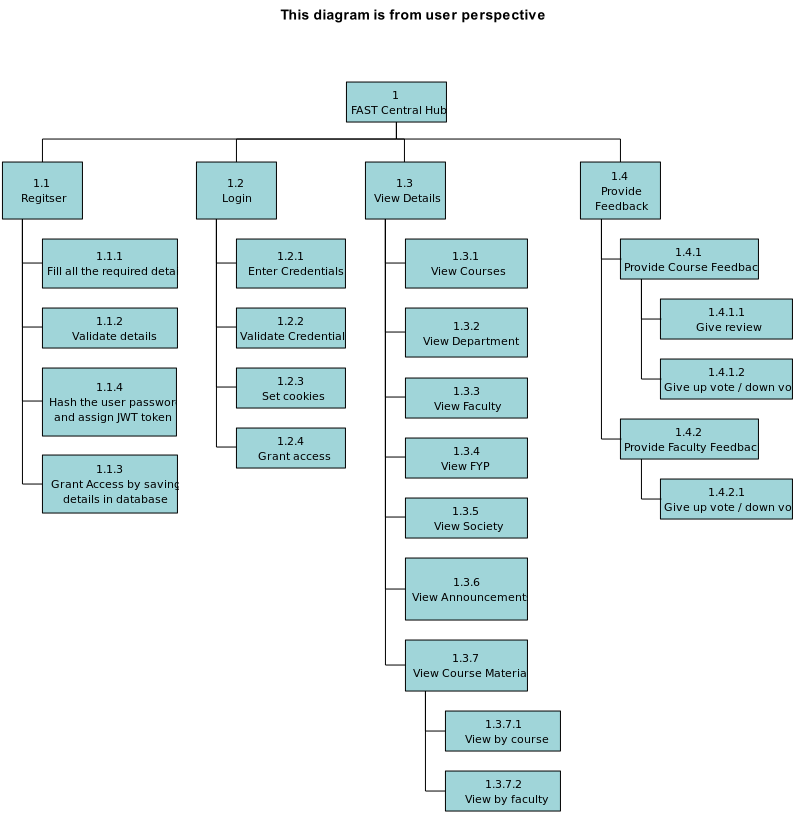
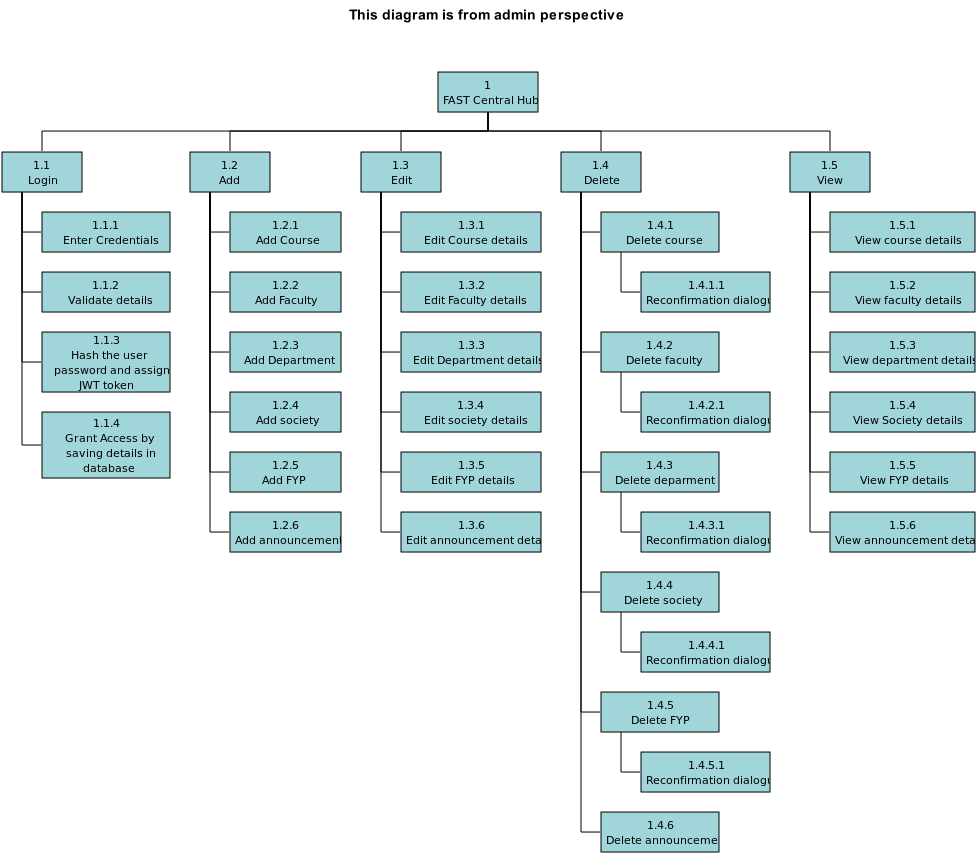
* 1. **Software Interfaces**

The system will use MongoDB ATLAS for storing the system's database. As it is a web application it can run on any operating system. The system will be implemented using Mern Stack and we will use JavaScript as our main language specifically ES6. The files will be stored in Firebase storage. The IP network protocol which will be used is HTTP(S) in conjunction with the JSON format for communication. Furthermore, this application will use the npm package to further enhance the performance.

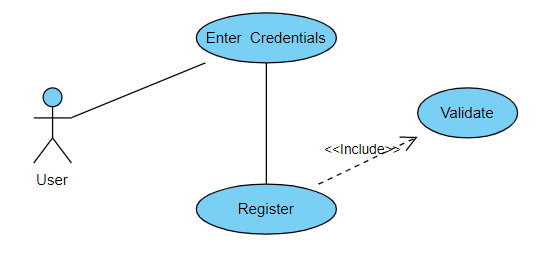
* 1. **Communications Interfaces**

The application will communicate with the server using Express APIs. Since security is also a priority in the system, bcrypt will be used to hash the password and which will then be saved to the system's database.

1. **Functional Requirements**
   1. **Functional Hierarchy**

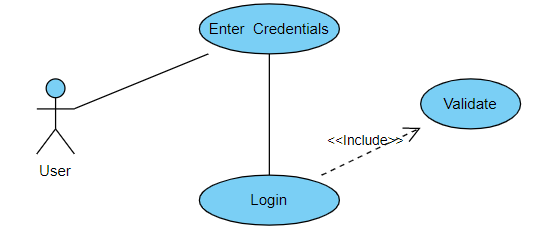
****

* 1. **Use Cases**
     1. **Registration**



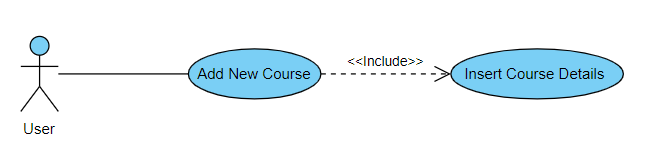
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.1 : Registration** | | | | |
| **Use case Id:** | | 4.2.1 | | |
| **Actors:**  Users-(Teacher/Admin and Students) | | | | |
| **Feature:** Sign-Up | | | | |
| **Pre-condition:** | | - | | |
| **Scenarios:** Before using the application, users like, students and teachers have to register on it using their university allotted email address or roll number. As soon as they register, the account gets verified and ready to be logged in for next time. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | First of all the user must click on the sign-up to enter their details and register | | | Redirects the user to the registration form. |
| **2.** | Enter email address or roll number | | | Email or roll number’s format will be checked. |
| **3.** | Enter password | | | Password format will be checked |
| **4.** | Enter Name | | | Checks if the field is not empty. |
| **5.** | Click the button to register | | | Data will be verified and the user will be redirected to a specific interface upon successful creation. |
| **Alternate Scenarios:** | | | | |
| **1a:** Password hash will be created which is then saved in the database.    **2a:** Unique IDs will be generated by the system and will be assigned to each user. | | | | |
| **Post Conditions:** Successful creation of an account which can be used to login to the system at a later date. | | | | |
| **Step#** | **Description** | | | |
|  | The user will have a system generated ID which will act as the primary key in the database. | | | |
|  | A record corresponding to the user's credentials will be entered into the database. | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | - | |

* + 1. **Log-In**

****

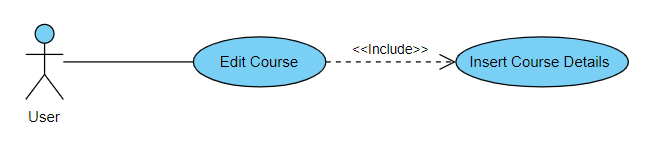
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.2 : Log-In** | | | | |
| **Use case Id:** | | 4.2.2 | | |
| **Actors:**  Users-(Teacher/Admin and Students) | | | | |
| **Feature:** Log-In | | | | |
| **Pre-condition:** | | The user needs to be registered before to Log-In. | | |
| **Scenarios:** Considering the user has registered already, he/she logs in to use the web application and perform further actions. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | First of all the user must click on the log-in to enter their details and procees | | | Redirects the user to the log-in page. |
| **2.** | Enter email address or roll number | | | Check if the email is correct and matches. |
| **3.** | Enter password | | | Check if the password matches. |
| **4.** | Click on the button to log-in | | | Authenticate and the user and proceed with the login process. |
| **Alternate Scenarios:** | | | | |
| **1a:** Email or roll number will be first checked if it exists in the database.  **2a:** Password entered and the password stored in the database of the respective roll number will be checked if they match. | | | | |
| **Post Conditions:** Successful authentication will lead the user to log into the system and use its services. | | | | |
| **Step#** | **Description** | | | |
|  | Access to use the system will be granted to the user according to its type. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.1 | |

* + 1. **Add Course**

****

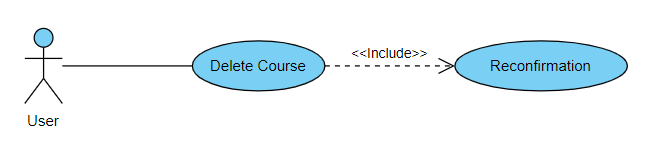
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.3 : Add Course** | | | | |
| **Use case Id:** | | 4.2.3 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Add Course | | | | |
| **Pre-condition:** | | The user type needs to be a admin and they need to be logged in. | | |
| **Scenarios:** Admin logs into the system and adds course in the course list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the add new course button, | | | Redirects the user to the add new course form. |
| **2.** | Enter Course Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the course and uploads it onto the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When adding new course, teachers names would be specified too for proper listing.  **2a:** | | | | |
| **Post Conditions:** After successful validation the new course is added to the courses list. | | | | |
| **Step#** | **Description** | | | |
|  | The course is available for the student users to view it and its details. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 | |

* + 1. **Edit Course**

****

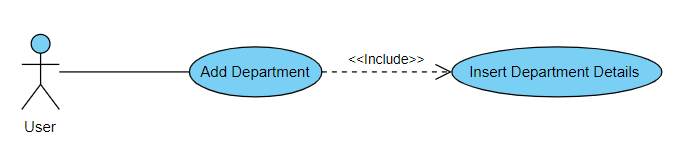
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.4 : Edit Course** | | | | |
| **Use case Id:** | | 4.2.4 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Edit Course | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in.  Course must be present in the list. | | |
| **Scenarios:** Admin logs into the system and edit course details in the course list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the edit course button, | | | Redirects the user to the edit course form. |
| **2.** | Enter Course new Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the course details, uploads and replaces it with the previous details in the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When editing course, teachers names would be specified too for proper listing.  **2a:** Proper details need to be entered for valid display of details. | | | | |
| **Post Conditions:** After successful validation the new course details are added to the courses list. | | | | |
| **Step#** | **Description** | | | |
|  | The course is available for the student users to view it and its details. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.3 | |

* + 1. **Delete Course**

****

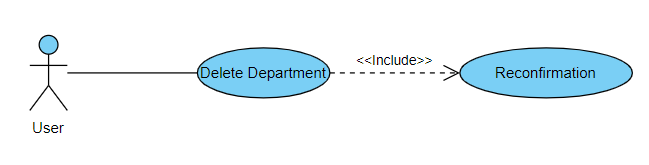
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.5 : Delete Course** | | | | |
| **Use case Id:** | | 4.2.5 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Delete Course | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in.  Course must be present in the list. | | |
| **Scenarios:** Admin logs into the system and deletes course from the course list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the delete course button | | | Redirects the user to the delete course option |
| **2.** | Delete specific course | | | Takes all the action as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the course presence and deletes it from the interface list and database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When deleting course, the course must be present or has been added before.  **2a:** - | | | | |
| **Post Conditions:** After successful validation the specific course is deleted. | | | | |
| **Step#** | **Description** | | | |
|  | The course is removed from the list and new list is available for the student users. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.3 | |

* + 1. **Add Department**

****

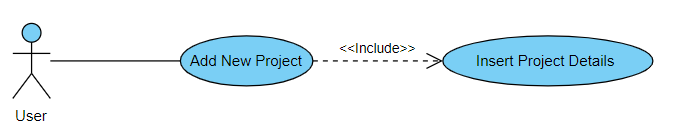
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.6 : Add Department** | | | | |
| **Use case Id:** | | 4.2.6 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Add Department | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in. | | |
| **Scenarios:** Admin logs into the system and adds department in the course list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the add new department button, | | | Redirects the user to the add new department form. |
| **2.** | Enter Department Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the department and uploads it onto the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When adding new department, it needs to be validated to view proper listing for the students.  **2a: -** | | | | |
| **Post Conditions:** After successful validation the new department is added to the courses list. | | | | |
| **Step#** | **Description** | | | |
|  | The department and its details are available for the student users to view it. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 | |

* + 1. **Delete Department**

****

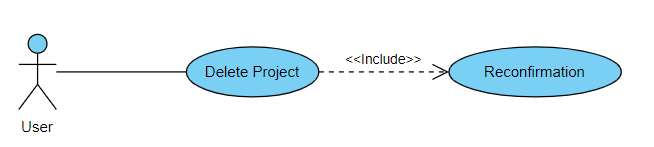
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.7 : Delete Department** | | | | |
| **Use case Id:** | | 4.2.7 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Delete Department | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in.  Department must be present in the list. | | |
| **Scenarios:** Admin logs into the system and deletes department from the department list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the delete department button | | | Redirects the user to the delete department option |
| **2.** | Delete specific department | | | Takes all the action as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the department presence and deletes it from the interface list and database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When deleting department, the department must be present or has been added before.  **2a:** - | | | | |
| **Post Conditions:** After successful validation the specific department is deleted. | | | | |
| **Step#** | **Description** | | | |
|  | The department is removed from the list and new list is available for the student users. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.6 | |

* + 1. **Add Project**

****

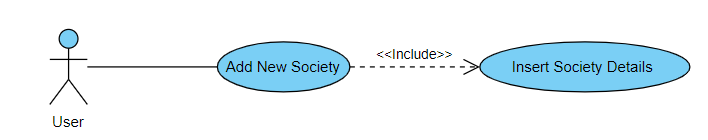
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.8 : Add Project** | | | | |
| **Use case Id:** | | 4.2.8 | | |
| **Actors:**  Users - Admin | | | | |
| **Feature:** Add Project | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in. | | |
| **Scenarios:** Admin logs into the system and adds project in the project list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the add new project button, | | | Redirects the user to the add new project form. |
| **2.** | Enter project Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the projects and uploads it onto the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When adding new projects, it needs to be validated to view proper listing for the students.  **2a:** The project needs to be associated with a respective course for better understanding of the user | | | | |
| **Post Conditions:** After successful validation the new project is added to the project list. | | | | |
| **Step#** | **Description** | | | |
|  | The project and its details are available for the student users to view it. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 | |

* + 1. **Delete Project**

****

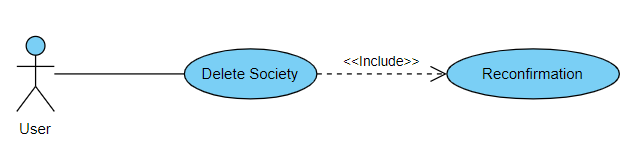
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.9 : Delete Project** | | | | |
| **Use case Id:** | | 4.2.9 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Delete Project | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in. Project must be present in the list. | | |
| **Scenarios:** Admin logs into the system and deletes project from the project list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the delete project button | | | Redirects the user to the delete project option |
| **2.** | Delete specific project | | | Takes all the action as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the project presence and deletes it from the interface list and database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When deleting project, the project must be present or has been added before.  **2a:** - | | | | |
| **Post Conditions:** After successful validation the specific project is deleted. | | | | |
| **Step#** | **Description** | | | |
|  | The project is removed from the list and new list is available for the student users. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.8 | |

* + 1. **Add Society**

****

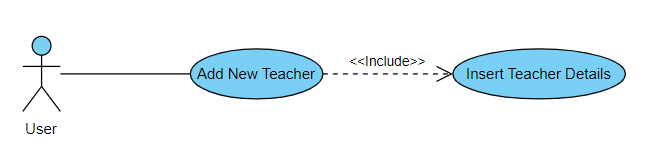
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.10 : Add Society** | | | | |
| **Use case Id:** | | 4.2.10 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Add Society | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in. | | |
| **Scenarios:** Admin logs into the system and adds society in the society list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the add new society button | | | Redirects the user to the add new society form. |
| **2.** | Enter society Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the society and uploads it onto the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When adding new society, it needs to be validated to view proper listing for the students.  **2a:** The project needs to be associated with a respective department, if needed, for better understanding of the user. | | | | |
| **Post Conditions:** After successful validation the new society is added to the society list. | | | | |
| **Step#** | **Description** | | | |
|  | The society and its details are available for the student users to view it. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 | |

* + 1. **Delete Society**

****

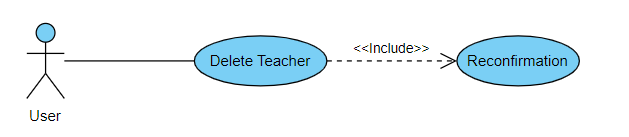
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.11 : Delete Society** | | | | |
| **Use case Id:** | | 4.2.11 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Delete Society | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in. Society must be present in the list. | | |
| **Scenarios:** Admin logs into the system and deletes project from the Society list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the delete society button | | | Redirects the user to the delete society option |
| **2.** | Delete specific society | | | Takes all the action as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the society presence and deletes it from the interface list and database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When deleting society, the project must be present or has been added before.  **2a:** - | | | | |
| **Post Conditions:** After successful validation the specific society is deleted. | | | | |
| **Step#** | **Description** | | | |
|  | The society is removed from the list and new list is available for the student users. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.10 | |

* + 1. **Add Teacher**

****

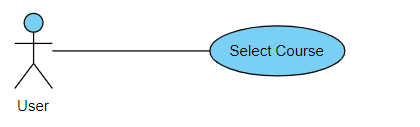
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.12 : Add Teacher** | | | | |
| **Use case Id:** | | 4.2.12 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Add Teacher | | | | |
| **Pre-condition:** | | The user type needs to be a admin and they need to be logged in. | | |
| **Scenarios:** Admin logs into the system and adds teacher in the teacher list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the add new teacher button | | | Redirects the user to the add new teacher form. |
| **2.** | Enter teacher Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the teacher and uploads it onto the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When adding new teacher, it needs to be validated to view proper listing for the students.  **2a:** The teacher needs to be associated with respective department and courses if needed, for better understanding of the user. | | | | |
| **Post Conditions:** After successful validation the new teacher is added to the teacher list. | | | | |
| **Step#** | **Description** | | | |
|  | The teacher and its details are available for the student users to view it. | | | |
|  | Courses taught by that specific teacher is associated with the teacher in the system. | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 | |

* + 1. **Delete Teacher**

****

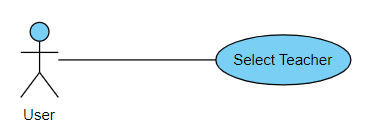
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.13 : Delete Teacher** | | | | |
| **Use case Id:** | | 4.2.13 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Delete Teacher | | | | |
| **Pre-condition:** | | The user type needs to be a admin and they need to be logged in. Teacher must be present in the list. | | |
| **Scenarios:** Admin logs into the system and deletes teacher from the Teacher list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the delete teacher button | | | Redirects the user to the delete teacher option |
| **2.** | Delete specific teacher | | | Takes all the action as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the teacher presence and deletes it from the interface list and database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When deleting teacher, the project must be present or has been added before.  **2a:** - | | | | |
| **Post Conditions:** After successful validation the specific society is deleted. | | | | |
| **Step#** | **Description** | | | |
|  | The teacher is removed from the list and new list is available for the student users. | | | |
|  | The course associated to that teacher is just deleted from that teachers information list but not from the courses list. | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.12 | |

* + 1. **View Courses**

****

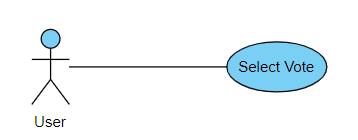
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.14 : View Courses** | | | | |
| **Use case Id:** | | 4.2.14 | | |
| **Actors:**  Users – Student | | | | |
| **Feature:** View Courses | | | | |
| **Pre-condition:** | | The user type needs to be a student and they need to be logged in. | | |
| **Scenarios:** Student logs into the system and views all the courses. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the Courses section | | | Redirects the user to the courses list. |
| **2.** | Then selects the specific teacher of that course from the list | | | Then redirects the user to the teachers list of that specific course. |
| **3.** | Then selects the specific material to be viewed. | | | Downloads the material selected. |
| **Alternate Scenarios:** | | | | |
| **1a:** -  **2a: -** | | | | |
| **Post Conditions:** The required and selected course material is downloaded. | | | | |
| **Step#** | **Description** | | | |
|  | After following the steps of action, student can get the material they require. | | | |
|  | Once material found, it can be downloaded. | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 | |

* + 1. **View Teachers**

****

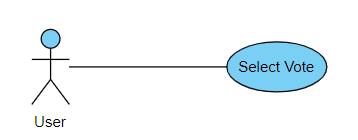
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.15 : View Teacher** | | | | |
| **Use case Id:** | | 4.2.15 | | |
| **Actors:**  Users – Teacher | | | | |
| **Feature:** View Teacher | | | | |
| **Pre-condition:** | | The user type needs to be a student and they need to be logged in. | | |
| **Scenarios:** Student logs into the system and views all the teachers. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the Teachers section | | | Redirects the user to the teachers list. |
| **2.** | Then selects the specific course of that teacher from the list | | | Then redirects the user to the courses list of that specific teacher. |
| **3.** | Then selects the specific material to be viewed. | | | Downloads the material selected. |
| **Alternate Scenarios:** | | | | |
| **1a:** -  **2a: -** | | | | |
| **Post Conditions:** The required and selected course material is downloaded. | | | | |
| **Step#** | **Description** | | | |
|  | After following the steps of action, student can get the material they require. | | | |
|  | Once material found, it can be downloaded. | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 | |

* + 1. **Course Vote**

****

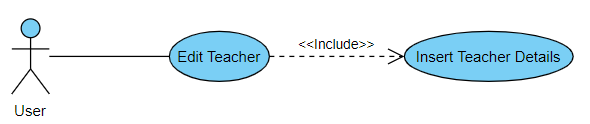
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.16 : Course Vote** | | | | |
| **Use case Id:** | | 4.2.16 | | |
| **Actors:**  Users – Students | | | | |
| **Feature:** Course Vote | | | | |
| **Pre-condition:** | | The user type needs to be a student and they need to be logged in. | | |
| **Scenarios:** Student logs into the system and views the course and vote them. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the Course section | | | Redirects the user to the course list. |
| **2.** | Then votes the course | | | Takes the vote as a response and saves it. |
| **Alternate Scenarios:** | | | | |
| **1a:** - Student gets 2 options to vote. Upvote and DownVote.  **2a:** - Student makes their choice. | | | | |
| **Post Conditions:** The selected vote is saved and added into the counter of that specific vote, i.e upvote/downvote. | | | | |
| **Step#** | **Description** | | | |
|  | Vote response goes to the database of that course. | | | |
|  | Each course has a vote counter which increments and decrements according to students’ action. | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 | |

* + 1. **Teacher Vote**

****

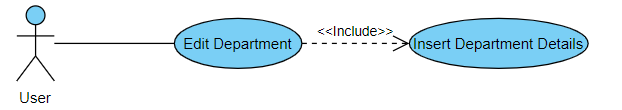
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.16 : Teacher Vote** | | | | |
| **Use case Id:** | | 4.2.16 | | |
| **Actors:**  Users – Students | | | | |
| **Feature:** Teacher Vote | | | | |
| **Pre-condition:** | | The user type needs to be a student and they need to be logged in. | | |
| **Scenarios:** Student logs into the system and views the teacher and vote them. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the Teacher section | | | Redirects the user to the teacher list. |
| **2.** | Then votes the teacher | | | Takes the vote as a response and saves it. |
| **Alternate Scenarios:** | | | | |
| **1a:** - Student gets 2 options to vote. Upvote and DownVote.  **2a:** - Student makes their choice. | | | | |
| **Post Conditions:** The selected vote is saved and added into the counter of that specific vote, i.e upvote/downvote. | | | | |
| **Step#** | **Description** | | | |
|  | Vote response goes to the database of that teacher. | | | |
|  | Each teacher has a vote counter which increments and decrements according to students’ action. | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 | |

* + 1. **Edit Teacher**

****

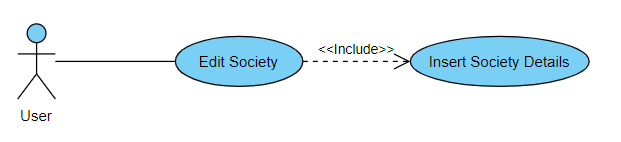
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.18 : Edit Teacher** | | | | |
| **Use case Id:** | | 4.2.18 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Edit Teacher | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in.  Teacher must be present in the list. | | |
| **Scenarios:** Admin logs into the system and edit teacher details in the teacher list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the edit teacher button, | | | Redirects the user to the edit teacher form. |
| **2.** | Enter Teacher new Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the course details, uploads and replaces it with the previous details in the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When editing teacher, courses names would be specified too for proper listing.  **2a:** Proper details need to be entered for valid display of details. | | | | |
| **Post Conditions:** After successful validation the new teacher details are added to the teacher list. | | | | |
| **Step#** | **Description** | | | |
|  | The teacher is available for the student users to view it and its details. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.12 | |

* + 1. **Edit Department**

****

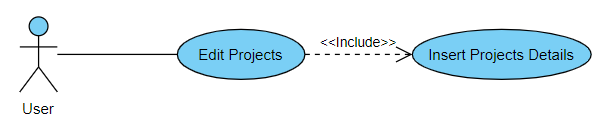
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.19 : Edit Department** | | | | |
| **Use case Id:** | | 4.2.19 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Edit Department | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in.  Department must be present in the list. | | |
| **Scenarios:** Admin logs into the system and edit department details in the department list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the edit department button, | | | Redirects the user to the edit department form. |
| **2.** | Enter Department new Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the department details, uploads and replaces it with the previous details in the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When editing department, courses names would be specified too for proper listing.  **2a:** Proper details need to be entered for valid display of details. | | | | |
| **Post Conditions:** After successful validation the new department details are added to the department list. | | | | |
| **Step#** | **Description** | | | |
|  | The department is available for the student users to view it and its details. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.6 | |

* + 1. **Edit Society**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.20 : Edit Society** | | | | |
| **Use case Id:** | | 4.2.20 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Edit Society | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in.  Society must be present in the list. | | |
| **Scenarios:** Admin logs into the system and edit society details in the society list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the edit society button, | | | Redirects the user to the edit society form. |
| **2.** | Enter Society new Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the society details, uploads and replaces it with the previous details in the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** Proper details need to be entered for valid display of details.  **2a: -** | | | | |
| **Post Conditions:** After successful validation the new society details are added to the society list. | | | | |
| **Step#** | **Description** | | | |
|  | The society is available for the student users to view it and its details. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.10 | |

* + 1. **Edit Project**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.2.21 : Edit Project** | | | | |
| **Use case Id:** | | 4.2.21 | | |
| **Actors:**  Users – Admin | | | | |
| **Feature:** Edit Project | | | | |
| **Pre-condition:** | | The user type needs to be a Admin and they need to be logged in.  Project must be present in the list. | | |
| **Scenarios:** Admin logs into the system and edit project details in the project list. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User clicks on the edit project button, | | | Redirects the user to the edit project form. |
| **2.** | Enter project new Details | | | Takes all the details as a response. |
| **3.** | User clicks on the button to the submit the details | | | Validates the project details, uploads and replaces it with the previous details in the database. |
| **Alternate Scenarios:** | | | | |
| **1a:** When editing project, courses names would be specified too for proper listing.  **2a:** Proper details need to be entered for valid display of details. | | | | |
| **Post Conditions:** After successful validation the new project details are added to the project list. | | | | |
| **Step#** | **Description** | | | |
|  | The project is available for the student users to view it and its details. | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | 4.2.2 and 4.2.9 | |

1. **Non-functional Requirements**
   1. **Performance Requirements**

The application will have interactive interfaces that will be user-friendly and pages will be loaded instantly. Since a large application size has adverse effects on performance we intend to compress images and other graphic elements to reduce the image size. There will be no delays even under strong traffic.

* 1. **Safety Requirements**

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to secondary storage device (typically a USB) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

* 1. **Security Requirements**

Proper input validation will be done to ensure that only legitimate users access the system's services. During registration Server-Side Validation will be done to ensure that all the necessary information was filled in. The password and email address compile with our expected format standards which are: passwords should contain at least 8 characters and must be alphanumeric, the email prefix as well as the email domain must be valid. Furthermore, the password hash will be stored in the database and upon login a hash of the entered password will be calculated and compared with the one stored in the database.

* 1. **User Documentation**

The user documentation that will be provided in the end will be:

1. User Manual
2. Online help

1. **References**

[This section should provide a complete list of all documents referenced at specific point in time. Each document should be identified by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. (This section is like the bibliography in a published book).]

1. **Appendices**

Not applicable here.