
SCHOOL OF ENGINEERING AND TECHNOLOGY

FINAL ASSESSMENT FOR THE BSC (HONS) INFORMATION TECHNOLOGY; BSC (HONS) COMPUTER SCIENCE; BACHELOR of SOFTWARE ENGINEERING (HONS) YEAR 2

ACADEMIC SESSION 2023; SEMESTER 3

PRG2104: OBJECT ORIENTED PROGRAMMING

Project

DEADLINE: Week 14

INSTRUCTIONS TO CANDIDATES

- This assignment will contribute 50% to your final grade.
- This is an individual assignment.

IMPORTANT

-The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.


-Coursework submitted after the deadline will be awarded 0 marks

Lecturer's Remark (Use additional sheet if required)

I Aisha Sofia Binti Najidi (Name) 20065231 std. ID received the assignment and read the comments (Signature/date)

Academic Honesty Acknowledgement

"I . Aisha Sofia Binti Najidi (student name). verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties (*refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme*) for any kind of copying or collaboration on any assignment."

.....  (Student's signature / Date)

Contents

UML Diagram.....	3
Personal Reflection	4
Object-Oriented Concepts in This Assignment.....	4
<input type="checkbox"/> Inheritance	4
<input type="checkbox"/> Polymorphism	4
<input type="checkbox"/> Abstract Class	4
<input type="checkbox"/> Generic Programming.	4
The Problems Encountered and Solutions.	5
<input type="checkbox"/> Issue with IntelliJ being unable to run my files.	5
<input type="checkbox"/> Issue with Date not Displaying correctly.	5
An Evaluation of The Strengths and Weaknesses of Your Submitted Work.....	5
Images Of Program	6
References	11

VIDEO DEMONSTRATION: <https://youtu.be/uBmbSaC1Bl8>

UML Diagram

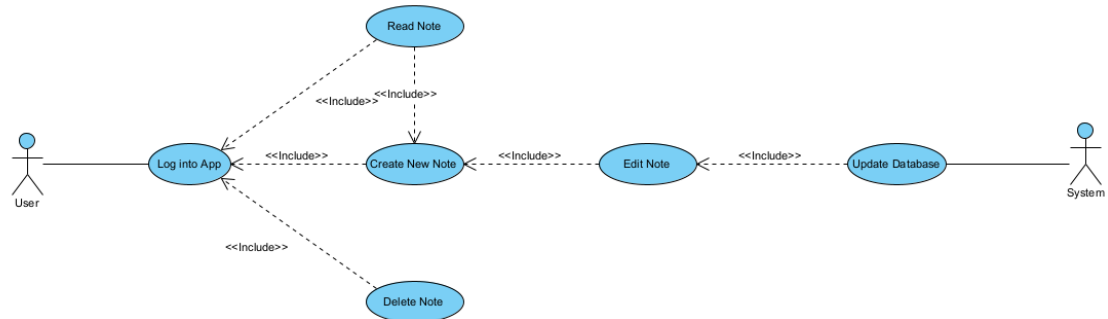


Figure 1 Use Case Diagram

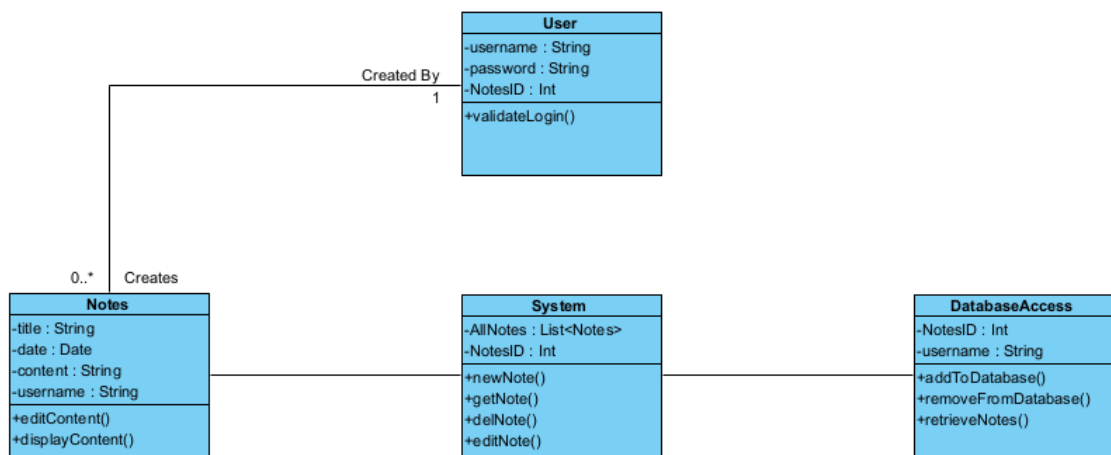


Figure 2 UML Class Diagram

Personal Reflection

Object-Oriented Concepts in This Assignment.

- *Inheritance*

Inheritance is a mechanism in object-oriented programming that allows one class to inherit properties and behaviors from another class.

Both the **Note** and **NotebookUser** classes extend the **Database** class, which means they inherit the functionality defined in the **Database** class. This functionality includes database setup and table initialization methods. By inheriting from the **Database** class, the **Note** and **NotebookUser** classes reuse the common database-related functionality without needing to reimplement it.

- *Polymorphism*

Polymorphism allows objects of different classes to be treated as objects of a common superclass, providing flexibility in code design and usage.

In the program, the **noteTable** is a **TableView** that takes a type parameter of **Note**. This enables various subclasses of **Note** to be displayed in the table. The **ObservableBuffer** holds a collection of **Note** objects, and different subclasses of **Note** can be added to it, which means that it is an example of polymorphism.

- *Abstract Class*

An abstract class is a class that cannot be instantiated on its own and often serves as a blueprint for other classes.

In the program, the **Database** trait defines common database functionality like database setup and table initialization. The **Note** and **NotebookUser** objects extend the **Database** trait and implement their specific table initialization logic, leveraging the common functionality provided by the abstract class.

- *Generic Programming.*

Generic programming enables the creation of classes, methods, and data structures that work with multiple types without specifying those types explicitly.

In the program, the use of generics can be seen in classes like **TableView** and **TableColumn**. For example, **TableView[Note]** specifies that the table can hold items of type **Note**, allowing the same table structure to be reused for different data types if needed. Similarly, the **TableColumn** class uses generics to define the type of data that the column holds.

The Problems Encountered and Solutions.

- *Issue with IntelliJ being unable to run my files.*

Due to this situation, it was unclear if IntelliJ, SBT, or Scala itself was the main cause of the problem. I looked into a number of options to solve this problem, including uninstalling and reinstalling Scala, making sure the Scala SDK was installed correctly, and checking the SBT integration in IntelliJ. Yet, the issue persisted, so I moved to using the command prompt to run my files.

- *Issue with Date not Displaying correctly.*

To address this issue, I modified the data type and operations related to date and time information in the **Note** class. Specifically, the data type for properties like **lastModified** and **dateCreated** has been updated from **LocalDate** to **OffsetDateTime**. This allows properties to store not only date information but also precise time and timezone offsets.

By using **OffsetDateTime** instead of just **LocalDate**, the program can now handle both date and time components accurately, including timezone offsets, to correctly display and manipulate date and time information.

An Evaluation of The Strengths and Weaknesses of Your Submitted Work.

In the development of my note app, an in-depth exploration of user login functionality, secure note storage, and user-specific data access was undertaken.

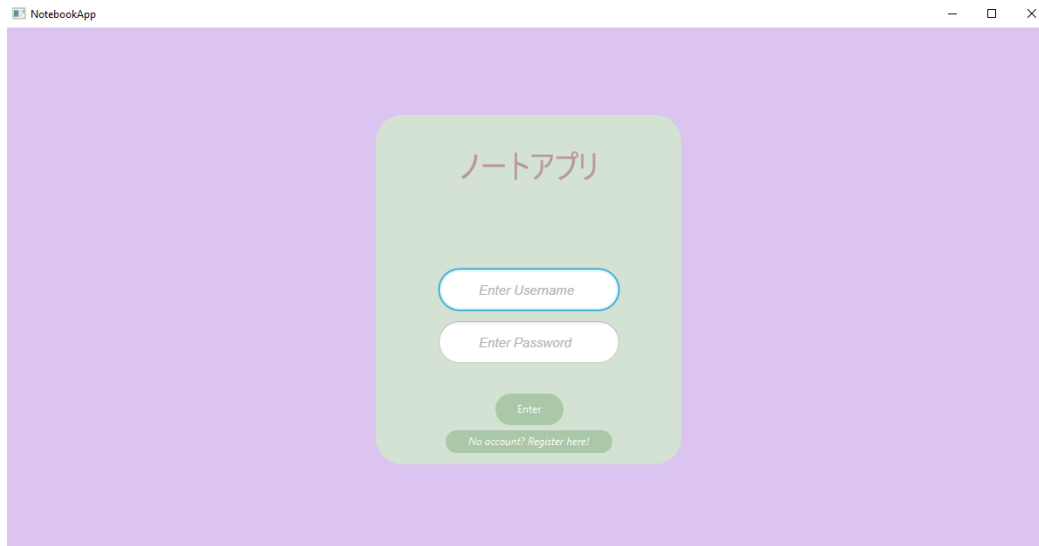
I believe one of the strengths of my program are user authentication and data isolation. By successfully implementing a user authentication system, the application enables users to safely log in using their own credentials. Furthermore, each user can only read and manage their own notes because of the user-specific data isolation feature. This proves that data security and privacy are prioritized. Next, another strength is database implementation. I believe that a thorough understanding of data persistence was demonstrated by the integration of a database to store user data and notes. This improves the overall user experience by allowing users to easily retrieve their notes upon subsequent logins. Next is scalability and multi-user support. The application's design was created to support numerous users. The system's capability to control several user accounts and restrict access to other people's notes shows its scalability and well-considered user management strategy.

Moving on to the weaknesses of this program, the main weakness of my program is the limited user experience enhancement. The project primarily focuses on essential fundamental features like data management and authentication. But in the future, adding features like note grouping, search functionality, or even collaborative note-sharing could result in an enhanced user experience. Another weakness is the lack of responsive design. The application's user interface might not be suited for different devices and screen sizes, which could cause usability problems on some platforms. Utilizing responsive design concepts may enhance usability and accessibility across a range of devices.

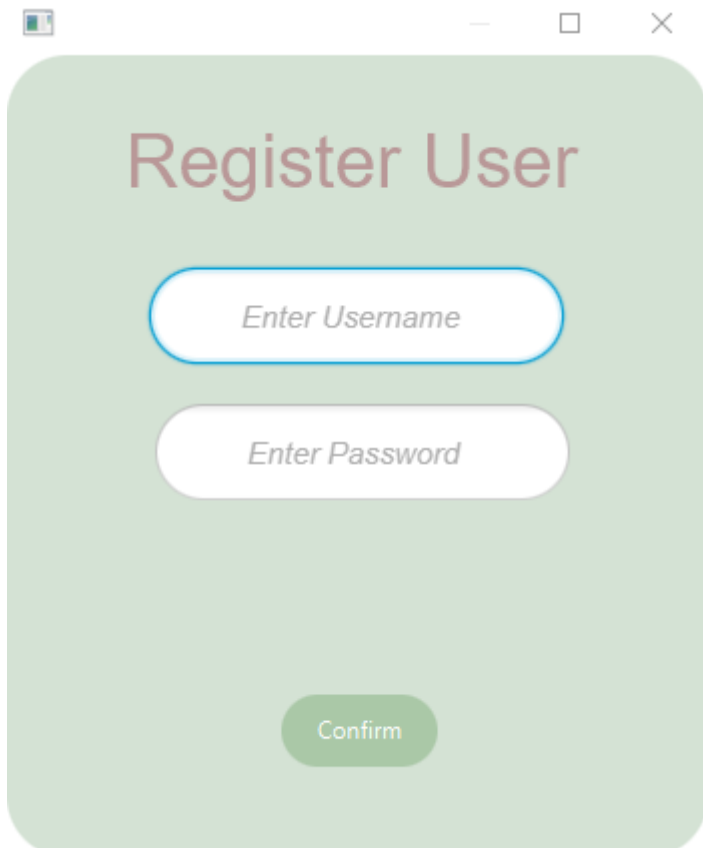
Some technical weaknesses are that the program is unable to display the NotesID without the user closing the program and reopening it once again, this is one problem I was not able to find the solution to due to my lack of experience with databases and scala.

Images Of Program

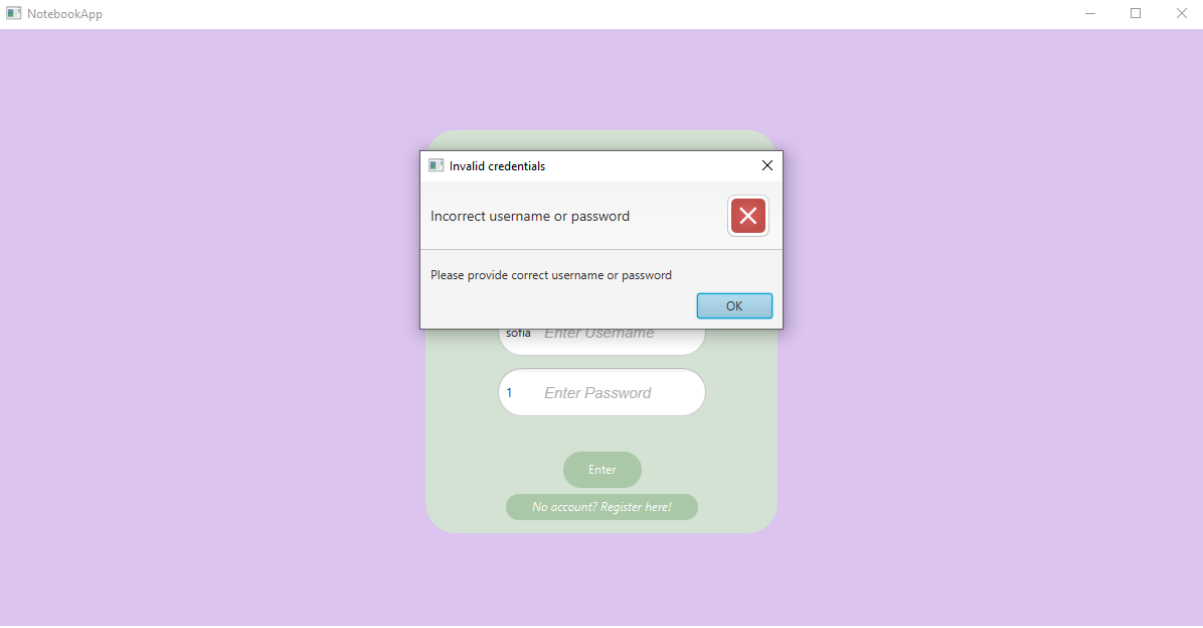
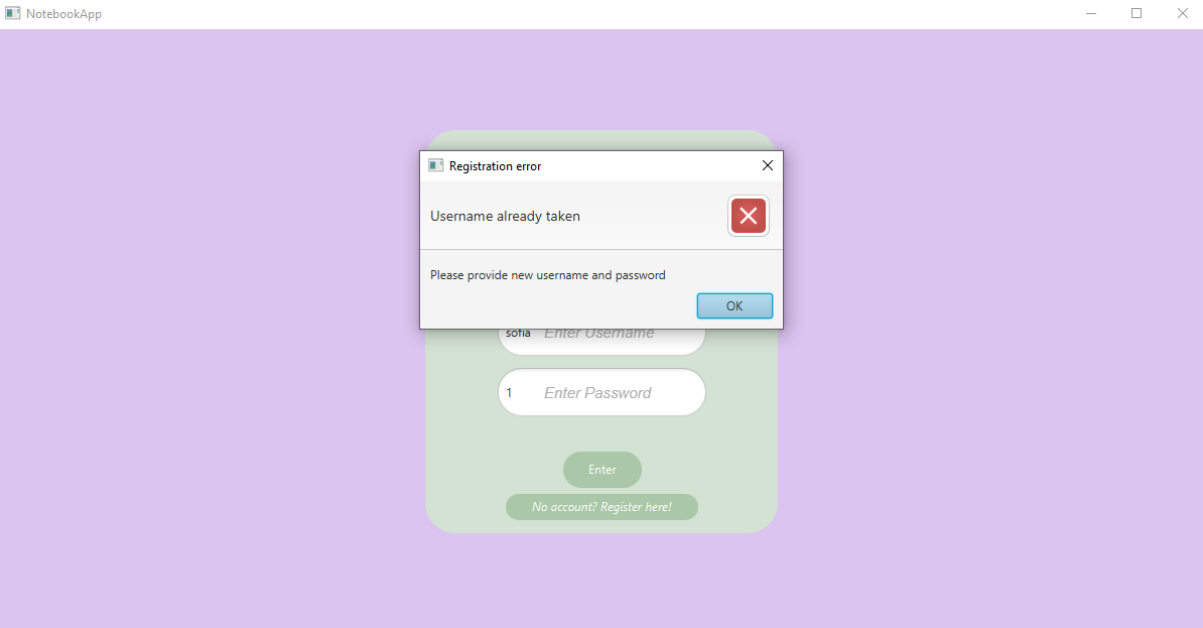
Log In Screen



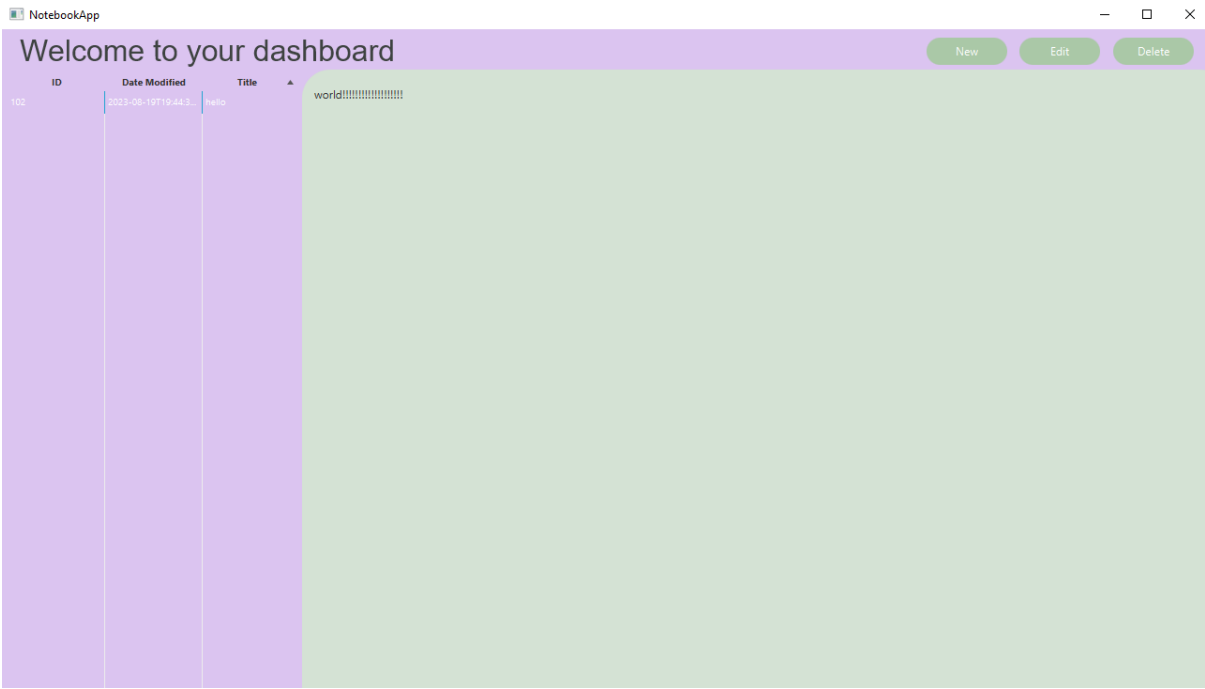
Register Screen



Validation Checks



Main Screen



New Note



Edit Note

— □ ×

EDITED Enter Title

sofia

2023-08-20T07:56:20...

2023-08-20T07:56:20...

AAAAAAAAAAAAAAAAAAAAAAAAAAAA

Save

Cancel

Main Screen

NotebookApp

— □ ×

Welcome to your dashboard

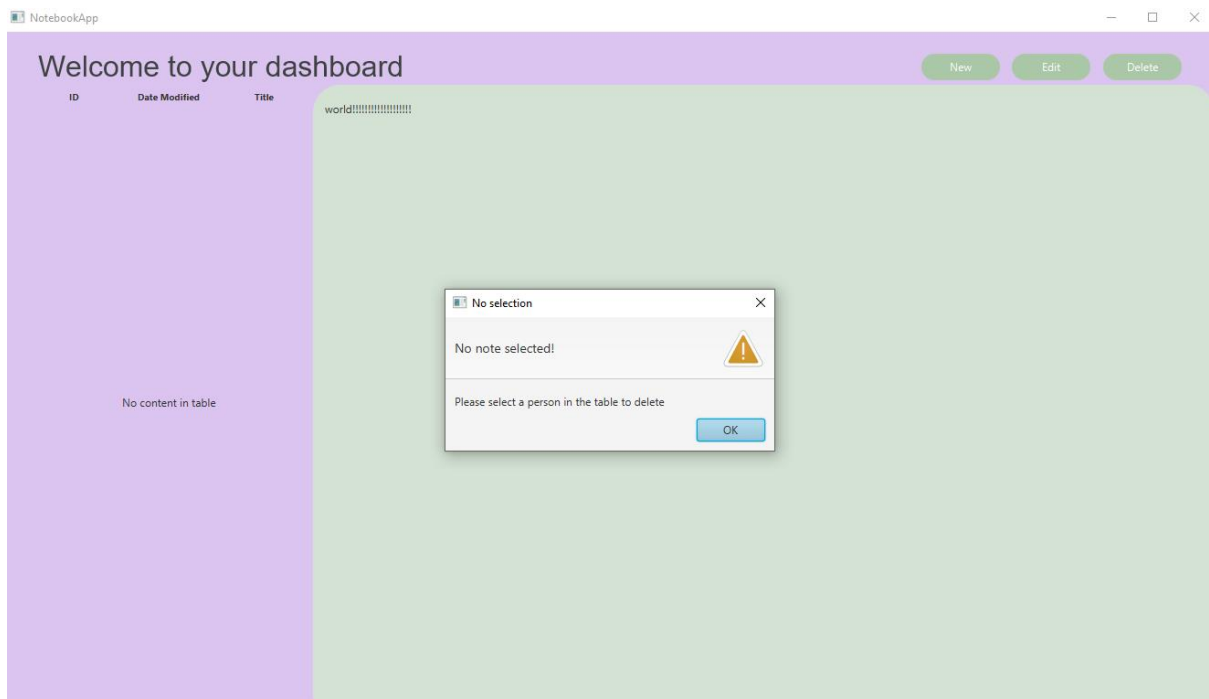
New

Edit

Delete

ID	Date Modified	Title	
102	2023-08-19T19:44:3...	hello	AAAAAAAAAAAAAAAAAAAAAAAAAAAA
	2023-08-20T07:56:2...	EDITED	

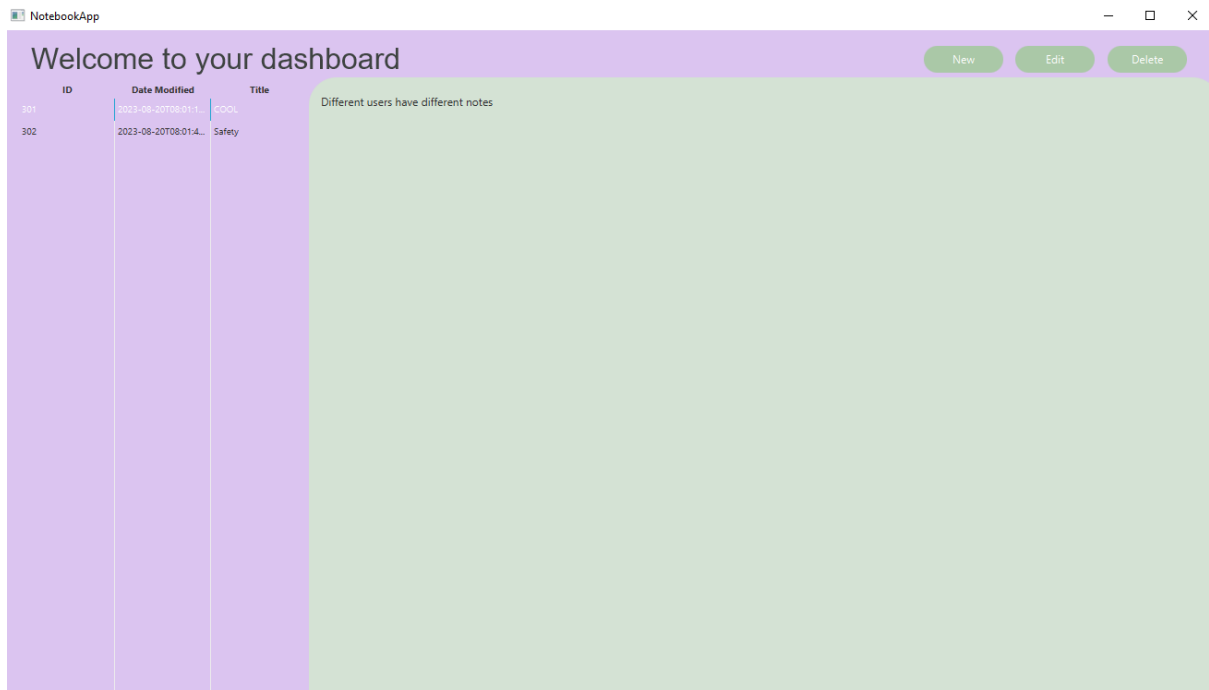
Validation Check



Different User Log In



Different Notes are Displayed.



References

AddressAppG1. Dr Chin Teck Min (2023)

Color palette. Color Hunt. (n.d.). <https://colorhunt.co/palette/faf3f0d4e2d4ffcaccd4bc4f0>

Operations. ScalikeJDBC. (n.d.). <http://scalikejdbc.org/documentation/operations.html>

Oracle Java™ Platform, Standard Edition 7 API Specification. Preparedstatement (Java Platform SE 7). (2020, June 24).
[https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/java/sql/PreparedStatement.html#executeUpdate\(\)](https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/java/sql/PreparedStatement.html#executeUpdate())

Scalafx. (n.d.). *Missing implicit conversions from property to observablevalue*. GitHub.
<https://github.com/scalafx/scalafx/issues/243>

Scala Standard Library. Scala Standard Library 2.13.11 - scala.collection. (n.d.).
<https://www.scala-lang.org/api/current/scala/collection/index.html>