

GNG<1103/2101>
Design Project User and Product Manual

< UG Exam Database Improvement A05 UGE1 >

Submitted by:

<Shuaib Salad, 300353229>

<Joshua Anton, 300195980 >

<Fareis Canoe, 300195980 >

<Saba Raei, 300383183>

<03/12/24>

University of Ottawa

Table of Contents

Table of Contents.....	ii
List of Figures	iv
List of Tables	v
List of Acronyms and Glossary	vi
1 Introduction.....	1
2 Overview	2
2.1 Conventions.....	2
2.2 Cautions & Warnings	2
3 Getting started.....	4
3.1 Set-up Considerations	5
3.2 User Access Considerations	6
3.3 Accessing the System.....	7
3.4 System Organization & Navigation	8
3.5 Exiting the System	10
4 Using the System	11
4.1 <Given Function/Feature>	11
4.1.1 <Given Sub-Function/Sub-Feature>	12
5 Troubleshooting & Support	13
5.1 Error Messages or Behaviors	13
5.2 Special Considerations	14
5.3 Maintenance	14
5.4 Support	14

6	Product Documentation.....	16
6.1	<Subsystem 1 of prototype>	16
6.1.1	BOM (Bill of Materials)	16
6.1.2	Equipment list	18
6.1.3	Instructions.....	18
6.2	Testing & Validation.....	19
7	Conclusions and Recommendations for Future Work	20
8	Bibliography	21
	APPENDICES	10
	APPENDIX I: Design Files	10
	APPENDIX II: Other Appendices	11

List of Figures

Figure 1: A simple flowchart showing the communication between the client and server

Figure 2: Dashboard Page

Figure 3: Create an Exam Page

Figure 4: View an Exam Page

Figure 5: Login Page

List of Tables

Table 1. Acronyms	vi
Table 2. Glossary	vi
Table 3. Referenced Documents	10

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
UX	<u>User Experience. It is the individual experience users get while interacting with an interface on a product</u>
UI	<u>User Interface is the point of human-computer interaction and communication in a device.</u>
SpEL	<u>Spring Expression Language is a powerful expression language that supports querying and manipulating an object graph at runtime.</u>

Table 2. Glossary

Term	Acronym	Definition
<u>Application</u> <u>Program</u> <u>Interface</u>	<u>API</u>	<u>An application refereeing to a software with a distinct function</u>

<u>Amazon</u> <u>Webservices</u>	<u>AWS</u>	<u>Application from Amazon that</u> <u>allows you to control specific</u> <u>software functions</u>
<u>QA</u>	<u>Quality</u> <u>Assurance</u>	<u>A systematic process of determining</u> <u>whether a product or service meets a</u> <u>specified requirement</u>
<u>JSW</u>	<u>JSON Web</u> <u>tokens</u>	<u>Internet standard for creating data</u> <u>with optional signature and/or</u> <u>optional encryption</u>
<u>CI/CD</u>	<u>Continuous</u> <u>Integration and</u> <u>continuous</u> <u>deployment</u>	<u>Software practice in which</u> <u>incremental code changes are made</u> <u>frequently and reliably</u>

1 Introduction

This User and Product Manual (UPM) provides the information necessary for exam practitioners to effectively use the Exam Database Web Application and for prototype documentation. With its clear step-by-step instructions, system explanations, and troubleshooting techniques, the handbook is designed to be easy to use for both technical and non-technical users. An outline of the system's architecture, setup needs, and intended purpose is given first. The document then goes on to describe how to manage examinations, log in, interact with the front-end interface, and maintain the database. Instructions for mistake rectification, recovery, and system maintenance are also included in the handbook. It is designed to give administrators, technical support teams, and exam takers all the tools they need to use and troubleshoot the system. In terms of security and privacy, the manual highlights the importance of secure login protocols, data protection measures for exam data, and safe handling of sensitive information during database transactions. The document assumes that users have basic knowledge of web applications but provides sufficient guidance for anyone new to the system.

2 Overview

The problem this Exam Database Web Application aims to solve is the inefficient and disorganized management of exam data for educational institutions or exam practitioners.

Managing exam information manually or through inefficient systems can lead to errors, data inconsistencies, and an increased administrative burden. This issue becomes even more critical in environments with large volumes of exam data that need to be updated, retrieved, and managed frequently. A robust, user-friendly, and efficient system is necessary to handle this data, ensuring accuracy, security, and ease of access.

The fundamental needs of the user include the ability to securely log in, add and modify exams, view and search for exam data, and import exam data from Excel files for bulk updates. Users require a simple and intuitive interface that streamlines these processes, reducing time spent on data management while maintaining data integrity. Additionally, the system must be secure, ensuring the privacy of exam data.

What differentiates our product from others is its combination of a user-friendly interface with robust backend features. Our system allows users to seamlessly manage and modify exams, import Excel files to streamline bulk updates, and view data in an organized, easy-to-navigate interface. The integration of security features, such as secure login protocols, further enhances the product's appeal, ensuring both ease of use and data protection.

The key features of the Exam Database Web Application include secure user authentication via AWS Cognito, exam management (add, modify, and delete), a user-friendly frontend built with React.js, and backend functionality powered by Node.js and MySQL for data storage.

Additionally, the system supports bulk data import through Excel, allowing exam administrators to quickly update exam records.

The architecture of the system is web-based, consisting of a frontend user interface built with React.js that interacts with a Node.js backend to manage database operations. The database, which is implemented using MySQL, stores exam-related data such as titles, descriptions, dates, and other details.

3 Getting started

Our product is a software web application which allows exam practitioners to easily manage exam related data for current medical exam practitioners. In order to get start with our product it is ideal that the user have the necessary infrastructure and technologies in place. It would be best for the user to have a remote server to host and store all the data for the exam related material. And a computer to act as the client, to show the front-end (what user will see) of our application. However, it is possible for a user to store its related data and present the frontend all in one computer.

Once the user has set-up their necessary set-up in terms of hardware, the client must then install and incorporate the necessary software. It is important to note that our product is a website. Thus, it is not necessary for the user to install this software but is optimal for them so they can later maintain, adjusts, and further improve the product. Specially our client's developers. However, from this point on the user can start using our product after entering the correct URL.

The necessary software to maintain our product varies from a range of front-end to back-end technologies. Our client must initially install NPM. A node package manager is a free and open-source technology to easily package and manage your JavaScript programs (which our site runs on). This will help to later easily update, install and improve your dependencies for your website. (Install at: <https://nodejs.org/en/download/package-manager>) Secondly, the client must install Node.js, a JavaScript runtime environment that allows you to run JavaScript code outside of a browser. It will help our client developers to maintain the server-side applications, maintain their exam data. For our front-end our client must install react.js, a JavaScript library to allow you to build user interfaces using components and to have nice user-experience. It is very important if the client ever wants to later update the view of the website. The client can install using NPM via:<npm install react>

After the user/developer maintainer has gotten the necessary setup complete. They can now start using our product on the innovation portal site.

3.1 Configuration Considerations

System Communication & Configuration

a. Communication Flow

The software prototype consists of several key components that work together seamlessly:

- **User Interface (UI):** This is where the user interacts with the system, entering data and receiving feedback. The UI could be a website or an application running on a computer or mobile device.
- **Application Logic:** This part of the system processes the information from the UI. It determines what actions to take, such as saving or retrieving data.
- **Database:** The database stores all the data (like exams, user information, and results). The application logic communicates with the database to retrieve or save information as needed.
- **Output:** After processing, the system provides the results or confirmation of actions to the user through the UI. The client needs to just access the UI via HTTP request. Simply searching for the innovation portal link.

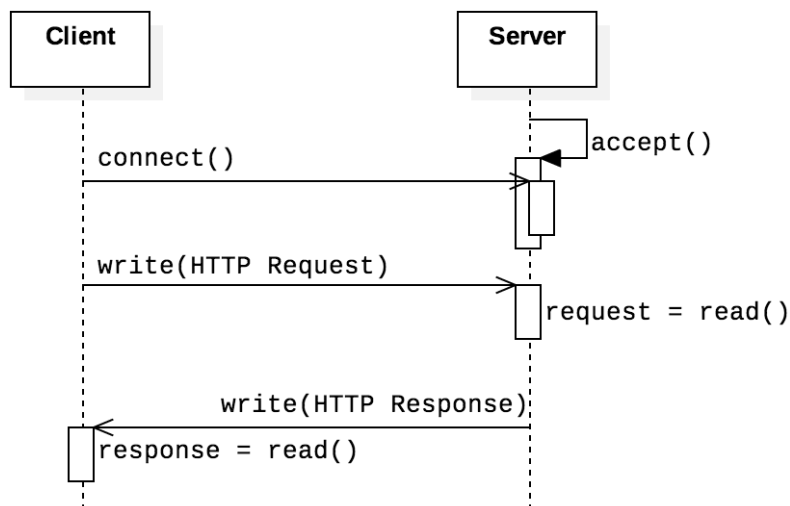


Figure 1: A simple flowchart showing the communication between the client and server (storing data).

Description: The user interacts with the interface, which sends input to the application logic (our server). The application logic processes this input, sometimes interacting with the database to retrieve or save data and then presents the output to the user.

b. Input & Output Devices

Our product is on a website so the user will only have to enter inputs for exam titles, questions and any necessary information that is prompted for them when it's needed. It is assumed that the user has a computer with keyboard or some way to enter inputs to their device. The outputs will be a any queried results and operation our codebase will perform and output it to the console/screen (ex: one of our web pages).

3.2 User Access Considerations

Our product is meant to restrict access and only allow specific users to interact with our site. Since we are presenting important exam-related information, thus, we made a secure log-in so not everyone can enter our site. So, user access will be restricted to exam practitioners, people giving out the exams. As well as, hired software developers that need to maintain the site.

Exam Practitioners

Description: Create, update, and manage exams and related content within the database.

- **Access Permissions:**
 - Add and edit exam details (e.g., title, date, description).
 - View exam data and reports.
 - Upload or modify supporting materials for exams (if applicable).
- **Restrictions:**
 - Cannot modify system settings or manage user accounts.
 - Limited access to reports containing sensitive data unless explicitly granted.

Developers

Description: This group handles technical maintenance, troubleshooting, and backend database management.

- **Access Permissions:**
 - Monitor system health and ensure uptime.
 - Access database-level configurations for backups or maintenance.
 - Resolve technical issues related to performance or user access.
- **Restrictions:**
 - Limited to backend operations and not involved in content creation or user management.
 - Should not view and analyze exam contents

3.3 Accessing/setting up the System

To turn on the system, ensure the computer or server hosting the application is powered on and connected to the internet (if cloud-based). Launch the software by double-clicking the desktop icon or entering the provided URL in a web browser. Once the system starts, the user is directed to the login screen. To access the system, enter your assigned user ID and password. New users can request an ID through the administrator, who will create an account and provide initial login credentials. To reset or change a password, users can click the “Forgot Password” link on the login screen, enter their registered email, and follow the steps in the email to set a new password securely.

For software maintainers, they must ensure the server is running by following the set of commands:

To access frontend:

```
cd frontend
```

```
npm install
```

```
npm start
```

To access backend:

```
cd DatabaseBackend
```

```
gradlew build
```

```
gradlew bootRun
```

OR

```
cd Databasebackend
```

```
./gradlew build
```

```
cd ..
```

```
docker-compose -f docker-compose-dev.yaml up --build
```

3.4 System Organization & Navigation

Organization and Navigation of the System

The software prototype is organized into a central **Dashboard** (home page) that acts as the main hub for accessing all system features. From the dashboard, users can navigate to various sections based on their roles and permissions. Each section is interlinked through clearly labeled menus and buttons, ensuring seamless access to the system's functionalities.

Dashboard (Home Page)

The **Dashboard** is the landing page after login and provides an overview of the system. It includes key metrics (e.g., total exams, upcoming exams, number of bugs), shortcuts to frequently used features, and a navigation bar for accessing other sections.

- **Navigation Bar:** Located at the top or side of the screen, it allows users to switch between sections such as "Manage Exams," "Create an Exam," and "View Exams."
- **Quick Links/Widgets:** Displayed as buttons or cards on the dashboard for instant access to popular actions like "Add Exam" or "View Results."



Figure 2: Dashboard Page

Manage Exams

This section enables users to create, edit, view, or delete exams. The navigation path begins from the **Dashboard** via the **Manage Exams** menu option.

- **Add Exam:** Opens a form to input exam details, including title, date, and description.
 - **Navigation Path:** Dashboard → Manage Exams → Add Exam
 - **Export Data:** Allows exporting reports in formats like Excel or CSV.
- **Edit Exam:** Lists all exams with an "Edit" button beside each. Clicking "Edit" opens a form pre-filled with the existing details for modification.
 - **Navigation Path:** Dashboard → Manage Exams → Select Exam → Edit Exam
- **View Exams:** Displays a table with all exams. Users can then look at the exams they have created, and the questions associated with that exam.
 - **Navigation Path:** Dashboard → Manage Exams → View Exams
- **Delete Exam:** Includes a "Delete" button beside each exam in the list. A confirmation dialog prevents accidental deletion.

- **Navigation Path:** Dashboard → Manage Exams → View Exams →
- **Export Data:** Allows exporting reports in formats like PDF or CSV.

Logout

The **Logout** option ensures users can securely exit the system, redirecting them to the login page.

It is accessible via the profile section.

- **Navigation Path:** Dashboard → Profile Icon → Logout

Table 1. Tabulation Organization of system

Feature	Navigation Path	User Role
Add Exam	Dashboard → Manage Exams → Add Exam	Admin/Educator
Edit Exam	Dashboard → Manage Exams → Select Exam → Edit Exam	Admin/Educator

3.5 Exiting the System

To properly exit or turn off the system, follow these steps:

1. **Save Work:** Ensure all changes, such as updates to exams or user profiles, are saved. Confirm that no unsaved data is left in progress forms or input fields.
2. **Log Out:** Navigate to the **Profile Icon** or **Settings Menu** in the top-right corner of the screen. Click the **Logout** button to securely exit your session. This prevents unauthorized access to the system if the device is shared. Important to keep edits secure.
3. **Close Application:**
 - a. For desktop applications: Close the program by clicking the "X" button in the top-right corner of the window or selecting "Exit" from the file menu.
 - b. For web-based applications: Close the browser tab or window used to access the system.

4. Power Down (if applicable):

- a. For local servers: Shut down the server hosting the application by following standard shutdown procedures for the operating system.
- b. For cloud-hosted systems: No additional action is needed unless specified by the hosting provider.

By following these steps, users ensure their session ends securely, and the system is ready for the next use.

4 Using the System

The following sub-sections provide detailed, step-by-step instructions on how to use the various functions or features of the Exam Database Web Application.

4.1 Add Exam Feature

The "Add Exam" feature allows users to create a new exam by clicking the "Add Exam" button on the dashboard or navigation menu. This action opens a form where users can input details such as the exam title, date, description, and assigned categories. Once all fields are completed, clicking "Save" stores the exam in the database, and a confirmation message is displayed. This process ensures that exams are organized and categorized appropriately for easy retrieval.

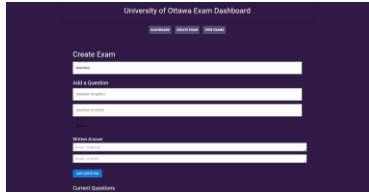


Figure 3: Create an Exam Page

4.1.1 View Exam Feature

Users can view all exams through the "View Exams" section, accessible via the main menu. This displays a searchable and sortable table listing all exams, with filters for criteria like date, category, or keyword. Clicking on an exam in the table reveals detailed information, such as the full description and associated metadata. This feature ensures quick access to exam data for review or management.

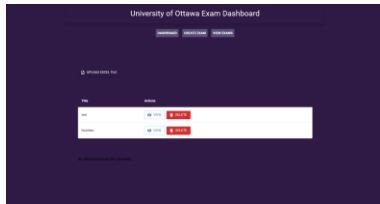


Figure 4: View an Exam Page

4.1.2 Secure Login Feature

Users log in securely by entering their credentials (user ID and password) on the login page. Passwords are encrypted and verified against stored data in the database, ensuring security. Multi-factor authentication (if enabled) adds an extra layer of protection. Failed login attempts trigger alerts, and users can reset their password through a secure email-based process. This feature ensures that only authorized individuals can access the system.

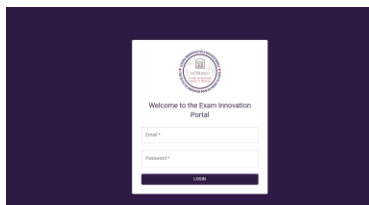


Figure 5: Login Page

4.1.3 Data Storage Feature

Exams are stored in a relational database with fields for title, date, description, category, and metadata like creator and timestamps. Each exam entry has a unique identifier, enabling seamless

updates and cross-referencing. Data is indexed to optimize search and retrieval and backed up regularly to prevent loss. This ensures that the exam database remains robust, secure, and efficient for all users.

5 Troubleshooting & Support

The system includes recovery and error correction procedures to address common issues and ensure uninterrupted functionality. Each procedure focuses on resolving specific errors effectively and providing guidance to non-technical users.

5.1 Error Messages or Behaviors

Login Issues

If users encounter login errors, such as incorrect credentials or account lockouts due to multiple failed attempts, they should first verify their entered username and password, ensuring caps lock is off. If credentials are forgotten, the user can click the "Forgot Password" link, which sends a secure email with password reset instructions. Account lockouts require contacting the system administrator to unlock the account.

Input Errors

When invalid or incomplete data is entered in forms, the system generates on-screen error messages specifying the fields that need correction. Users must review these messages, amend the input, and re-submit the form. This ensures that all required data is accurate and consistent before being saved to the database.

Excel Import Issues

For errors encountered during Excel spreadsheet uploads, such as mismatched data formats or missing columns, the system provides an error log detailing the specific problems. Users can refer to this log to adjust the spreadsheet formatting as required and re-upload the corrected file.

5.2 Special Considerations

In the event of a system crash or power failure, if the data has not been saved to our database the current progress may be lost. Although all current saved information is stored in our remote server. Thus, the user must simply restart their device and log back onto the website. The client devices shall connect back to our server and can use the site as intended.

5.3 Maintenance

Although our client is the exam practitioners and would not necessarily be working to maintain the website. There is a possibility that maintenance is needed. In that case, the client in house developers might need to maintain the exam database website for optimal performance, security, and usability. They would mainly work to ensure security updates are done at appropriate times. Conduct periodic scans for malware and unauthorized access attempts. As well as our connection to AWS third party provider is secure.

Secondly, they will mostly work with server maintenance, ensuring the connection to the site is secure, following TCP/IP universal protocol. Also ensuring that stored data is accessible.

5.4 Support

For emergency assistance or system support, users should reach out to the faculty of medicine at the university of Ottawa. They will have the software engineers that can help them to help them get the product back to working. They can report the problems by checking the bugs they have

detected on the dashboard. On the innovation their should be a place to discuss issue, although that is not a part of the project we worked on.

6 Product Documentation

The final prototype of the exam database system was designed with a strong emphasis on user experience, security, and efficient data management.

6.1 Subsystems of prototype

6.1.1 BOM (Bill of Materials)

Item	Description	Quantity	Unit Cost	Total Cost	Link
AWS Cognito	Authentication and Authorization platform for user management	1	\$0 (Free Tier)	\$0	https://aws.amazon.com/cognito/
IntelliJ IDEA	Integrated development environment (IDE) for Java development	1	\$0 (Community Edition)	0\$	https://www.jetbrains.com/idea/download/#section=windows
GitHub	Version control platform for collaboration and source code management	1	0\$	0\$	https://github.com/
Node.js	Backend runtime environment for JavaScript	1	0\$	0\$	https://nodejs.org/en

React.js	Frontend framework for building the UI	1	0\$	0\$	https://react.dev/
SQL (MySQL or PostgreSQL)	SQL-based relational database management for structured data storage	1	0\$	0\$	https://dev.mysql.com/downloads/installer/

Focusing on the user experience perspective, we focused on creating an intuitive and seamless interface. The homepage of the system presents a clean layout with clearly labeled buttons and navigation options, such as "Add Exam," "View Exams," and "Modify Exam." Users can easily navigate between sections, with each button performing specific actions like opening the exam management page or displaying a detailed view of the selected exam. Clicking on the "Add Exam" button brings up a form where users can input exam details like the exam name, date, and associated questions. To ensure ease of use, the system also provides users with tooltips and pop-up instructions where necessary, helping them understand the actions they can take at any given point.

The security aspects of the system were an important client requirement. Our group decided to use a third-party provider to authenticate our clients. Using AWS Cognito to securely allow users to log-in with a robust token-based authentication method. Users can create accounts and reset passwords, with multi-factor authentication (MFA) enabled for additional security during login. These measures significantly reduce the risk of data breaches and ensure only authorized users can access the exam data.

Database storage is a critical component of the system. We chose MySQL as the relational database management system due to its reliability and scalability. The exams, questions, and user information are stored in well-organized tables, each with proper indexing for quick access and efficient searching.

6.1.2 Equipment list

To build the exam database subsystem, the following equipment are required.

1. **Development Machines (Laptops/Desktops):**
 - a. Required for coding, database management, and server configuration.
2. **Network Equipment:**
 - a. **Router/Modem:** For internet connectivity during development and testing, including for cloud services and online repositories.
3. **Servers (For Development/Testing/Production):**
 - a. **Local Development Server:** Could be a local server or a cloud-based server for hosting the development environment.
 - b. **Production Server (Cloud):** Used for hosting the live application, which interacts with the database and serves the front-end.
 - c. **Database Server:** Required to host and manage the relational database (e.g., MySQL or PostgreSQL) that stores exam data.
4. **Security Equipment:**
 - a. **AWS Cognito:** Subscription to the amazon web servies so you can ensure connection are good.
5. **Testing Devices:**
 - a. **Web Browsers:** For testing the front-end (React.js) and verifying that all functionality, including login, data display, and interactivity, works as expected.
6. **Development Tools:**
 - a. **Integrated Development Environment/Git:** For running code and sharing code with other developers.
 - b. **API Testing Tools:** (e.g., Postman) for testing and validating API endpoints and database interactions.

6.1.3 Instructions

First set up the development environment, source control, and the necessary software dependencies.

- a. Install IntelliJ
- b. Install Git
- c. Install MySQL, Node, React

Setting up the development environment is an important first step to ensure collaboration in the exam database subsystem. **IntelliJ IDEA** is needed as it provides an integrated environment for writing, testing, ensuring that the backend logic for managing exam data is efficiently developed. **Git** is essential for version control, allowing the team to collaborate seamlessly and track code changes. **MySQL** is a reliable relational database management system to store and organize exam-related data such as exam details, dates, and descriptions. Node.js & React.js is needed for the frontend and backend. Since software architecture is a long process there is no right way to do it. Thus, no step-by-step instruction to how to build the prototype. Once they have the necessary tool stated above, they can code the code at our github repository: https://github.com/Jrjkar-pk/GNG2101_UGE

6.2 Testing & Validation

To validate the final design of the exam database prototype, we employed a combination of **Selenium** automated tests and manual testing based on strict user requirements. Selenium was used to automate user interactions, ensuring that the system's frontend responded correctly to actions such as adding, modifying, and viewing exams. Manual tests were conducted to evaluate the system's performance, usability, and adherence to the defined user requirements. This approach ensured that the system met the expected functionality, handled edge cases, and provided a seamless user experience. All tests confirmed that the system performed as intended with minimal issues.

7 Conclusions and Recommendations for Future Work

The significance of striking a balance between features and time restrictions is one important lesson. If we had had more time, we would have given top priority to upgrading the exam modification interface, introducing SMS-based authentication, and scalably and performance-enhancing the backend. A more advanced reporting system that could have produced comprehensive statistics on exam usage, revisions, and user interactions would also have been included. Another key takeaway was project management and making sure our code is felixable and robust.

In terms of enhancing the authentication process, we initially considered implementing SMS-based authentication to further secure the user login experience. This would involve integrating an SMS API, such as Twilio, which would send a one-time passcode (OTP) to the user's phone number during the login process. This would ensure a higher level of security, especially for exam practitioners who require a more robust verification system. Although we did not have the time to implement this feature, it would significantly improve the overall security of the system by adding an additional layer of verification.

To improve the user experience when modifying an exam, we would focus on making the interface even more intuitive. This could include adding features such as auto-complete for fields, real-time validation for input fields, and tooltips or help icons to guide the user through the process. We would also streamline the navigation to make it more fluid, reducing the number of steps required to modify exam details. Additionally, we could implement version control or audit logs for exam modifications, so that users can track the history of changes made to each exam.

On the backend side, there is significant potential to improve the efficiency and scalability of the system. We could implement optimized database queries, use caching mechanisms for frequently accessed data, and consider a more sophisticated architecture (e.g., microservices) to better handle increasing amounts of data and users. Integrating more advanced database management features, such as indexing and partitioning, would help with faster data retrieval and ensure that the system can scale as more exams and users are added.

8 Bibliography

Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Boston: Addison-Wesley.

GitHub, Inc. (n.d.). *GitHub Official Website*. Retrieved from <https://github.com/>

Oracle Corporation. (n.d.). *Download MySQL Community Server*. Retrieved from <https://dev.mysql.com/downloads/installer/>

Rosell, G. (2016). *Pro Node.js for Developers* (1st ed.). Berkeley: Apress.

Hackl, R. (2019, May 28). *AWS Cognito: The Authentication Service for Modern Apps*. Amazon Web Services Blog. Retrieved from <https://aws.amazon.com/blogs/mobile/aws-amplify-authentication-with-aws-cognito/>

Node.js Foundation. (n.d.). *Download Node.js*. Retrieved from <https://nodejs.org/>