JavaFX University Management System

Introduction

The JavaFX University Management System is a comprehensive application designed to streamline various administrative and student-related processes within an academic institution. Developed using JavaFX, this application leverages the Model-View-Controller (MVC) architecture to ensure a responsive, intuitive user interface and a robust backend system. The project's primary objective is to provide a seamless digital platform for managing student and administrative activities, enhancing the efficiency of university operations.

The system caters to two main user roles: Students and Administrators (Admins), each with distinct functionalities. It integrates a secure login mechanism, ensuring that each user accesses only the functionalities pertinent to their role. The system's scope includes managing student applications for various academic requests and administrative tasks like managing student records and overseeing application statuses.

Application Description

The JavaFX University Management System offers a range of functionalities tailored to the specific needs of Students and Admins:

1. Admin Functionalities:
    - Login: Admins have exclusive access to an administrative dashboard after a secure login process.
    - Manage Students: Admins can add new students to the system or delete existing student records. This feature includes viewing a list of all students and performing actions like adding or removing students from the system.
    - Handle Applications: Admins have the capability to view, approve, or deny student applications. These applications can range from leave requests to transcript applications. Admins can update the status of each application to either 'Approved' or 'Denied'.
    - Logout: For security purposes, admins can securely log out of the dashboard.
2. Student Functionalities:
    - Login: Students can access a personalized dashboard upon logging in.
    - Apply for Services: Students have the option to apply for various services including Leave, Registration, Revaluation, Transcript requests, Additional Slot booking, and Supplementary exams.
    - View Status: Students can view the current status of all their submitted applications, keeping track of their requests and the corresponding responses.
    - Logout: Students can securely exit their dashboard.

Startup and Login Instructions

Launching the Application:

- Ensure JavaFX and the required libraries are installed on your system.
- Locate the main executable Java file in the project directory (typically named Main.java or similar).
- Run the application using a suitable Java IDE or from the command line.

Logging In:

- The application supports two types of users: Students and Admins. Each user type has its own set of functionalities and dashboard.
- To log in as an Admin, use the following credentials:
    - Username: admin
    - Password: admin
- To log in as a Student, use these sample credentials or those of any registered student:
    - Username: john
    - Password: john

Upon successful login, users will be directed to their respective dashboards where they can access the various functionalities offered by the system.

Implementation Details

JavaFX Usage and MVC Architecture

The University Management System is implemented using JavaFX, a state-of-the-art framework for developing rich client applications. JavaFX offers a wide range of functionalities, from basic UI elements to advanced graphics and media. The application leverages these capabilities to provide an interactive and responsive user interface.

The architecture of the application is based on the Model-View-Controller (MVC) pattern, which is instrumental in separating the application's concerns, thereby enhancing maintainability and scalability.

- Model: Represents the data structure and business logic. It directly manages the data, logic, and rules of the application. In this project, the models include UniversityUser and UniversityApplication, corresponding to the database tables.
- View: Responsible for displaying data (the model) to the user. It's the application's UI. JavaFX's FXML files and controllers are used to define the application's graphical user interface.
- Controller: Acts as an intermediary between the view and the model. It listens to the events triggered by the view and executes the appropriate reactions.

Database Structure and Integration

The application integrates a relational database to manage and store data. The database consists of two main tables:

UniversityUsers Table

- Stores user data, including credentials and roles.
- Fields: user_id, name, email, hashed_password, salt, role.
- The hashed_password and salt fields ensure secure password storage.

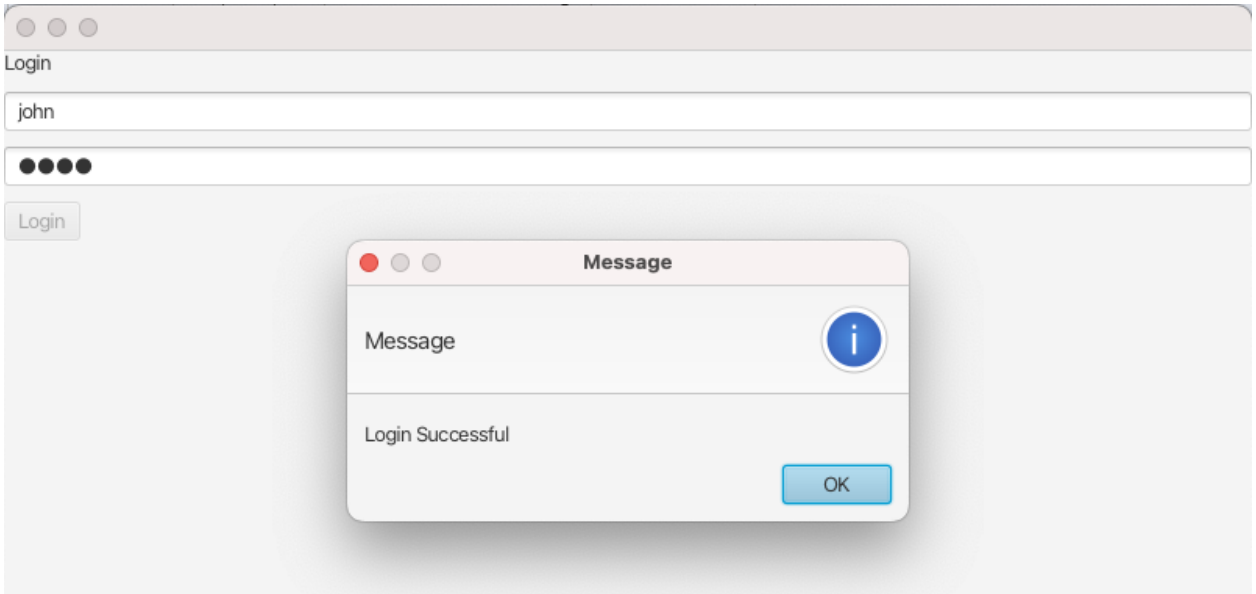
UniversityApplications Table

- Manages various applications submitted by students.
- Fields: application_id, user_id, type, status, details, application_date.
- Linked to UniversityUsers via user_id to keep track of which student submitted which application.

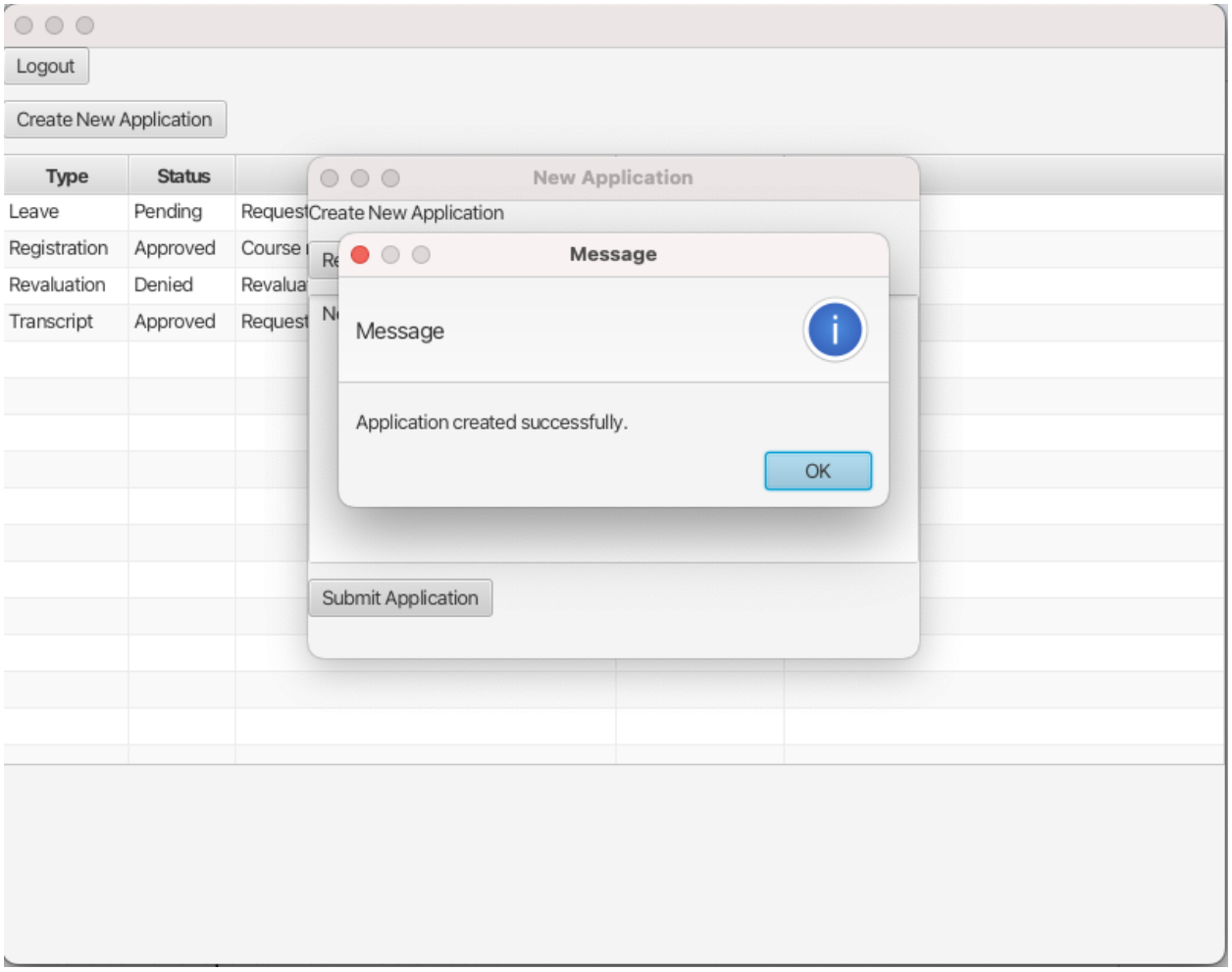
The database is integrated using JDBC, allowing for seamless communication between the JavaFX application and the database. The DAO (Data Access Object) pattern is utilized for database operations, providing a clear separation between the application logic and database access logic.

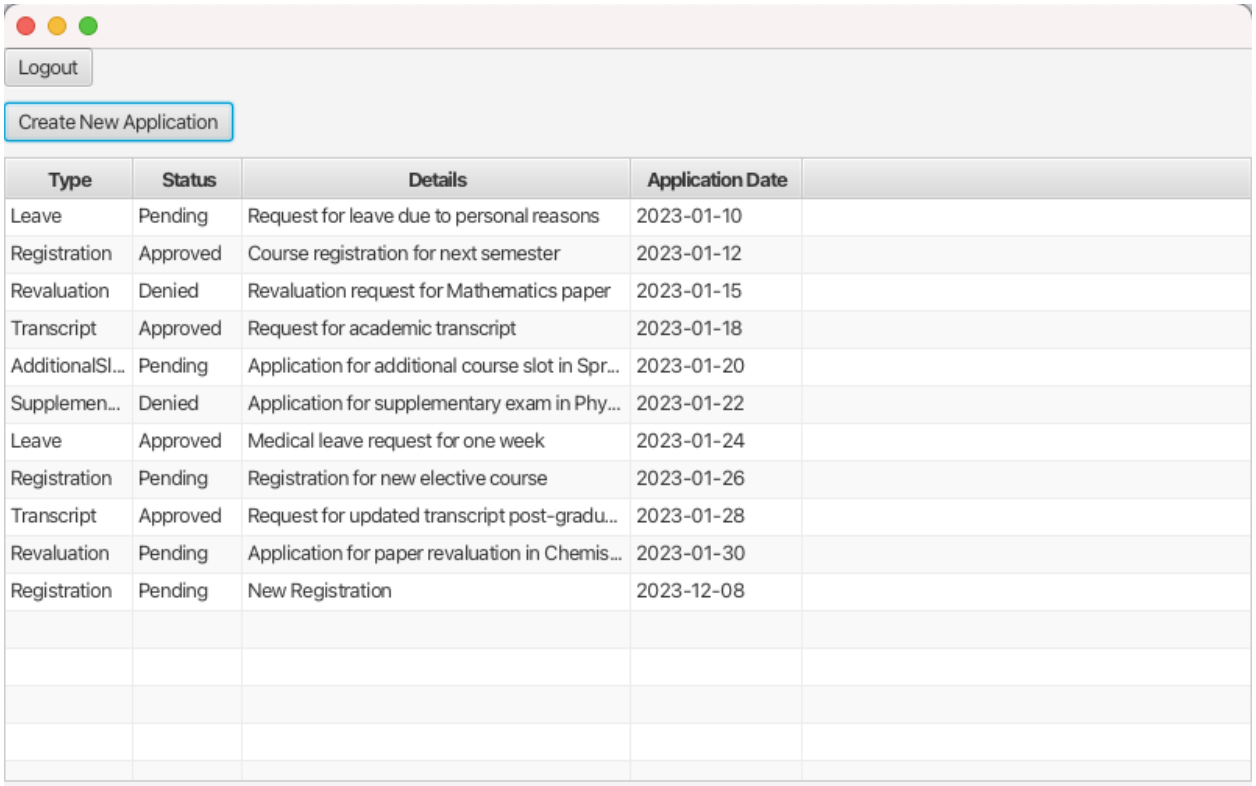
Snapshot Documentation

Successful Login for a Regular User
Description: This snapshot shows the successful login screen for a regular student user.



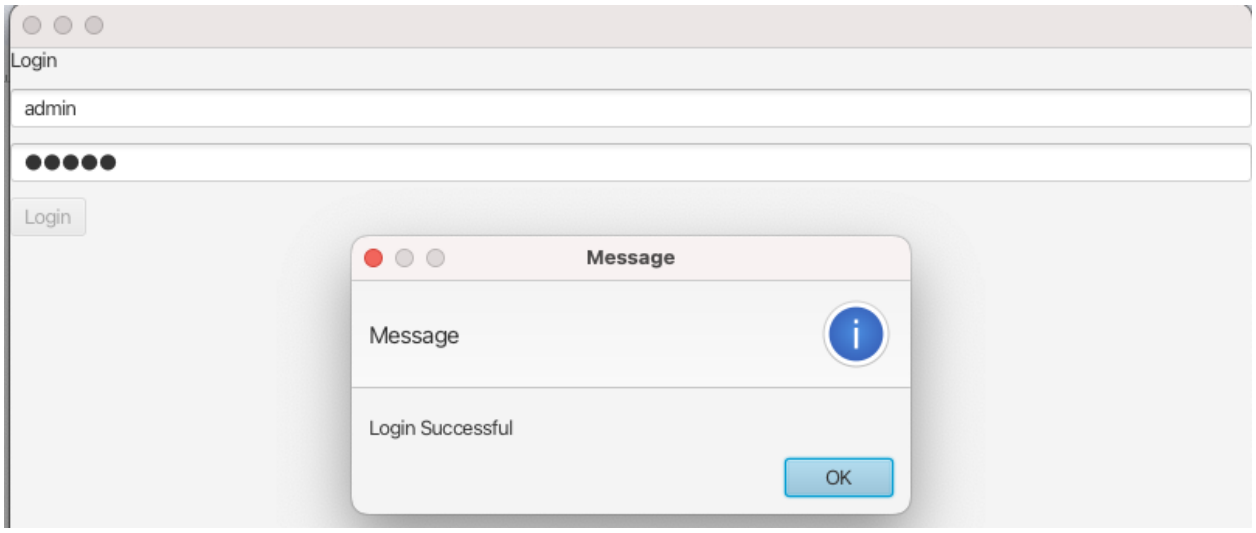
10 Records Added to the Database
Description: Display of 10 records added by a user, showcasing the application's ability to handle multiple entries.


Logout
Create New Application
Type
Status
Leave
Pending
Registration
Approved
Revaluation
Denied
Transcript
Approved
New Application
Create New Application
Message
Message
Application created successfully.
OK
Submit Application

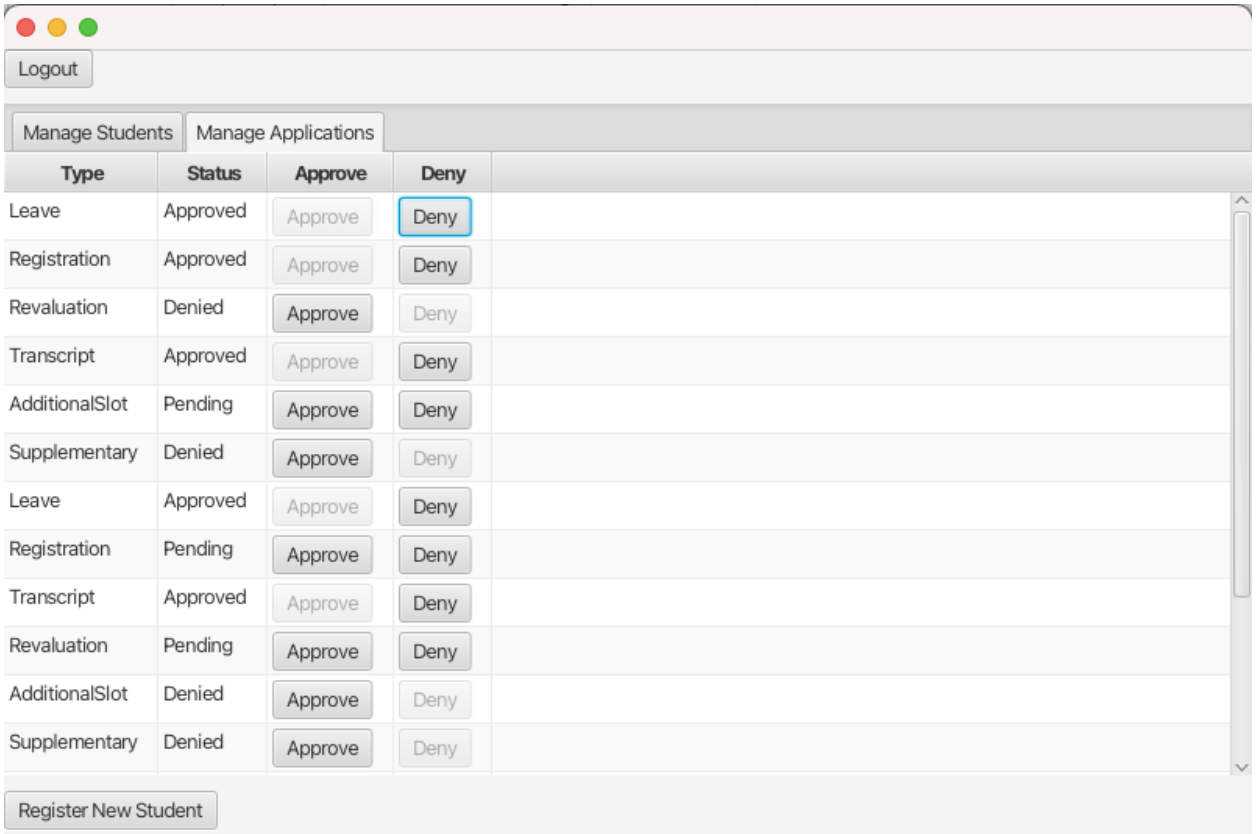


Type	Status	Details	Application Date
Leave	Pending	Request for leave due to personal reasons	2023-01-10
Registration	Approved	Course registration for next semester	2023-01-12
Revaluation	Denied	Revaluation request for Mathematics paper	2023-01-15
Transcript	Approved	Request for academic transcript	2023-01-18
AdditionalSl...	Pending	Application for additional course slot in Spr...	2023-01-20
Supplemen...	Denied	Application for supplementary exam in Phy...	2023-01-22
Leave	Approved	Medical leave request for one week	2023-01-24
Registration	Pending	Registration for new elective course	2023-01-26
Transcript	Approved	Request for updated transcript post-gradu...	2023-01-28
Revaluation	Pending	Application for paper revaluation in Chemis...	2023-01-30
Registration	Pending	New Registration	2023-12-08

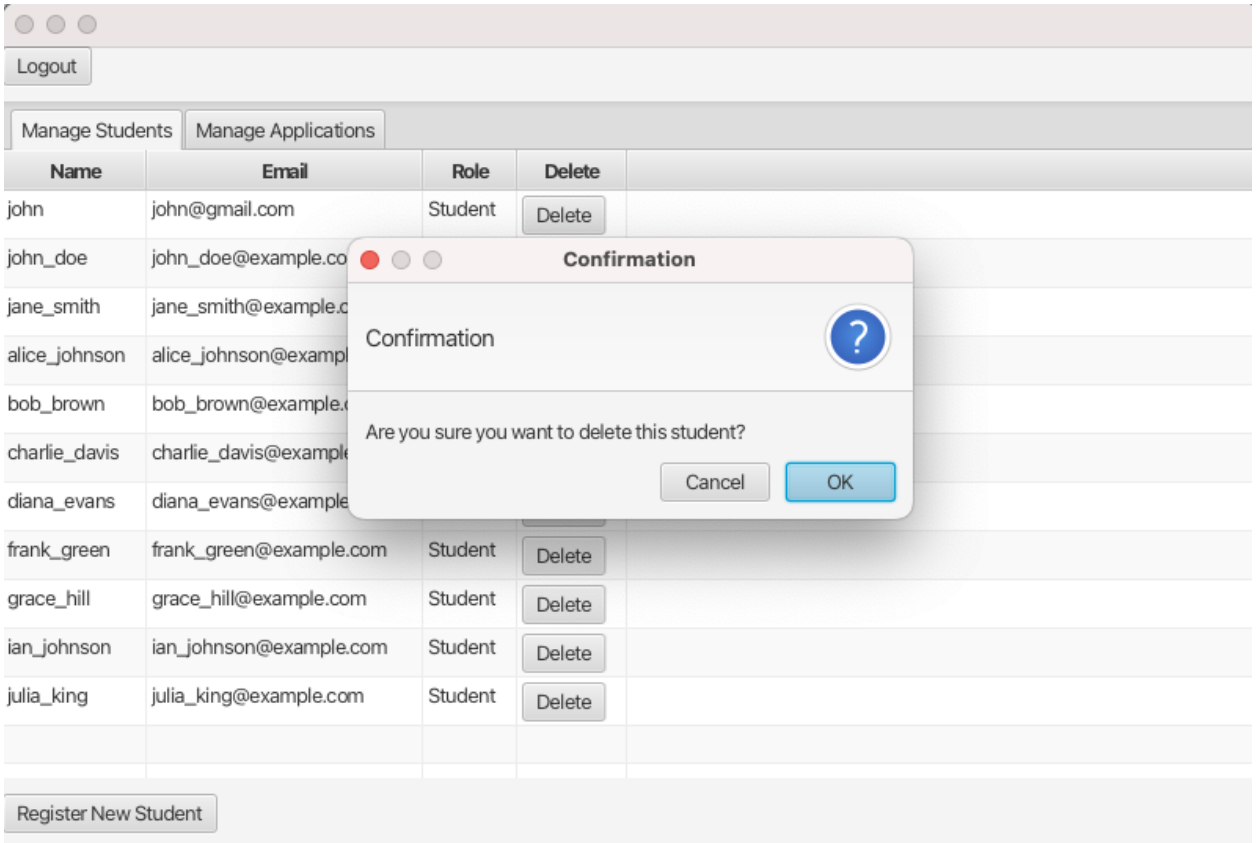
Successful Login for an Admin
Description: This snapshot demonstrates the admin user successfully logging into the system.



Update Performed by Admin
Description: Illustration of an admin user updating the first record in the database.



Deletion of a Record by Admin
Description: This snapshot shows an admin user deleting the last record in the database.



Admin Mode: Remaining Records
Description: A view of the remaining records in the database as seen by an admin in a columnar layout.

Logout

Manage Students | Manage Applications

Name	Email	Role	Delete
john	john@gmail.com	Student	Delete
john_doe	john_doe@example.com	Student	Delete
jane_smith	jane_smith@example.com	Student	Delete
alice_johnson	alice_johnson@example.com	Student	Delete
bob_brown	bob_brown@example.com	Student	Delete
charlie_davis	charlie_davis@example.com	Student	Delete
diana_evans	diana_evans@example.com	Student	Delete
frank_green	frank_green@example.com	Student	Delete
grace_hill	grace_hill@example.com	Student	Delete
ian_johnson	ian_johnson@example.com	Student	Delete
julia_king	julia_king@example.com	Student	Delete

Register New Student

Logout

Manage Students | Manage Applications

Type	Status	Approve	Deny
Leave	Approved	Approve	Deny
Registration	Approved	Approve	Deny
Revaluation	Denied	Approve	Deny
Transcript	Approved	Approve	Deny
AdditionalSlot	Pending	Approve	Deny
Supplementary	Denied	Approve	Deny
Leave	Approved	Approve	Deny
Registration	Pending	Approve	Deny
Transcript	Approved	Approve	Deny
Revaluation	Pending	Approve	Deny
AdditionalSlot	Denied	Approve	Deny
Supplementary	Denied	Approve	Deny

User Table Snapshot
Description: Snapshot of the UniversityUsers table, displaying user details (excluding sensitive information).

universityusers | Enter a SQL expression to filter results (use Ctrl+Space)

	user_id	name	email	hashed_password	salt	role
1	1	admin	admin@gmail.com	7fd51278f0b396e9b70baa9748615c43fa9	6700870e20677aaa8fbb954872dbbf6d	Admin
2	2	john	john@gmail.com	995cb2e4cd85b97fb84db22f637f32ee2a1	b53ca2a9c02938091c502e6d5a1b4d33	Student
3	3	john_doe	john_doe@example.com	7a4e95c229f665e7383c7a57c9fc0ecce05	114a23c4ea68e144d6c5105c6739a477	Student
4	4	jane_smith	jane_smith@example.com	e2a18e70dd08674e6101b37a060ff9c1e63	35e61b9dba08e1f0d81b07f3f4aa6114	Student
5	5	alice_johnson	alice_johnson@example.com	7c5161651b9daa2a32661af71f6fb18371f36	64f3e052f8036214615f8cf39a8a0bb7	Student
6	6	bob_brown	bob_brown@example.com	e47462cbd85c92a84c3c16a6d754cde5d7	c1509f4affc8710f1dcc1c96eb15b103	Student
7	7	charlie_davis	charlie_davis@example.com	9e829d253b3b5301d123175680123f2b19	229a10e5156d3b59fd7ff486f296354b	Student
8	8	diana_evans	diana_evans@example.com	24baa5bcaa3f669995178000d6101e495a	7b0267cb77b0500ac1d4b95649bdef74	Student
9	9	frank_green	frank_green@example.com	4969696156339491298092d6448ce3671	1df7c758cdc3072c1414ebf8b082a276	Student
10	10	grace_hill	grace_hill@example.com	1057802f8d2a77ec466572a4de4c86846b	0fb7559e0c024e0fbdf949532233331b	Student
11	11	ian_johnson	ian_johnson@example.com	302d0471650259216a71f099af68c6c62c5	3088313ac95b098d375159bd280a19eb	Student
12	12	julia_king	julia_king@example.com	44390212e3a664cbd1e239eac19c0a6125	0adda0cbdf3851195602289c993a1366	Student

Extra Credit Documentation

Password Hashing Implementation

In the University Management System, we have implemented an advanced password hashing mechanism to ensure the security of user credentials. This mechanism is crucial for protecting sensitive user data, particularly in an environment where personal and academic information is stored.

Code Snippets and Explanation:
The password hashing is implemented in the PasswordUtil class within the util package. We use SHA-256, a cryptographic hash function, along with a salt to hash user passwords. Here's a brief overview of the code:

```java
public static String getSalt() {
    SecureRandom random = new SecureRandom();
    byte[] salt = new byte[16];
    random.nextBytes(salt);
    return bytesToHex(salt);
}

public static String hashPassword(String passwordToHash, String salt) {
    String generatedPassword = null;
    try {
        MessageDigest md = MessageDigest.getInstance(algorithm:"SHA-256");
        md.update(hexStringToByteArray(salt));
        byte[] bytes = md.digest(passwordToHash.getBytes());
        generatedPassword = bytesToHex(bytes);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return generatedPassword;
}
```

Salt Generation: A random salt is generated for each user. This salt adds an additional layer of security, making it more difficult for attackers to use precomputed hash tables (rainbow tables) to crack passwords.

Hashing: The user's password is hashed using SHA-256 along with the salt. This hashed password is stored in the database, instead of the plain text password.

Security Enhancement: This approach significantly enhances the security of the system. Even if the database is compromised, attackers cannot easily decipher the actual passwords due to the complexity added by the hashing and salting process.

.jar File

An executable .jar file of the application is included for easy deployment and execution. This file packages the entire application, allowing it to be run on any system with a compatible Java Runtime Environment.

Execution Instructions:

- Ensure Java is installed on your system.
- Download the .jar file to your preferred location.
- Open a terminal or command prompt.
- Navigate to the directory where the .jar file is located.
- Run the command java -jar [filename].jar, replacing [filename] with the actual file name of the .jar file.

Conclusion

The JavaFX University Management System project has successfully demonstrated the creation of a comprehensive and secure application using modern software development techniques. Key achievements include the implementation of a user-friendly interface with JavaFX, the application of the MVC architectural pattern for clear separation of concerns, and the integration of a secure database system.

Throughout the project, we have learned valuable skills in JavaFX UI design, database management, and implementing security measures like password hashing. These skills are essential for modern software development and can be applied to a wide range of future projects, especially in areas requiring high data security and user interaction.

This project not only serves as a testament to our technical abilities but also as a foundation for future development and learning in the field of software engineering.