MAY SEMESTER 2024

TFB1033/TEB1043

OBJECT ORIENTED PROGRAMMING

GROUP NAME: BYTE KURMA

PROJECT TITLE: FINANCIAL TRACKER

LECTURER: DR. NORDIN ZAKARIA

PREPARED BY:

NO	NAME	STUDENT ID	PROGRAMME
1	SYAHIR AMRI BIN MOHD AZHA	22007728	COMPUTER SCIENCE
2	MANAS ISMAIL ABDYLAS	22008600	COMPUTER SCIENCE
3	MUHAMMAD AQIL BIN ANUAR	24000198	INFORMATION TECHNOLOGY
4	DANIELA ADLIN BINTI RAZWAN	22008820	INFORMATION TECHNOLOGY
5	AHMAD AFIF DANIAL BIN AZHARI	22009264	INFORMATION TECHNOLOGY

TABLE OF CONTENTS

INTRODUCTION

In today's fast-paced world, effective personal finances management is vital for acquiring financial stability and peace of mind. However, keeping track of daily expenses and ensuring that financial goals are met can be a challenging task. To address this issue, we are developing a user-friendly and reliable expense recording application that allow individuals to easily track and record their expenses. This project utilizes object-oriented programming (OOP) concepts to develop scalable and sustainable software that meets users' current demands while also adapting to future requirements. Our application will offer features like expense categorization, budgeting, and extensive financial reporting, allowing users to gain insight into their spending habits and make informed financial decisions. With an emphasis on intuitive design and user experience, we aim to create a tool that encourages continuous use and promotes better financial planning. By leveraging OOP, we provide a modular, manageable codebase that can adapt to future needs, resulting in improved financial knowledge and well-being.

OBJECTIVES

1. Develop an Intuitive User Interface
   - Create a visually appealing and easy-to-navigate interface.
   - Ensure that the design allows for effortless expense recording and tracking.
   - Implement user-friendly features such as quick input forms, searchable categories, and customizable dashboards.

2. Implement Robust Expense Tracking and Categorization
   - Develop a system to record and categorize daily expenses.
   - Allow users to create and manage custom expense categories.
   - Enable users to add notes or tags to individual transactions for better context.

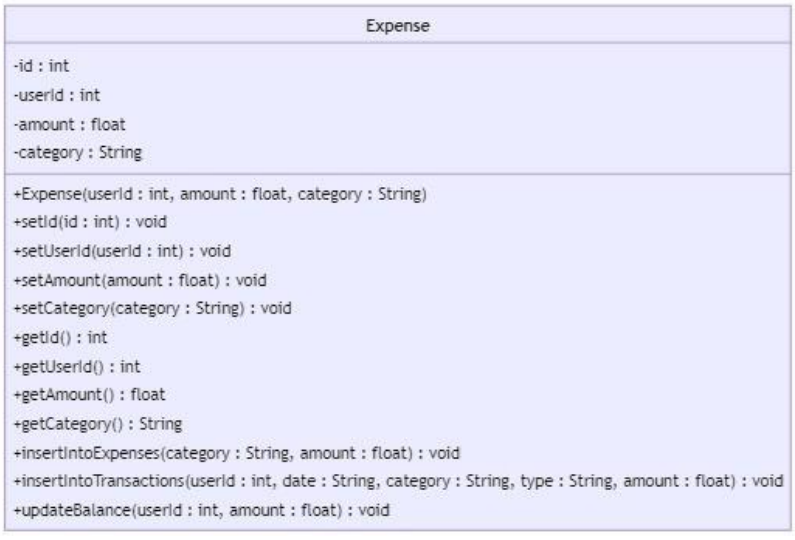
3. Design Comprehensive Budgeting Tools
   - Introduce budgeting features that help users set and monitor spending limits.
   - Provide alerts and notifications when budgets are close to being exceeded.
   - Include options for recurring budgets (e.g., monthly, quarterly).

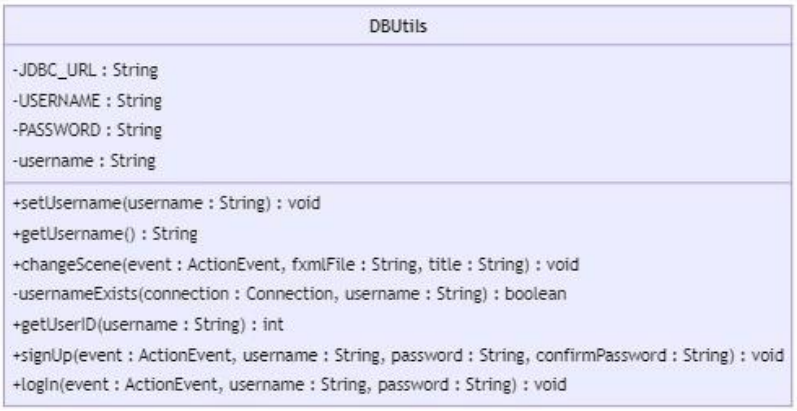
4. Develop Advanced Financial Reporting
   - Create detailed reports and visualizations (graphs, charts) to illustrate spending patterns.
   - Offer comparative analysis over different time periods (e.g., month-over-month, year-over-year).
   - Include export options for reports in various formats (PDF, Excel).

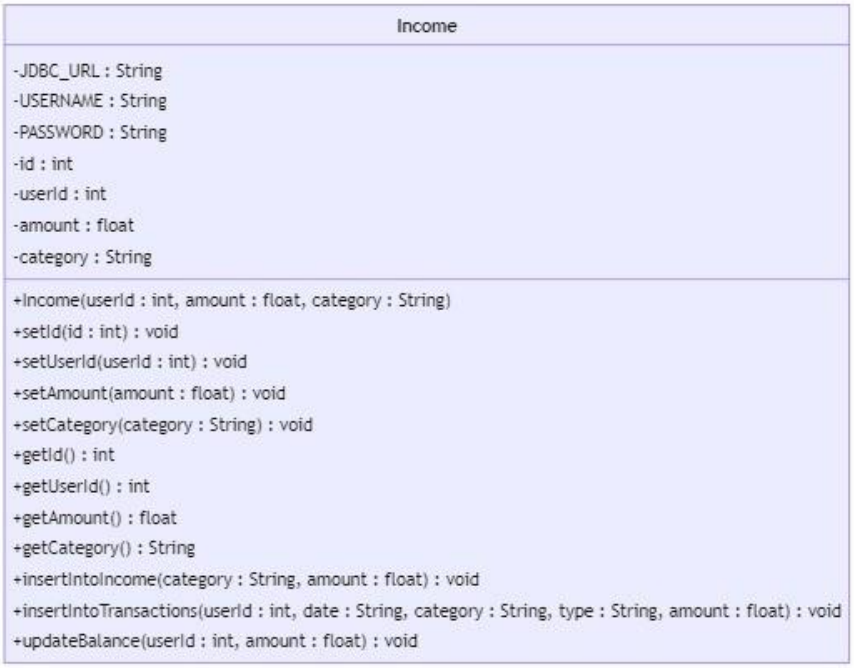
5. Ensure Data Security and Privacy
   - Implement secure authentication and authorization mechanisms.
   - Ensure that all user data is encrypted both in transit and at rest.
   - Provide users with clear privacy policies and data management options.

UML DIAGRAM


User
-id : int
-username : String
-password : String
-firstName : String
-lastName : String
-email : String
-DOB : LocalDate
-balance : float
+User(id : int, balance : float, username : String, password : String, firstName : String, lastName : String, DOB : LocalDate)
+setId(id : int) : void
+setUsername(username : String) : void
+setPassword(password : String) : void
+setFirstName(firstName : String) : void
+setLastName(lastName : String) : void
+setEmail(email : String) : void
+setDOB(DOB : LocalDate) : void
+setBalance(balance : float) : void
+getId() : int
+getUsername() : String
+getPassword() : String
+getFirstName() : String
+getLastName() : String
+getEmail() : String
+getDOB() : LocalDate
+getBalance() : float

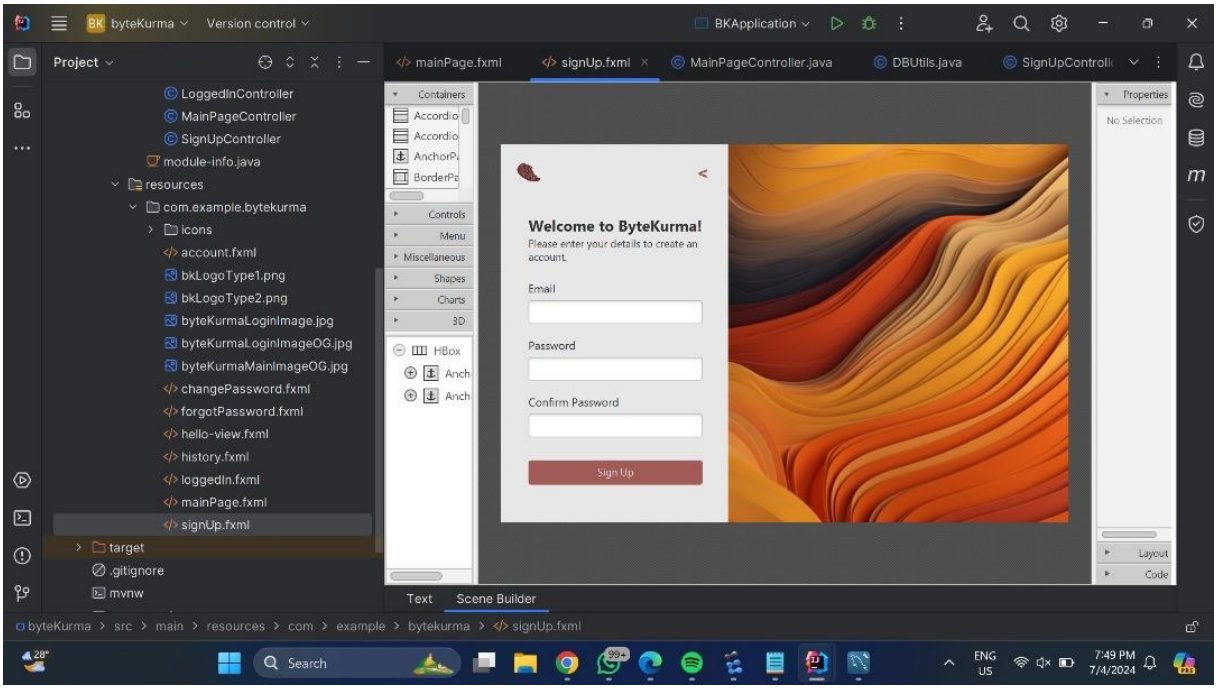

Expense
-id : int
-userId : int
-amount : float
-category : String
+Expense(userId : int, amount : float, category : String)
+setId(id : int) : void
+setUserId(userId : int) : void
+setAmount(amount : float) : void
+setCategory(category : String) : void
+getId() : int
+getUserId() : int
+getAmount() : float
+getCategory() : String
+insertIntoExpenses(category : String, amount : float) : void
+insertIntoTransactions(userId : int, date : String, category : String, type : String, amount : float) : void
+updateBalance(userId : int, amount : float) : void


DBUtils
-JDBC_URL : String
-USERNAME : String
-PASSWORD : String
-username : String
+setUsername(username : String) : void
+getUsername() : String
+changeScene(event : ActionEvent, fxmlFile : String, title : String) : void
-usernameExists(connection : Connection, username : String) : boolean
+getUserID(username : String) : int
+signUp(event : ActionEvent, username : String, password : String, confirmPassword : String) : void
+logIn(event : ActionEvent, username : String, password : String) : void

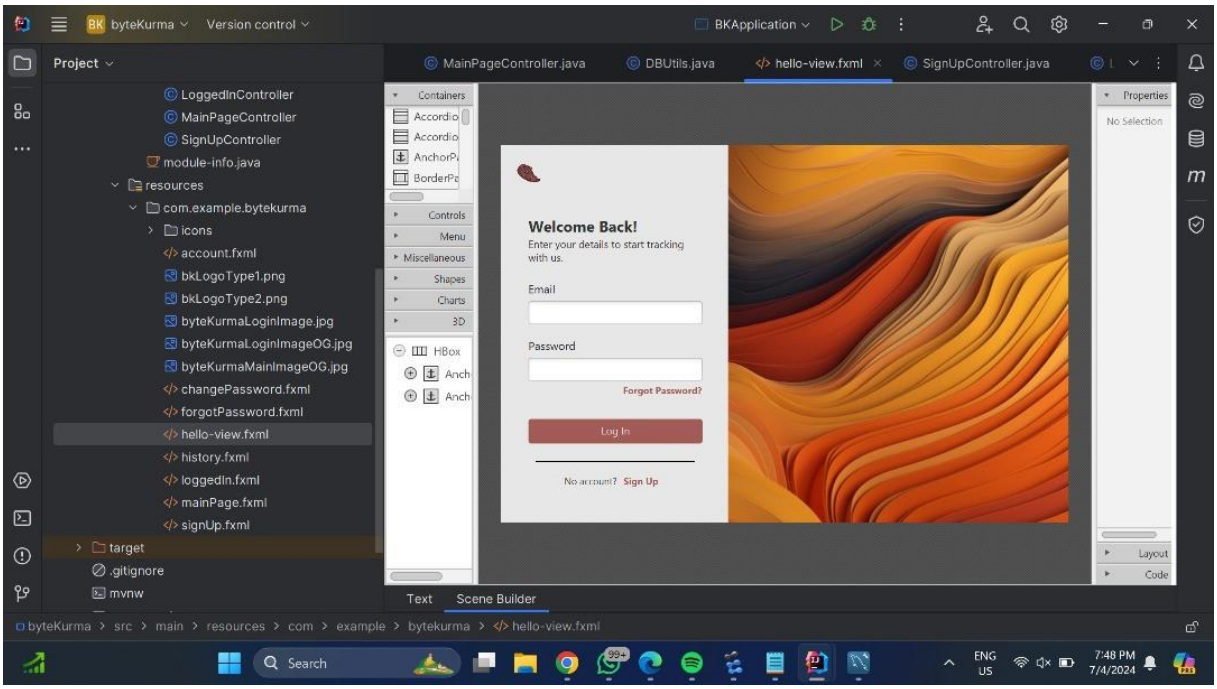

Income
-JDBC_URL : String
-USERNAME : String
-PASSWORD : String
-id : int
-userId : int
-amount : float
-category : String
+Income(userId : int, amount : float, category : String)
+setId(id : int) : void
+setUserId(userId : int) : void
+setAmount(amount : float) : void
+setCategory(category : String) : void
+getId() : int
+getUserId() : int
+getAmount() : float
+getCategory() : String
+insertIntoIncome(category : String, amount : float) : void
+insertIntoTransactions(userId : int, date : String, category : String, type : String, amount : float) : void
+updateBalance(userId : int, amount : float) : void


History
-id : int
-userId : int
-date : String
-category : String
-type : String
-amount : float
+History(date : String, category : String, type : String, amount : float)
+setId(id : int) : void
+setUserId(userId : int) : void
+setDate(date : String) : void
+setCategory(category : String) : void
+setType(type : String) : void
+setAmount(amount : float) : void
+getId() : int
+getUserId() : int
+getDate() : String
+getCategory() : String
+getType() : String
+getAmount() : float

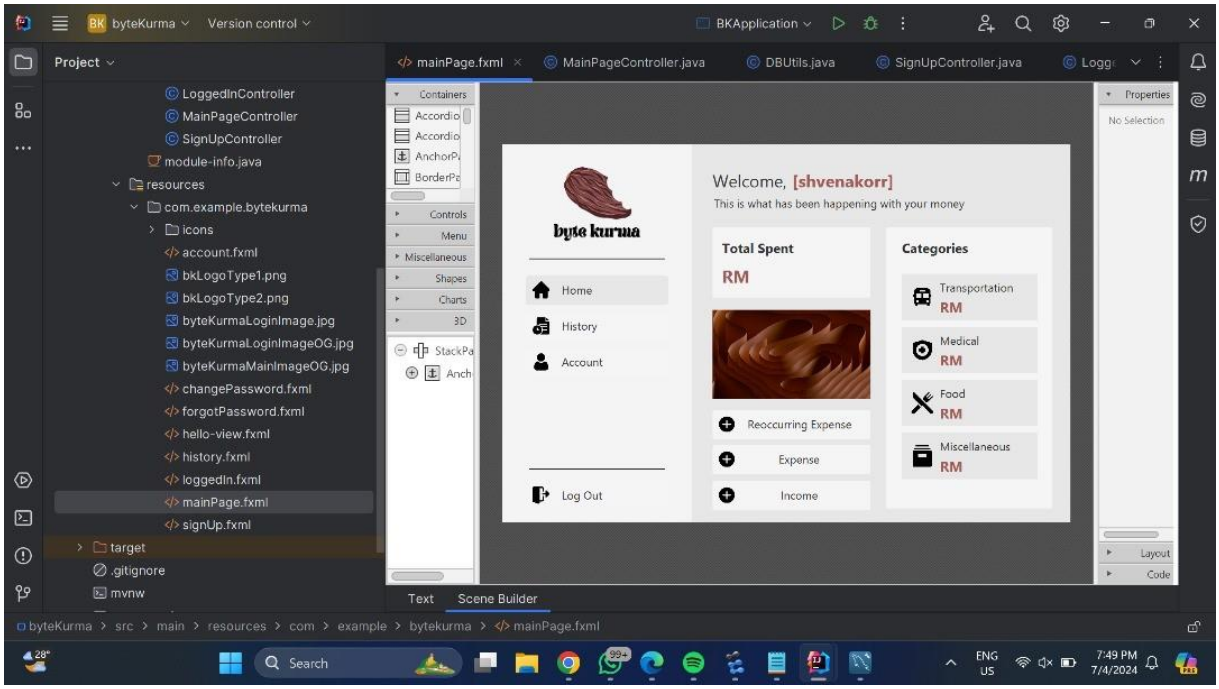
GRAPHICAL USER INTERFACE (GUI)



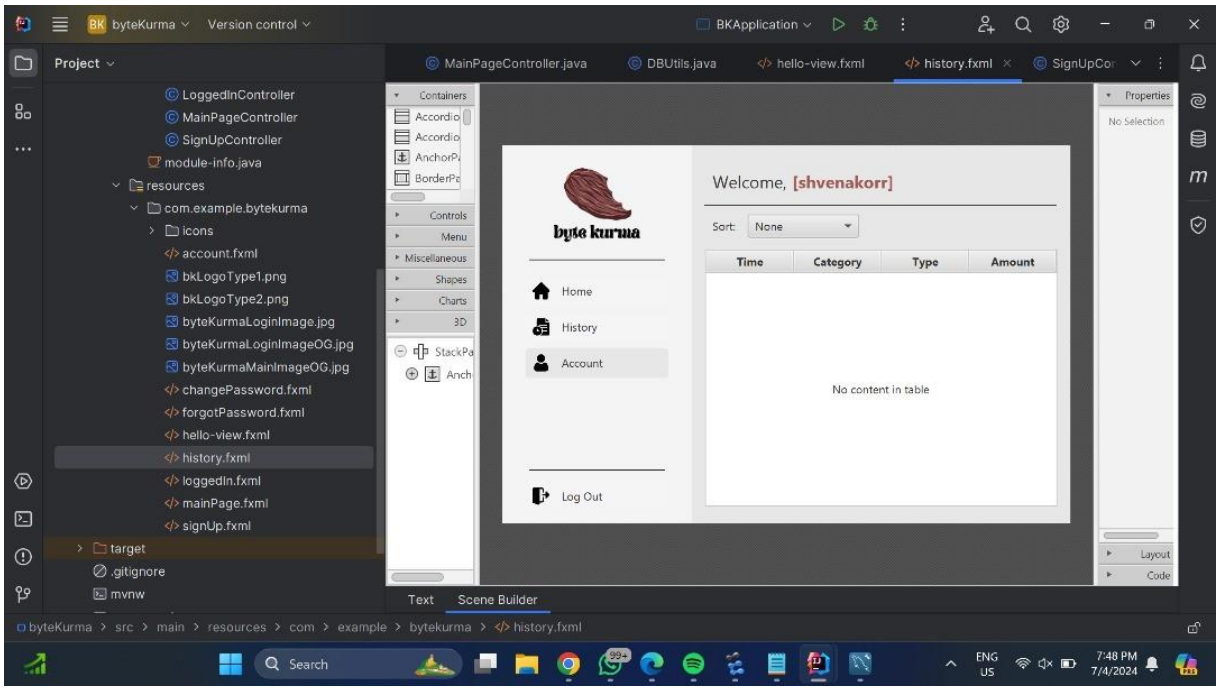
Sign up



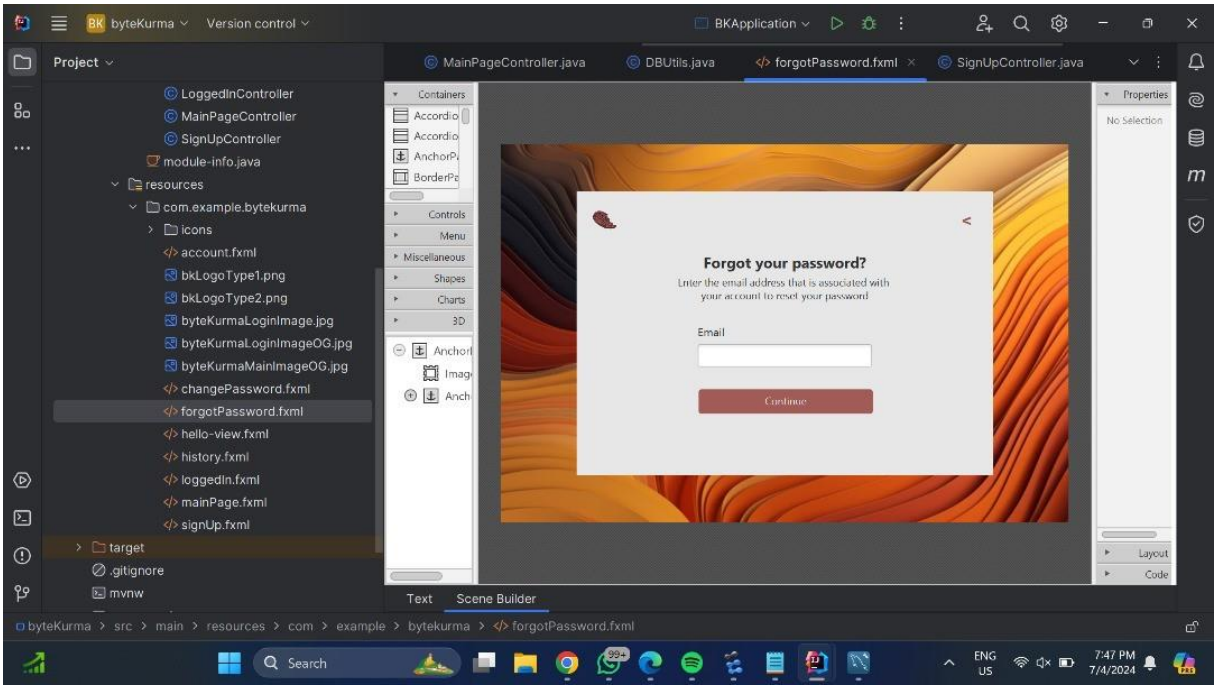
Log in



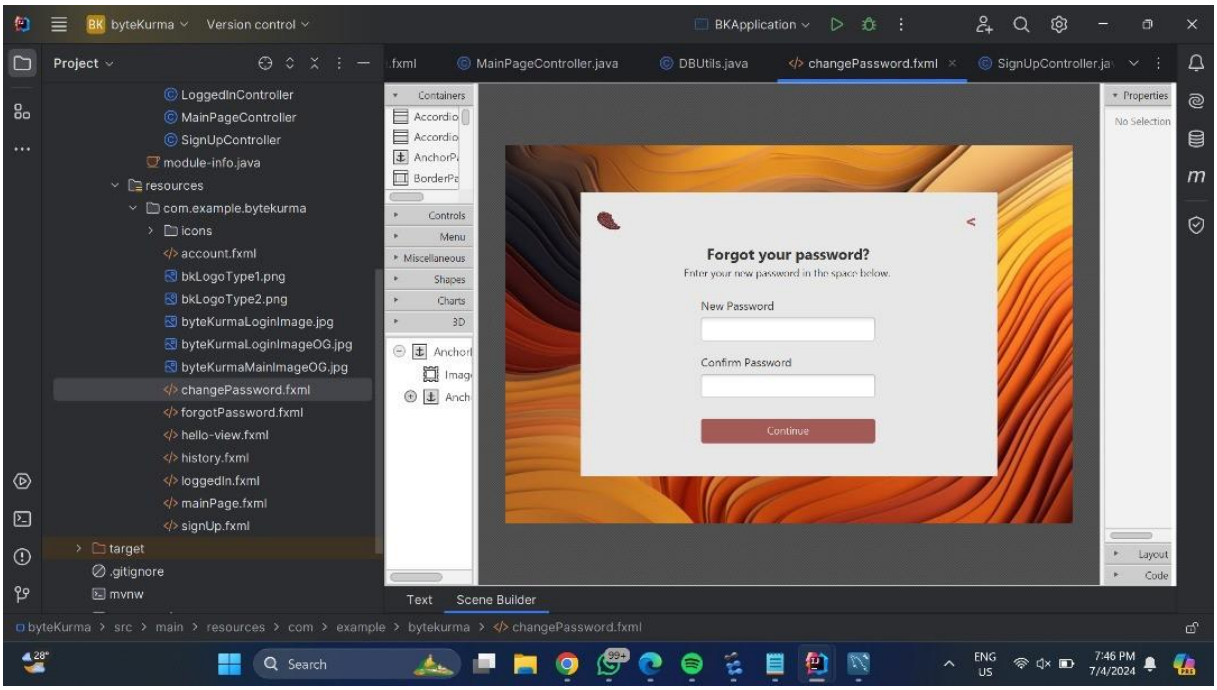
Main page



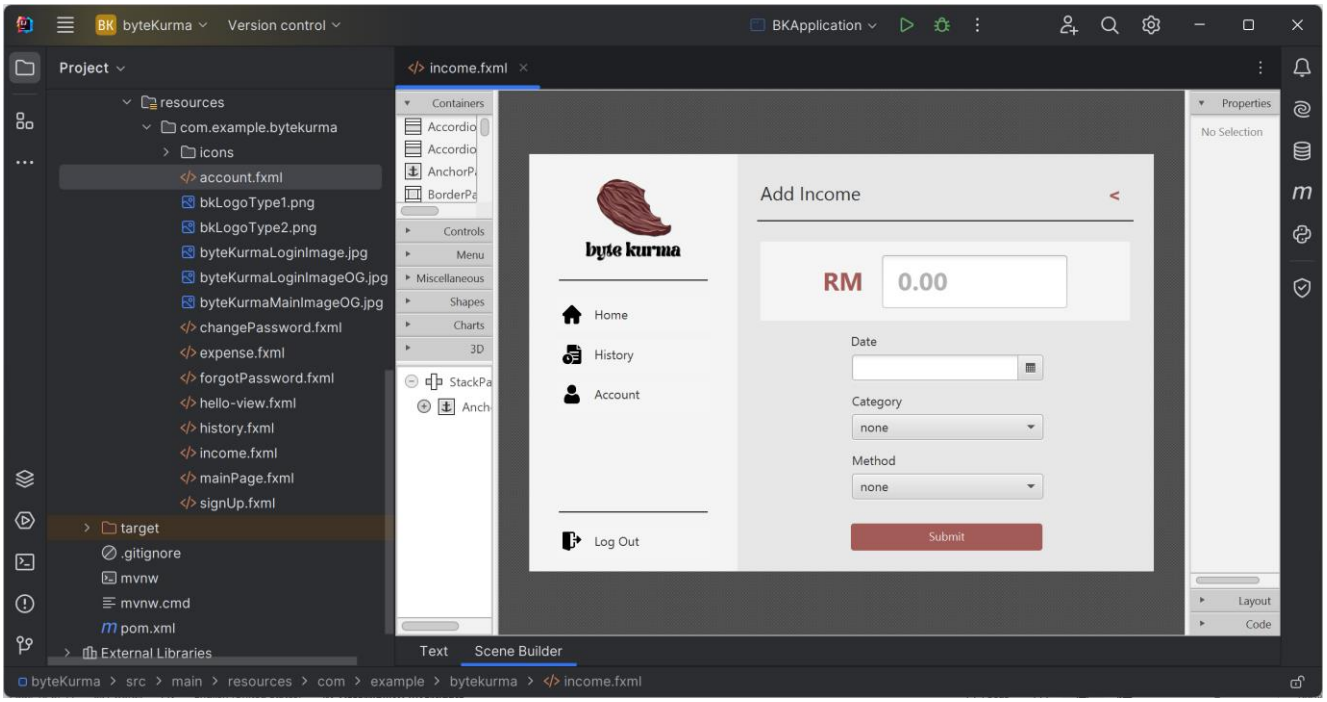
History



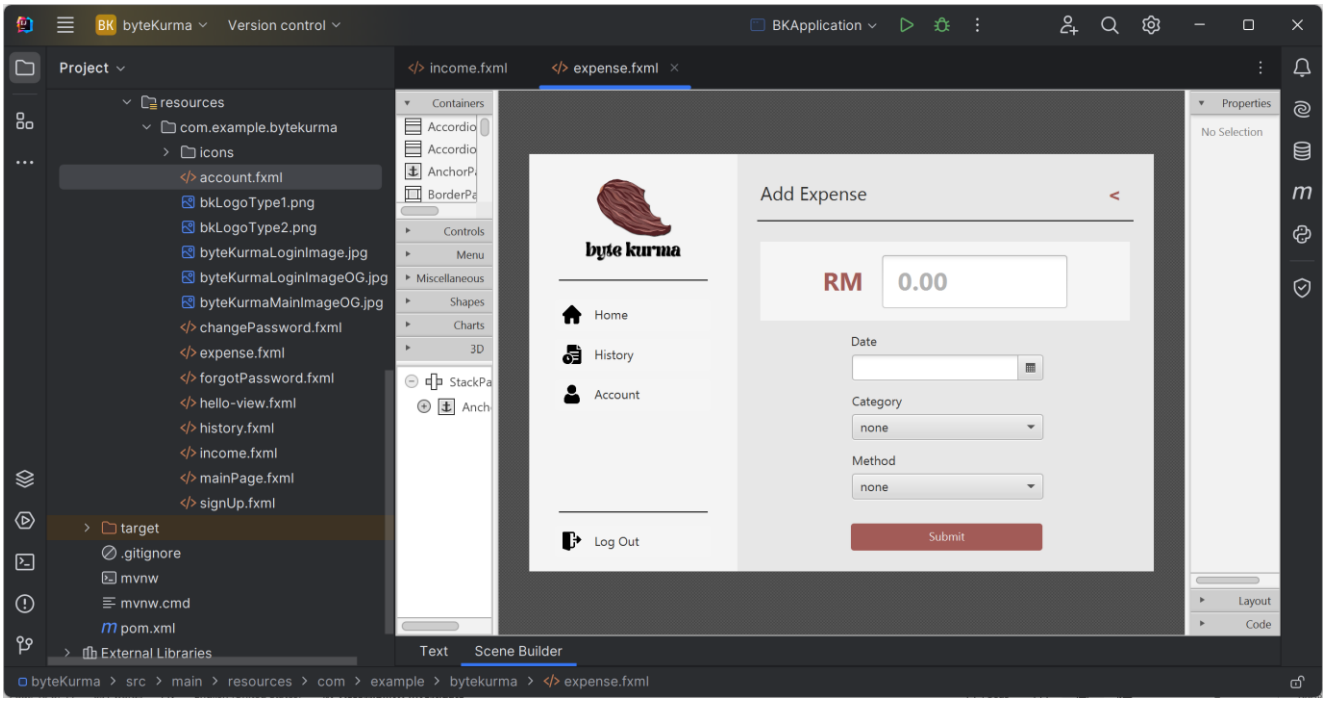
Forgot Password



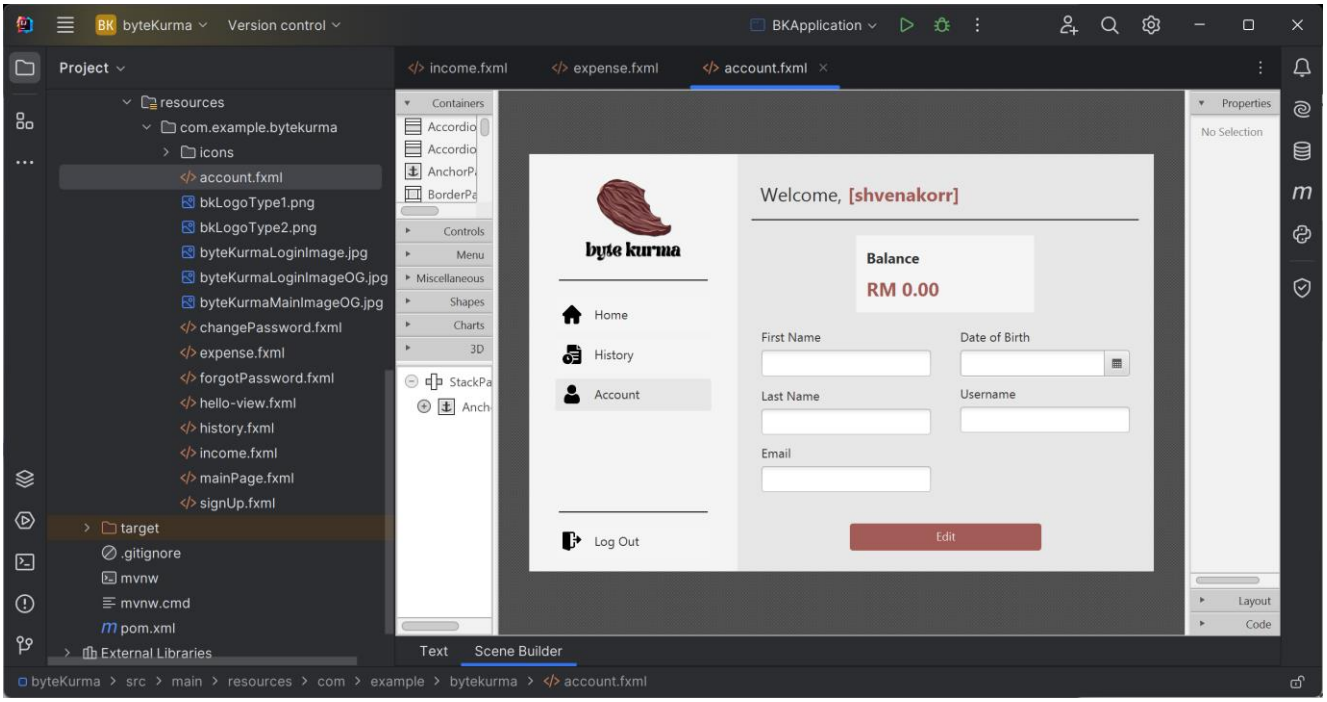
Change Password



Income



Expense


byteKurma
Version control
BKApplication
Project
income.fxml
expense.fxml
account.fxml
resources
com.example.bytekurma
icons
account.fxml
bkLogoType1.png
bkLogoType2.png
byteKurmaLoginImage.jpg
byteKurmaLoginImageOG.jpg
byteKurmaMainImageOG.jpg
changePassword.fxml
expense.fxml
forgotPassword.fxml
hello-view.fxml
history.fxml
income.fxml
mainPage.fxml
signUp.fxml
target
.gitignore
mvnw
mvnw.cmd
pom.xml
External Libraries
Containers
Controls
Menu
Miscellaneous
Shapes
Charts
3D
Properties
No Selection
Layout
Code
byte kurma
Welcome, [shvenakorr]
Balance
RM 0.00
Home
History
Account
Log Out
First Name
Date of Birth
Last Name
Username
Email
Edit
Text
Scene Builder
byteKurma > src > main > resources > com > example > bytekurma > account.fxml

Account

CONCLUSION

To sum up, our expense recording tool uses Object-Oriented Programming (OOP) principles to efficiently handle personal finances. This software solution is modular and flexible, allowing it to satisfy the present demands of users and adapt to future requirements. The application includes a user-friendly interface, powerful spending monitoring and categorisation, extensive budgeting tools, advanced financial reporting, and strict data protection measures. These features enable users to monitor their daily expenses, establish and oversee budgets, analyse their spending patterns, and make well-informed financial choices, all while guaranteeing the security of their data.

The intuitive design and comprehensive reporting features promote consistent usage and facilitate enhanced financial planning, ultimately leading to enhanced financial stability and well-being. Our project showcases the collective endeavour of our team in developing a functional and influential application that improves financial management techniques. Through the utilisation of object-oriented programming (OOP) principles, we have created a codebase that is easy to maintain and expand. This ensures that the application remains up-to-date and valuable as user requirements change, ultimately fostering financial literacy and stability.