College of Informatics and Computing Sciences

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus

COLLEGE OF INFORMATICS AND COMPUTING SCIENCES
BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

FINAL PROJECT REPORT
CS 211: Object-Oriented Programming
1ST Semester, A.Y 2023-2024

TESTING AND ADMISSION MANAGEMENT SYSTEM

Submitted by
De Leon, Gecelyn T.
Hernandez, Jhavie G.
Pentinio Mhark Anthony O.
Robles, Iris M.

Instructor

Ms. Aileen V. Suarez

DECEMBER 2023

College of Informatics and Computing Sciences

OVERVIEW

The **Testing and Admission Management System** is a console application meticulously designed to simplify and enhance the admission process for educational institutions. With a user-centric approach, the system offers a range of features catering to the specific needs of administrators and applicants alike. From multi-level authentication ensuring data security to real-time updates facilitating transparent communication, the system optimizes administrative tasks. For administrators, it streamlines application processing and document verification, providing a dynamic dashboard for efficient management. Simultaneously, applicants benefit from an intuitive application process, immediate feedback, and a centralized hub for tracking application status. In essence, the Testing and Admission Management System not only addresses the complexities of admissions but also fosters a user-friendly and secure environment for both educational institutions and aspiring students.

OBJECTIVE

- **To streamline the admission process**: Simplify workflows, automate tasks, and minimize manual interventions to expedite the admission procedure for both educational institutions and applicants.
- **To ensure transparent communication**: Provide real-time status updates and clear notifications, fostering an open and informed dialogue between administrators and applicants throughout the admission cycle.
- **To minimize administrative costs**: Implement efficient, automated processes to reduce paperwork and administrative workloads, resulting in potential financial savings for educational institutions.
- **To optimize user experience**: Enhance the user interface, functionality, and performance of the Admission Management System to create a seamless and user-friendly experience for both administrators and applicants.

METHODOLOGY

In our software development journey, we have meticulously applied the pillars of Object-Oriented Programming (OOP) to create a robust and user-friendly registration system. Let's embark on a brief exploration of the methodology that governs the intricate interplay of classes and components within our codebase. These principles serve as the cornerstone of our project, fostering a well-organized and easily maintainable codebase .

The **AccountRegister** class encapsulates the fields required for registration: last name, first name, email address, password, and verify password. User authentication is a key

College of Informatics and Computing Sciences

component, with methods like **authenticateUser** and **authenticateAdmin** verifying login credentials against the database.

The **Admin** class models an administrator, storing information such as admin ID, username, and password. On the other hand, the Applicant class represents an applicant, storing information such as the applicant's ID, last name, first name, email, and password. Constructors provide flexibility for creating instances with or without an applicant ID. Getter and setter methods ensure controlled access to the applicant's and admin information.

The **LogInControlle**r includes FXML elements such as a Choice Box for selecting the account type (applicant or admin), TextFields for entering the username and password, and buttons for login and registration. During initialization, the controller sets up the initial state, populating the account type ChoiceBox and establishing a listener for changes in the selected account type.

The core functionality lies in the **handleLogin** method, triggered when the login button is pressed. This method retrieves the entered username, password, and account type, and attempts to authenticate the user or admin using a database query. Depending on the account type, it checks the entered credentials against the database, and if authentication is successful, it creates a new controller (either **UserController** or **AdminController**) and displays the corresponding window.

The code handles various scenarios, including invalid account type selection, authentication failure, and unexpected errors during the process. It provides user feedback through JavaFX Alert dialogs, displaying error messages when necessary. Additionally, the **handleRegistration** method is responsible for navigating to the registration window when the registration button is pressed.

Dashboard (Admin): This class abstracts the management of the user dashboard's display, including the integration of a Text field for showing the applicant's name. It abstracts the complexities of dashboard management and collaborates with the **UserMenuController,** indicating a level of abstraction in handling user interface components.

Home (Applicant): This class abstracts the control of the main user interface, specifically the central BorderPane that displays different views based on user interactions. The

initialization process involves setting up a listener on the selected menu item, abstracting away the details of the UI handling.

The **applicantController** class serves as a controller for a user interface, featuring a **TableView** displaying applicant details. The code encompasses functionalities such as searching, reviewing, and deleting applicants. Notable components include event handlers for button actions, database interactions through the appQuery class, and an observable list for handling applicant data.

The **ApplicantSchoolInfo** class represents detailed information about an applicant's school history. It includes attributes like ID, high school name, address, completion year, senior high school name, address, completion year, and strand. The class follows a standard structure with constructors for instantiation and getters and setters for accessing and modifying information.

The **ApplicantParentInfo** class provides information about an applicant's parents, including their last name, first name, middle name, and contact number. The **ApplicantInfo** class captures detailed information about an applicant, such as their name, age, birthday, contact details, and address. Both classes follow a similar structure with constructors, getters, and setters, adhering to encapsulation principles for data access and modification.

The **ApplicantGradeInfoSHS** and **ApplicantGradeInfoHS** classes store information related to an applicant's grades in senior high school (SHS) and high school (HS), respectively. These classes include attributes for various subjects and semesters. Constructors initialize instances of these classes, and getters and setters facilitate data access and modification.

The **Announcement** class encapsulates the properties of an announcement and includes fields for the announcement's ID, title, content, an administrative table ID, and the announcement date. The class provides multiple constructors to instantiate objects with different sets of parameters, allowing flexibility in creating announcements. Constructors cover cases where the announcement ID is provided or generated, and they also account for variations in the availability of administrative table IDs and announcement dates. The class features getter and setter methods for each field, facilitating access and modification of the announcement's attributes. Overall, this class serves as a model for managing announcement data within a Java application.

College of Informatics and Computing Sciences

The **onSignOut()** method is responsible for handling the sign-out functionality. It closes the current stage and shows the login window using the **ViewsFactory** in the Model class.

SOURCE CODE

https://drive.google.com/drive/folders/1uF1TeIFWdrYsU8IfD7CaNKY2Ex10JjvC