BUS TICKET BOOKING ANDROID APP

SUBMITTED BY:

Participant 01: Nitiwit Kuldiloke
Roll No : 2K19/CO/263
Participant 02: Pooja Jaiswal
Roll No : 2K19/CO/275

SUBMITTED TO:

MS.Pavi Saraswat
Software Engineering

INTRODUCTION

The “BUS TICKET BOOKING SYSTEM “ has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardship faced by the traditional ticket booking system.The ticket booking is the method for confirmation to attend in the bus which is traveling to their destination. As the pace of technology is going fast as well as the pace of lifestyle which people would like to get as comfortable as they could, people who are still using the service would like to guarantee their seat in the bus. Thus, we found this statement as the problem which we could develop an innovation to solve it. With the problem above, instead of contacting booking a ticket, we can apply the technology to solve this problem for those who want to discard their chance of missing the bus in case it is full of people.

PROBLEM STATEMENT

A problem statement always expresses the words that will be used to keep the effort on tracking things out to represent a solvable problem with the existing system. Below are the problems with the current system.

-> Customers have to go to the counter to buy bus tickets or ask for bus schedules.
->Furthermore, customers need to pay cash when they buy the bus ticket and sometimes need to queue up for a long time to get the bus ticket.

-> Besides that, customers are also not allowed to buy bus tickets through telephone and the bus company’s telephones are always busy. Operation times of the bus station to book tickets are limited to office hours only. People need to go to the bus station or terminal to book the ticket.

ABSTRACT

It is a Digital Bus Management System for Bus Transportation services. People who are traveling through a Bus Transportation System or have their own Bus Travel Agency or State/National Government Bus Transport Service can use this system. The key features of the system are Manage Bus, Manage Route, Manage Fares, Manage Bookings, Entire Bus Transport System, Book Online Seats, and much more. This system was made keeping in mind a Real-World Bus Transport Management Problems, Risk and Digital solutions to it.

AIM AND OBJECTIVE OF THE PROJECT

The fundamental aim to carry out this project is to create and design an Android Mobile Application of an Online Bus Booking System.

The objectives of this project are as follows:

->To investigate and analyze the problems on the existing systems and provide a solution to the customer and management by having an under-stable system that individuals will be able to operate within a short period.

->To identify the relevant features of various components and methods needed for the Android Mobile Application of Online Bus Booking System and also to give power of choice to consumers to choose that bus operator's services to use based on standard rating derived from consumer polling, traffic summonses obtained, and sales .

RATIONALE OF THE STUDY

The old manual system was suffering from a series of drawbacks . Since the whole of the system was to be maintained with hands the process of keeping , maintaining and retrieving the information was very tedious and lengthy . The records were never used to be in systematic order , there used to be lots of difficulties in associating any particular transaction with the particular context . If any information was to be found it was required to go through all the registers , documents there would have been never existing anything like report generation . There would always be unnecessary consumption of time while entering records and retrieving records . One more problem was that it was difficult to find errors while entering the records .

The main purpose of this study is to automate the manual procedures of reserving a bus ticket for any journey made.This system is said to be an automatic system and customers can select seats by themselves.

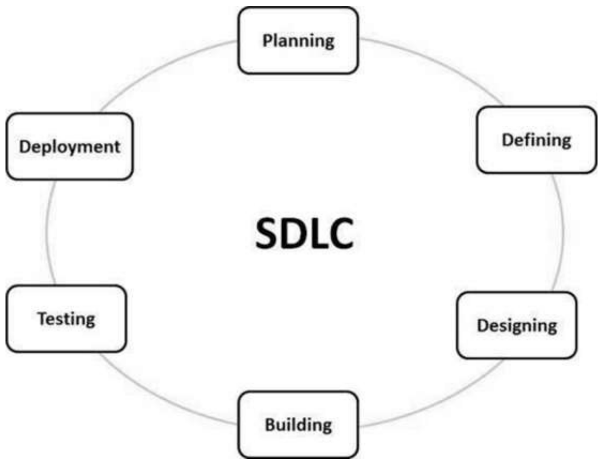
LITERATURE REVIEW

This chapter defines facts and findings on electronic ticketing or e-ticketing after reading some articles, books, websites or journals that are related to the system,decide to describe methodology that are used to develop the system, state out project requirements, explain action plans prior to the end of the project . All the information related to the online system is found by surfing the Internet and going to the library. Literature review is done and findings come out after reading through all the information.

SYSTEM DESIGN AND METHODOLOGY

Choice of Methodology

For any project to be completed, it has to go through stages called Development Life Cycles. System Development Life Cycle is the process of understanding how an Information System can support business needs,designing the system, building it and delivering it to users. The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within time and cost estimates. The life cycle defines a methodology for improving the quality of software and the overall development process.



The SDLC comprises of following phases:

I. Planning and Analysis

II. Design and Implementation.

System analysis

It is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system is identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is a problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken. Here in the project , a detailed study of the existing system is carried along with all the steps in system analysis. An idea for creating a better project was carried and the next steps were followed.

Problems with existing system

Existing system refers to the system that is being followed till now. The existing system requires more computational time, more manual calculations, and the complexity involved in Selection of features is high. Below are the problems with the current system.

-> Customers have to go to the counter to buy bus tickets or ask for bus schedules.

->Furthermore, customers need to pay cash when they buy the bus ticket and sometimes need to queue up for a long time to get the bus ticket.

-> Besides that, customers are also not allowed to buy bus tickets through telephone and the bus company's telephones are always busy.Operation times of the bus station to book tickets are limited to the office hours only. People need to go to the bus station or terminal to book the ticket.

->The other disadvantages are lack of security of data, Deficiency of Data accuracy, Time consuming etc. To avoid all these limitations and make the working more accurate the system needs to be computerized.

Objectives of Proposed system

The fundamental aim to carry out this project is to create and design an Android Mobile Application of Online Bus Booking System.

The objectives of this project are as follows:

To investigate and analyze the problems on the existing systems and provide a solution to the customer and management by having an under stable system that individuals will be able to operate within a short period of time.

To identify the relevant features of various components and methods needed for the Android

Mobile Application of Online Bus Booking System and also to give power of choice to consumers to choose which bus operator's services to use based on standard rating derived from consumer polling, traffic summonses obtained, and sales performance.

System design

It is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. System Design is the most creative and challenging phase in the system life cycle. Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to an effective system. System design is a solution to how to approach the creation of a new system. System design transforms a logic representation of what is required to do into the physical specification. The specification is converted into physical reality during development. In this phase, a logical system is built which fulfills the given requirements . Design phase of software development deals with transforming the Client's requirements into a logically working system . Normally , Design is performed in the following two steps :

1. Primary Design Phase

   → In this Phase , the system is designed at block level . The blocks are created on the basis of analysis done in the problem identification phase . Different blocks are created for different functions; emphasis is put on minimising the information flow between blocks . Thus, all activities which require more interaction are kept in one block .

2. Secondary Design phase

→ In the secondary phase the detailed design of every block is performed .

The general tasks involved in the design process are :

- Design various blocks for overall system processes.
- Design compact and workable modules in each block.
- Design various database structures.
- Specify details of programs to achieve desired functionality.
- Design the form of inputs and outputs of the system
- Perform documentation of the design.
- System reviews.

Feature Provided In Bus Ticket Booking Application

- **Login and registration**

  The user can just add his/her details and create an account on the app and will be able to use it whenever desired. The data will be stored in cloudFlare which works as a database to keep the data safe. During the process of Registration, the authentication of your mobile number to verify your identity, by the Firebase, will be sent automatically after you fill in the mobile number. One-Time-Password will be generated to your mobile via SMS. We have also included the feature of updating our profile where the user can update his/her name, phone number, and other details.

- **Add Start & Destination Station**

  Just add your Journey Starting Spot & Destination Spot and get all the options of buses and routes to reach there.See details of all the stops and stations so that you can reach the precise destination.

- **Book Seats**

  Select the seats of your comforts and convenience and book it along with your amigos and family to be together in a few mins.

- **Reservation Detail**

  Once you are done with selecting your desired seat and booking it for yourself you will be able to see the details of the reservation including your details and the bus details.

- **See History**

  See the history of booked tickets and rebook if you are often traveling through the same route.

- **Ticket cancellations**

  We even have a feature to cancel books for each of your assigned members. By keeping real-world problems and solutions in mind We have designed a system that eliminates real-world bus transport problems

- **User Friendly**

  The application will be so interactive that if any customer faces any problem, it will be reserved within the system itself.

SCOPE OF THE PROJECT

It will help both user and administrator to easily manage their work. Admin which is the bus company will be easier to get the customer information no need to contact by offline or by phone call which might have some issues during the conversation. Same as the customer or user which will be easier in order to book the ticket and know the timetable of the bus and manage their time for it.

Our project aim for the feasibility for both Bus Provider and Bus customer to feel comfortable to use the App

- To assist the staff in capturing the effort spent on their respective working areas.
- To utilize resources in an efficient manner by increasing their productivity through automation.
- The system generates types of information that can be used for various purposes.
- It satisfy the user requirement
- Be easy to understand by the user and operator
- Be easy to operate
- Have a good user interface
- Be expandable

Software requirements

The Software Requirements specification is a step-by-step process that helps define the requirements for the software that is used for system engineering. This specification is based on a conceptual framework that defines the various functions and performance of the software. In this project, softwares which are we require are the following:

- Android Operating System
- Java Programming Languages Compiler

- Android Studio IDE

The proposed system has the following requirements:

- System needs to store information related to the new entry of Bus.
- Application will maintain the data of all previous ticket booking.
- Applications need to have a security system to prevent data leakage.
- Applications need to provide a search engine for the data looking up.

User Interface Design

User Interface Design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventual presentation of desired inputs and outputs. The overall flow of screens and messages is called a dialogue.

Guideline for User Interface Designing

- The system user should alway know what they want to perform
- The interface should be formatted,thus, the same type of information will be in the same area.
- Sparingly use display attributes.
- If there is an error. App will not perform the next step until the error is corrected.
- The users should never receive a fatal error or error in the operating system.

Preliminary Software Description:

The preliminary investigation to determine the system's feasibility is the first step in the system development life cycle. The preliminary investigation's goal is to assess project requests. It is not a design study, and it does not include any detailed collection to describe the business system in any way. Rather, gathering information assists committee members in evaluating the merits of the project request and making an informed decision about the project's feasibility.

The following goals should be met by analysts working on the preliminary investigation:

- Make sure you have a clear understanding of the project request.
- Determine the project's scope.
- Analyze the costs and benefits of different approaches.
- Determine whether alternative approaches are technically and operationally feasible.
- Report your findings to management, along with recommendations for whether the proposal should be accepted or rejected.

Benefit of Developing

We are going to provide such an application and plan to the country which needs it and talk to their government for maintenance of the application. It might make profit in term of financial to us and also improve us as the software engineering by executing the real project as the learning material in the same time.

Requirement in term of incorporation

Users and administrators must be trained at the time of system implementation to ensure that the system runs smoothly. The training location will be provided by the client.

We spoke with the center's financial management team, the staff who kept records in multiple registers, and the reporting manager about their current system, their requirements, and their expectations from the new proposed system. Then, based on their requirements and the additional features they wanted to include in this system, we conducted a system study of the entire system.

The new system that I proposed and then developed will make the organization's job easier. It will assist the staff in producing the necessary reports, which will allow them to track their progress and services.As a result, it will make management much easier because all of the major activities will be computerized through this system.

Project Category

This portion of the project is Software Engineering based project which is implemented by the method of Waterfall Model of software development and with the help of Java Programming and database management by MongoDB.

Introduction about Software Engineering

Software engineering is the application of engineering principles to the design, development, and management of software. To solve the problems of low-quality software projects, software engineering was introduced. When a software development's timeframes, resources, and quality levels are all exceeded, issues occur. It ensures that the app is developed consistently, properly, on scheduled, on budget, according to requirements. A need for software engineering arose in response to the rapid changes in user requirements and the environment in which an application is expected to operate.

Developing Model

The Waterfall Model is the development model which we used in this project.

Waterfall model

It is also known as a linear-sequential life cycle model. Development in this model is simple and easy to understand. Each phase must be completed before another phase of the software development begins. There will be no overlapping phases in this model.

The waterfall model depicts the software development process as a sequential flow of events. This means that any phase of the development process can start only after the previous one has finished. The phases in this waterfall model do not overlap.

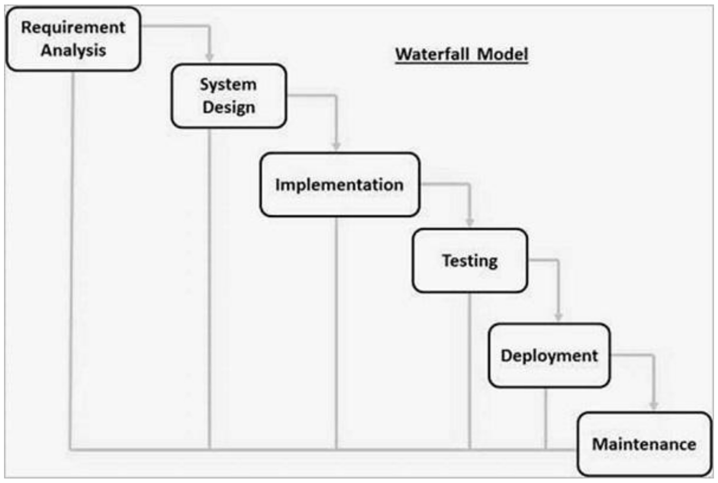


Figure 1: Different phases of Waterfall Model

The Waterfall Model's sequential phases are as follows:

- Requirement Gathering and analysis: This phase captures all possible requirements for the system to be developed and documents them in a requirement specification document.
- System Design: This phase examines the requirements specifications from the previous phase and prepares the system design. This system design aids in defining the overall system architecture as well as specifying hardware and system requirements.
- Implementation: The system is first built as discrete programs called units, which are then merged in the next phase, using inputs from the system design. Unit testing is the process of developing and testing each unit for its functioning.
- Integration and Testing: After each unit has been tested, all of the units built during the implementation phase are merged into a system. The entire system is then tested for any flaws or failures after it has been integrated.
- Deployment of System: The product is deployed in the client environment or released into the market once functional and non-functional testing is completed.
- Maintenance: In the client environment, there are a few challenges that arise. Patches are published to address these vulnerabilities. In order to improve the product, newer versions have been produced. Maintenance is carried out in order to bring about these modifications in the customer's environment.

Benefit of Using Waterfall Model in this Project

Developers of this project are only 2 people which means we have less human resources in terms of development. The Waterfall model is the model which will help us work efficiently and easily. We could define clearly in each stage of Development and we could separate the developing part precisely. Thus, with the people of 2, the process of work and documentation would be well-informed to each other during the implementation process.

Project Planning

<u>In order to complete a successful software project. The actions to take are as follows</u>:

- ❖ Choose a project to work on.
    - ➢ Determining the project's goals and objectives
    - ➢ Comprehending the specifications and requirements.
    - ➢ Analytical, design, and implementation methods.
    - ➢ Techniques for testing
    - ➢ Documentation
- ❖ Project deliverables and milestones
- ❖ Budget allocation
    - ➢ Exceeding limitations while remaining under control.
- ❖ Estimates for the project
    - ➢ Cost
    - ➢ Time
    - ➢ Code Size
- ❖ Resource Allocation
    - ➢ Hardware
    - ➢ Software
    - ➢ Digital Library
- ❖ Risk Management
    - ➢ Previous relevant project information
    - ➢ Risk detection
    - ➢ Risk avoidance

Project scheduling

	September				October				November			
Requirement and gathering	🟩											
Analysis		🟩										
Design			🟩									
Coding				🟩	🟩	🟩	🟩	🟩				
Testing									🟩	🟩	🟩	
Implement												🟩
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4

Estimation of the project's costs:

The cost of software is a modest part of the total cost of a computer system.There are a lot of elements that are taken into account when determining the final cost of software, including human, technical, hardware, and software availability, among others.The sizing of the project was the most important factor to consider during the cost estimation process.Despite thorough software sizing, function points and approximate lines of code were employed to "size" and cost each component of the Software.

My cost estimation for the Project was also based on baseline data gathered from previous projects, which were used in conjunction with estimation.

1) Effort Estimation - This refers to the overall number of man-hours needed to complete the project. It also accounts for the time spent on documentation and the user manual.

2) Hardware Estimation - This comprises the cost of the PCs as well as the hardware required for the project's development.

Requirement Gathering and analysis

In this project, the requirement is a problem statement which is mentioned in the problem statement section in the introduction chapter. Such that the requirement of this project is there and we do analysis and find the solution of such problems.

- **User Requirement:**The main objective of this project is to solve the problem of people using the bus and missing it as the timetable of the bus is untold. The seat is already full and they have not been informed. Thus, they wasted their time waiting for the bus which they were unable to travel by. Hence, we come with the idea of the bus inform the customer whether the bus has the available seats for
- Software Requirement

Name Of Component	Specification
Operating System	Window XP, Window 7, Window 10, MacOS Linux, Android
Languages	Java, SQL
Database	MongoDB
Browser	Mozilla, Opera, Chrome, etc.
Software Development Kit	Android Studios
Scripting Languages Enable	JSP

- Hardware Requirement

Name Of Component	Specification
Processor	Pentium III
RAM	128 MB
Hard Disk	20GB
Monitor	15″ Color Monitor
Keyboard	122 keys

Project Profile

There has been a constant push to build technologies that can make the software development process easier. However, with the emergence of new programming paradigms, today's software engineers have significant challenges in keeping up with rapidly changing technologies. Software re-engineering is regarded as an important process in the software development business, among other things. One of the most important responsibilities here is to comprehend existing software systems and adapt them to a new software environment. In most cases, working through a program written by another programmer necessitates a significant amount of human labor. This project is a novel attempt to handle the problem of program analysis and the development of diagrams that can better portray a program's structure. UML is now widely accepted as an industry standard for software engineering design. It is necessary to have numerous diagramming tools to express various aspects/ features of a program, such as

Use case: Collect user requirements in manageable parts. The construction design is based on supplying a set of use cases for each interaction as the basis for system testing. It displays the static structure of concepts, types, and classes.

Class Diagram : Concepts depict how people perceive the world; type depicts software component interfaces; classes.

Interaction Diagram: depict software component implementation.

Package Diagram:depicts a group of classes and their interdependencies.

State Diagram: depicts how a single object acts across multiple use cases.

Activity Diagram: displays behavior in relation to a control framework. Numerous objects can be displayed across multiple uses, many objects in a single use case, or implementation methods promoted.

UML

The successor to the wave of Object Oriented Analysis and Design (OOA&D) methods that emerged in the late 1980s is the Unified Modeling Language (UML).

It connects Booch's, Rumbaugh's (OMT), and Jacobson's methods in the most direct way. A modeling language, not a method, is what the UML is called. Most methods include both a modeling language and a process, at the very least in theory. The modeling language is a notation that designers use to communicate their ideas.

Class Diagram

Within object-oriented methods, the class diagram technique has become truly central. This technique has been incorporated into almost every method.

The most diverse range of modeling concepts also applies to class diagrams. Although everyone requires the basic elements, advanced concepts are used less frequently. A class diagram depicts the various types of objects in the system, as well as the various types of static relationships that exist between them. Static relationships can be divided into two categories:

- Association
- Subtype

A class diagram also depicts a class's attributes and operations, as well as the constraints that govern how objects are connected.

Package Diagram

The question of how to break down a large system into smaller systems is one of the oldest in software methods. It becomes difficult to comprehend, as well as the modifications we make to them.Structured methods used functional decomposition, in which the entire system was mapped as a function, which was then broken down further into sub functions, and so on. The separation of process data is no longer necessary, and functional decomposition is no longer necessary, but the old question remains. One option is to combine the classes into higher-level units. This concept, which is used very loosely, can be found in a lot of places.objects. Package is the UML term for this grouping mechanism. A diagram that shows packages of classes and their dependencies is referred to as a package diagram.

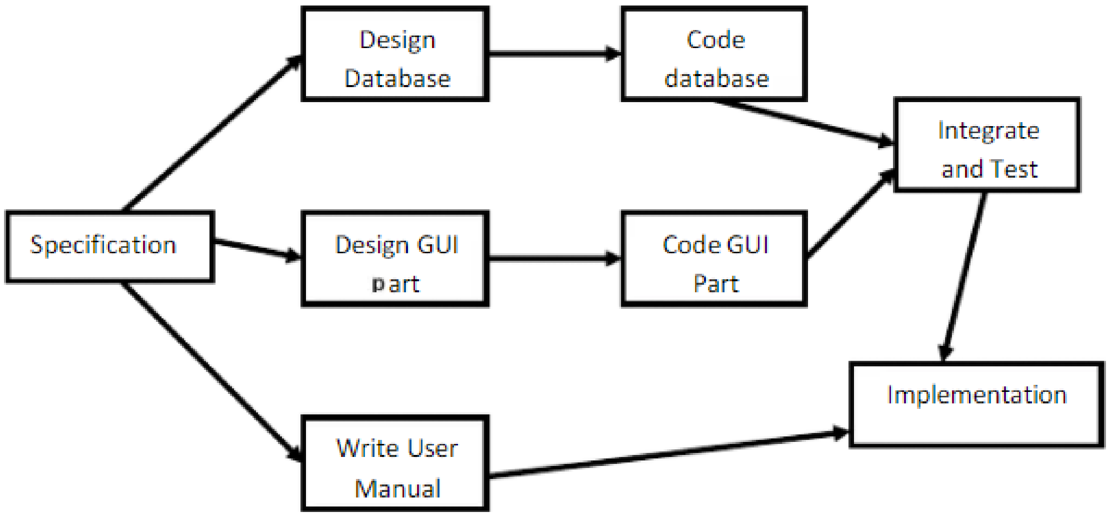
If changes to one element's definition affect the other, the two elements are said to be dependent. Dependencies exist in classes for a variety of reasons: one class sends a message to another; one class contains another as part of its data; one class refers to another as a parameter in an operation. There is a dependency between two packages, as well as between any two classes in the package.

State diagrams

A well-known technique for describing the behavior of a system. They describe all of the possible states in which a given object can be found, as well as how the state of the object changes as a result of events that occur near it. State diagrams are drawn for a single class in most OO techniques to show the lifetime behavior of a single object. State diagrams come in a variety of shapes and sizes, each with its own set of semantics. The state chart created by David Harel is the most widely used in OO techniques.

PERT CHART (Program Evaluation Review Technique)

The PERT chart is used to organize events, activities, and tasks. It's a task-scheduling tool that displays the order of tasks to be completed graphically. It enables the critical path to be calculated. The time and cost associated with a path are calculated, and the path in the critical path requires the most elapsed time.



Project Use Case Model:

Any system's use case model is made up of "use cases." The user can use the system in a variety of ways, which are represented by use cases. Asking the questions "What can the user do with the system?" is a simple way to find all of a system's use cases.

The use cases divide the system's behavior into transactions, with each transaction performing a specific action for the users.

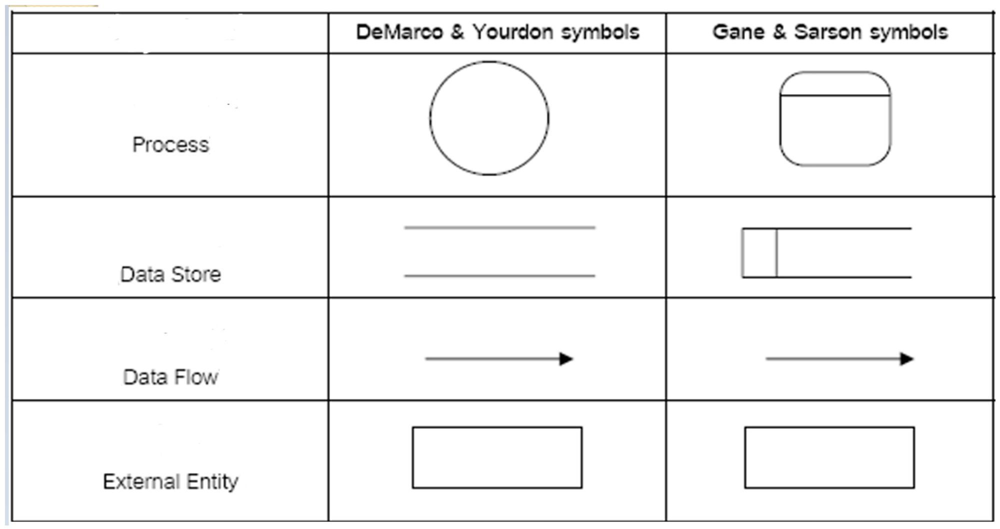
The use case's goal is to define a piece of consistent behavior without revealing the system's internal structure. A use case is a sequence of interactions between a user and a computer system. The normal interaction between the user and the system is represented by a single main line sequence in these interactions. The use case model is a crucial tool for analysis and design (task). Drawing a use case diagram and writing an accompanying text that elaborates on the drawing can be used to represent use cases.

Data Flow Diagram

The data flow diagram, which functionally decomposes the requirements specification, is the starting point of the design phase. A DFD is made up of a series of bubbles that are connected by lines. The lines represent data flows in the system, while the bubbles represent data transformation. A data flow diagram (DFD) describes the flow of data rather than how it is processed, so it does not include hardware, software, or data structure.A data-flow diagram (DFD) depicts the "flow" of data through an information system graphically. DFDs are also useful for visualizing data processing (structured design). A data flow diagram (DFD) is a useful modeling tool for analyzing and building information processes.

DFD literally translates to "a diagram that explains the flow of information in a process." Based on the inputs and outputs, DFD depicts the flow of information in a process. A Process Model is another name for a DFD.

The data flow diagram is a graphical representation of a system's data and how the data is processed and transformed (DFD).



Data flows through an internal process from an external data source or an internal data store to an internal data store or an external data sink on a DFD. It's standard practice to start by drawing a high-level data flow diagram, which depicts the interaction between the system and external agents that serve as data sources and sinks. The system's interactions with the outside world are modeled entirely in terms of data flows across the system boundary on the context diagram (also known as the Level O DFD').

The context diagram depicts the entire system as a single process with no indication of how it is organized internally.

This context-level DFD is then "exploded" to produce a Level 1 DFD that depicts some of the system's details. The Level 1 DFD depicts how the system is divided into subsystems (processes), each of which handles one or more data flows to or from an external agent, and which together provide the system's entire functionality. As level 2 DFD, the level 1 DFD is further spread out and divided into more descriptive and detailed descriptions of the project. The level 2 DFD can be made up of a number of data flows that will eventually display the entire software project description.

SYSTEM DESIGN AND ANALYSIS

System Analysis

It is the process of gathering and interpreting data, identifying problems, and breaking down a system into its constituent parts.A system analysis is carried out to investigate a system or its components in order to determine its goals. It is a problem-solving technique that improves the system and ensures that all of the system's components work together to achieve their goals.The system's behavior is defined by the analysis.

System Design

It is the process of defining the components or modules of a new business system or replacing an existing system in order to meet specific requirements. Before you begin planning, you must first thoroughly understand the old system and determine how computers can be used most effectively in order to operate efficiently.The goal of system design is to figure out how to achieve the system's goal.The main focus of System Analysis and Design (SAD) is on

- Systems
- Processes
- Technology

Detail of System Design in Project

The system is designed to be the most feasible as possible for both admin and customer the detail of it will be explain in Use case diagram and Data Flow Diagram blow

The goal of the proposed system of bus ticket booking

to develop a system with improved facilities. The proposed system can overcome all of the existing system's flaws. The system ensures proper security while also reducing manual labor.

- Data security is important.

- Ascertain data accuracy.
- Control of higher officials in a proper manner.
- Reduce the amount of manual data entry.
- The amount of time required for various processing is reduced, resulting in increased efficiency.
- Better customer service.
- Interactivity and user friendliness.
- The bare minimum of time is required.

Use Case Diagram

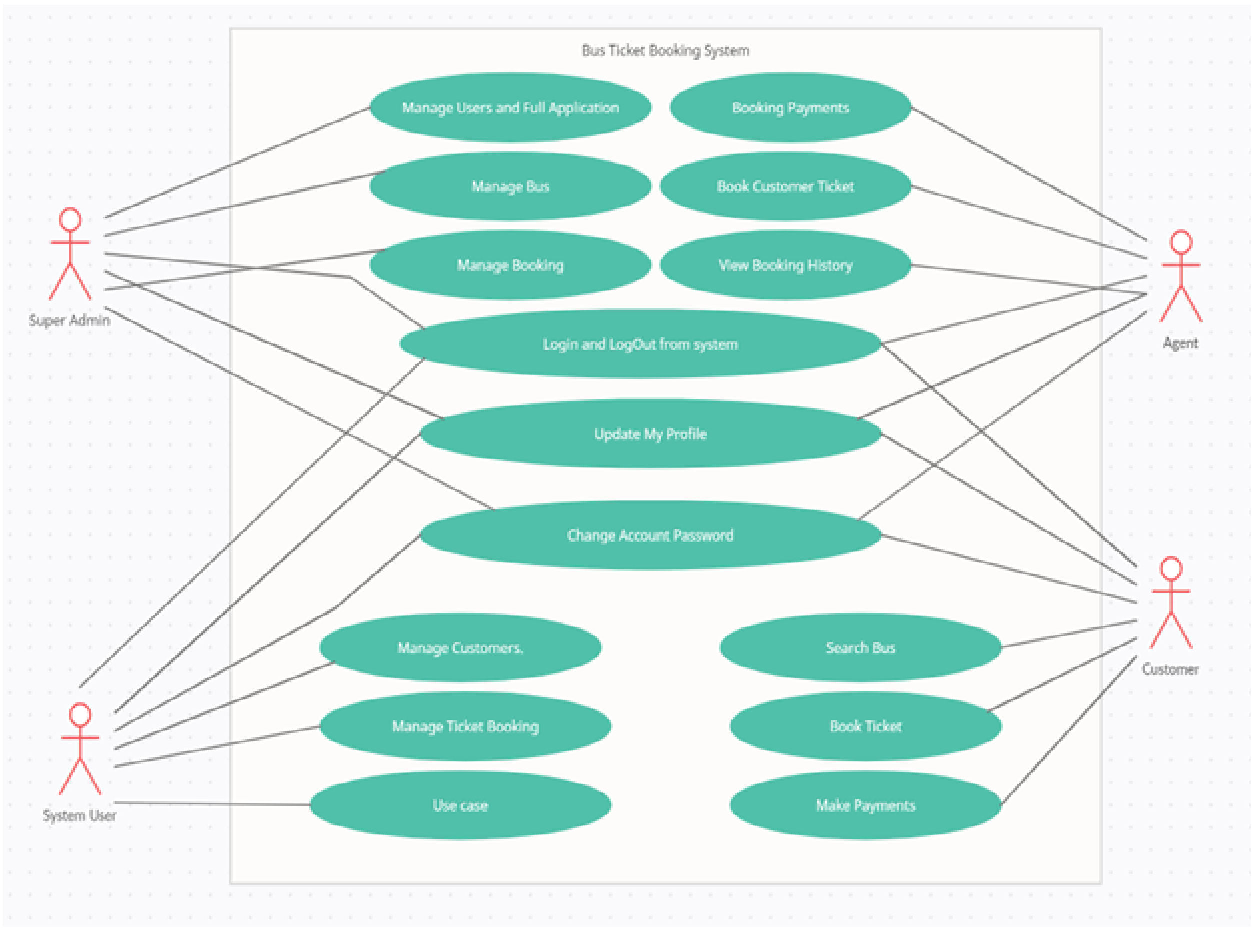


Figure 2: Use case diagram

Data Flow Diagram Level 0

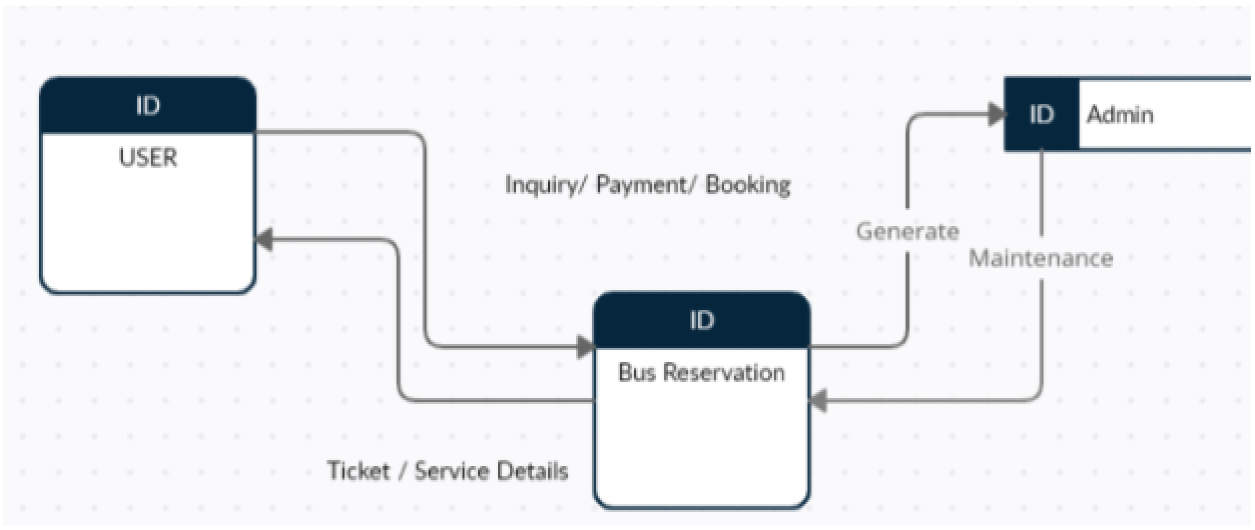


Figure 3: Data Flow Diagram Level 0

Data Flow Diagram Level 1

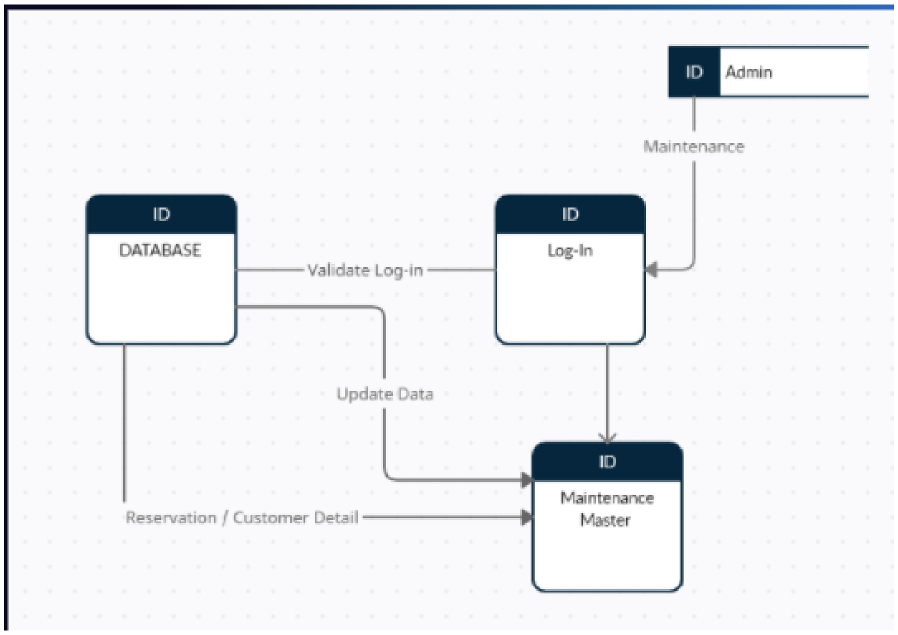


Figure 4: Data Flow Diagram Level 1

IMPLEMENTATION

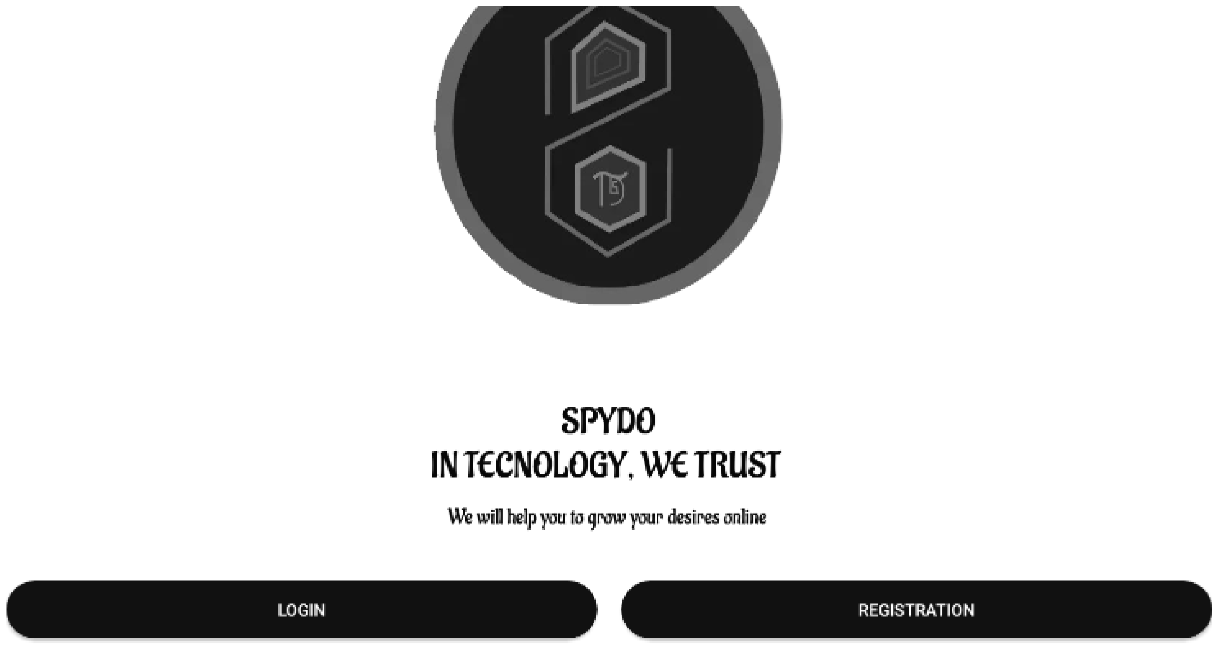
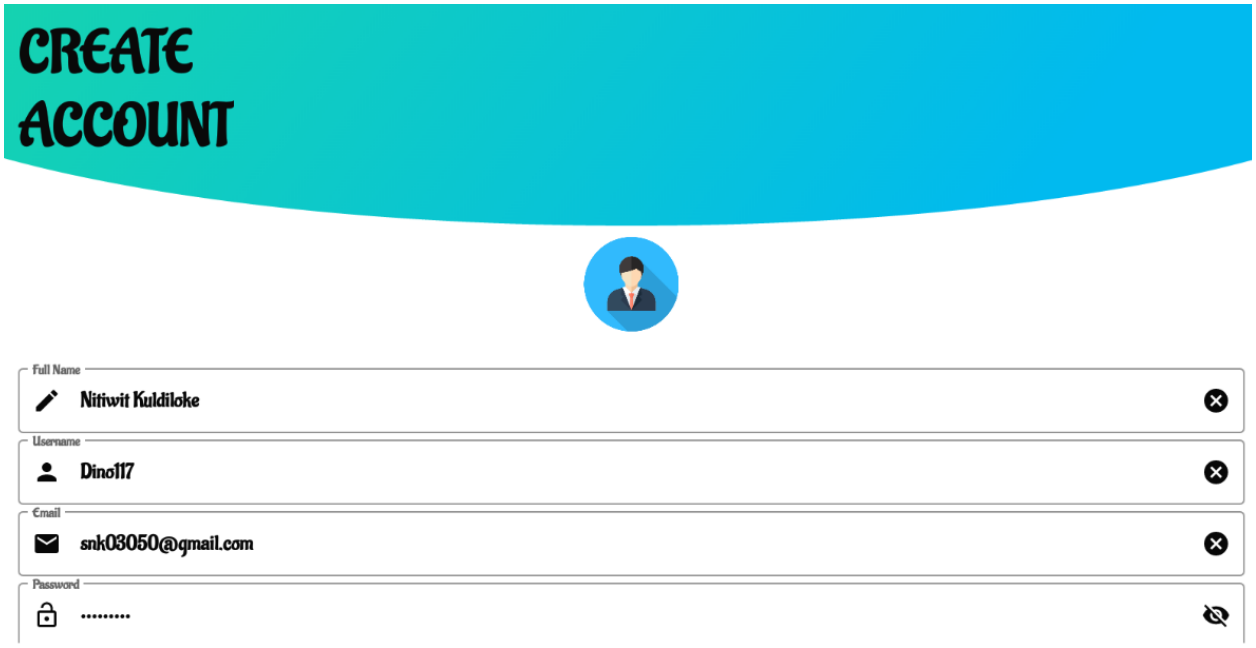


Figure 5: Front Page


CREATE ACCOUNT

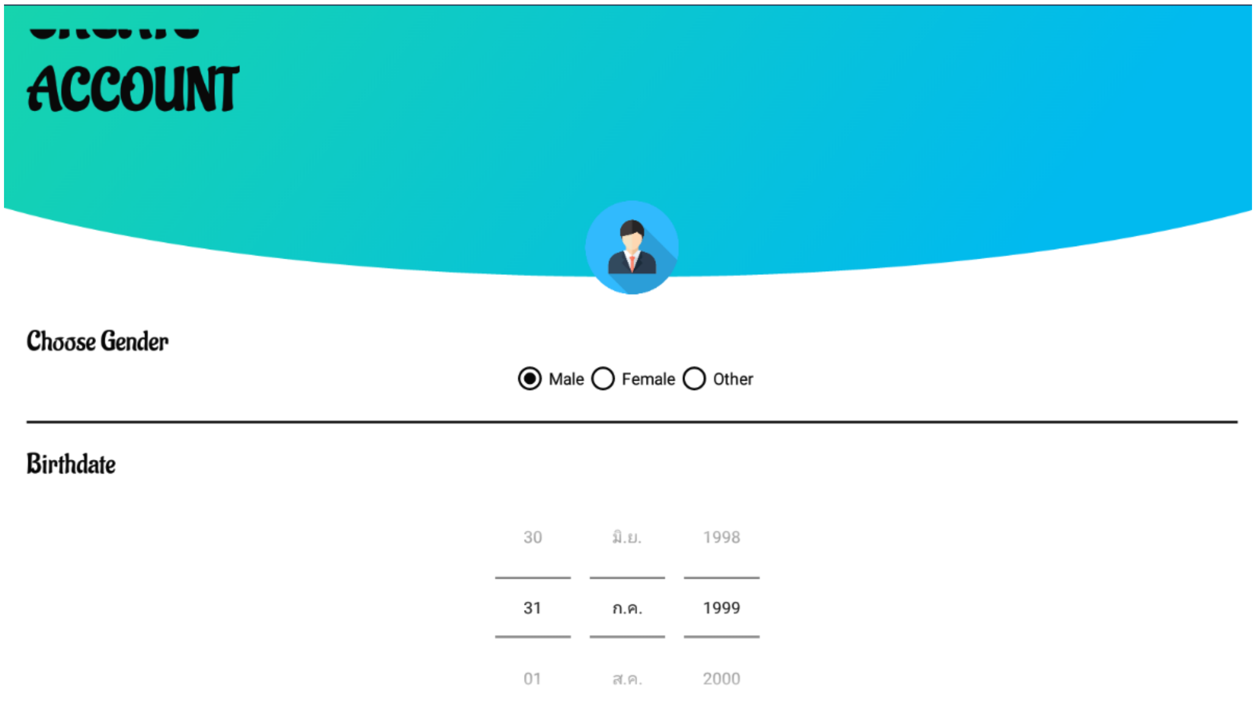
Full Name: Nitiwit Kuldiloke

Username: Dino117

Email: snk03050@gmail.com

Password: •••••••••

Figure 6: Registration


ACCOUNT

Choose Gender

[✓ Male] Female Other

Birthdate

30	มิ.ย.	1998
31	ก.ค.	1999
01	ส.ค.	2000

Figure 7: Registration

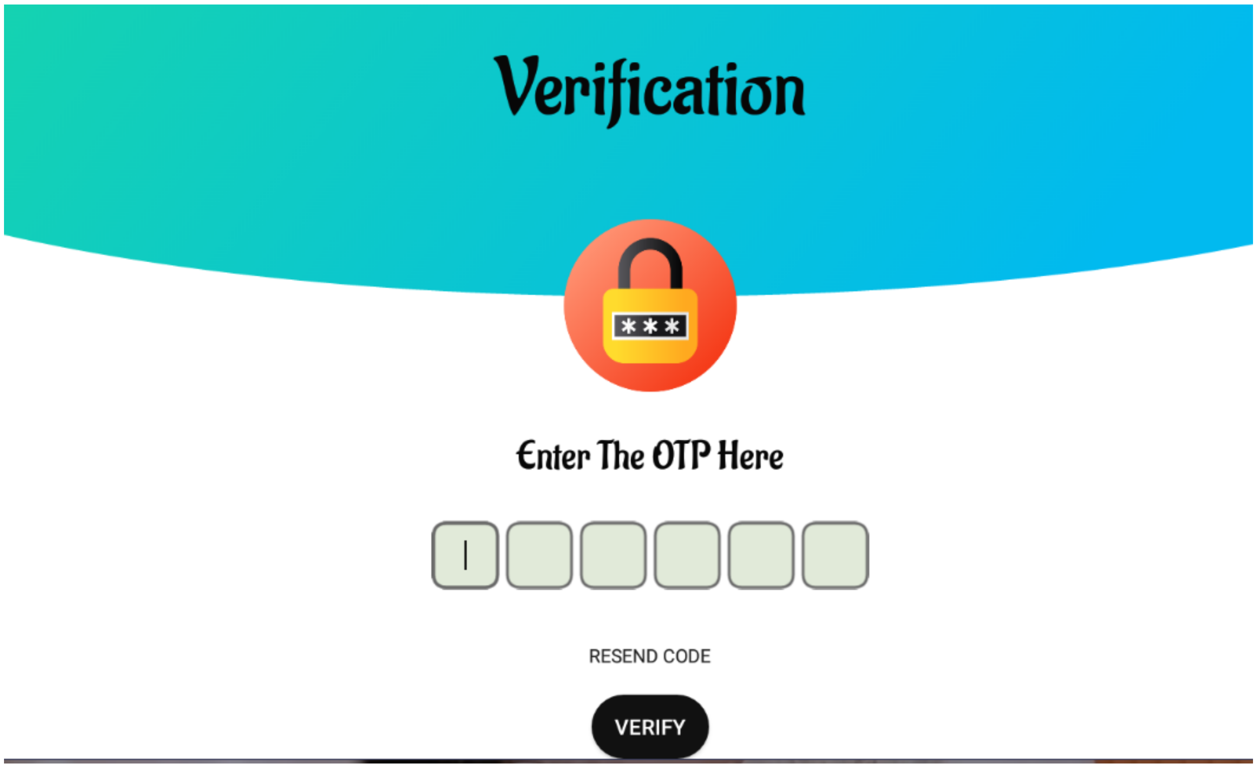


Figure 8: Verification OTP

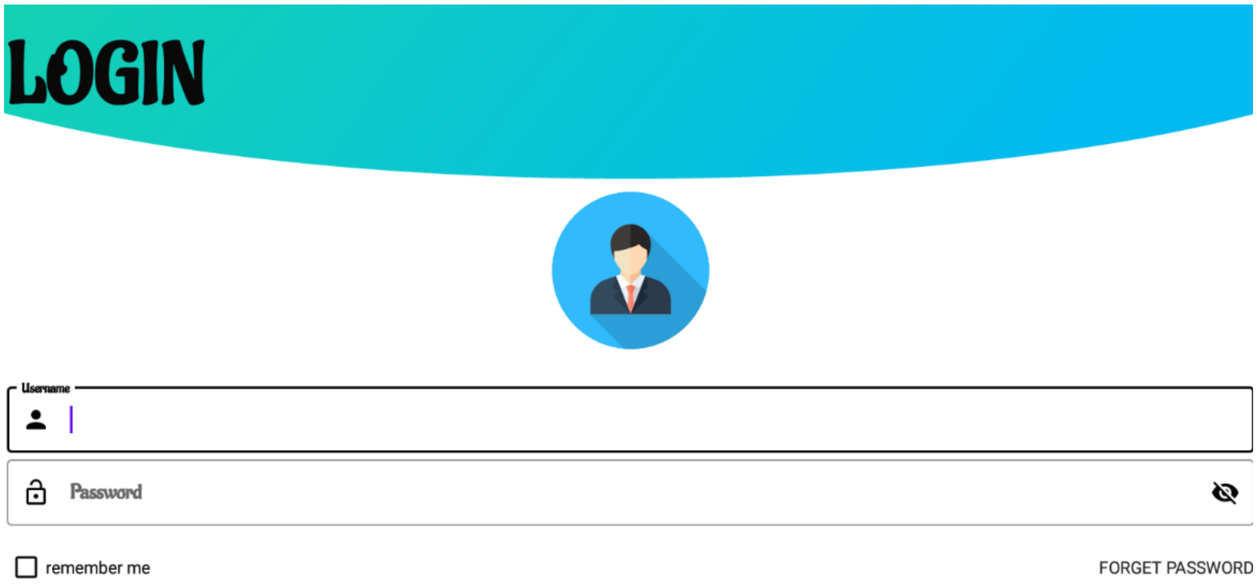


Figure 9: Log In

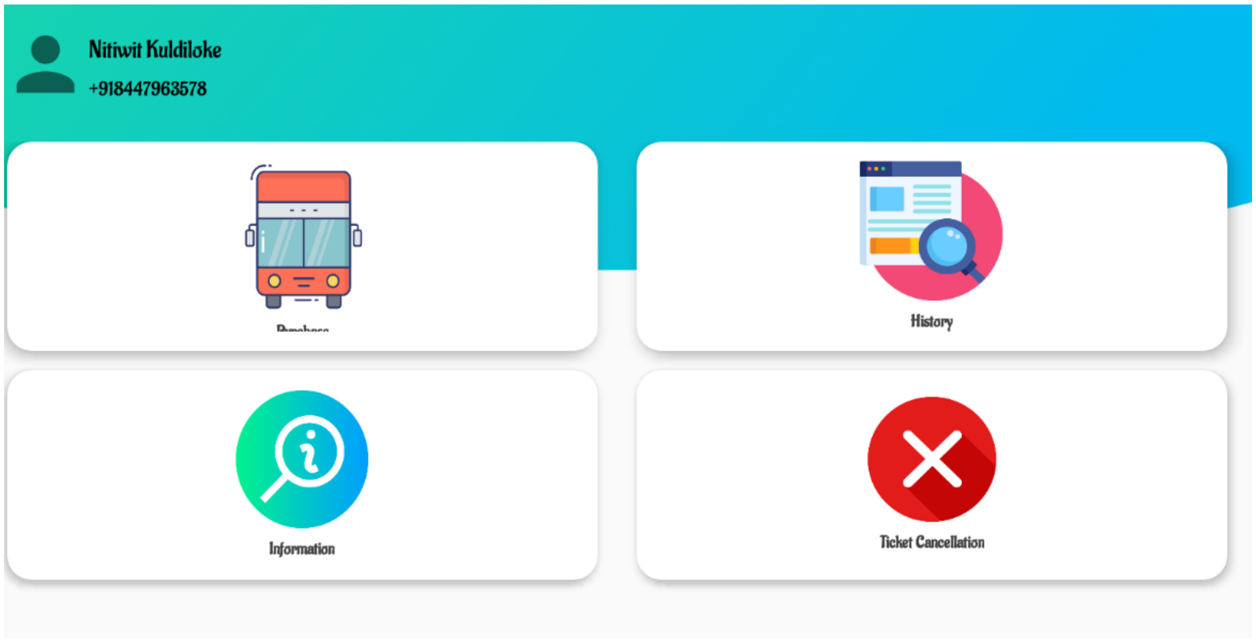


Figure 10: Main Menu

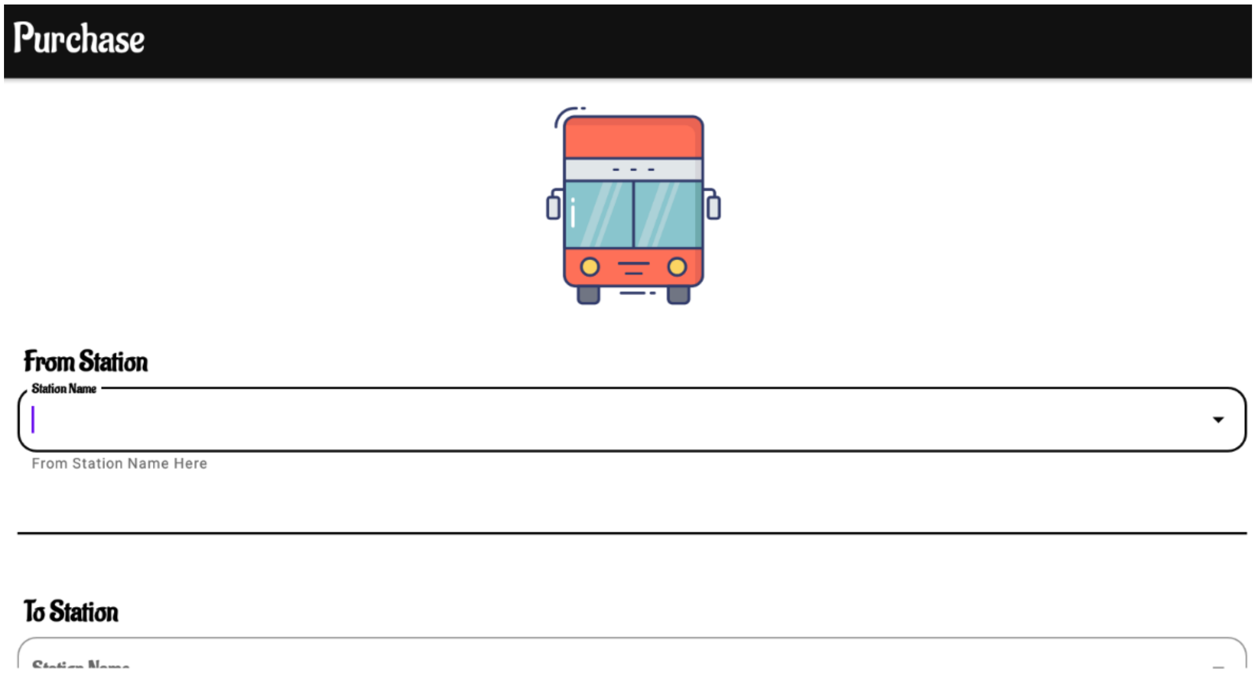


Figure 11: Reservation

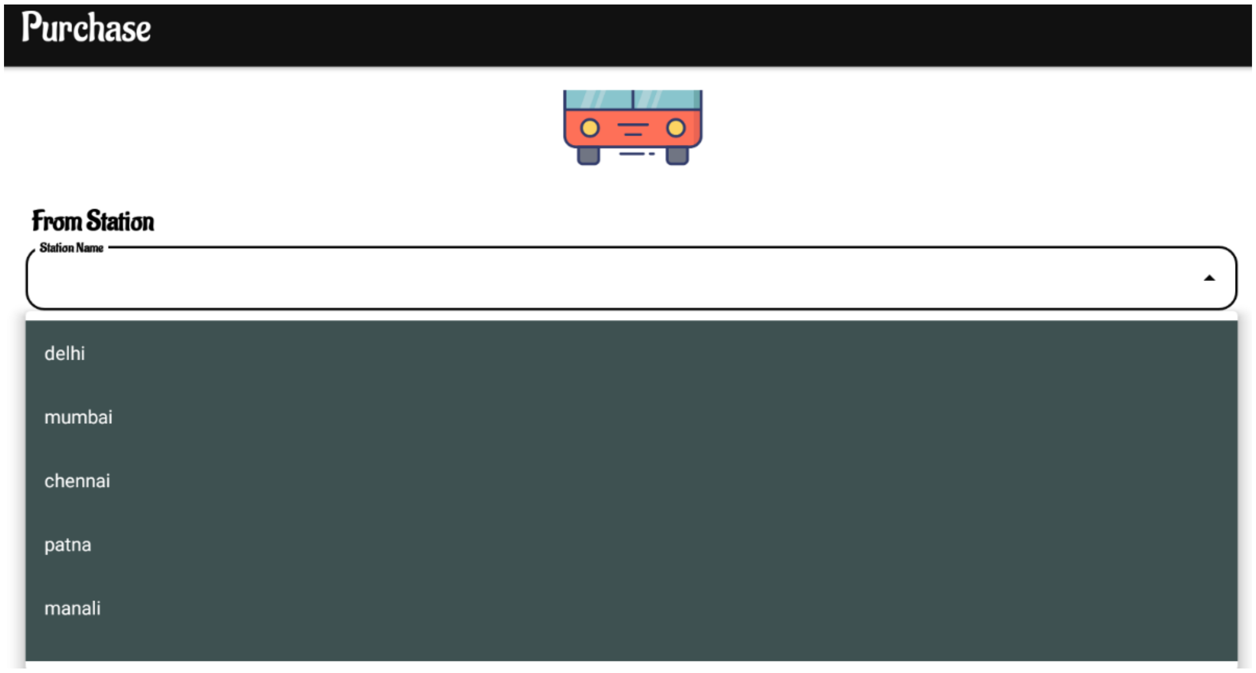


Figure 12: Reservation Selection Destination

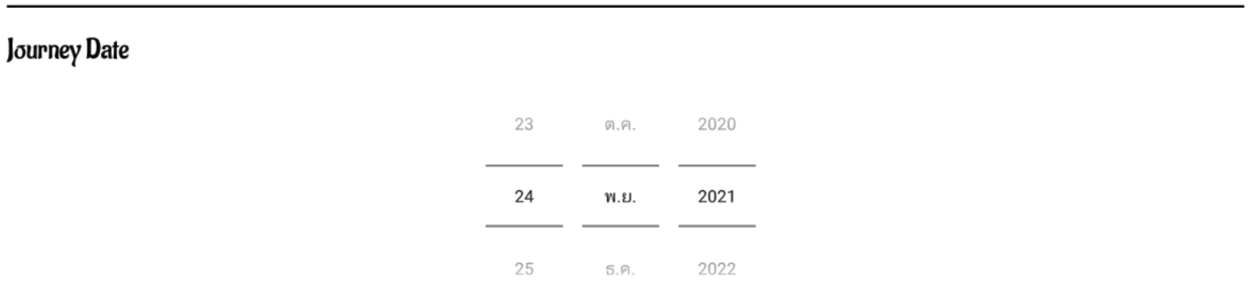
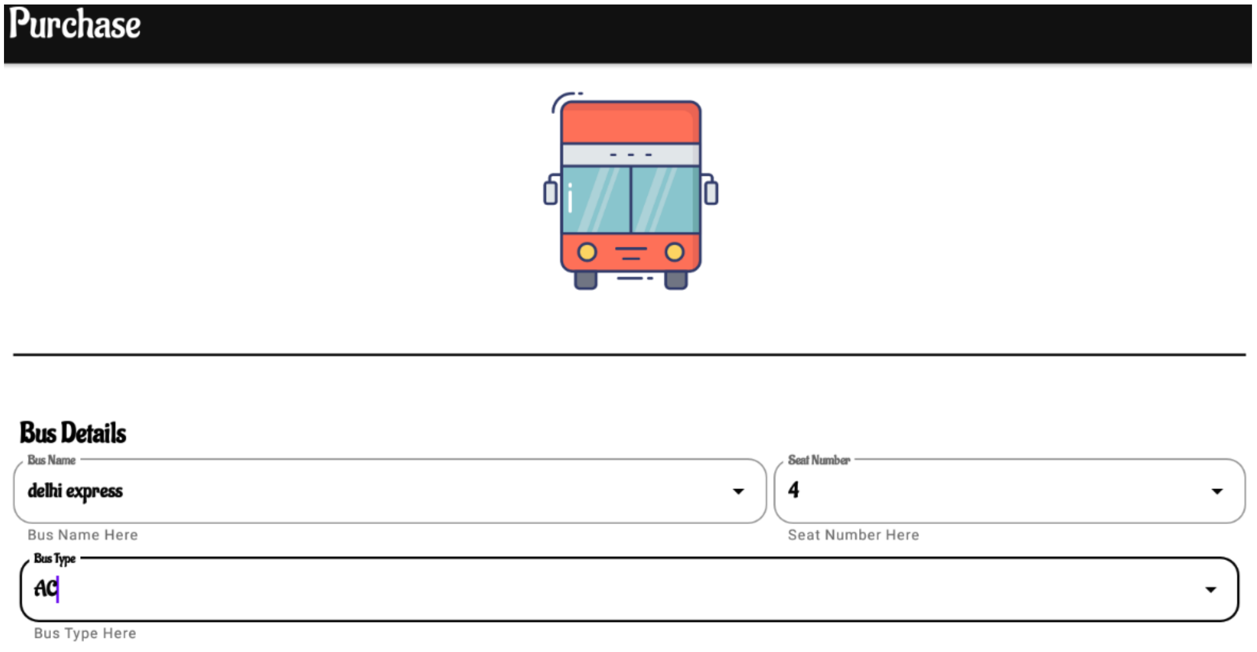


Figure 13: Reservation Date



Purchase

Bus Details

Bus Name

delhi express

Bus Name Here

Seat Number

4

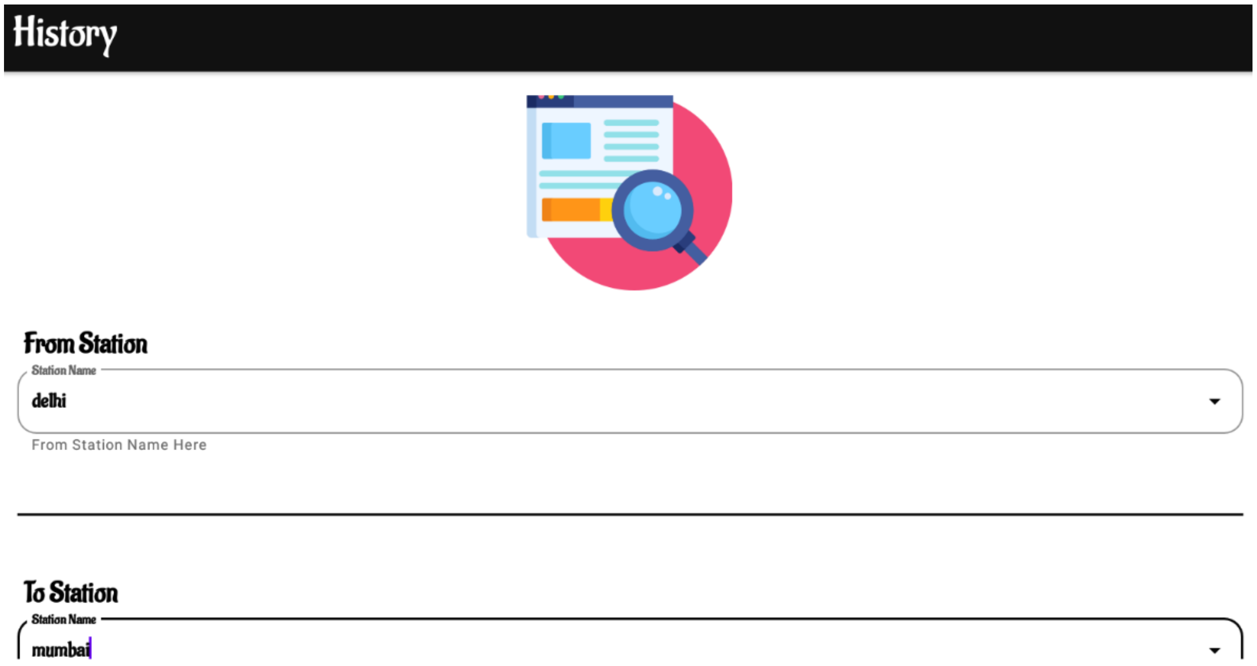
Seat Number Here

Bus Type

AC

Bus Type Here

Figure 14: Reservation Bus type



History

From Station

Station Name

delhi

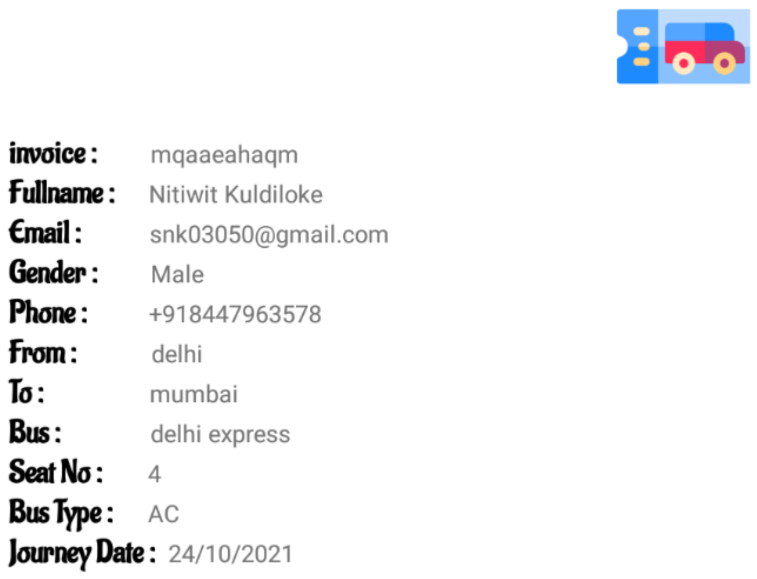
From Station Name Here

To Station

Station Name

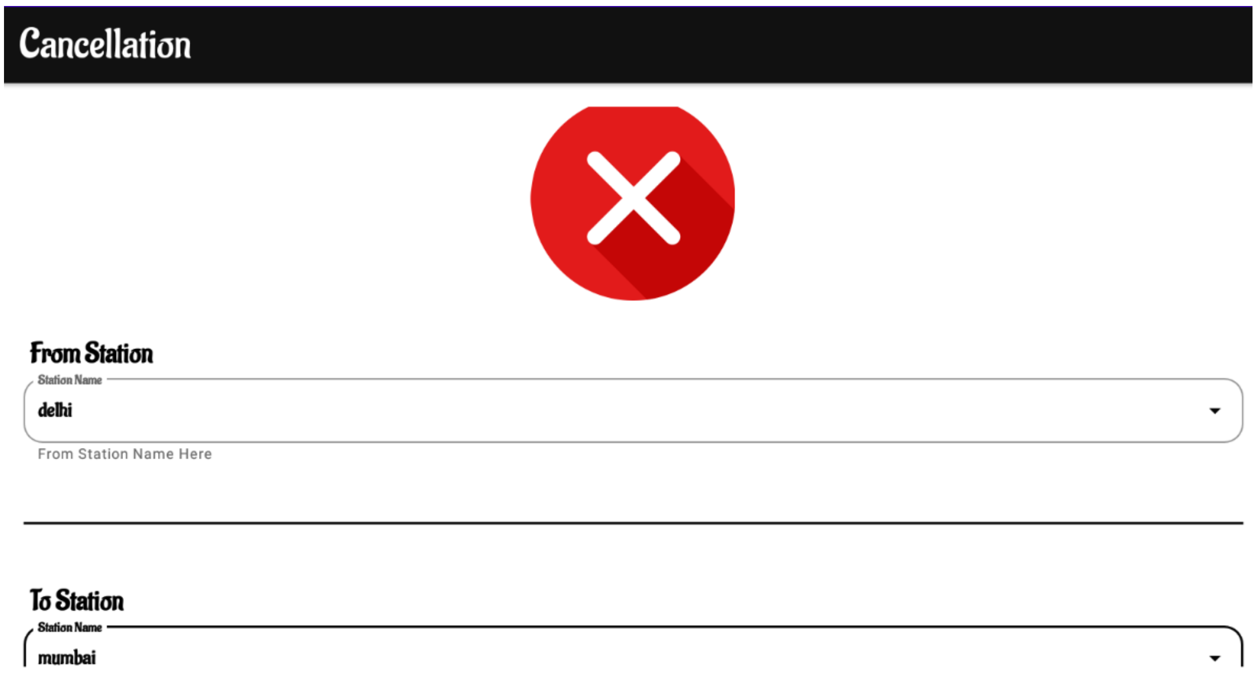
mumbai

Figure 15: History



invoice :	mqaaeahaqm
Fullname :	Nitiwit Kuldiloke
Email :	snk03050@gmail.com
Gender :	Male
Phone :	+918447963578
From :	delhi
To :	mumbai
Bus :	delhi express
Seat No :	4
Bus Type :	AC
Journey Date :	24/10/2021

Figure 16: Ticket from History



Cancellation

From Station

Station Name

delhi

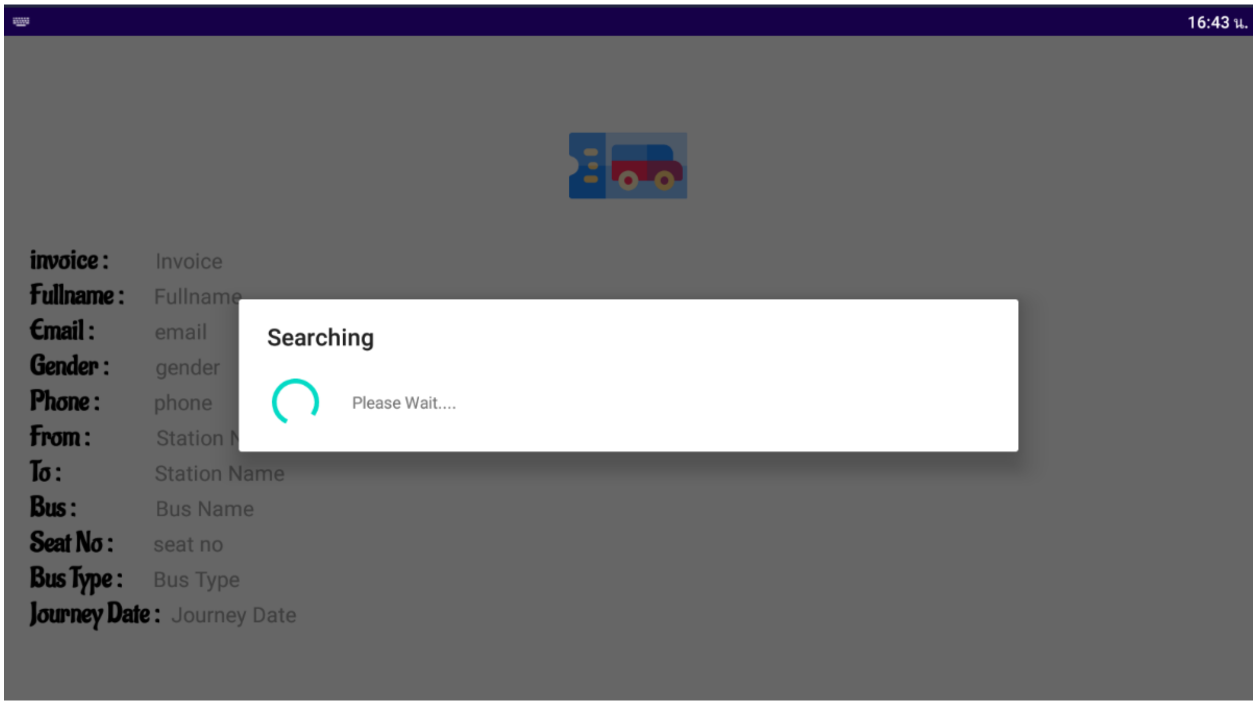
From Station Name Here

To Station

Station Name

mumbai

Figure 17: Cancellation of Ticket

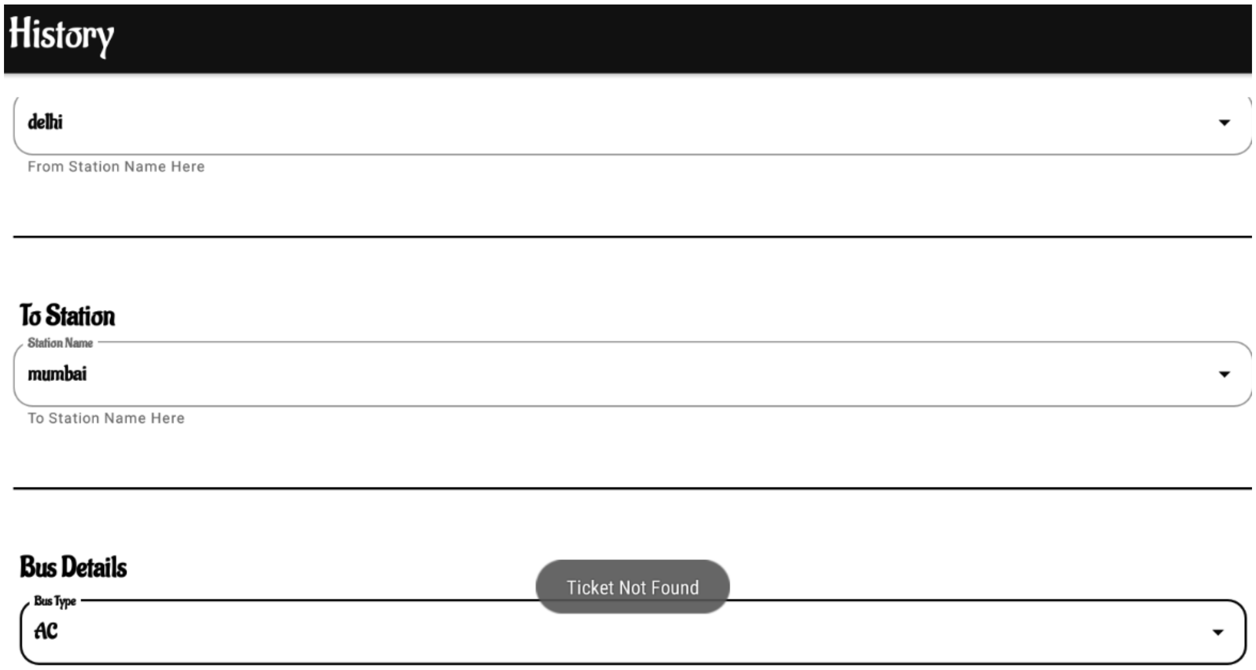

16:43 น.

invoice : Invoice
Fullname : Fullname
Email : email
Gender : gender
Phone : phone
From : Station N
To : Station Name
Bus : Bus Name
Seat No : seat no
Bus Type : Bus Type
Journey Date : Journey Date

Searching

Please Wait....

Figure 18: Cancellation in Progress


History

delhi

From Station Name Here

To Station

Station Name

mumbai

To Station Name Here

Bus Details

Bus Type

AC

Ticket Not Found

Figure 19: Ticket Cancelled

CONCLUSION/FUTURE SCOPE

As the working process took months, we could accomplish both of our goals for our project. We could finish implementation of application which solve the problem of ticket booking application and together we could learn the concept of software engineering and process of developing the software project which will make us be better in future project.

FUTURE SCOPE

There is still an available to be developed in this project. The live location feature could be provided in the future if we could contact the bus company or authority to give the bus a track thus the live time-stamp of the bus will be there. And the app could be available to perform multi choice of payment and transaction such as PayPals, Credit Card, Visa, or else. The future of technology is not finished yet; they have the long road to go. We could maintain it and develop it at the same time too.

The following is the feature which could be improved in the future:

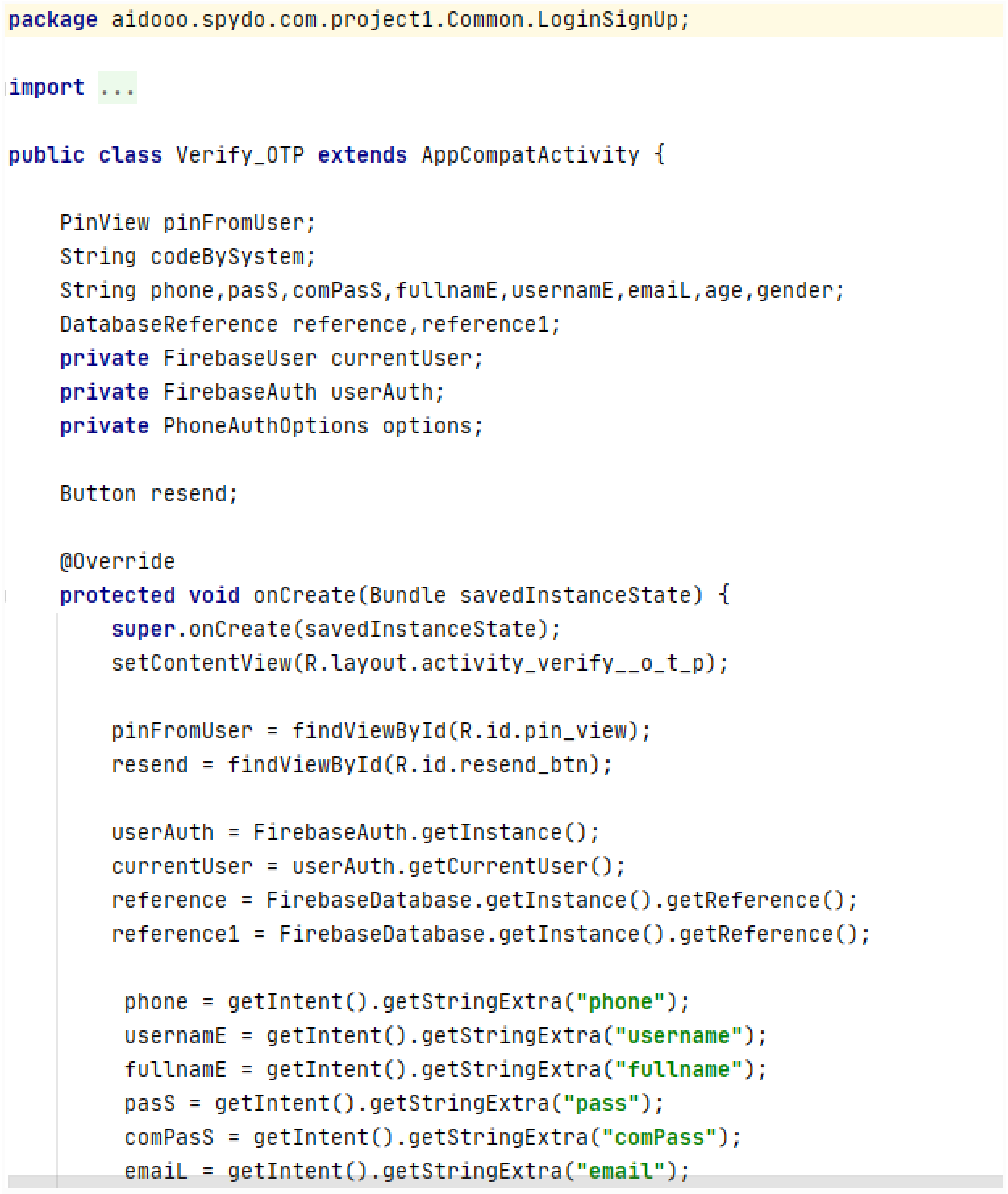
- Live tracking of the bus: For those who are waiting for the bus it will help them to time manage themselves.
- Deploy web application: Not to restricted in only mobile even those people who are more available with PC might could book the ticket by themselves
- Develop in iOS: This project is available in Android Version of phone only.
- More Security in Database: We could provide 2 factor authentication in the application.

REFERENCE

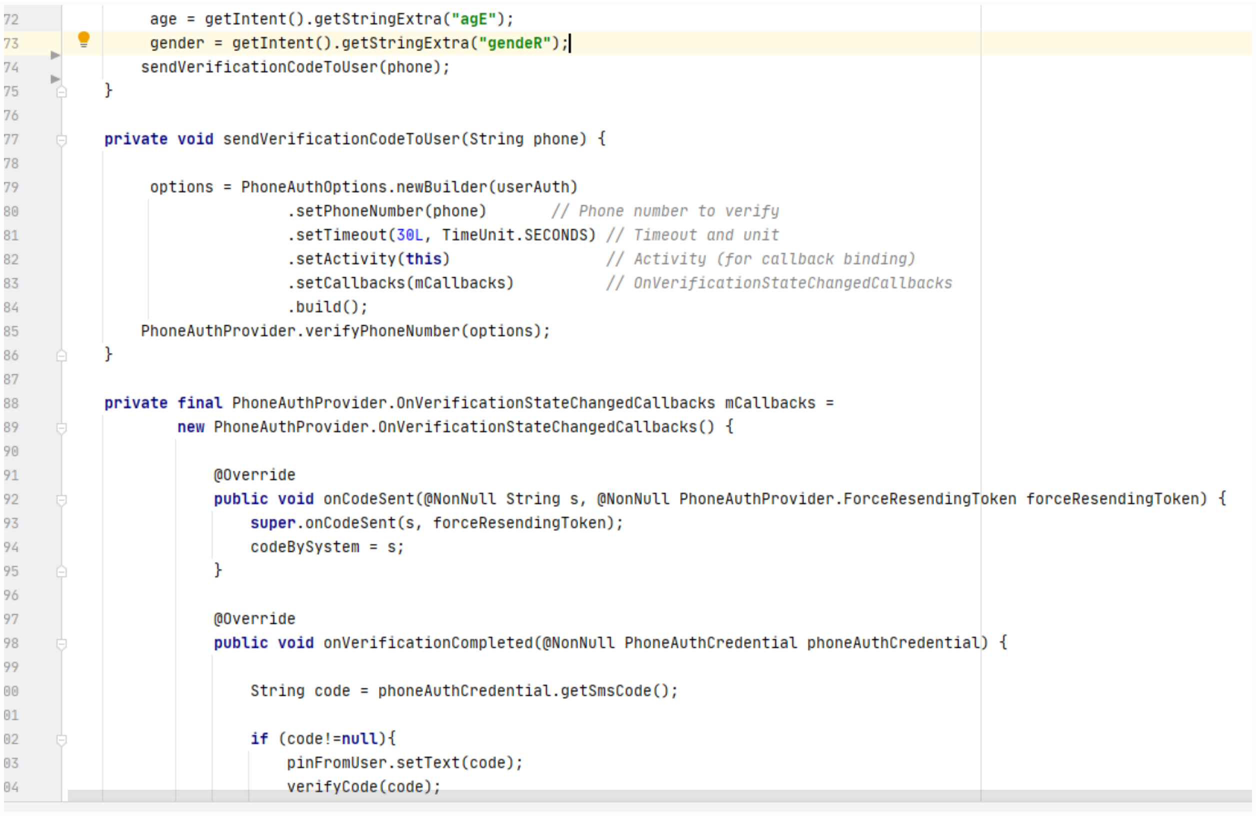
1. ***System Analysis and Design - Overview*. (n.d.). Tutorialspoint. Retrieved November 2, 2021, from** https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_overview.htm
2. ***What is a Data Flow Diagram* (n.d.). Lucidchart. Retrieved November 5, 2021, from https://www.lucidchart.com/pages/data-flow-diagram**
3. **Shiklo, B. (2021, June 23). *8 Software Development Models: Sliced, Diced and Organized in Charts*. ScienceSoft. Retrieved October 15, 2021, from https://www.scnsoft.com/blog/software-development-models**
4. **Grzelak M., Napierała Ł., Napierała Ł., Karovič V., Ivanochko I. (2020) Bus Ticket Reservation System Agile Methods of Projects Management. In: Barolli L., Nishino H., Miwa H. (eds) Advances in Intelligent Networking and Collaborative Systems. INCoS 2019. Advances in Intelligent Systems and Computing, vol 1035. Springer, Cham. https://doi.org/10.1007/978-3-030-29035-1_48**
5. **Nwakanma, Ifeanyi & Etus, Chukwuemeka & Ajere, Ikenna & Agomuo, Uchechukwu. (2015). Online Bus Ticket Reservation System. Statistics and Computing. Vol. 1.**
6. **Harini, K. & Saithri, A. & Shruthi, M.. (2021). Smart Digital Bus Ticketing System. 10.1007/978-981-15-8221-9_79.**

IMPORTANT CODE IN USE FOR DEVELOPING

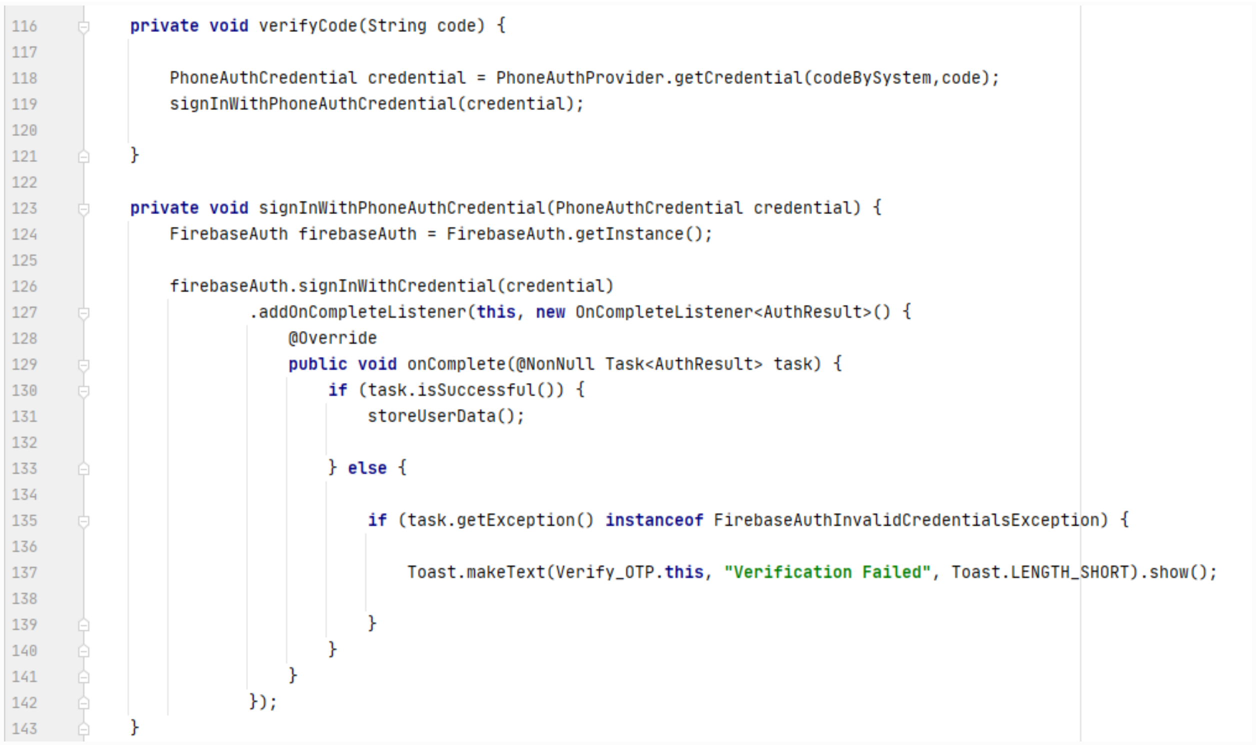
Authentication Library Implementation



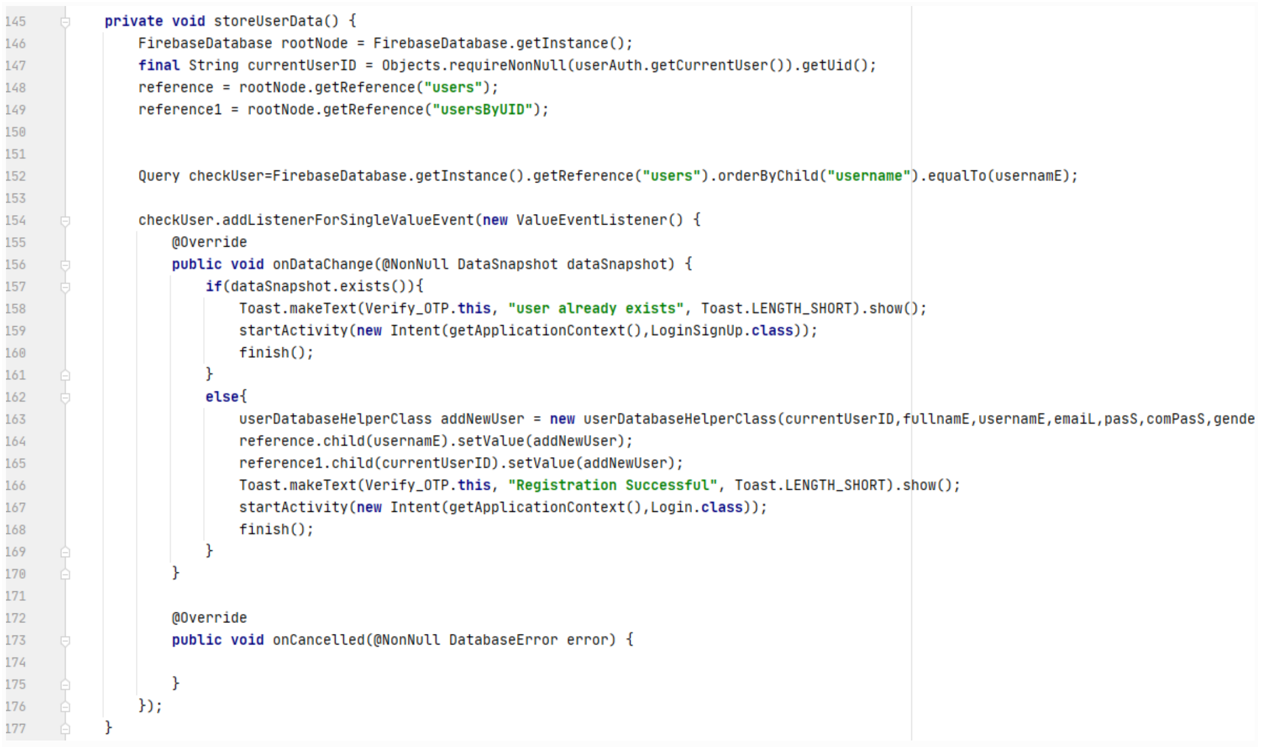
```
package aidooo.spydo.com.project1.Common.LoginSignUp;

import ...

public class Verify_OTP extends AppCompatActivity {

    PinView pinFromUser;
    String codeBySystem;
    String phone,pasS,comPasS,fullnamE,usernamE,emaiL,age,gender;
    DatabaseReference reference,reference1;
    private FirebaseUser currentUser;
    private FirebaseAuth userAuth;
    private PhoneAuthOptions options;

    Button resend;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_verify__o_t_p);

        pinFromUser = findViewById(R.id.pin_view);
        resend = findViewById(R.id.resend_btn);

        userAuth = FirebaseAuth.getInstance();
        currentUser = userAuth.getCurrentUser();
        reference = FirebaseDatabase.getInstance().getReference();
        reference1 = FirebaseDatabase.getInstance().getReference();

         phone = getIntent().getStringExtra("phone");
         usernamE = getIntent().getStringExtra("username");
         fullnamE = getIntent().getStringExtra("fullname");
         pasS = getIntent().getStringExtra("pass");
         comPasS = getIntent().getStringExtra("comPass");
         emaiL = getIntent().getStringExtra("email");
```



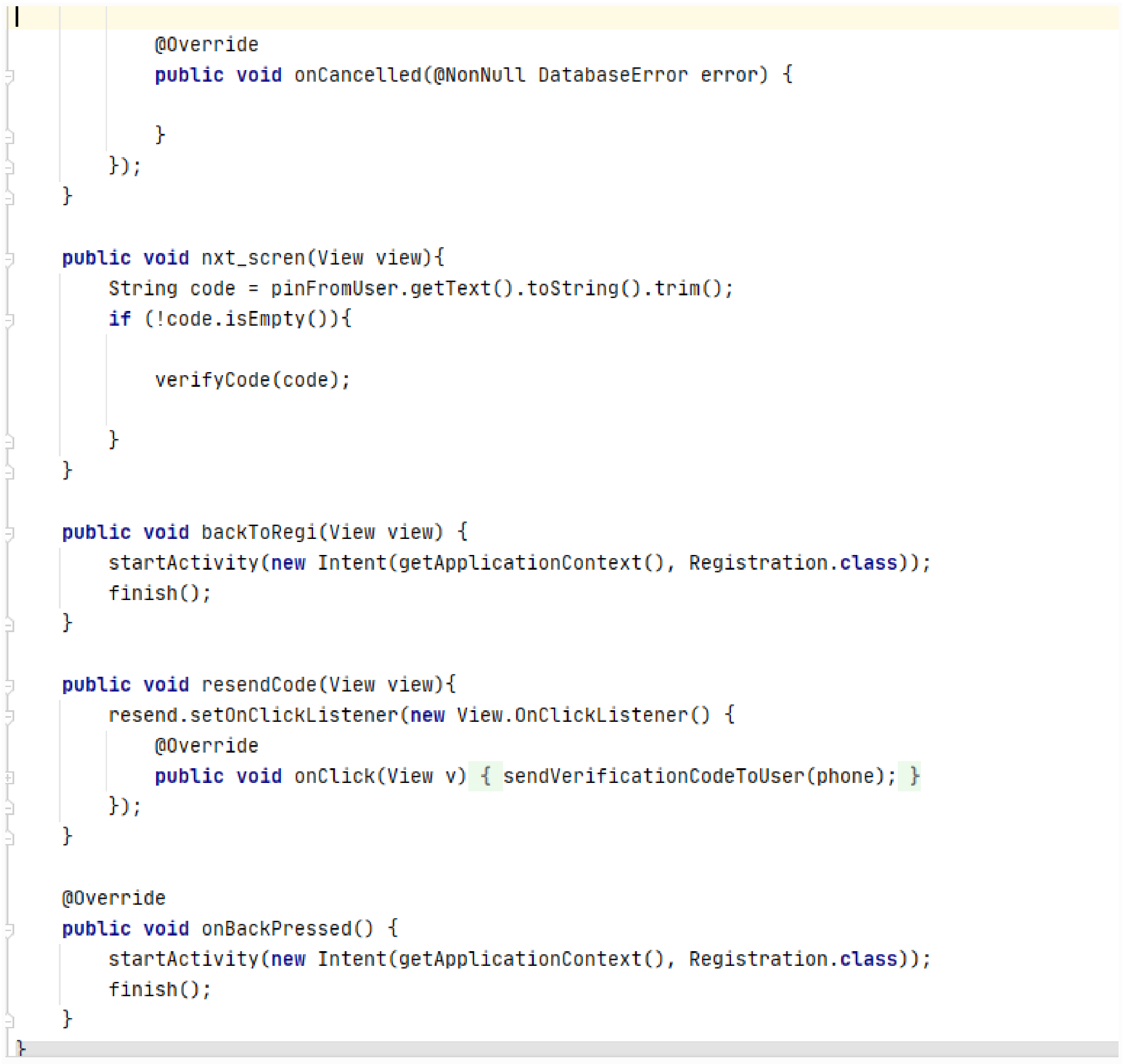
```
        age = getIntent().getStringExtra("agE");
        gender = getIntent().getStringExtra("gendeR");
        sendVerificationCodeToUser(phone);
    }

    private void sendVerificationCodeToUser(String phone) {

        options = PhoneAuthOptions.newBuilder(userAuth)
                        .setPhoneNumber(phone)       // Phone number to verify
                        .setTimeout(30L, TimeUnit.SECONDS) // Timeout and unit
                        .setActivity(this)                 // Activity (for callback binding)
                        .setCallbacks(mCallbacks)          // OnVerificationStateChangedCallbacks
                        .build();
        PhoneAuthProvider.verifyPhoneNumber(options);
    }

    private final PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks =
            new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

                @Override
                public void onCodeSent(@NonNull String s, @NonNull PhoneAuthProvider.ForceResendingToken forceResendingToken) {
                    super.onCodeSent(s, forceResendingToken);
                    codeBySystem = s;
                }

                @Override
                public void onVerificationCompleted(@NonNull PhoneAuthCredential phoneAuthCredential) {

                    String code = phoneAuthCredential.getSmsCode();

                    if (code!=null){
                        pinFromUser.setText(code);
                        verifyCode(code);
```



```
    private void verifyCode(String code) {

        PhoneAuthCredential credential = PhoneAuthProvider.getCredential(codeBySystem,code);
        signInWithPhoneAuthCredential(credential);

    }

    private void signInWithPhoneAuthCredential(PhoneAuthCredential credential) {
        FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();

        firebaseAuth.signInWithCredential(credential)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            storeUserData();

                        } else {

                            if (task.getException() instanceof FirebaseAuthInvalidCredentialsException) {

                                Toast.makeText(Verify_OTP.this, "Verification Failed", Toast.LENGTH_SHORT).show();

                            }
                        }
                    }
                });
    }
```

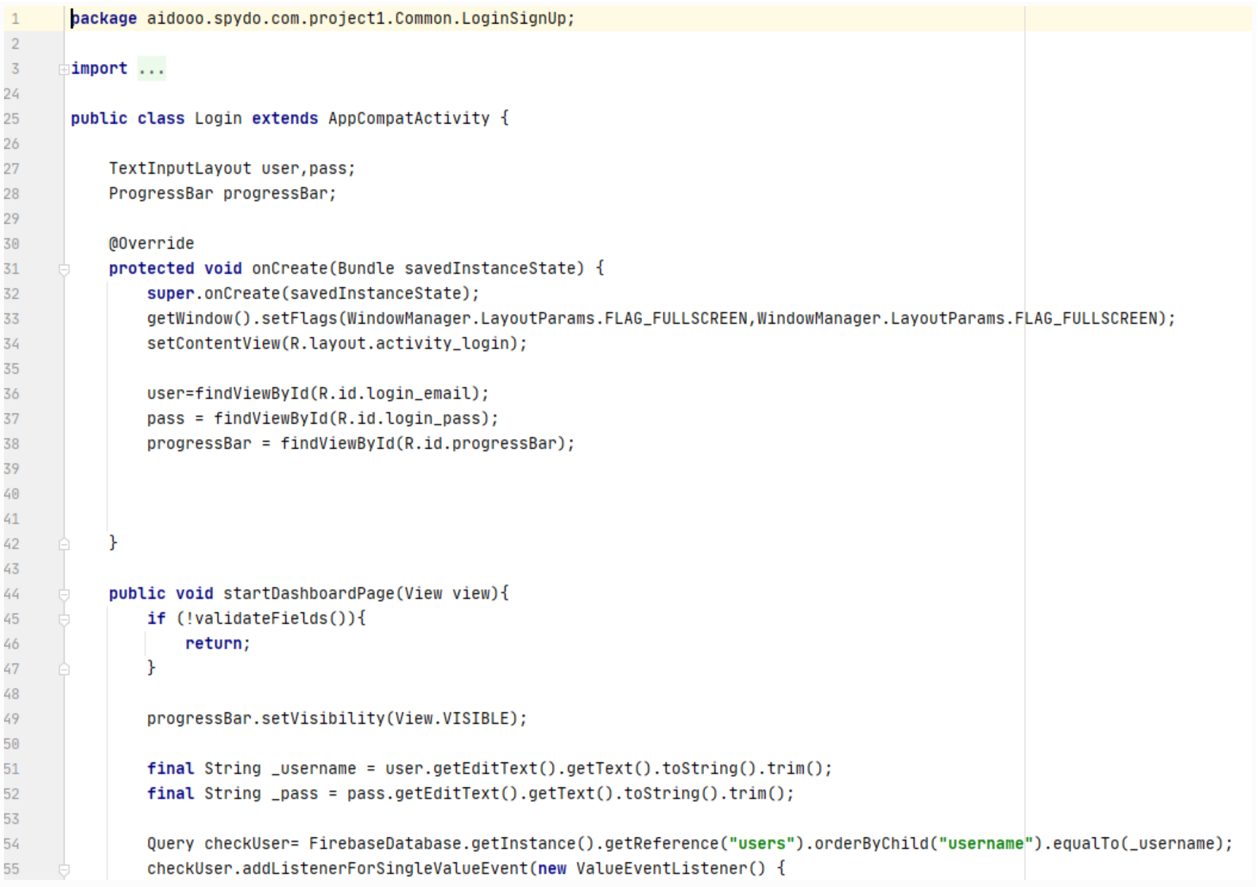


```
private void storeUserData() {
    FirebaseDatabase rootNode = FirebaseDatabase.getInstance();
    final String currentUserID = Objects.requireNonNull(userAuth.getCurrentUser()).getUid();
    reference = rootNode.getReference("users");
    reference1 = rootNode.getReference("usersByUID");


    Query checkUser=FirebaseDatabase.getInstance().getReference("users").orderByChild("username").equalTo(usernamE);

    checkUser.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()){
                Toast.makeText(Verify_OTP.this, "user already exists", Toast.LENGTH_SHORT).show();
                startActivity(new Intent(getApplicationContext(),LoginSignUp.class));
                finish();
            }
            else{
                userDatabaseHelperClass addNewUser = new userDatabaseHelperClass(currentUserID,fullnamE,usernamE,emaiL,pasS,comPasS,gende
                reference.child(usernamE).setValue(addNewUser);
                reference1.child(currentUserID).setValue(addNewUser);
                Toast.makeText(Verify_OTP.this, "Registration Successful", Toast.LENGTH_SHORT).show();
                startActivity(new Intent(getApplicationContext(),Login.class));
                finish();
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```

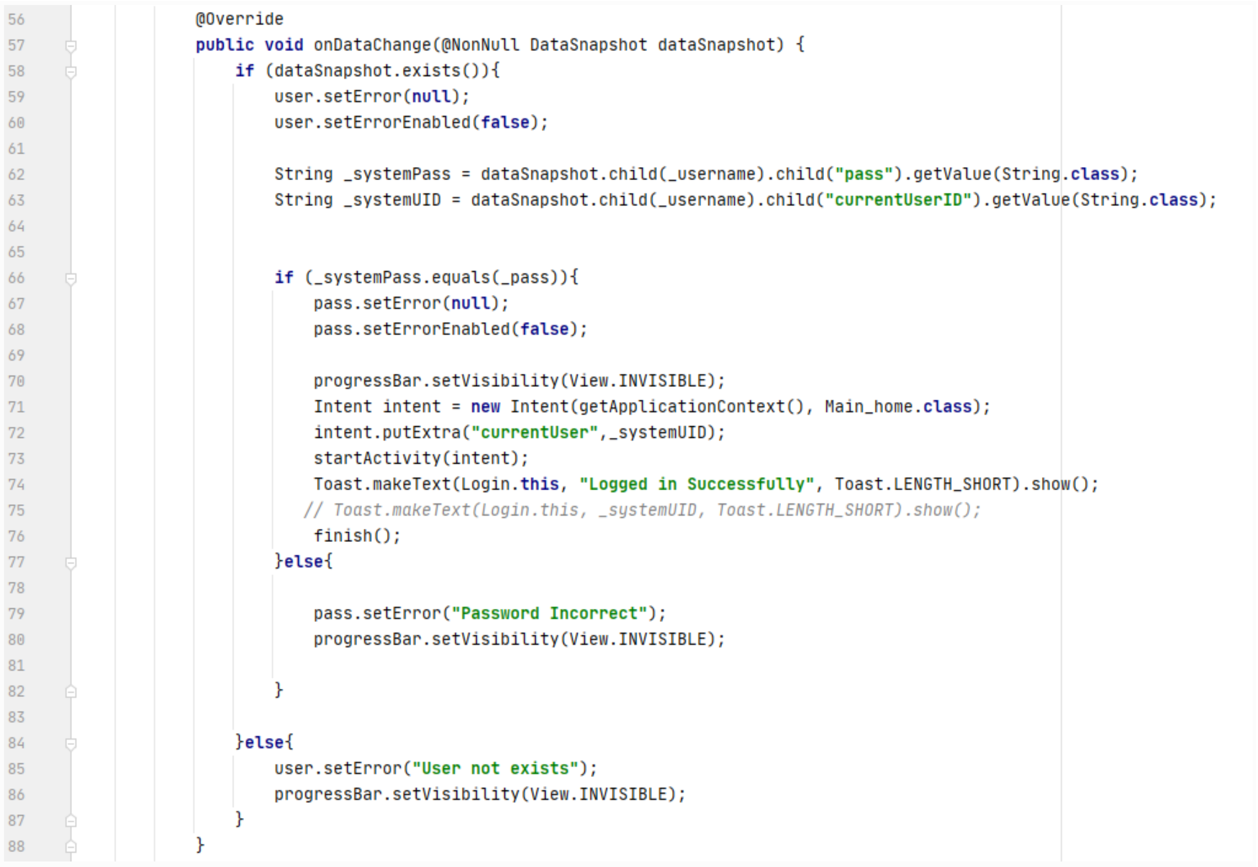


```
            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }

    public void nxt_scren(View view){
        String code = pinFromUser.getText().toString().trim();
        if (!code.isEmpty()){

            verifyCode(code);

        }
    }

    public void backToRegi(View view) {
        startActivity(new Intent(getApplicationContext(), Registration.class));
        finish();
    }

    public void resendCode(View view){
        resend.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { sendVerificationCodeToUser(phone); }
        });
    }

    @Override
    public void onBackPressed() {
        startActivity(new Intent(getApplicationContext(), Registration.class));
        finish();
    }
}
```

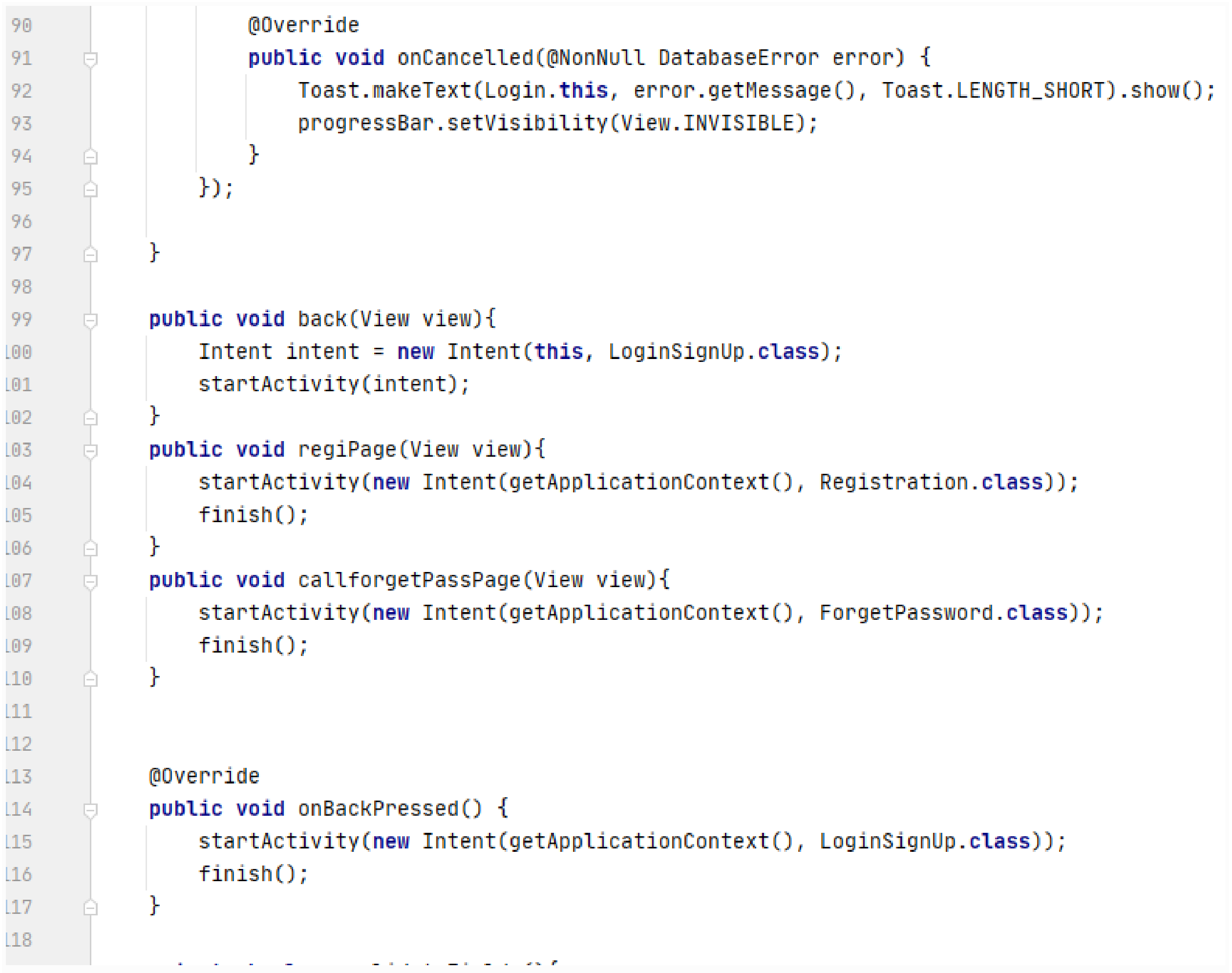
Sign In



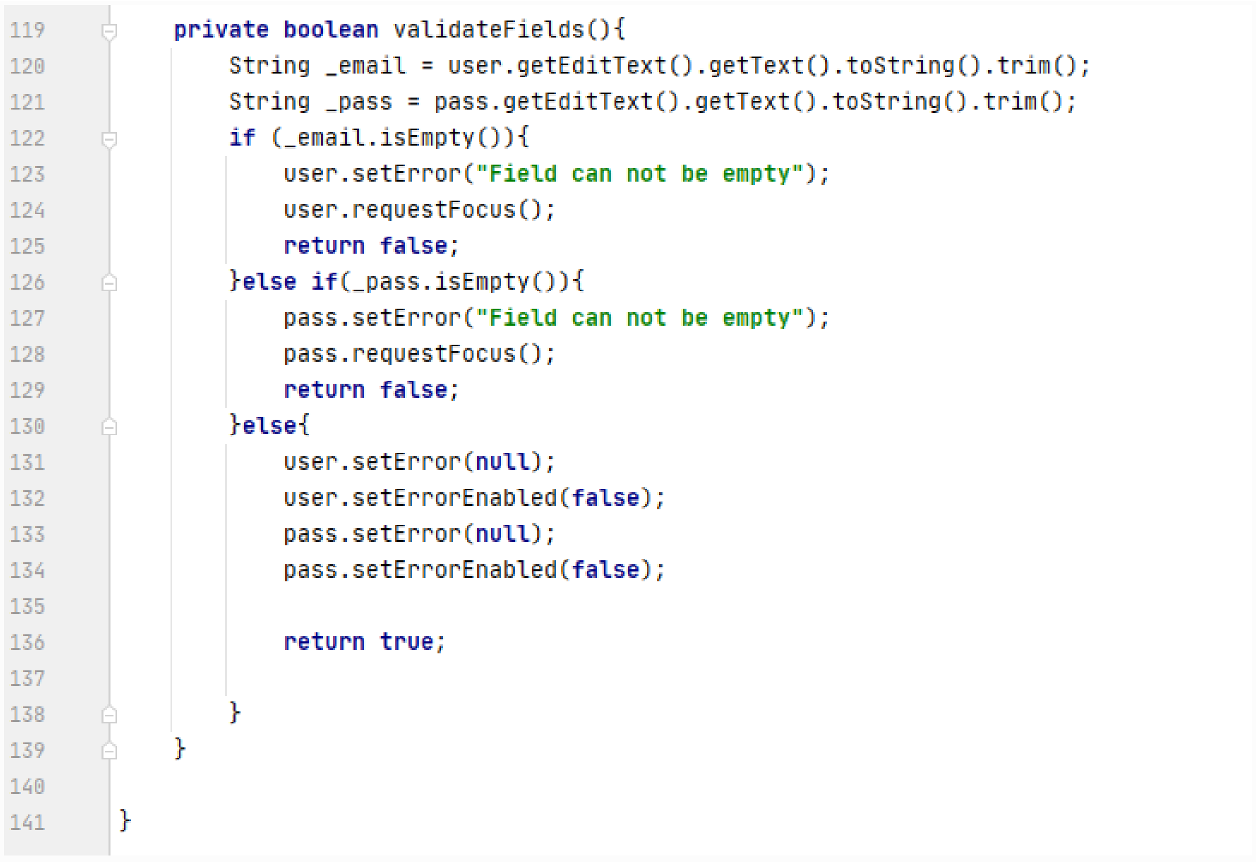
```
package aidooo.spydo.com.project1.Common.LoginSignUp;

import ...

public class Login extends AppCompatActivity {

    TextInputLayout user,pass;
    ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_login);

        user=findViewById(R.id.login_email);
        pass = findViewById(R.id.login_pass);
        progressBar = findViewById(R.id.progressBar);



    }

    public void startDashboardPage(View view){
        if (!validateFields()){
            return;
        }

        progressBar.setVisibility(View.VISIBLE);

        final String _username = user.getEditText().getText().toString().trim();
        final String _pass = pass.getEditText().getText().toString().trim();

        Query checkUser= FirebaseDatabase.getInstance().getReference("users").orderByChild("username").equalTo(_username);
        checkUser.addListenerForSingleValueEvent(new ValueEventListener() {
```



```
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists()){
                user.setError(null);
                user.setErrorEnabled(false);

                String _systemPass = dataSnapshot.child(_username).child("pass").getValue(String.class);
                String _systemUID = dataSnapshot.child(_username).child("currentUserID").getValue(String.class);


                if (_systemPass.equals(_pass)){
                    pass.setError(null);
                    pass.setErrorEnabled(false);

                    progressBar.setVisibility(View.INVISIBLE);
                    Intent intent = new Intent(getApplicationContext(), Main_home.class);
                    intent.putExtra("currentUser",_systemUID);
                    startActivity(intent);
                    Toast.makeText(Login.this, "Logged in Successfully", Toast.LENGTH_SHORT).show();
                   // Toast.makeText(Login.this, _systemUID, Toast.LENGTH_SHORT).show();
                    finish();
                }else{

                    pass.setError("Password Incorrect");
                    progressBar.setVisibility(View.INVISIBLE);

                }

            }else{
                user.setError("User not exists");
                progressBar.setVisibility(View.INVISIBLE);
            }
        }
```

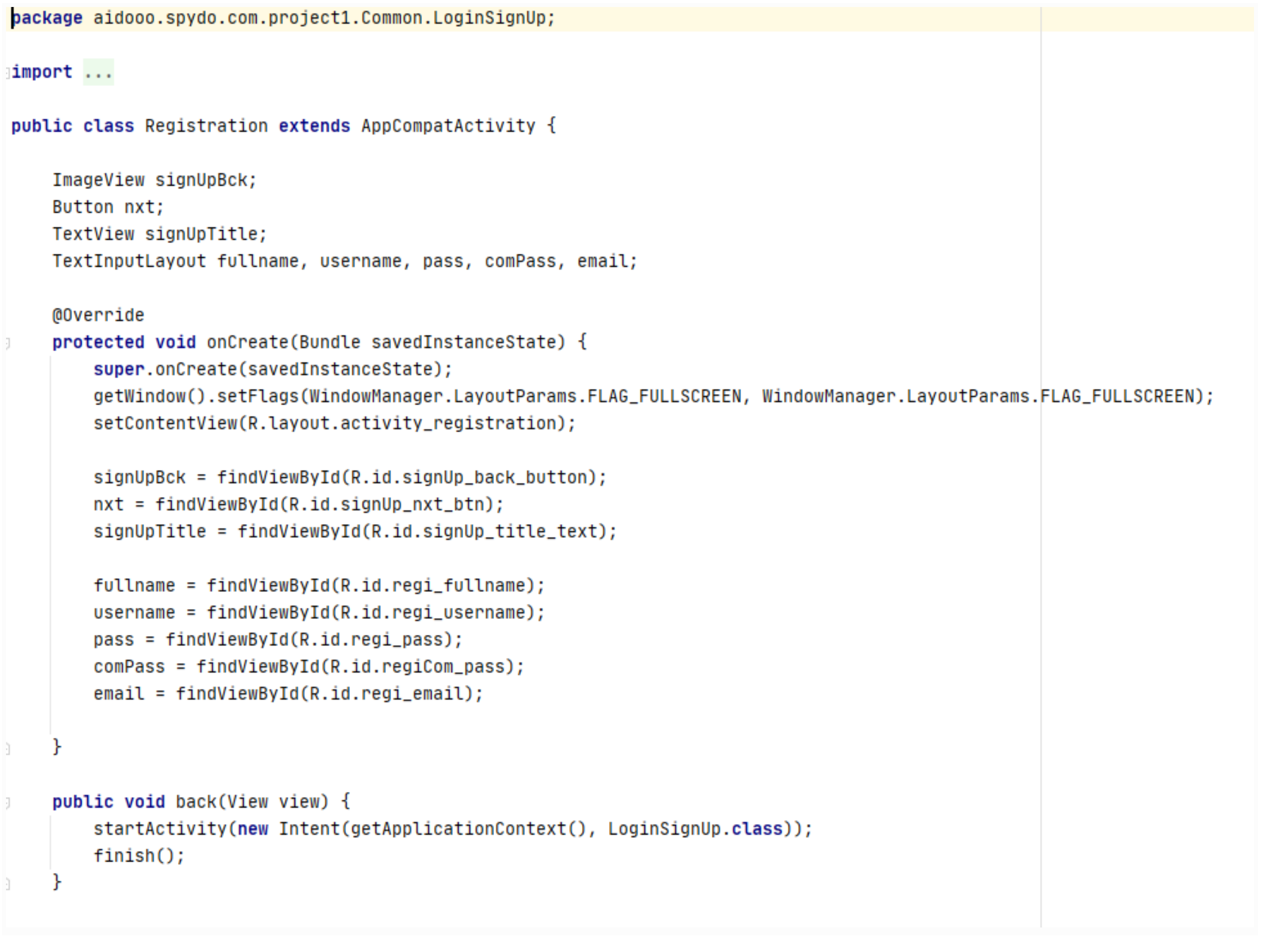


```
                @Override
                public void onCancelled(@NonNull DatabaseError error) {
                    Toast.makeText(Login.this, error.getMessage(), Toast.LENGTH_SHORT).show();
                    progressBar.setVisibility(View.INVISIBLE);
                }
            });

        }

        public void back(View view){
            Intent intent = new Intent(this, LoginSignUp.class);
            startActivity(intent);
        }
        public void regiPage(View view){
            startActivity(new Intent(getApplicationContext(), Registration.class));
            finish();
        }
        public void callforgetPassPage(View view){
            startActivity(new Intent(getApplicationContext(), ForgetPassword.class));
            finish();
        }


        @Override
        public void onBackPressed() {
            startActivity(new Intent(getApplicationContext(), LoginSignUp.class));
            finish();
        }

```

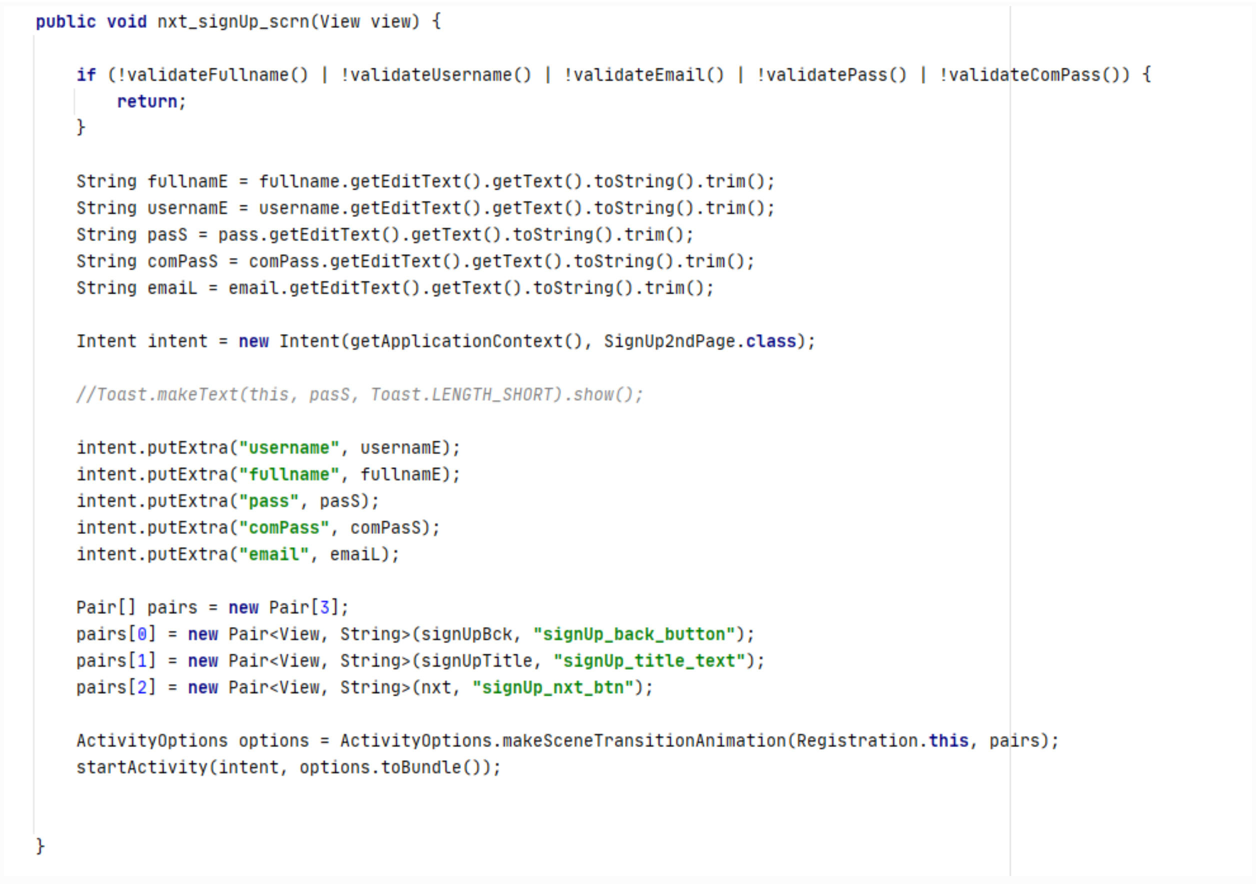


```java
    private boolean validateFields(){
        String _email = user.getEditText().getText().toString().trim();
        String _pass = pass.getEditText().getText().toString().trim();
        if (_email.isEmpty()){
            user.setError("Field can not be empty");
            user.requestFocus();
            return false;
        }else if(_pass.isEmpty()){
            pass.setError("Field can not be empty");
            pass.requestFocus();
            return false;
        }else{
            user.setError(null);
            user.setErrorEnabled(false);
            pass.setError(null);
            pass.setErrorEnabled(false);

            return true;

        }
    }

}
```

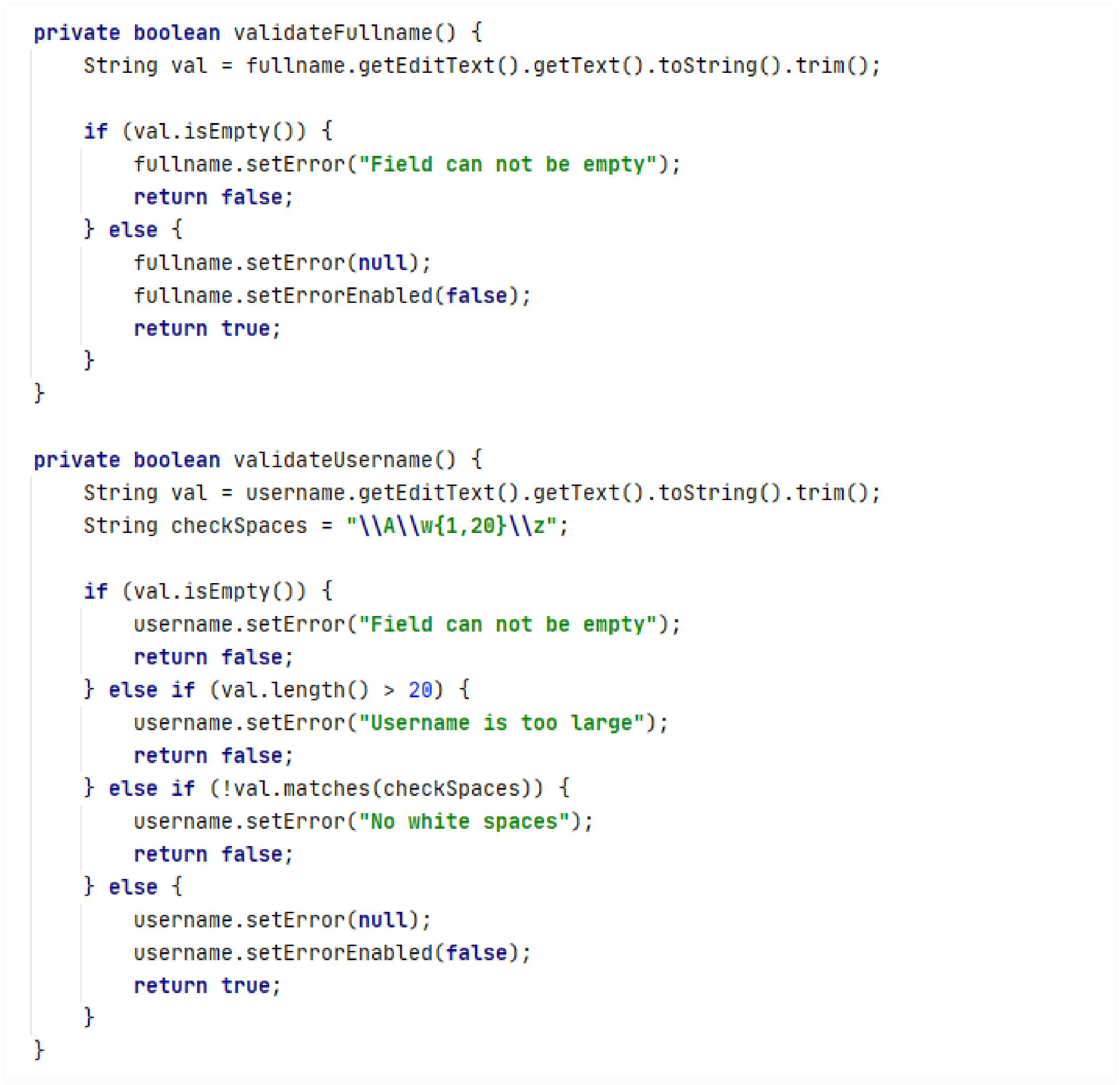
Registration



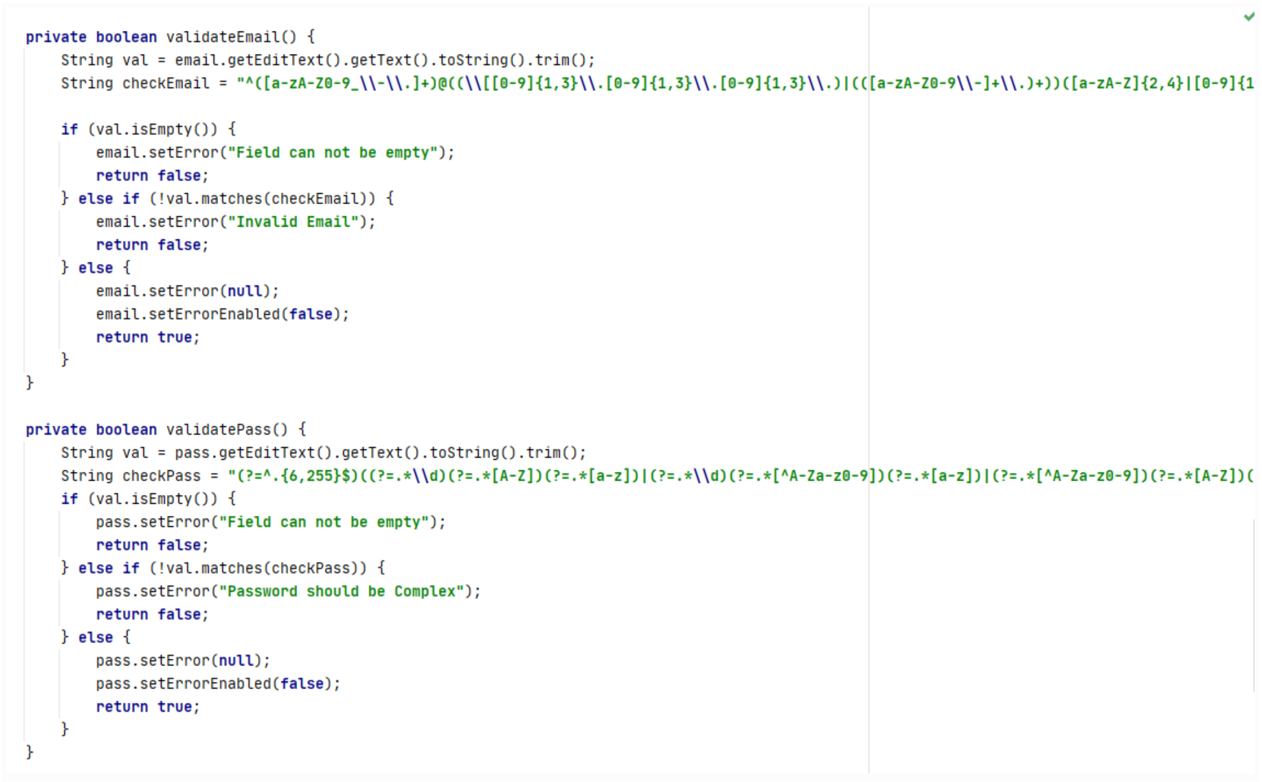
```
package aidooo.spydo.com.project1.Common.LoginSignUp;

import ...

public class Registration extends AppCompatActivity {

    ImageView signUpBck;
    Button nxt;
    TextView signUpTitle;
    TextInputLayout fullname, username, pass, comPass, email;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_registration);

        signUpBck = findViewById(R.id.signUp_back_button);
        nxt = findViewById(R.id.signUp_nxt_btn);
        signUpTitle = findViewById(R.id.signUp_title_text);

        fullname = findViewById(R.id.regi_fullname);
        username = findViewById(R.id.regi_username);
        pass = findViewById(R.id.regi_pass);
        comPass = findViewById(R.id.regiCom_pass);
        email = findViewById(R.id.regi_email);

    }

    public void back(View view) {
        startActivity(new Intent(getApplicationContext(), LoginSignUp.class));
        finish();
    }
```



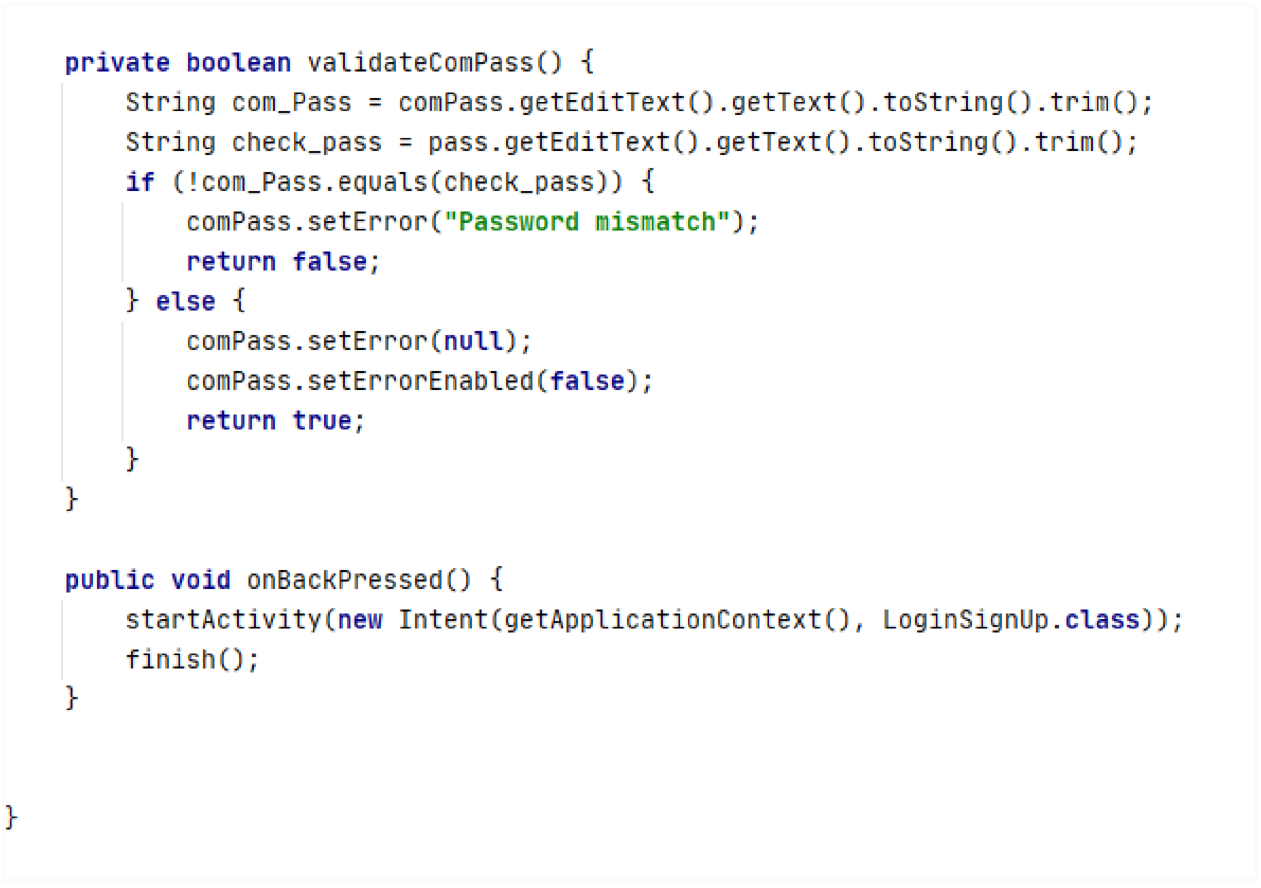
```java
public void nxt_signUp_scrn(View view) {

    if (!validateFullname() | !validateUsername() | !validateEmail() | !validatePass() | !validateComPass()) {
        return;
    }

    String fullnamE = fullname.getEditText().getText().toString().trim();
    String usernamE = username.getEditText().getText().toString().trim();
    String pasS = pass.getEditText().getText().toString().trim();
    String comPasS = comPass.getEditText().getText().toString().trim();
    String emaiL = email.getEditText().getText().toString().trim();

    Intent intent = new Intent(getApplicationContext(), SignUp2ndPage.class);

    //Toast.makeText(this, pasS, Toast.LENGTH_SHORT).show();

    intent.putExtra("username", usernamE);
    intent.putExtra("fullname", fullnamE);
    intent.putExtra("pass", pasS);
    intent.putExtra("comPass", comPasS);
    intent.putExtra("email", emaiL);

    Pair[] pairs = new Pair[3];
    pairs[0] = new Pair<View, String>(signUpBck, "signUp_back_button");
    pairs[1] = new Pair<View, String>(signUpTitle, "signUp_title_text");
    pairs[2] = new Pair<View, String>(nxt, "signUp_nxt_btn");

    ActivityOptions options = ActivityOptions.makeSceneTransitionAnimation(Registration.this, pairs);
    startActivity(intent, options.toBundle());


}
```



```java
private boolean validateFullname() {
    String val = fullname.getEditText().getText().toString().trim();

    if (val.isEmpty()) {
        fullname.setError("Field can not be empty");
        return false;
    } else {
        fullname.setError(null);
        fullname.setErrorEnabled(false);
        return true;
    }
}

private boolean validateUsername() {
    String val = username.getEditText().getText().toString().trim();
    String checkSpaces = "\\A\\w{1,20}\\z";

    if (val.isEmpty()) {
        username.setError("Field can not be empty");
        return false;
    } else if (val.length() > 20) {
        username.setError("Username is too large");
        return false;
    } else if (!val.matches(checkSpaces)) {
        username.setError("No white spaces");
        return false;
    } else {
        username.setError(null);
        username.setErrorEnabled(false);
        return true;
    }
}
```

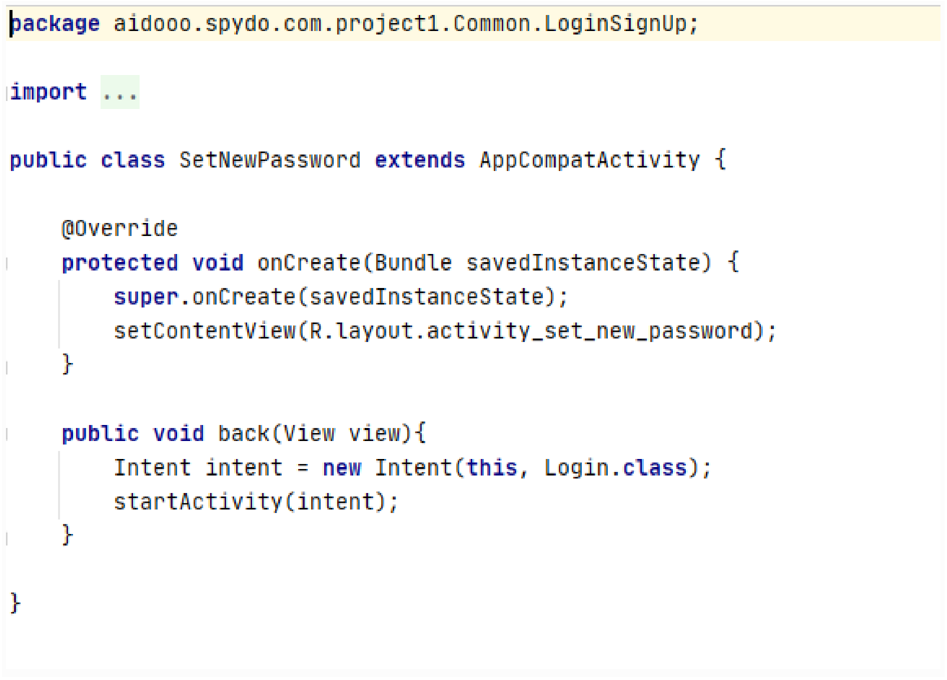


```
private boolean validateEmail() {
    String val = email.getEditText().getText().toString().trim();
    String checkEmail = "^([a-zA-Z0-9_\\-\\.]+)@((\\[[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.)|(([a-zA-Z0-9\\-]+\\.)+))([a-zA-Z]{2,4}|[0-9]{1

    if (val.isEmpty()) {
        email.setError("Field can not be empty");
        return false;
    } else if (!val.matches(checkEmail)) {
        email.setError("Invalid Email");
        return false;
    } else {
        email.setError(null);
        email.setErrorEnabled(false);
        return true;
    }
}

private boolean validatePass() {
    String val = pass.getEditText().getText().toString().trim();
    String checkPass = "(?=^.{6,255}$)((?=.*\\d)(?=.*[A-Z])(?=.*[a-z])|(?=.*\\d)(?=.*[^A-Za-z0-9])(?=.*[a-z])|(?=.*[^A-Za-z0-9])(?=.*[A-Z])(
    if (val.isEmpty()) {
        pass.setError("Field can not be empty");
        return false;
    } else if (!val.matches(checkPass)) {
        pass.setError("Password should be Complex");
        return false;
    } else {
        pass.setError(null);
        pass.setErrorEnabled(false);
        return true;
    }
}
```



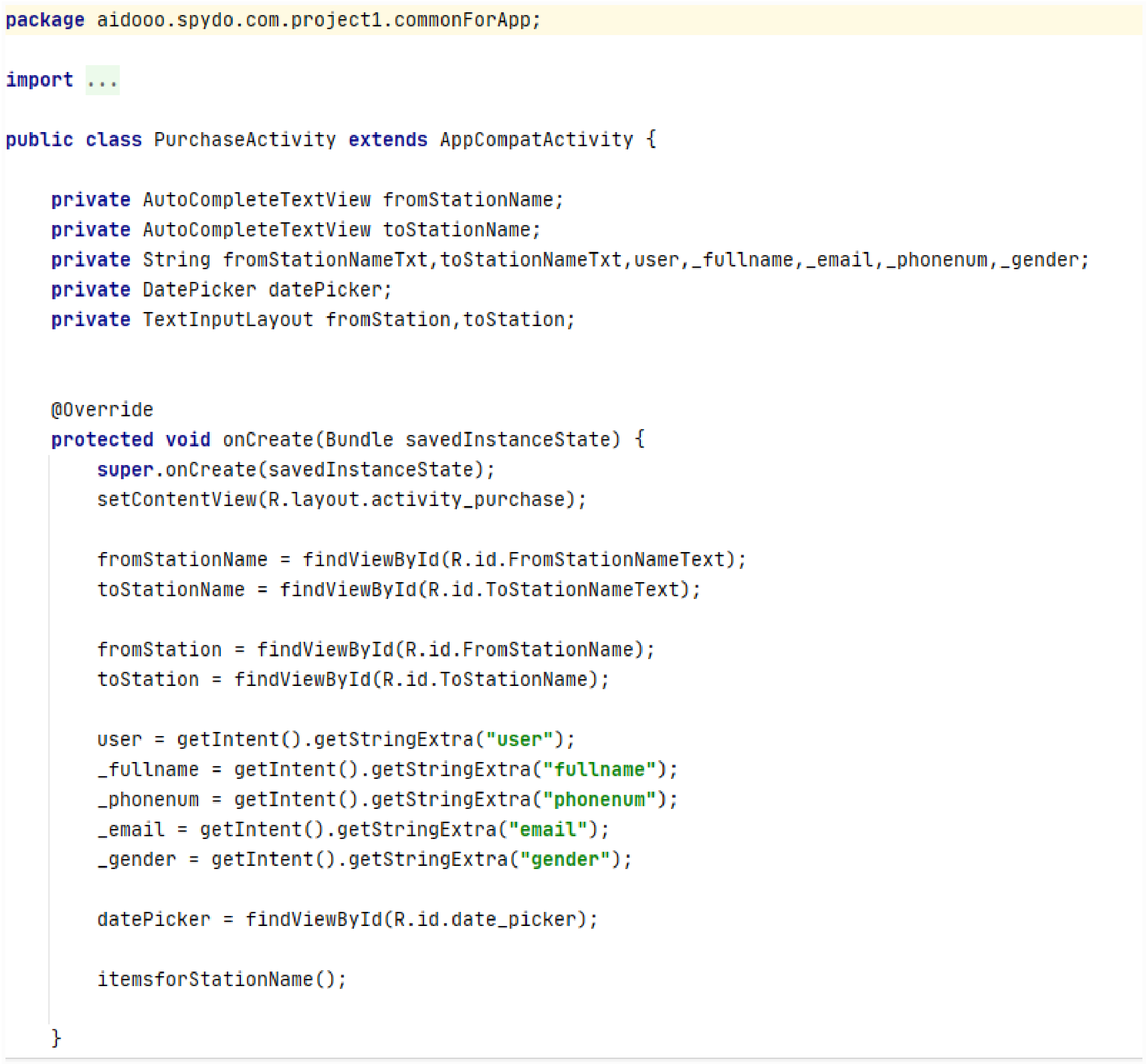
```java
    private boolean validateComPass() {
        String com_Pass = comPass.getEditText().getText().toString().trim();
        String check_pass = pass.getEditText().getText().toString().trim();
        if (!com_Pass.equals(check_pass)) {
            comPass.setError("Password mismatch");
            return false;
        } else {
            comPass.setError(null);
            comPass.setErrorEnabled(false);
            return true;
        }
    }

    public void onBackPressed() {
        startActivity(new Intent(getApplicationContext(), LoginSignUp.class));
        finish();
    }


}
```

Set Password

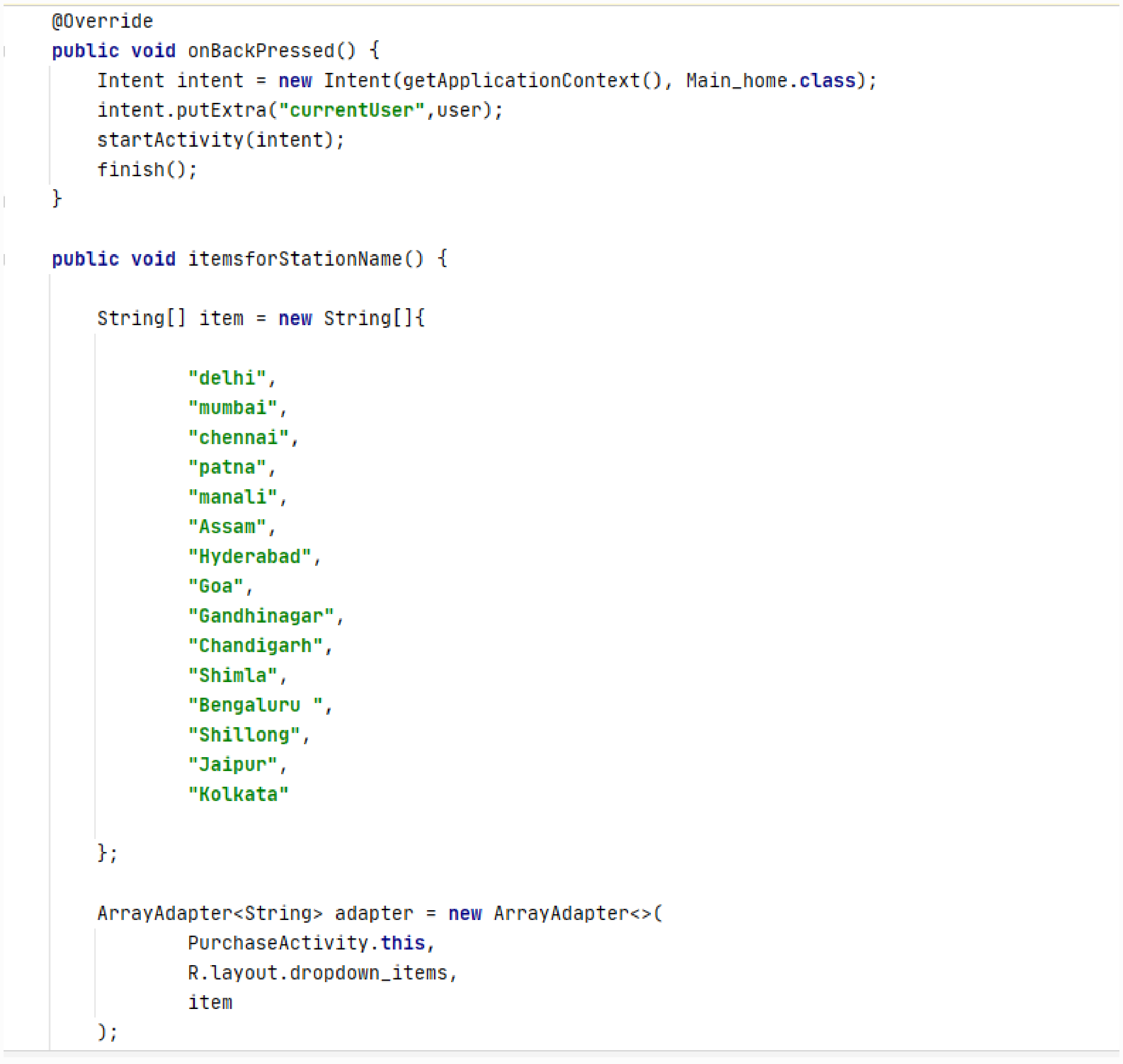


```java
package aidooo.spydo.com.project1.Common.LoginSignUp;

import ...

public class SetNewPassword extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_set_new_password);
    }

    public void back(View view){
        Intent intent = new Intent(this, Login.class);
        startActivity(intent);
    }

}
```

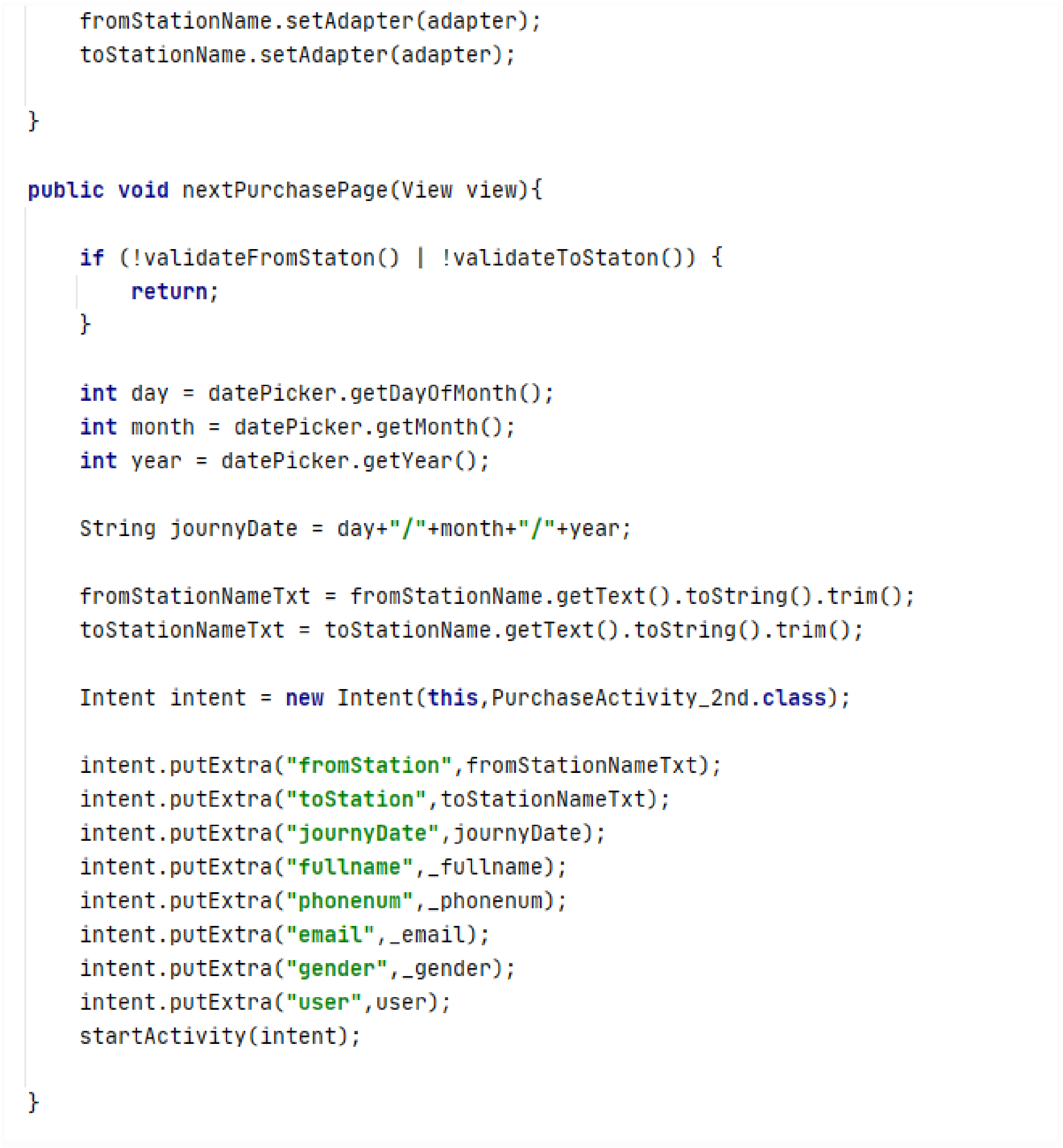
Booking



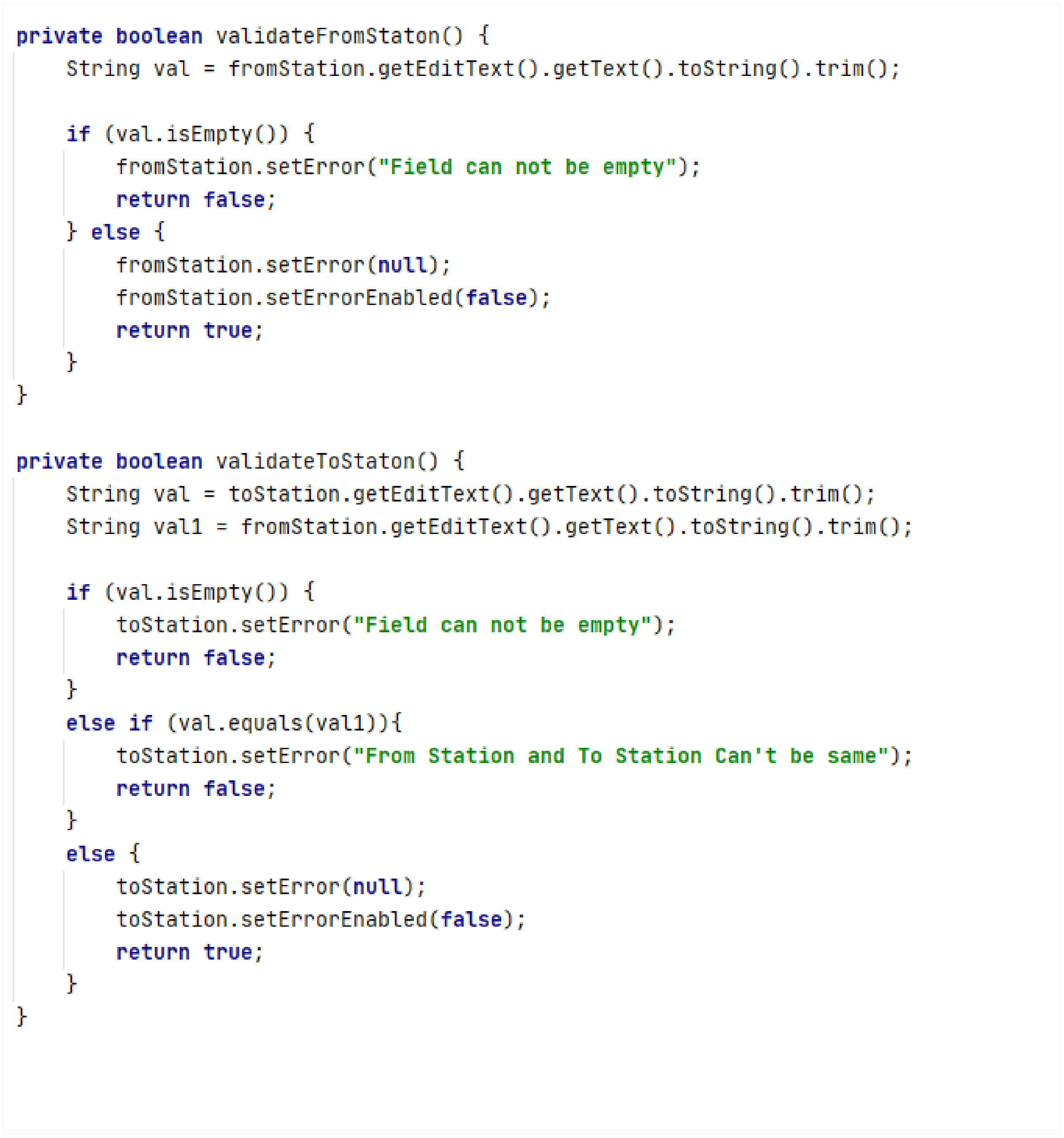
```java
package aidooo.spydo.com.project1.commonForApp;

import ...

public class PurchaseActivity extends AppCompatActivity {

    private AutoCompleteTextView fromStationName;
    private AutoCompleteTextView toStationName;
    private String fromStationNameTxt,toStationNameTxt,user,_fullname,_email,_phonenum,_gender;
    private DatePicker datePicker;
    private TextInputLayout fromStation,toStation;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_purchase);

        fromStationName = findViewById(R.id.FromStationNameText);
        toStationName = findViewById(R.id.ToStationNameText);

        fromStation = findViewById(R.id.FromStationName);
        toStation = findViewById(R.id.ToStationName);

        user = getIntent().getStringExtra("user");
        _fullname = getIntent().getStringExtra("fullname");
        _phonenum = getIntent().getStringExtra("phonenum");
        _email = getIntent().getStringExtra("email");
        _gender = getIntent().getStringExtra("gender");

        datePicker = findViewById(R.id.date_picker);

        itemsforStationName();

    }
```



```java
@Override
public void onBackPressed() {
    Intent intent = new Intent(getApplicationContext(), Main_home.class);
    intent.putExtra("currentUser",user);
    startActivity(intent);
    finish();
}

public void itemsforStationName() {

    String[] item = new String[]{

            "delhi",
            "mumbai",
            "chennai",
            "patna",
            "manali",
            "Assam",
            "Hyderabad",
            "Goa",
            "Gandhinagar",
            "Chandigarh",
            "Shimla",
            "Bengaluru ",
            "Shillong",
            "Jaipur",
            "Kolkata"

    };

    ArrayAdapter<String> adapter = new ArrayAdapter<>(
            PurchaseActivity.this,
            R.layout.dropdown_items,
            item
    );
```



```java
    fromStationName.setAdapter(adapter);
    toStationName.setAdapter(adapter);

}

public void nextPurchasePage(View view){

    if (!validateFromStaton() | !validateToStaton()) {
        return;
    }

    int day = datePicker.getDayOfMonth();
    int month = datePicker.getMonth();
    int year = datePicker.getYear();

    String journyDate = day+"/"+month+"/"+year;

    fromStationNameTxt = fromStationName.getText().toString().trim();
    toStationNameTxt = toStationName.getText().toString().trim();

    Intent intent = new Intent(this,PurchaseActivity_2nd.class);

    intent.putExtra("fromStation",fromStationNameTxt);
    intent.putExtra("toStation",toStationNameTxt);
    intent.putExtra("journyDate",journyDate);
    intent.putExtra("fullname",_fullname);
    intent.putExtra("phonenum",_phonenum);
    intent.putExtra("email",_email);
    intent.putExtra("gender",_gender);
    intent.putExtra("user",user);
    startActivity(intent);

}
```



```java
private boolean validateFromStaton() {
    String val = fromStation.getEditText().getText().toString().trim();

    if (val.isEmpty()) {
        fromStation.setError("Field can not be empty");
        return false;
    } else {
        fromStation.setError(null);
        fromStation.setErrorEnabled(false);
        return true;
    }
}

private boolean validateToStaton() {
    String val = toStation.getEditText().getText().toString().trim();
    String val1 = fromStation.getEditText().getText().toString().trim();

    if (val.isEmpty()) {
        toStation.setError("Field can not be empty");
        return false;
    }
    else if (val.equals(val1)){
        toStation.setError("From Station and To Station Can't be same");
        return false;
    }
    else {
        toStation.setError(null);
        toStation.setErrorEnabled(false);
        return true;
    }
}
```