A Personalized Travel Planning And Tracking App

1.1 OVERVIEW

User Account Creation: The first step in using the app would be creating a user account. This would require the user to enter their personal information, including their name, email address, and password.

Personalization: Once the user has created an account, the app would ask them to enter their travel preferences and interests. This information would be used to create a personalized travel plan for the user.

Trip Planning: After the user has entered their travel preferences, the app would suggest destinations and itineraries based on their interests. The user could then select the places they want to visit and the activities they want to do.

Travel Booking: Once the user has selected their travel plan, the app would help them book flights, hotels, rental cars, and other travel-related services.

Itinerary Creation: After the user has booked their travel arrangements, the app would create a detailed itinerary for their trip. This would include information about their flights, hotels, and activities, as well as recommendations for restaurants, attractions, and other points of interest.

Real-Time Tracking: While the user is on their trip, the app would provide real-time information about their travel plans, including flight delays, hotel changes, and other updates.

Sharing: The user could share their travel itinerary with family and friends via social media or email, allowing them to stay connected and up-to-date on their travels.

1.2 Purpose

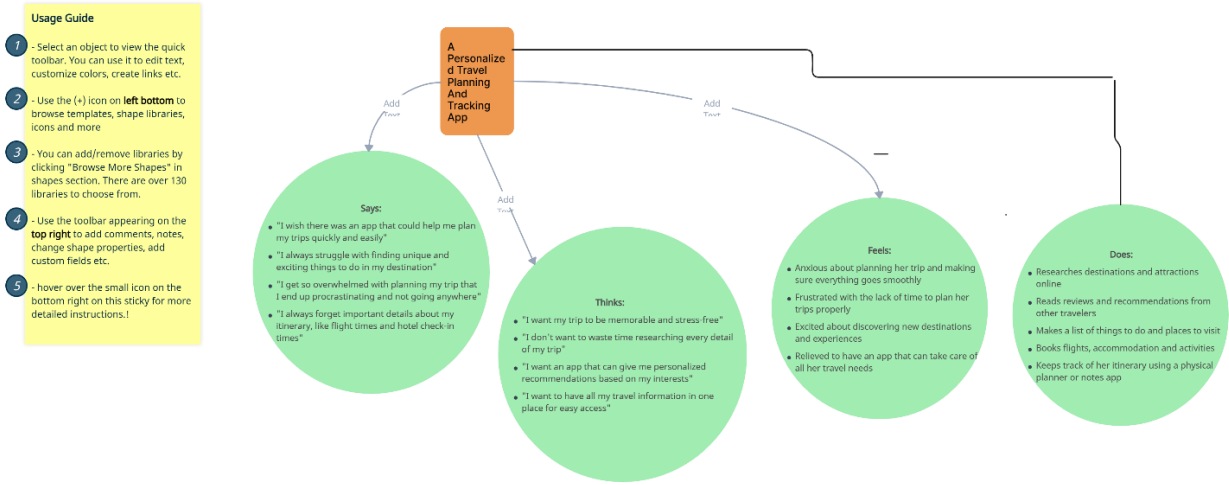
The purpose of a personalized travel planning and tracking app project is to provide a comprehensive platform for users to plan and manage their travels, as well as track their progress and experiences along the way. The app should allow users to easily search and book flights, accommodations, and activities, create customized itineraries, and receive personalized recommendations based on their preferences and travel history.

Additionally, the app should include a feature for users to track their expenses, such as transportation costs, accommodation fees, and food expenses, as well as allow them to upload photos and notes to document their experiences. This will help users stay organized and on budget during their travels, as well as create a digital memory book of their adventures.

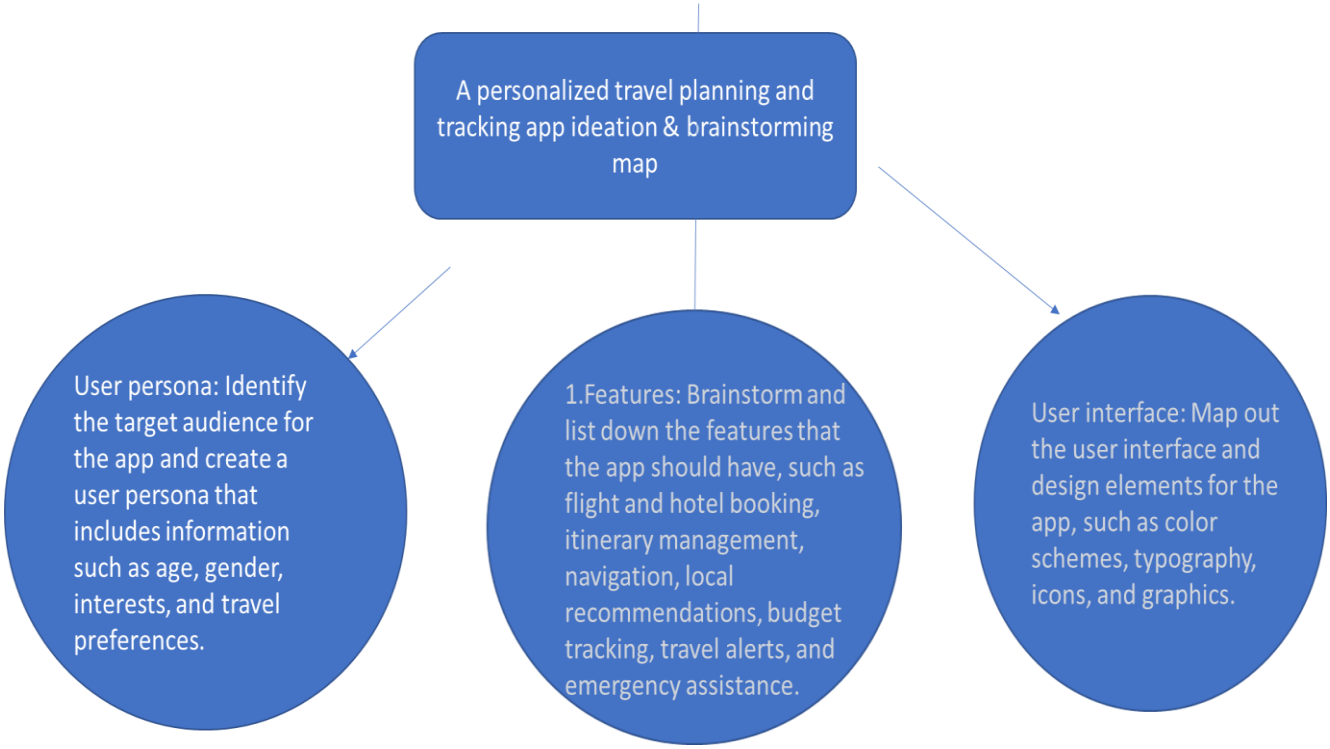
Overall, the goal of the app is to enhance the travel experience for users by providing a convenient, personalized, and comprehensive platform for planning, tracking, and sharing their travels.

2.Problem Definitions & Design Thinking

2.1 Empathy Map



2.2 Ideation & Brainstorming Map



4 ADVANTAGES & DISADVANTAGES

Advantages of a personalized travel planning and tracking app:

1. Customization: A personalized travel planning and tracking app allows users to customize their travel plans based on their preferences, interests, and budget. Users can easily choose the destinations they want to visit, activities they want to do, and places they want to stay at.

2. Time-saving: By using a personalized travel planning and tracking app, users can save a lot of time that would otherwise be spent on researching and planning their trip. The app provides all the necessary information about the destination, including flights, hotels, attractions, and local transportation options, in one place.
3. Cost-effective: A personalized travel planning and tracking app can help users save money on their trip by offering them the best deals and discounts on flights, hotels, and activities.
4. Real-time updates: With a personalized travel planning and tracking app, users can get real-time updates on their trip, including flight delays, cancellations, and gate changes. This can help them plan their travel accordingly and avoid any inconvenience.
5. User-friendly: A personalized travel planning and tracking app is usually very easy to use and user-friendly. Users can quickly navigate through the app and find the information they need.

Disadvantages of a personalized travel planning and tracking app:

1. Dependence on technology: A personalized travel planning and tracking app requires a stable internet connection and a smartphone or tablet to function. If the internet connection is weak or the device malfunctions, the user may not be able to access the information they need.
2. Lack of personal touch: While a personalized travel planning and tracking app provides customized recommendations, it may lack the personal touch and human interaction that a travel agent can offer.
3. Limited information: A personalized travel planning and tracking app may not provide all the information that a user needs to plan their trip. For example, the app may not have information about local customs, culture, or history.
4. Privacy concerns: Personalized travel planning and tracking apps may collect personal information such as location, browsing history, and travel plans. Users need to ensure that their data is secure and not shared with third parties without their consent.
5. Incomplete or inaccurate information: There is always a risk of incomplete or inaccurate information being provided by the app. Users need to verify the information they receive before making any travel arrangements:

5.Applications;

Trip planning: Users can use the app to plan their trip by selecting the destination, travel dates, and activities they want to do.

Flight and hotel booking: The app can help users find and book the best flights and hotels based on their preferences and budget.

Itinerary management: The app can create and manage a detailed itinerary for the trip, including information about flights, hotels, activities, and local transportation.

Navigation: The app can provide users with maps and directions to help them navigate the destination and find their way around.

Local recommendations: The app can offer personalized recommendations for local restaurants, bars, and attractions based on the user's interests and preferences.

Travel alerts: The app can provide real-time alerts and updates on flight delays, gate changes, and other travel-related information.

Budget tracking: The app can help users keep track of their expenses and stick to their travel budget.

Travel journal: The app can allow users to document their trip by adding photos, notes, and memories.

Language translation: The app can offer language translation services to help users communicate with locals.

Emergency assistance: The app can provide emergency assistance and support, such as contacting the embassy or local authorities in case of any mishaps or emergencies.

Conclusion :

In conclusion, a personalized travel planning and tracking app is a useful tool for travelers who want to plan and manage their trips efficiently. The app can help users customize their travel plans based on their preferences and budget, find the best deals on flights and hotels, create and manage a detailed itinerary, get real-time updates on their trip, and navigate their destination easily. While there are some drawbacks to using a personalized travel planning and tracking app, such as dependence on technology and lack of personal touch, the benefits outweigh the disadvantages. Overall, a personalized travel planning and tracking app can enhance the travel experience and make it more enjoyable and stress-free for users.

Future scope :

1. Artificial intelligence and machine learning: These technologies can be used to analyze user preferences and behavior to provide more accurate and personalized travel recommendations.
2. Augmented and virtual reality: These technologies can be used to provide users with immersive travel experiences, allowing them to virtually explore destinations and attractions before they visit.
3. Blockchain technology: This technology can be used to create a secure and transparent travel ecosystem, allowing users to make and receive payments, and book travel arrangements with complete transparency and security.

4. Integration with smart home devices: These apps could integrate with smart home devices, allowing users to control their home environment while they are away, such as adjusting the thermostat or turning on lights.
5. Sustainable and responsible travel: These apps could promote sustainable and responsible travel practices, such as eco-friendly accommodation, reducing carbon footprint, and supporting local communities.

Appendix :

A.Source code

```
package com.example.travelapp

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
```

```
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import
androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat

class RegisterActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            RegistrationScreen(this, databaseHelper)
        }
    }
}

@Composable
fun RegistrationScreen(context: Context, databaseHelper:
UserDatabaseHelper) {
```

```
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var email by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxSize().background(Color.White),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        Image(painterResource(id = R.drawable.tra), contentDescription
= "")

        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
            text = "Register"
        )

        Spacer(modifier = Modifier.height(10.dp))
        TextField(
```

```
    value = username,
    onValueChange = { username = it },
    label = { Text("Username") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)

)

TextField(
    value = email,
    onValueChange = { email = it },
    label = { Text("Email") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)

TextField(
    value = password,
    onValueChange = { password = it },
    label = { Text("Password") },
    visualTransformation = PasswordVisualTransformation(),
    modifier = Modifier
```

```
            .padding(10.dp)
            .width(280.dp)
        )


        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }

        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty() &&
email.isNotEmpty()) {
                    val user = User(
                        id = null,
                        firstName = username,
                        lastName = null,
                        email = email,
                        password = password
                    )
```

```
            databaseHelper.insertUser(user)
            error = "User registered successfully"
            // Start LoginActivity using the current context
            context.startActivity(
                Intent(
                    context,
                    LoginActivity::class.java
                )
            )

        } else {
            error = "Please fill all fields"
        }
    },
    modifier = Modifier.padding(top = 16.dp)
) {
    Text(text = "Register")
}
Spacer(modifier = Modifier.width(10.dp))
Spacer(modifier = Modifier.height(10.dp))

Row() {
    Text(
```

```
            modifier = Modifier.padding(top = 14.dp), text = "Have an
account?"
        )
        TextButton(onClick = {
            context.startActivity(
                Intent(
                    context,
                    LoginActivity::class.java
                )
            )
        })

        {
            Spacer(modifier = Modifier.width(10.dp))
            Text(text = "Log in")
        }
    }
  }
}
private fun startLoginActivity(context: Context) {
  val intent = Intent(context, LoginActivity::class.java)
  ContextCompat.startActivity(context, intent, null)
}
```

User Interface

```
package com.example.travelapp

import androidx.room.*

@Dao
interface UserDao {

    @Query("SELECT * FROM user_table WHERE email = :email")
    suspend fun getUserByEmail(email: String): User?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)

    @Update
    suspend fun updateUser(user: User)

    @Delete
    suspend fun deleteUser(user: User)
}
```

User Database:

```
package com.example.travelapp

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {

    abstract fun userDao(): UserDao

    companion object {

        @Volatile
        private var instance: UserDatabase? = null

        fun getDatabase(context: Context): UserDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    UserDatabase::class.java,
                    "user_database"
                ).build()
```

```
                instance = newInstance
                newInstance
            }
        }
    }
}
```

UserDatabaseHelper:

```
package com.example.travelapp

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null,
DATABASE_VERSION) {

    companion object {
        private const val DATABASE_VERSION = 1
```

```
    private const val DATABASE_NAME = "UserDatabase.db"

    private const val TABLE_NAME = "user_table"
    private const val COLUMN_ID = "id"
    private const val COLUMN_FIRST_NAME = "first_name"
    private const val COLUMN_LAST_NAME = "last_name"
    private const val COLUMN_EMAIL = "email"
    private const val COLUMN_PASSWORD = "password"
  }

  override fun onCreate(db: SQLiteDatabase?) {
    val createTable = "CREATE TABLE $TABLE_NAME (" +
        "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, "
+
        "$COLUMN_FIRST_NAME TEXT, " +
        "$COLUMN_LAST_NAME TEXT, " +
        "$COLUMN_EMAIL TEXT, " +
        "$COLUMN_PASSWORD TEXT" +
        ")"

    db?.execSQL(createTable)
  }
```

```
    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int,
newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }

    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME WHERE $COLUMN_FIRST_NAME = ?",
arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
```

```
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getUserById(id: Int): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME WHERE $COLUMN_ID = ?", arrayOf(id.toString()))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
```

```
            firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
            lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
            email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
            password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
        )
      }
      cursor.close()
      db.close()
      return user
   }

   @SuppressLint("Range")
   fun getAllUsers(): List<User> {
      val users = mutableListOf<User>()
      val db = readableDatabase
      val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME", null)
      if (cursor.moveToFirst()) {
         do {
            val user = User(
               id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
```

```
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
            users.add(user)
        } while (cursor.moveToNext())
    }
    cursor.close()
    db.close()
    return users
  }

}
```

LoginActivity:

```
package com.example.travelapp

import android.content.Context
import android.content.Intent
```

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import
androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            LoginScreen(this, databaseHelper)
        }
    }
}
@Composable
fun LoginScreen(context: Context, databaseHelper:
UserDatabaseHelper) {

    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxSize().background(Color.White),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        Image(painterResource(id = R.drawable.trav),
contentDescription = "")
```

```
Text(
    fontSize = 36.sp,
    fontWeight = FontWeight.ExtraBold,
    fontFamily = FontFamily.Cursive,
    text = "Login"
)
Spacer(modifier = Modifier.height(10.dp))

TextField(
    value = username,
    onValueChange = { username = it },
    label = { Text("Username") },
    modifier = Modifier.padding(10.dp)
        .width(280.dp)
)

TextField(
    value = password,
    onValueChange = { password = it },
    label = { Text("Password") },
    visualTransformation = PasswordVisualTransformation(),
    modifier = Modifier.padding(10.dp)
        .width(280.dp)
```

```
        )

        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }

        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty()) {
                    val user =
databaseHelper.getUserByUsername(username)
                    if (user != null && user.password == password) {
                        error = "Successfully log in"
                        context.startActivity(
                            Intent(
                                context,
                                MainActivity::class.java
                            )
                        )
                        //onLoginSuccess()
```

```
            }
            else {
                error =  "Invalid username or password"
            }

        } else {
            error = "Please fill all fields"
        }
    },
    modifier = Modifier.padding(top = 16.dp)
) {
    Text(text = "Login")
}
Row {
    TextButton(onClick = {context.startActivity(
        Intent(
            context,
            RegisterActivity::class.java
        )
    )}
    )
    { Text(text = "Register") }
    TextButton(onClick = {
    })
```

```
        {
            Spacer(modifier = Modifier.width(60.dp))
            Text(text = "Forget password?")
        }
    }
  }
}
private fun startMainPage(context: Context) {
  val intent = Intent(context, MainActivity::class.java)
  ContextCompat.startActivity(context, intent, null)
}
```

RegisterActivity:

```
package com.example.travelapp

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
```

```
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import
androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat

class RegisterActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            RegistrationScreen(this, databaseHelper)
```

```
        }
    }
}

@Composable
fun RegistrationScreen(context: Context, databaseHelper:
UserDatabaseHelper) {

    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var email by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxSize().background(Color.White),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        Image(painterResource(id = R.drawable.tra), contentDescription
= "")

        Text(
            fontSize = 36.sp,
```

```
    fontWeight = FontWeight.ExtraBold,
    fontFamily = FontFamily.Cursive,
    text = "Register"
)

Spacer(modifier = Modifier.height(10.dp))
TextField(
    value = username,
    onValueChange = { username = it },
    label = { Text("Username") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)

)

TextField(
    value = email,
    onValueChange = { email = it },
    label = { Text("Email") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)
```

```
    TextField(
        value = password,
        onValueChange = { password = it },
        label = { Text("Password") },
        visualTransformation = PasswordVisualTransformation(),
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)
    )



    if (error.isNotEmpty()) {
        Text(
            text = error,
            color = MaterialTheme.colors.error,
            modifier = Modifier.padding(vertical = 16.dp)
        )
    }


    Button(
        onClick = {
            if (username.isNotEmpty() && password.isNotEmpty() &&
email.isNotEmpty()) {
```

```
            val user = User(
                id = null,
                firstName = username,
                lastName = null,
                email = email,
                password = password
            )
            databaseHelper.insertUser(user)
            error = "User registered successfully"
            // Start LoginActivity using the current context
            context.startActivity(
                Intent(
                    context,
                    LoginActivity::class.java
                )
            )

        } else {
            error = "Please fill all fields"
        }
    },
    modifier = Modifier.padding(top = 16.dp)
) {
    Text(text = "Register")
```

```
        }
        Spacer(modifier = Modifier.width(10.dp))
        Spacer(modifier = Modifier.height(10.dp))

        Row() {
            Text(
                modifier = Modifier.padding(top = 14.dp), text = "Have an
account?"
            )
            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,
                        LoginActivity::class.java
                    )
                )
            })

            {
                Spacer(modifier = Modifier.width(10.dp))
                Text(text = "Log in")
            }
        }
    }
```

```
}
private fun startLoginActivity(context: Context) {
    val intent = Intent(context, LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

MainAcitvity:

```
package com.example.travelapp

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.Card
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
```

```
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.scale
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            TravelApp(this)
        }
    }


    @Composable
    fun TravelApp(context: Context) {
        Column(
            modifier = Modifier
```

```
        .padding(20.dp)
        .verticalScroll(rememberScrollState())

) {

    Text(
        fontSize = 40.sp,
        color = Color(android.graphics.Color.rgb(120, 40, 251)),
        fontFamily = FontFamily.Cursive,
        text = "Wanderlust Travel"
    )

    Spacer(modifier = Modifier.height(20.dp))

    // 01
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .height(250.dp)
            .clickable {
            context.startActivity(
                Intent(context, BaliActivity::class.java)

            )
```

```
            },
            elevation = 8.dp
        )
        {
            Column(
                horizontalAlignment = Alignment.CenterHorizontally
            ) {
                Image(
                    painterResource(id = R.drawable.bali),
contentDescription = "",
                    modifier = Modifier
                        .height(150.dp)
                        .scale(scaleX = 1.2F, scaleY = 1F)
                )

                Text(
                    text = stringResource(id = R.string.place_1),
                    fontSize = 18.sp
                )



                Text(
                    text = stringResource(id = R.string.description),
                    fontWeight = FontWeight.Light,
```

```
                fontSize = 16.sp,
                textAlign = TextAlign.Center,
            )

            Text(
                text = stringResource(id = R.string.plan), color =
Color.Gray,
                fontSize = 16.sp
            )
        }
    }

    Spacer(modifier = Modifier.height(20.dp))



    //02
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .height(250.dp)
            .clickable {
            context.startActivity(
                Intent(context, ParisActivity::class.java)
```

```
            )
            },
            elevation = 8.dp
        )
        {
            Column(
                horizontalAlignment = Alignment.CenterHorizontally
            ) {
                Image(
                    painterResource(id = R.drawable.paris),
contentDescription = "",
                    modifier = Modifier
                        .height(150.dp)
                        .scale(scaleX = 1.2F, scaleY = 1F)
                )

                Text(
                    text = stringResource(id = R.string.place_2),
                    fontSize = 18.sp
                )



                Text(
                    text = stringResource(id = R.string.description),
```

```
                fontWeight = FontWeight.Light,
                fontSize = 16.sp,
                textAlign = TextAlign.Center,
            )

            Text(
                text = stringResource(id = R.string.plan), color =
Color.Gray,
                fontSize = 16.sp
            )
        }
    }

    Spacer(modifier = Modifier.height(20.dp))

    //03
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .height(250.dp)
            .clickable {
            context.startActivity(
                Intent(context, SingaporeActivity::class.java)
```

```
            )
            },
        elevation = 8.dp
    )
    {
        Column(
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Image(
                painterResource(id = R.drawable.singapore),
contentDescription = "",
                modifier = Modifier
                    .height(150.dp)
                    .scale(scaleX = 1.2F, scaleY = 1F)
            )

            Text(
                text = stringResource(id = R.string.place_3),
                fontSize = 18.sp
            )



            Text(
                text = stringResource(id = R.string.description),
```

```
                fontWeight = FontWeight.Light,
                fontSize = 16.sp,
                textAlign = TextAlign.Center,
            )

            Text(
                text = stringResource(id = R.string.plan), color =
Color.Gray,
                fontSize = 16.sp
            )
        }
    }

    Spacer(modifier = Modifier.height(20.dp))
  }
 }
}
```

ParisActivity:

```
package com.example.travelapp

import android.os.Bundle
import androidx.activity.ComponentActivity
```

```
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.scale
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.travelapp.ui.theme.TravelAppTheme

class ParisActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        setContent {
            TravelAppTheme {
                // A surface container using the 'background' color from the
theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    Greeting()
                }
            }
        }
    }
}

@Composable
fun Greeting() {

    Column(
        modifier = Modifier.background(color = Color.White)
            .padding(20.dp)
            .verticalScroll(rememberScrollState())
    ) {
        Text(
```

```
        fontSize = 40.sp,
        color = Color(android.graphics.Color.rgb(120, 40, 251)),
        fontFamily = FontFamily.Cursive,
        text = stringResource(id = R.string.place_2),
    )
    Image(
        painterResource(id = R.drawable.paris), contentDescription =
"",
        modifier = Modifier
            .padding(16.dp)
            .fillMaxWidth()
            .height(200.dp)
            .scale(scaleX = 1.2F, scaleY = 1F)
    )
    Text(
        color=Color.Black,
        text = "Day 1: Arrival and Introduction\n" +

            "Check into your accommodation and freshen up\n" +
            "Take a stroll around the neighborhood to get
acquainted\n" +
            "Visit the Eiffel Tower, preferably in the evening when it is
lit up\n" +
            "Have a relaxing dinner at a nearby restaurant\n" +
```

"\n" +

"Day 2: Art and History\n" +

"Visit the Louvre Museum to see some of the world's most famous art pieces\n" +

"Stroll through the Tuileries Garden and the Place de la Concorde\n" +

"Visit the Orsay Museum, which houses a large collection of impressionist art\n" +

"Have dinner at a local French restaurant\n" +

"\n" +

"Day 3: French Culture and Food\n" +

"Visit the Montmartre neighborhood to see the famous Basilique du Sacré-Cœur and Place du Tertre\n" +

"Explore the historic neighborhood of Le Marais\n" +

"Try some delicious French pastries at a local bakery\n" +

"Have dinner at a brasserie to taste some classic French cuisine\n" +

"\n" +

"Day 4: Architecture and Gardens\n" +

"Visit the Palace of Versailles, a UNESCO World Heritage site, and explore its beautiful gardens\n" +

"Walk along the Champs-Elysées and stop at the Arc de Triomphe\n" +

"Visit the Sainte-Chapelle, a beautiful Gothic chapel with stunning stained-glass windows\n" +

"Have dinner at a local restaurant in the 7th arrondissement\n" +

"\n" +

"Day 5: Shopping and Sightseeing\n" +

"Visit the Notre-Dame Cathedral and climb up to the top for a stunning view of the city\n" +

"Explore the Latin Quarter and visit the Panthéon\n" +

"Go shopping at the famous Galeries Lafayette or Printemps department stores\n" +

"Have dinner at a local bistro\n" +

"\n" +

"Day 6: Parisian Parks and Museums\n" +

"Visit the Musée Rodin and explore its beautiful gardens\n" +

"Stroll through the Luxembourg Gardens and visit the Luxembourg Palace\n" +

```
            "Visit the Centre Pompidou, a modern art museum in the
Marais neighborhood\n" +
            "Have dinner at a local restaurant in the Latin Quarter\n"
+
            "\n" +
            "Day 7: River Cruise and Farewell\n" +

            "Take a boat cruise along the Seine River to see the city
from a different perspective\n" +
            "Visit the Musée de l'Orangerie, which houses Monet's
famous water lilies paintings\n" +
            "Have a farewell dinner at a Michelin-starred restaurant"
        )
    }
}
```

AndroidMainFest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
```

```
android:dataExtractionRules="@xml/data_extraction_rules"
android:fullBackupContent="@xml/backup_rules"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:supportsRtl="true"
android:theme="@style/Theme.TravelApp"
tools:targetApi="31">
<activity
   android:name=".RegisterActivity"
   android:exported="false"
   android:label="RegisterActivity"
   android:theme="@style/Theme.TravelApp" />
<activity
   android:name=".SingaporeActivity"
   android:exported="false"
   android:label="@string/title_activity_singapore"
   android:theme="@style/Theme.TravelApp" />
<activity
   android:name=".ParisActivity"
   android:exported="false"
   android:label="@string/title_activity_paris"
   android:theme="@style/Theme.TravelApp" />
<activity
   android:name=".BaliActivity"
```

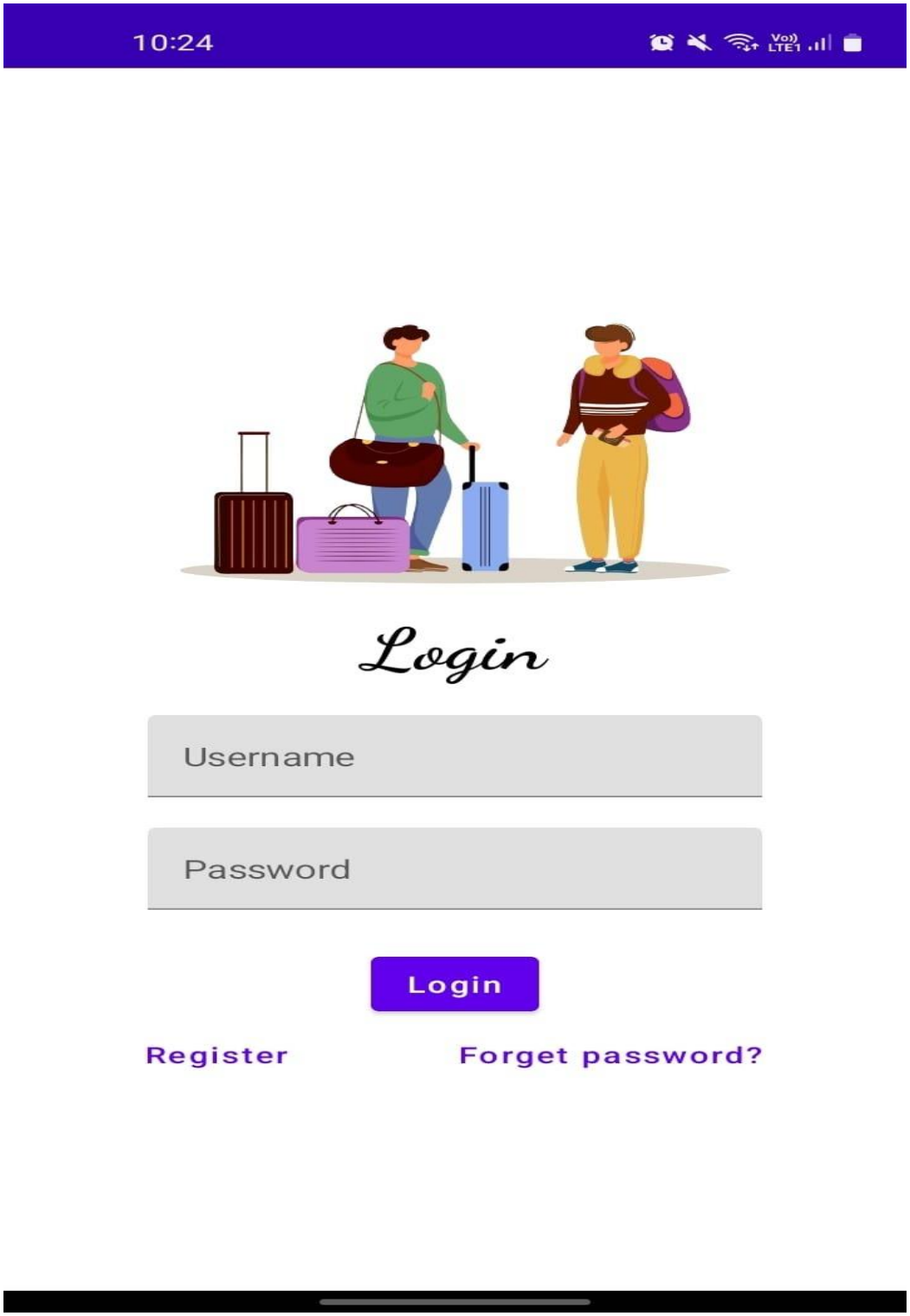
```
        android:exported="false"
        android:label="@string/title_activity_bali"
        android:theme="@style/Theme.TravelApp" />
    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:label="@string/app_name"
        android:theme="@style/Theme.TravelApp"/>
    <activity
        android:name=".LoginActivity"
        android:exported="true"
        android:label="@string/app_name"
        android:theme="@style/Theme.TravelApp">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category
android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
  </application>

</manifest>
```

Pictures :

Login Page:



Register page:

Main Page:

Location Page:

Bali

Day 1: Arrival and Relaxation
Arrive in Bali and check into your hotel or accommodation.
Spend the day relaxing and getting acclimated to the island.
If you have time, explore the nearby area or head to the beach.

Day 2: Ubud Tour
Start your day early and head to Ubud, a cultural and artistic hub in Bali.
Visit the Monkey Forest and the Ubud Palace.
Take a tour of the Tegalalang Rice Terrace, a beautiful UNESCO World Heritage Site.
End your day with a traditional Balinese dance performance.

Day 3: Temple Hopping
Visit some of Bali's most famous temples, such as Tanah Lot and Uluwatu.
Take in the stunning views of the ocean and cliffs.
Enjoy a sunset dinner at one of the many restaurants near the temples.