VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Hoang Le Minh, Dang Nguyen

TRAVEL APP FOR EASYVIET TRAVEL

A Mobile Application for Booking Travel Tours with EasyViet Travel

School of Technology
2020

ABSTRACT

Author	Le Minh Hoang, Dang Nguyen
Title	Travel App for EasyViet Travel
Year	2020
Language	English
Pages	80
Name of Supervisor	Timo Kankaanpää

This thesis concentrated on designing and developing a new mobile application (Android/iOS) for booking travel tour with EasyViet Travel. Nowadays, mobile phone is available to most people and a mobile application will help the user to book their favorite tours for holiday easily and improve the customer experience.

The project was planned and designed using the core principles of the Human-Computer Interactions (HCI). As a powerful framework for developing mobile application, Ionic was used to create a hybrid mobile application for both Android and iOS platform. The Ionic application was written with Angular with Type-Script. Firebase serves as both database server and backend server.

The project has achieved basic requirements until this working point.

Keywords Travel app, mobile application, Ionic, Android and iOS

CONTENTS

LIST OF ABBREVIATIONS

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
HCI	Human-Computer Interaction
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
UI	User Interface
UX	User Experience

LIST OF FIGURES AND TABLES

LIST OF CODE SNIPPETS

1 INTRODUCTION

1.1 Background

Nowadays, a smartphone is very popular with many convenient services. Because it is very easy to use a smartphone everywhere, mobile applications are growing in number. They provide assistance as well as entertain a lot of people. YouTube and Facebook mobile applications are even considered causing addition.

Although almost everyone in Vietnam has a smartphone and Internet access, there are just a few travel applications for booking tours. Most of tourism companies only allows booking offline or through sending message to their agent. This traditional method is also an inefficient way to market the service. Therefore, it is necessary to have a travel booking applications to simplify the booking process and advertise travel tours to more potential customers, and they can focus on enjoying their holiday.

1.2 About EasyViet Travel

Founded in 2010, EasyViet Travel is a company that focuses on providing travel services, such as organizing tours both locally and internationally, booking hotels and flights, supporting visa process and many more. After nine years of growing, the company is now expanding its business further into the digital market.

1.3 Objectives

The main objective of this thesis is to create a mobile application for booking travel tours. The application should come with great UI and be easy to use.

The application should allow the user to authenticate with Facebook and view the travel tour combos. They can search for combos according to their travel destinations of interest, and easily add or remove these combos from their favorites. The booking and payment process should also be easy to use for them. If they are not clear where to travel in the holiday, the application also provides some travel articles or news to suggest some great travel destination. Besides, users should be able to control easily their information, which they provided in the application.

2 TOOLS AND TECHNOLOGIES

2.1 Android

Android is a mobile operating system. It is based on the Linux kernel and other open source software, designed mainly for mobile devices such as smartphones and tablets. The main source code contributor and marketing commercial company for Android is Google. Google is also the main provider for a lot of Android default application.

Android has been the most popular operating system internationally on smartphones and tablets for almost a decade. Nowadays, there are billion monthly active Android users, and it is still the world's most popular operating system, and as of January 2020, there are also over 2.9 million apps in the Google Play Store. /1/

2.2 iOS

iOS is a mobile operating system, which was created and developed by Apple Inc. exclusively for its hardware. The operating system currently powers many of the company's mobile devices, including 1.4 billion iPhones, and over 100 million Pod Touch; it also powered the iPad before the introduction of iPadOS in 2019. Besides, it is the second most widely used mobile operating system worldwide after Android, which held 13.9% of the market share in 2019. /2/

Originally introduced in 2007 for the iPhone, iOS has been extended to support other Apple devices, such as the iPod Touch (September 2007) and the iPad (January 2010). As of March 2018, Apple's App Store has more than 2.1 million iOS applications, 1 million of which are native for iPads. /3/

2.3 Ionic Framework

Ionic Framework is an open source UI framework for building high-performance, high-quality mobile and web application by utilizing the web technologies — HTML, CSS, and JavaScript — with integrations for the most popular frameworks now like Angular and React. /4/

Ionic enables developers to build applications and mobile web from only a single codebase, which saves time a lot comparing to developing separate applications for Android and iOS. It also comes with plenty of components, which adapt to the platform very well, along with many native plugins.

The libraries and plugins used in this application is declared at *package.json* file:

```json
{
  "name": "easyviet-travel-app",
  "version": "0.0.1",
  "author": "Ionic Framework",
  "homepage": "https://ionicframework.com/",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/common": "~8.2.14",
    "@angular/core": "~8.2.14",
    "@angular/fire": "^6.0.0",
    "@angular/forms": "~8.2.14",
    "@angular/platform-browser": "~8.2.14",
    "@angular/platform-browser-dynamic": "~8.2.14",
    "@angular/router": "~8.2.14",
    "@ionic-native/core": "^5.23.0",
    "@ionic-native/facebook": "^5.23.0",
    "@ionic-native/splash-screen": "^5.23.0",
    "@ionic-native/status-bar": "^5.23.0",
    "@ionic/angular": "^5.0.7",
    "@ionic/storage": "^2.2.0",
    "@ngx-translate/core": "^12.1.2",
    "@ngx-translate/http-loader": "^4.0.0",
    "cordova-android": "8.1.0",
    "cordova-browser": "^6.0.0",
    "cordova-ios": "^5.1.1",
    "cordova-plugin-facebook4": "^6.4.0",
    "cordova-sqlite-storage": "^5.0.0",
    "core-js": "^2.5.4",
    "firebase": "^7.13.2",
    "ion2-calendar": "^3.0.0-rc.0",
    "ionic5-star-rating": "https://github.com/JeongJun-Lee/ionic5-
star-rating/tarball/master",
    "moment": "^2.24.0",
    "ngx-paypal": "^6.1.0",
    "rxjs": "~6.5.1",
    "tslib": "^1.9.0",
    "zone.js": "~0.9.1"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.803.20",
    "@angular/cli": "~8.3.23",
    "@angular/compiler": "~8.2.14",
    "@angular/compiler-cli": "~8.2.14",
    "@angular/language-service": "~8.2.14",
    "@ionic/angular-toolkit": "^2.1.1",
    "@types/jasmine": "~3.3.8",
    "@types/jasminewd2": "~2.0.3",
    "@types/node": "~8.9.4",
    "codelyzer": "^5.0.0",
    "cordova-plugin-device": "^2.0.2",
    "cordova-plugin-ionic-keyboard": "^2.2.0",
    "cordova-plugin-ionic-webview": "^4.1.3",
    "cordova-plugin-splashscreen": "^5.0.2",
    "cordova-plugin-statusbar": "^2.4.2",
    "cordova-plugin-whitelist": "^1.3.3",
    "jasmine-core": "~3.4.0",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "~4.1.0",
    "karma-chrome-launcher": "~2.2.0",
    "karma-coverage-istanbul-reporter": "~2.0.1",
    "karma-jasmine": "~2.0.1",
    "karma-jasmine-html-reporter": "^1.4.0",
    "protractor": "~5.4.0",
    "ts-node": "~7.0.0",
    "tslint": "~5.15.0",
    "typescript": "~3.4.3"
  },
  "description": "An Ionic project",
  "cordova": {
    "plugins": {
      "cordova-plugin-whitelist": {},
      "cordova-plugin-statusbar": {},
      "cordova-plugin-device": {},
      "cordova-plugin-splashscreen": {},
      "cordova-plugin-ionic-webview": {
        "ANDROID_SUPPORT_ANNOTATIONS_VERSION": "27.+"
      },
      "cordova-plugin-ionic-keyboard": {},
      "cordova-plugin-facebook4": {
        "APP_ID": "655763464969239",
        "APP_NAME": "easyviet-travel-app",
        "FACEBOOK_HYBRID_APP_EVENTS": "false",
        "FACEBOOK_ANDROID_SDK_VERSION": "5.13.0"
      },
      "cordova-sqlite-storage": {}
    },
    "platforms": [
      "android",
      "browser",
      "ios"
    ]
  }
}
```

Code Snippet 1. Libraries and Ionic plugins in package.json file

2.4 Angular Framework

Angular is a very popular platform and framework for front-end development. It is mainly used for building single-page client applications using HTML and Type-Script. Angular is written in TypeScript. It implements core and optional function-ality as a set of TypeScript libraries, which can be imported into the applications.

Ionic allows developers to develop a mobile application with Angular. The ad-vantages of Ionic are its good architecture and useful libraries.

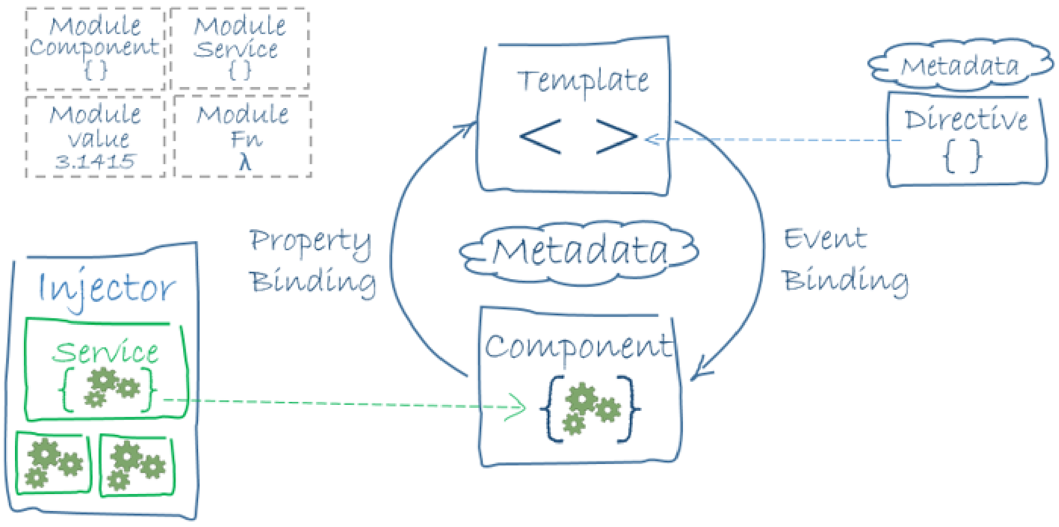


Figure 1. Architecture of an Angular application

The architecture of an Angular application depends on some basic concepts. The fundamental building blocks are *NgModules*, which provide a compilation context for *components*. NgModules accumulate a related code into functional sets, and an Angular application has been defined by a set of NgModules. An application always has at least a *root module* that enables bootstrapping, and usually has several other *feature modules.*

2.4.1 Views and Templates

Views are the HTML template with Angular binding to make sure that Angular will be able to control and modify the front-end depending on the program logic and data.

A template combines HTML with Angular markup that can alter the HTML elements before they are shown. Template directives offer the program logic, and binding markup links the data of application and the HTML DOM.

Before displaying a view for the user, Angular evaluates the directives and solves the binding syntax in the template to update the HTML elements and the DOM, in accordance with the application data and logic. Angular also supports two-way data binding, which means that the modifications in the HTML DOM, for example new input data from the user, are also reflected in the program data model. It helps the developer to save a lot of time updating those data.

2.4.2 Service

For data or logic that is not linked to a particular view, or need to be shared among components, Angular helps web developers with the service class. Service providers can be imported into components as dependencies, helping the developer to write code in a modular, reusable, and efficient way. Components use services, which give a particular functionality, which is not directly linked to the views. Besides, dependency injection allows the keeping the component classes lean and efficient. The components now do not retrieve the data from the server, verify user input, or write log directly to the console because they already delegate those tasks to the services.

2.4.3 Routing

The Angular Router NgModule offers a convenient service that enables to the definition of a navigation path, which is similar to some backend server framework, among the different application states and view hierarchies in the application. It may sound complicated, but the router is patterned on the normal browser navigation conventions:

- Enter a URL in the address bar and the browser navigates to a corresponding page.
- Click on links and the browser should navigate user to a new web page.
- Click the browser's back and forward buttons and the browser navigates backward and forward through the history of pages you have seen.

Instead of routing the user to pages in the same way as HTML, the router maps URL-like paths to views. When the user performs an action, such as clicking on a link that would transfer the user to a new web page in the browser, the router stops the browser's behavior, then it can display or hide view depend on the application logic. /5/

Routing can be implemented easily in both Angular templates and logic TypeScript file with router service.

More detail information about Angular could be found at https://angular.io/docs

2.5 TypeScript

TypeScript is an open-source programming language built on JavaScript, the most popular language. It is developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and is a great improvement to JavaScript, for example it encourages static typing or allow the developer to create class and interface in a clearly way. Thank to these improvements, TypeScript is much easier to debug and develop, as well as it provides a better security for the web application. Therefore, TypeScript is also designed for better development of huge applications and trans-compiles to JavaScript.

TypeScript can be used to develop web applications for both client-side and server-side execution (as with Node.js or Angular). Actually, it is the main programming language used for Angular Framework. Besides, it is also recommended by many industrial companies to replace JavaScript. /6/

Information and tutorial about TypeScript can be found at https://www.typescript-lang.org/

2.6 Firebase

Firebase is a mobile and web application development platform which has been developed by Firebase, Inc. in 2011, then acquired by Google in 2014. At the moment, Firebase platform currently has 19 products and services, which integrate with each other very well, and they are utilized by more than 1.5 million applications. /7/

In this project, Firebase Realtime Database will be used as an application database as well as proving API for the Ionic application. Firebase Authentication will also help the application to authenticate users easily.

2.6.1 Firebase Realtime Database

The Realtime Database is a NoSQL database. It is clear that NoSQL has a lot of different optimizations and functionality in comparison to a relational database.

In NoSQL database, data is stored as JSON. Firebase also allows database to be synchronized in real-time to every connected client. When building and deploying cross-platform applications with iOS, Android, and JavaScript SDKs, all of the application clients share one Realtime Database instance and automatically get updates with the latest available data. Besides, the data is persisted locally, and even while the user has no Internet connection, these real-time events are continuing to fire, providing the end user a highly responsive experience. When the device regains connection, the Realtime Database synchronizes the local changes to the data with the remote updates that happened while the user was offline, merging any conflicts automatically. Firebase can also provide assistance; help the application's data to meet the requirements at scale by dividing the data across several database instances in the same Firebase project. /8/

2.6.2 Firebase Authentication

Firebase Authentication is a very convenient service of Firebase. It offers backend services to validate users to the application. It provides support for authentication using email and passwords or popular authentication providers such as Google, Facebook, and more. Therefore, developers do not need to worry much about authentication and security, and they can spend more time on developing the main application.

After a successful sign in, developers can access the user's basic profile information, and the user's access to data stored in other Firebase products can be controlled. Then, developers can also use the provided authentication token in the response data to verify the identity of users with backend services. Developers can also access user information from other identity providers to help user have great experiences with the application. /9/

More information about Firebase can be found at https://firebase.google.com/

2.7 Facebook Login

Facebook Login is a very popular authentication method nowadays. It provides a quick and easy way for users to create new accounts and log into an application. It is used to allow better user experiences, for example:

- Authentication: Facebook Login allows users to quickly and easily create an account in the application without having to set (and likely later forget) a password. It is also more suitable for mobile applications, which can make the user feel a bit annoying when creating a new account for a simple task. This straightforward and convenient experience can convert to a higher number of users for the application. After allowing the application to access user's basic information on Facebook and creating an account on the application, users are now able to login quickly with a single click on "Login with Facebook" button. Besides, Facebook also helps the application to validate user email addresses, which implies that developers are able to connect to that user more in the future.
- Personalized Settings: Facebook Login allows developers to access information, which would be complex or illegal to collect via the application's own registration form with a long term and condition, which no user would read anyway. Even just importing the user's profile image from Facebook and setting it as the avatar in the application account, the application already

makes the user feel happier, and offers them a stronger connection when comparing between similar applications. /9/

More information about Facebook API can be found at https://developers.facebook.com/

2.8 Cordova

Apache Cordova is an open-source mobile development framework, which helps to develop an Ionic application with a lot of native plugins. Cordova enables developers to use standard web technologies - HTML5, CSS3, and JavaScript for cross-platform development. Applications run within wrappers, which is aimed at the specific platform, and depend on the API bindings to access each device's native resources, for example camera, map, and installed application.

The architecture of Cordova:

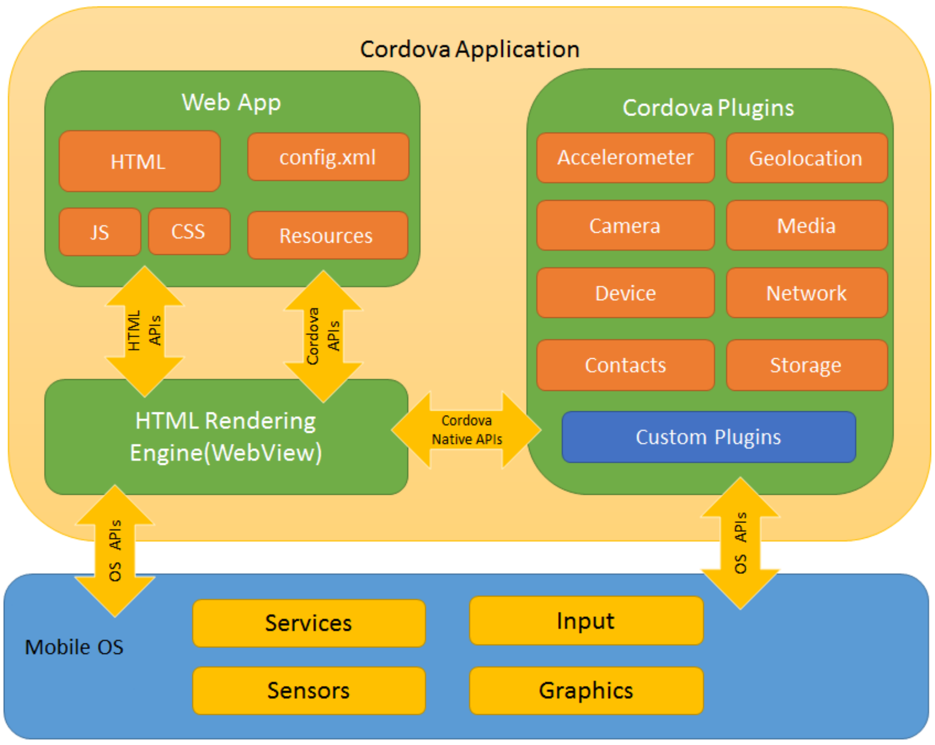


Figure 2. Architecture of Cordova /10/

More information about Cordova can be found at https://cordova.apache.org/docs/en/latest/guide/overview/index.html

3 USER INTERFACE DESIGN

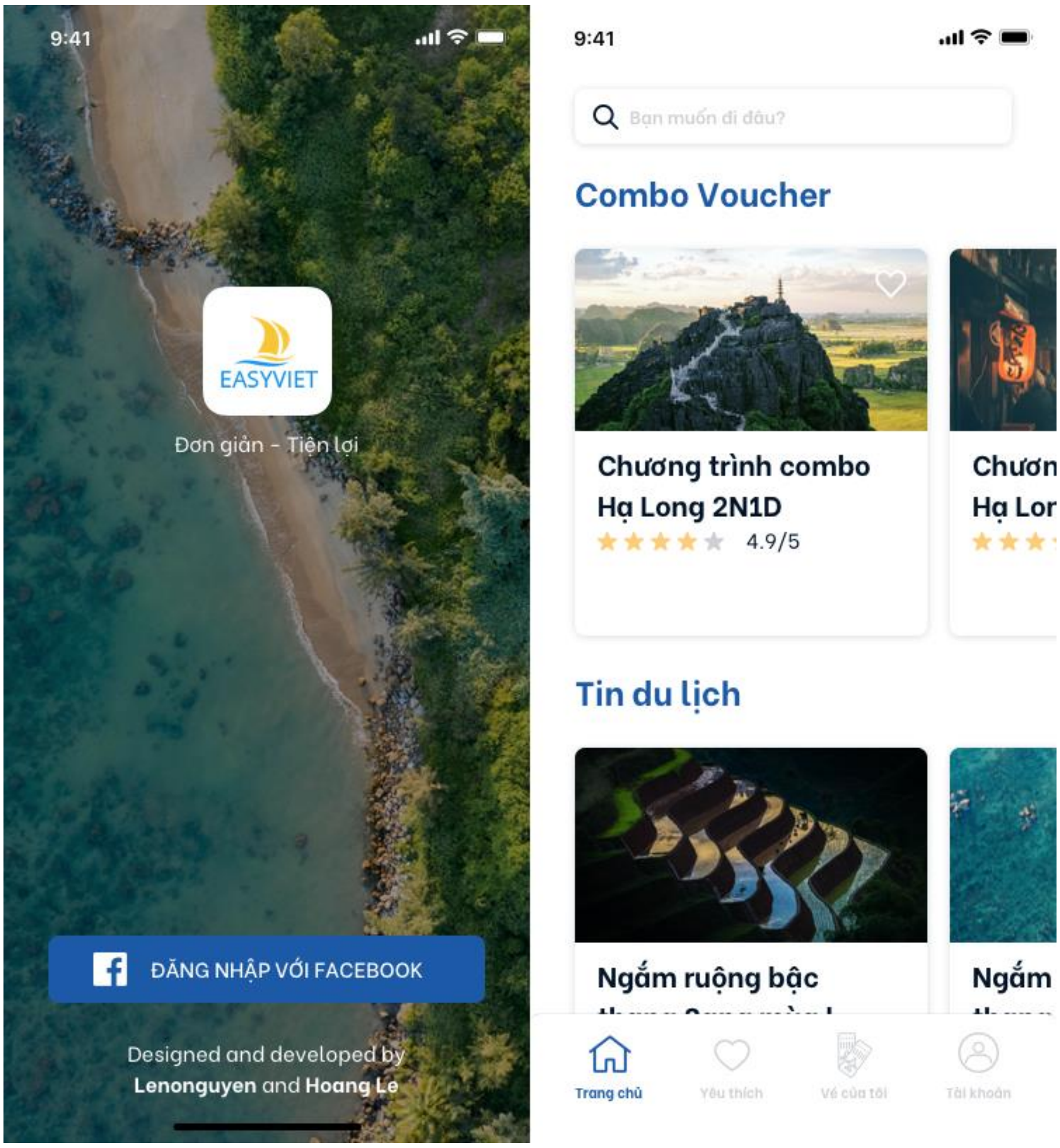
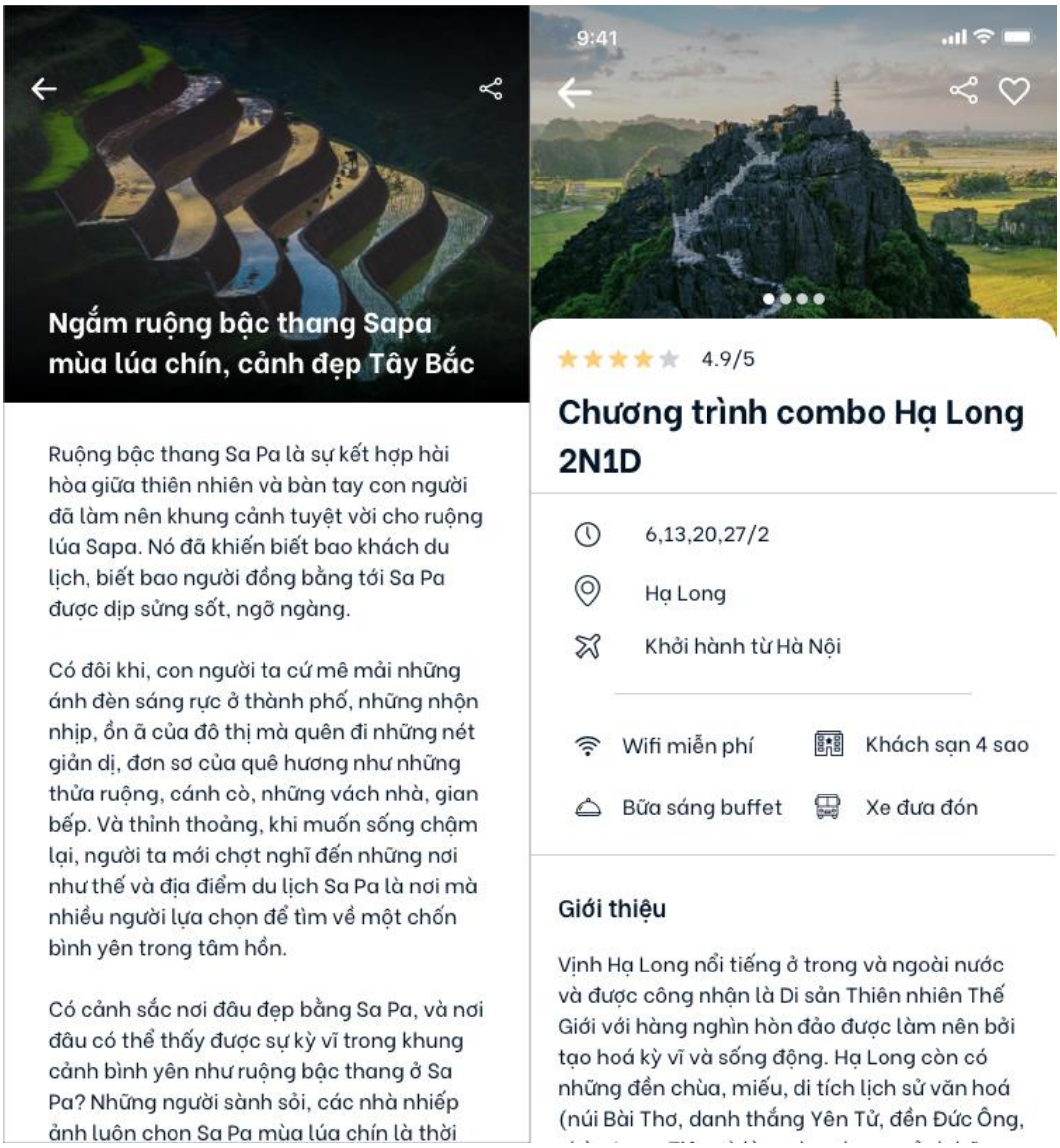


Figure 3. Log in screen and Home Page

The Log in Screen is designed to have the company logo on top of a destination photo, which is used to set the travel vibe for the application. The main method used for logging in is Facebook Authentication due to the high amount of Facebook users in Vietnam, which will help to make the log in process easier for the user.

The Home Page focus on 2 factors: Combo Voucher and Articles. This decision is made based on a few meetings with the Client, to align the application with the company's business goal.



Ngắm ruộng bậc thang Sapa mùa lúa chín, cảnh đẹp Tây Bắc

Ruộng bậc thang Sa Pa là sự kết hợp hài hòa giữa thiên nhiên và bàn tay con người đã làm nên khung cảnh tuyệt vời cho ruộng lúa Sapa. Nó đã khiến biết bao khách du lịch, biết bao người đồng bằng tới Sa Pa được dịp sửng sốt, ngỡ ngàng.

Có đôi khi, con người ta cứ mê mải những ánh đèn sáng rực ở thành phố, những nhộn nhịp, ồn ã của đô thị mà quên đi những nét giản dị, đơn sơ của quê hương như những thửa ruộng, cánh cò, những vách nhà, gian bếp. Và thỉnh thoảng, khi muốn sống chậm lại, người ta mới chợt nghĩ đến những nơi như thế và địa điểm du lịch Sa Pa là nơi mà nhiều người lựa chọn để tìm về một chốn bình yên trong tâm hồn.

Có cảnh sắc nơi đâu đẹp bằng Sa Pa, và nơi đâu có thể thấy được sự kỳ vĩ trong khung cảnh bình yên như ruộng bậc thang ở Sa Pa? Những người sành sỏi, các nhà nhiếp ảnh luôn chọn Sa Pa mùa lúa chín là thời

9:41

4.9/5

Chương trình combo Hạ Long 2N1D

6,13,20,27/2

Hạ Long

Khởi hành từ Hà Nội

Wifi miễn phí | Khách sạn 4 sao

Bữa sáng buffet | Xe đưa đón

Giới thiệu

Vịnh Hạ Long nổi tiếng ở trong và ngoài nước và được công nhận là Di sản Thiên nhiên Thế Giới với hàng nghìn hòn đảo được làm nên bởi tạo hoá kỳ vĩ và sống động. Hạ Long còn có những đền chùa, miếu, di tích lịch sử văn hoá (núi Bài Thơ, danh thắng Yên Tử, đền Đức Ông,

Figure 4. Article Details and Combo Details

The Article Details will serve as a medium for the company to share the travel news, promotions, or to give the users some suggestions on where to travel. The Combo

Details, on the other hand, focus on showing a brief description of the travel destination, along with information of the trip such as travel date, the basic utilities of the trip and the exact arrival destination.

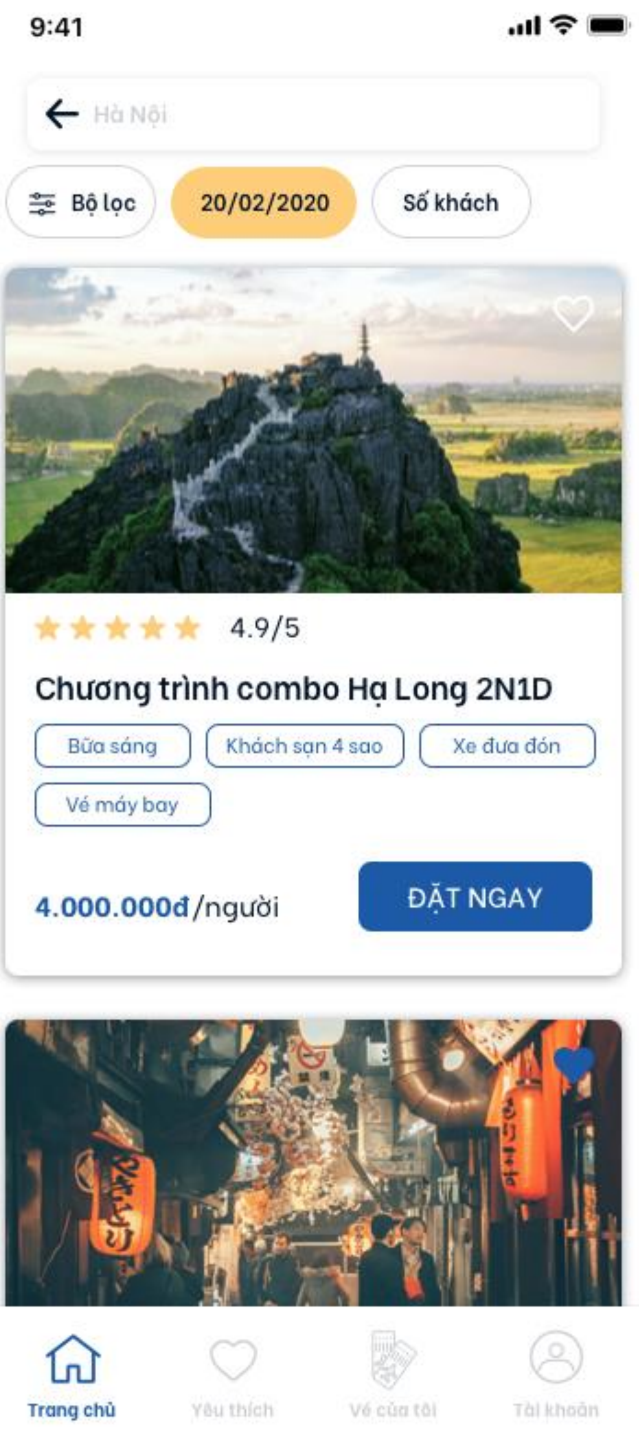
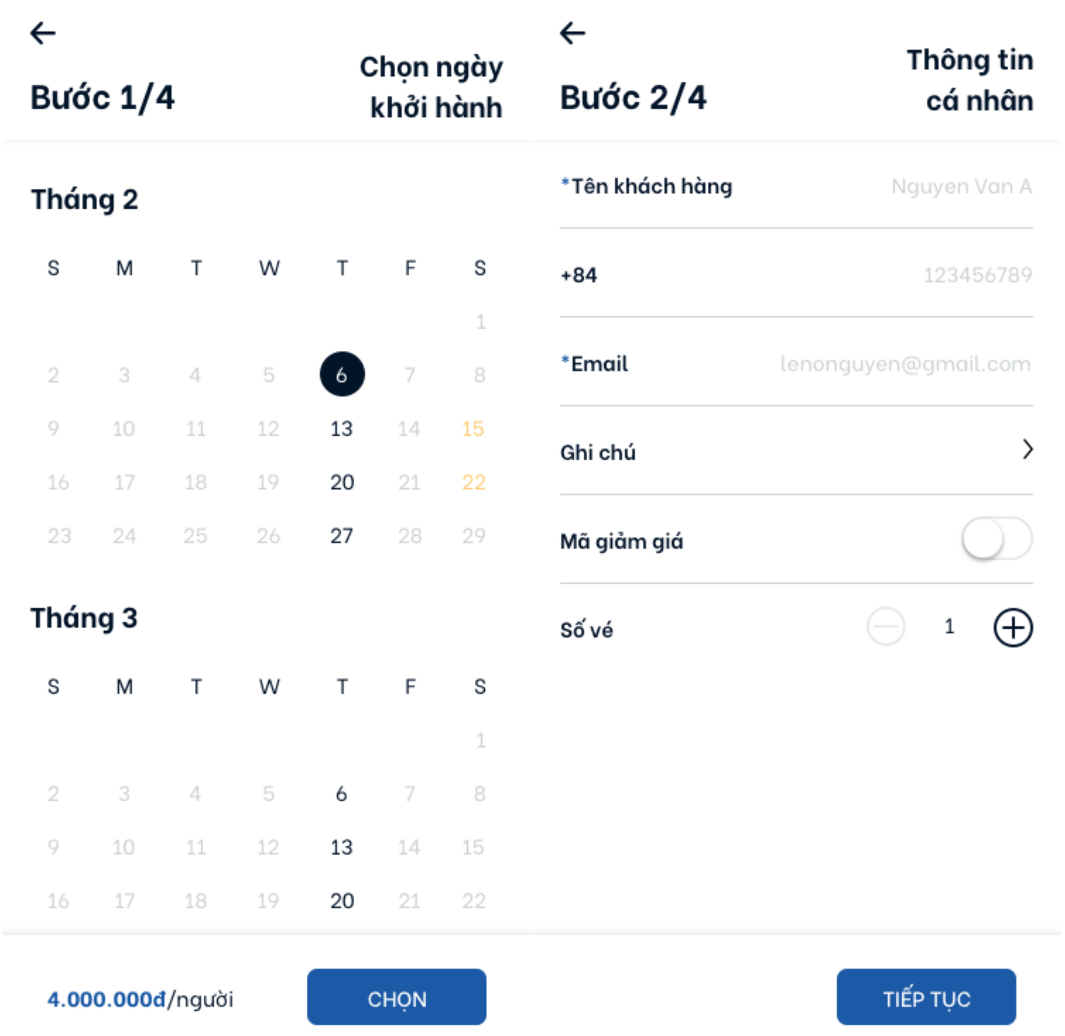


Figure 5. Search Screen

The Search Screen allows the users to find the desired travel combo, based on the provided destination. The Combo is shown with the name, price, and basic information to help the users with decision- making.



*Tên khách hàng Nguyen Van A

+84 123456789

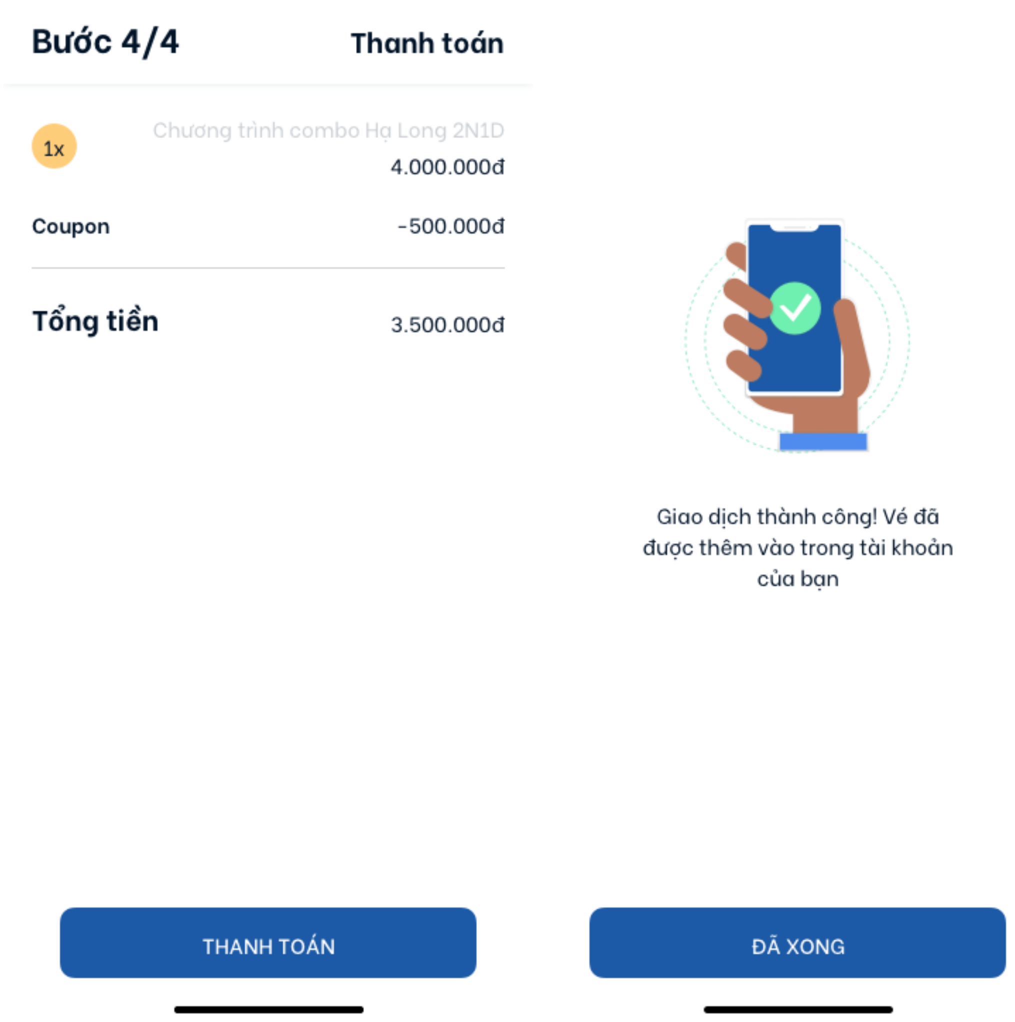
*Email lenonguyen@gmail.com

Số vé 1

Figure 6. Date Selection and Personal Information

The Booking flow includes four steps: Date Selection, Personal Information, Save Card for payment and Checkout form. However, during the development process, the Save Card for payment was removed due to the acknowledgement of the privacy policy of Vietnam.

After choosing a travel combo, the application will show a list of dates for the users to choose. Then, the users will be asked to fill in the Personal Information.



Bước 4/4 Thanh toán

1x	Chương trình combo Hạ Long 2N1D 4.000.000đ
Coupon	-500.000đ
Tổng tiền	3.500.000đ

Giao dịch thành công! Vé đã được thêm vào trong tài khoản của bạn

Figure 7. Checkout and Successful Payment

The Checkout Screen is implemented to show the total price of a trip, including the coupon promotion and other expenses. The payment will then be executed using PayPal and the users will be prompted to a Successful Payment Screen.

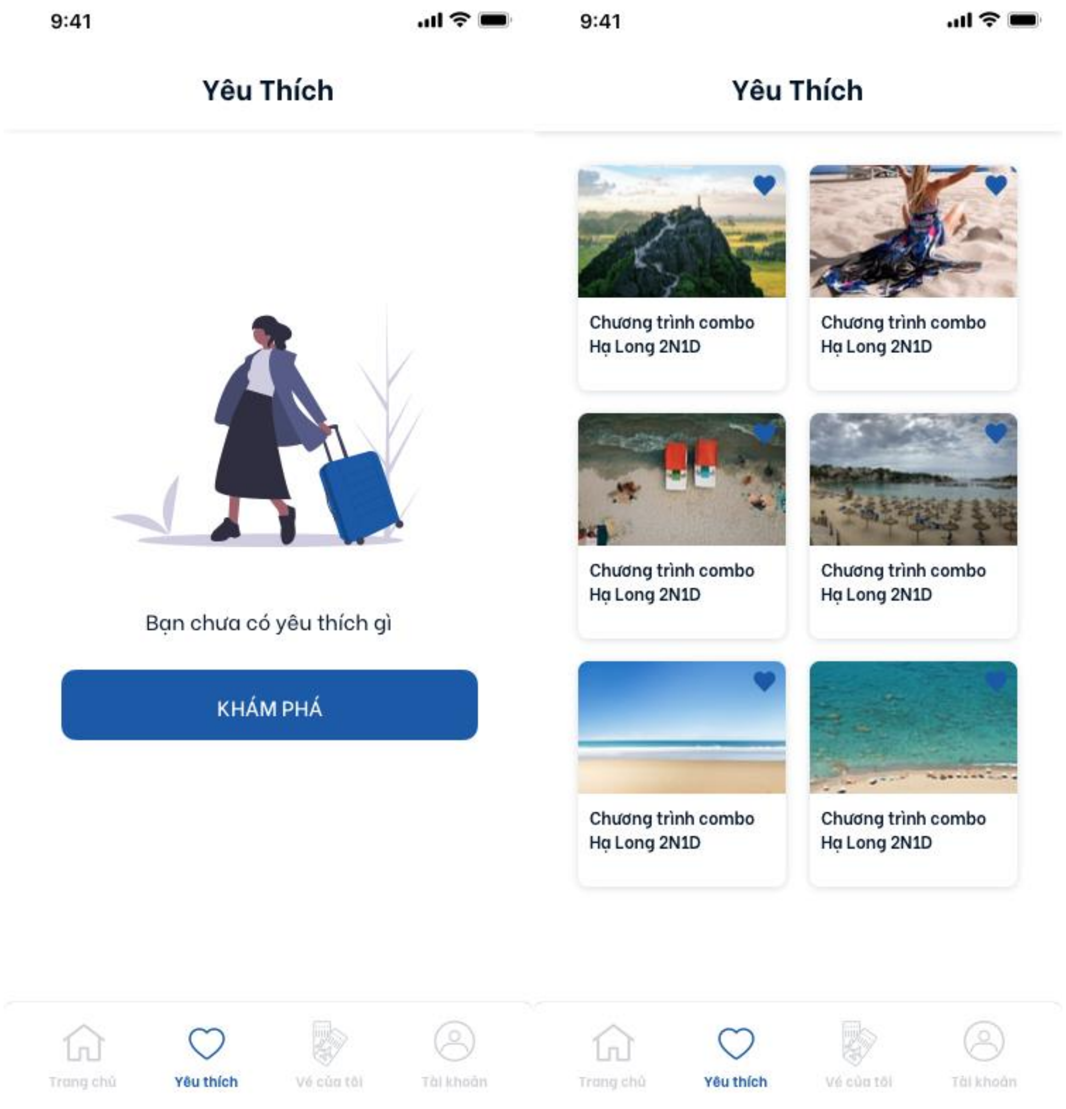

9:41
Yêu Thích
Bạn chưa có yêu thích gì
KHÁM PHÁ
Trang chủ
Yêu thích
Vé của tôi
Tài khoản
9:41
Yêu Thích
Chương trình combo Hạ Long 2N1D
Chương trình combo Hạ Long 2N1D
Chương trình combo Hạ Long 2N1D
Chương trình combo Hạ Long 2N1D
Chương trình combo Hạ Long 2N1D
Chương trình combo Hạ Long 2N1D
Trang chủ
Yêu thích
Vé của tôi
Tài khoản

Figure 8. Favourite Screen

The Favourite Screen allows the users to save the travel combos for later. Every combo has a heart shaped button on the top right, which serve as a toggle for the users to save the combo in the Favourite list.

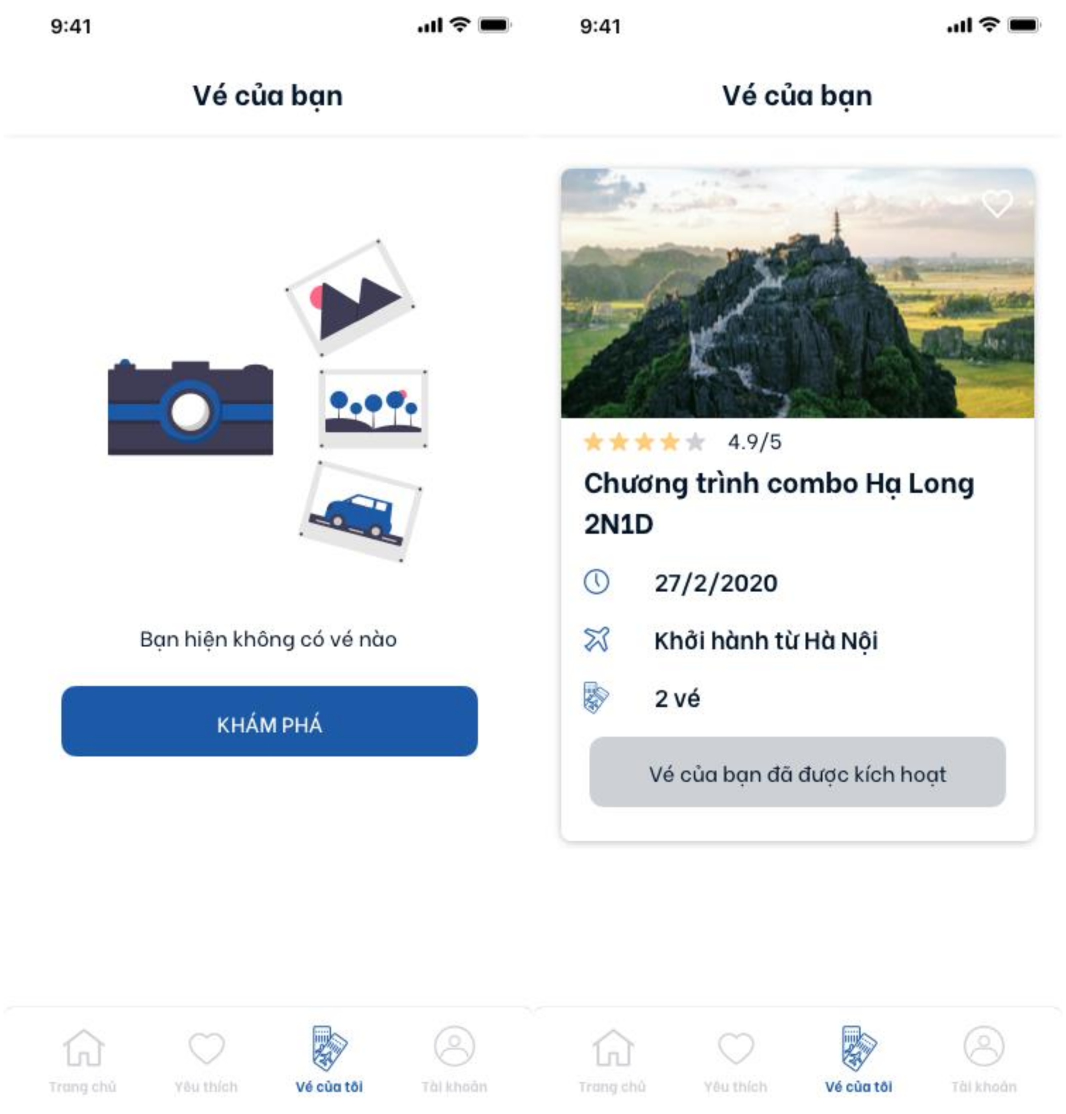
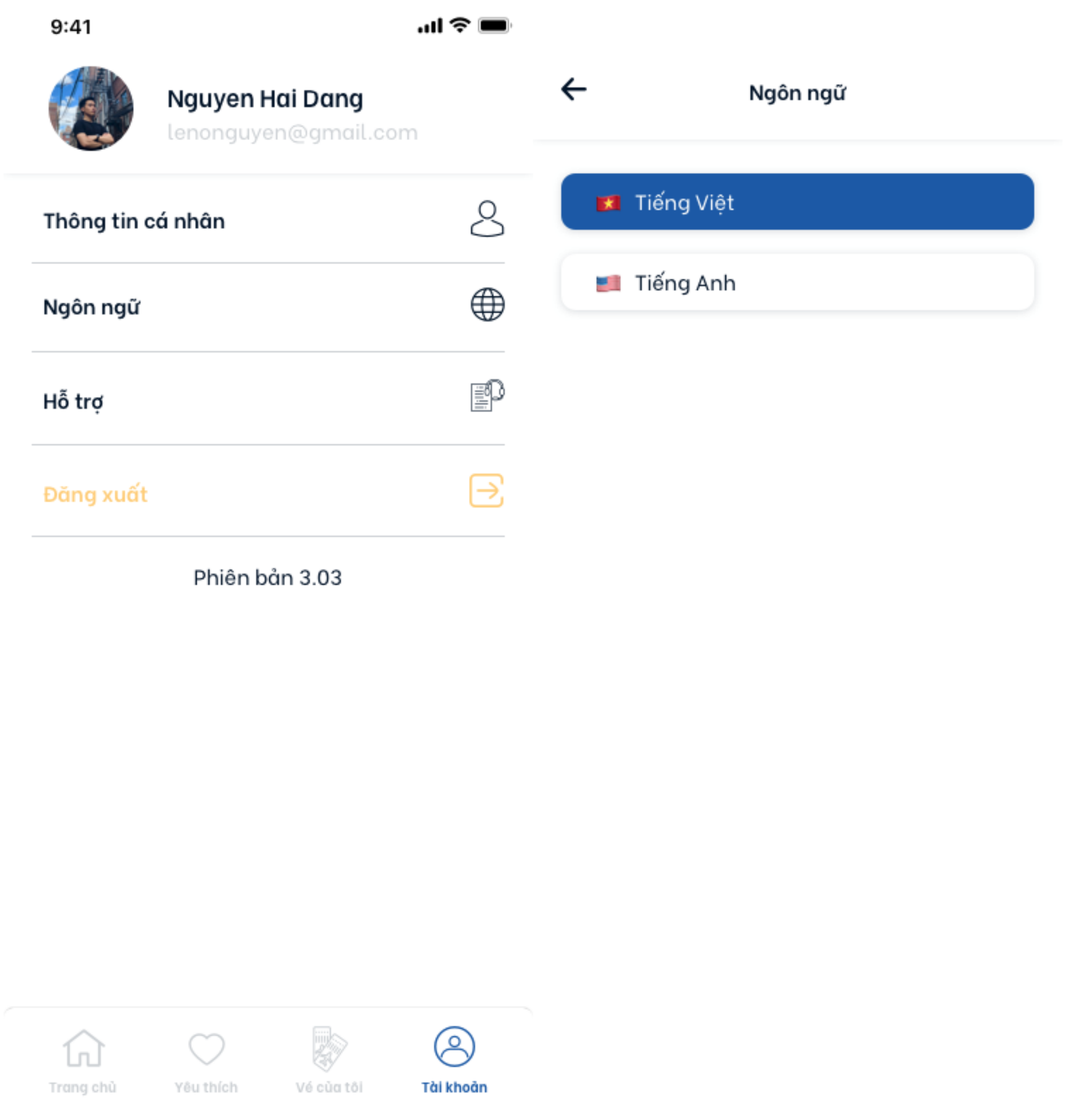


Figure 9. My Ticket Screen

After a combo is booked, it will be moved to the My Ticket Screen. This Screen allows the users to view the booked combo, along with basic information of that combo such as Date of the combo, destination, and number of tickets.



9:41

Nguyen Hai Dang
lenonguyen@gmail.com

Thông tin cá nhân

Ngôn ngữ

Hỗ trợ

Đăng xuất

Phiên bản 3.03

Ngôn ngữ

Tiếng Việt

Tiếng Anh

Figure 10. My Account and Language Settings

The Account Screen is designed for the users to easily edit the Personal Information, set the Language of the application, contact the company for support, and log out.

4 SYSTEM DESCRIPTION

4.1 Software Requirements Specification

These are the requirements that were agreed upon by the company. The requirements are divided into 2 categories:

- Functional: The functions of the application are divided into must have, should have, and nice to have features.
- Non-functional: The characteristics of the application.

Table 1 specifies the functional requirements of the application.

Table 2 specifies the non-functional requirements of the application.

Priority levels:

1 - Must have
2 - Should have
3 - Nice to have

Reference	**Description**	**Priority**
F1	Displaying the details of the trips	1
F2	Displaying the status of booked trips	1
F3	Login with social media	1
F4	Displaying articles details	1
F5	Log out of the application	1
F6	Payment with PayPal	1

F7	Search bar to search for trips	2
F8	Bookmark favourite trips	2
F9	Ability to change the application language	3

Table 1. Function

Characteristic	**Description**
Usability	The UX will be optimized so that the user can operate with the application at ease.
Safety	Only authorized users can access the application features
Response time	The system should respond to user query in less than 1 seconds, but the response data depends mostly on user's Internet connection speed
Aesthetic	The UI should be clean and matched with the company's branding

Table 2. Non-functional characteristics

4.2 Use Case Diagram

The use case diagram includes all user's operations with the application. There are several use cases in the diagram below. The principal use case is to display details of a travel combo, book a travel combo and display booked trips.

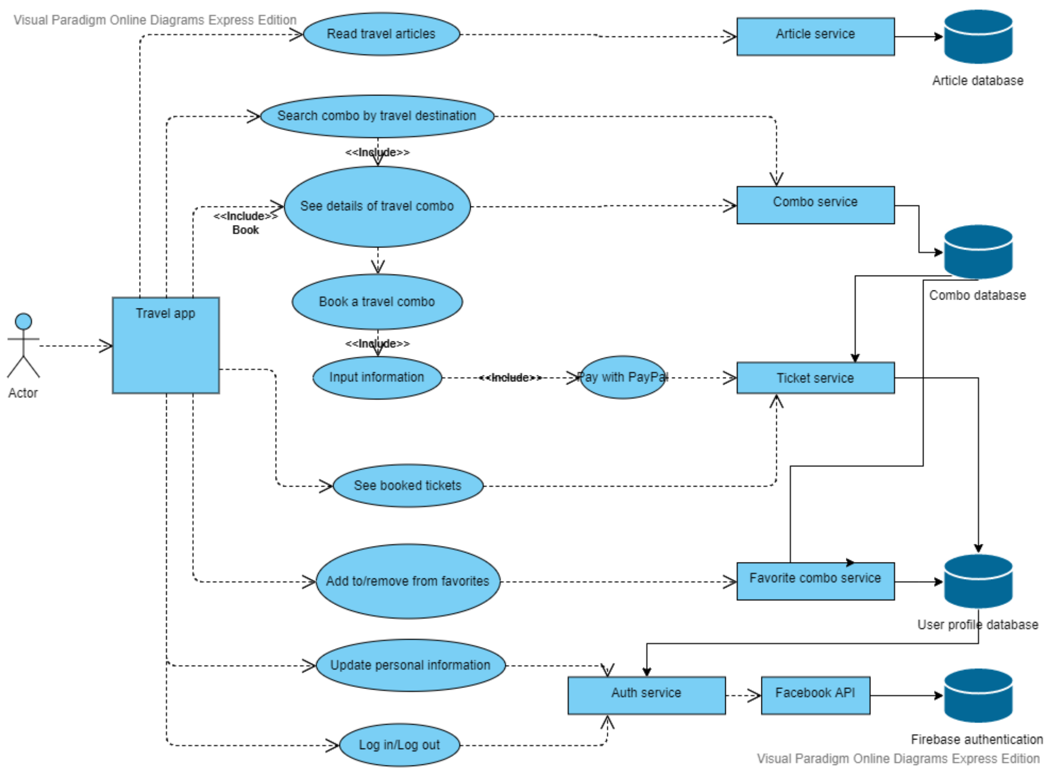


Figure 11. Use Case Diagram

5 APPLICATION DESIGN

5.1 Software Architecture

This system has a mobile client with the travel app installed and Firebase database. There is no server because Firebase allows clients to connect to the database without server.

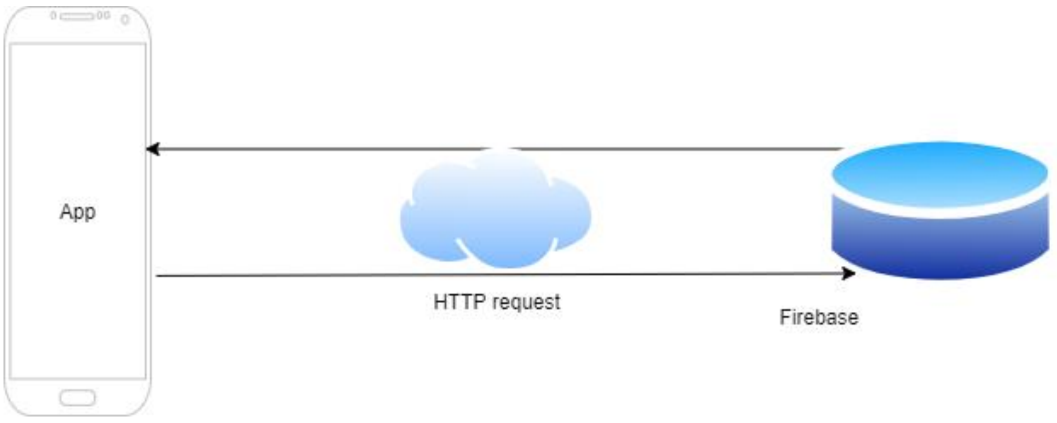


Figure 12. Application architecture

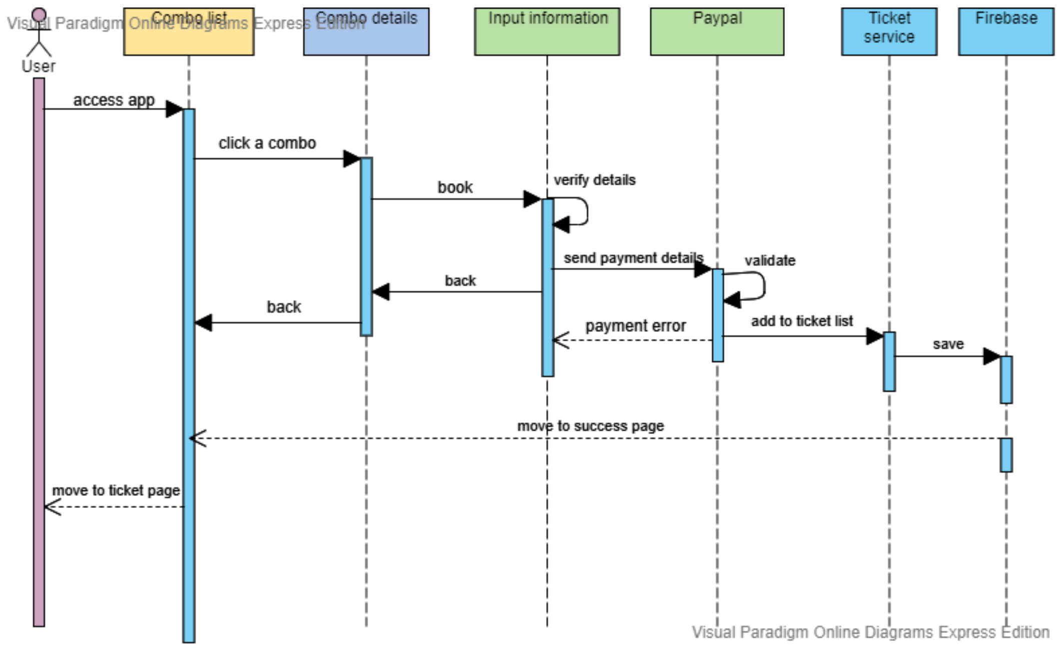
The client connects to Firebase through HTTP requests established by Firebase Angular library. All the connection is handled through Angular services.

5.2 Sequence Diagrams

The sequence diagram present specific use case to show how Angular services and pages interact with each other. Several main use cases of the application are illustrated with sequence diagrams in this part.

5.2.1 Booking a travel combo

The main function of the application is to book a specific combo. Users can choose a specific combo from the combo list. After inputting the personal information and paying with PayPal, users expect the ticket will be available in the Ticket tab of the application.



Visual Paradigm Online Diagrams Express Edition
Visual Paradigm Online Diagrams Express Edition

Figure 13. Sequence diagram of booking a travel combo

5.2.2 See details of a specific combo

An important use case is to see the details of a specific travel combos. After clicking on a specific combo in the combo list, users expect the details of the selected travel combo will be represented. This process is illustrated in figure 13.

5.2.3 Search for travel combos by travel destination

Another use case is to search for travel combos by user’s travel destination. After inputting the travel destination to the search bar, users expect the correct travel combo will be found. This process is illustrated in figure 14.

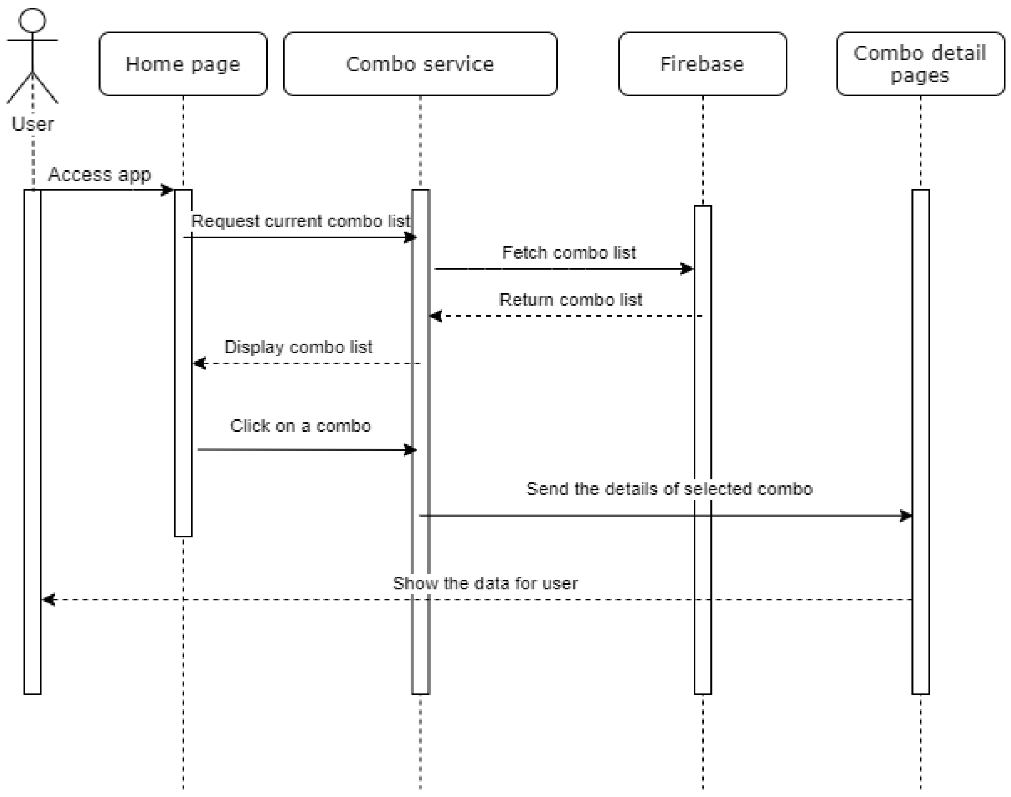


Figure 14. Sequence diagram of showing details of a travel combo

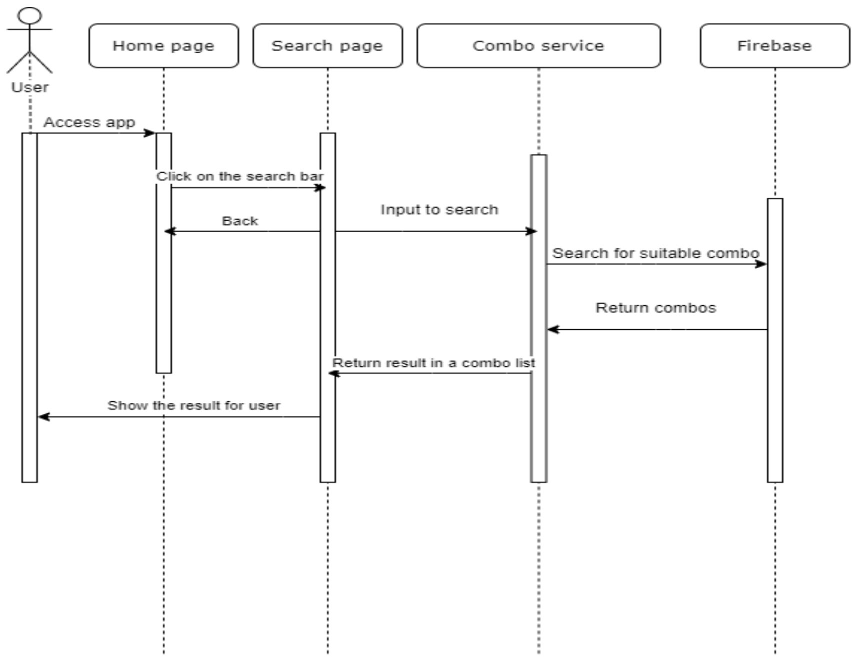


Figure 15. Sequence diagram of searching a combo by travel destination

6 IMPLEMENTATION

The implementation of the travel application is described in this chapter.

6.1 Travel Combos

The travel combos are the travel tour data, which are declared with Combo class. It contains all information about a combo tour: title, images, price, start destination, travel destination, and description.

```
export class Combo {

  constructor(
    public id: string,
    public title: string,
    public imgUrl: string,
    public price: number,
    public discount: number,
    public rate: number,
    public startDest: string,
    public travelDest: string,
    public wifi: boolean,
    public breakfast: boolean,
    public hotelRating: number,
    public taxi: boolean,
    public description: string,
    public availableWeek: AvailableWeek,
    public coupon: Coupon,
  ) { }
}

export interface Coupon {
  [key: string]: number;
}

export class AvailableWeek {
  constructor(
    public mon: boolean,
    public tue: boolean,
    public wed: boolean,
    public thu: boolean,
    public fri: boolean,
    public sat: boolean,
    public sun: boolean,
  ) { }
}
```

Code Snippet 2. Combo class declaration

The list of combos is loaded in the combo service from Firebase:

```
  private _combos = new BehaviorSubject<Combo[]>([]);

  constructor(private db: AngularFireDatabase) { }

  get combos() {
    this.db.list<Combo>(`combos`).snapshotChanges().sub-
scribe(res => {
      const comboList = [];
      res.forEach(item => {
        const combo = item.payload.toJSON();
        const id = item.key;
        comboList.push({...combo, id});
      });
      this._combos.next(comboList);
    });
    return this._combos.asObservable();
  }
```

Code Snippet 3. Function to load combos from Firebase

Then, combos can be loaded from this service and display with the template:

```
<ion-slides [options]="slideOpts" class="tab1-slides">
          <ion-slide *ngFor="let combo of loadedCombos">
            <ion-card>
              <img [src]="combo.imgUrl">
              <ion-button
                fill="clear"
                color="light"
                class="fav-button"
                (click)="onAddingFavCombo(combo.id, combo.ti-
tle, combo.imgUrl)"
              >
                <ion-icon name="heart-outline"></ion-icon>
              </ion-button >
              <ion-card-header
                [routerLink]="[
                  '/',
                  'tabs',
                  'tab1',
                  'combo-detail',
                  combo.id
                ]"
              >
                <ion-card-title class="combo-title"> {{ combo.ti-
tle }}</ion-card-title>
                <ion-card-subtitle>
                  <ionic5-star-rating #rating
                    activeIcon = "star"
                    defaultIcon = "star-outline"
                    activeColor = "#ffce73"
                    defaultColor = "#cdd1d5"
                    readonly="false"
                    rating="{{combo.rate}}"
                    fontSize="13px"
                  >
                  </ionic5-star-rating>
                  {{ combo.rate }} / 5
                </ion-card-subtitle>
              </ion-card-header>
            </ion-card>
          </ion-slide>
        </ion-slides>
```

Code Snippet 4. Template to display a list of combos

The detail of each combo also needs to be shown clearly and accessed easily. Therefore, whenever the user clicks on a combo in the combo list, the application has to find the right combo from combo list:

```
  getCombo(id: string) {
    return this.combos.pipe(
      take(1),
      map(combos => {
        return {...combos.find(c => c.id === id)};
      })
    );
  }
```

Code Snippet 5. Find a combo with ID

Then, it will display detail of the selected combo: starting destination, travel destination, services, prices, etc. as well as the button to book the combo with the Angular template:

```
<ion-header>
  <ion-toolbar class="ion-color" style.back-
ground="'url(' + combo.imgUrl + ')'" >
    <ion-buttons>
      <ion-back-button icon="arrow-back-out-
line" slot="start" text=""></ion-back-button>
    </ion-buttons>
      <ion-img [src]="combo.imgUrl"></ion-img>
  </ion-toolbar>
</ion-header>

<ion-content padding>
  <ion-grid>
    <ion-row>
      <ion-col size="3">
        <ionic5-star-rating #rating
          activeIcon = "star"
          defaultIcon = "star-outline"
          activeColor = "#ffce73"
          defaultColor = "#cdd1d5"
          readonly="false"
          rating="{{combo.rate}}"
          fontSize="13px"
        >
        </ionic5-star-rating>
      </ion-col>
      <ion-col>
        {{ combo.rate }} / 5
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        <ion-label color="dark"> {{ combo.title }}</ion-label>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        <ion-label>
          <ion-icon name="time-outline"></ion-icon>
          {{ availableDay }}
        </ion-label>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        <ion-label color="dark">
          <ion-icon name="pin-outline"></ion-icon>
          {{ combo.travelDest }}
        </ion-label>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        <ion-label color="dark">
          <ion-icon name="airplane-outline"></ion-icon>
          {{ from }} {{ combo.startDest }}
        </ion-label>
      </ion-col>
    </ion-row>

    <ion-row>
      <ion-col size=6 *ngIf="combo.wifi">
        <ion-icon name="wifi-outline"></ion-icon>
        <ion-label> {{ freeWifiTitle }} </ion-label>
      </ion-col>
      <ion-col size=6 *ngIf="combo.hotelRating && combo.hotelRat-
ing > 0">
        <ion-icon name="bed-outline"></ion-icon>
        <ion-label> {{ hotelTitle }} {{ combo.hotelRat-
ing }} {{ starTitle }}</ion-label>
      </ion-col>
    </ion-row>

    <ion-row>
      <ion-col size=6 *ngIf="combo.breakfast">
        <ion-icon name="fast-food-outline"></ion-icon>
        <ion-label> {{ breakfastTitle }} </ion-label>
      </ion-col>
      <ion-col size=6 *ngIf="combo.taxi">
        <ion-icon name="bus-outline"></ion-icon>
        <ion-label> {{ taxiTitle }} </ion-label>
      </ion-col>
    </ion-row>

    <ion-row>
      <ion-col>
        <ion-label color="dark"> {{ introTitle }} </ion-label>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        <ion-text color="dark">
          <h3 class="combo-description">
            {{ combo.description }}
          </h3>
        </ion-text>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        <ion-label color="dark"> {{ offerTitle }} </ion-label>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
  <ion-toolbar>
    <ion-grid>
      <ion-row>
        <ion-col size="6" class="ion-text-center">
          <ion-label color="primary">{{ combo.price }}đ</ion-label>
        </ion-col>
        <ion-col size="6">
          <ion-button color="primary" ex-
pand="block" (click)="onBookCombo()">
            {{ bookButtonTitle }}
          </ion-button>
        </ion-col>
      </ion-row>
    </ion-grid>
  </ion-toolbar>
```

Code Snippet 6. Template to display details of a specific combo

6.2 Travel Articles

The travel articles are some of popular articles about travelling. They are declared with the Article class, which contains information about article title, article image and article description.

```
export class Article {

    constructor(
      public id: string,
      public title: string,
      public imgUrl: string,
      public content: string,
    ) { }
  }
```

Code Snippet 7. Article class declaration

The list of articles then can be loaded in the article service from Firebase to a BehaviorSubject class:

```
  private _articles = new BehaviorSubject<Article[]>([]);

  constructor(private db: AngularFireDatabase) { }

  get articles() {
    this.db.list<Article>(`articles`).snapshotChanges().sub-
scribe(res => {
      const articleList = [];
      res.forEach(item => {
        const article = item.payload.toJSON();
        const id = item.key;
        articleList.push({...article, id});
      });
      this._articles.next(articleList);
    });
    return this._articles.asObservable();
  }
```

Code Snippet 8. Function to load articles from Firebase

The list of articles can be presented with the template:

```
        <ion-slides [options]="slideOpts">
          <ion-slide *ngFor="let article of loadedArticles">
            <ion-card
              [routerLink]="[
                '/',
                'tabs',
                'tab1',
                'article-detail',
                article.id
              ]">
              <img [src]="article.imgUrl">
              <ion-card-header text-wrap>
                <ion-card-title class="article-title"> {{ arti-
cle.title }}</ion-card-title>
              </ion-card-header>
            </ion-card>
          </ion-slide>
        </ion-slides>
```

Code Snippet 9. Template to display a list of articles

When the user clicks on an article, the application should also display the details of that article:

```
<ion-header>
  <ion-toolbar class="ion-color" style.background="'url(' + arti-
cle.imgUrl + ')'" >
    <ion-buttons>
      <ion-back-button icon="arrow-back-outline" slot="start"></ion-
back-button>
    </ion-buttons>
      <ion-img [src]="article.imgUrl"></ion-img>
      <ion-label class="ion-text-center"> {{ article.title }}</ion-
label>
  </ion-toolbar>

</ion-header>

<ion-content>
  <ion-grid>
    <ion-row>
      <ion-col>
        <ion-text> {{ article.content }}</ion-text>
      </ion-col>
    </ion-row>
  </ion-grid>
```

```
</ion-content>
```

Code Snippet 10. Template to display detail of an article

6.3 Favorite Travel Combos

The user can save their favorite travel combos to access it quickly later. The stored data is declared in FavCombo class, which saves the ID of the travel combo. Due to recommendation of Firebase about data structure, the title and images URL are also saved to the database. /12/

```
export class FavCombo {

    constructor(
      public id: string,
      public comboId: string,
      public title: string,
      public imgUrl: string,
    ) { }
  }
```

Code Snippet 11. FavCombo class declaration

The FavCombosService is implemented to load favorite combos, add new favorite combos, and remove favorite combos in Firebase Realtime Database.

```
import { Injectable } from '@angular/core';
import { FavCombo } from './favCombo.model';
import { BehaviorSubject } from 'rxjs';
import { AuthService } from '../auth/auth.service';
import { AngularFireDatabase, AngularFireList, AngularFireOb-
ject  } from '@angular/fire/database';
import { take, switchMap, map, tap } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class FavCombosService {
  favComboListRef: AngularFireList<any>;
  favComboRef: AngularFireObject<any>;
  private _favCombos = new BehaviorSubject<FavCombo[]>([]);

  constructor(
    private authService: AuthService,
    private db: AngularFireDatabase,
  ) { }

  get favCombos() {
    return this._favCombos.asObservable();
  }


  fetchFavCombos() {
    return this.authService.userId.pipe(
      switchMap(uid => {
        if (!uid) {
          throw new Error('User not found!');
        }
        return this.db.list<FavCombo>(`users/${uid}/favcom-
bos`).snapshotChanges();
      }),
      map(res => {
        const favComboList = [];
        res.forEach(item => {
          const combo = item.payload.toJSON();
          const id = item.key;
          favComboList.push({...combo, id});
        });
        return favComboList;
      }),
      tap(favComboList => {
        this._favCombos.next(favComboList);
      })
    );
  }

  public addFavCombo(comboId: string, comboTitle: string, comboIm-
gUrl: string) {
    let newFavCombo: FavCombo;
    let generatedId: string;
    return this.authService.userId.pipe(
      take(1),
      switchMap(uid => {
        console.log(uid);
        if (!uid) {
          throw new Error('No user id found!');
        }
        newFavCombo = new FavCombo(Math.random().toString(), com-
boId, comboTitle, comboImgUrl);
        console.log(newFavCombo);
        this.favComboListRef = this.db.list(`users/${uid}/favcom-
bos`);
        return this.favComboListRef.push({...new-
FavCombo, id: null});
      }),
      switchMap(resData => {
        console.log(resData);
        generatedId = resData.key;
        return this.favCombos;
      }),
      take(1),
      tap(favCombos => {
        newFavCombo.id = generatedId;
        this._favCombos.next(favCombos.concat(newFavCombo));
      })
    );
  }

  public removeFavCombo(favId: string) {
    this.authService.userId.pipe(take(1)).subscribe(uid => {
      if (!uid) {
        throw new Error('No user id found!');
      }
      this.favComboRef = this.db.object(`users/${uid}/favcom-
bos/${favId}`);
      this.favComboRef.remove();
    });
  }
}
```

Code Snippet 12. Favourite combo service

These above functions are applied to the Angular template to represent all favorite combos for the user. If there are no favorite combo, it should notice the users and show them the link to navigate back to the home page. Otherwise, it should display all favorite combos and the heart button to remove a specific combo from this list.

```
<ion-content>
  <ion-grid>
    <div *ngIf="!loadedFavCombos || loadedFavCombos.length <= 0">
      <ion-row>
        <ion-col size="6" offset="3">
          <ion-label>{{ noFavNotice }}</ion-label>
        </ion-col>
      </ion-row>
      <ion-row>
        <ion-col size="10" offset="1">
          <ion-button
            expand="block"
            color="primary"
            [routerLink]="[
              '/',
              'tabs',
              'tab1'
            ]"
          >
            {{ discover }}
          </ion-button>
        </ion-col>
      </ion-row>
    </div>
    <div *ngIf="loadedFavCombos && loadedFavCombos.length > 0">
      <ion-row *ngFor="let combo of dummyArray; index as i">
        <ion-col size=4>
          <ion-card>
            <img [src]="loadedFavCombos[i*2].imgUrl">
            <ion-button
              fill="clear"
              color="light"
              class="fav-button"
              (click)="onRemoveFavCombo(loadedFavCombos[i*2].id)"
            >
              <ion-icon name="heart" color="danger"></ion-icon>
            </ion-button>
            <ion-card-header
              [routerLink]="[
                '/',
                'tabs',
                'tab1',
                'combo-detail',
                loadedFavCombos[i*2].comboId
              ]"
            >
              <ion-card-title> {{ loadedFavCombos[i*2].ti-
tle }}</ion-card-title>
            </ion-card-header>
          </ion-card>
        </ion-col>
        <ion-col size=4 offset=2 *ngIf="loadedFavCombos[i*2+1]">
          <ion-card>
            <img [src]="loadedFavCombos[i*2+1].imgUrl">
            <ion-button
              fill="clear"
              color="light"
              class="fav-button"
              (click)="onRemoveFavCombo(loadedFavCombos[i*2+1].id)"
            >
              <ion-icon name="heart" color="danger"></ion-icon>
            </ion-button>
            <ion-card-header
              [routerLink]="[
                '/',
                'tabs',
                'tab1',
                'combo-detail',
                loadedFavCombos[i*2+1].comboId
              ]"
            >
              <ion-card-title> {{ loadedFavCombos[i*2+1].ti-
tle }}</ion-card-title>
            </ion-card-header>
          </ion-card>
        </ion-col>
      </ion-row>
    </div>
  </ion-grid>
</ion-content>
```

Code Snippet 13. Template to display favourite combos

6.4 Ticket

Ticket contains information about the booked trip for a user including combo ID, customer name, phone number, email, and note. Due to recommendation of Firebase about data structure, the combo information, for example, combo title or image URLs is also stored with ticket in the database.

```
export class Ticket {
  constructor(
    public id: string,
    public comboId: string,
    public title: string,
    public imgUrl: string,
    public rate: number,
    public startDest: string,
    public name: string,
    public phoneNumber: string,
    public email: string,
    public note: string,
    public coupon: string,
    public startDate: number,
    public numTickets: number,
  ) {
  }
}
```

Code Snippet 14. Ticket class declaration

The ticket service uses this Ticket class to load all tickets for the user and adds a new ticket after the user has purchased a combo to Firebase Realtime Database.

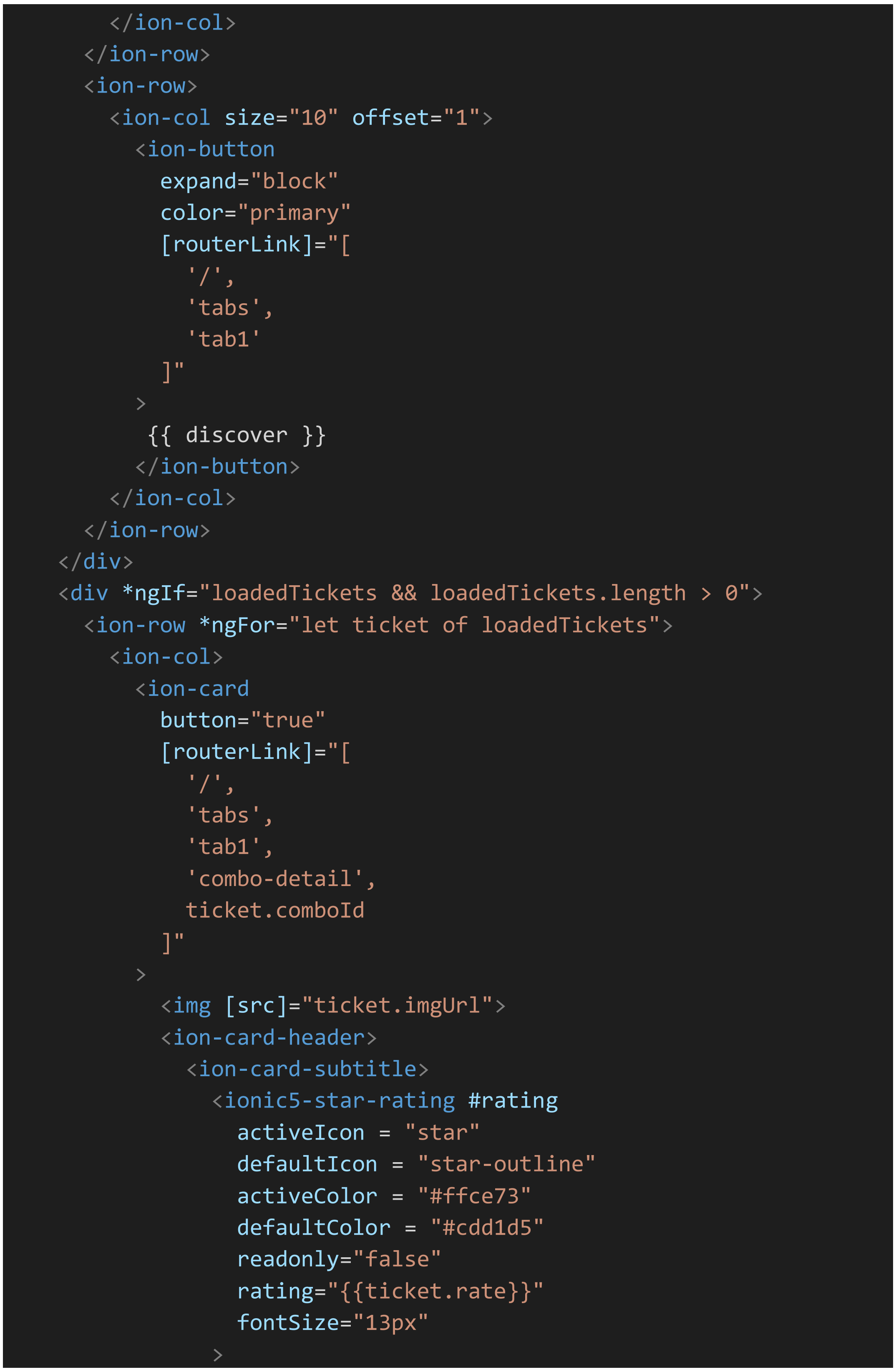
```
import { Injectable } from '@angular/core';
import { Ticket } from './ticket.model';
import { AuthService } from '../auth/auth.service';
import { AngularFireDatabase, AngularFireList } from '@angu-
lar/fire/database';
import * as moment from 'moment';
import { BehaviorSubject } from 'rxjs';
import { take, switchMap, map, tap } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
```

```
})
export class TicketService {
  private _ticket = new BehaviorSubject<Ticket[]>([]);
  ticketListRef: AngularFireList<any>;

  constructor(
    private authService: AuthService,
    private db: AngularFireDatabase,
  ) { }

  get tickets() {
    return this._ticket.asObservable();
  }

  fetchTickets() {
    return this.authService.userId.pipe(
      switchMap(uid => {
        if (!uid) {
          throw new Error('User not found!');
        }
        console.log(uid);
        this.ticketListRef = this.db.list<Ticket>(`us-
ers/${uid}/tickets`);
        return this.ticketListRef.snapshotChanges();
      }),
      map(res => {
        const ticketList = [];
        res.forEach(item => {
          const combo = item.payload.toJSON();
          const id = item.key;
          ticketList.push({...combo, id});
        });
        return ticketList;
      }),
      tap(ticketList => {
        console.log(ticketList);
        this._ticket.next(ticketList);
      })
    );
  }

  public addTicket(
    comboId: string,
    title: string,
    imgUrl: string,
    rate: number,
    startDest: string,
    name: string,
    phoneNumber: string,
    email: string,
    note: string,
    coupon: string,
    startDate: Date,
    numTickets: number,
  ) {
    const newTicket = new Ticket(
      Math.random().toString(),
      comboId,
      title,
      imgUrl,
      rate,
      startDest,
      name,
      phoneNumber,
      email,
      note,
      coupon,
      moment(startDate).unix(),
      numTickets
    );
    this.authService.userId.pipe(take(1)).subscribe(uid => {
      this.ticketListRef = this.db.list(`users/${uid}/tickets`);
      this.ticketListRef.push({...newTicket, id: null});
    });
    return this.ticketListRef.valueChanges();
  }
}
```

Code Snippet 15. Ticket service to load and add tickets

The template uses theses function to show all tickets of the user.

```
<ion-content padding>
  <ion-grid>
    <div *ngIf="!loadedTickets || loadedTickets.length <= 0">
      <ion-row>
        <ion-col size="6" offset="3">
          <ion-label>{{ noTicketNotice }}</ion-label>
```



```
    </ion-col>
  </ion-row>
  <ion-row>
    <ion-col size="10" offset="1">
      <ion-button
        expand="block"
        color="primary"
        [routerLink]="[
          '/',
          'tabs',
          'tab1'
        ]"
      >
       {{ discover }}
      </ion-button>
    </ion-col>
  </ion-row>
</div>
<div *ngIf="loadedTickets && loadedTickets.length > 0">
  <ion-row *ngFor="let ticket of loadedTickets">
    <ion-col>
      <ion-card
        button="true"
        [routerLink]="[
          '/',
          'tabs',
          'tab1',
          'combo-detail',
          ticket.comboId
        ]"
      >
        <img [src]="ticket.imgUrl">
        <ion-card-header>
          <ion-card-subtitle>
            <ionic5-star-rating #rating
              activeIcon = "star"
              defaultIcon = "star-outline"
              activeColor = "#ffce73"
              defaultColor = "#cdd1d5"
              readonly="false"
              rating="{{ticket.rate}}"
              fontSize="13px"
            >
```

```
                </ionic5-star-rating>
                {{ ticket.rate }} / 5
              </ion-card-subtitle>
              <ion-card-title> {{ ticket.title }}</ion-card-title>
              <ion-card-subtitle>
                <ion-icon name="time-outline" color="primary"></ion-
icon>
                {{ ticket.startDate * 1000 | date:'dd/MM/yyyy' }}
              </ion-card-subtitle>
              <ion-card-subtitle>
                <ion-icon name="airplane-outline" color="pri-
mary"></ion-icon>
                {{ from }} {{ ticket.startDest }}
              </ion-card-subtitle>
              <ion-card-subtitle>
                <ion-icon name="pricetags-outline" color="pri-
mary"></ion-icon>
                {{ ticket.numTickets }} {{ ticketUnit }}
              </ion-card-subtitle>
              <ion-button
                expand="full"
                shape="round"
                class="ion-text-center">
                {{ appliedMessage }}
              </ion-button>
            </ion-card-header>
          </ion-card>
        </ion-col>
      </ion-row>
    </div>
  </ion-grid>
</ion-content>
```

Code Snippet 16. Template for tickets page

The adding ticket function is used after the user has booked a combo to add a new ticket to the ticket list.

6.5 Booking Form

The booking form appears when the user clicks on the book button on the combo detail page. There are three steps to book a travel combo: choosing a departure date, input personal information to book and payment.

6.5.1 Choosing a Departure Date

Although Ionic provides developers with ion-cal component as their default calendar component, in this project ion2-calendar package at https://www.npmjs.com/package/ion2-calendar will be used. Ion2-calendar has better support for UI implementation with a lot of customization for the application.

6.5.2 Input Customer Information

In this section, the application just displays a normal form for the user to input their personal information, for example name, phone number and email to book the travel combo.

6.6 Payment

In the application, PayPal is chosen as a payment method because it supports both pay with PayPal and pay with debit/credit card.

Actually, the native plugins for Ionic also support PayPal payment method at https://ionicframework.com/docs/native/paypal. However, the plugin is based on old API from PayPal. Ionic is also based on Angular architecture, therefore, Angular PayPal library can be applied in the application. The ngx-paypal library at https://www.npmjs.com/package/ngx-paypal seems to be the most suitable for the application. This library is compatible with all platform, while the native plugin is only compatible with mobile device. Besides, the ngx-paypal library is implemented using newest API and UI from PayPal, thereby bringing better user experience.

Firstly, a Client ID is required to use PayPal API for payment. By visiting PayPal developer website at https://developer.paypal.com/ a new application can be created. It will be created with a new Client ID.

Then, the package will be installed with npm:

npm install ngx-paypal –save

Then, NgxPayPalModule is imported to the payment page and the ngx-paypal component is configured according to the application requirement and the client ID.

```
    this.payPalConfig = {
      currency: 'USD',
      clientId: sandBoxClientId,
      // tslint:disable-next-line: no-angle-bracket-type-assertion
      createOrderOnClient: (data) => <ICreateOrderRequest> {
        intent: 'CAPTURE',
        purchase_units: [
          {
            amount: {
              currency_code: 'USD',
              value: totalPrice.toFixed(2),
              breakdown: {
                item_total: {
                  currency_code: 'USD',
                  value: totalPrice.toFixed(2)
                },
                discount: {
                  currency_code: 'USD',
                  value: discount.toFixed(2)
                }
              }
            },
            items: [
              {
                name: 'Travel combo',
                quantity: this.quantity.toString(),
                category: 'DIGITAL_GOODS',
                unit_amount: {
                  currency_code: 'USD',
                  value: itemPrice.toFixed(2),
                },
              }
            ]
          }
        ]
      },
      advanced: {
        commit: 'true'
      },
      style: {
        label: 'paypal',
        layout: 'vertical'
      },
      onApprove: (data, actions) => {
        actions.order.get().then((details: any) => {
          console.log('onApprove - you can get full order de-
tails inside onApprove: ', details);
        });
      },
      onClientAuthorization: () => {
        this.onBookCombo();
      },
      onCancel: () => {
        this.presentAlert(this.failMessage, this.errorMessage);
      },
      onError: err => {
        this.presentAlert(this.failMessage, err);
      },
    };
```

Code Snippet 17. PayPal payment config

The configuration is about order details such as total price, unit price, quantity. It also allows developers to control what will happen when the order is approved, authorized, or canceled. Compared to using JavaScript API from PayPal, this package helps developers to save a lot of time as well as write a well readable code.

Then, this configuration is applied to the ngx-paypal component in the template. The button will be displayed as follows:

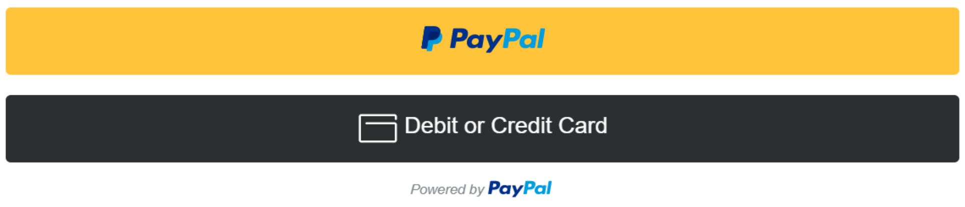
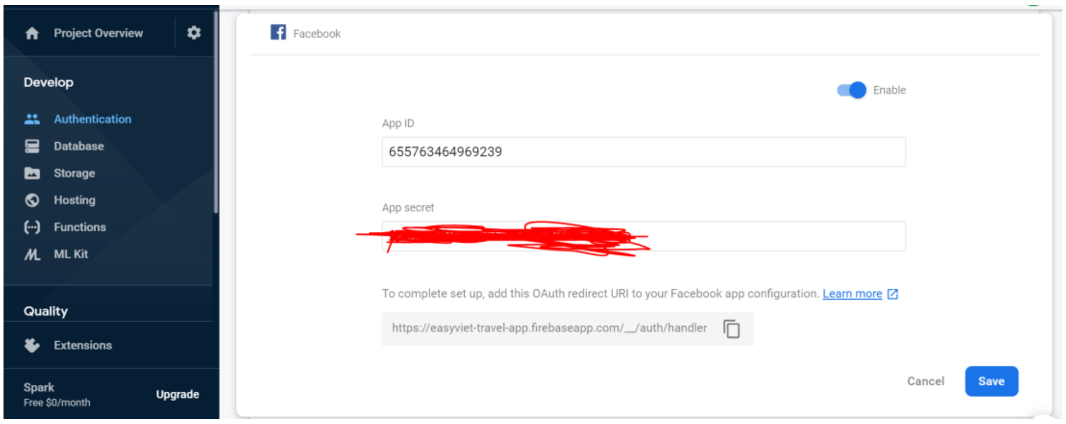


Figure 16. Pay with PayPal buttons

6.7 Facebook Login

Due to most people in Vietnam having a Facebook account, creating a new user account, and logging in with Facebook will be the most suitable method for the application.

First, Firebase needs to know that the authentication method is Facebook Login. Move to the Authentication tabs in the Firebase console, choose sign-in method and enable login with Facebook.



Enable

App ID
655763464969239

App secret
[redacted]

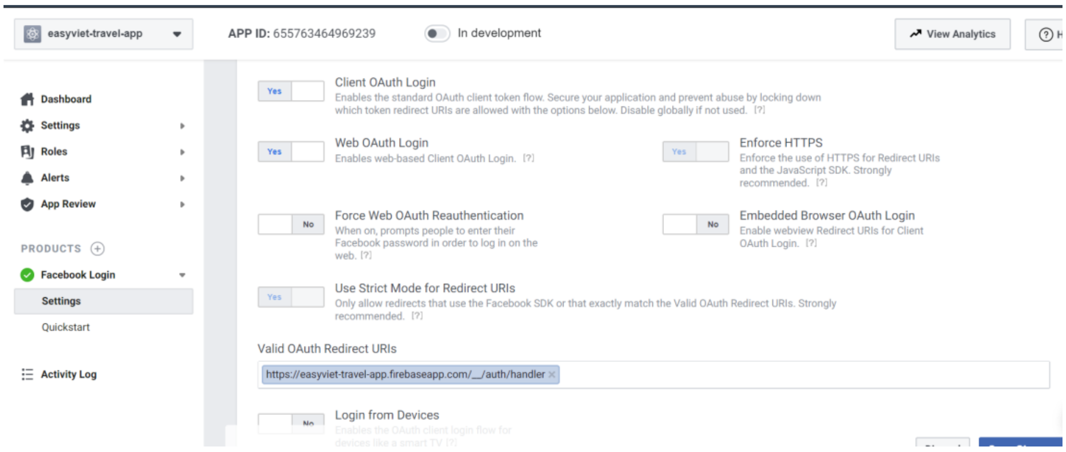
To complete set up, add this OAuth redirect URI to your Facebook app configuration. Learn more

https://easyviet-travel-app.firebaseapp.com/__/auth/handler

Figure 17. Facebook login with Firebase

To get the app ID and app secret, creating a new Facebook app is required. It also allows developers to access the user Facebook account. A Facebook developer account at https://developers.facebook.com/ is required for doing that. Then, use that Facebook developer account to create a new project. Facebook will provide developers a new app ID and app secret with that project. Then, the Facebook Login configurations have to be completed for both iOS and Android.

After filling all setting for the application, move to the setting page for Facebook Login product and add OAuth redirect URI of Firebase to the configuration.



Yes — Client OAuth Login
Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]

Yes — Web OAuth Login
Enables web-based Client OAuth Login. [?]

Yes — Enforce HTTPS
Enforce the use of HTTPS for Redirect URIs and the JavaScript SDK. Strongly recommended. [?]

No — Force Web OAuth Reauthentication
When on, prompts people to enter their Facebook password in order to log in on the web. [?]

No — Embedded Browser OAuth Login
Enable webview Redirect URIs for Client OAuth Login. [?]

Yes — Use Strict Mode for Redirect URIs
Only allow redirects that use the Facebook SDK or that exactly match the Valid OAuth Redirect URIs. Strongly recommended. [?]

Valid OAuth Redirect URIs
https://easyviet-travel-app.firebaseapp.com/__/auth/handler

No — Login from Devices
Enables the OAuth client login flow for devices like a smart TV [?]

Figure 18. Facebook login OAuth Redirect URIs

6.8 Authentication and Security in the Application

For the best user experience, the application should login the user with the native Facebook application on their mobile devices. The Facebook connect plugin at https://ionicframework.com/docs/native/facebook will be used to access that native Facebook application.

In the source code folder, type:

ionic cordova plugin add cordova-plugin-facebook4 --variable APP_ID="APP_ID" --variable APP_NAME="myApplication"

to install the Facebook connect plugin to the application.

Then, add Facebook to the providers in the app module declaration file. Now, the plugin is ready to use. Firstly, the application needs to authenticate the user with Facebook, and send the received credential token to the Firebase authentication to authenticate the user with Firebase.

```
  async nativeFacebookAuth() {
    return this.fb.login(['email']).then((response: FacebookLoginRe-
sponse) => {
      if (response.authResponse) {
          // Build Firebase credential with the Facebook auth token.
          const credential = firebase.auth.FacebookAuthProvider.cre-
dential(
            response.authResponse.accessToken
          );
          // Sign in with the credential from the Facebook user.
          firebase
            .auth()
            .signInWithCredential(credential)
            .then((result) => {
              this.setCurrentUser(
                result.user.uid,
                result.user.email,
                result.user.displayName,
                result.user.photoURL,
              );
            })
            .catch(error => {
              this.presentAlert(this.errorTitle, this.errorMessage);
            });

      } else {
        // User is signed-out of Facebook.
        firebase.auth().signOut();
      }
    }).catch(err => {
      this.presentAlert(this.errorTitle, this.errorMessage);
    });
  }
```

Code Snippet 18. Login with Facebook and Firebase implementation

To make sure the user cannot access application without login, Angular provides the Route Guards, which are the interface to decide whether the user can access a particular route. For this application, the authentication guard will check the user's data with the Firebase Authentication API. If the user has not logged in, the application will navigate the user to the authentication page. Otherwise, the application will set up him as the current user. This implementation also makes sure that the current user is not required to authenticate again after leaving the application, thereby provides a better user experience on a smartphone.

```
import { Injectable } from '@angular/core';
import { UrlTree, Router, CanActivate } from '@angular/router';
import { Observable, of } from 'rxjs';
import { AuthService } from './auth.service';
import '@firebase/auth';
import firebase from '@firebase/app';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  constructor(private authService: AuthService, pri-
vate router: Router) {}

  canActivate(): Observable<boolean | UrlTree> | Promise<bool-
ean | UrlTree> | boolean | UrlTree {
    return new Promise((resolve, reject) => {
      firebase.auth().onAuthStateChanged((user: firebase.User) => {
        if (user) {
          this.authService.setCurren-
tUser(user.uid, user.email, user.displayName, user.photoURL);
          resolve(true);
        } else {
          this.router.navigateByUrl('/auth');
          resolve(false);
        }
      });
    });
  }
}
```

Code Snippet 19. Authentication guard

6.9 User Account

The template to display current user account information:

```
<ion-header>

  <ion-toolbar>
    <ion-back-button icon="arrow-back-out-
line" slot="start" text=""></ion-back-button>
    <ion-title class="ion-text-center"> {{ info }} </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-list>
    <ion-item>
      <ion-avatar slot="start">
        <img [src]="currentUser.photoUrl">
      </ion-avatar>
      <ion-label>
        <h2><b> {{ currentUser.name }}</b></h2>
        <p>{{ currentUser.email }}</p>
      </ion-label>
    </ion-item>
    <ion-item>
      <ion-label>
        <ion-text color="primary">*</ion-text>
        <ion-text> {{ customerNameField }}</ion-text>
      </ion-label>
      <ion-input value={{currentUser.name}} [(ngModel)]="curren-
tUser.name" type="text" (ionBlur)="updateName()"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>
        <ion-text>+84</ion-text>
      </ion-label>
      <ion-input value={{currentUser.phoneNumber}} type="tel" input-
mode="numeric" [(ngModel)]="currentUser.phoneNumber" (ionBlur)="up-
datePhoneNumber()"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>
        <ion-text color="primary">*</ion-text>
        <ion-text>Email</ion-text>
      </ion-label>
      <ion-label text-wrap>
        {{ currentUser.email }}
      </ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```

Code Snippet 20. Template to display user account information

6.10 Search for Travel Combos

The application also allows users to search for travel combo by their travel destination. When the user clicks on the search bar, the application will navigate to the search page, then allow the user to input data in there.

After receiving search input from the user, the search page will call combo services to handle the input.

```
  findCombo(travelDest: string) {
    return this.combos.pipe(
      take(1),
      map(combos => {
        return combos.filter(
          c => this.changeAlias(c.travelDest).toLowerCase().in-
cludes(this.changeAlias(travelDest).toLowerCase())
        );
      })
    );
  }


  private changeAlias(input: string) {
    // Convert special Vietnamese character to English character
    let str = input;
    str = str.toLowerCase();

    str = str.replace(/à|á|ạ|ả|ã|â|ầ|ấ|ậ|ẩ|ẫ|ă|ằ|ắ|ặ|ẳ|ẵ/g, 'a');
    str = str.replace(/è|é|ẹ|ẻ|ẽ|ê|ề|ế|ệ|ể|ễ/g, 'e');
    str = str.replace(/ì|í|ị|ỉ|ĩ/g, 'i');
    str = str.replace(/ò|ó|ọ|ỏ|õ|ô|ồ|ố|ộ|ổ|ỗ|ơ|ờ|ớ|ợ|ở|ỡ/g, 'o');
    str = str.replace(/ù|ú|ụ|ủ|ũ|ư|ừ|ứ|ự|ử|ữ/g, 'u');
    str = str.replace(/ỳ|ý|ỵ|ỷ|ỹ/g, 'y');
    str = str.replace(/đ/g, 'd');
    str = str.re-
place(/!|@|%|\^|\*|\(|\)|\+|\=|\<|\>|\?|\/|,|\.|\:|\;|\'|\"|\&|\#|\[
|\]|~|\$|_|`|-|{|}|\||\\/g, ' ');
    str = str.replace(/ + /g, ' ');

    str = str.trim();
    return str;
  }
```

Code Snippet 21. Find combos by travel destination

Due to the characteristic of Vietnamese, the application firstly needs to normalize it to English and then find the suitable travel combo for the user. This problem can be easily solved with regex. After that, it will return the matched combos to the search page.

```
  ngOnInit() {
    this.loadingCtrl.create({
      message: this.waitMessage
    })
    .then(loadingEl => {
      loadingEl.present();
      this.combosSub = this.comboService.combos.subscribe(com-
bos => {
        this.allCombos = combos;
      });
      this.loadedCombos = this.allCombos;
      loadingEl.dismiss();
    });
  }

  search() {
    if (!this.searchInput || this.searchInput === '') {
      this.loadedCombos = this.allCombos;
    } else {
      this.comboService.findCombo(this.searchInput).subscribe(com-
bos => {
        this.loadedCombos = combos;
      });
    }
  }
```

Code Snippet 22. Search page implementation

Then, display them with Angular template:

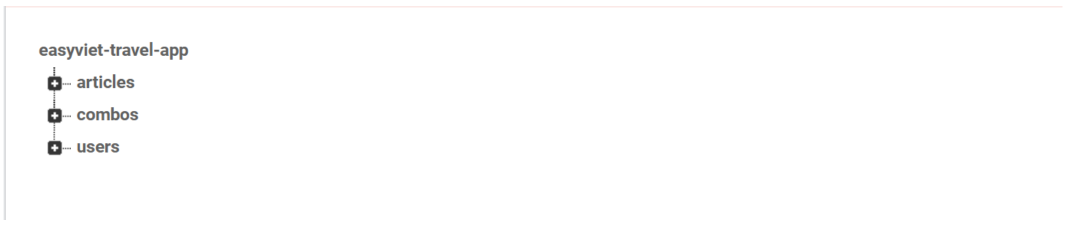
```
<ion-header padding [translucent]="true">
  <ion-toolbar>
    <ion-buttons>
      <ion-button fill=clear color=dark (click)="closeModal()">
        <ion-icon name="arrow-back-outline"></ion-icon>
      </ion-button>
      <ion-input
        [(ngModel)]="searchInput"
        debounce=1000
        placeholder="{{ searchMessage }}"
        (ionChange)=search()
      >
      </ion-input>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-list>
    <ion-item-sliding *ngFor="let combo of loadedCombos">
      <ion-item>
        <ion-card>
          <img [src]="combo.imgUrl">
          <ion-button
            fill="clear"
            color="light"
            class="fav-button"
            (click)="onAddingFavCombo(combo.id, combo.ti-
tle, combo.imgUrl)"
          >
            <ion-icon name="heart-outline"></ion-icon>
          </ion-button >

          <ion-card-header>
            <ion-card-subtitle>
              <ionic5-star-rating #rating
                activeIcon = "star"
                defaultIcon = "star-outline"
                activeColor = "#ffce73"
                defaultColor = "#cdd1d5"
                readonly="false"
                rating="{{combo.rate}}"
                fontSize="13px"
              >
              </ionic5-star-rating>
              {{ combo.rate }} / 5
            </ion-card-subtitle>
            <ion-card-title class="combo-title">
```

```
              {{ combo.title }}
              <ion-grid>
                <ion-row>
                  <ion-item *ngIf="combo.breakfast" class="border">
                    {{ breakfastTitle }}
                  </ion-item>
                  <ion-item *ngIf="combo.hotelRating" class="bor-
der">
                    {{ hotelTitle }} {{ combo.hotelRat-
ing }} {{ starTitle }}
                  </ion-item>
                  <ion-item *ngIf="combo.taxi" class="border">
                    {{ taxiTitle }}
                  </ion-item>
                  <ion-item  *ngIf="combo.wifi" class="border">
                    {{ freeWifiTitle }}
                  </ion-item>
                </ion-row>
                <ion-row>
                  <ion-col size=6>
                    <ion-label color="primary" style="padding-
top: 12px;">
                      <h3>{{ combo.price }}đ/{{personTitle}}</h3>
                    </ion-label>
                  </ion-col>
                  <ion-col size=6>
                    <ion-button class="book-button" color="pri-
mary" expand="block" (click)="onBookingCombo(combo.id)">
                      {{ bookButtonTitle }}
                    </ion-button>
                  </ion-col>
                </ion-row>
              </ion-grid>
            </ion-card-title>
          </ion-card-header>
        </ion-card>
      </ion-item>
    </ion-item-sliding>
  </ion-list>
</ion-content>
```

Code Snippet 23. Search page template

6.11 Firebase Database

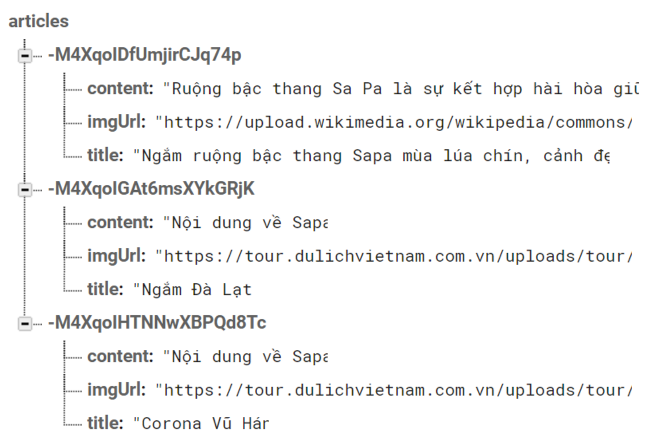
The Firebase Realtime Database with three main keys to store data of this application:



```
easyviet-travel-app
├ articles
├ combos
└ users
```

Figure 19. Firebase Realtime Database main keys

6.11.1 Articles database



```
articles
├ -M4XqoIDfUmjirCJq74p
│   ├ content: "Ruộng bậc thang Sa Pa là sự kết hợp hài hòa giữ
│   ├ imgUrl: "https://upload.wikimedia.org/wikipedia/commons/
│   └ title: "Ngắm ruộng bậc thang Sapa mùa lúa chín, cảnh đẹ
├ -M4XqoIGAt6msXYkGRjK
│   ├ content: "Nội dung về Sapa
│   ├ imgUrl: "https://tour.dulichvietnam.com.vn/uploads/tour/
│   └ title: "Ngắm Đà Lạt
└ -M4XqoIHTNNwXBPQd8Tc
    ├ content: "Nội dung về Sapa
    ├ imgUrl: "https://tour.dulichvietnam.com.vn/uploads/tour/
    └ title: "Corona Vũ Hán
```

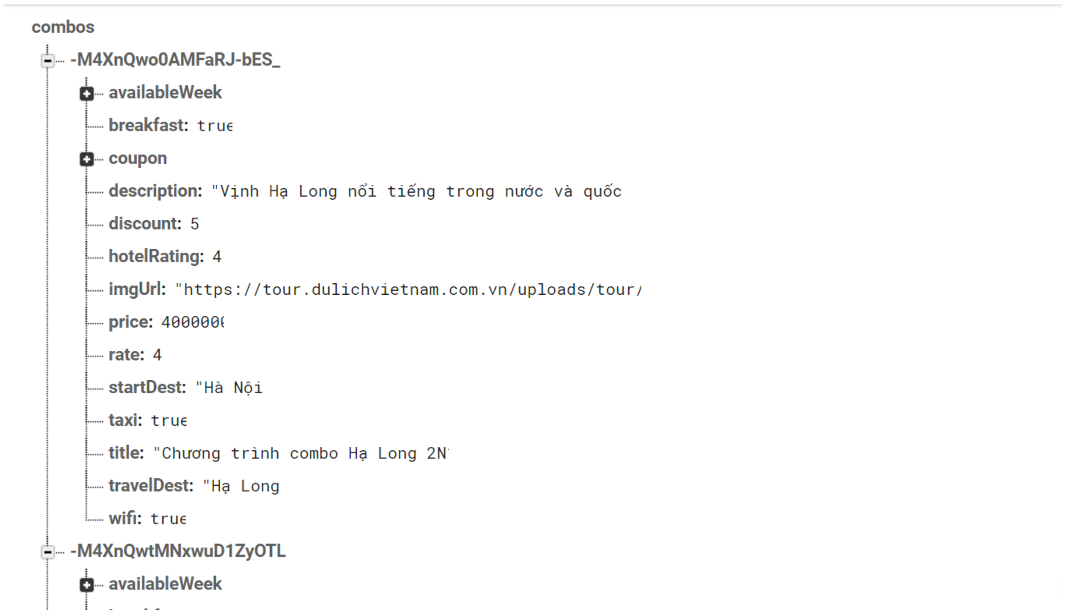
Figure 20. Articles data in Firebase

The articles key stores all data about travel articles in the application. Each article entry has a unique ID as key, and the value includes three fields: title, content, and image URL.

```
"articles" : {
    "id" : {
      "content" : string
      "imgUrl" : string
      "title": string
    }
}
```

Code Snippet 24. Article data in JSON tree

6.11.2 Combos database



```
combos
-M4XnQwo0AMFaRJ-bES_
    availableWeek
    breakfast: true
    coupon
    description: "Vịnh Hạ Long nổi tiếng trong nước và quốc
    discount: 5
    hotelRating: 4
    imgUrl: "https://tour.dulichvietnam.com.vn/uploads/tour/
    price: 400000(
    rate: 4
    startDest: "Hà Nội
    taxi: true
    title: "Chương trinh combo Hạ Long 2N
    travelDest: "Hạ Long
    wifi: true
-M4XnQwtMNxwuD1ZyOTL
    availableWeek
```

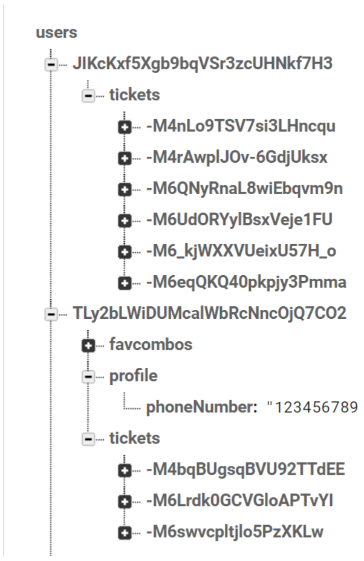
Figure 21. Combos data in Firebase

The articles key stores all data about travel combos offered in the application. Each combo entry has a unique ID as key, and the value includes the necessary information about the combo. In addition, due to the requirement about available day of travel combos, their value needs to be stored by days in week. These values can also be updated later in order to add more functionality to the application.

```
  "combos" : {
    "id" : {
      "availableWeek" : {
        "fri" : boolean,
        "mon" : boolean,
        "sat" : boolean,
        "sun" : boolean,
        "thu" : boolean,
        "tue" : boolean,
        "wed" : boolean
      },
      "breakfast" : boolean,
      "coupon" : {
        "test" : number
      },
      "description" : string,
      "discount" : number,
      "hotelRating" : number,
      "imgUrl" : string,
      "price" : number,
      "rate" : number,
      "startDest" : string,
      "taxi" : boolean,
      "title" : string,
      "travelDest" : string,
      "wifi" : boolean
    },
```

Code Snippet 25. Combo data in JSON tree

6.11.3 Users database



```
users
  JIKcKxf5Xgb9bqVSr3zcUHNkf7H3
    tickets
      -M4nLo9TSV7si3LHncqu
      -M4rAwplJOv-6GdjUksx
      -M6QNyRnaL8wiEbqvm9n
      -M6UdORYylBsxVeje1FU
      -M6_kjWXXVUeixU57H_o
      -M6eqQKQ40pkpjy3Pmma
  TLy2bLWiDUMcalWbRcNnc0jQ7CO2
    favcombos
    profile
      phoneNumber: "123456789
    tickets
      -M4bqBUgsqBVU92TTdEE
      -M6Lrdk0GCVGloAPTvYI
      -M6swvcpltjlo5PzXKLw
```

Figure 22. User data summary in Firebase

The articles key stores all data about user activities in the application. Each combo entry has a unique ID as key, and the value includes the information of the user, such as their favourite combos, their profiles, and their booking tickets. Firebase automatically encrypts data in transit using HTTPS and logically isolates user data.

```
    "id" : {
      "favcombos" : {
        "favComboid" : {
          "comboId" : string,
          "imgUrl" : string,
          "title" : string
        },
        "id" : {
            …
        },
      },
      "profile" : {
        "phoneNumber" : string
      },
      "tickets" : {
        "id" : {
          "comboId" : string,
          "coupon" : string,
          "email" : string,
          "imgUrl" : string,
          "name" : string,
          "note" : string,
          "numTickets" : number,
          "phoneNumber" : string,
          "rate" : number,
          "startDate" : number,
          "startDest" : string,
          "title" : string
        },
        "id" : {
          …
        }
      }
```

Code Snippet 26. User data in JSON tree

There is a two-way relationship between favourite combos and combos, as well as between tickets and combos. Some fields in the database, for example, combo title or combo image URL, are duplicated. However, in this design, the necessary data is still stored and can be fetched directly from database instead of using complicated query. The database is designed so as to reduce unnecessary redundancy of two-way relationships. It allows the user data to be fetched quickly and efficiency, even when the data scales much larger in the future. Therefore, it is a faster and better way than querying data in the database.

6.12 Language

In Android development, normally there is a strings.xml file, which allows developers to store constant string value. That file also helps the developer to translate their application into other languages easily. However, there is no such file for Ionic and Angular Framework. Fortunately, there is still an available package named *@ngx-translate/core* at https://www.npmjs.com/package/@ngx-translate/core#translateservice to work around language, although it is more complicated than using strings.xml file.

Firstly, ngx-translate package needs to be installed:

npm i @ngx-translate/core –save

And http-loader package is also required for ngx-translate to function properly:

npm install @ngx-translate/http-loader –save.

To use it, simply import these packages to the main module of application, and add the Translate Module into providers:

```
import { TranslateHttpLoader } from '@ngx-translate/http-loader';
import { HttpClientModule, HttpClient } from '@angular/common/http';
import { IonicStorageModule } from '@ionic/storage';

firebase.initializeApp(environment.firebaseConfig);

export function HttpLoaderFactory(http: HttpClient) {
  return new TranslateHttpLoader(http, './assets/i18n/', '.json');
}
```

```
@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [
    BrowserModule,
    IonicModule.forRoot(),
    AppRoutingModule,
    AngularFireModule.initializeApp(environment.firebaseConfig),
    AngularFireAuthModule,
    AngularFireDatabaseModule,
    AngularFireStorageModule,
    TranslateModule.forRoot({
      loader: {
        provide: TranslateLoader,
        useFactory: (HttpLoaderFactory),
        deps: [HttpClient]
      }
    }),
    HttpClientModule,
    IonicStorageModule.forRoot(),
  ],
  providers: [
    Facebook,
    PayPal,
    StatusBar,
    SplashScreen,
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Code Snippet 27. Add translate module into app modules file

Normally, this translate module will automatically import the language file, for example en.json or vi.json, from /i18n folder. However, there is no such file in Ionic. Therefore, a language folder has to be set up the at /assets/i18n folder.

Note that the same import is necessary for every page using translate module in the application. Now, the translate service is ready to be imported to logic files.

```
  ionViewDidEnter() {
    this._initialiseTranslation();
  }

  _initialiseTranslation(): void {
    this.translateService.onLangChange.subscribe((event: LangChang-
eEvent) => {
      this.articlePart = this.translateService.instant('ARTICLES');
      this.waitMessage = this.translateService.instant('WAIT');
      this.addFavMessage = this.translateService.instant('ADDFAV');
      this.searchMessage = this.translateService.instant('SEARCH');
    });
    this.translateService.get('ARTICLES').sub-
scribe((res: string) => {
      this.articlePart = res;
    });
    this.translateService.get('WAIT').subscribe((res: string) => {
      this.waitMessage = res;
    });
    this.translateService.get('ADDFAV').subscribe((res: string) => {
      this.addFavMessage = res;
    });
    this.translateService.get('SEARCH').subscribe((res: string) => {
      this.searchMessage = res;
    });
  }
```

Code Snippet 28. Implementation with translate service

The application must call the translate service before rendering page to load data from language files. It also needs to set a promise to update these loaded data in case of the user changing application language. These similar codes have to be implemented for every page using the translate service in the application.

In comparison to using string.xml files for storing data in normal Android development, this implementation method is much more complicated and results in writing a lot of duplicated code. Still, it is the best choice for Angular and Ionic developer right now.

6.13 Testing

Tests were executed to make sure that the application will work as intended.

The static testing tool used in the project is TSLint by Microsoft. It is already a built-in tool of VS Code and is always applied during implementation process.

The smoke tests were the main test performed. These tests made sure that the application's functions are working correctly. Below are some main tests performed:

Test ID	Test scenarios	Description	Test step	Expected result
1	Valid login credentials	Test the login functionality of the web application to ensure that a user can login with his Facebook account	1. Launch the application	Application should get launched
			2. Click on the Login with Facebook button	Facebook Application should be displayed and ask user for permissions
			3. Allow the app to access Facebook public profile	Login successfully. The application should open the home page.
2	Booking a combo tour	Test the booking function to make sure that a user can book a travel combo	1. Click on a specific combo	Application should show the combo details page

			2. Click on book button	Application should show the form to input information
			3. Click on next button	Application should display the total price
			4. Click on Pay with PayPal button	Application should open PayPal payment page
			5. Enter PayPal sandbox account and pay	Application should navigate to the success page
			6. Click on the complete button	Application should navigate to the ticket tab. The ticket should be available there.
3	Log out function	Check log out functionality	1. Move to account tab and click on log out	The user should be able to sign out.

				The application should navigate to authentication page

Table 3. Smoke tests

7 DEPLOYING AND PUBLISHING

7.1 Deploying for Debugging the Application

To run quickly for debugging an Ionic application in development, simply move to the source code folder, open the terminal and type:

ionic serve

The Ionic is based on the Angular architecture, so this command will open the Ionic application as a web application in the browser served by localhost. Developers can easily debug it with Chrome Developer Tools. The tools also provide mobile views, so it is very visible and easy to debug the application in Chrome.

For testing the application on a smartphone, Cordova is used. For example, with Android, firstly add Android to Cordova platform:

ionic cordova platform add android

Then, Ionic can build an application for Android:

ionic cordova build android

Running on an Android emulator opened:

ionic cordova emulate android

For running the application on a real Android device (connected with computer by USB connection):

ionic cordova run android --device

7.2 Deploying for Publishing the Application

To publish the application on Google Play/Play Store, a developer account is required in these stores. For example, with Google Play, the developer can register by paying 25 dollars and verifying ID with Google.

After registering and verification process, developers are able to start uploading the app to Google Play.

Now, the application bundle must comply with the Google requirements for publishing an Android application on Google Play. To do that, firstly build and release version of the Ionic application:

ionic cordova build android --prod --release

A released version in apk file will be built. That unsigned apk file is already enough to run in an Android device. However, publishing requires a signed app bundle instead of the built apk file.

Now, move to */platforms/android* folder and run following command:

gradlew bundle

An unsigned Android app bundle now is created in */platforms/android/app/build/outputs/bundle/release* folder. The default name for it is “app.aab”.

Then, generate the private key file to sign the bundle:

keytool -genkey -v -keystore my-key.keystore -alias alias-name -keyalg RSA -keysize 2048 -validity 10000

The program requires developers to fill in password and some information. After it's is done, a file called my-key.keystore is created in current directory. Use this file to sign the bundle:

jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore app.aab alias-name

Finally, the final app bundle is ready to release on Google Play.

Navigate to https://play.google.com/apps/publish/ and create a new application. After that, some required data will be in the left menu with a checkbox that needs to be completed:

- Store listing: Title, description, icon, screenshots, application type, categories, etc.
- Pricing and distribution: Information about app price (free/paid), country to distribute and legal laws.
- App content: Privacy policy, advertisements, app access and target audience and content (most questions are about whether this app is for children).
- Content rating: Type of app, categories of app, and purchasing in app.
- App releases: Releasing app in production, beta testing, alpha testing, or internal testing as well as setting about testers for the application.

At the moment, the application is released under internal testing. The process starts by specifying testers, then uploading the app bundle under internal testing form and finishing the review. The application will be set under pending publication. According to Google, after a few hours, the application will be published successfully and an opt-in link for testing will be available. For production, it required much longer time, approximately 7 days or longer for Google to review the application.

The application can be updated later by increasing the version value in *package.json* file and *config.xml* file and repeat the release process again.

8 CONCLUSIONS

As already stated, the idea for this application is to develop a mobile application which makes the travel booking process easier and mover convenient. Users can log in into the application with their Facebook account and control their account. They can read some travel articles, search for travel combos and add/remove travel combos in favorites. Most importantly, they are able to book travel combos directly in the application.

The application was developed with Ionic, Angular and TypeScript. The complexity of JavaScript libraries created many challenges during the development process, for example, the outdated Ionic PayPal plugins and the deprecated rating component. The Ionic ready-made component is also not always suitable with the application design and needs a lot of customized style fixing. Besides, while it is quicker than developing separate native applications for both Android and iOS, a hybrid framework such as Ionic requires a lot of work to develop and deploy successfully on mobile devices as well as Google Play/App Store.

Choosing to use Firebase actually is a good point in the development process. It has great services for both databases and authentication. It reduces the complexity, as clients are able to access Firebase without servers. It also has great documentation as well as good support libraries for Ionic and Angular. Therefore, Firebase is probably the best choice for mobile development.

As for improvement, some test cases should be done to make sure that the application will work correctly. The rating component can also be improved with a feedback system, so after finishing the trip, users can rate their travel combo.

In conclusion, the thesis objective was completed by creating a well-working mobile application for booking travel easier.

REFERENCES

/1/ Android. 2020. Accessed 2.5.2020. https://en.wikipedia.org/wiki/Android_(operating system).

/2/ Smartphone Market Share. 2020. Accessed 30.4.2020. https://www.idc.com/promo/smartphone-market-share/os

/3/ iOS. 2020. Accessed 30.4.2020. https://en.wikipedia.org/wiki/IOS

/4/ Ionic Framework - Ionic Documentation. 2020. Accessed 2.4.2020. https://ion-icframework.com/docs

/5/ Angular - Introduction to Angular Concept. 2020. Accessed 30.4.2020. https://angu-lar.io/guide/architecture

/6/ Typescript. 2020. Accessed 20.4.2020. https://en.wikipedia.org/wiki/TypeScript

/7/ Firebase. 2020. Accessed 13.4.2020. https://en.wikipedia.org/wiki/Firebase

/8/ Firebase Realtime Database. 2020. Accessed 2.5.2020. https://fire-base.google.com/docs/database

/9/ Firebase Authentication. 2020. Accessed 2.5.2020. https://fire-base.google.com/docs/auth

/10/ Overview - Facebook Login. 2020. Accessed 2.5.2020. https://developers.face-book.com/docs/facebook-login/overview

/11/ Architecture overview of Cordova platform - Apache Cordova. 2020. Accessed 2.5.2020. https://cordova.apache.org/docs/en/latest/guide/overview/index.html

/12/ Structure Your Database | Firebase Realtime Database. 2020. Accessed 2.5.2020. https://firebase.google.com/docs/database/admin/structure-data