Invoice Generation Application

Harish S

Introduction The Invoice Generation Application

Invoice Generation Application

- The Simple Invoice Generation Application is designed to simplify the process of creating and managing invoices for individual users.
- This project comprises two main components: a native Android client and a RESTful API backend developed using Spring Boot.

Features

1. User Registration

Users must register with a unique username, password, and email.
Additional information collected includes the date of account creation.

2. Invoice Management

Registered users can create multiple invoices, each linked to a user account.

Invoice creation requires the client name (minimum 5 characters), invoice amount (positive and at least INR 3000), invoice date (past or current date), and a description (optional).
All fields except the description are mandatory.

3. Dashboard

Upon login, users see a dashboard displaying a list of invoices they have created.

4. Invoice Details View:

Clicking an invoice in the dashboard opens a detailed view showing all relevant information about the invoice.

5. Invoice Editing:

Users can edit existing invoices, modifying the client name, amount, date, and description.

6. Invoice Deletion:

Users can delete invoices after confirming the action through a dialog prompt.

Project Rationale

This project is developed as a capstone project to help us to gain hands-on experience in building full-stack applications.

It aims to:

Provide practical experience in Android app development using Kotlin.

Teach the development of RESTful APIs using Spring Boot in Java.

Illustrate the integration of frontend and backend systems.

Demonstrate best practices in validation, error handling, and user interface design.

Technologies Used

Frontend: Android application developed using Kotlin.

Backend: RESTful API developed using Spring Boot (Java).

Backend Development with Spring Boot

Spring Boot Project:

Go to [Spring Initializr](https://start.spring.io/).

Project: Maven **Language:** Java **Spring Boot:** 3.2.5

Fill in the project metadata:

Group:	com.example
Artefact:	invoice-app
Name:	invoice-app
Description:	Simple Invoice Generation Application
Package name:	com.example.invoiceapp
Packaging:	Jar
Java:	17

Dependencies:

Spring Web
Spring Data JPA
MySQL Driver
Spring Boot DevTools
Lombok, H2 Database

Backend Architecture:

Authentication:

User authentication is managed using Spring Security.

Upon registration, user credentials are stored securely using passwords.

During login, users are authenticated.

System and Hardware Requirements

System Requirements for backend server

Operating System:
Windows 10
macOS

Java Development Kit (JDK):

JDK 11 or later

Database:

MySQL
H2 Database (for development and testing purposes)

System and Hardware Requirements

Tools:
IDE:Eclipse,
Postman (for API testing)

Android Client - Operating System:
Windows
macOS

Android Studio: Android Studio
Android SDK: Android SDK 29
Emulator: Android Emulator

RESTful API Development

Number of REST APIs Needed:

User: Register User and Login User

Invoice: Create Invoice Update Invoice
Delete Invoice Get All Invoices
Get Invoice Details
In total, 7 REST APIs are needed.

API Endpoints:

User Registration: `POST /api/register`
User Login: `POST /api/login`
Invoice Creation: `POST /api/invoices
Get Invoices: `GET /api/invoices`
Get Invoice Details: `GET /api/invoices/{id}`
Update Invoice: `PUT /api/invoices/{id}`
Delete Invoice: `DELETE /api/invoices/{id}`

Frontend Architecture:

Android Project: Name Of project as "**Invoice Generation Application**".
language as **Kotlin**.

User Interface:

Login and Registration Screens: Allow users to register and log in.

Dashboard: Displays a list of invoices.

Invoice Detail View: Shows detailed information about a selected invoice.

Invoice Form: Used for creating and editing invoices.

App Workflow:

---

1. **User Registration and Login:**

Users register by providing their username, password, and email.

After registration, they can log in..

2. **Invoice Management:**

Users can create new invoices by entering the required details.

The dashboard shows all invoices created by the logged-in user.

Users can view, edit, or delete invoices from the dashboard.

Communication with Backend:

JSON Passing:

Data is exchanged between the Android app and the backend using JSON.

Retrofit library is used on the Android side to make API calls.

GSON is used to parse JSON responses into Kotlin data classes.

Data Validation and Error Handling

- Both Android and Spring Boot implement data validation and error handling to ensure data inputs.
- Spring Boot API returns error responses in case of validation failures or other errors, which are handled by the Android application.

Entity-Relationship (ER) diagram for your Simple Invoice Generation Application:

User table/entity:

id: int (Primary Key)

username: String
password: String

email: String

createdAt: LocalDateTime

Invoice table/entity:

id: Long (Primary Key)

amount: Double

date: LocalDate
description: String

clientName: String

user_id: int (Foreign Key referencing User table)

One user can have multiple invoices (One-to-Many relationship between User and Invoice).

Backend Setup:

Environment: Java, Spring Boot, and database (MySQL).

Project Initialisation: Used Spring Initializr to create a new Spring Boot project with dependencies such as Spring Web, Spring Data JPA, and Spring Security.

Database Configuration: Set up the database schema and connection properties in `application.properties`.

Security Configuration: Implement user authentication and authorisation using Spring Security.

API Development: Define entities, repositories, and controllers for user and invoice management.

Run the Backend: Use `spring-boot:run` to start the backend server.

Frontend Setup

Environment: Install Android Studio.

Project Initialisation: Create a new Android project with Kotlin as the language.

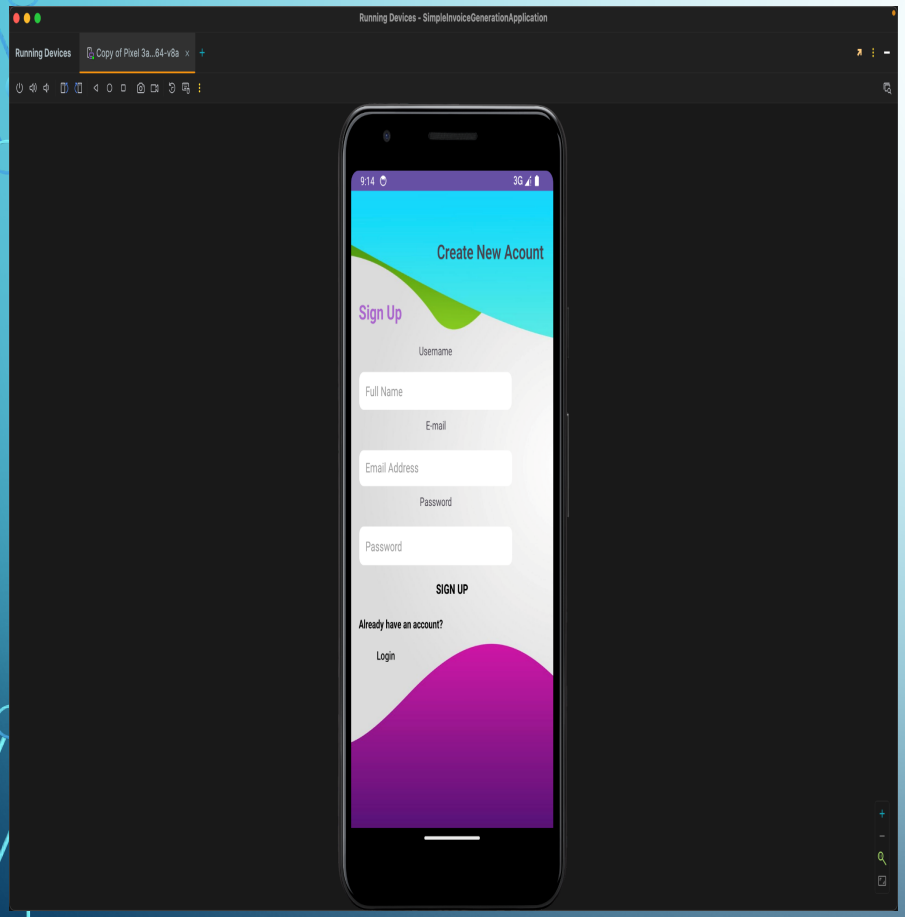
UI Design: Design the UI using XML layouts and implement the necessary Activities and Fragments.

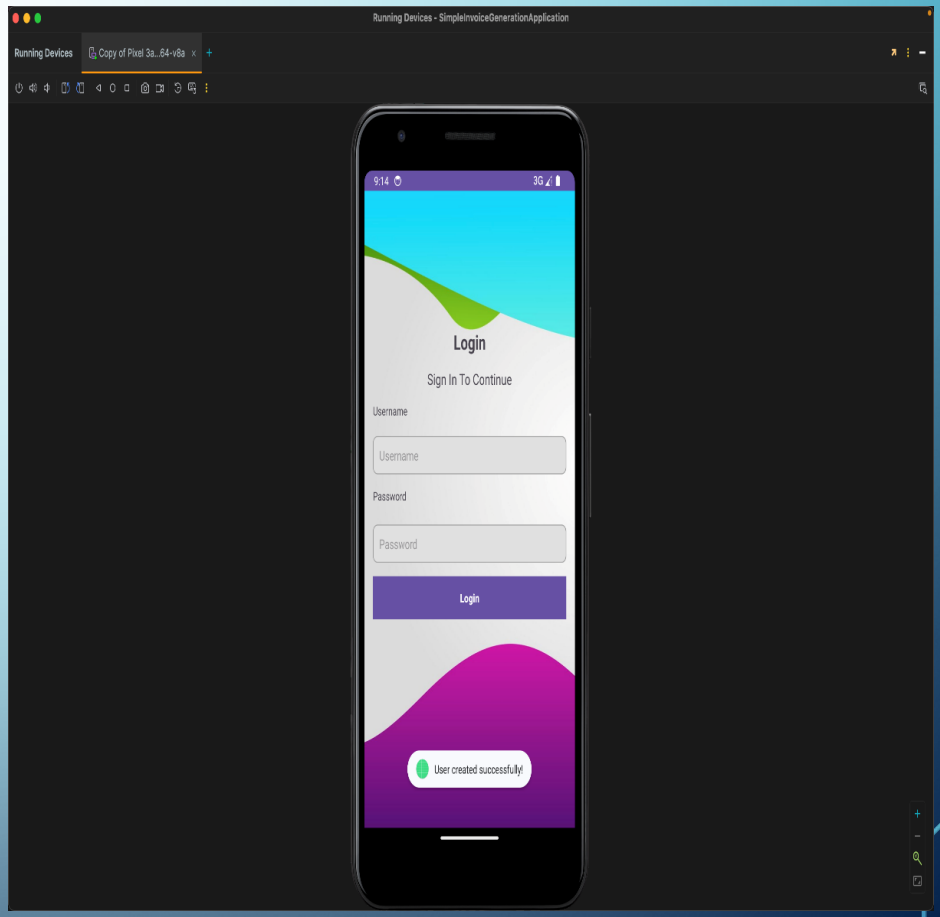
Networking: Set up Retrofit for API calls and GSON for JSON parsing.

Validation: Add form validation to ensure user inputs are correct before sending data to the backend.

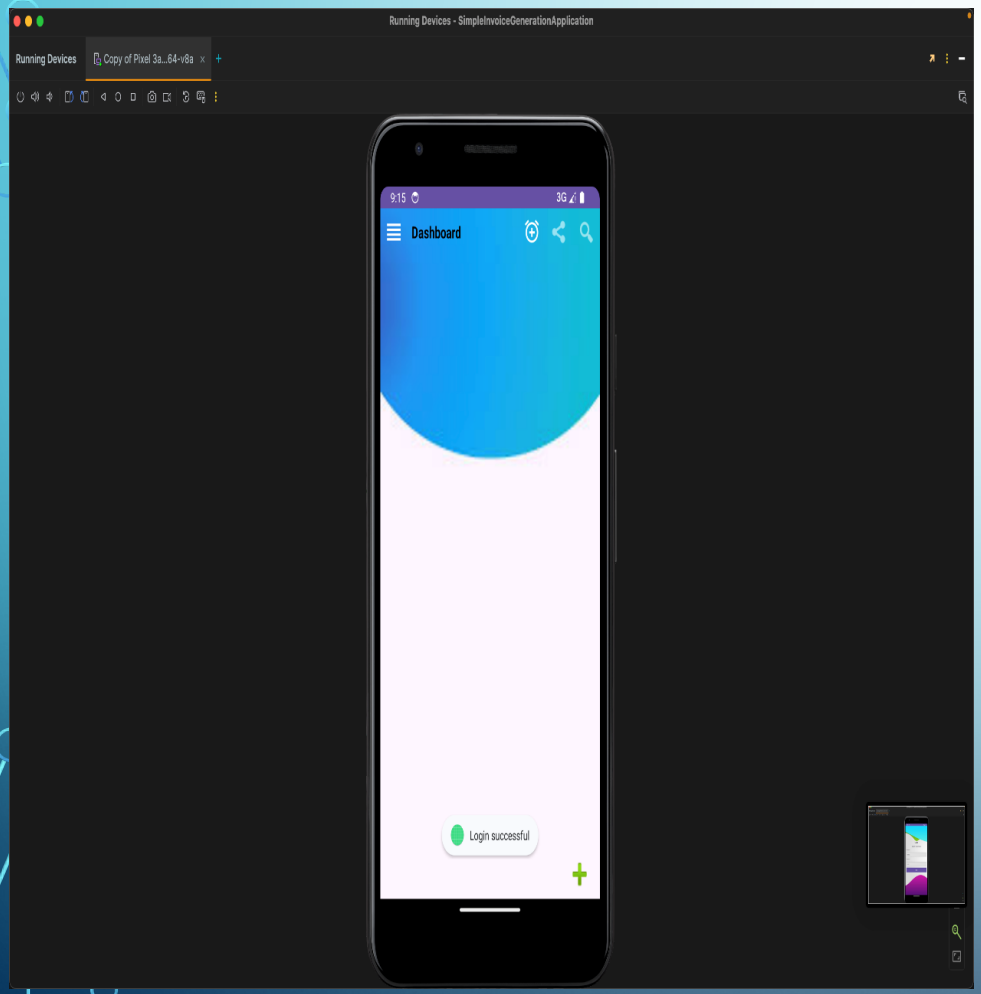
Run the App: Build and run the app on an emulator or physical device.

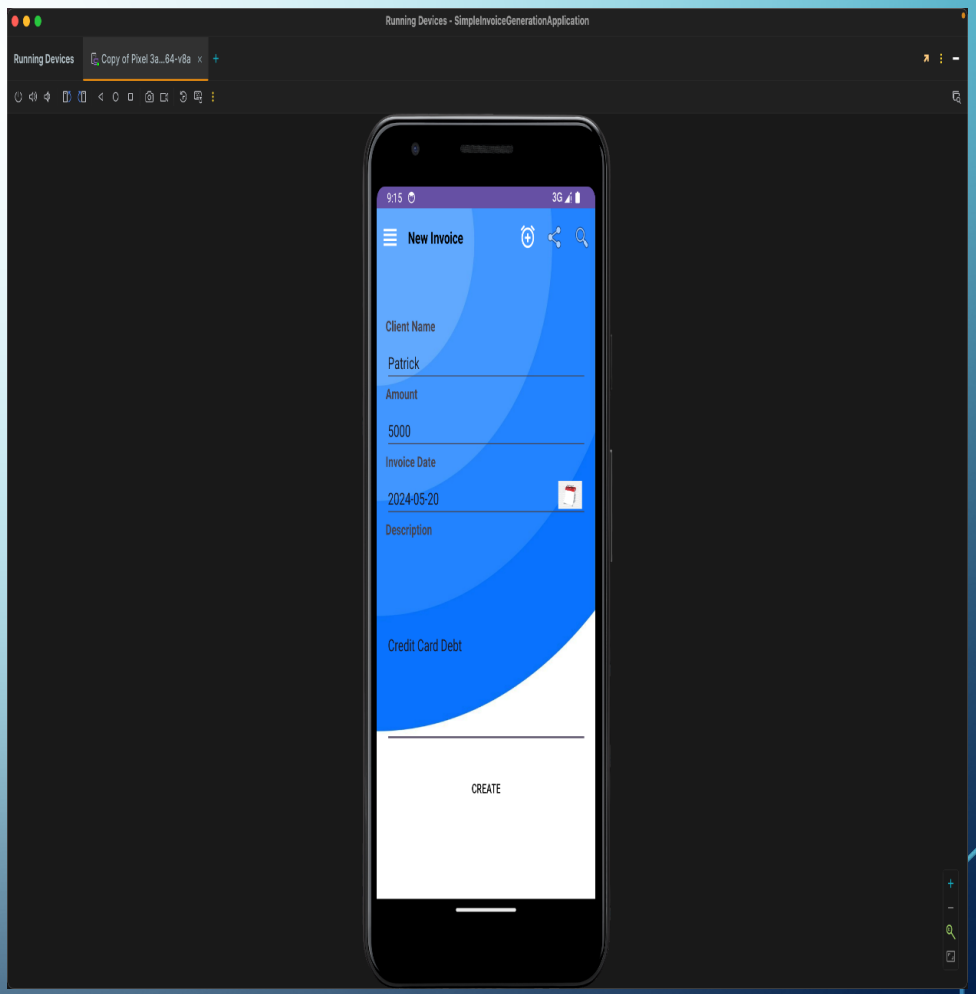
Application Review



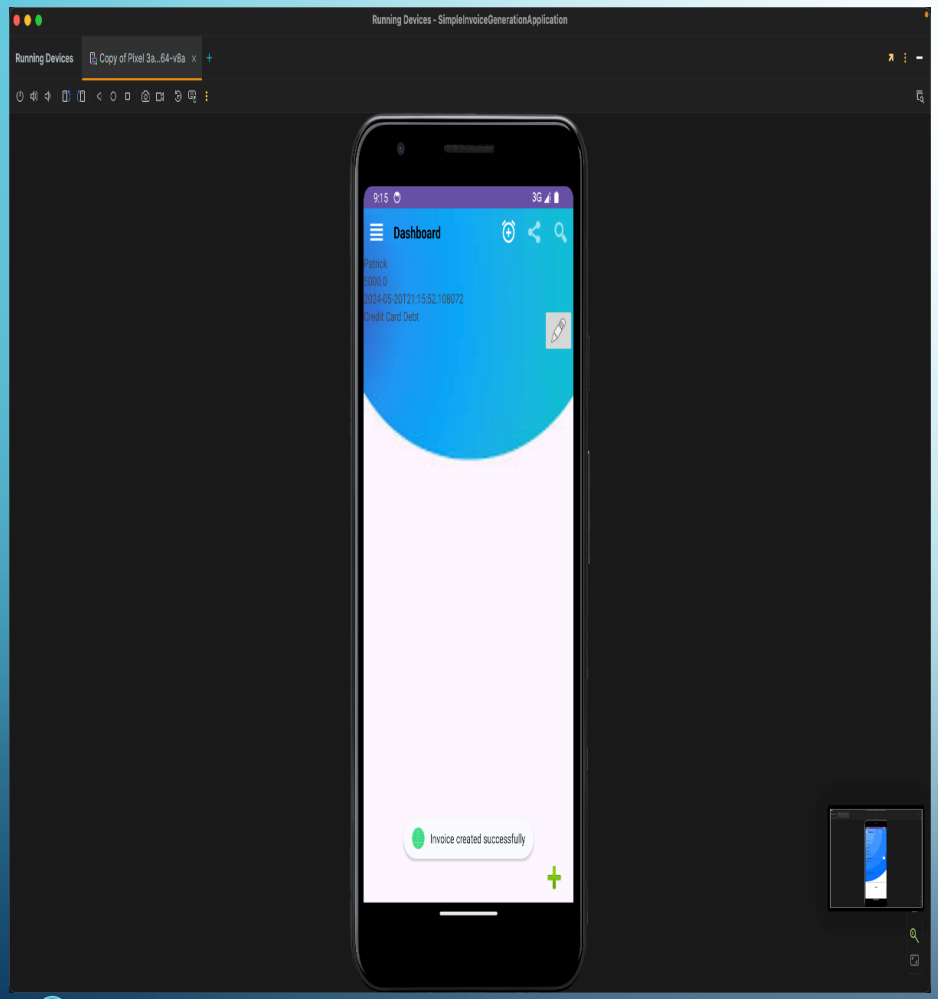


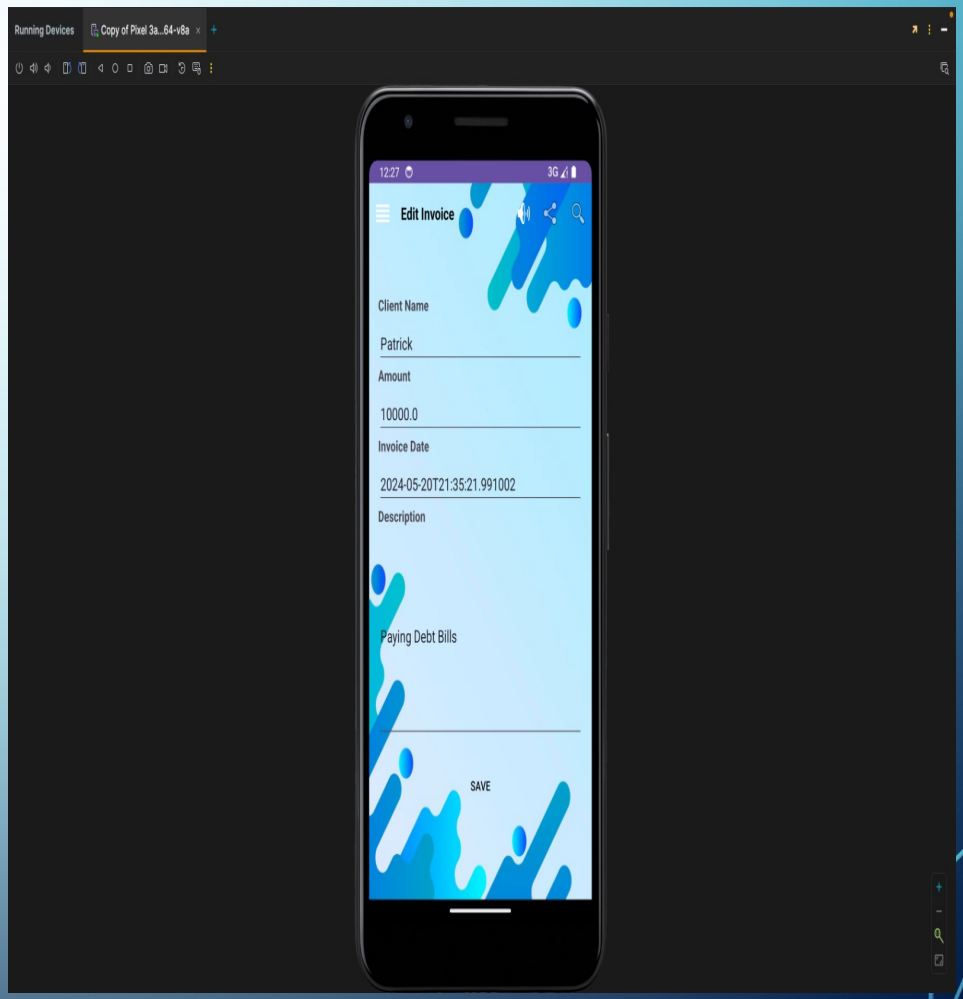
Application Review





Application Review





Application Review