A REPORT OF SIX MONTH TRAINING

at

VProtech Digital

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Information Technology)

Jan-May, 2023

SUBMITTED BY:

Ekam Preet Singh

1905327

DEPARTMENT OF INFORMATION TECHNOLOGY

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

(An Autonomous College Under UGC ACT)

A REPORT OF SIX MONTH TRAINING

at

VProtech Digital

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Information Technology)

Jan-May, 2023

SUBMITTED BY:

Ekam Preet Singh

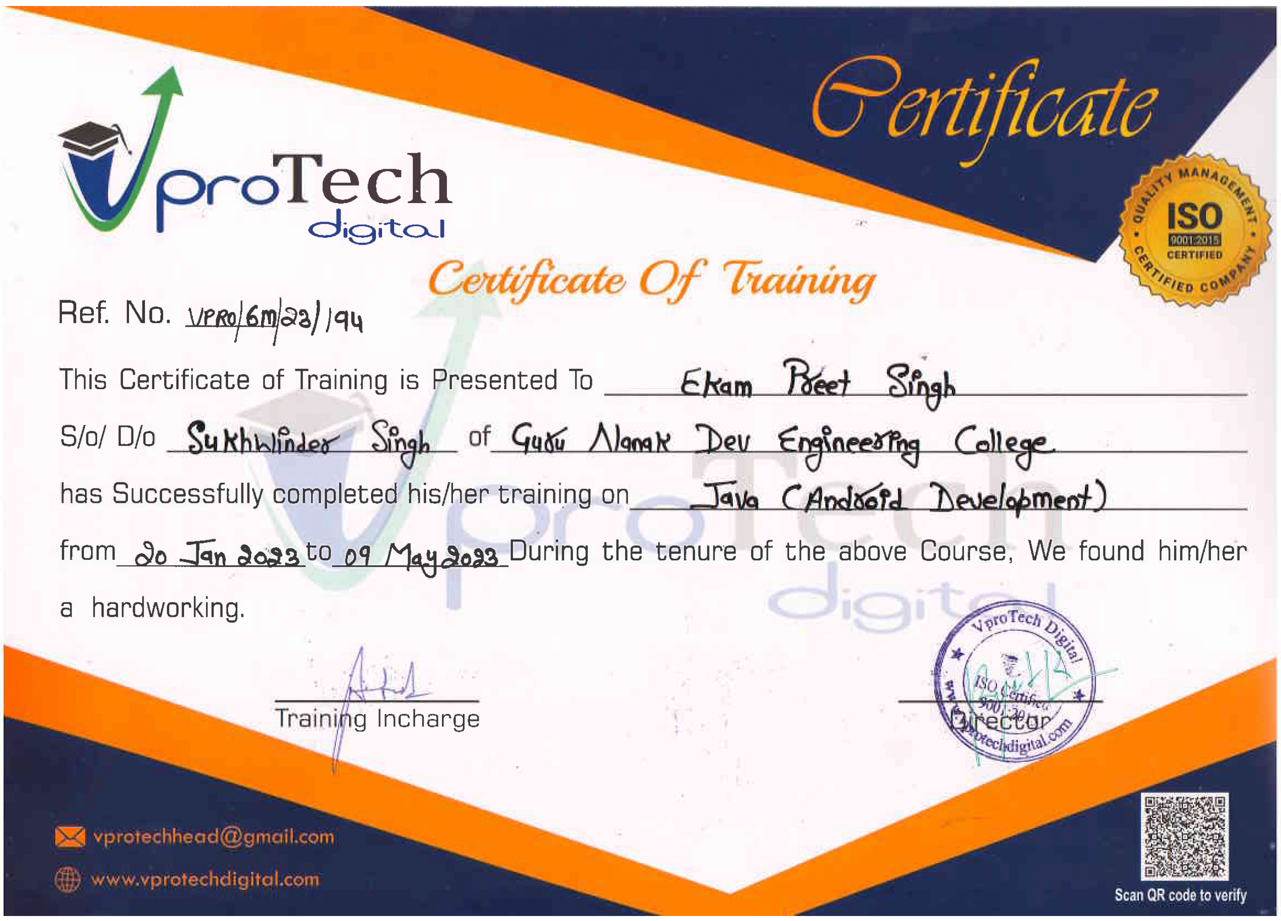
1905327

DEPARTMENT OF INFORMATION TECHNOLOGY

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

(An Autonomous College Under UGC ACT)

CERTIFICATE



VproTech digital

Certificate

ISO 9001:2015 CERTIFIED · QUALITY MANAGEMENT · CERTIFIED COMPANY

Certificate Of Training

Ref. No. VPRO/6M/23/194

This Certificate of Training is Presented To Ekam Preet Singh

S/o/ D/o Sukhwinder Singh of Guru Nanak Dev Engineering College

has Successfully completed his/her training on Java (Android Development)

from 20 Jan 2023 to 09 May 2023 During the tenure of the above Course, We found him/her a hardworking.

[illegible]
Training Incharge

VproTech Digital ISO Certified 9001:2015 www.vprotechdigital.com
[illegible]
Director

vprotechhead@gmail.com

www.vprotechdigital.com

Scan QR code to verify

STUDENT'S DECLARATION

I hereby certify that the work which is being presented in this project report with the project entitled 'Invoice Pro' by Ekam Preet Singh, University Roll No. 1905327 in partial fulfillment of requirements for the award of degree of B.Tech. (Information Technology) submitted in the Department of Information Technology at **GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA under I.K. GUJRAL PUNJAB TECHNICAL UNIVERSITY** is an authentic record of my own work carried out under the supervision of Ms. Mamta Panwar, VProtech Digital. The matter presented has not been submitted by me in any other University / Institute for the award of B.Tech. Degree.

Student Name: Ekam Preet Singh
Univ. Roll No. 1905327

(Signature of Student)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of Internal Examiner

The External Viva-Voce Examination of the student has been held on 13.06.2023.

Signature of External Examiner

Signature of HOD

ABSTRACT

This report presents the development of 'Invoice Pro', an Android application designed to streamline bill management processes and integrate Goods and Services Tax (GST) functionality. The application offers user-friendly interfaces that automate GST tax calculations and provide comprehensive features for bill creation, tracking, sorting, filtering, and data management.

The objective of the 'Invoice Pro' project was to create an efficient solution for businesses and individuals to manage their bills effectively while incorporating the complexities of GST tax calculations. Leveraging the capabilities of the Android platform, the application was developed using industry-standard programming languages and frameworks. The user interfaces were meticulously designed to ensure ease of use and intuitive navigation, enabling users to perform various tasks effortlessly.

With 'Invoice Pro', users can create, track, and manage their bills with convenience and accuracy. The application automates GST tax calculations, reducing manual errors and saving time for accurate tax assessments. Additionally, advanced features such as sorting, filtering, and comprehensive data management empower users with enhanced control and organization of their bill-related information.

ACKNOWLEDGEMENT

I am highly grateful to the Dr. Sehijpal Singh , Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out Six Months industrial training at Vprotech Digital.

The constant guidance and encouragement received from Dr. K. S. Mann, Head of the Department, GNDEC Ludhiana has been of great help during the training and project work and is acknowledged with reverential thanks.

I would like to express a deep sense of gratitude and thanks profusely to Mr. Rajat Kumar, Director . Without the wise counsel and able guidance, it would have been impossible to complete the report in this manner.

The help rendered by Ms. Mamta Panwar, VProtech Digital for experimentation is greatly acknowledged.

Finally, I am indebted to all whosoever have contributed in this report work and friendly stay at VProtech Digital.

Ekam Preet Singh (1905327)

ABOUT THE INSTITUTE

VProtech Digital, founded in March 2017 by Rajat Kumar, is a leading provider of industrial training and digital solutions based in Sahibzada Ajit Singh Nagar, Punjab. With a strong commitment to technical excellence and professional development, it specializes in offering comprehensive training programs to B.Tech, Diploma, BCA, and MCA students.

The team of experienced trainers delivers high-quality training in various domains, including web development, web design, Android applications, SEO, social media marketing (SMM), and digital marketing. Through our rigorous six-month/6-week industrial training program in Mohali, Chandigarh, equip students with the skills and knowledge needed to excel in the ever-evolving digital landscape.

In addition to the training services, it excel in creating impactful virtual branding strategies and developing mobile applications for both Android and iPhone devices. There expertise extends to designing responsive websites that ensure optimal user experience across all platforms. We also provide assistance in online business setup and promotion, leveraging social media platforms for maximum reach and engagement.

With it's unwavering commitment to helping students and professionals succeed, VProtech Digital is your trusted partner in bridging the gap between education and industry requirements.

List of Figures

Table of Contents

1 Introduction

1.1 Introduction to Organization

VProtech Digital, founded in March 2017 by Rajat Kumar, is a leading provider of industrial training and digital solutions based in Sahibzada Ajit Singh Nagar, Punjab. With a strong commitment to technical excellence and professional development, it specializes in offering comprehensive training programs to B.Tech, Diploma, BCA, and MCA students.

The team of experienced trainers delivers high-quality training in various domains, including web development, web design, Android applications, SEO, social media marketing (SMM), and digital marketing. Through our rigorous six-month/6-week industrial training program in Mohali, Chandigarh, equip students with the skills and knowledge needed to excel in the ever-evolving digital landscape.

In addition to the training services, it excel in creating impactful virtual branding strategies and developing mobile applications for both Android and iPhone devices. There expertise extends to designing responsive websites that ensure optimal user experience across all platforms. We also provide assistance in online business setup and promotion, leveraging social media platforms for maximum reach and engagement.

With it's unwavering commitment to helping students and professionals succeed, VProtech Digital is your trusted partner in bridging the gap between education and industry requirements.

Figure 1.1: *Company Logo*

1.2 Introduction to Android Development

Android development refers to the process of creating application software specifically for the Android operating system. Android applications, commonly known as Android apps, are designed to be accessed and used on Android devices such as smartphones and tablets. These apps are delivered through the Google Play Store and can be accessed by users with an active internet connection.

Android apps offer a wide range of functionalities and can be developed for various purposes. They can include email clients, productivity tools, gaming apps, e-commerce platforms, and much more. Android apps are built using programming languages like Java or Kotlin, along with the Android SDK (Software Development Kit), which provides a set of tools and libraries for app development.

In Android development, the focus is on creating the user interface (UI) and user experience (UX) of the app, as well as implementing features and functionalities specific to the Android platform. Developers utilize Java or Kotlin for coding the app's front-end, while server-side programming languages like JavaScript, Python, or Java are used for creating the back-end scripts and handling database management.

Figure 1.2: *Android*

Figure 1.3: *Android Studio*

1.3 Objectives

1. Develop an efficient Android application for bill management with integrated GST tax functionality.
2. Create user-friendly interfaces for automating GST tax calculations and comprehensive bill management features.
3. Provide seamless bill creation, tracking, sorting, filtering, and data management capabilities in the Invoice Pro Android application.

1.4 Need to Learn Android Development

1. Market Demand and Career Opportunities: Android is the dominant mobile operating system worldwide, with a massive user base. Learning Android development opens up a wide range of career opportunities in the tech industry, as there is a high demand for skilled Android developers.
2. Versatility and Market Reach: Android powers a variety of devices, including smartphones, tablets, smart TVs, wearables, and IoT devices. By learning Android development, you gain the ability to create apps that reach a broad audience and cater to different device types and form factors.
3. Entrepreneurial Opportunities: Android development provides a platform for entrepreneurship. You can develop and launch your own apps on the Google Play Store, monetize them through ads or in-app purchases, and potentially generate income. It offers the freedom to pursue your own ideas and start your own app-based business.
4. Extensive Development Resources: Android has a large and active development community, with abundant resources, libraries, and frameworks available. You can leverage these resources to speed up development, solve problems, and learn from others' experiences. The community support and documentation make the learning process smoother.
5. Constant Learning and Innovation: Android development is a dynamic field, with regular updates, new features, and emerging technologies. By learning Android, you become part of an ecosystem that encourages continuous learning and innovation. It keeps you engaged and challenges you to stay updated with the latest trends and best practices in mobile app development.

6. Creative Expression and Problem Solving: Android development allows you to bring your creative ideas to life through mobile apps. You can design user interfaces, create interactive experiences, and solve real-world problems through app development. It combines technical skills with creativity, offering a fulfilling and rewarding avenue for expression and problem-solving.

1.5 Introduction of Basic Concepts

1. Activity: An Activity represents a single screen or user interface in an Android application. It is responsible for managing the user interaction and UI components. Activities are the building blocks of an Android app and are crucial for navigation and user flow.

2. Layouts: Layouts define the structure and appearance of the user interface. Android provides a variety of layout options such as LinearLayout, RelativeLayout, ConstraintLayout, etc., to arrange UI elements in a specific manner. XML files are used to define the layout structure, and Java code is used to interact with the UI elements.

3. Views and Widgets: Views are UI elements that are used to interact with the user. Examples of views include buttons, text fields, images, lists, etc. These views are referred to as widgets in Android development. Widgets can be added to the layout files and then accessed and manipulated using Java code.

4. Intents: Intents are a fundamental component of Android development for communication between different components of an application or even between different applications. They are used to start activities, launch services, broadcast messages, and facilitate inter-app communication.

5. Resources: Android provides a resource system that allows you to separate the application's code and UI elements from the content and assets. Resources include strings, colors, dimensions, images, layouts, and more. By utilizing resources, you can easily adapt your app to different devices and translations.

6. Data Storage: Android provides different mechanisms for data storage, including SQLite databases, SharedPreferences for storing key-value pairs, and file storage. These storage options allow you to persist and retrieve data in your application.

7. Event Handling: Event handling is essential for capturing and responding to user interactions. Android uses event listeners and callback methods to handle events such as button clicks, touch events, gestures, etc. This allows you to create interactive and responsive user experiences.

1.6 Overview of Android Application Architecture

Android application architecture refers to the structure and organization of an Android app's codebase, components, and interactions. It provides a systematic approach to developing and maintaining Android applications. Here is an overview of the typical Android application architecture:

1. User Interface (UI) Layer:
    - Activities: Activities represent the screens or UI components visible to the user. They handle user interactions, manage UI elements, and navigate between different screens.
    - Fragments: Fragments are reusable UI components within an activity that can be dynamically added or removed. They allow for modular UI design and support flexible screen layouts.
    - Views and ViewModels: Views are UI elements, such as buttons and text fields, while ViewModels hold the UI-related data. The ViewModel separates the UI logic from the UI components, enabling better code organization and testing.
2. Data Layer:
    - Model: The model represents the data and business logic of the application. It includes data structures, classes, and repositories for handling data retrieval, storage, and manipulation.
    - Networking: Android apps often interact with remote servers or web services. Networking components, such as HTTP clients and API interfaces, handle data retrieval and sending requests.
    - Persistence: Android provides various data storage options, such as SQLite databases, SharedPreferences, and file storage. These allow data to be stored locally and retrieved when needed.
3. Architecture Patterns:
    - Model-View-ViewModel (MVVM): MVVM is a widely used architecture pattern in Android. It separates the UI logic (View) from the data and business logic (Model) through a ViewModel. It promotes testability, maintainability, and separation of concerns.
    - Model-View-Presenter (MVP): MVP separates the UI (View) from the data (Model) using a Presenter as an intermediary. It helps in achieving a clean separation between UI and business logic.

- Clean Architecture: Clean Architecture emphasizes separation of concerns and independence of the components. It defines layers (such as presentation, domain, and data) and enforces clear boundaries between them.

4. Libraries and Frameworks:

   - Android Jetpack: Jetpack is a collection of libraries and tools provided by Google to accelerate Android app development. It includes components like LiveData, ViewModel, Room, and Navigation, which simplify common tasks and provide architectural guidance.
   - Retrofit: Retrofit is a popular HTTP client library for making network requests and handling API interactions.
   - Dagger/Hilt: These dependency injection frameworks help manage object dependencies and improve code maintainability and testability.

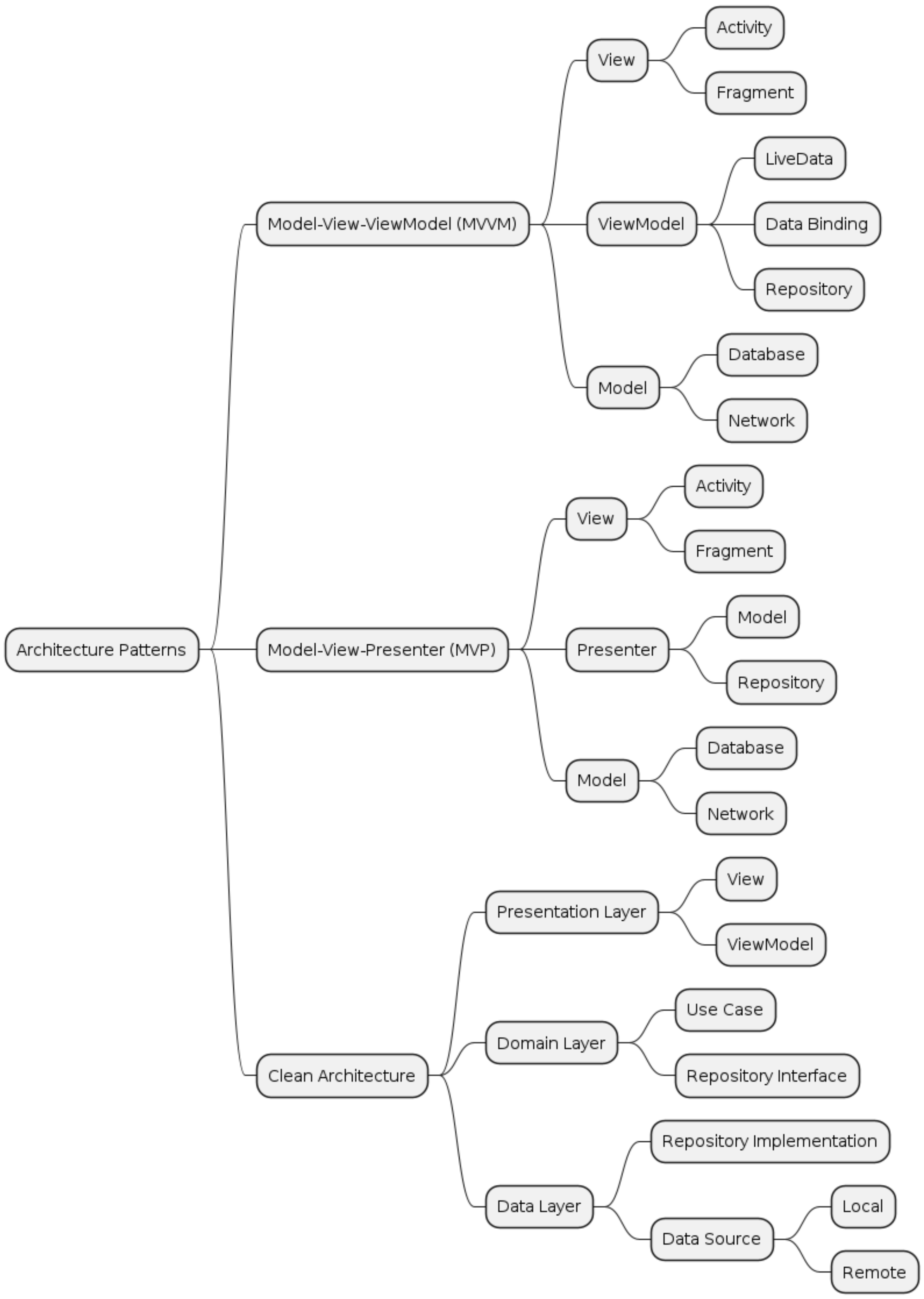


Figure 1.4: *Architecture*

2 Training Work Undertaken

2.1 Android Development - Detail Overview

Android development is the process of creating applications for the Android operating system, which is used on various mobile devices such as smartphones and tablets. It involves designing, coding, testing, and deploying Android applications using the Java or Kotlin programming languages.

1. Android Platform:
   - Android OS: Android is an open-source operating system developed by Google. It provides a rich set of features, libraries, and APIs for building mobile applications.
   - Android SDK: The Android Software Development Kit (SDK) includes tools, libraries, and resources necessary for developing Android applications.
   - Android Studio: Android Studio is the official integrated development environment (IDE) for Android development. It provides a powerful set of tools and features for coding, debugging, and testing Android apps.
2. User Interface (UI):
   - Activities and Fragments: Activities represent individual screens in an Android app, while fragments are reusable UI components that can be combined to create flexible layouts.
   - Views and Layouts: Android provides a wide range of UI elements called views, such as buttons, text fields, and images. Views are organized using layouts, such as LinearLayout or ConstraintLayout, to define the visual structure of the app.
   - Material Design: Google's Material Design guidelines provide a set of UI principles and components for creating modern and visually appealing Android apps.

3. Android Application Components:

    - Activities: Activities are the entry points of an app and represent a single screen with a user interface.
    - Services: Services are background processes that perform tasks without a user interface, such as playing music in the background or fetching data from a server.
    - Broadcast Receivers: Broadcast receivers listen for system-wide or app-specific events and allow apps to respond to them.
    - Content Providers: Content providers enable sharing data between different apps or accessing system data, such as contacts or media files.

4. Data Management:

    - SQLite Database: Android provides built-in support for SQLite databases, allowing apps to store and retrieve structured data.
    - SharedPreferences: SharedPreferences allow storing simple key-value pairs for persistent app settings.
    - Network Communication: Android apps often interact with remote servers using HTTP requests. Libraries like Retrofit or Volley facilitate network communication.

5. Android Development Languages:

    - Java: Java is the traditional programming language used for Android development. It offers a robust set of libraries and has a large developer community.
    - Kotlin: Kotlin is a modern programming language officially supported by Google for Android development. It offers concise syntax, null safety, and interoperability with Java.

6. Testing and Deployment:

    - Unit Testing: Android apps can be tested using frameworks like JUnit and Mockito for unit testing individual components.
    - Instrumentation Testing: Instrumentation tests verify app behavior using simulated user interactions and can be written using the Espresso framework.
    - Deployment: Android apps are typically distributed through the Google Play Store or can be manually installed on devices.

2.2 Android Application Structure

1. App Level:

   - build.gradle: This file contains the configuration and dependencies for the entire application. It defines the build settings, dependencies, and other project-specific configurations.
   - proguard-rules.pro: ProGuard rules file that specifies how to obfuscate and optimize the code during the build process.
   - settings.gradle: This file includes the module settings for the application.

2. Module Level:

   - build.gradle: This file contains the module-level configuration, such as dependencies, build types, flavors, and other specific settings for the module.
   - AndroidManifest.xml: The Android manifest file describes essential information about the application, including package name, activities, services, permissions, and more.

3. Java Packages:

   - app package: This package typically contains the main Java code for the application.
   - MainActivity.java: The main entry point of the application. It represents the initial screen and handles user interactions.
   - Other activity classes: Additional activity classes representing different screens and user interactions.
   - Adapter classes: Classes for handling data binding between views and data sources, such as RecyclerView adapters.
   - Utility classes: Helper classes for performing common tasks or functionalities used across the application.
   - Service classes: Classes for implementing background services.
   - BroadcastReceiver classes: Classes for handling broadcast events.
   - model package: This package contains classes representing the data model of the application.
   - network package: Classes related to network communication, such as APIs, network clients, and data parsing.
   - database package: Classes for handling database operations, including SQLiteOpenHelper, DAOs, and database contracts.

4. Resources:
    - res/layout: XML layout files defining the user interface for different activities and views.
    - res/values: XML files for storing string constants, colors, dimensions, styles, and other resource values.
    - res/drawable: Image and drawable resources used in the application.
    - res/mipmap: App icons for different densities.
    - res/menu: XML files defining menus for options and context menus.
    - res/anim: Animation resources.
    - res/drawable: XML files defining vector drawables.
5. Gradle Files:
    - build.gradle (Module): Contains module-specific dependencies, build configurations, and other settings.
    - build.gradle (Project): Contains project-wide build configurations, such as build tools version, repositories, and other global settings.

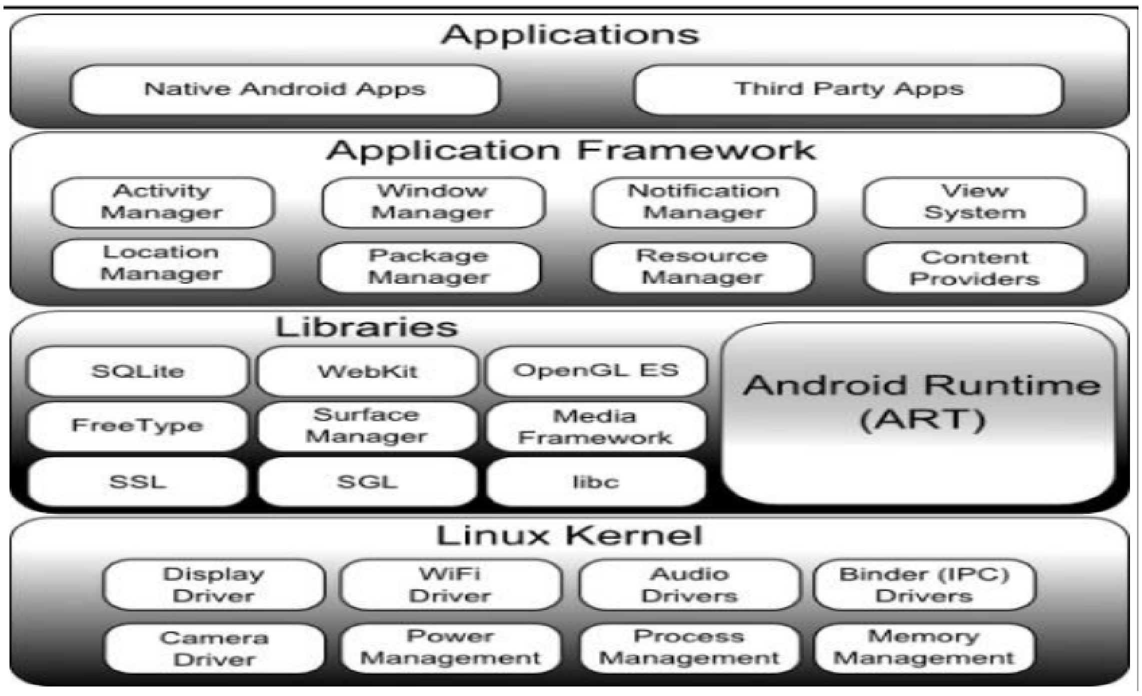


Figure 2.1: *Framework*

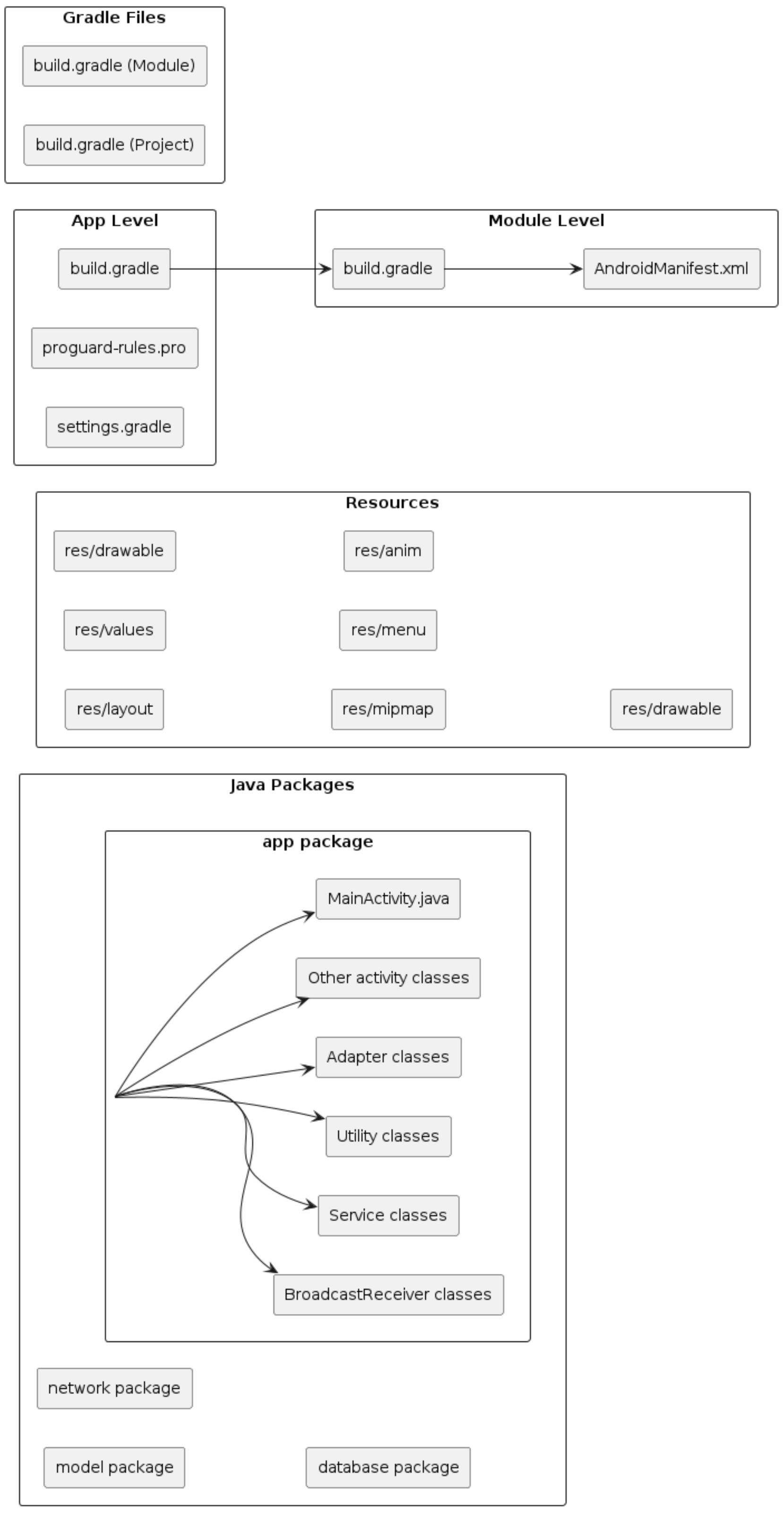


Figure 2.2: *Structure*

2.3 Learning Methodology

1. Understand the Basics:
    - Start by familiarizing yourself with the fundamentals of Java programming language. Learn about variables, data types, control structures, functions, and object-oriented programming (OOP) concepts like classes, objects, and inheritance.
2. Setup Development Environment:
    - Install Android Studio, the official IDE for Android development.
    - Set up the necessary SDKs, emulators, and virtual devices for testing your applications.
3. Learn Android Components:
    - Study the core components of Android applications, such as activities, fragments, services, content providers, and broadcast receivers.
    - Understand the Android application lifecycle and how these components interact with each other.
4. User Interface Design:
    - Learn XML layout files and how to create user interfaces using Views, ViewGroups, and layouts.
    - Explore different UI components like buttons, text views, image views, and more.
    - Understand how to handle user input and events.
5. Data Management:
    - Study how to use SQLite database for data storage in Android applications.
    - Learn about CRUD (Create, Read, Update, Delete) operations and how to interact with the database using SQLiteOpenHelper and SQLite APIs.
6. Networking and Web Services:
    - Explore how to make HTTP requests, handle JSON/XML parsing, and integrate web services into your Android applications using libraries like Retrofit or Volley.
    - Understand how to handle network connectivity and asynchronous operations.

7. User Interaction and Navigation:
    - Learn how to handle user interactions, such as button clicks, menu selections, and gestures.
    - Explore navigation patterns like using navigation drawers, tabs, and bottom navigation for seamless user experience.
8. Multimedia and Device Features:
    - Understand how to work with multimedia elements like images, audio, and video.
    - Explore accessing device features like camera, location, sensors, and permissions.
9. Testing and Debugging:
    - Learn how to write unit tests for your Android code using JUnit or Espresso.
    - Understand how to debug and troubleshoot issues using Android Studio's debugging tools.
10. Publish Your App:
    - Gain knowledge about the app publishing process on the Google Play Store.
    - Learn about app signing, store listings, screenshots, and app descriptions.
    - Understand the importance of app optimization, performance, and security.
11. Continuous Learning:
    - Stay updated with the latest Android development trends, new releases, and best practices.
    - Join developer communities, forums, and attend Android-related events or meetups to network and learn from others.

2.4 Project Setup

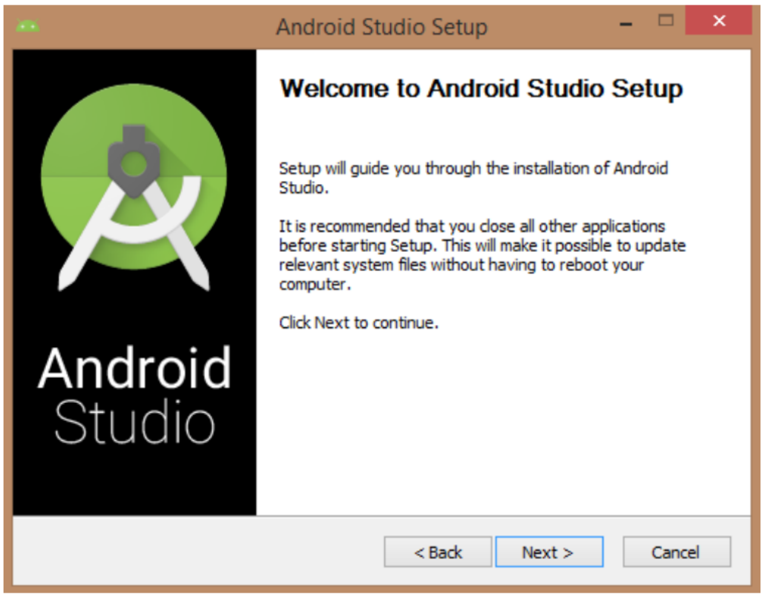
2.4.1 Installing and Running Applications on Android Studio

1. System Requirements: The required tools to develop Android applications are open source and can be downloaded from the Web. Following is the list of software's you will need before you start your Android application programming.

   - Java JDK or later version
   - Java Runtime Environment (JRE)
   - Android Studio

2. Setup Android Studio: Android Studio is the official IDE for android application development.It works based on IntelliJ IDEA, You can download the latest version of android studio from Android Studio 2.2 Download, If you are new to installing Android Studio on windows,you will find a file, which is named as android-studiobundle-143.3101438-windows.exe.So just download and run on windows machine according to android studio wizard guideline.

3. So let's launch Android Studio.exe,Make sure before launch Android Studio, Our Machine should required installed Java JDK. To install Java JDK,take a references of Android environment setup


Android Studio Setup

Welcome to Android Studio Setup

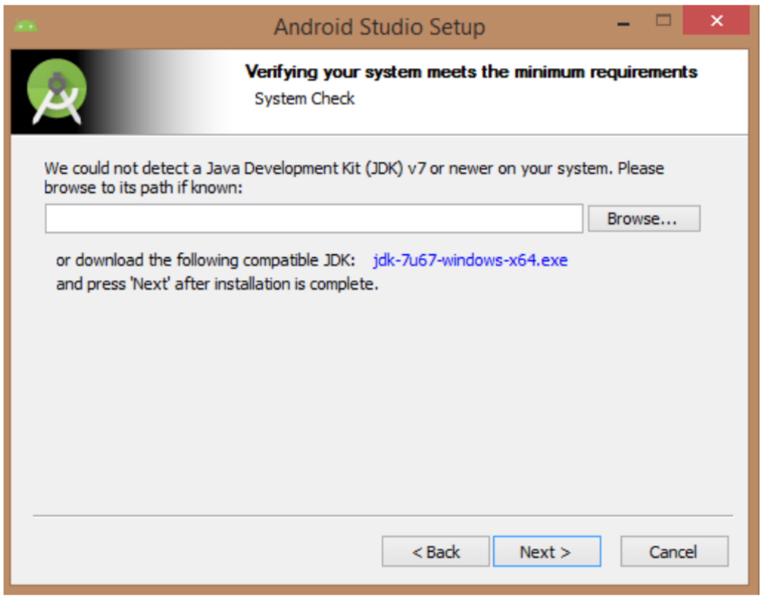
Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 2.3: *Setup*

4. Once you launched Android Studio, its time to mention JDK7 path or later version in android studio installer.


Android Studio Setup

Verifying your system meets the minimum requirements

System Check

We could not detect a Java Development Kit (JDK) v7 or newer on your system. Please browse to its path if known:

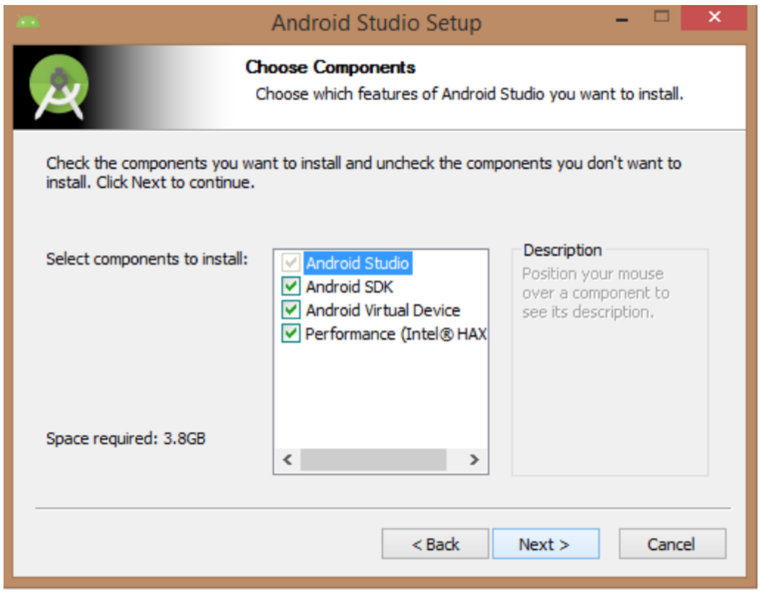
Browse...

or download the following compatible JDK: jdk-7u67-windows-x64.exe
and press 'Next' after installation is complete.

< Back | Next > | Cancel

Figure 2.4: *Downloading*

5. Need to check the components, which are required to create applications, below the image has selected Android Studio, Android SDK, Android Virtual Machine and performance(Intel chip).


Android Studio Setup

Choose Components

Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- ✓ Android Studio
- ✓ Android SDK
- ✓ Android Virtual Device
- ✓ Performance (Intel® HAX

Space required: 3.8GB

Description
Position your mouse over a component to see its description.

< Back | Next > | Cancel

Figure 2.5: *Components*

6. Need to specify the location of local machine path for Android studio and Android SDK, below the image has taken default location of windows 8.1 x64 bit architecture.

7. After done all above steps perfectly, you must get finish button and it gonna be open android studio project with Welcome to android studio message as shown below

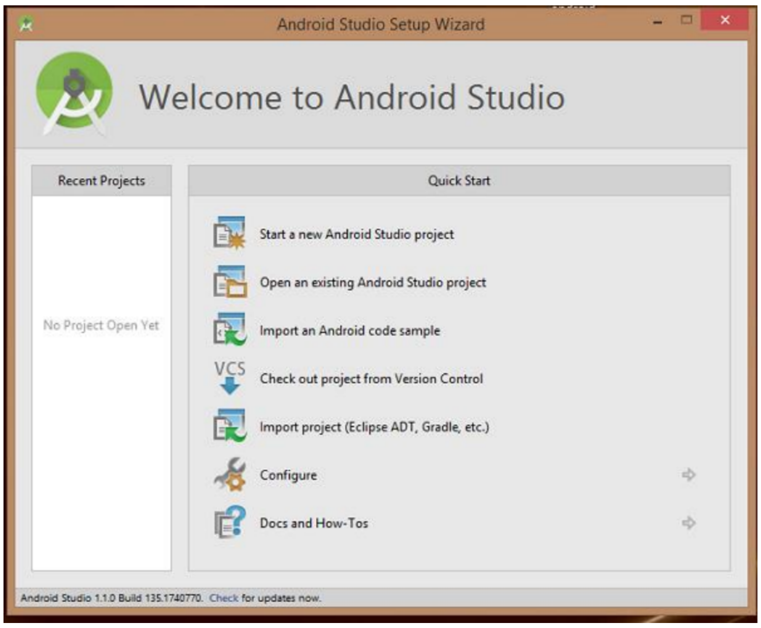

Android Studio Setup Wizard
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos
Android Studio 1.1.0 Build 135.1740770. Check for updates now.

Figure 2.6: *Complete Setup*

2.4.2 Creating an Android Project

1. Open Android Studio if it is not already opened.
2. In the main Welcome to Android Studio window, click Start a new Android Studio project.
3. In the Create Android Project window, enter Hello World for the Application name.
4. Verify that the default Project location is where you want to store your Hello World app and other Android Studio projects, or change it to your preferred directory.
5. Accept the default android.example.com for Company Domain, or create a unique company domain. If you are not planning to publish your app, you can accept the default. Be aware that changing the package name of your app later is extra work.
6. Leave unchecked the options to Include C++ support and Include Java support, and click Next.

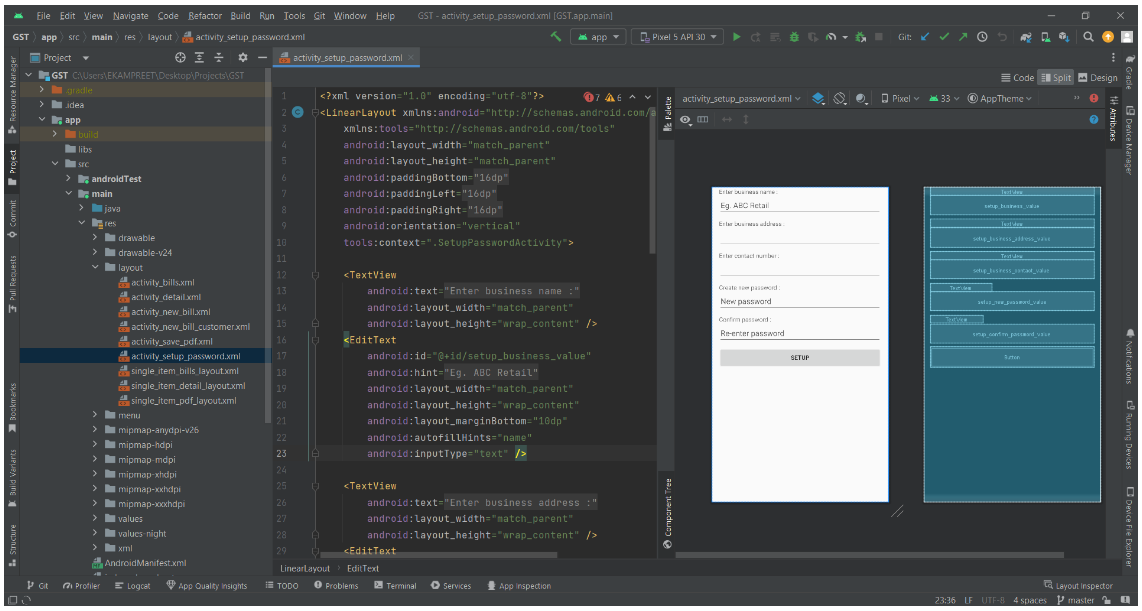
7. On the Target Android Devices screen, Phone and Tablet should be selected. Ensure that API 15: Android 4.0.3 IceCreamSandwich is set to Minimum SDK; if not, use the popup menu to set it.

8. Leave unchecked the Include Instant App support and all other options. Then click Next. If your project requires additional components for your chosen target SDK, Android Studio will install them automatically.

9. The Add an Activity window appears. An Activity is a single, focused thing that the user can do. It is a crucial component of any Android app. An Activity typically has a layout associated with it that defines how UI elements appear on a screen. Android Studio provides Activity templates to help you get started. For the Hello World project, choose Empty Activity as shown below, and click Next.

10. The Configure Activity screen appears (which differs depending on which template you chose in the previous step). By default, the empty Activity provided by the template is named MainActivity. You can change this if you want, but this lesson uses MainActivity.

11. Make sure that the Generate Layout file option is checked. The layout name by default is activitymain. You can change this if you want, but this lesson uses activitymain.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical"
    tools:context=".SetupPasswordActivity">

    <TextView
        android:text="Enter business name :"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/setup_business_value"
        android:hint="Eg. ABC Retail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:autofillHints="name"
        android:inputType="text" />

    <TextView
        android:text="Enter business address :"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
```

Figure 2.7: Activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:orientation="vertical"
    tools:context=".SetupPasswordActivity">

    <TextView
        android:text="@string/setup_business_label"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/setup_business_value"
        android:hint="@string/setup_business_hint"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:autofillHints="name"
        android:inputType="text" />

    <TextView
        android:text="@string/setup_business_address_label"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/setup_business_address_value"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:autofillHints="postalAddress"
        android:inputType="text" />

    <TextView
        android:text="@string/setup_business_contact_label"
        android:layout_width="match_parent"
```

```xml
    android:layout_height="wrap_content" />
<EditText
    android:id="@+id/setup_business_contact_value"
    android:inputType="number"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:autofillHints="phone" />

<TextView
    android:text="@string/setup_new_password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<EditText
    android:id="@+id/setup_new_password_value"
    android:hint="@string/setup_new_password_hint"
    android:inputType="textPassword"
    android:maxLines="1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:autofillHints="password" />

<TextView
    android:text="@string/setup_confirm_password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<EditText
    android:id="@+id/setup_confirm_password_value"
    android:hint="@string/setup_confirm_password_hint"
    android:inputType="textPassword"
    android:maxLines="1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:autofillHints="text" />

<Button
    android:text="@string/setup"
    android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:id="@+id/setup_password_btn" />

    </LinearLayout>
```

2.4.3 Backend Architecture

The back-end architecture of the Invoice Pro system consists of three main components: the server, the application, and the database.

1. Server: The server serves as the central hub that handles the communication between the Android application and the database. It receives requests from the front-end application and processes them accordingly. This component plays a crucial role in managing the flow of data and ensuring that requests are handled efficiently.

    - The server is responsible for handling various tasks, such as authentication and authorization of user requests, routing requests to the appropriate endpoints, performing data validation and verification, and executing business logic. It acts as an intermediary between the front-end application and the database, ensuring that data is retrieved, stored, and manipulated accurately.

2. Application: The back-end application is responsible for implementing the business logic of the Invoice Pro system. It receives requests from the front-end application and processes them to generate appropriate responses. This component contains the necessary algorithms and computations to perform operations such as creating new bills, retrieving bill details, updating bill information, and performing calculations.

    - The application layer encapsulates the core functionality of the Invoice Pro system. It handles the logic behind generating bills, calculating taxes, managing customer information, and other operations required for proper bill management. The application layer communicates with the server to receive and process requests, and it interacts with the database to retrieve and update data as needed.

3. Database: The database component stores and manages the data required by the Invoice Pro application. It provides a reliable and structured storage solution for storing customer information, bill details, and other relevant data. In the case of Invoice Pro, the SQLite database is used, which is a lightweight and efficient database management system specifically designed for Android applications.

- The database stores data in tables, which are organized in a structured manner based on the application's data model. It allows for efficient retrieval, insertion, and modification of data. The application layer interacts with the database through queries and commands to retrieve, update, or delete data as necessary.
- The database component plays a critical role in ensuring data integrity and consistency. It enforces constraints, such as uniqueness or referential integrity, to maintain the accuracy and reliability of the stored data. The server and application layers interact with the database to fetch data for generating responses or to store new information provided by the users.
- Overall, these back-end components work together to handle the processing of data in the Invoice Pro system. The server manages the communication between the front-end application and the database, while the application layer implements the business logic and the database stores and manages the data required by the system. This architecture ensures the smooth functioning of the Invoice Pro application by managing data storage, processing, and communication between the different layers.

Together, these back-end components ensure the smooth functioning of the Invoice Pro application by managing data storage, processing, and communication between the front-end and the database.

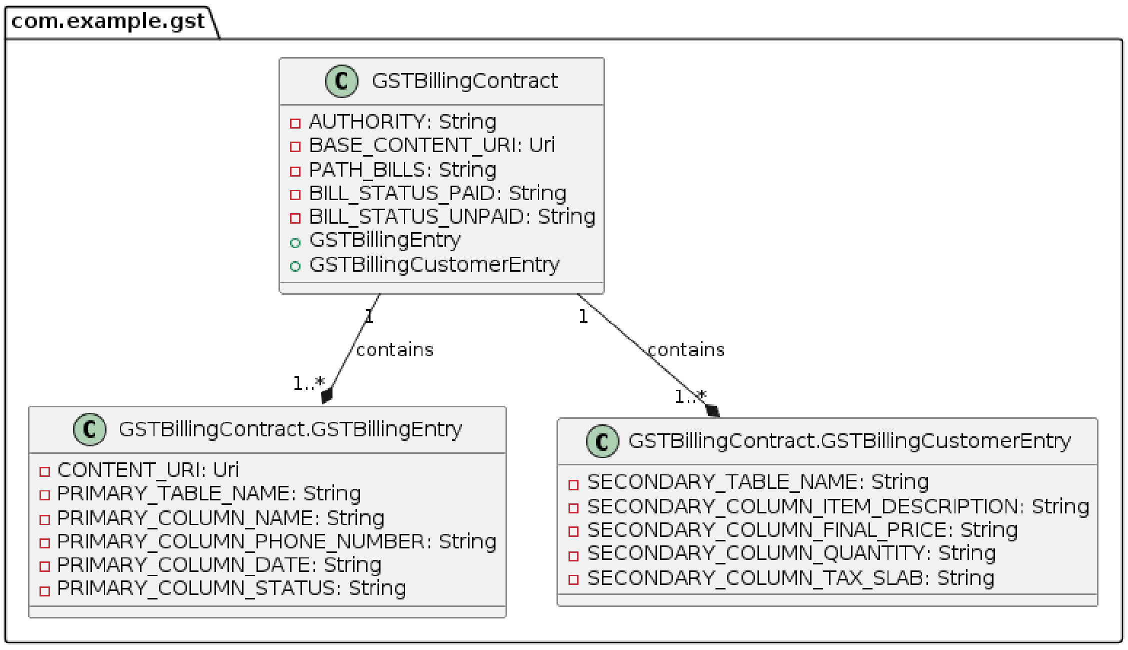

com.example.gst
GSTBillingContract
AUTHORITY: String
BASE_CONTENT_URI: Uri
PATH_BILLS: String
BILL_STATUS_PAID: String
BILL_STATUS_UNPAID: String
GSTBillingEntry
GSTBillingCustomerEntry
1
contains
1..*
1
contains
1..*
GSTBillingContract.GSTBillingEntry
CONTENT_URI: Uri
PRIMARY_TABLE_NAME: String
PRIMARY_COLUMN_NAME: String
PRIMARY_COLUMN_PHONE_NUMBER: String
PRIMARY_COLUMN_DATE: String
PRIMARY_COLUMN_STATUS: String
GSTBillingContract.GSTBillingCustomerEntry
SECONDARY_TABLE_NAME: String
SECONDARY_COLUMN_ITEM_DESCRIPTION: String
SECONDARY_COLUMN_FINAL_PRICE: String
SECONDARY_COLUMN_QUANTITY: String
SECONDARY_COLUMN_TAX_SLAB: String

Figure 2.8: *Database*

```java
package com.example.gst;

import android.net.Uri;
import android.provider.BaseColumns;

public class GSTBillingContract {

    public static final String AUTHORITY = "com.example.gst";
    public static final Uri BASE_CONTENT_URI = Uri.parse("content://" +
        AUTHORITY);
    public static final String PATH_BILLS = "billsMain";

    public static final String BILL_STATUS_PAID = "paid";
    public static final String BILL_STATUS_UNPAID = "unpaid";

    public static final class GSTBillingEntry implements BaseColumns {

        public static final Uri CONTENT_URI =
            BASE_CONTENT_URI.buildUpon().appendPath(PATH_BILLS).build();

        public static final String PRIMARY_TABLE_NAME = "billsMain";
        public static final String PRIMARY_COLUMN_NAME = "customerName";
        public static final String PRIMARY_COLUMN_PHONE_NUMBER = "phoneNumber";
        public static final String PRIMARY_COLUMN_DATE = "billDate";
        public static final String PRIMARY_COLUMN_STATUS = "billStatus";

    }

    public static final class GSTBillingCustomerEntry implements BaseColumns{

        public static final String SECONDARY_TABLE_NAME = "bill";
        public static final String SECONDARY_COLUMN_ITEM_DESCRIPTION = "item";
        public static final String SECONDARY_COLUMN_FINAL_PRICE = "finalPrice";
        public static final String SECONDARY_COLUMN_QUANTITY = "quantity";
        public static final String SECONDARY_COLUMN_TAX_SLAB = "taxSlab";

    }

}
```

```java
package com.example.gst;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class GSTBillingDbHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "GSTBillsDB.db";
    private static final int VERSION = 1;

    public GSTBillingDbHelper(Context context) {
        super(context, DATABASE_NAME, null, VERSION);
    }


    @Override
    public void onCreate(SQLiteDatabase db) {

        final String CREATE_TABLE = "CREATE TABLE " +
            GSTBillingContract.GSTBillingEntry.PRIMARY_TABLE_NAME + " (" +
                GSTBillingContract.GSTBillingEntry._ID + " INTEGER PRIMARY KEY
                    AUTOINCREMENT, " +
                GSTBillingContract.GSTBillingEntry.PRIMARY_COLUMN_NAME + " TEXT
                    NOT NULL, " +
                GSTBillingContract.GSTBillingEntry.PRIMARY_COLUMN_PHONE_NUMBER + "
                    TEXT NOT NULL, " +
                GSTBillingContract.GSTBillingEntry.PRIMARY_COLUMN_DATE + " TEXT
                    NOT NULL, " +
                GSTBillingContract.GSTBillingEntry.PRIMARY_COLUMN_STATUS + " TEXT
                    NOT NULL);";

        db.execSQL(CREATE_TABLE);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }

    public static void createBillTable(SQLiteDatabase db, String billId){

        final String CREATE_TABLE = "CREATE TABLE " +
            GSTBillingContract.GSTBillingCustomerEntry.SECONDARY_TABLE_NAME +
            billId + " (" +
                GSTBillingContract.GSTBillingCustomerEntry._ID + " INTEGER PRIMARY
                    KEY AUTOINCREMENT, " +
                GSTBillingContract.GSTBillingCustomerEntry.
                SECONDARY_COLUMN_ITEM_DESCRIPTION + " TEXT NOT NULL, " +
                GSTBillingContract.GSTBillingCustomerEntry.SECONDARY_COLUMN_FINAL_PRICE
                    + " REAL NOT NULL, " +
                GSTBillingContract.GSTBillingCustomerEntry.SECONDARY_COLUMN_QUANTITY
                    + " INTEGER NOT NULL, " +
                GSTBillingContract.GSTBillingCustomerEntry.SECONDARY_COLUMN_TAX_SLAB
                    + " INTEGER NOT NULL);";

        db.execSQL(CREATE_TABLE);

    }

    public static void dropBillTable(SQLiteDatabase db, String billId){

        final String DROP_TABLE = "DROP TABLE IF EXISTS " +
            GSTBillingContract.GSTBillingCustomerEntry.SECONDARY_TABLE_NAME +
            billId;
        db.execSQL(DROP_TABLE);

    }

}
```

3 Results and Discussion

3.1 Detail Design

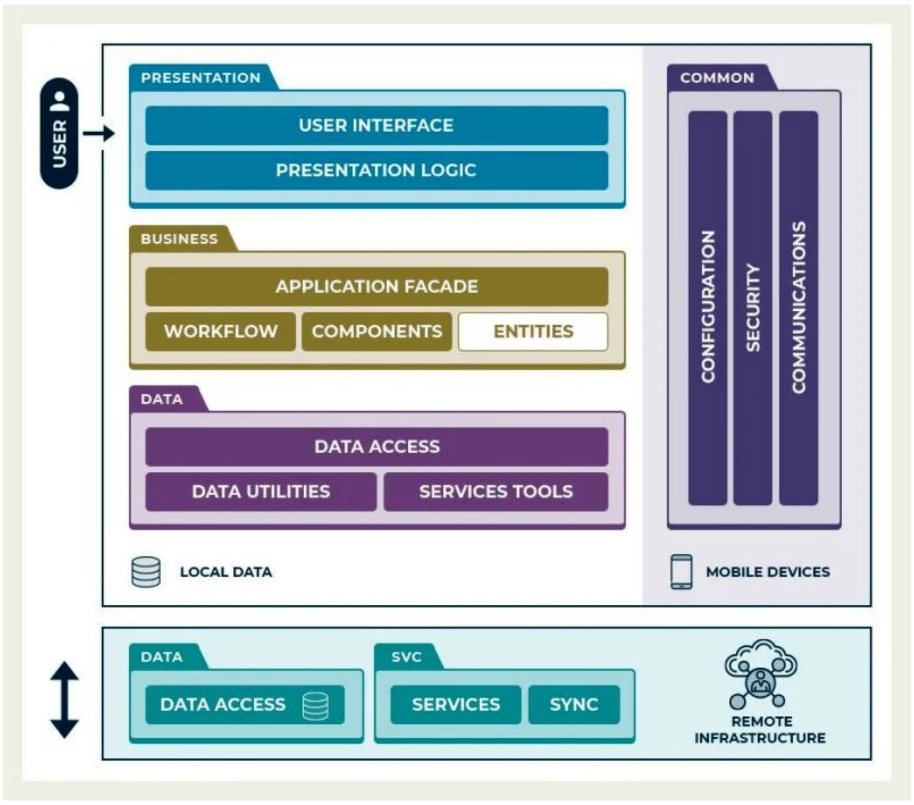


Figure 3.1: *Design Architecture*

1. User Interface Design: The user interface design involves creating visually appealing and intuitive interfaces that facilitate seamless interaction with the application. It includes designing screens, layouts, navigation flows, and input forms to ensure a user-friendly experience. Attention is given to factors such as responsiveness, accessibility, and adherence to design principles.

2. Database Design: The database design involves designing the structure and relationships of the underlying database that stores the bill and tax-related data. It includes defining tables, columns, primary keys, foreign keys, and constraints to ensure efficient data storage and retrieval. The design also considers data normalization techniques to minimize redundancy and improve data integrity.

3. Component Design: The component design focuses on breaking down the system into smaller functional components or modules. Each component is designed to encapsulate specific functionality and adhere to the principles of modularity and reusability. The design specifies the interfaces, methods, and data structures required for each component, ensuring loose coupling and high cohesion.

4. Algorithm Design: The algorithm design involves designing the logic and algorithms necessary for automating GST tax calculations, bill sorting, filtering, and other computational tasks. It includes selecting appropriate algorithms, optimizing for performance and efficiency, and considering error handling and exception scenarios.

The detailed design phase also considers other aspects such as error handling, exception management, security measures, and integration with external systems if required.

3.2 System Design using various Structured analysis and design tools

Various structured analysis and design tools are utilized to represent and document the system's architecture, data flow, and functionality. These tools include DFDs (Data Flow Diagrams), a Data Dictionary, Structured Charts, Flowcharts, and UML (Unified Modeling Language).

1. Data Flow Diagrams (DFDs): DFDs depict the flow of data within the system, illustrating how information moves between different processes, external entities, and data stores. They provide a visual representation of the system's data flow and help identify inputs, outputs, and transformations. DFDs in the project show the flow of data related to bill management, GST tax calculations, and other relevant processes.

2. Data Dictionary: The Data Dictionary is a documentation tool that defines and describes the data elements used in the system. It provides detailed information about each data item, including its name, type, length, constraints, and relationships with other data items. The Data Dictionary for the project specifies the data elements used for bills, taxes, customers, and other entities.

3. Structured Charts: Structured Charts represent the hierarchical structure of the system's modules or functions. They illustrate the organization and interrelationships between different program modules or subroutines. Structured Charts help in understanding the control flow and modular structure of the system, aiding in the implementation and maintenance phases.

4. Flowcharts: Flowcharts provide a graphical representation of the logical steps and decision points within a process. They illustrate the sequence of operations and the conditions that determine the flow of control. Flowcharts are used to represent the flow of activities and decision-making processes within the ”Billing Pad GST” application.

5. UML (Unified Modeling Language): UML is a standard modeling language that offers a comprehensive set of graphical notations for representing systems. It includes various diagrams such as Use Case Diagrams, Class Diagrams, Sequence Diagrams, and Activity Diagrams. UML diagrams are employed to depict the system’s structure, behavior, and interactions, providing a holistic view of the application.

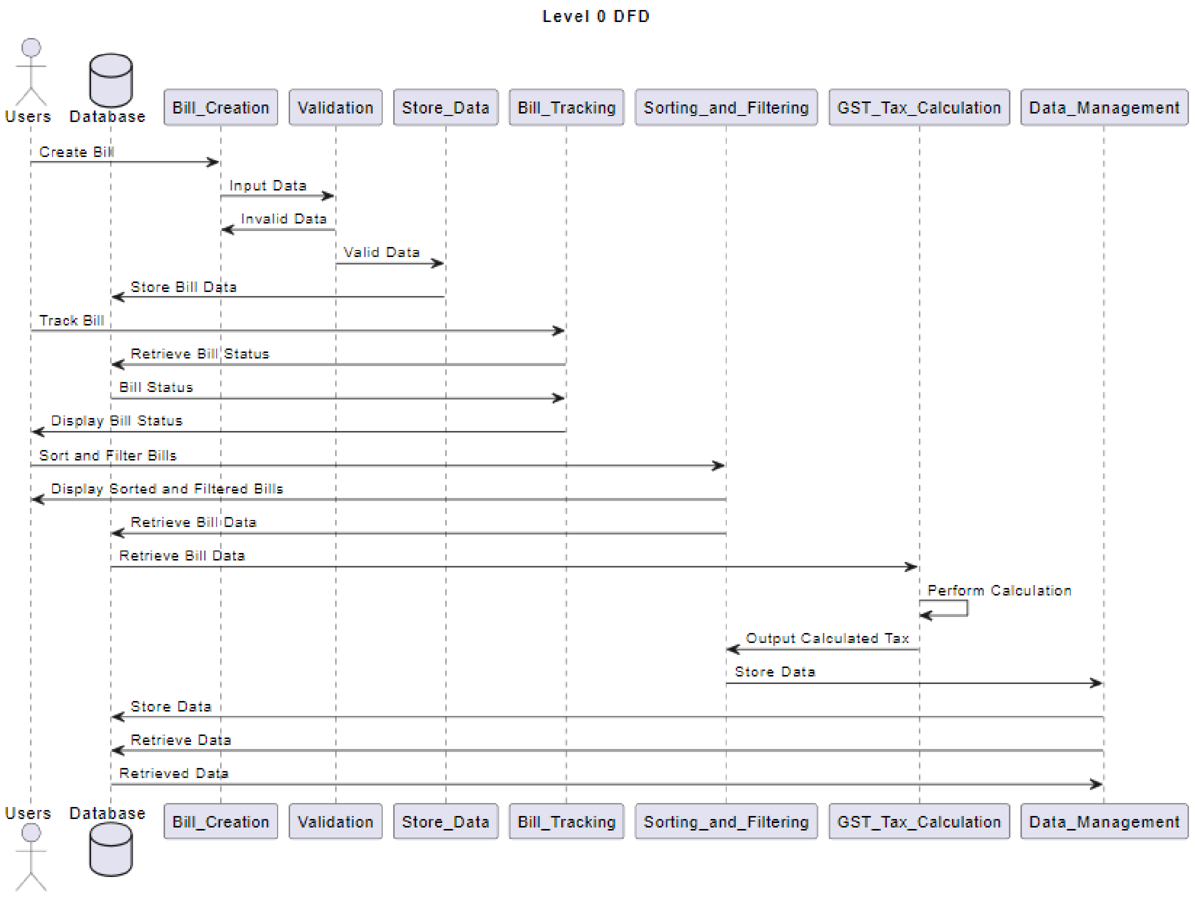


Figure 3.2: *Data Flow Diagram: L0 (Creating Bill*

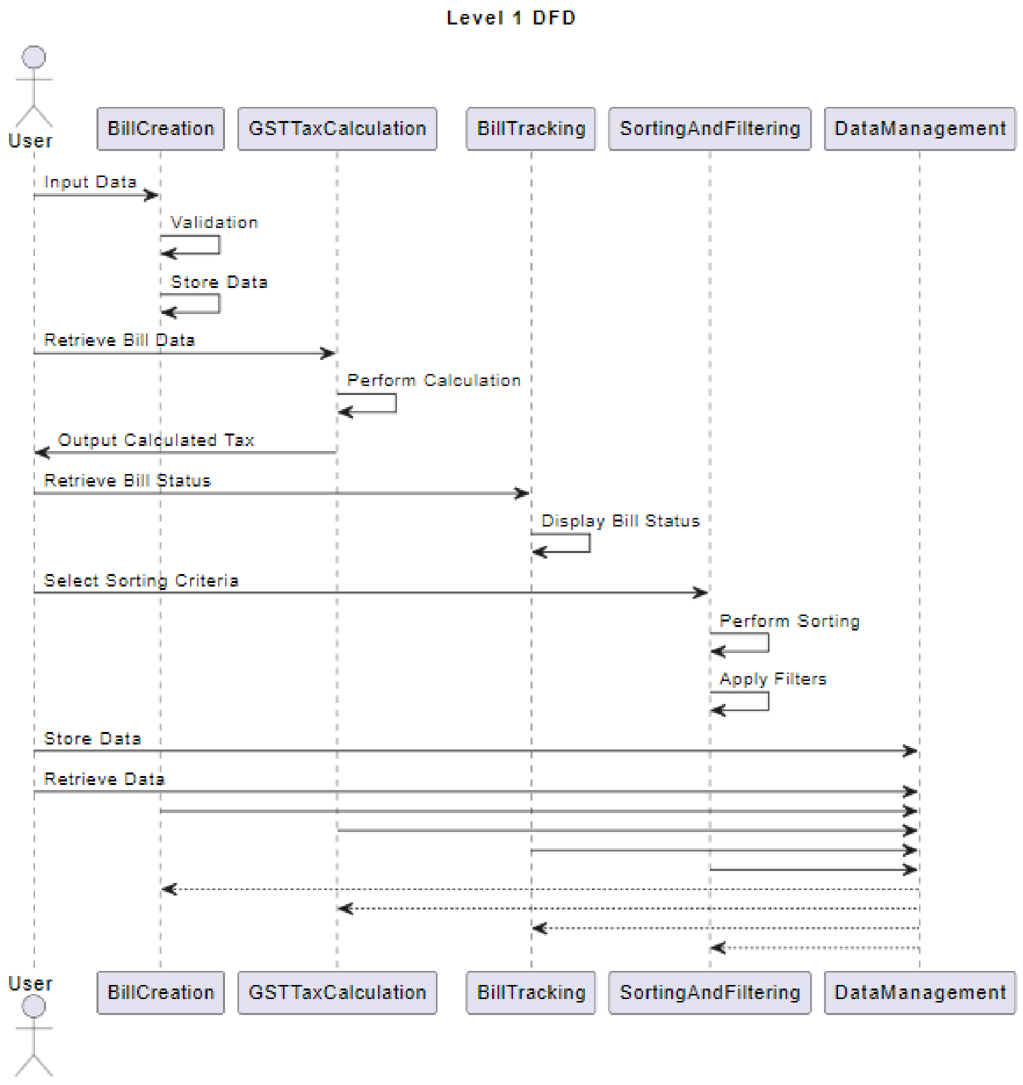


Figure 3.3: *Data Flow Diagram: L1 (Storing Bill)*

3.3 User Interface Representation

1. Upon launching the application, users are greeted with a login screen where they can securely access their accounts or create new ones.

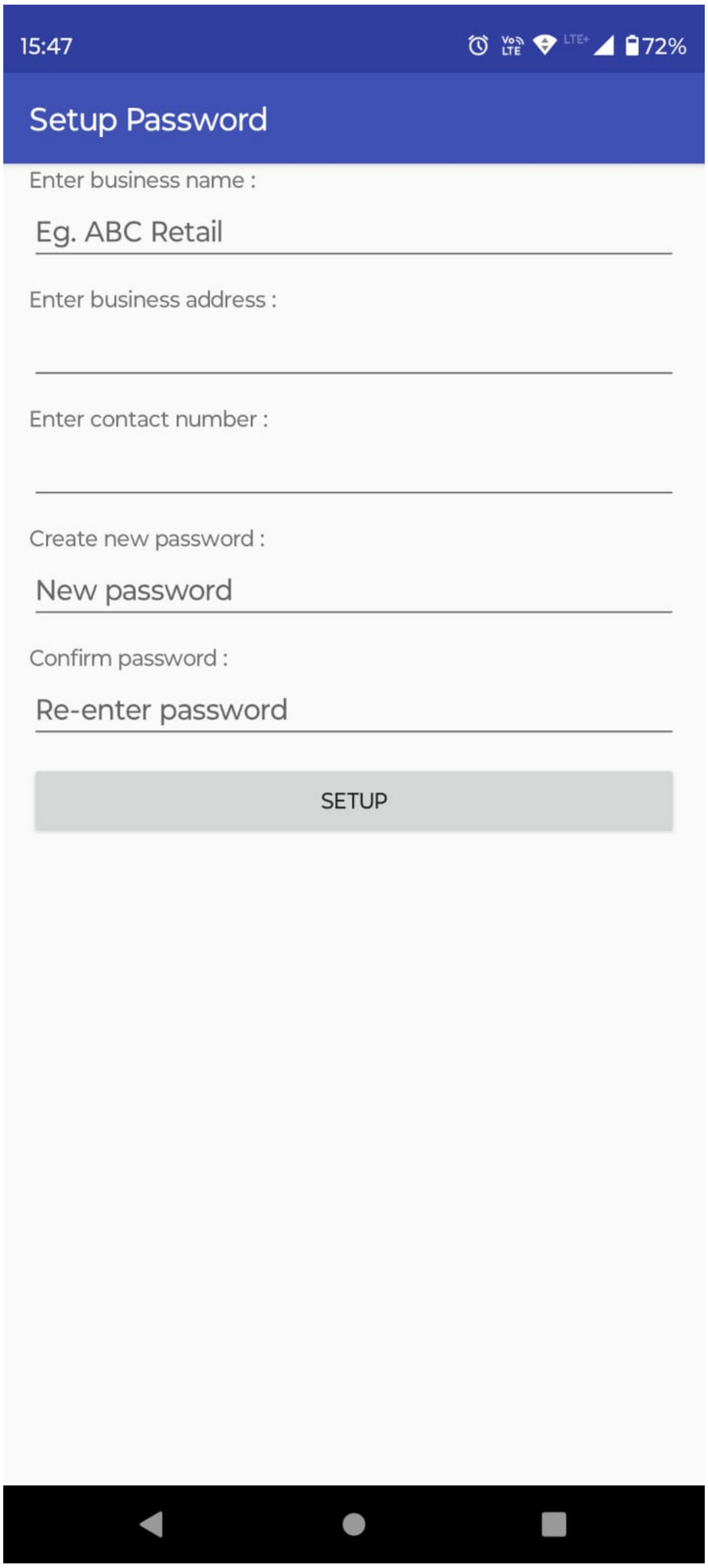
2. Once logged in, the main dashboard provides an overview of the user's bill management activities. The dashboard showcases important information such as the total number of bills, pending payments, and recent activities.

3. The navigation menu, typically located on the right side, allows users to easily access different sections of the application. These sections include bill creation, tracking, sorting, filtering, and data management.

4. In the bill creation section, users can input relevant details such as the customer's name, product or service description, quantity, and price. The application automatically calculates the GST tax based on the integrated tax functionality.

5. The tracking section provides a comprehensive view of all created bills, allowing users to monitor their payment status, due dates, and any outstanding balances. Users can easily search for specific bills using filters such as date, customer name, or bill number.

6. Throughout the application, attention is given to ensuring a smooth and seamless user experience. Visual cues such as icons, color schemes, and intuitive buttons enhance usability and guide users through different actions.



Setup Password

Enter business name :

Eg. ABC Retail

Enter business address :

Enter contact number :

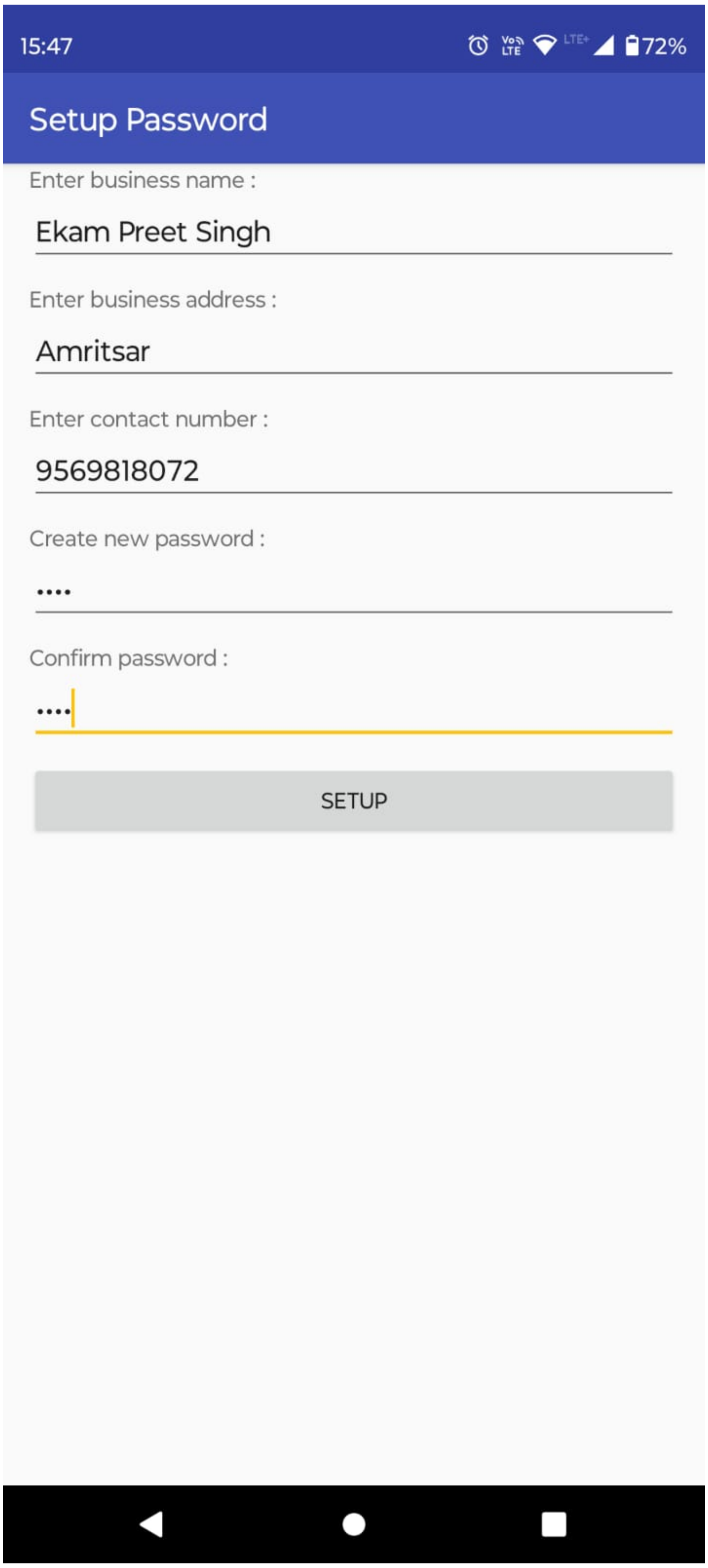
Create new password :

New password

Confirm password :

Re-enter password

Figure 3.4: *Create Account*


15:47 72%

Setup Password

Enter business name :

Ekam Preet Singh

Enter business address :

Amritsar

Enter contact number :

9569818072

Create new password :

••••

Confirm password :

••••

Figure 3.5: *Information*

Figure 3.6: Dashboard

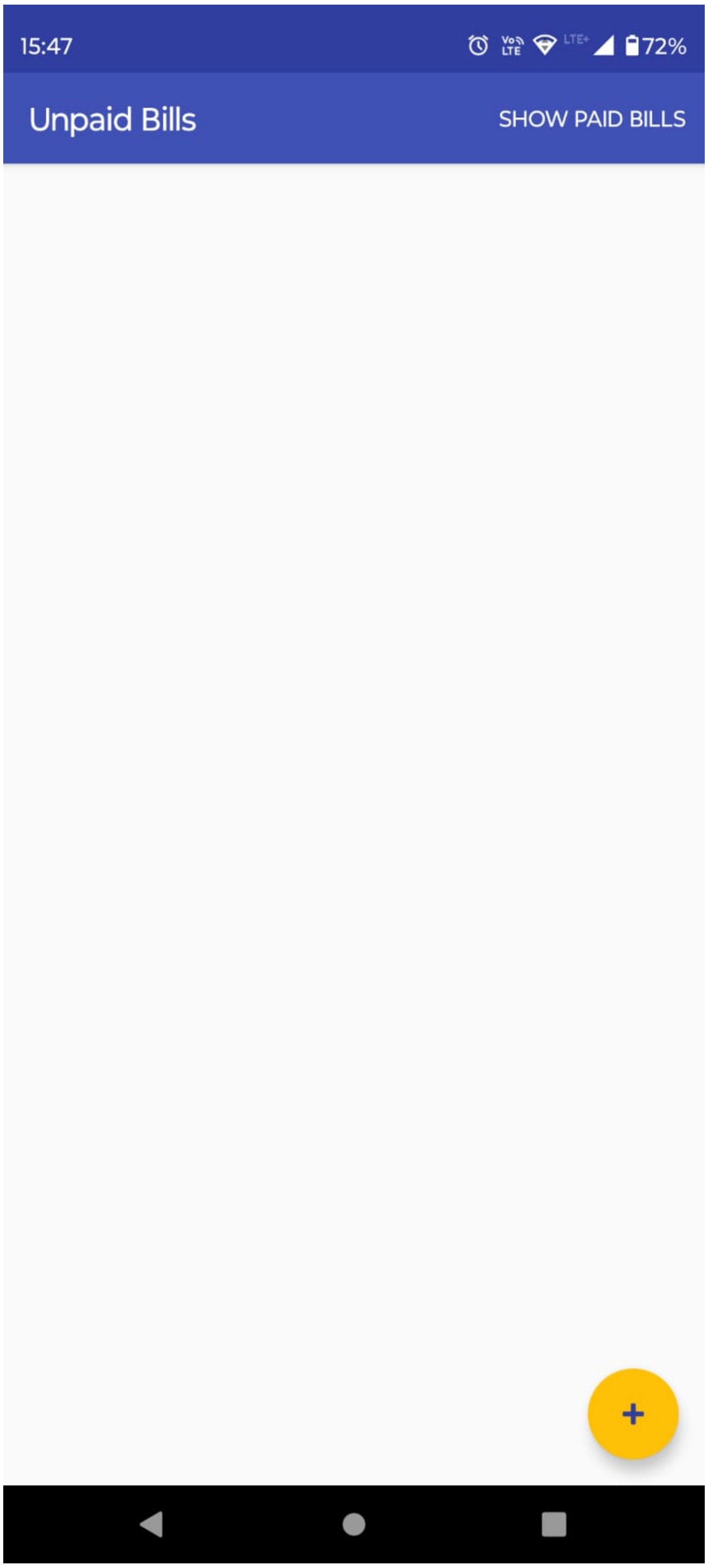


Figure 3.6: *Dashboard*

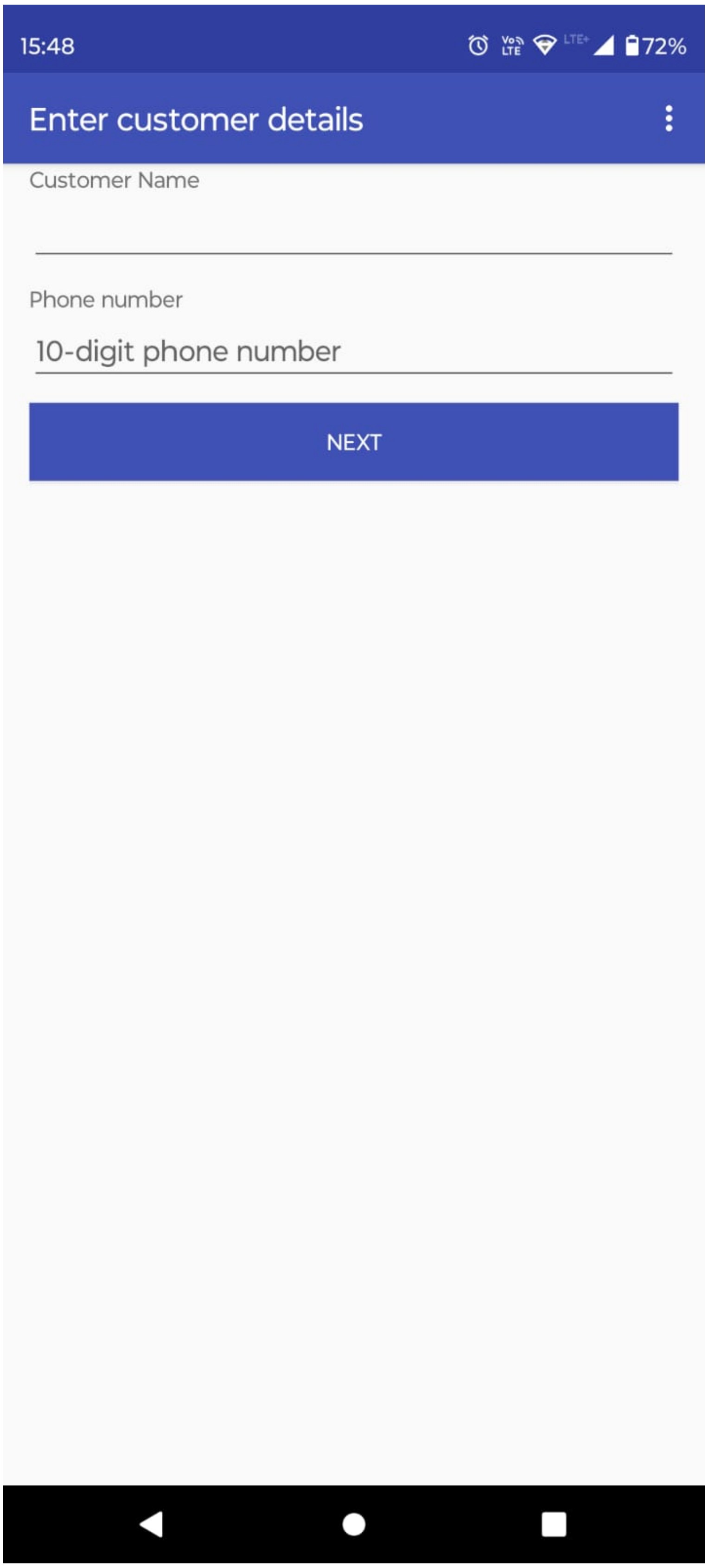
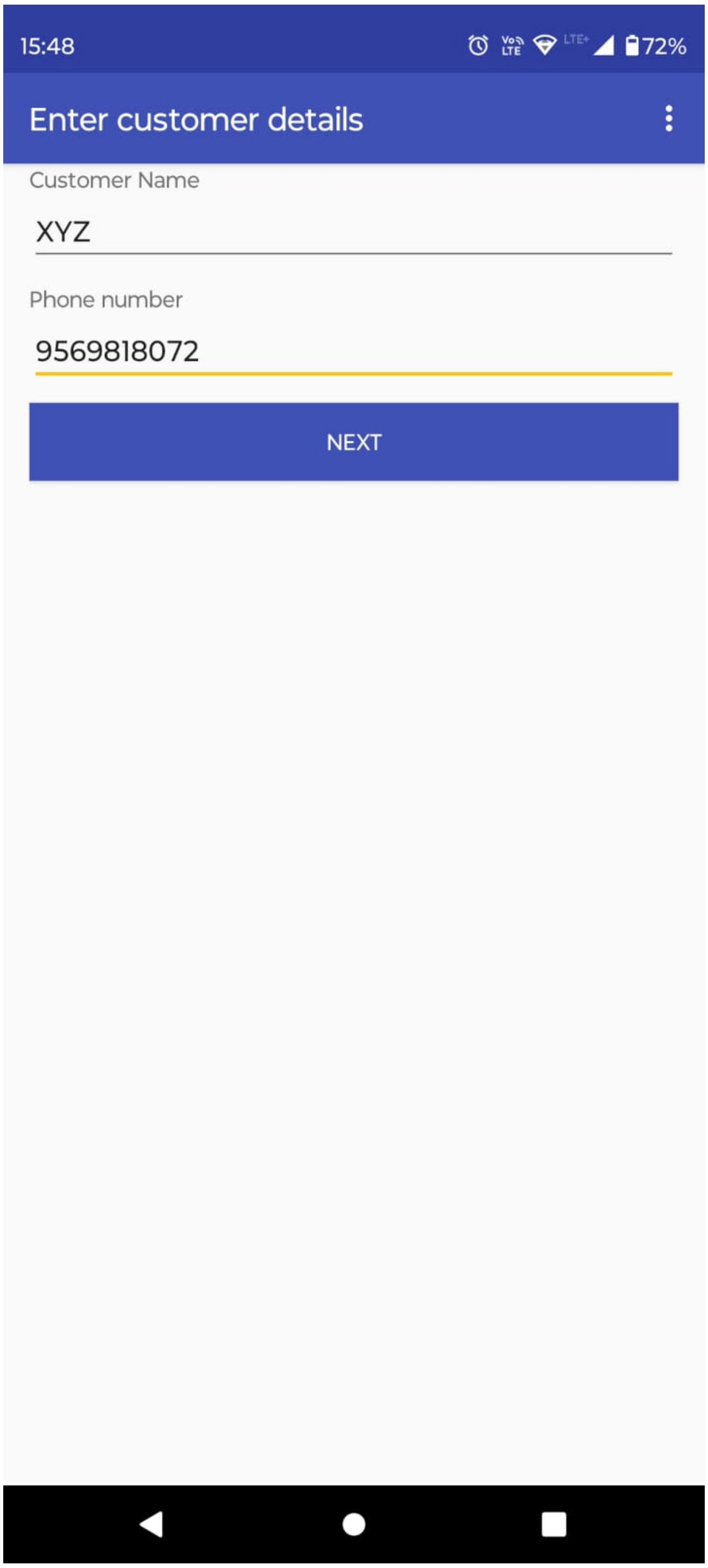


Figure 3.7: *New Bill*

Figure 3.8: Bill Account



Enter customer details

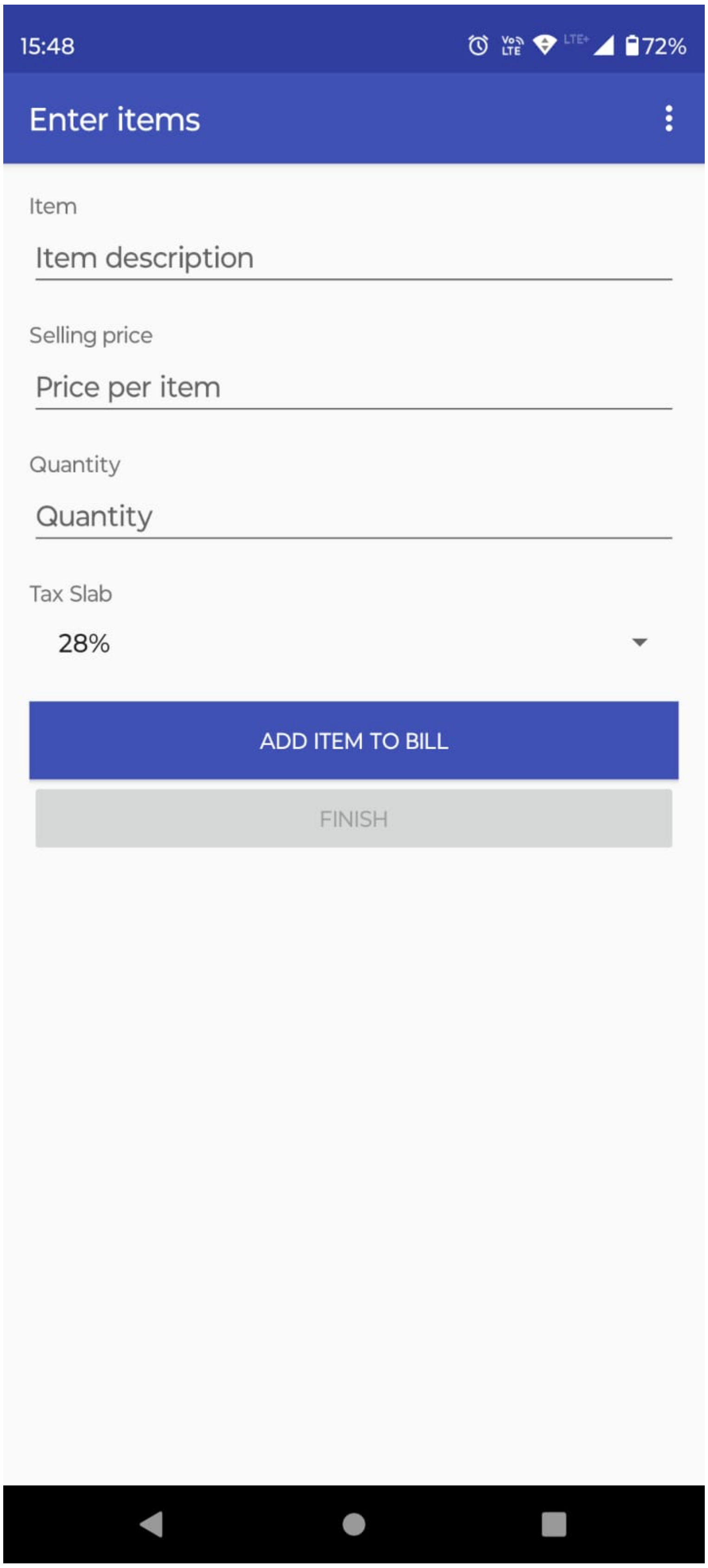
Customer Name

XYZ

Phone number

9569818072

Figure 3.8: *Bill Account*


15:48
72%

Enter items

Item

Item description

Selling price

Price per item

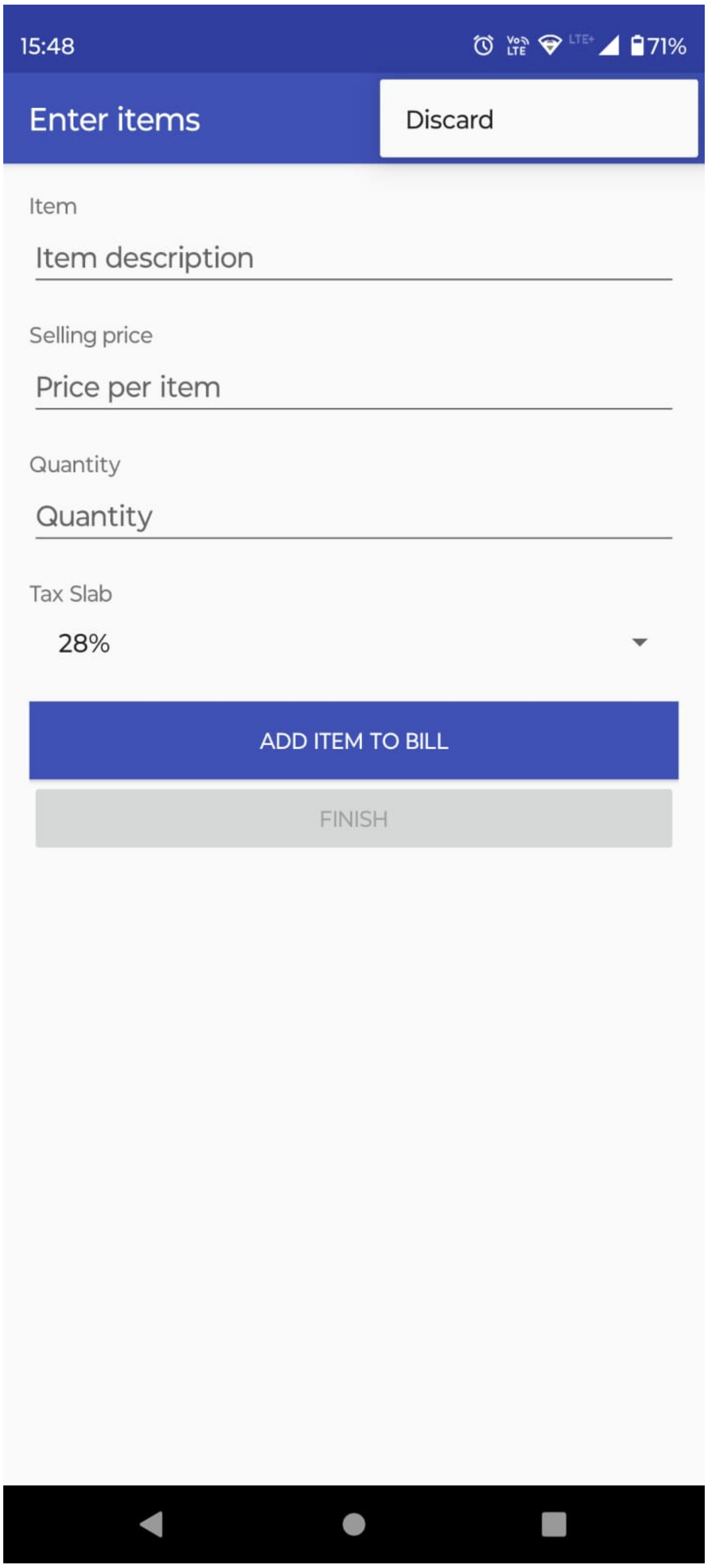
Quantity

Quantity

Tax Slab

28%

Figure 3.9: *Bill Item*



Item

Item description

Selling price

Price per item

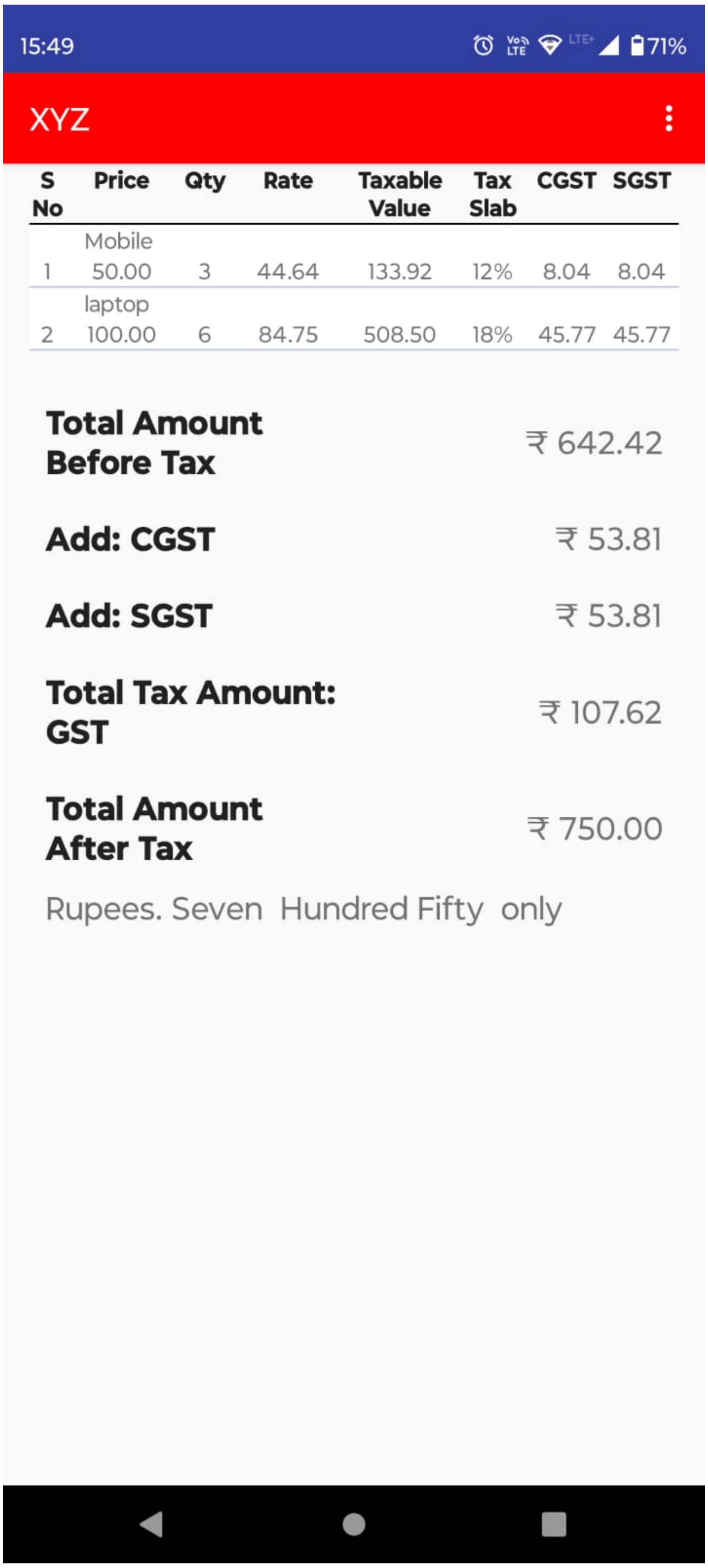
Quantity

Quantity

Tax Slab

28%

Figure 3.10: *Discard Button*



15:49 71%

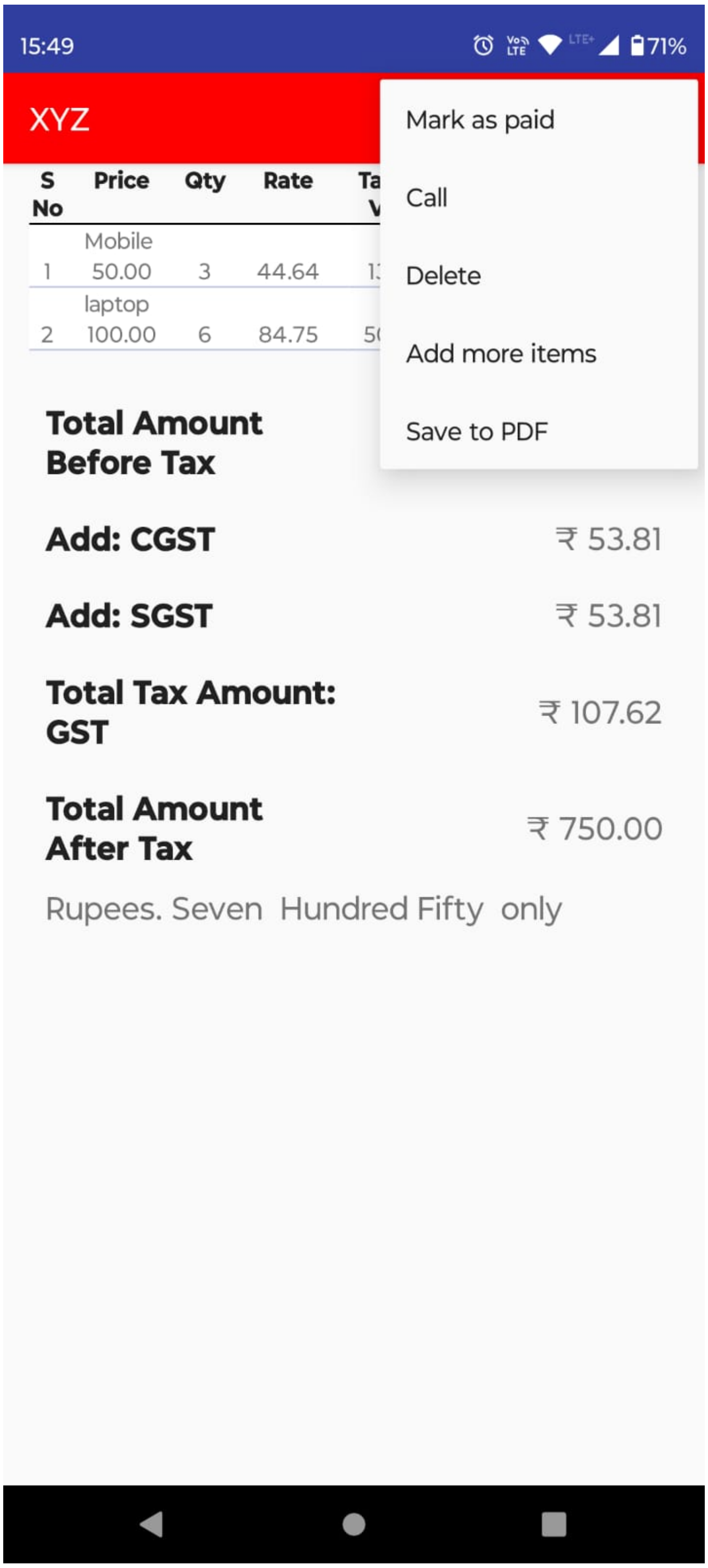
XYZ

S No	Price	Qty	Rate	Taxable Value	Tax Slab	CGST	SGST
1	Mobile 50.00	3	44.64	133.92	12%	8.04	8.04
2	laptop 100.00	6	84.75	508.50	18%	45.77	45.77

Total Amount Before Tax	₹ 642.42
Add: CGST	₹ 53.81
Add: SGST	₹ 53.81
Total Tax Amount: GST	₹ 107.62
Total Amount After Tax	₹ 750.00

Rupees. Seven Hundred Fifty only

Figure 3.11: *Calculated Bill*


15:49 71%

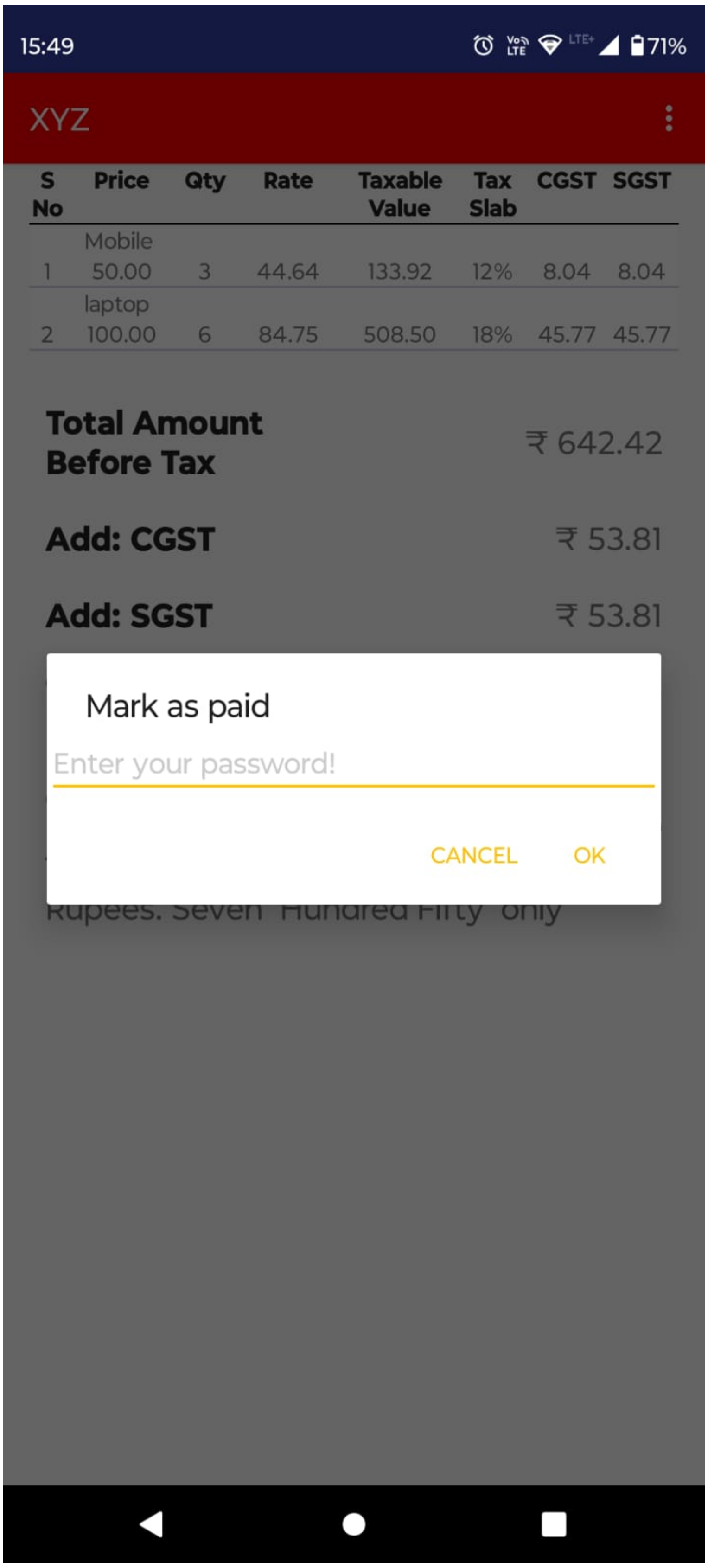
XYZ

S No	Price	Qty	Rate	Ta V
	Mobile			
1	50.00	3	44.64	1:
	laptop			
2	100.00	6	84.75	5(

Total Amount Before Tax	
Add: CGST	₹ 53.81
Add: SGST	₹ 53.81
Total Tax Amount: GST	₹ 107.62
Total Amount After Tax	₹ 750.00

Rupees. Seven Hundred Fifty only

Figure 3.12: *More Features*



15:49 71%

XYZ

S No	Price	Qty	Rate	Taxable Value	Tax Slab	CGST	SGST
1	Mobile 50.00	3	44.64	133.92	12%	8.04	8.04
2	laptop 100.00	6	84.75	508.50	18%	45.77	45.77

Total Amount Before Tax ₹ 642.42

Add: CGST ₹ 53.81

Add: SGST ₹ 53.81

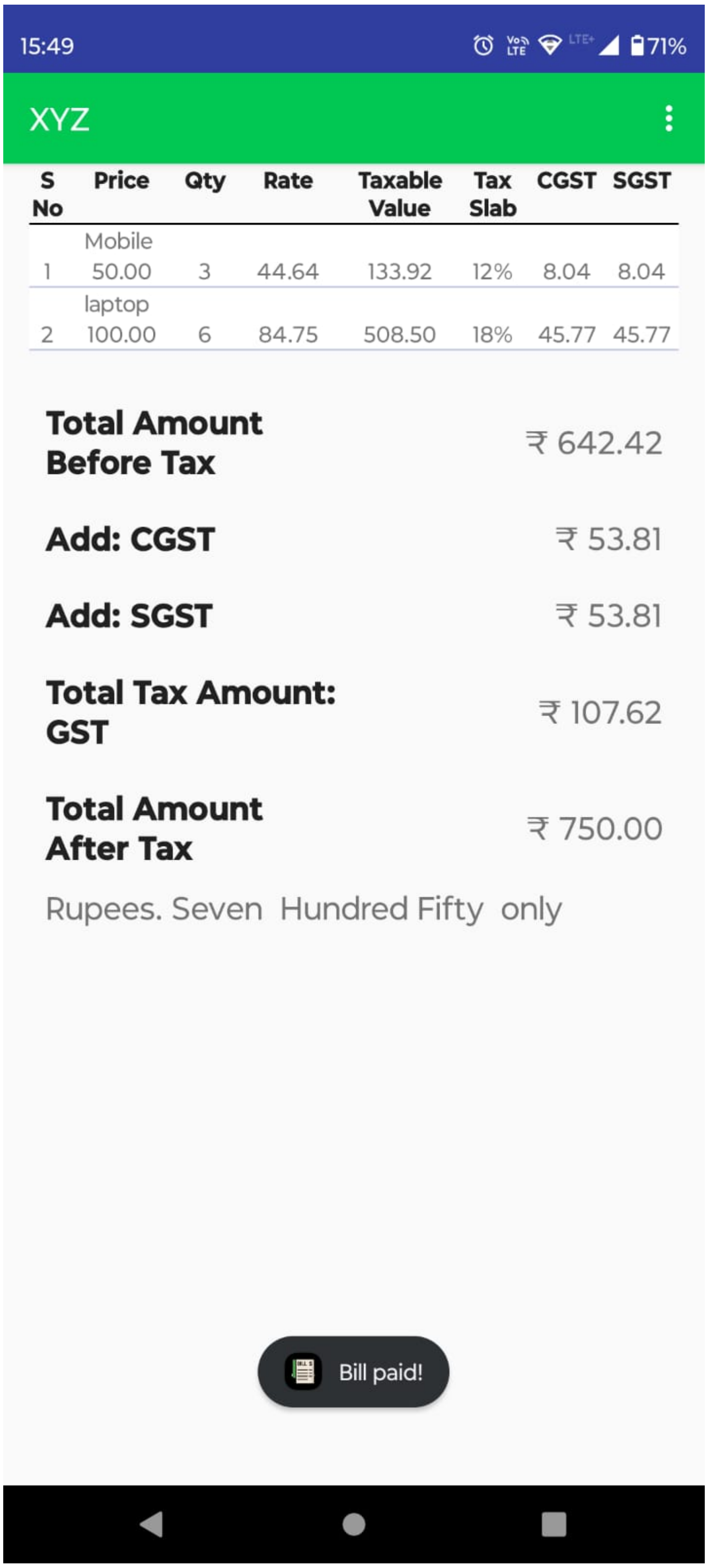
Mark as paid

Enter your password!

CANCEL OK

Rupees: Seven Hundred Fifty only

Figure 3.13: *User Authentication*


15:49 71%

XYZ

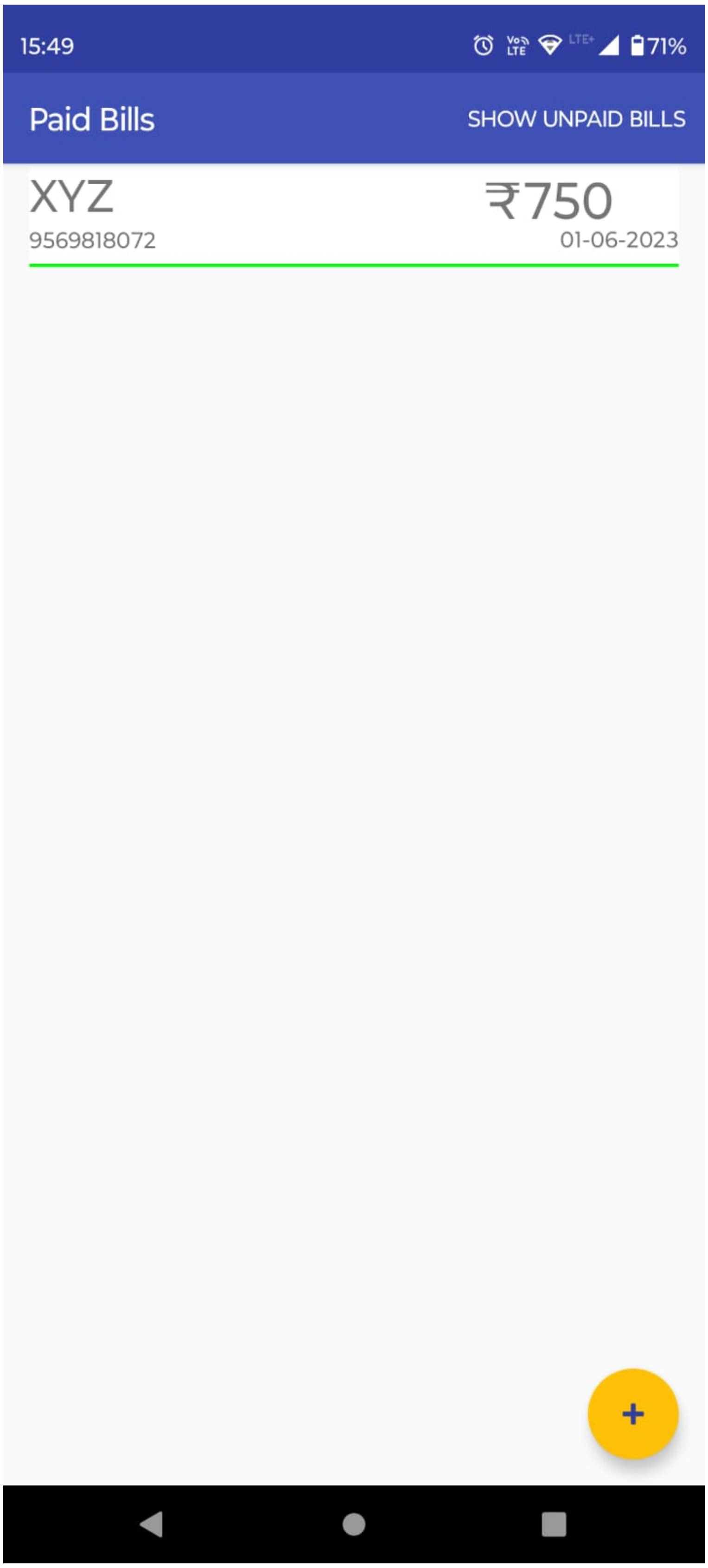
S No	Price	Qty	Rate	Taxable Value	Tax Slab	CGST	SGST
	Mobile						
1	50.00	3	44.64	133.92	12%	8.04	8.04
	laptop						
2	100.00	6	84.75	508.50	18%	45.77	45.77

Total Amount Before Tax	₹ 642.42
Add: CGST	₹ 53.81
Add: SGST	₹ 53.81
Total Tax Amount: GST	₹ 107.62
Total Amount After Tax	₹ 750.00

Rupees. Seven Hundred Fifty only

Bill paid!

Figure 3.14: *Mark Bill Paid*



15:49 71%

Paid Bills SHOW UNPAID BILLS

XYZ ₹750

9569818072 01-06-2023

Figure 3.15: *Dashboard-2*

4 Conclusion and Future Scope

4.1 Conclusion

In conclusion, the Invoice Pro project, developed by a single individual, has successfully created an Android application for efficient bill management with integrated GST tax functionality. The project focused on addressing the identified needs of automated bill management and accurate tax calculations.

Throughout the project, the developer followed a systematic approach, starting with requirement gathering and analysis to understand the specific needs and challenges related to bill management and GST tax calculations. This initial phase helped in defining the project scope and objectives.

The design phase involved creating user-friendly interfaces that are intuitive and easy to navigate. The developer utilized tools like flowcharts and data flow diagrams to plan and map out the application's functionality, ensuring a logical flow of information and operations.

During the development phase, the individual developer implemented the identified features and functionalities, adhering to software engineering best practices and coding standards. They ensured code quality, maintainability, and scalability, taking into account the specific requirements of the project.

Testing played a crucial role in ensuring the application's reliability and usability. The developer conducted various testing techniques, such as unit testing and user acceptance testing, to identify and fix any issues or bugs. This helped in delivering a high-quality and error-free application.

Data management was also given due importance in the project. The developer utilized a structured and efficient database system like SQLite to store and manage customer information, bill details, and other relevant data. This ensured the integrity and effective retrieval of data when needed.

Overall, the Invoice Pro project, developed by a single individual, successfully achieved its objectives of automated bill management and accurate GST tax calculations. The application provides a seamless user experience, streamlines the billing process, and eliminates manual tax calculations. This project demonstrates the developer's dedication to software engineering practices, attention to user needs, and the ability to create a reliable and efficient Android application as an individual contributor.

4.2 Future Scope

Some key areas of future development and improvement include:

1. Integration with Additional Payment Gateways: Currently, the application supports basic bill management and GST tax calculations. In the future, integrating additional payment gateways will enable users to make seamless online payments directly from the application. This enhancement will enhance the overall convenience and user experience.

2. Integration with Cloud Services: Implementing cloud integration will allow users to store and access their bill data securely from anywhere, at any time. This feature will enhance data accessibility, backup, and collaboration among multiple users or devices.

3. Advanced Reporting and Analytics: Adding advanced reporting and analytics capabilities will enable users to gain valuable insights into their billing patterns, expenses, and tax calculations. This feature will help businesses make informed decisions and optimize their financial processes.

4. Multi-platform Compatibility: Expanding the application's compatibility to other platforms, such as iOS, will widen its user base and market reach. This will allow a broader range of users to benefit from the efficient bill management and GST tax functionality offered by the application.

5. Integration with Accounting Software: Integrating the application with popular accounting software such as QuickBooks or Tally will enable seamless synchronization of bill data, further streamlining financial processes for businesses.

6. User-Driven Customization: Providing users with the ability to customize the application's interface, themes, and preferences will enhance user satisfaction and personalization.

7. Continuous Updates and Bug Fixes: Regular updates and bug fixes are essential to ensure the application's stability, security, and compatibility with evolving operating systems and devices.

By focusing on these future development areas, the Invoice Pro project can continue to meet the evolving needs of businesses and individuals, providing an efficient and user-friendly solution for bill management and GST tax calculations.

References

[1] Android SDK.[Online]
Available:https://developer.android.com/studio
[Accessed: 2-Jan-2023].

[2] Download Diagrams.[Online]
Available:https://diachat.inmytree.co.za/
[Accessed: 15-April-2023].

[3] Android Application. [Online]
Available:https://vemanait.edu.in/Android-based-leave-management-system.pdf
[Accessed: 25-March-2023].

[4] Sowbhagya Visweswaran, *Create PDF from layout XML in Android Studio using PDFDocument class*, [Accessed: 15-Feb-2023]

[5] Dawn Griffiths, *Head First Android Development: A Brain-Friendly Guide*, Issue: 25 August 2015 [Accessed: 15-May-2023]