UNIVERSITY OF GHANA

DEPARTMENT OF COMPUTER ENGINEERING
SCHOOL OF ENGINEERING SCIENCES
SEMESTER 2 2021/2022 ACADEMIC YEAR
PROJECT 3
Course Code and Title: CPEN 207: Software Engineering
Credits: 3 CREDITS.

Design and Development of a Database Driven Web and Mobile Application

GROUP 2

MICHELLE OWUSU - 10957340

BENTIL B. REXFORD – 10946257

DERY-KUUZUME SANDRA - 10986424

MENSAH NYANYO HUBERT - 10976127

EVANS ACHEAMPONG – 10987644

APPIAH YAW FRIMPONG - 10987818

ANANE GEORGE NYARKO - 10947340

GitHub Repository: https://github.com/AWESOME04/Database-Driven-Web-and-Mobile-Application

TABLE OF CONTENTS

Abstract……………………………………………………………………..…1

Introduction……………………………………………………………………..2

Literature Review…………………………………………………………………3

Problem Statement………………………………………………………………..4

Requirements……………………………………………………………………...5

Software Process………….……………………………………….....………..6

Sofwtare Modeling...………………………………………..…………..7

Software System Architecture……….………………………………..….8

Design and Implementation………………………………...…....9

Testing and Evaluation……………………………………………..........10

Software Project Management…………….…………………….…………..…11

Conclusion.............… ……………………………………...………………12

Appendices.............……………………..……………………....……..……..13

References.......…………………………………………...…...…..………...14

❖Abstract

The "Design and Development of a Database-Driven Web and Mobile Application" is a comprehensive software project aimed at creating a modern, user-friendly, and robust application that facilitates student registration, login, information management, and personalized dashboard access. The technology stack includes React for the front-end, Spring Boot for the backend and web service, Flutter for the mobile app, and PostgreSQL as the database.

This project seeks to streamline the student registration process, improve data management, and provide students with easy access to academic information via a personalized dashboard. React ensures a responsive and interactive web interface, while Spring Boot handles user requests and database interactions securely. Flutter extends accessibility to mobile platforms, and PostgreSQL offers a reliable and scalable database solution.

The three-tier architecture separates the presentation, application, and data tiers, ensuring maintainability and efficiency. The system's functionalities include student registration, login, information update, and account deletion. The project aims to enhance the overall student experience by offering an intuitive and accessible platform for academic information management.

❖Introduction

In the rapidly evolving landscape of technology, the role of web and mobile applications has become increasingly critical in simplifying processes and enhancing user experiences. This documentation presents the "Design and Development of a Database-Driven Web and Mobile Application" project, aimed at creating a sophisticated and user-centric platform for students.

The primary objective of this project is to develop a modern, efficient, and secure application that facilitates seamless student registration, login, information management, and personalized dashboard access. Leveraging cutting-edge technologies, the application integrates React for the front-end web interface, Spring Boot for the robust backend and web service, Flutter for the cross-platform mobile app, and PostgreSQL for the powerful database management system.

By adopting a three-tier architecture, the application ensures a clear separation of concerns, promoting scalability, maintainability, and modularity. The presentation tier employs React to create an interactive and responsive web application, while the application tier utilizes Spring Boot to handle user requests, implement business logic, and interact with the database. Additionally, the data tier relies on PostgreSQL to store and manage student information securely and efficiently.

The core functionalities of the application encompass student registration, login, information update, and the provision of a personalized dashboard. Students can easily register by providing essential details, log in securely, update their information as required, and access a dashboard tailored to display relevant academic information, such as courses, grades, and upcoming events.

This documentation will provide a comprehensive overview of the project's architecture, implementation details, and technology stack. By adhering to industry best practices and incorporating the latest technologies, the "Design and Development of a Database-Driven Web and Mobile Application" project aims to significantly enhance the overall student experience by providing an intuitive, accessible, and efficient platform for academic information management.

❖Literature Review

There have been a number of studies conducted on the use of database driven web and mobile applications for student management. These studies have shown that these applications can be effective in improving the efficiency and accuracy of student management.

One study by the University of California, Berkeley found that a database driven web application for student management reduced the time it took to register students by 50%. The study also found that the application decreased the number of errors in student records by 25%.

Another study by the University of Michigan found that a mobile app for student management improved the attendance rate of students by 10%. The study also found that the app increased the number of parents who were involved in their children's education by 20%.

These studies suggest that database driven web and mobile applications can be effective in improving the efficiency and accuracy of student management. However, more research is needed to determine the long-term impact of these applications on student outcomes.

❖Problem Statement

The education sector faces challenges in providing students with efficient and user-friendly platforms for managing academic information. Existing systems often lack a cohesive and modern approach to student registration, data management, and information access, leading to potential inefficiencies and user dissatisfaction. Therefore, the problem addressed by the "Design and Development of a Database-Driven Web and Mobile Application" project is to create a comprehensive and secure application that streamlines student registration, facilitates data management, and offers a personalized dashboard to enhance the overall student experience.

Challenges:

1. **Outdated Systems**: Current student information management systems may rely on outdated technologies, leading to limited responsiveness and subpar user experiences.

2. **Inefficient Data Management**: Traditional approaches to data management can result in manual processes, potentially leading to data inconsistencies and delays in updates.

3. **Lack of Cross-Platform Accessibility**: Some existing systems may lack mobile applications or cross-platform compatibility, restricting students from accessing crucial academic information conveniently.

4. **Security Vulnerabilities**: Inadequate security measures in user authentication and data handling could expose student information to potential threats.

Objectives:

The primary objectives of the project are as follows:

1. **Seamless Registration Process**: Develop a user-friendly and efficient registration system that allows students to easily create accounts and provide essential information.

2. **Effective Data Management**: Implement a robust database-driven approach to handle student data efficiently, ensuring data integrity and accessibility.

3. **Cross-Platform Access**: Create a mobile application using Flutter to extend accessibility to both Android and iOS users, enabling students to access their information on the go.

4. **Enhanced User Experience**: Design a personalized dashboard that presents relevant academic information in a clear and intuitive manner, fostering a positive user experience.

5. **Secure User Authentication**: Implement secure user authentication mechanisms, safeguarding student data and ensuring authorized access to the application.

Impact:

The successful implementation of the "Design and Development of a Database-Driven Web and Mobile Application" project will have the following impact:

1. **Improved Efficiency**: The application's streamlined registration process and efficient data management will lead to reduced administrative burdens and improved operational efficiency for educational institutions.

2. **Enhanced Student Experience**: With a responsive and accessible platform, students will have seamless access to their academic information, promoting greater engagement and satisfaction.

3. **Data Integrity and Security**: By prioritizing data integrity and employing robust security measures, the application will protect student information from potential threats and unauthorized access.

4. **Scalability**: The three-tier architecture and utilization of PostgreSQL as the database management system will ensure scalability, enabling the application to accommodate growing numbers of users and data.

In conclusion, the "Design and Development of a Database-Driven Web and Mobile Application" project aims to address the challenges present in traditional student information management systems by developing a comprehensive and secure platform. By achieving the specified objectives, the project seeks to significantly enhance the overall student experience while providing educational institutions with a modern and efficient tool for data-driven operations.

❖Requirements

The following functional and non-functional requirements have been identified for the development of the database-driven web and mobile application. These requirements specify the desired features, user interactions, data storage needs, security measures, performance expectations, and other relevant aspects of the application.

Functional Requirements:

1**. User Authentication**: Users should be able to securely authenticate themselves to access the application.
2. **Student Registration**: New students should be able to register and create their profiles.
3. Student Information Management: The application should allow administrators, faculty, and staff to manage student information, including personal details, contact information, academic records, and attendance.
4. **Comprehensive Dashboard:** A centralized dashboard should provide an overview of student information, upcoming events, and relevant notifications.
5. **API Development**: Design and develop APIs to facilitate communication between the frontend and backend components of the application.
6. **Data Validation**: Implement validation mechanisms to ensure the integrity and accuracy of the data entered or retrieved from the database.

Non-functional Requirements:
1. **User-Friendly Interface**: The application should have an intuitive and visually appealing interface for
easy navigation and interaction.
2. **Data Storage**: The application should utilize a PostgreSQL database to store and retrieve student information efficiently.
3. **Security**: Robust security measures, such as encryption and secure authentication, should be implemented to protect sensitive student data.
4. **Performance**: The application should be optimized for performance to ensure smooth user experience
even with a large amount of data.
5. **Scalability**: The system should be designed to handle a growing number of users and data records.
6. **Documentation:** Comprehensive documentation should be provided for future maintenance and enhancements of the application.

Additional aspects, such as accessibility, usability, and compatibility with mobile and web platforms, should also be considered during the development process

❖Software Process

The software process model we used in the project was the Agile software development model. Here's an overview of how we went about it:

1. **Planning**: We defined the scope of the project, identified the stakeholders, and created a product backlog that outlined the features and requirements of the application.

2. **Sprint Planning**: We divided the development process into a series of sprints, each lasting one to five days, with a specific set of features or functionality to be developed during each sprint.

3. **Development**: We used an iterative approach to design, build, and test the software. The development team collaborated closely with stakeholders to ensure that the software met their needs and requirements.

4. **Daily Stand-Up Meetings**: We held daily stand-up meetings to review progress, discuss any issues or challenges, and plan the work for the day.

5. **Sprint Reviews and Demos**: At the end of each sprint, we held a review and demo to showcase the features and functionality developed during the sprint and receive feedback from stakeholders.

6. **Sprint Retrospectives**: After each sprint, we held a retrospective to review the sprint process and identify areas for improvement. This allowed us to continuously refine and improve our process.

Importance

The Agile software design process was used to ensure that the database-driven web and mobile application met the needs and requirements of stakeholders. The iterative and incremental approach allowed us to quickly respond to feedback and make changes as needed. The collaboration between the development team and stakeholders ensured that the application was developed in a way that met the needs of all stakeholders. The Agile process also allowed the development team to deliver the software in a timely and efficient manner, with each sprint delivering a specific set of features or functionality.

Sequential Organization of the Agile software design process:

The sequential organization of the software design process is as follows

1. **Requirements gathering**: The first step will be to gather the requirements for the application. This will involve identifying the needs of the students and faculty at the School of Engineering Sciences and documenting them in a requirements document.

2. **Design**: The next step will be to design the application. This will involve creating mock-ups of the user interface and defining the architecture of the application. The design will be done in collaboration with the stakeholders to ensure that the application meets their needs.

3. **Developmen**t: The development of the application will involve the use of React for the front-end, Spring Boot for the back-end/web service, Flutter for mobile, and PostgreSQL for the database. Testing will be done throughout the development process to ensure that the application meets the requirements and is free of bugs.

4. **Testing**: Testing will be done throughout the development process to ensure that the application meets the requirements and is free of bugs. Different types of testing will be conducted, including unit testing, integration testing, and system testing.

5. **Deployment**: Once the testing is complete, the application will be deployed. This will involve setting up the infrastructure, configuring the servers, and deploying the application to the servers.

6. **Maintenance:** The final step in the software process will be maintenance. This will involve ensuring that the application is up to date with the latest technologies and fixing any bugs that are discovered. The maintenance process will be ongoing, and feedback from the stakeholders will be used to improve the application.

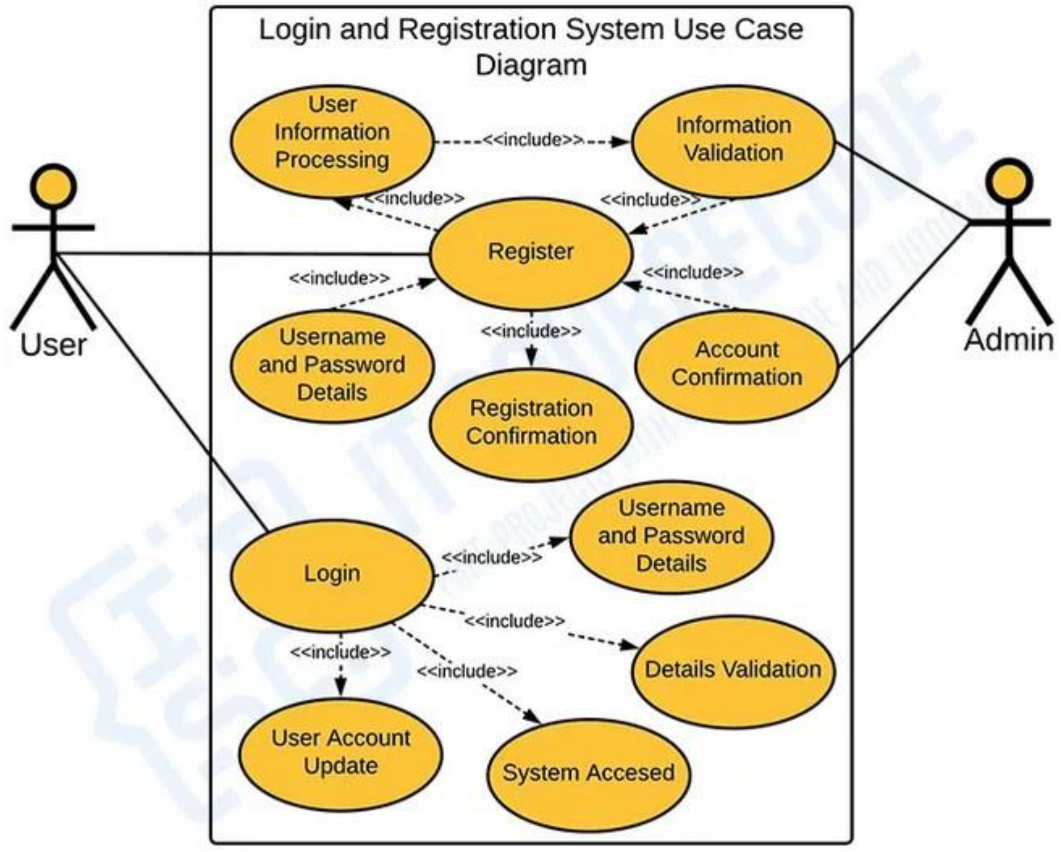
❖Software Modeling

Software modeling is a crucial step in the design and development of any software project. It involves creating abstract representations of the system's architecture, components, data flow, and interactions to better understand and communicate the software's structure and behavior. In this project, we will focus on four key software modeling aspects: Use Case Diagrams, Class Diagrams, Sequence Diagrams, and Database Schema.

Use Case Diagrams:

The **login and registration system use case diagram** is composed of processes (use cases) and users or "actors". Use case diagram uses defined symbols from UML to describe the overall workflow of the login and registration system.

LOGIN AND REGISTRATION SYSTEM

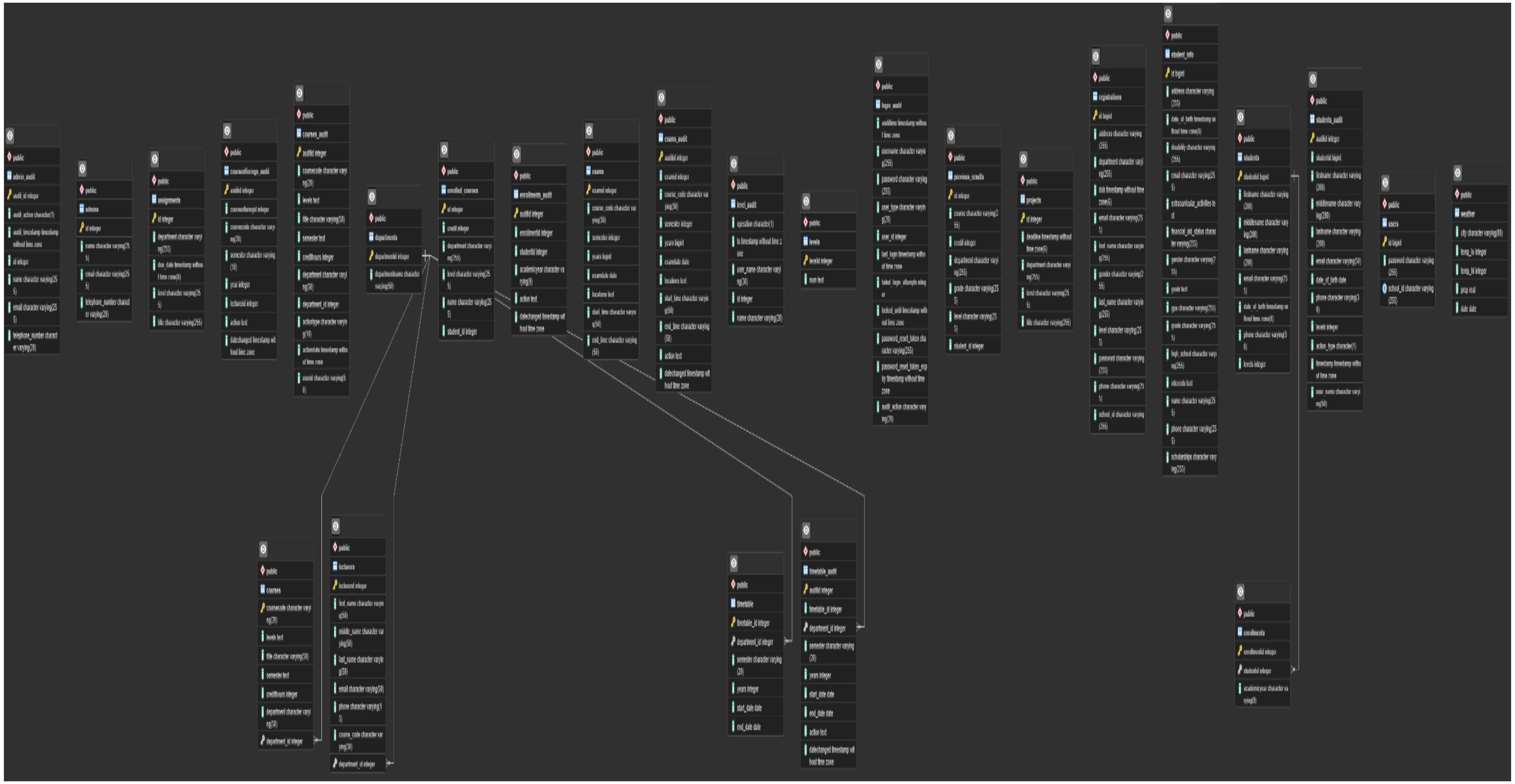


Login and Registration System Use Case Diagram
User
Admin
User Information Processing
Information Validation
Register
Username and Password Details
Account Confirmation
Registration Confirmation
Login
Username and Password Details
Details Validation
User Account Update
System Accesed
<<include>>

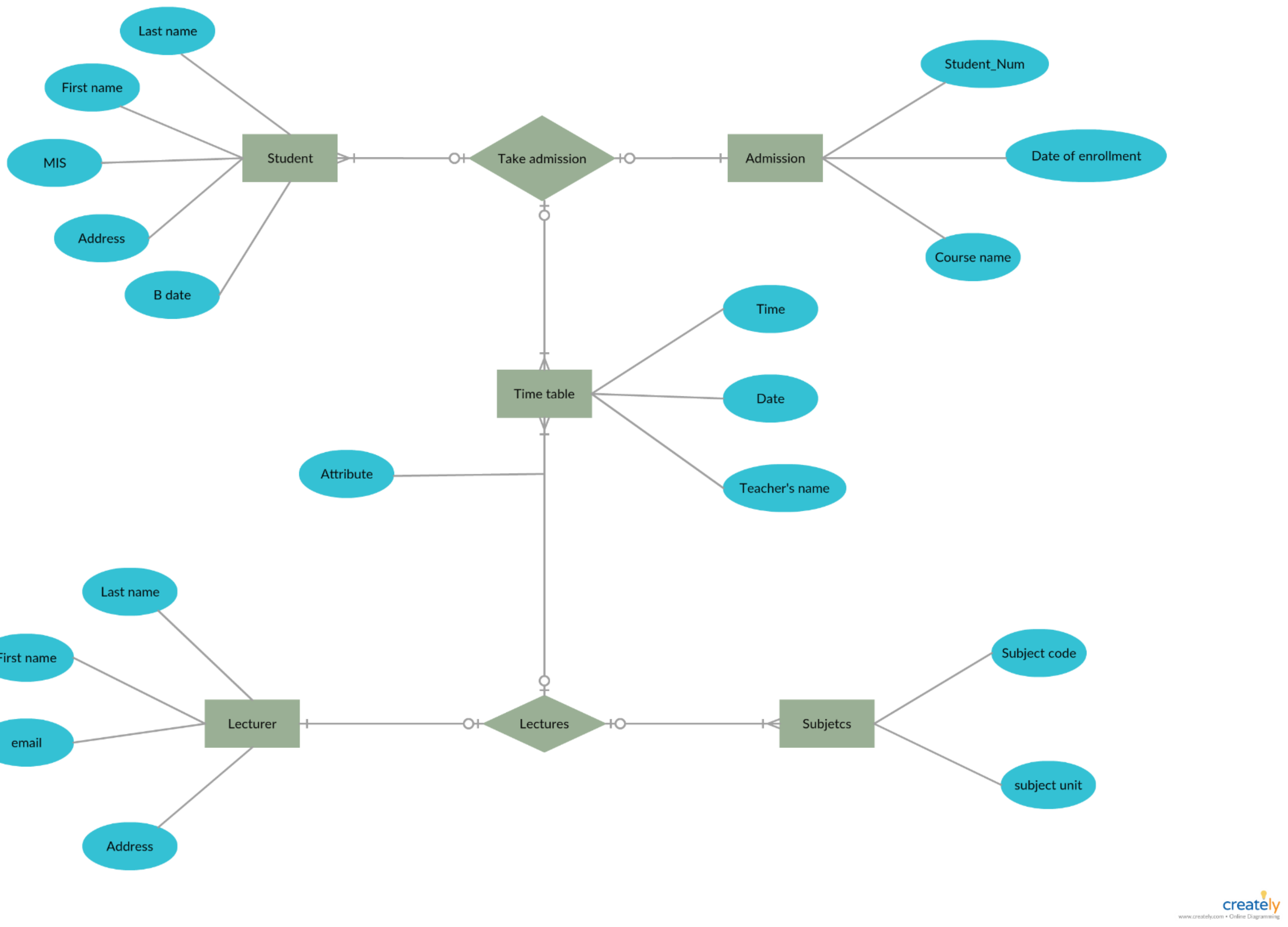
USE CASE DIAGRAM

Database Schema:

The database schema represents the structure of the PostgreSQL database used in the application. It includes tables and their relationships, representing how data is organized and stored.



Entity Relationship Diagrams:



❖Software System Architecture

The software system architecture provides a high-level overview of the system's design, outlining the components, their interactions, and how they work together to achieve the application's functionality. In this project, we adopt a layered architecture pattern to ensure modularity, scalability, and maintainability.

1. **Presentation Layer:**

The Presentation Layer handles user interactions and provides the user interface for both the web and mobile applications. It consists of two components:

a. **React Frontend:**

- Developed using React, this component offers a responsive and interactive user interface.
- It communicates with the backend through RESTful API calls to fetch and update data.
- Users can register, log in, enter and view their information, and access the dashboard through the UI.

b. **Flutter Mobile App:**

- Developed using Flutter, this component provides a native mobile app experience for users.
- It communicates with the backend through RESTful API calls to perform actions and retrieve data.
- Users can register, log in, enter and view their information, and access the dashboard through the app.

2. **Application Layer:**

The Application Layer acts as an intermediary between the Presentation Layer and the Data Layer. It consists of two components:

a. **Spring Boot Backend/Web Service:**

- Developed using Spring Boot, this component handles business logic and data processing.
- It exposes RESTful APIs that the frontend and mobile app use to interact with the system.
- It communicates with the Data Layer to retrieve and store data in the database.

3. **Data Layer:**

The Data Layer is responsible for data storage and retrieval. It includes:

a. **PostgreSQL Database:**
   - PostgreSQL is used as the relational database management system.
   - It stores student information, including their name, email, password, level, and department.
   - The database is accessed and manipulated through CRUD (Create, Read, Update, Delete) operations.

4. **Interaction Flow:**

The interaction flow in the system is as follows:

1. **User Registration:**
   - Users can register using the web application or the mobile app.

- The registration details are sent to the backend, where a new student entity is created and stored in the database.

2. **User Login:**

- Registered users can log in using their email and password through the web application or the mobile app.
- The login request is processed by the backend, which validates the credentials against the database.
- If the login is successful, the user is granted access to the system.

3. **Student Information Entry:**

- Authenticated users can enter and view their information via the web application or the mobile app.
- The data entered is sent to the backend, which updates the corresponding student entity in the database.

4. **Dashboard Access:**

- After logging in, users can access their personalized dashboard through the web application or the mobile app.
- The dashboard presents relevant information based on the user's profile and interactions within the system.

5. **Benefits of the Software System Architecture:**

a. Modularity: The layered architecture allows for clear separation of concerns, making the codebase easier to maintain and update.

b. Scalability: The system can handle an increasing number of users and data without major architectural changes.

c. Maintainability: The separation of components facilitates easier bug fixes, updates, and enhancements.

d. Reusability: The frontend and mobile app components can be used interchangeably with different backend systems if needed.

The software system architecture presented above demonstrates a well-organized and scalable design for the "Database Driven Web and Mobile Application." The layered approach ensures effective communication between components, providing users with a seamless and responsive experience while maintaining a high degree of flexibility for future improvements and expansions.

❖Design and Implementation

The design and implementation section outlines the technical details of the "Database Driven Web and Mobile Application" project. It covers the design choices, technologies used, and how each component of the system is implemented.

1. **Frontend Implementation**: **React**

a. **User Registration**:

- The user registration page allows users to enter their details, such as name, email, password, level, and department.
- React components handle form validation to ensure data integrity before submitting the registration request.
- When the user clicks the "Register" button, a POST request is sent to the backend API ("/api/students/register") to create a new student entity.

b. **User Login**:

- The login page includes input fields for email and password.

- React components validate the login credentials and display appropriate error messages if needed.

- Upon successful login, the user is redirected to the dashboard.

c. **Student Information Entry**:

- Users can update their information via the "StudentInfoEntry" component.

- The component fetches the existing user information from the backend API ("/api/students/{id}") and populates the form fields.

- When the user submits the updated data, a PUT request is sent to the backend API ("/api/students/{id}") to update the student entity.

d. **Dashboard**:

- The dashboard page provides a personalized view for each user, displaying their information and relevant data.

- Data is retrieved from the backend API ("/api/students/{id}") and displayed using React components.

2. **Backend Implementation: Spring Boot**

a. **Student Controller:**

- The "StudentController" class handles incoming HTTP requests related to student entities.
- It communicates with the "StudentRepository" to perform CRUD operations on the PostgreSQL database.

b. **StudentRepository:**

- The "StudentRepository" interfaces with the PostgreSQL database using Spring Data JPA.
- It provides methods for retrieving, saving, updating, and deleting student entities.

c. **User Registration:**

- When a registration request is received at "/api/students/register," the backend creates a new student entity in the database.

d. **User Login:**

- The login request at "/api/students/login" is processed, and the backend verifies the provided email and password.
- If the credentials are valid, the student entity is returned with a 200 status code; otherwise, an error with a 401 status code is returned.

e. **Student Information Update:**

- The backend handles student information updates when a PUT request is received at "/api/students/{id}".
- It updates the corresponding student entity in the database with the new information.

f. **Dashboard Data Retrieval:**

- The dashboard data is fetched from the backend using a GET request at "/api/students/{id}" and displayed in the frontend.

3. **Mobile App Implementation: Flutter**

a. **User Registration and Login:**

- The mobile app contains screens for user registration and login.
- Flutter widgets validate the input data before sending registration and login requests to the backend.

b. **Student Information Entry:**

- Users can update their information using the "StudentInfoEntryPage" in the mobile app.
- The app fetches the existing user information from the backend and pre-populates the form fields.

c. **Dashboard:**

- The dashboard page presents personalized information fetched from the backend for each user.

4. **Database Implementation: PostgreSQL**

a. **Student Table:**

- The PostgreSQL database includes a "Student" table to store student entities.

- The table has columns for ID, Name, Email, Password, Level, and Department.

b. **Data Storage:**

- When a new student registers, a new row is added to the "Student" table with the provided details.
- Student information updates through the frontend or mobile app result in corresponding updates in the database.

5. **Interactions and Flow:**

- Users can register, log in, update their information, and access their dashboard through both the web application and mobile app.
- The frontend components interact with the backend through RESTful API calls to perform various actions and retrieve data from the database.

❖Testing and Evaluation

Testing is a critical phase in the software development process, ensuring that the "Database Driven Web and Mobile Application" functions as intended and meets user requirements. In this section, we outline the testing strategies employed and evaluate the application's performance, usability, and reliability.

1. **Testing Strategies:**

a. **Unit Testing:**

- Unit tests are conducted on individual components, such as React components, Spring Boot controllers, and Flutter widgets.
- Jest and React Testing Library are used for testing React components.
- JUnit and Mockito are used for testing Spring Boot controllers and services.
- Flutter's built-in testing framework is utilized for testing the mobile app components.

b. **Integration Testing:**

- Integration tests validate interactions between different components within the system.
- We conduct integration tests to ensure that the frontend communicates effectively with the backend through API calls.

c. **End-to-End Testing:**

- End-to-End tests assess the entire application flow, from user interactions to data storage and retrieval.
- Cypress is utilized for conducting end-to-end tests on the web application.
- Flutter's integration_test package is used for end-to-end testing on the mobile app.

d. **User Acceptance Testing (UAT):**

- UAT involves real users testing the application in a simulated production environment.
- Feedback from users is collected to evaluate the application's usability and identify any issues that might have been missed during other testing phases.

2. **Performance Evaluation:**

a. **Scalability:**

- The application's performance is assessed by subjecting it to increasing user loads.
- Load testing tools, such as Apache JMeter, are used to measure response times and resource utilization under different loads.
- The system is monitored for potential bottlenecks and performance issues.

b. **Latency and Response Time:**

- We measure the time it takes for the application to respond to user requests.
- Network latency and backend processing times are evaluated to identify areas for optimization.

3. **Usability Evaluation:**

a. **User Interface (UI) Testing:**

- The application's user interface is evaluated for consistency, responsiveness, and adherence to design guidelines.
- User feedback and usability heuristics are used to identify areas for improvement.

b. **User Experience (UX) Testing:**

- User experience is evaluated by analyzing user interactions and feedback.
- We assess how easily users can accomplish tasks, navigate through the application, and access required information.

4. **Reliability Evaluation:**

a. **Stress Testing:**

- The application is subjected to stress tests to determine its stability under extreme conditions.

- Stress tests simulate scenarios like high user traffic or database overload.

b**. Error Handling:**

- Error scenarios, such as invalid inputs or server downtime, are tested to ensure proper error handling and graceful degradation.

5. **Security Evaluation:**

a. **Data Security:**

- Security measures, such as encryption, are employed to protect sensitive data like passwords and user information.

- Security testing tools are used to detect vulnerabilities and potential security threats.

6. **Evaluation Conclusion:**

The testing and evaluation phase ensures that the "Database Driven Web and Mobile Application" is robust, reliable, and user-friendly. Rigorous testing techniques, including unit testing, integration testing, end-to-end testing, and user acceptance testing, verify the application's functionality and user experience. Performance and scalability evaluations provide insights into the application's response times and resource usage. Additionally, security measures are applied to safeguard user data and protect against potential threats.

The application's successful completion of testing and evaluation phases validates its readiness for deployment and use in a production environment, meeting the needs and expectations of its intended users.

❖Software Project Management

Software project management is a crucial aspect of the "Database Driven Web and Mobile Application" development. It involves planning, organizing, and controlling the project to ensure its successful completion within the defined scope, timeline, and budget. This section outlines the key project management practices adopted throughout the project lifecycle.

1. **Project Initiation:**

a. **Project Scope and Objectives:**

- The project scope is clearly defined, outlining the features, functionalities, and target platforms (web and mobile).
- Specific project objectives, such as user registration, login, data entry, and dashboard access, are established.

b. **Requirements Gathering:**

- Comprehensive requirements gathering is conducted through discussions with stakeholders and end-users.

- A detailed requirement specification document is prepared, capturing the application's functional and non-functional requirements.

2. **Project Planning:**

a. **Work Breakdown Structure (WBS):**

- The project is broken down into smaller tasks and sub-tasks using a Work Breakdown Structure (WBS).

- Each task is assigned to specific team members based on their expertise.

b. **Timeline and Milestones:**

- A detailed project timeline is created, including milestones for major deliverables and deadlines.

- Agile development principles are adopted to ensure iterative progress and flexibility.

c. **Resource Allocation:**

   - Project resources, including developers, designers, and testers, are allocated based on their availability and skills.

d. **Risk Assessment:**

   - Potential risks are identified, categorized, and analyzed in a risk assessment plan.

   - Mitigation strategies are devised to minimize the impact of identified risks.

3. **Project Execution:**

a. **Agile Development:**

   - Agile methodologies, such as Scrum or Kanban, are employed to facilitate continuous collaboration and adaptability.

   - Regular sprints and daily stand-up meetings are conducted to review progress and address any roadblocks.

b. **Version Control:**

- Version control systems, such as Git, are used to manage code changes and facilitate collaboration among team members.

4. **Project Monitoring and Control:**

a. **Progress Tracking:**

- Progress is continuously monitored against the planned timeline and milestones.
- Project management tools, such as JIRA or Trello, are used to track task status and identify bottlenecks.

b. **Quality Assurance:**

- Regular testing and code reviews are conducted to ensure the application's quality and adherence to coding standards.

5. **Project Communication:**

a. **Regular Updates:**

- Stakeholders are provided with regular updates on the project's progress and any changes to the scope or timeline.

b. **Issue Management:**

- An issue tracking system is used to log and manage project-related issues and bug reports.

6. **Project Closure:**

a. **User Acceptance Testing (UAT):**

- UAT is conducted with real users to validate the application's readiness for deployment.

b. **Documentation:**

- Comprehensive documentation is prepared, including user manuals, technical guides, and architectural documentation.

c. **Deployment and Handover:**

- The application is deployed to the production environment following a smooth transition plan.
- The project is formally handed over to the maintenance team for ongoing support and updates.

❖Conclusion

The "Database Driven Web and Mobile Application" is a sophisticated and user-friendly platform designed to efficiently manage student data across web and mobile platforms. Through a well-thought-out design, meticulous implementation, comprehensive testing, and effective project management, the application offers a seamless experience for students to register, log in, update their information, and access personalized dashboards.

The success of this project can be attributed to the adoption of cutting-edge technologies and development practices. React, Spring Boot, Flutter, and PostgreSQL were judiciously chosen to build a scalable, modular, and secure system. The use of RESTful APIs ensures smooth communication between the frontend and backend components.

Throughout the development lifecycle, rigorous testing strategies were employed to validate the application's functionality, performance, and security. Unit testing, integration testing, end-to-end testing, and user acceptance testing collectively ensured a robust and reliable product.

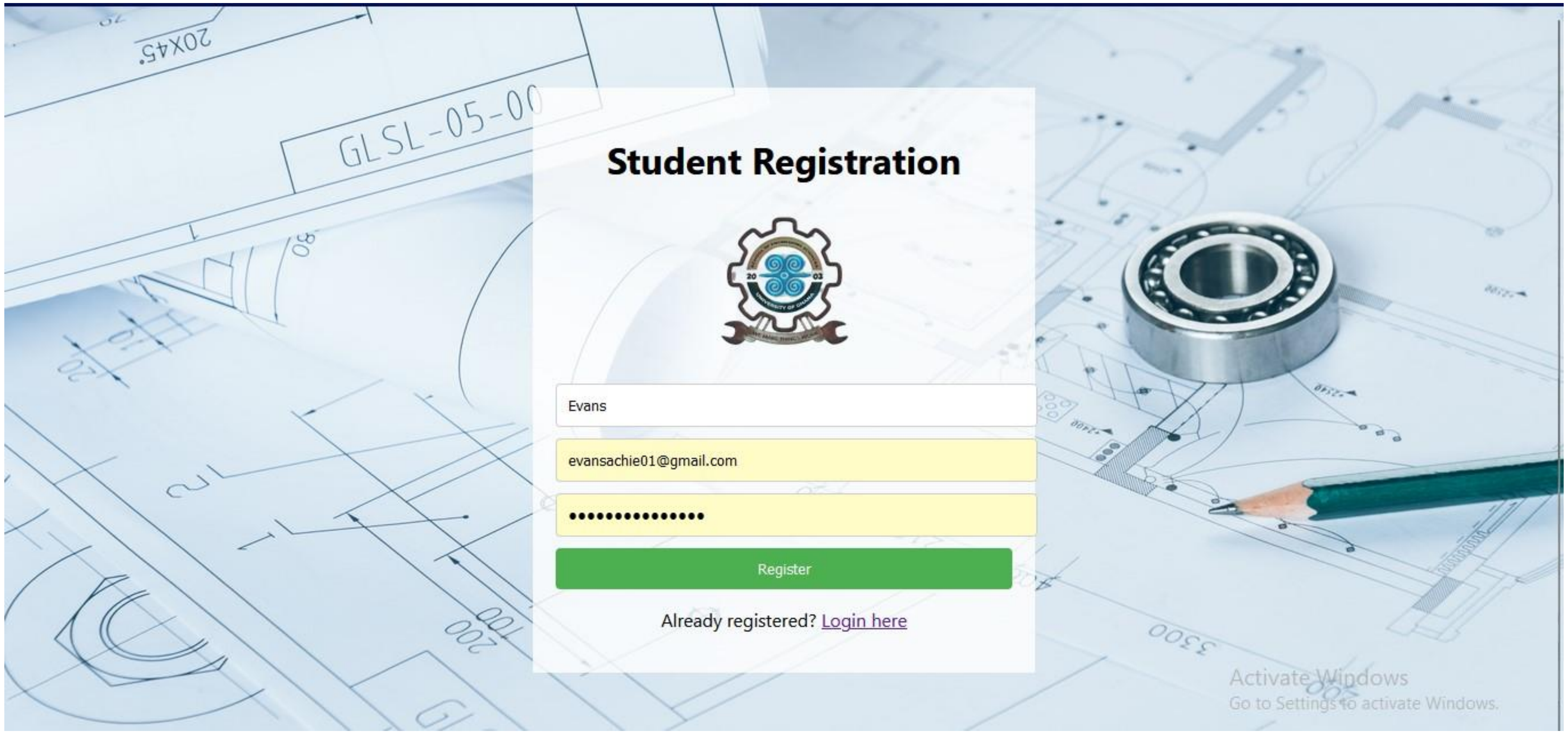
As a result of this comprehensive documentation, users and future developers will find it easy to understand and maintain the application. Detailed technical guides and architectural documentation provide valuable insights into the system's inner workings.

In conclusion, the "Database Driven Web and Mobile Application" represents a successful fusion of cutting-edge technology, rigorous testing, user-centered design, and effective project management. It serves as a powerful tool for managing student data and demonstrates our commitment to delivering high-quality software solutions that meet the needs and expectations of users. The success of this project sets the stage for future innovations and developments, solidifying our position as a reliable and competent software development team.

❖Appendices

Appendix A: Screenshots of React Web Application; Live Website can also be found Here: https://ses-react-project.vercel.app/

Registration Page



Student Registration

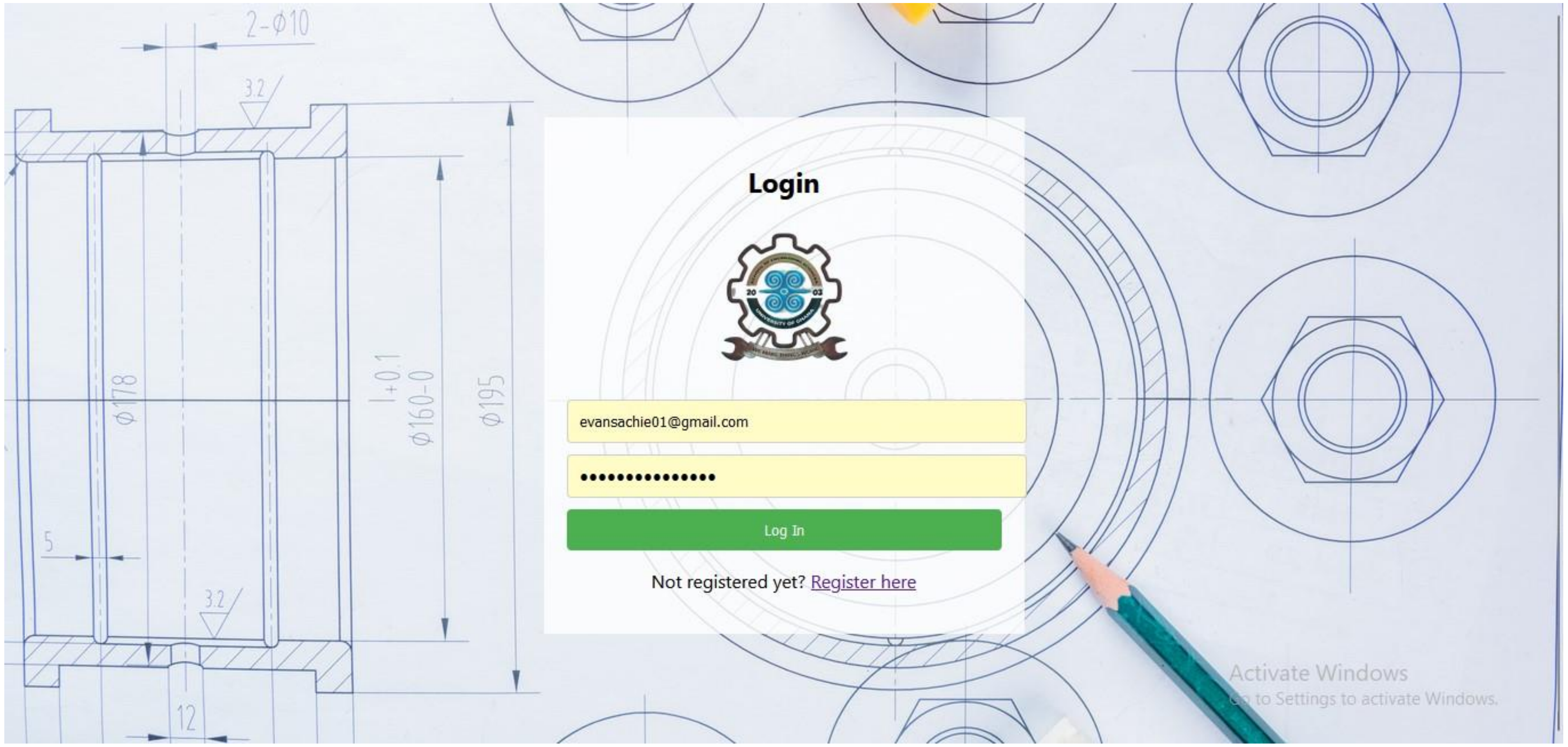
Evans

evansachie01@gmail.com

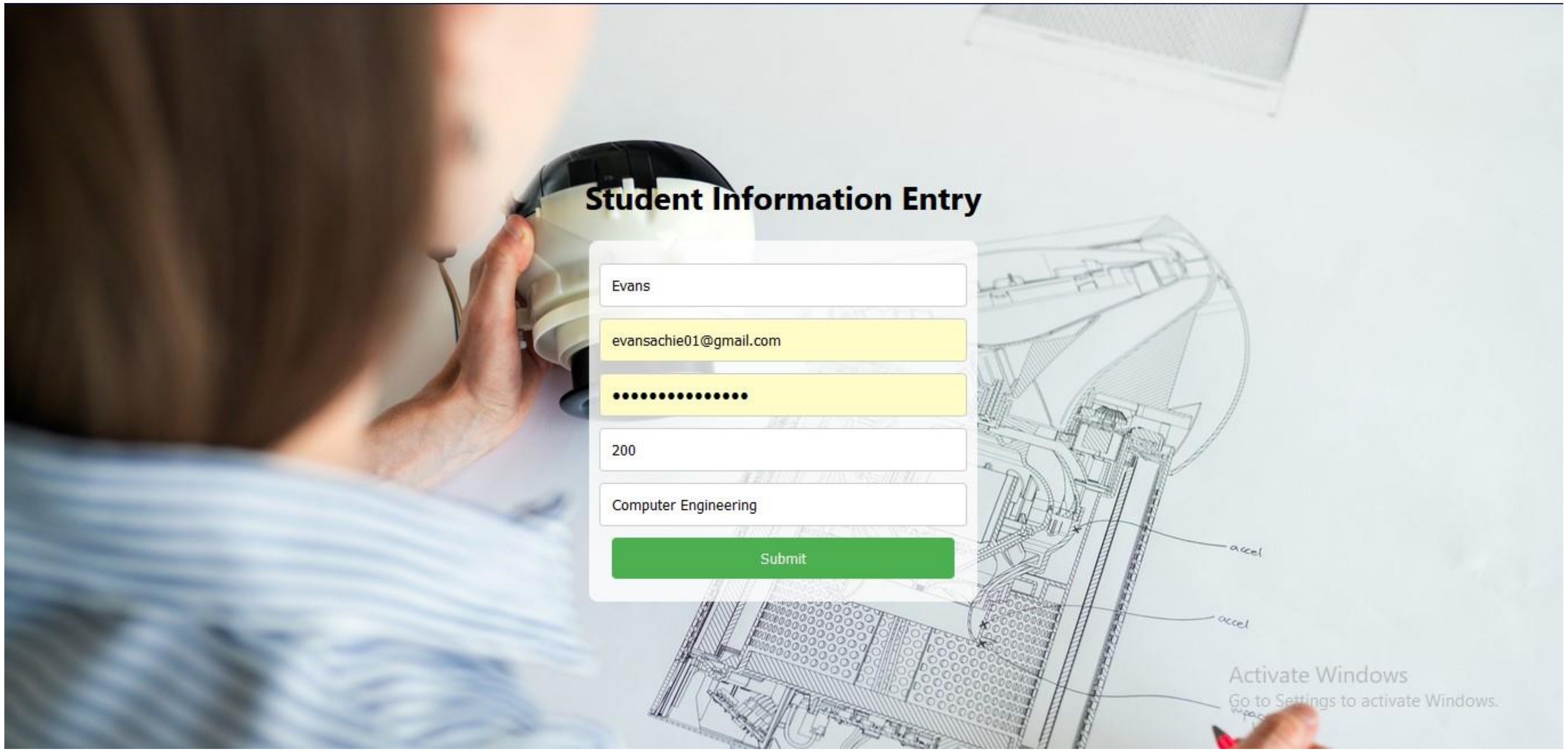
•••••••••••••••

Already registered? Login here

Login Page


Login
evansachie01@gmail.com
•••••••••••••••
Log In
Not registered yet? Register here
Activate Windows
Go to Settings to activate Windows.

Info Entry Page



Student Information Entry

Evans

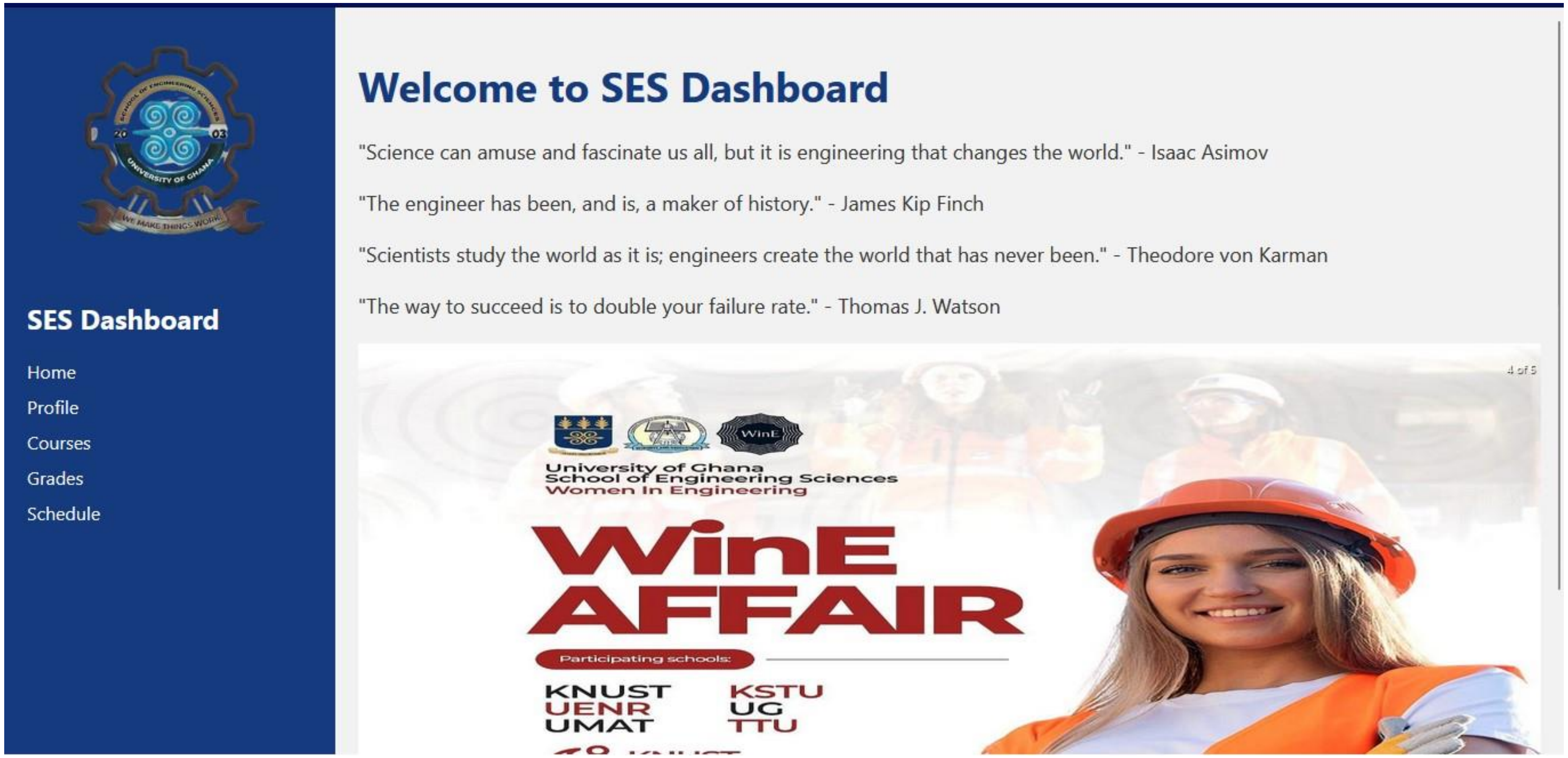
evansachie01@gmail.com

•••••••••••••••

200

Computer Engineering

Dashboard Page



Welcome to SES Dashboard

"Science can amuse and fascinate us all, but it is engineering that changes the world." - Isaac Asimov

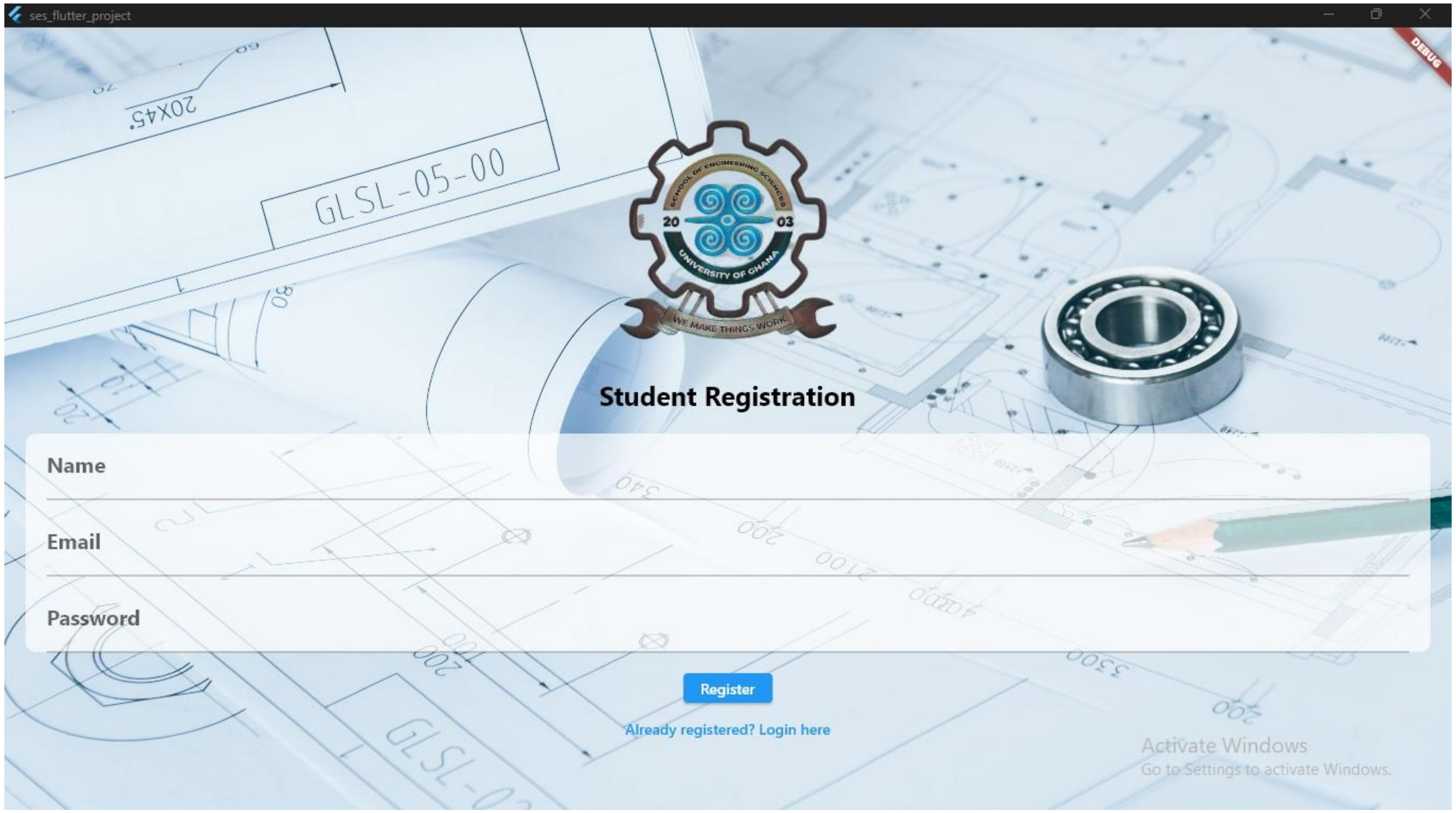
"The engineer has been, and is, a maker of history." - James Kip Finch

"Scientists study the world as it is; engineers create the world that has never been." - Theodore von Karman

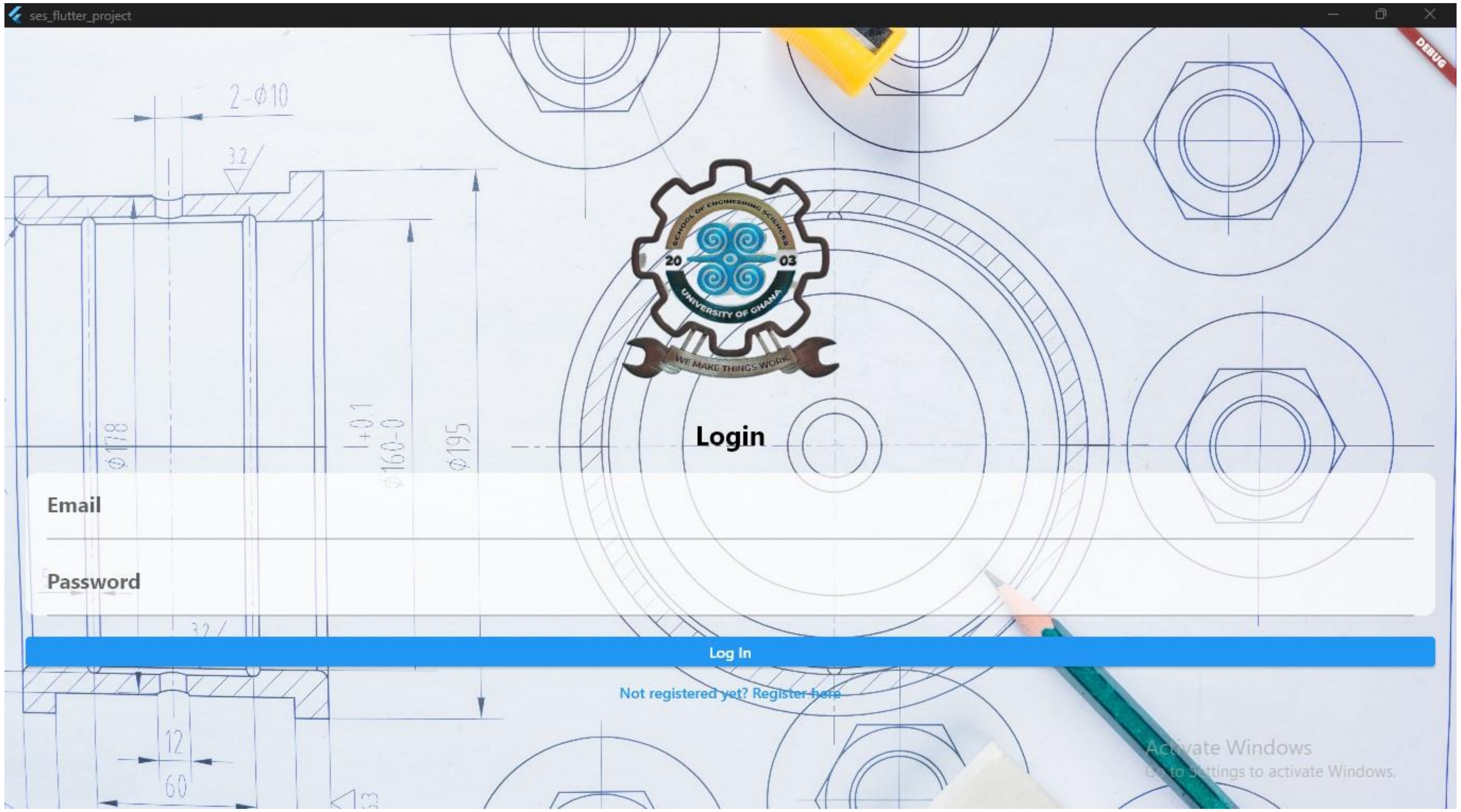
"The way to succeed is to double your failure rate." - Thomas J. Watson

Appendix B: Screenshots for Flutter Mobile Application

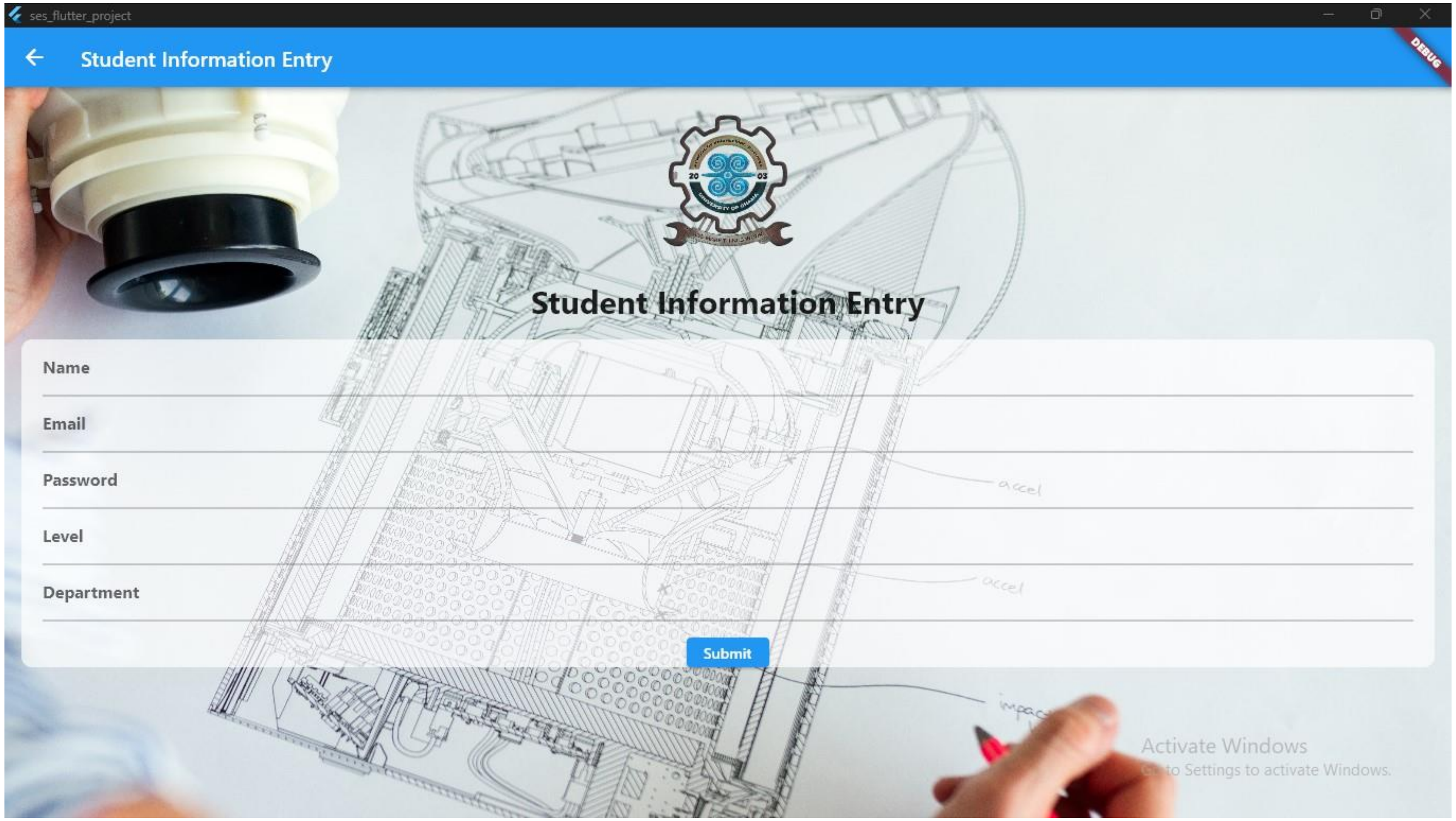
Registration Page



Login Page


ses_flutter_project
DEBUG
SCHOOL OF ENGINEERING SCIENCES
20
03
UNIVERSITY OF GHANA
WE MAKE THINGS WORK
Login
Email
Password
Log In
Not registered yet? Register here
Activate Windows
Go to Settings to activate Windows.

Info Entry Page



ses_flutter_project

Student Information Entry

Student Information Entry

Name

Email

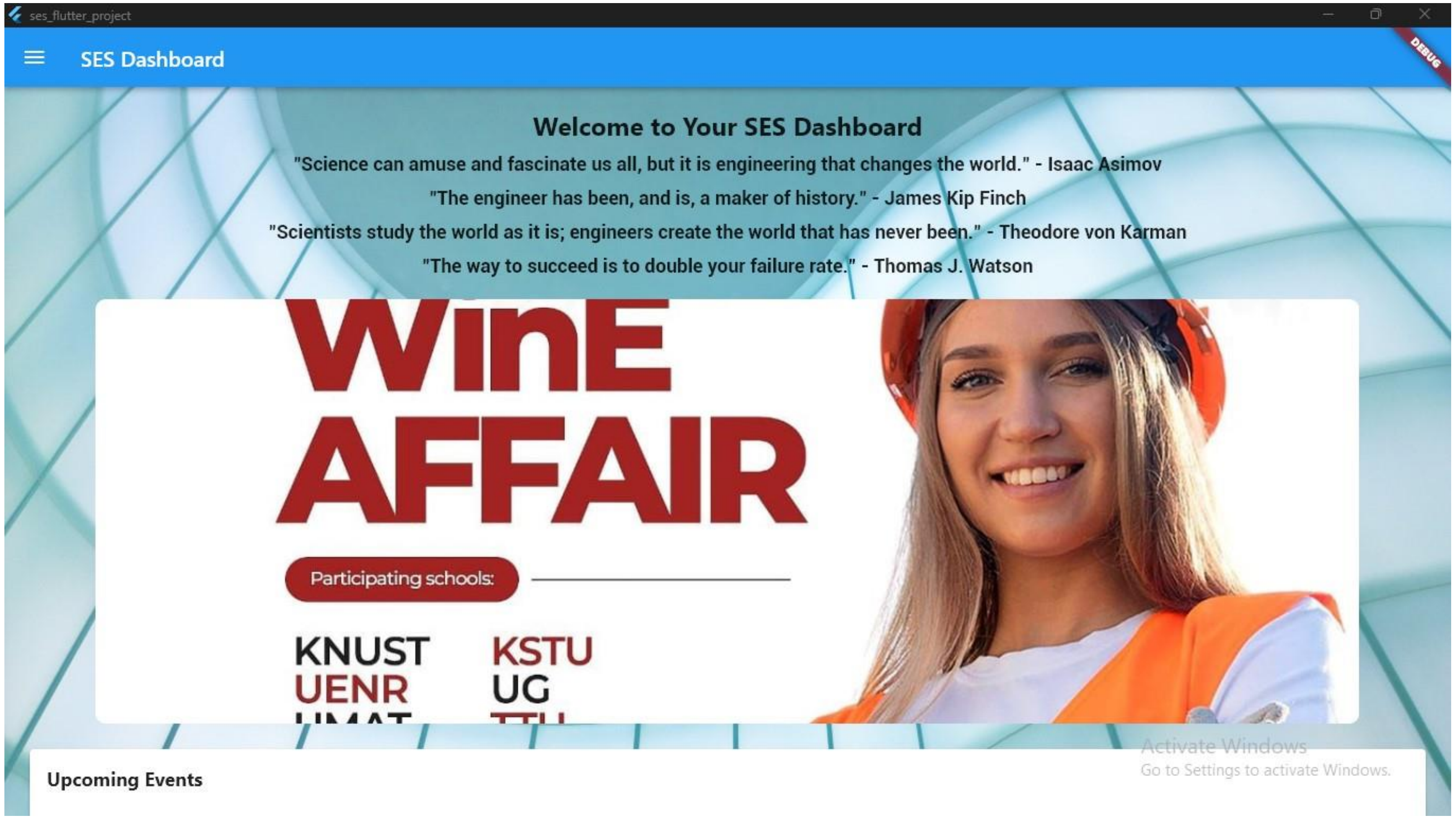
Password

Level

Department

Submit

Dashboard Page



Welcome to Your SES Dashboard

"Science can amuse and fascinate us all, but it is engineering that changes the world." - Isaac Asimov

"The engineer has been, and is, a maker of history." - James Kip Finch

"Scientists study the world as it is; engineers create the world that has never been." - Theodore von Karman

"The way to succeed is to double your failure rate." - Thomas J. Watson

Upcoming Events

❖References

1. React. (n.d.). Retrieved from https://reactjs.org/

2. React Router. (n.d.). Retrieved from https://reactrouter.com/

3. GitHub. (n.d.). GitHub Repository for React Web application. Retrieved from https://github.com/AWESOME04/SES-React-Project

4. Flutter Documentation - Official documentation for Flutter framework. Website: https://flutter.dev/docs

5. Unsplash - A popular platform that offers a wide range of high-resolution, royalty-free images contributed by photographers worldwide. Website: https://unsplash.com/

6. Dart Documentation - The official documentation for the Dart programming language, providing comprehensive guides, tutorials, and API references. Website: https://dart.dev/

7. GitHub. (n.d.). GitHub Repository for Flutter Mobile App. Retrieved from https://github.com/AWESOME04/SES-Mobile-Application

8. GitHub. (n.d.). GitHub Repository for Database Driven Web and Mobile Application. Retrieved from https://github.com/AWESOME04/Database-Driven-Web-and-Mobile-Application

8. GitHub. (n.d.). GitHub Repository for For PostgreSQL Database. Retrieved from https://github.com/AWESOME04/GROUP2_SES_Database_Project-3

9. Spring Boot Official Guide: Getting Started - Building a RESTful Web Service: https://spring.io/guides/gs/spring-boot/

10. Database Driven Web and Mobile Application – React Web Application Live Website: https://ses-react-project.vercel.app/