1. Introduction

1.1 Purpose

Purpose of this document is to build a digital system to maintain the records of the students OF Tezpur University clearing competitive exams like NET/SLET/GATE etc. This document contains the details of the system and its components.

1.2 Document Conventions

The following conventions are used throughout the document

- **Competitive Exams** means state/ national/ international level examinations (e.g., NET/SLET/GATE/GMAT/CAT/GRE/TOEFL/Civil Services/State government examinations)
- **System** means A Database maintenance system about students qualifying in Competitive exams.
- **Admin** refers to administrators of the database having special permissions to edit it.

1.3 Intended Audience and Reading Suggestions

This project is intended to produce a working system for recording the database of students of Tezpur University clearing various competitive exams. Also this project is a part of the curriculum of the author. Hence the document is restricted to be used only inside the campus. This document will be useful for department admin's as well as student developers in the campus to come up with such a system. This document should be read in the sequence as shown in the table of contents.

1.4 Project Scope

The project aims to produce a system that manages the data of students clearing various entrances each year. The system is based on a relational database with CRUD operations on student data. An online portal will be set up to make the data visible to the public. Also the system will be scalable to include more and more departments. The data will be maintained by the department officials. A comfortable user experience will be provided to the audience as well as admin to see/edit the data.

1.5 References

➔ Node.js
➔ Mongodb
➔ Cloud Hosting Platforms

2. Overall Description

2.1 Product Perspective:

The system database stores the following information:

- Student Description:

It contains the information about students such as Name, Department, Roll No etc.

- Entrance Description:

It contains the details of the entrances such as Name, Frequency of the entrance each year, type of the entrance(National/Regional) etc.

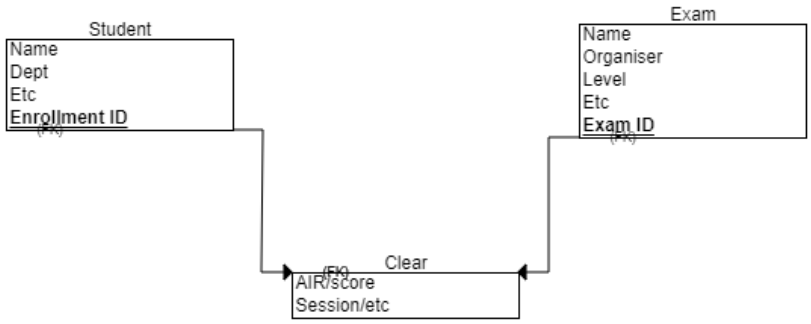
- Students clearing entrances Description:

It contains the Information of students clearing specific entrances as entry to the database.

2.2 Product Features:

The major features of the system are:

- Database: The system contains the database of students clearing various entrances. The following ER Model represents the database.



- Web Portal: A web portal allowing users to see the database content will be constructed. Admin will be able to modify the database through the portal.
- Authenticity: The web portal will process authentication.

2.3 User Classes and Characteristics

The system will support two types of users.

- Non-Admin
- Admin

Any public user visiting the portal will be classified as a non-Admin user. They will be able to see the database contents presented in a suitable user interface to the user. No login will be required. These users should be able perform the following functionality:

- Search the database
- View the database
- View statistics

There will be a login button on the public version of the webpage. But the login credentials will be available to only the admins. A user after logging in will be treated as an admin user. He will be able to perform the following functions:

- All functions of Non-Admin user
- Create entry to database
- Update the database
- Delete entries from database

2.4 Operating Environment

The operating environment of the system is shown below:

- Operating system for server program: Linux(Preferably)/windows.
- Deployment of the server program: Any cloud provider like heroku/google cloud/AWS.
- Database: MongDB
- Platform: NodeJS

2.5 Design and Implementation Constraints

- The loading time of the portal shouldn't be more than a few sec.
- The ram and database should fit in the free tier of the cloud hosting platform that is around 1 or 2 gb ram and 500 mb database.
- No sensitive information should be available to the non-admin users.
- The HTML/CSS of the page shouldn't reveal unwanted information.
- The UI should be suitable for even non-technical people.

2.6 Assumptions and Dependencies

The system is assumed to be not limited within campus and accessible to anyone outside the campus. Also it is assumed that the initial prototype can be hosted on non-university servers. It is assumed that the admins before uploading the data will take consent from the students to upload his/her information. As a result no ecxtra consent checking step included,

3. System Features

The system features are shown below.

3.1 View record details (Available For Both Non-Admin/Admin users)

3.1.1 The users should be able to view the records of the students clearing various entrances. They should be able to view students' records by selecting examination name or department name. That is after visiting the portal they can select examinations or departments and get a list of students. (Priority level: 5/5)

3.1.2 Stimulus/Response Sequences

- visit the portal
- Select view category: Exam or Department.
- Select a specific exam or department.
- get the records

3.1.3 Functional Requirements

- ❖ REQ-1: Search database records according to the department and display results.
- ❖ REQ-2: Search database records based on Exam and display results.

3.2 Search record details (Available For Both Non-Admin/Admin users)

3.2.1 The users should be able to search records after viewing. That is after viewing records on the portal they can search the results by selecting different categories as shown below. (Priority level: 5/5)

3.2.2 Stimulus/Response Sequences

- View records
- Select search category
- Enter search parameters
- search and get the results

3.2.3 Functional Requirements

- ❖ REQ-3: Search shown records based on Name.
- ❖ REQ-4: Search shown records based on Exam.
- ❖ REQ-5: Search shown records based on Time-Period.

3.3 View sorted record (Available For Both Non-Admin/Admin users)

3.3.1 The users should be able to sort shown records depending upon the sort criteria. That is after viewing, searching they can sort the records (Priority level: 5/5).

3.3.2 Stimulus/Response Sequences

- select sorting criteria
- sort records
- view sorted records

3.3.3 Functional Requirements

- ❖ REQ-6: Sort shown records based on Alphabet of Student names.
- ❖ REQ-7: Sort shown records based on Date.

3.4 Login (Admin users)

3.4.1 The Admin users should be able to login to the system to get the admin privileges of editing the database. (Priority level: 5/5)

3.4.2 Stimulus/Response Sequences

- press the login button on the landing page.
- enter credentials to login prompt.
- get logged into the system as admin.

3.4.3 Functional Requirements

- ❖ REQ-8: Authentication through Login for admin users.

3.5 CRUD operations on the database (Admin users)

3.5.1 Along with all the non-user privileges, the admins should have edit privileges to the database. Each record when viewed in the admin mode should have an edit button at the right most part. By clicking the edit button the admin user should be able to do CRUD operation on the given record. (Priority level: 5/5)

3.5.2 Stimulus/Response Sequences

- login to the system as admin and view records.
- press on the edit button at the end of any record.
- edit record.

3.5.3 Functional Requirements

- ❖ REQ-9: CRUD operation on the records by admin users using an edit button at the end of each record.

3.6 Show statistics or trends using charts (Available for both Admin/Non admin users)

3.6.1 The records shown on the portal should have graphical representation whenever it seems appropriate for the user. Then it would be easier to see the trends or patterns on the data. (Priority level: 4/5)

3.6.2 Stimulus/Response Sequences

- view records on the portal.
- view records on the chart.

3.6.3 Functional Requirements

- ❖ REQ-10: Graphical representation of the records shown on the pages of the portal whenever possible.

4. External Interface Requirements

4.1 User Interfaces

The user interface should be designed neatly as per the convenience of the user. The webpages should be reactive i.e. it should scale automatically for different size screens. Twitter Bootstrap CSS library should be used. For the charts that should accompany the records, suitable javascript libraries must be used. The front end should be kept on HTML and CSS as much as possible. Javascript should be used only when necessary. Material design should be preferred.

4.2 Hardware Interfaces

The system should be executable on a device running any modern web browser like Edge, Chrome, Safari, Opera etc. The system should prompt users to enable javascript on the browser.

4.3 Software Interfaces

The server program will be hosted on a windows or linux machine depending on the availability of the cloud platform. So it must be a cross platform server program. Node.js is such an option that will be used. Similarly the database is mongodb. The front end can be hosted as static HTML/CSS pages on github or within the server environment.

4.4 Communications Interfaces

A standard modern day browser should be able to operate the system through HTTP/HTTPS connections.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system should perform without much lag. If the database grows too large, then there should be scope to add techniques such as paging to fetch the data to the user without much lag, when loading the portal. The UI/UX must not be too complex and heavy that might slow down the system. The connection between server and database should be fluent and must provide data consistency. System should be tested for concurrent requests and load balancers should be placed if needed. The server code as well as database should be backed up in case of any system failure.

5.2 Safety Requirements

The login functionality must be handled with care. No user should ever gain access to the system who is not an admin. Admins should be informed to use non-trivial passwords. Admin controls should be limited to part of the database if needed. regular backups should be made. Care must be taken to prevent attacks like SQL injection.

5.3 Security Requirements

Admins should be instructed to take student consent before entering their details like current working position, email etc to the database. No personal information should ever be publicly displayed without their consent.

5.4 Software Quality Attributes

AVAILABILITY: The student records should be available as per specified exam/department/date. The database should be updated regularly for this purpose.

CORRECTNESS: Correct data should be entered to the system and displayed.

MAINTAINABILITY: The administrators should be able to maintain the system easily and make suitable changes as per requirement.

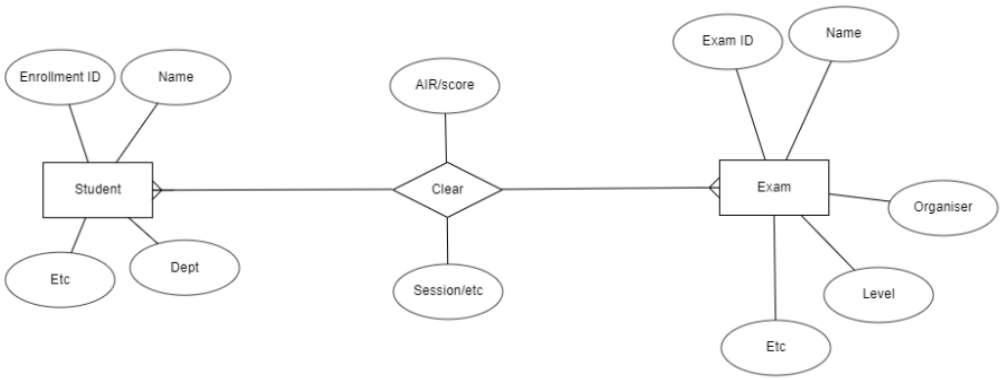
USABILITY: An average user of the internet should be able to use the system without any problem.

AESTHETIC: The UX/UI should be aesthetically pleasing for the users.

6. Other Requirement

The first prototype should be ready by June.

Appendix A: Analysis Models



The ER diagram of the database

Appendix B: Issues List

- Determine cloud hosting provider to be used.
- Determine OS of the server program.