WANDERLUST: A PERSONALIZED TRAVEL PLANNING AND TRACKING APP

ABSTRACT:

A project built using the Android Compose UI toolkit. It demonstrates how to create a simple travel app using the Compose libraries. It also features a personalized feed of recommended accommodations based on the locations. Since we are communal creatures, traveling holds immense benefits for us humans. The staggering number is a testament to the popularity of traveling as a means of relaxation and improving one's health. From staving off burnouts to promoting overall healthy wellbeing, traveling proves to be a valuable leisure activity. Moreover, on a wider spectrum, tourism increases the economy's revenue, produces thousands of employment, improves a country's infrastructure, and fosters cultural interchange between outsiders and natives. To ensure that traveling isn't a Herculean task and adds to the stress of the whole activity, one can find a plethora of apps that assist with this. The growth of travel apps/ trip planner apps has been on the rise due to ease of use and access to information.

INTRODUCTION

The global pandemic brought the world to a standstill. Various sectors faced the brunt of it, especially the travel and tourism industry. As countries began easing travel restrictions, it became important travel websites and apps caught up with the new trends. People became wary of health regulations, therefore travel planner apps and websites had to work closely with their booking agents and policy advisories to guarantee that hotels and airlines were safe partners. There was a preference for traveling in smaller groups, such as with their families or close friends rather than tour groups. Individual bookings and smaller group tours were more popular due to the safety assurance they provided. Another major shift in traveling habits was that there was a demand for domestic travel over outbound travel due to the irregularity in country-wise travel regulations.

Why do we need a Travel Planning App?

In today's fast-paced world, travelers are always looking for ways to simplify and streamline their trip preparations. A travel planner app is a perfect solution for these needs, offering numerous benefits that make the planning process easier and more enjoyable. In this article, we'll discuss the key reasons why using a trip planner is essential for modern travelers, and how travel app development has revolutionized the way we explore the world. Here are some reasons why we need a Travel Planning App:

- Time-Saving Convenience:

A travel planner app takes the hassle out of planning your next adventure. With just a few taps on your smartphone, you can create a detailed itinerary that includes flights, accommodations, and activities.

- Personalized Experiences:

Trip planner build space allows users to customize their itineraries based on their preferences, ensuring a unique and tailored experience for every traveler. Users can easily add, edit or remove items from their trip planner, making it simple to create a personalized journey.

- Organized Information:

A travel planner app keeps all your trip details in one place, making it easy to access and manage. From flight confirmations to hotel reservations, everything is neatly organized, eliminating the need for endless paper printouts or searching through countless emails.

- Seamless Collaboration:

Traveling with friends or family? A travel planner app allows users to share their plans with others, making it easy for everyone to stay on the same page and contribute to the planning process.

- Budget Management:

Travel app development has made it possible for users to track their expenses and set budget limits. This feature is particularly helpful for those looking to stick to a specific budget while on vacation.

- Build a Space Trip Planner:

With the growing interest in space travel, travel planner apps have started to include options for space trips. Build a space trip planner feature in your app, and be at the forefront of this exciting new frontier in travel.

- Offline Access:

One of the most significant advantages of using a travel planner app is the ability to access your information offline. This ensures that you always have your itinerary at your fingertips, even when you're without an internet connection.

A travel planner app is an essential tool for any traveler in the modern world. From saving time and offering personalized experiences to facilitating seamless collaboration and budget management, travel app development has truly transformed the way we plan our trips. With the inclusion of cutting-edge features like a trip planner build space, these apps continue to innovate and cater to the ever-evolving needs of travelers around the globe. So, if you haven't

already, it’s time to embrace the future of travel planning with a comprehensive and feature-rich travel planner app.

Benefits of Travel Apps

With the enforcement of precautionary measures for COVID, the hospitality industry has had to incorporate new facilities to adapt to the “new normal”. Sanitization protocols, the use of protective gears such as masks and glasses, and screening tools such as thermal scanners and pulse oximeters were mandatory equipment at shops, hotels, restaurants, and airlines. The easiest way one could stay updated about these regulations was through the regular updates on travel apps. Even at a time when travel was restricted, trip-planning apps became the new source of information on the various travel regulations that each country had imposed. Travel planner apps and technology were able to disseminate pertinent information about COVID-19 to its users with ease. Along with its benefits during the pandemic, travel planner apps come with great benefits.

Admin Panel

A well-designed admin panel is crucial for managing and maintaining a successful trip planner app. Key features of the admin panel include:

- **User management:** Easily manage and track user profiles, activities, and preferences.
- **Content management:** Update and manage app content such as destination information, images, and recommendations.
- **Analytics and reporting:** Monitor app performance, user engagement, and other valuable metrics for data-driven decision-making.

User Panel

The user panel is at the heart of any travel planning app, allowing users to access and manage their travel information. Essential user panel features are:

- **Personal profile:** Users can create and manage their profiles, including preferences, travel history, and saved trips.
- **Trip planner build space:** Build trip planner itineraries with ease, customizing every aspect of the journey to suit individual preferences.
- **Calendar setup and sync:** Sync the trip planner app with external calendars, ensuring all travel dates and events are seamlessly integrated.

Agency panel

An agency panel enables travel agencies and other industry professionals to collaborate and contribute to the holiday planner app. Key agency panel features include:

- **Agency profile management:** Create and manage agency profiles, showcasing services and offerings.
- **Trip planning collaboration:** Work with clients to create customized itineraries within the trip planning app.
- **Booking management:** Track and manage client bookings and reservations.

Creating Trip Itineraries

Building comprehensive and customizable itineraries is a core feature of any travel planner app. Crucial itinerary-building features include:

- **Point-of-interest suggestions:** Offer users curated suggestions for attractions, restaurants, and activities based on their preferences.
- **Map integration:** Seamlessly integrate maps to help users visualize and navigate their trip.
- **Build a space trip planner:** Incorporate space travel options into your app, catering to the growing interest in space tourism.

Calendar Setup and Sync

An efficient calendar setup and sync feature is essential for any trip planner buildai.space. This functionality should include:

- **Easy integration:** Connect with popular calendar apps such as Google Calendar and Apple Calendar for seamless synchronization.
- **Event reminders:** Automatically set reminders for upcoming events and activities within the trip planning app.
- **Multi-device sync:** Ensure calendar information is synced across all devices for easy access.

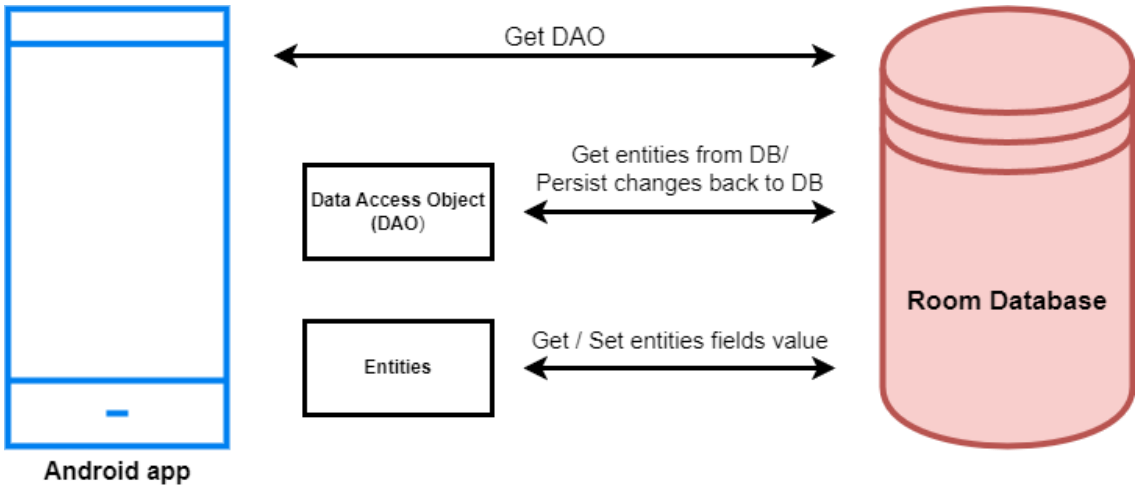
Push Notifications and Alerts

Keep users informed and engaged with timely push notifications and alerts. Key aspects of this feature are:

- **Real-time updates:** Notify users of flight changes, weather conditions, and other important travel updates.
- **Personalized recommendations:** Offer users relevant suggestions based on their preferences and location.
- **App engagement:** Encourage users to explore new features and promotions within the travel planner app.

SYSTEM MODEL

Architecture:



CODE:

```
<?xml version="1.0" encoding="utf-8"?>
    package="com.example.snowsoultrips">
    <uses-permission
  android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyD9jRQy9npx3zuvOK7o-sUz4FA4KOMyKRM" />
      <activity android:name=".LandingActivity" />
      <activity android:name=".MainActivity">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />

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

      <meta-data
        android:name="preloaded_fonts"
        android:resource="@array/preloaded_fonts" />
    </application>

  </manifest>

  package com.example.snowsoultrips;
  import android.app.Activity;
```

```
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.FrameLayout;
import android.widget.ImageView;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.ListResult;
import com.google.firebase.storage.StorageReference;
import com.squareup.picasso.Picasso;

import static android.app.Activity.RESULT_OK;


public class AvatarFragment extends Fragment{
    ImageView image1, image2, image3, image4, image5, image6;
```

```
    private static final String KEY = "key";

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_avatar, container, false);
    }

    private OnFragmentCommunicationListener mListener;

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnFragmentCommunicationListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        image1 = view.findViewById(R.id.imageView1);
```

```
image2 = view.findViewById(R.id.imageView2);
image3 = view.findViewById(R.id.imageView3);
image4 = view.findViewById(R.id.imageView4);
image5 = view.findViewById(R.id.imageView5);
image6 = view.findViewById(R.id.imageView6);

String urlf1 = "https://firebasestorage.googleapis.com/v0/b/snowsoultrips-f09a0.appspot.com/o/images%2Favatar_f_1.png?alt=media&token=62d28c94-a582-466e-a209-3e993522a564";
String urlf2 = "https://firebasestorage.googleapis.com/v0/b/snowsoultrips-f09a0.appspot.com/o/images%2Favatar_f_2.png?alt=media&token=922045c3-1f6f-44ed-ac9f-cfc90c2e7e7a";
String urlf3 = "https://firebasestorage.googleapis.com/v0/b/snowsoultrips-f09a0.appspot.com/o/images%2Favatar_f_3.png?alt=media&token=936525a4-f4b0-4243-8a10-96abbdac0f9d";
String urlf4 = "https://firebasestorage.googleapis.com/v0/b/snowsoultrips-f09a0.appspot.com/o/images%2Favatar_m_1.png?alt=media&token=9be53150-82fc-413a-8dfa-f21a64f035d6";
String urlf5 = "https://firebasestorage.googleapis.com/v0/b/snowsoultrips-f09a0.appspot.com/o/images%2Favatar_m_2.png?alt=media&token=3ec87e28-073c-4ef6-be4c-1c17076128ae";
String urlf6 = "https://firebasestorage.googleapis.com/v0/b/snowsoultrips-f09a0.appspot.com/o/images%2Favatar_m_3.png?alt=media&token=066ce22f-1c5b-4298-bc8b-f901e80737fa";

Picasso.get().load(urlf1).into(image1);
image1.setTag(urlf1);
Picasso.get().load(urlf2).into(image3);
image3.setTag(urlf2);
Picasso.get().load(urlf3).into(image5);
image5.setTag(urlf3);
Picasso.get().load(urlf4).into(image2);
image2.setTag(urlf4);
```

```
        Picasso.get().load(urlf5).into(image4);
        image4.setTag(urlf5);
        Picasso.get().load(urlf6).into(image6);
        image6.setTag(urlf6);

//      Fragment fragment = new Fragment();
//      Bundle bundle = new Bundle();
//      bundle.putString(KEY, urlf1);
//      fragment.setArguments(bundle);



        image1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mListener.OnImageSelected(image1.getTag().toString());
                NavController navController = Navigation.findNavController(view);
                navController.navigate(R.id.mainFragment);
            }
        });
        image3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mListener.OnImageSelected(image3.getTag().toString());
                NavController navController = Navigation.findNavController(view);
                navController.navigate(R.id.mainFragment);
            }
        });
        image5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mListener.OnImageSelected(image5.getTag().toString());
                NavController navController = Navigation.findNavController(view);
```

```
            navController.navigate(R.id.mainFragment);
        }
    });
    image2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mListener.OnImageSelected(image2.getTag().toString());
            NavController navController = Navigation.findNavController(view);
            navController.navigate(R.id.mainFragment);
        }
    });
    image4.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mListener.OnImageSelected(image4.getTag().toString());
            NavController navController = Navigation.findNavController(view);
            navController.navigate(R.id.mainFragment);
        }
    });
    image6.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mListener.OnImageSelected(image6.getTag().toString());
            NavController navController = Navigation.findNavController(view);
            navController.navigate(R.id.mainFragment);
        }
    });
  }
  public interface OnFragmentCommunicationListener {
    void OnImageSelected(String name);
  }
}
```

```
package com.example.snowsoultrips;
import android.graphics.Bitmap;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.NavigationUI;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.libraries.places.api.Places;
import com.google.android.gms.common.api.Status;
import com.google.android.libraries.places.api.model.PhotoMetadata;
import com.google.android.libraries.places.api.model.Place;
import com.google.android.libraries.places.api.net.FetchPhotoRequest;
import com.google.android.libraries.places.api.net.FetchPhotoResponse;
import com.google.android.libraries.places.api.net.FetchPlaceRequest;
import com.google.android.libraries.places.api.net.FetchPlaceResponse;
import com.google.android.libraries.places.api.net.PlacesClient;
import
com.google.android.libraries.places.widget.AutocompleteSupportFragment;
```

```
import
com.google.android.libraries.places.widget.listener.PlaceSelectionListener;
import
com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import java.sql.Array;
import java.sql.Time;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;

import static android.content.ContentValues.TAG;
import static com.example.snowsoultrips.MainActivity.mAuth;


public class AddTripFragment extends Fragment {
TextView placeLat, placeLong, placeName;
ImageView placeCover;
String location,latitude,longitude,coverFromPhotoID;
Button createTrip;
    private static final String TRIPID = "tripid";
    private static final String CREATERID = "createrid";
    private static final String MEMBERID = "memberid";
    private static final String COVERPHOTOID = "coverphotoid";
    private static final String LOCATION = "location";
```

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_add_trip, container,
false);
        placeName = view.findViewById(R.id.tv_addplaceName);
        placeLat = (TextView) view.findViewById(R.id.tv_addplaceLat);
        placeLong = (TextView) view.findViewById(R.id.tv_addplaceLong);
        placeCover = (ImageView) view.findViewById(R.id.addPlaceCover);
        createTrip = (Button) view.findViewById(R.id.btnCreateTrip);
        return view;
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        String apiKey = getResources().getString(R.string.api_key);

        // Initialize Places.
        Places.initialize(getContext(), apiKey);
        // Create a new Places client instance.
        final PlacesClient placesClient = Places.createClient(getContext());

        // Initialize the AutocompleteSupportFragment.
        AutocompleteSupportFragment autocompleteFragment =
(AutocompleteSupportFragment)

getChildFragmentManager().findFragmentById(R.id.autocomplete_fragme
nt);
```

```
        // Specify the types of place data to return.
        autocompleteFragment.setPlaceFields(Arrays.asList(Place.Field.ID,
Place.Field.NAME,Place.Field.LAT_LNG,Place.Field.PHOTO_METADA
TAS));

        // Set up a PlaceSelectionListener to handle the response.
        autocompleteFragment.setOnPlaceSelectedListener(new
PlaceSelectionListener() {
            @Override
            public void onPlaceSelected(final Place place) {
                // TODO: Get info about the selected place.
//                LatLng queriedLocation = place.getLatLng();
                placeName.setText("Place: "+place.getName());
                location = place.getName();
                placeLat.setText("Latitude: "+place.getLatLng().latitude);
                latitude = String.valueOf(place.getLatLng().latitude);
                placeLong.setText("Longitude: "+place.getLatLng().longitude);
                longitude = String.valueOf(place.getLatLng().longitude);

                //-------------------------------------------------------------------------
---------------------------------------------------------------

                // Define a Place ID.
                final String placeId = place.getId();
                coverFromPhotoID = placeId;

                // Specify fields. Requests for photos must always have the
PHOTO_METADATAS field.
                List<Place.Field> fields =
Arrays.asList(Place.Field.PHOTO_METADATAS);
```

```
        // Get a Place object (this example uses fetchPlace(), but you can
also use findCurrentPlace())
        FetchPlaceRequest placeRequest =
FetchPlaceRequest.newInstance(placeId, fields);

placesClient.fetchPlace(placeRequest).addOnSuccessListener(new
OnSuccessListener<FetchPlaceResponse>() {
            @Override
            public void onSuccess(FetchPlaceResponse
fetchPlaceResponse) {
                //Place place = response.getPlace();

                // Get the photo metadata.
                final PhotoMetadata photoMetadata =
place.getPhotoMetadatas().get(0);
                // Get the attribution text.
                String attributions = photoMetadata.getAttributions();



                // Create a FetchPhotoRequest.
                final FetchPhotoRequest photoRequest =
FetchPhotoRequest.builder(photoMetadata)
                        .setMaxWidth(1000) // Optional.
                        .setMaxHeight(300) // Optional.
                        .build();

placesClient.fetchPhoto(photoRequest).addOnSuccessListener(new
OnSuccessListener<FetchPhotoResponse>() {
                    @Override
                    public void onSuccess(FetchPhotoResponse
fetchPhotoResponse) {
                        Bitmap bitmap = fetchPhotoResponse.getBitmap();
                        placeCover.setImageBitmap(bitmap);
```

```
                    }
                })
                        .addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Log.e(TAG, "Place not found: ");
                            }
                        });
                }
            });

            //-----------------------------------------------------------------------------
----------------------------------------------------------
        }

        @Override
        public void onError(Status status) {
            // TODO: Handle the error.
            Log.i(TAG, "An error occurred: " + status);
        }

    });
  createTrip.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
          final FirebaseUser user = mAuth.getCurrentUser();
          final FirebaseFirestore db = FirebaseFirestore.getInstance();
          final CollectionReference userList = db.collection("Trips");
          Date currentTime = Calendar.getInstance().getTime();
          UUID uuid = UUID.randomUUID();
          String newTripId = uuid.toString();
          DocumentReference userInfo = userList.document(newTripId);
          Map<String, Object> note = new HashMap<>();
```

```
            note.put(CREATERID, user.getUid());
            note.put(LOCATION, location);
            note.put(TRIPID, newTripId);
            note.put(COVERPHOTOID, coverFromPhotoID);
            note.put(MEMBERID, Arrays.asList(user.getUid()));

            userInfo.set(note).addOnSuccessListener(new
OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    BottomNavigationView bottomNavigationView =
getActivity().findViewById(R.id.bottom_navigation);
                    Toast.makeText(getContext(), "Trip Added",
Toast.LENGTH_SHORT).show();
                    final NavController navController =
Navigation.findNavController(getView());
                    navController.navigate(R.id.tripFragment);
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(getContext(), "Error adding Trip!",
Toast.LENGTH_SHORT).show();
                }
            });
        }
    });

    }
}
```

OUTPUT:

Login Page :

Login

Username

Password

Login

Register

Forget password?

Register Page :

Register

Username

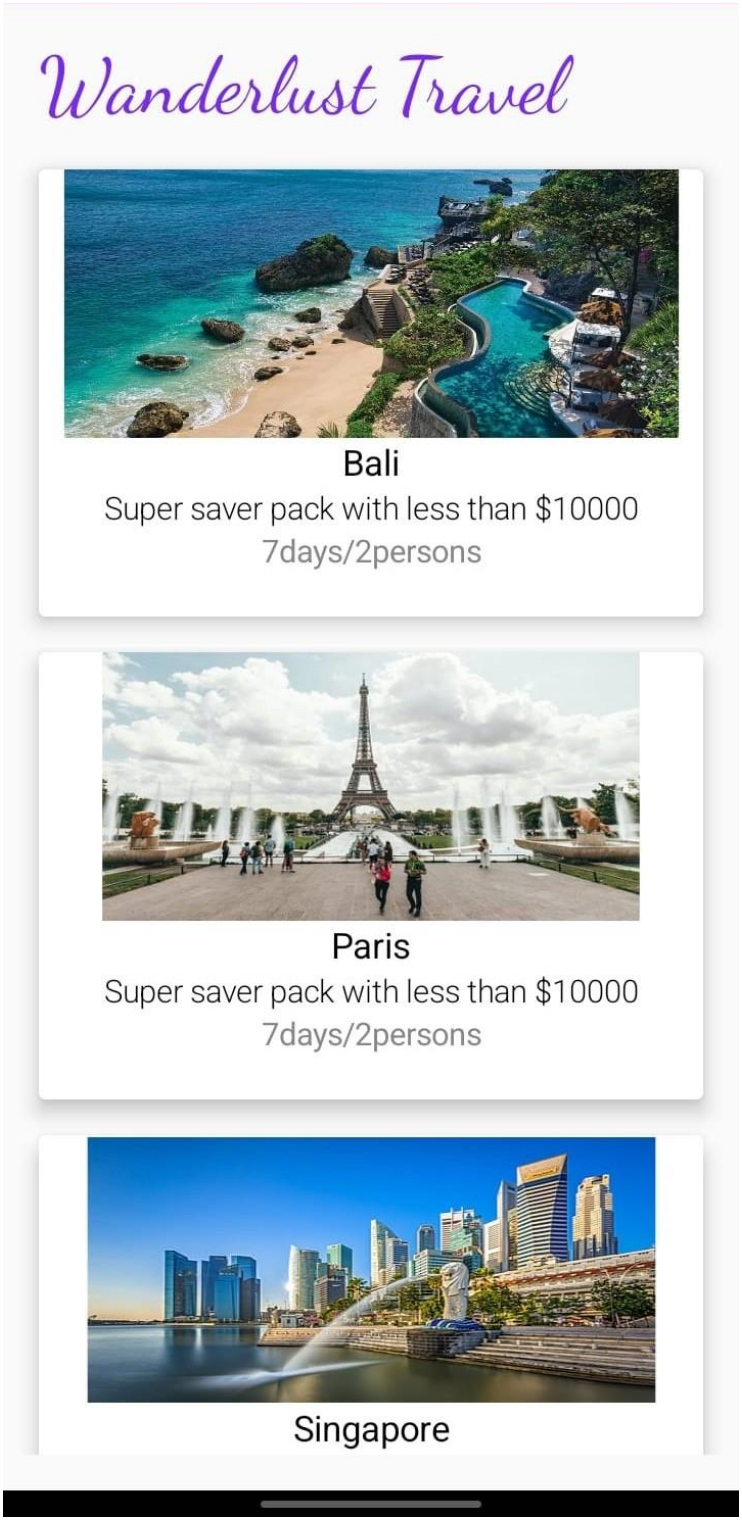
Email

Password

Register

Have an account? Log in

MainPage :


Wanderlust Travel
Bali
Super saver pack with less than $10000
7days/2persons
Paris
Super saver pack with less than $10000
7days/2persons
Singapore

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.

CONCLUSIONS

The pandemic has surely left a mark on the travel and tourism industry. This has been evident with the slump in revenue in the hospitality and tourism sector, which was the most affected economic sector. However, with the alleviation of travel restrictions, people have taken to vacationing in their favorite destinations, albeit with necessary precautions. The availability of travel planning apps helps people navigate through government policies during these difficult times. The convenience of booking flight/train tickets, reserving hotel rooms, hiring rental cars, and charting out a travel itinerary from the comfort of one's home on a single app has boosted the travel planner app industry. Travel agencies have recognized the merit of travel planning apps and are scouting the best means to learn how to create a travel app. The app development tools and case studies will help one have a better understanding of the app they wish to create and the requirements one must consider. Or simply skip the deliberations, and let us at CronJ help you! Having worked with the best, we know exactly how to create a travel planner app that caters to your needs in a cost-effective and efficient manner.