Bilkent University

Department of Computer Engineering

Senior Design Project

Low-Level Design Report

Project Name: Nomad

Group Members: Gülnihal Muslu
Berk Ataç
Can Ozan Kaş
Ali Kemal Özkan
Tarık Emin Kaplan

Supervisor: Uğur Güdükbay

Jury Members: H. Altay Güvenir, Özcan Öztürk

Innovation Expert: Veysi İşler

Low-Level Design Report
February 17, 2020
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

Nowadays, travelling is an easy and usual activity for people[1]. Everybody wants to explore places that they have never been. However, there are a lot of buildings and places or activities to do in a country/city. Therefore, people have to choose the best of them according to their interests because they usually have a limited time to see a city. Deciding which place is worth seeing or which food is worth eating can be tricky in most times due to people usually preferring to explore cities that they have never seen before. There are some applications and websites for that, but they have never been enough for people that are such travel lovers.

To solve that problem, we thought of a project -which name is Nomad- where people can give trip advice to other people. Nomad means a member of a group of people who move from one place to another instead of living in the same place all the time. Our project idea comes from there. In this project, a person can create a route in a city or country and s/he can give advice on that trip. Users will have a chance to select a route from various list of routes that are created by other users and they can follow that path step by step. Since routes can be seen before and they can be scored, interpreted; people can choose their route easily and more close to their dream trip. With this project travels will become easier and more fun.

This report contains analysis of our project with three subtitles: current system, proposed system and other analysis elements. All system models and user interface included.

1.1. Object Design Trade-offs

1.1.1. Memory vs Time

Nomad stores a mass amount of data such as ratings, routes, user infos, checkpoint and personal information for each user. The processing time of the system increases and response time between operation can slowdown because more data cause more delay in the system.

1.1.2. Scalability vs Performance

Performance is one of the main priorities that need to be provided to our system. However, we also want multiple users to use Nomad at the same time. Therefore we also need to build a scalable application in which it can handle several requests simultaneously. Moreover, if the system is designed in a way to serve multiple users at the same time meaning that being scalable, maintaining the performance as it was serving for only one user is challenging.

1.1.3. Functionality vs Usability

Our application would favor usability over functionality. In order to ensure desirable user experience, UI is designed to stick to core functionalities of the application and avoid unnecessary features like direct messaging or posting anything on profile pages.

1.1.4. Security vs Cost

Nomad will collect lots of data from the users. Keeping the data secure will be more important the cost. That is why we want to build a secure application and promise users that privacy is the one of the most important thing for this app.

1.2. Interface Documentation Guidelines

In this report, all the class names are named ‘ClassName’ format. All class names are singular. The variable and method names are named following the same convention: ‘variableName’ and ‘methodName()’. An example is given in the table below.

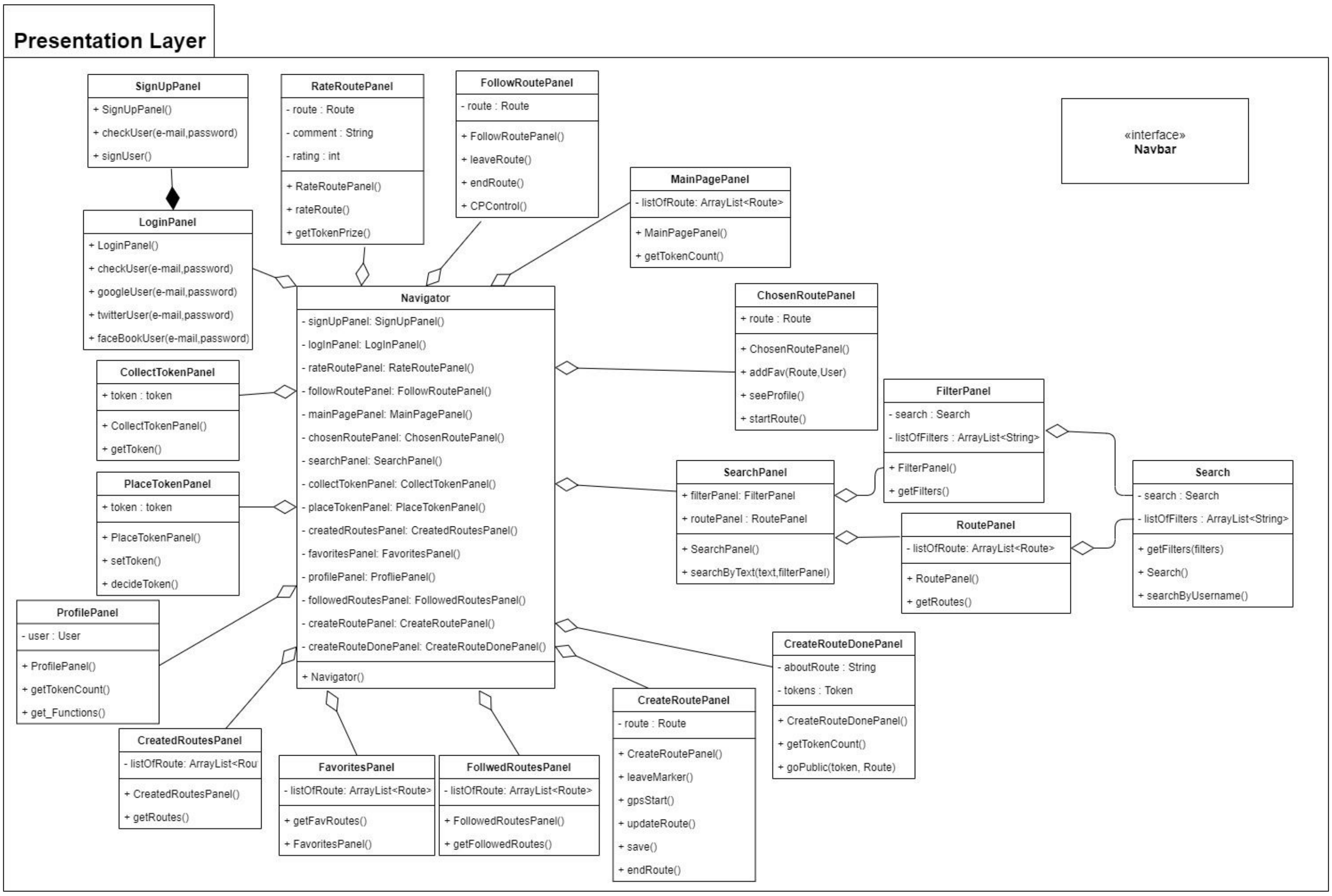
Class Name
Description of Class
Attributes
Name of Attribute: Type of Attribute
Methods
MethodName(ParameterOfMethod): description of method

1.3. Engineering Standards

In the description of class diagrams, interfaces, use cases, scenarios, subsystem decompositions and description of hardware-software components, UML design principles are used. In addition, we have referred to The Institute of Electrical and Electronics Engineers (IEEE) guides to cite the references.

2. Packages

2.1. View



This package consists of UI classes, which are responsible for rendering UI content of Nomad. Each panel above represents one page and side panels or popup windows in our application. All panels above are controlled by Navigator.

Navigator: This is the manager component that shifts between the pages according to user actions. Also communicates with Controller layer.

SignUpPanel: This class is for the panel where user signs up to our system.

LoginPanel: This class is for the panel where user logs in to our system. Either by our system or with google, facebook, twitter accounts.

MainPagePanel:This class is for the panel where user what to do. They can either create or follow a route or choose one of the Top 10 routes listed here, they can see their token count.

RateRoutePanel: This class is for the panel where user rates a route, writes a comment and gets token rewards.

FollowRoutePanel: This class is for the panel where user sees the route he/she is following and tracks progress. They can leave a route early (leaveRoute()) if they wish. CheckPoint controls are done in this panel (CPControl()).

ChosenRoutePanel: This class is for the panel where user sees detailed information about a route they choose.

SearchPanel: Search panel is the class where user searches routes to follow. This panel contains two more panels which are FilterPanel and RoutePanel.

FilterPanel: This the panel which opens when user wants to filter search results.

RoutePanel: This panel is the show search results (routes). May be one page or more pages.

CreateRoutePanel: This class is for the panel where user creates a route. User can leave markers and code tracks and shows users movements here.

CreateRouteDonePanel: When desired route is drawn, in this panel user enters detailed info about route selects checkpoints and make his/her route public.

FollowedRoutesPanel: This class is for the panel where user routes they followed.

FavoritesPanel: This class is for the panel where user sees their favorite routes.

CreatedRoutesPanel: This class is for the panel where user sees their created routes.

ProfilePanel: This class is for the panel where user sees their user information, edit it, goes to followed or created routes panels or goes to favorites panel.

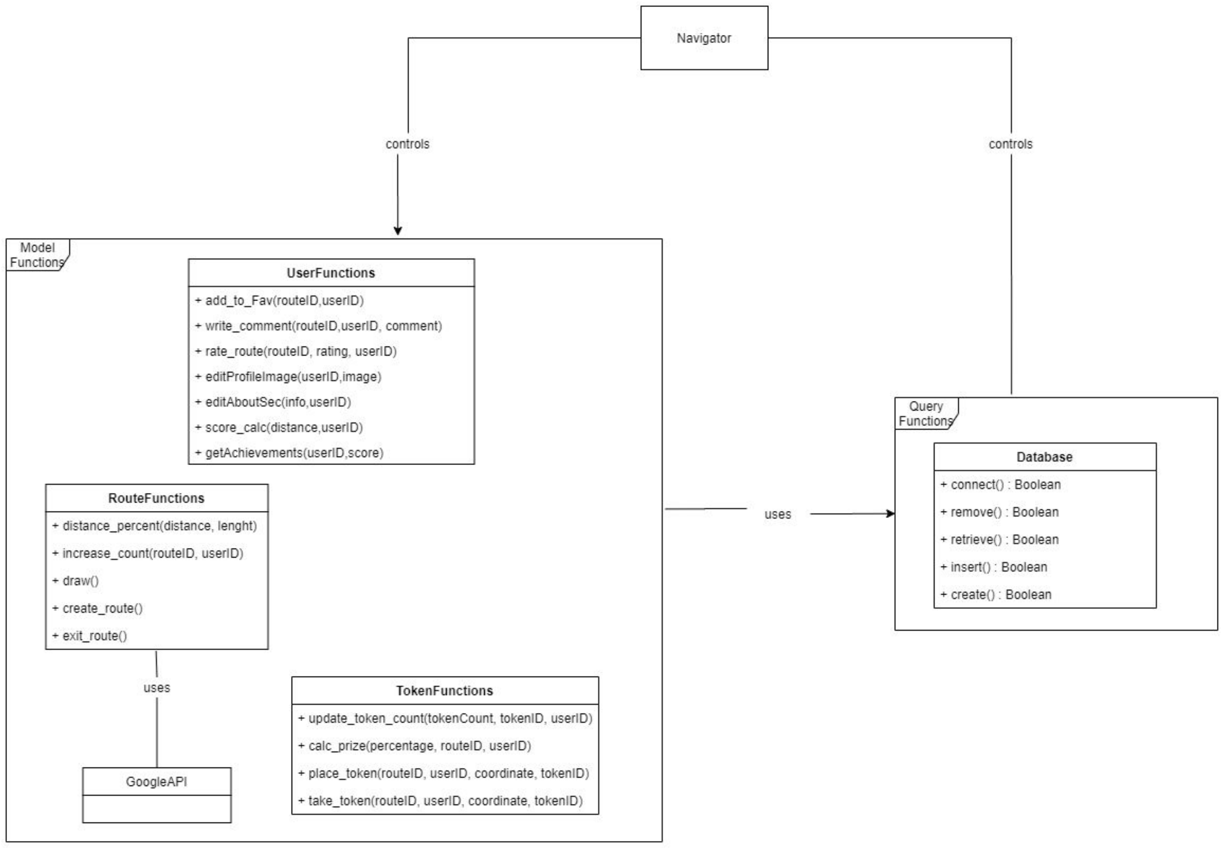
PlaceTokenPanel: This class is for the panel where user access their mobile phone camera and choses a token to place it on screen.

CollectTokenPanel:This class is for the panel where user access their mobile phone camera and collects a token from screen.

Search: This class is for search functions. For example, searching routes by their creator names, searchByUsername().

Navbar: Navbar will be rendered almost in every panel in application.

2.2. Controller


Navigator
controls
controls
Model Functions
UserFunctions
+ add_to_Fav(routeID,userID)
+ write_comment(routeID,userID, comment)
+ rate_route(routeID, rating, userID)
+ editProfileImage(userID,image)
+ editAboutSec(info,userID)
+ score_calc(distance,userID)
+ getAchievements(userID,score)
RouteFunctions
+ distance_percent(distance, lenght)
+ increase_count(routeID, userID)
+ draw()
+ create_route()
+ exit_route()
uses
GoogleAPI
TokenFunctions
+ update_token_count(tokenCount, tokenID, userID)
+ calc_prize(percentage, routeID, userID)
+ place_token(routeID, userID, coordinate, tokenID)
+ take_token(routeID, userID, coordinate, tokenID)
uses
Query Functions
Database
+ connect() : Boolean
+ remove() : Boolean
+ retrieve() : Boolean
+ insert() : Boolean
+ create() : Boolean

In this package classes that provide functionalities and make the program operate are being represented. Navigator was defined in the previous part so here it’s just shown to represent the hierarchy between classes.

Model Functions

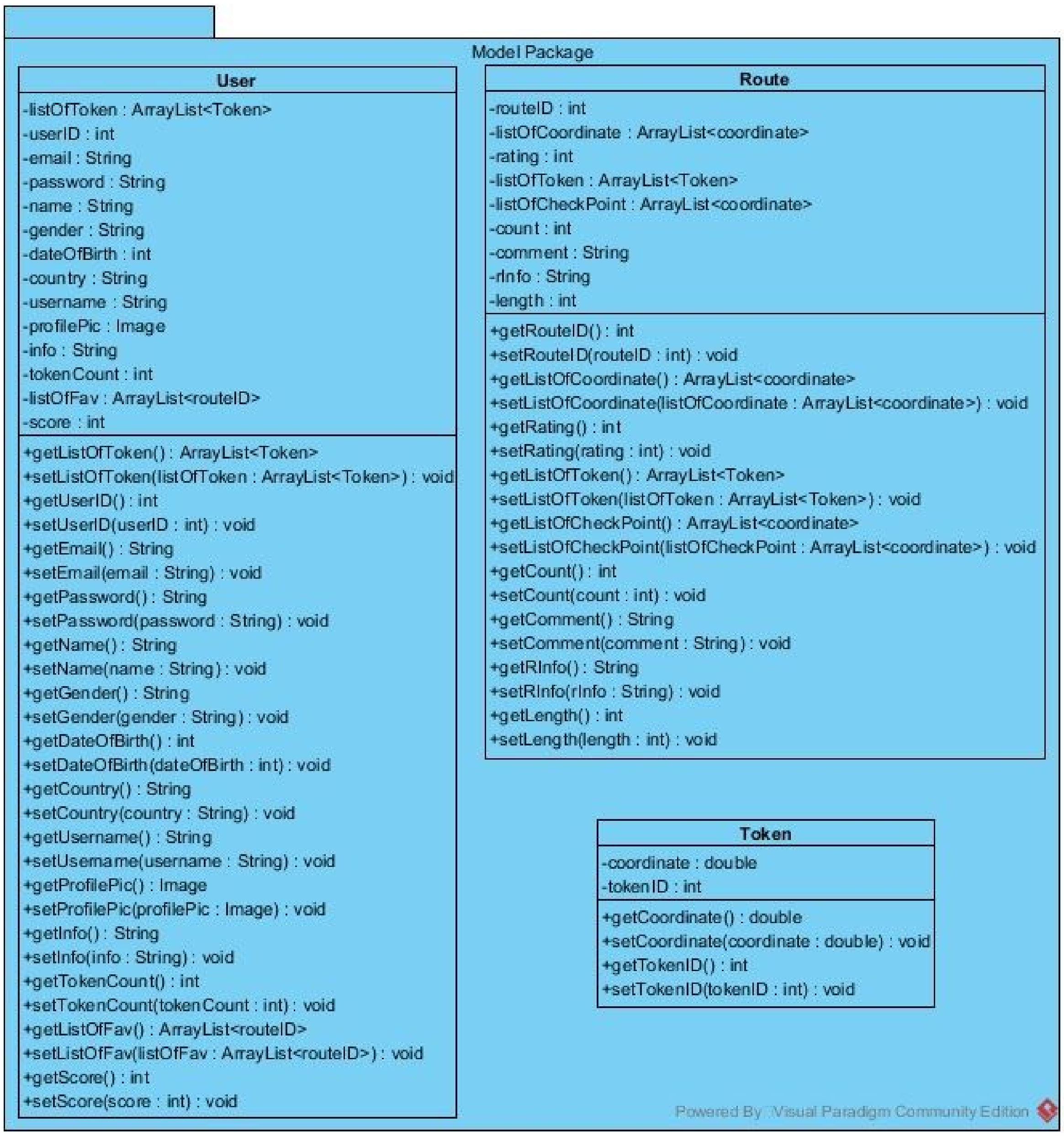
These are the functions that are being used for interacting with our models (objects) and provide most of the functionality available to the user.

> **User Functions**: This class stores the functions that provide personalized functionalities between users so they depend heavily on user instances.
>
> **Route Functions**: This class stores the functions that are related to routes in any way.
>
> **Token Functions**: This class stores the functions that are related to tokens placed on routes.

Query Functions

These are the functions that are used in the application layer to connect with the database.

2.3. Model


Model Package
User
-listOfToken : ArrayList<Token>
-userID : int
-email : String
-password : String
-name : String
-gender : String
-dateOfBirth : int
-country : String
-username : String
-profilePic : Image
-info : String
-tokenCount : int
-listOfFav : ArrayList<routeID>
-score : int
+getListOfToken() : ArrayList<Token>
+setListOfToken(listOfToken : ArrayList<Token>) : void
+getUserID() : int
+setUserID(userID : int) : void
+getEmail() : String
+setEmail(email : String) : void
+getPassword() : String
+setPassword(password : String) : void
+getName() : String
+setName(name : String) : void
+getGender() : String
+setGender(gender : String) : void
+getDateOfBirth() : int
+setDateOfBirth(dateOfBirth : int) : void
+getCountry() : String
+setCountry(country : String) : void
+getUsername() : String
+setUsername(username : String) : void
+getProfilePic() : Image
+setProfilePic(profilePic : Image) : void
+getInfo() : String
+setInfo(info : String) : void
+getTokenCount() : int
+setTokenCount(tokenCount : int) : void
+getListOfFav() : ArrayList<routeID>
+setListOfFav(listOfFav : ArrayList<routeID>) : void
+getScore() : int
+setScore(score : int) : void
Route
-routeID : int
-listOfCoordinate : ArrayList<coordinate>
-rating : int
-listOfToken : ArrayList<Token>
-listOfCheckPoint : ArrayList<coordinate>
-count : int
-comment : String
-rInfo : String
-length : int
+getRouteID() : int
+setRouteID(routeID : int) : void
+getListOfCoordinate() : ArrayList<coordinate>
+setListOfCoordinate(listOfCoordinate : ArrayList<coordinate>) : void
+getRating() : int
+setRating(rating : int) : void
+getListOfToken() : ArrayList<Token>
+setListOfToken(listOfToken : ArrayList<Token>) : void
+getListOfCheckPoint() : ArrayList<coordinate>
+setListOfCheckPoint(listOfCheckPoint : ArrayList<coordinate>) : void
+getCount() : int
+setCount(count : int) : void
+getComment() : String
+setComment(comment : String) : void
+getRInfo() : String
+setRInfo(rInfo : String) : void
+getLength() : int
+setLength(length : int) : void
Token
-coordinate : double
-tokenID : int
+getCoordinate() : double
+setCoordinate(coordinate : double) : void
+getTokenID() : int
+setTokenID(tokenID : int) : void
Powered By Visual Paradigm Community Edition

2.3.1. User Class

listOfToken: This attribute is to keep track of the token of a user collected. Since the overall project highly depends on tokens that the users collected, it is crucial to prevent from unfair advantages.

userID: This attribute is to give an identity to a user.

email: This attribute will be used when the login and sign up processes.

password: Users can create a custom password if they do not want to login via Google, Facebook or Twitter.

name: When a user wants to share his/her name, this attribute is used.

gender: This is to show the user's gender.

dateOfBirth: This is to show the user's date of birth.

country: This is used to distinguish the routes that will be shown to users when they search about it.

username: This attribute will be used in their profile when other users want to see other users' profile.

profilePic: Users will be able to put a profile image.

info: This is for users to share their information.

tokenCount: The user will be able to share a route after collecting enough tokens.

listOfFav: When users want to store their favorite routes, this list fill hold that.

score: This will be their overall score that are collected from path followings.

2.3.2. Route Class

routeID: This attribute is for other users to be able to add it to favorites and for us to store it.

listOfCoordinate: Our routes will be kept as samples from the routes. These samples are the coordinates.

rating: This attribute is a search criteria for other users.

listOfToken: This is to keep the tokens of the route.

listOfCheckPoint: This is to keep the check points' coordinates. It is different from tokens in the sense that a user can get partial points by passing checkpoints.

count: It is used to count the number of people who followed that route.

comment: This attribute is used when a user wants to add comments about the route while the creation stage.

rInfo: It is used to show the information about a route.

length: It is used to show the length of the route.

2.3.3. Token

coordinate: This attribute is to show the coordinate of the token.

tokenID: It is to eliminate the reuse of the tokens.

3. Class Interfaces

3.1. View

Class Name Navigator
This is the manager component that shifts between the pages according to user actions. Also communicates with Controller layer.
Attributes
signUpPanel: SignUpPanel() loginPanel: LoginPanel() rateRoutePanel: RateRoutePanel() followRoutePanel: FollowRoutePanel() mainPagePanel: MainPagePanel() chosenRoutePanel: ChosenRoutePanel() searchPanel: SearchPanel() collectTokenPanel: CollectTokenPanel() placeTokenPanel: PlaceTokenPanel() createdRoutesPanel: CreatedRoutesPanel() favoritesPanel: FavoritesPanel() profilePanel: ProfilePanel()

followedRoutesPanel: FollowedRoutesPanel() createRoutePanel: CreateRoutePanel() createRouteDonePanel: CreateRouteDonePanel()
Methods
Navigator(): Calls the navigator

Class Name SignUpPanel
This class is for the panel where user signs up to our system.
Attributes
Methods
SignUpPanel(): calls the sign up panel checkUser (e-mail, password): Check user login info signUser(): Signs the user

Class Name Login Panel
This class is for the panel where user logs in to our system. Either by our system or with google, facebook, twitter accounts.
Attributes

Methods
LoginPanel(): Calls the login panel checkUser (e-mail, password): Check user login info googleUser (e-mail, password): get info for google user twitterUser (e-mail, password): get info for twitter user facebookUser (e-mail, password): get info for facebook user

Class Name MainPagePanel
This class is for the panel where user what to do. They can either create or follow a route or choose one of the Top 10 routes listed here, they can see their token count.
Attributes
listOfRoute: ArrayList <Route>
Methods
mainPagePanel(): calls the main page panel getTokenCount(): gets the token count

Class Name RateRoutePanel
This class is for the panel where user rates a route, writes a comment and gets token rewards.

Attributes
route: Route comment: String rating: int
Methods
RateRoutePanel(): calls the rate route panel rateRoute(): method for rating route getTokenPrize(): gets the token

Class Name FollowRoutePanel
This class is for the panel where user sees the route he/she is following and tracks progress. They can leave a route early (leaveRoute()) if they wish. CheckPoint controls are done in this panel (CPControl()).
Attributes
route: Route
Methods
FollowRoutePanel(): calls the follow route panel leaveRoute(): exits the route endRoute(): finishes route CPControl(): to control check points

Class Name ChosenRoutePanel
This class is for the panel where user sees detailed information about a route they choose.
Attributes
route: Route
Methods
ChosenRoutePanel(): calls the chosen route panel addFav(Route, User): add route to the favorites seeProfile(): to see user’s profile startRoute(): starts to route

Class Name SearchPanel
Search panel is the class where user searches routes to follow. This panel contains two more panels which are FilterPanel and RoutePanel.
Attributes
filterPanel: FilterPanel routePanel: RoutePanel
Methods

SearchPanel(): calls the search panel searchByText(text, filterPanel): search routes by text and filter

Class Name FilterPanel
This the panel which opens when user wants to filter search results.
Attributes
search: Search listOfFilters: ArrayList <String>
Methods
FilterPanel(): calls the filter panel getFilters(): return the filters

Class Name RoutePanel
This panel is the show search results (routes). May be one page or more pages.
Attributes
listOfRoute: ArrayList <Route>
Methods

RoutePanel(): calls the route panel getRoutes(): return the routes

Class Name CreateRoutePanel
This class is for the panel where user creates a route. User can leave markers and code tracks and shows users movements here.
Attributes
route: Route
Methods
CreateRoutePanel(): calls the create route panel leaveMarker(): leave marker for checkpoints gpsStart(): to start the gps updateRoute(): updates the route save(): save the route endRoute(): ends the route

Class Name CreateRouteDonePanel
When desired route is drawn, in this panel user enters detailed info about route selects checkpoints and make his/her route public.

Attributes
aboutRoute: String tokens: Token
Methods
CreateRouteDonePanel(): calls the create route done panel getTokenCount(): returns the token count goPublic(token, Route): publishes the root

Class Name FollowedRoutesPanel
This class is for the panel where user routes they followed.
Attributes
listOfRoute: ArrayList <Route>
Methods
FollowedRoutesPanel(): calls the followed routes panel getFollowedRoutes(). returns the followed routes

Class Name FavoritesPanel
This class is for the panel where user sees their favorite routes.

Attributes
listOfRoute: ArrayList <Route>
Methods
getFavRoutes(): returns the favorite routes FavoritesPanel(): calls the favorites panel

Class Name CreatedRoutesPanel
This class is for the panel where user sees their created routes.
Attributes
listOfRoute: ArrayList <Route>
Methods
CreatedRoutesPanel(): calls the created routes getRoutes(): return routes

Class Name ProfilePanel
This class is for the panel where user sees their user information, edit it, goes to followed or created routes panels or goes to favorites panel.

Attributes
user: User
Methods
ProfilePanel(): calls the profile panel getTokenCount(): returns the token count getFunctions(): return the functions

Class Name PlaceTokenPanel
This class is for the panel where user access their mobile phone camera and choses a token to place it on screen.
Attributes
token: token
Methods
PlaceTokenPanel(): calls the place token panel setToken(): set method for token decideToken(): decide the tokens

Class Name CollectTokenPanel

This class is for the panel where user access their mobile phone camera and collects a token from screen.
Attributes
token: token
Methods
CollectTokenPanel(): calls the collect token panel getToken(): return the tokens

Class Name Search
This class is for search functions. For example, searching routes by their creator names, searchByUsername().
Attributes
search: Search listOfFilter: ArrayList <String>
Methods
getFilters(filters): return the filters Search(): general search method searchByUsername(): searchs by username

3.2. Controller

3.2.1. Model Functions

Class Name UserFunctions
This class is to share the users experience and the information they will give about themselves and to save the users experience and achievements in their account.
Attributes
Methods
add_to_Fav(routeID, userID): Adds the selected route to users favorites. write_comment(routeID, userID, comment): Adds the users comment to the selected route. rate_route(routeID, rating, userID): Changes the selected routes rating when the user rates it. editProfileImage(userID, image): Uploads the selected image to users profile. editAboutSec(info, userID): Changes the users information about himself/herself. score_calc(distance, userID): Calculates the score that the user got from the route while following the route. getAchievements(userID, score): Determines if the user will get an achievement with his/her score.

Class Name RouteFunctions
This class contains functions that creates route and calculate the distance that the users followed on the route.
Attributes
Methods
distance_percent(distance, length): Calculates the completed percentage of the route. increase_count(routeID, userID): Keeps a counter to calculate the distance that the user followed. draw(): Draws a line indicating the route on the map. create_route(): Gives an id to the new routes created by users. exit_route(): terminates the route.

Class Name TokenFunctions
This class is to place/take/update the tokens in the routes and calculate the prizes accordingly.
Attributes
Methods
distance_percent(distance, length): Calculates the completed percentage of the route. increase_count(routeID, userID): Keeps a counter to calculate the distance that the user walkthrough.

draw(): Draws a line indicating the route on the map. create_route(): Gives an id to the new routes created by users. exit_route(): terminates the route.

3.2.2. Query Functions

Class Name Database
It contains functions used in the application layer to connect with the database.
Attributes
Methods
connect(): Connects to the database. remove(): Removes from the database. retrieve(): Retrieves data from the database. insert(): Puts new data to the database. create(): Creates new tables in the database.

3.3. Model

3.3.1. User

Class Name User
It contains the necessary information of user object according to the database.

Attributes
listOfToken: ArrayList<Token> userID: int password: String name: String gender: String dateOfBirth: int country: String username: String profilePic: Image info:String tokenCount: int listOfFav: ArrayList<routeID> score: int
Methods
getListOfToken() : Returns the list of tokens the user collected. setListOfToken(listOfToken : ArrayList<Token>) : Adds token to the collected tokens list. getUserID() : Returns the ID of the user. setUserID(userID : int) : Edits the ID of the user. getEmail() : Returns the email address of the user. setEmail(email : String) : Edits the email address of the user. getPassword() : Returns the password of the user. setPassword(password : String) : Edits the password of the user.

getName() : Returns the name of the user. setName(name : String) : Edits the name of the user getGender() : Returns the gender of the user. setGender(gender : String) : Edits the gender of the user. getDateOfBirth() : Returns the birthday of the used. setDateOfBirth(dateOfBirth : int) : Edits the birthday of the user. getCountry() : Returns the country the user lives in. setCountry(country : String) : Edits the country the user lives in. getUsername() : Returns the username of the user. setUsername(username : String) : Edits the username of the user.

3.3.2. Route

Class Name Route
It contains necessary information of user object according to the database.
Attributes
routeID : int listOfCoordinate : ArrayList<coordinate> rating : int listOfToken : ArrayList<Token> listOfCheckPoint : ArrayList<coordinate> count : int

comment : String rInfo : String length : int
Methods
getRouteID() : Returns the ID of the route. setRouteID(routeID : int) : Edits the ID of the route. getListOfCoordinate() : Returns the coordinates that constitutes the route. setListOfCoordinate(listOfCoordinate : ArrayList<coordinate>) : Edits the coordinates of the route. getRating() : Returns the rating of the route. setRating(rating : int) : Edits the rating of the route. getListOfToken() : Returns the tokens on the route. setListOfToken(listOfToken : ArrayList<Token>) : Edits the tokens on the route. getListOfCheckPoint() : Returns the checkpoints of the route. setListOfCheckPoint(listOfCheckPoint : ArrayList<coordinate>) : Edits the checkpoints on the route. getCount() : Returns how many times the route is used. setCount(count : int) : Edits the number of times the route is used. getComment() : Returns the comments of the route. setComment(comment : String) : Edits the comments of the route. getRInfo() : Returns the information about the route. setRInfo(rInfo : String) : Edits the information about the route. getLength() : Returns the length of the route.

setLength(length : int) : Edits the length of the route.

3.3.3. Token

Class Name Token
It contains the necessary information of user object according to the database.
Attributes
coordinate : double tokenID : int
Methods
getCoordinate() : Returns the coordinates of the token. setCoordinate(coordinate : double) : Edits the coordinates of the token. getTokenID() : Returns the ID of the token. setTokenID(tokenID : int) : Edits the ID of the token.

4. References

[1] J. Fund, “Travel Is So Much Better Than It Was,” National Review, 02-Jan-2017. [Online]. Available: www.nationalreview.com/2017/01/international-travel-today-much-easier-cheaper-it-was/. [Accessed: 12-Oct-2019].