Abstract

This project is aimed at the creation of an easy-to-use web-based database maintenance system containing records of students qualifying various entrance exams from Tezpur University. Public users would be able to view the achievements whereas the Admin would be able to create, read, update and delete those records. This document discusses the design and architecture of the project in detail.

Overview

The Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

1. Introduction

1.1 Purpose and Scope

The purpose of this document is to describe the design and implementation of the student achievement maintenance system. The name of the project is "A Database maintenance system containing records of students qualifying various entrance exams".

Purpose - The purpose of this software is to allow Tezpur University administration to keep records of students, clearing various entrance exams each year, in a digital format and also to display those achievements in a reasonable way to the public. The general public can see the achievements whereas the admin can edit and modify the records.

Objective - To make the system web-accessible, so that it can be viewed by anyone with an active Internet connection.

Benefits and Goals -

i) Will provide easy access to the public to see the student achievement records.

ii) Will help the university to maintain the success of students across various entrance exams. iii) Will inspire students of following matches and also help to set a better public view of the university.

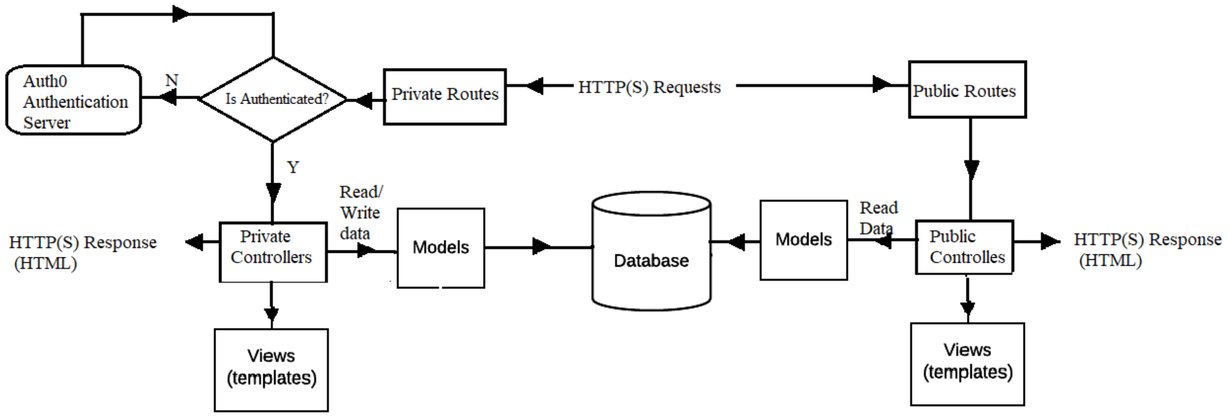
iv) Student's achievements will be recorded and maintained for years.

2. System Architecture

This section describes the system and/or sub-system(s) architecture for the project. The implementation of the software should follow a similar structure, the tech stack that shall be used in each module is listed in the *tech stack requirement* section.

2.1 System Overview

The following diagram shows a high-level overview of how the system is designed.



2.2 External Interface Requirements

This section describes the external system hardware and interfaces which will be used during implementation and development.

Operating System: Windows/Linux.

Internet Connectivity: Required and is a must for the software to function.

2.3 System Software Architecture

This section describes the overall system software and organization. This mostly includes a list of use-cases or software modules (functions, subroutines, or classes). We also describe the software and design constraints here.

Constraints

One of the constraints in this project is that the credentials for authorization of the Admin is predetermined. So that no unauthorized person outside can modify the database. Another constraint is that - to access the database one will need an internet connection.

2.4 Use Cases

This subsection describes the various use cases in detail:

1. Access Homepage

Use Case Name	Access Homepage
Allowed user types	Public, Admin
Brief Description	The web server is waiting for the user to connect.
Trigger	The user visits the website route: “/catalog/”
Precondition	The user is connected to the Internet.
Basic Path	1. The Client checks if the request is authorised. 2. If authorised, the admin version of the homepage is sent as

	response. 3. Else public version of the homepage is sent as response. [Admin version of the homepage contains links to modify student, exam, results records]
Post condition	User is on the Home Page

2. Access Achievements Page

Use Case Name	Access Achievements
Allowed user types	Public, Admin
Brief Description	The web server is waiting for the user to connect.
Trigger	The user visits the website route: "/catalog/achievements/"
Precondition	The user is connected to the Internet.
Basic Path	1. The Client checks if the request is authorised. 2. If authorised, the admin version of the homepage is sent as response. 3. Else public version of the homepage is sent as response. [Admin version of the homepage contains links to modify/create Achievements]
Post condition	User is on the Achievements Page

3. Admin Login

Use Case Name	Admin Login
Allowed user types	Admin
Brief Description	The operation handles authentication of the admin. If the user has logged in a few seconds ago, or the token is not yet expired the user is logged in automatically. Else it prompts login form.
Trigger	The user visits the website route: "/login/"
Precondition	The user should have the necessary credentials.
Basic Path	1. User clicks on the Login button. 2. Server redirects to Auth0 server. 3. Auth0 server handles the login using login form Or Automatic login if the token is not expired. [Auth0 is a third party cloud library that helps in secure login. Auth0 users can be managed through it's cloud platform].
Post condition	If successful, the admin is logged in and can access private routes.

4. Student Record CRU operations

Use Case Name	Create/Read/Update student records.
Allowed user types	Admin
Brief Description	This operation creates/reads/updates student records.
Trigger	Admin routes to the page: ”/catalog/student/create” or “catalog/student/update/:id” or “catalog/students” (read all students) or “catalog/student/:id” (read a specific student)
Precondition	Admin must be authorized and logged in. That is requests must contain tokens.
Basic Path	For create and update: 1. Admin clicks on create/update. 2. The Server returns a form. 3. Admin fills in the form and clicks submit. 4. The Server checks to see if any required field is empty. 5. If any required field is empty Server will send an alert to fill all fields. (It expects alpha numeric inputs). 6. Server checks if duplicate roll no (in case of create). If yes it will send an alert. 7. If no required field is empty and no duplicate roll no the Server will create a new record in student collection in database 8. Admin may press cancel before submitting the form. If Admin selects Cancel, the form is cleared. For Read 1. Admin clicks on “Students" or “a specific student”. 2. Server returns a page with details of the student and links to update the results.
Post condition	A student record is created/read/updated by the admin.

5. Exams CRU operations

Use Case Name	Create/Read/Update exam records.
Allowed user types	Admin
Brief Description	This operation creates/reads/updates exam records.
Trigger	Admin routes to the page: ”/catalog/exam/create” or “catalog/exam/update/:id” or “catalog/exams” (read all exams) or “catalog/exam/:id” (read a specific exam)
Precondition	Admin must be authorized and logged in. That is requests must contain tokens.
Basic Path	For create and update: 1. Admin clicks on create/update. 2. The Server returns a form.

	3. Admin fills in the form and clicks submit. 4. The Server checks to see if any required field is empty. 5. If any required field is empty Server will send an alert to fill all fields. (It expects alpha numeric inputs). 7. If no required field is empty the Server will create a new record in exam collection in database 8. Admin may press cancel before submitting the form. If Admin selects Cancel, the form is cleared. For Read 1. Admin clicks on “exams" or “a specific exam”. 2. Server returns a page with details of the exam and links to update the exam details.
Post condition	An exam record is created/read/updated by admin.

6. Achievements CRUD operations

Use Case Name	Create/Read/Update/Delete exam records.
Allowed user types	Admin
Brief Description	This operation creates/reads/updates/deletes achievements records.
Trigger	Admin routes to the page: ”/catalog/achievement/create” or “catalog/achievement/update/:id” or “catalog/achievement” (read all students) or “catalog/achievement/:id” (read a specific student) or “catalog/achievement/delete/:id”
Precondition	Admin must be authorized and logged in. That is requests must contain tokens.
Basic Path	For create and update: 1. Admin clicks on create/update. 2. The Server returns a form. 3. Admin fills in the form and clicks submit. 4. The Server checks to see if any required field is empty. 5. If any required field is empty Server will send an alert to fill all fields. (It expects alpha numeric inputs). 7. If no required field is empty the Server will create a new record in achievements collection in database 8. Admin may press cancel before submitting the form. If Admin selects Cancel, the form is cleared. For Read 1. Admin clicks on “achievements” or “a specific achievements”. 2. Server returns a page with details of the achievements and links to update/delete the achievement record. For Delete 1. Admin clicks on the “delete” option of an achievement.

	2. Server response with a confirmation page? 3. If the Admin cancels, the server redirects to all achievement pages. 4. If the Admin confirms cancellation the achievement record gets deleted.
Post condition	An achievement record is created/read/updated/deleted by the admin.

7. Search Achievements

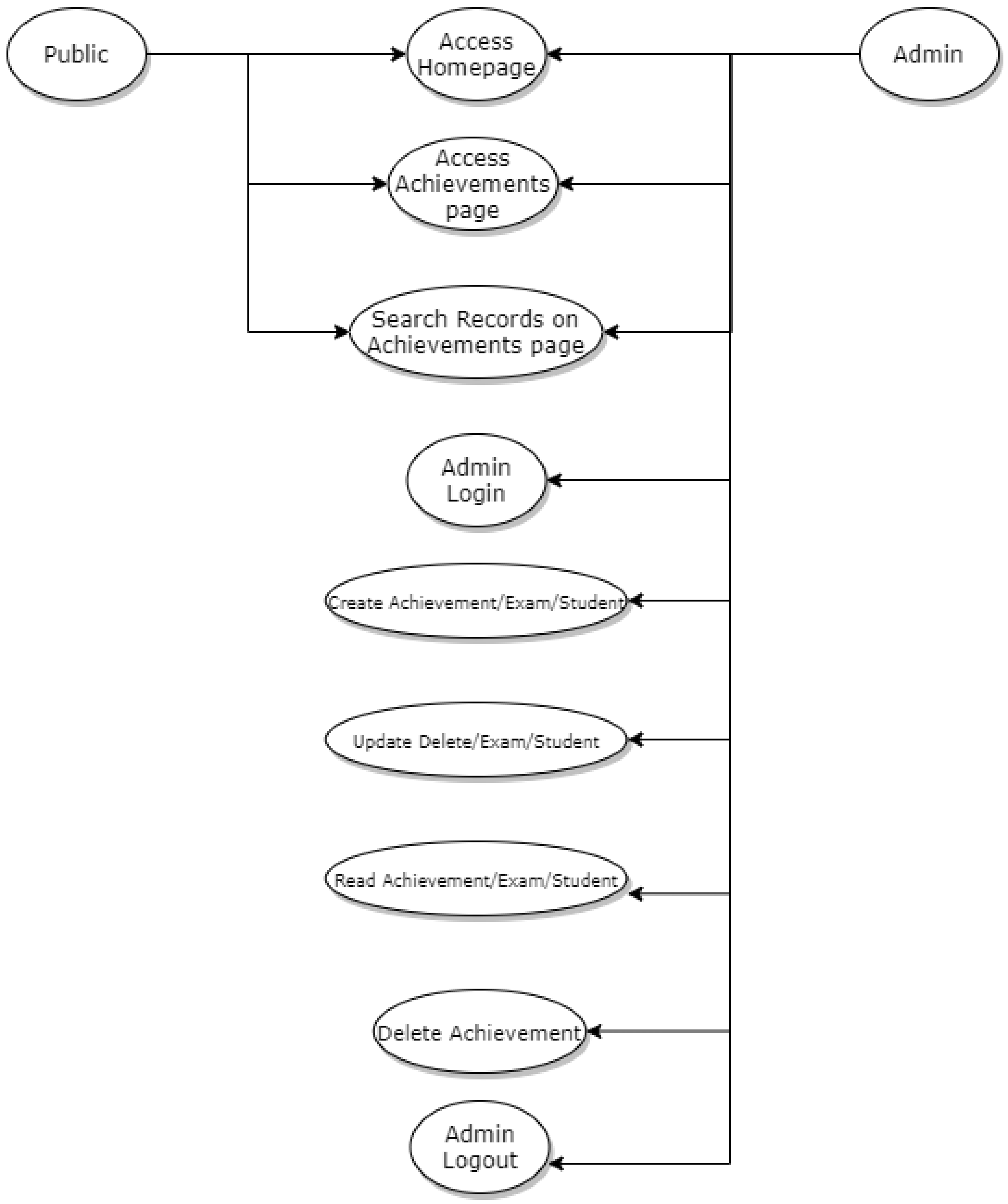
Use Case Name	Search Achievements
Allowed user types	Admin, Public
Brief Description	This operation permits the user to search for information on the achievements page. That is, the user can search using the name, department, exam name or any other keyword of an achievement.
Trigger	The user enters a keyword on the search bar of the “/achievements” page.
Precondition	Users must be connected to the internet.
Basic Path	1. The user visits the achievements page. 2. User types keyword related to an achievement record on the search page. 3. The javascript of the page searches for any match and displays only those records that have matches..
Post condition	Achievement records are filtered and shown as per the given keyword.

8. Admin Logout

Use Case Name	Admin Logout
Allowed user types	Admin
Brief Description	The operation logs the admin out
Trigger	The user visits the website route: “/logout/”
Precondition	The user should have the necessary credentials.
Post condition	If successful, the admin is logged out.

2.5 Overall View

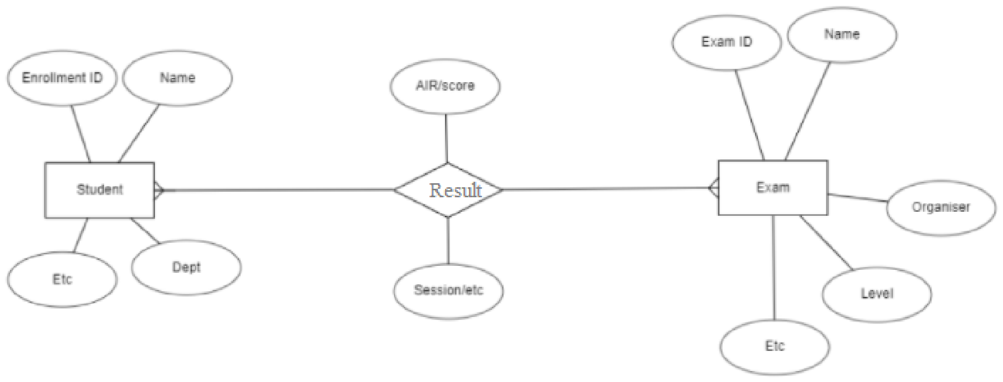
The overall view of the use cases is shown below:


Public
Access Homepage
Admin
Access Achievements page
Search Records on Achievements page
Admin Login
Create Achievement/Exam/Student
Update Delete/Exam/Student
Read Achievement/Exam/Student
Delete Achievement
Admin Logout

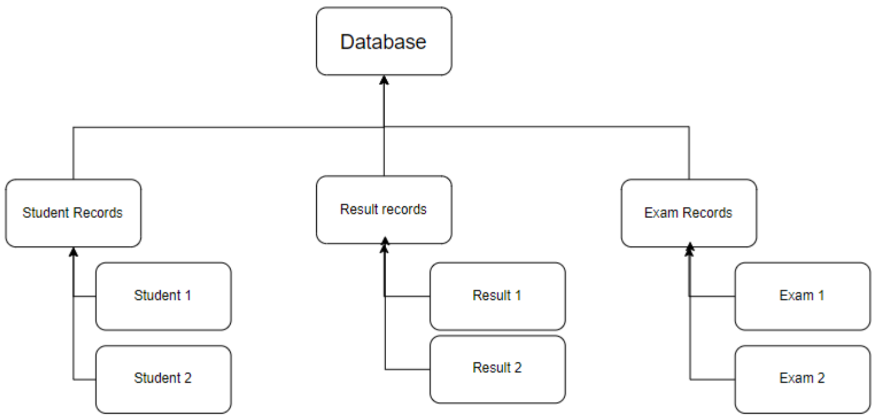
3. Database Organization

For this project, MongoDB which is a NoSQL database is used. MondoDB databases implements records as documents. Each document gets its specific ID. Hence it is easier to maintain relations in the database, We have implemented the database as shown below. Also we have made enrollment no with 'unique' option. That is no two documents with same Enrollment no can be inserted in the students records. The "Achievements" are implemented using "Result" records. "Student" and "Exams" are implemented using database records of the same name.

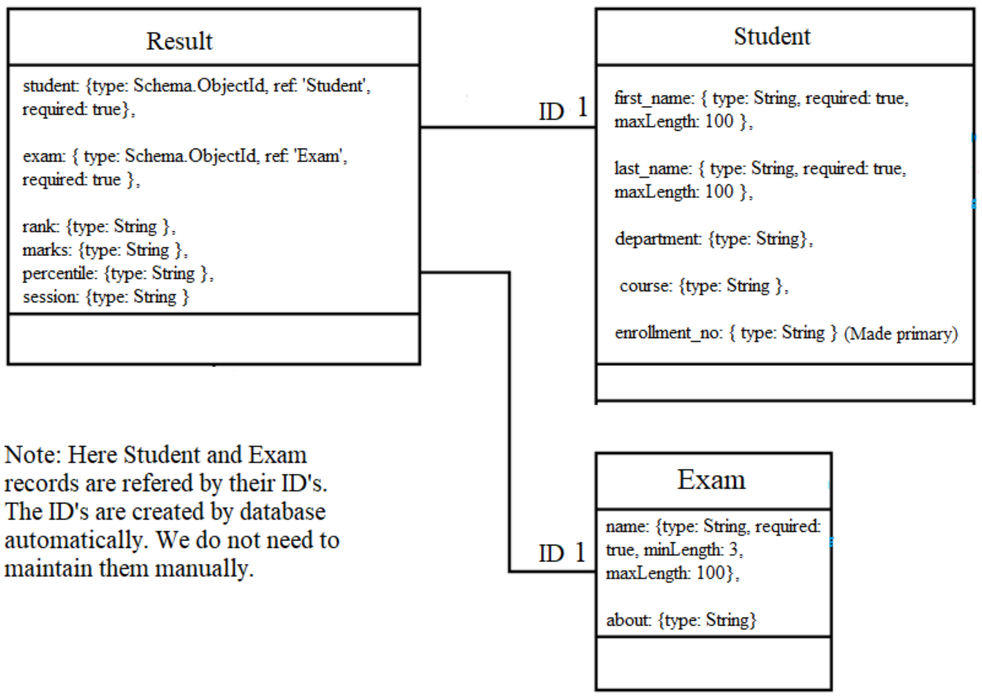
ER Diagram:



Records of the Database:

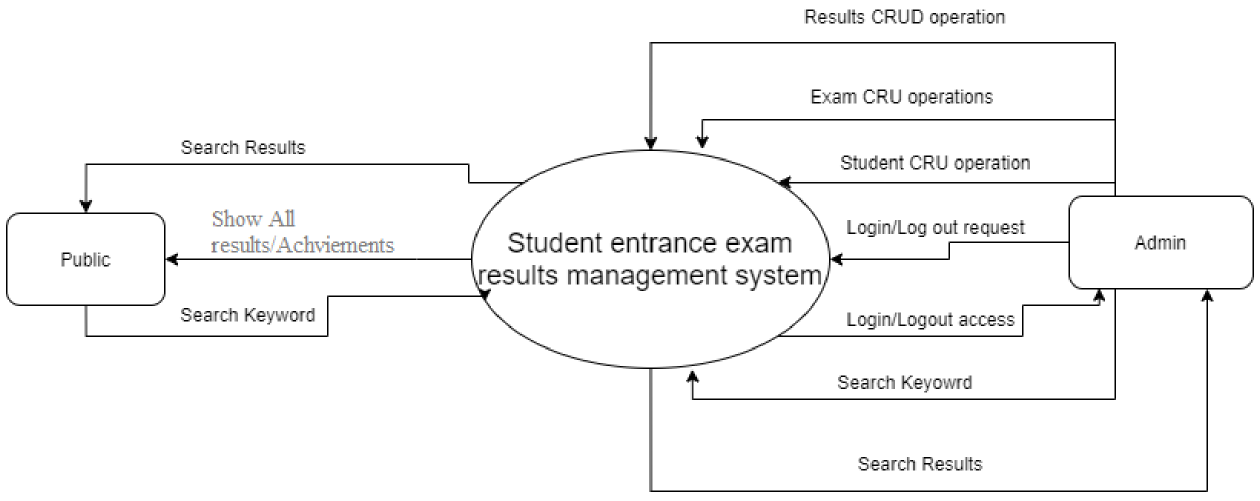


Database Schema:

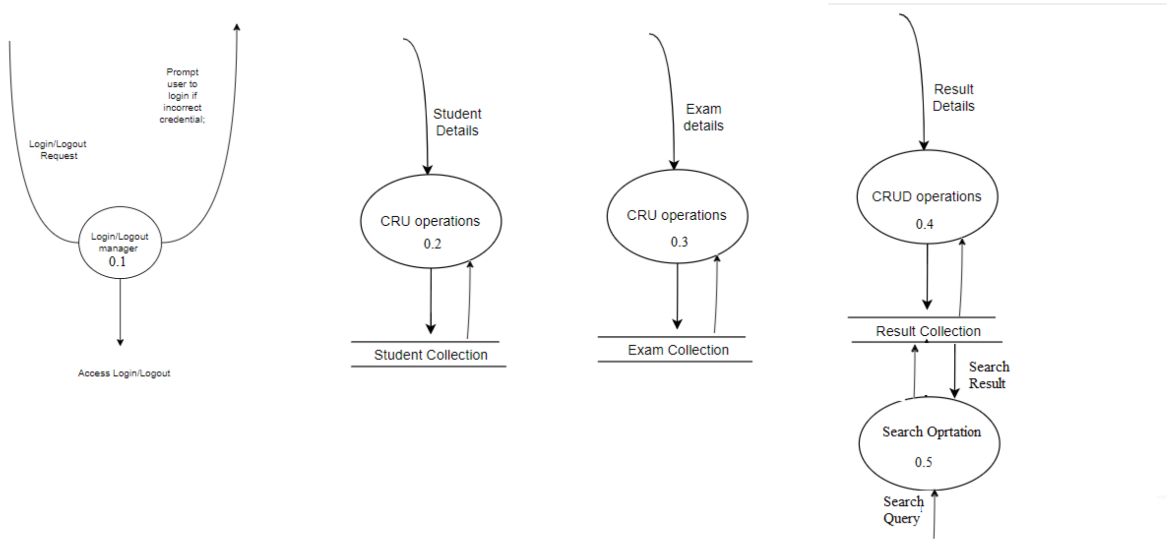

Result
student: {type: Schema.ObjectId, ref: 'Student', required: true},
exam: { type: Schema.ObjectId, ref: 'Exam', required: true },
rank: {type: String },
marks: {type: String },
percentile: {type: String },
session: {type: String }
Student
first_name: { type: String, required: true, maxLength: 100 },
last_name: { type: String, required: true, maxLength: 100 },
department: {type: String},
course: {type: String },
enrollment_no: { type: String } (Made primary)
Exam
name: {type: String, required: true, minLength: 3, maxLength: 100},
about: {type: String}
ID 1
ID 1
Note: Here Student and Exam records are refered by their ID's. The ID's are created by database automatically. We do not need to maintain them manually.

4. Data Flow Diagrams

4.1 Level 0 DFD

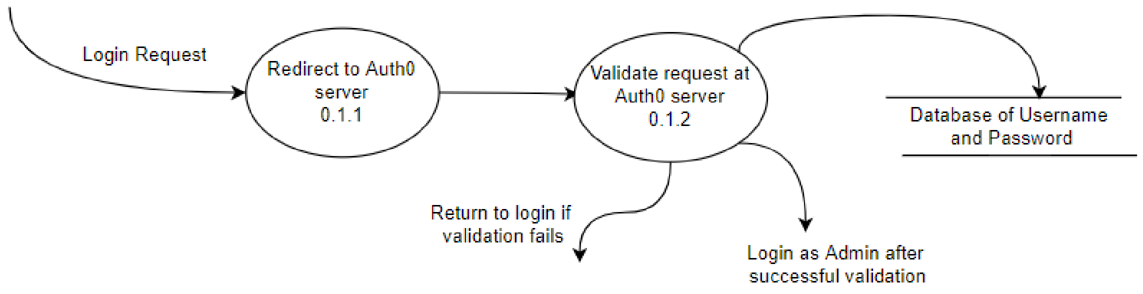


4.2 Level 1 DFD

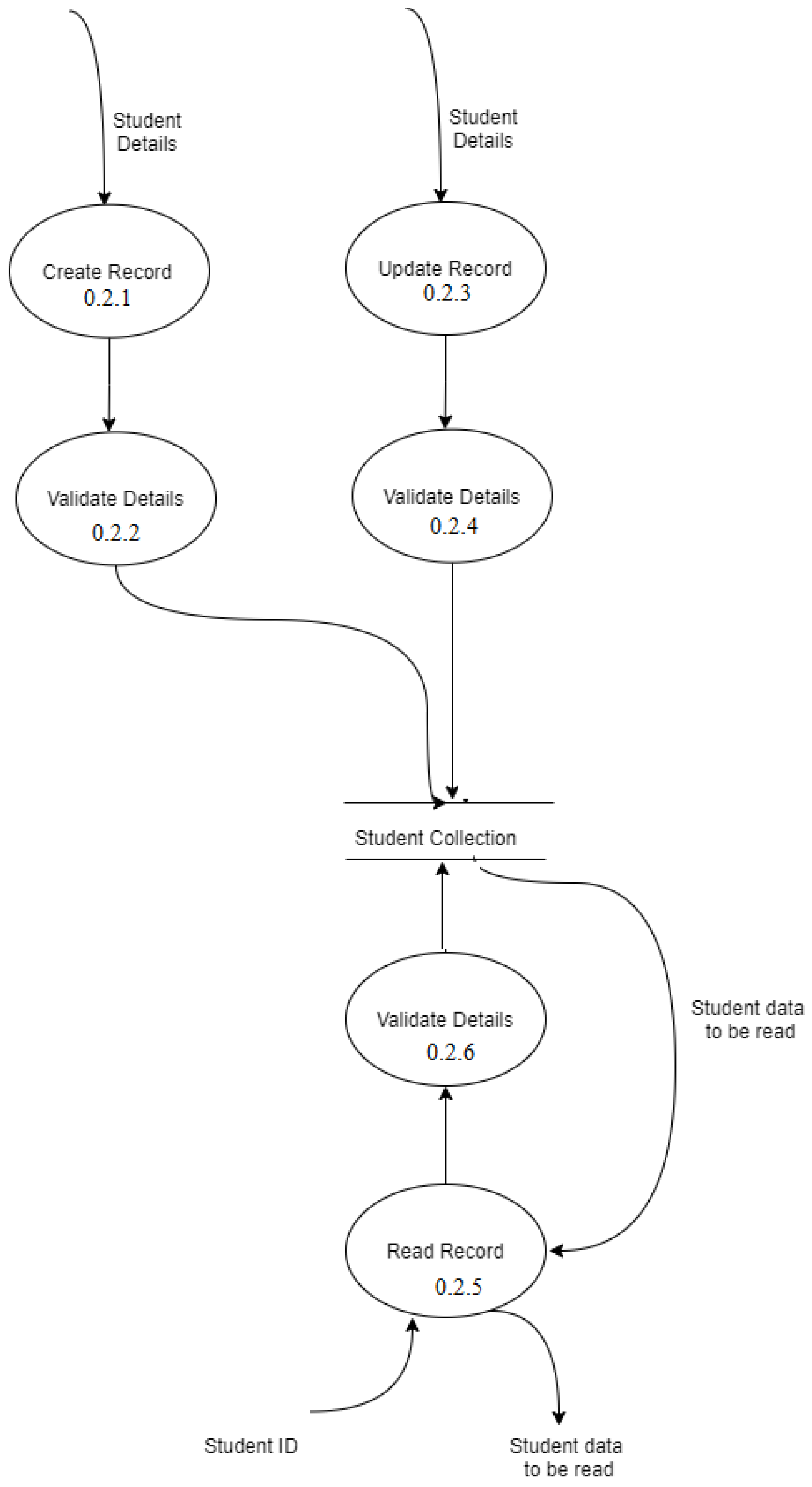


4.3 Level 2 DFD

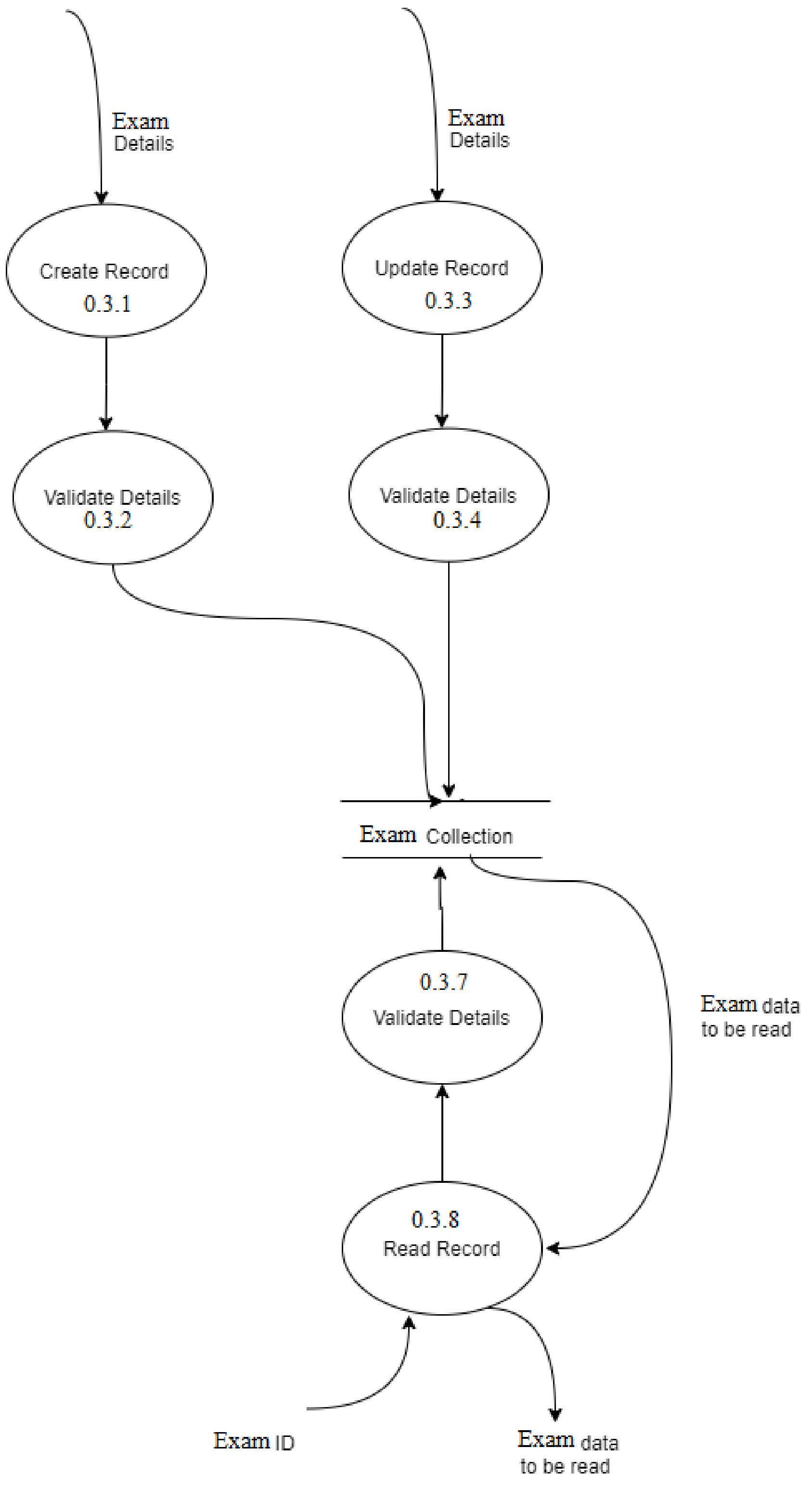
0.1 Login/Logout:


Login Request
Redirect to Auth0 server 0.1.1
Validate request at Auth0 server 0.1.2
Database of Username and Password
Return to login if validation fails
Login as Admin after successful validation

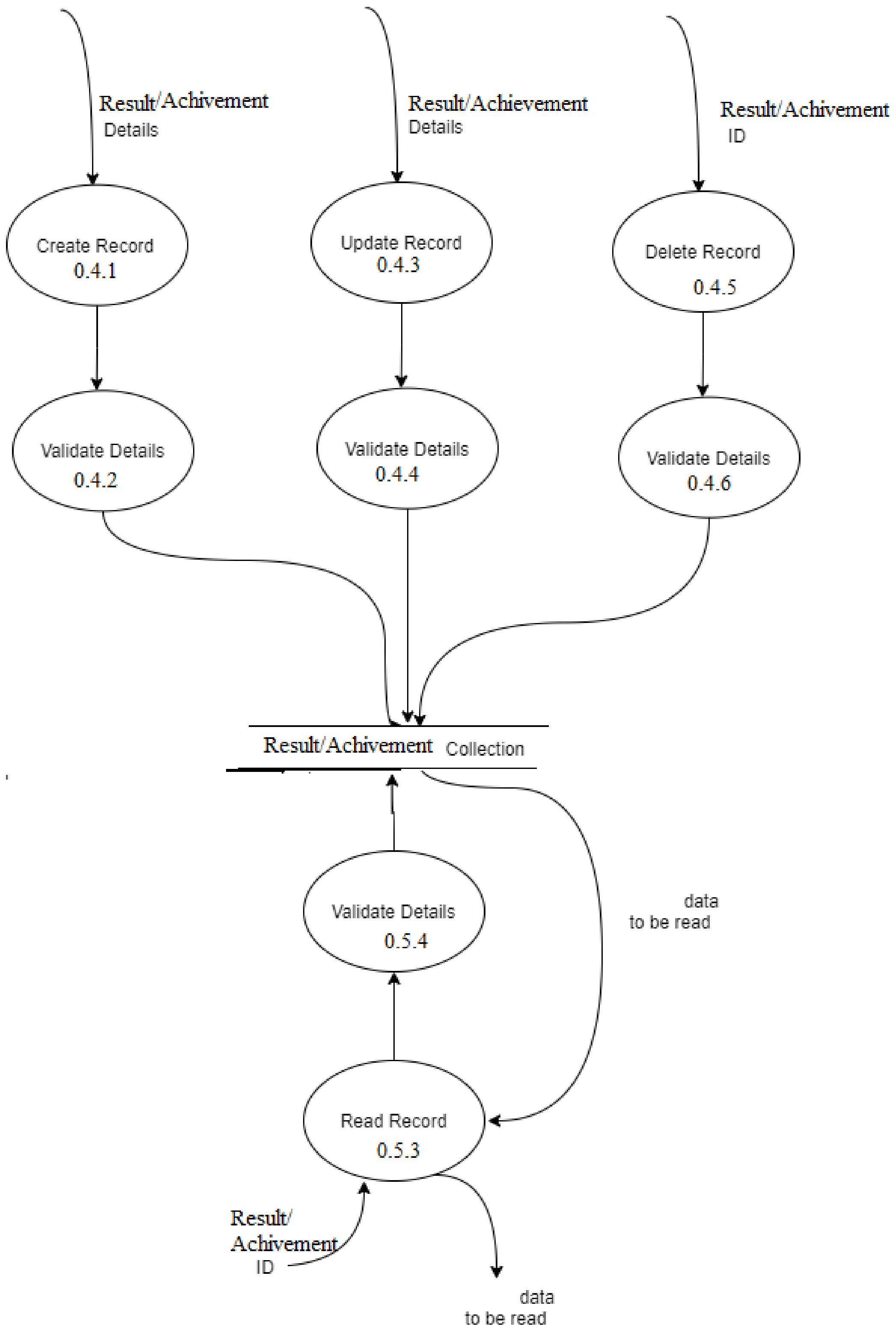
0.2 CRU operations on student records:



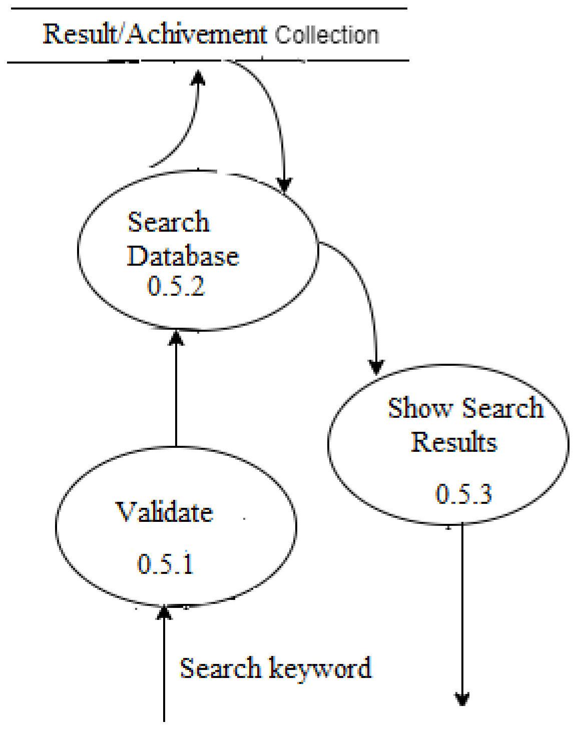
0.3 CRU operations on exam records:



0.4 CRUD operations on Results/Achievements records:



0.5 Search result/achievement record:



5.Technology Stack:

5.1 HTML: The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. These days there are various templating languages that produce html documents and make it easier to produce HTML docs. One such language is pug. In this project all HTML documents will be prepared using pug template. These templates get rendered as HTML files through the server.

5.2 CSS: Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS adds to the aesthetic of the website. It gives better accessibility to the website. In this project the CSS framework titled “Bootstrap”, made by famous social network site “Twitter”, will be used. This will make the website reactive that is suitable to view on different size screens.

5.3. Javascript: Javascript will be extensively used in this project. In the front end part javascript is used as scripts for html pages. The back end will be entirely built on javascript.

5.4. Nodejs: Node.js is a runtime environment that enables javascript to run outside a browser. Javascript was initially designed to run inside the browser. But then it was made possible to run outside the browser. This enabled rapid development of server side technology and today it is possible to build entire websites with the same language at the front end as well as backend. The project will use the “Express” library of Node.js to build the server.

5.5. MongoDB: MongoDB is a modern NoSql based database that uses documents to store data items. It is useful and performs better than traditional RDBMS in many scenarios. Also it makes maintenance of data easier. MongoDB also gives free cloud hosted database instance in it free plan under “MongoDB Atlas”. This makes it suitable for student and developers to test their product. In this project MongoDB database along with an Object Relational Mapper called Mongoose will be used.

5.6. Security and performance: Helmet.js and Compress.js should be used to secure and improve performance of the webapp. The Helmet.js implements general security best practices like preventing cross site scripting etc and compress.js compresses the html responses.

5.7. Heroku: Heroku is a cloud hosting provider that provides free hosting options for students and developers to host their apps publicly. Also it provides SSL encryption to the apps which makes it secure. It’s completely free and doesn’t require any credit card etc. Hence suitable for student developers. This project will be hosted using Heroku.

5.8. Git and Github: Git as source control tool will be used and github as remote repository to host the code.

Conclusion:

The design of the database web app is now complete. All tech stack used is open source and available on github. Now the design will be implemented and tested.