Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: AeroCast

Project High-Level Design Report

Ahmet Alparslan Çelik, Ömer Berk Uçar, Muhammed Yusuf Satıcı, Yasin İlkağan Tepeli, Ahmet Taha Albayrak

Supervisor: Çiğdem Gündüz Demir
Jury Members: Selim Aksoy and Mustafa Özdal

Progress Report

Dec 22, 2017

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

Introduction 2
Purpose Of The System 3
Design Goals 3
Definitions, Acronyms and Abbrevations 5
Overview 6
Current Architectures 7
System Architecture 10
Overview 10
Subsystem Decomposition 10
Hardware/Software Mapping 13
Persistent Data Management 14
Access Control And Security 14
Global Software Control 15
Boundary Conditions 15
Subsystem Services 16
Client 16
Presentation Tier 17
Application Server 18
Logic Tier 19
Data Tier 20
Machine Learning Server 21
Logic Tier 22
Data Tier 23
Glossary 25
References 26

1. Introduction

While planning a holiday or a business trip, everyone once in awhile tries to find the best airfare deal according to their needs. In general, it is a tedious work due to the dynamic changes in prices. Airlines change their prices according to various criteria such as supply and demand, how far the departure date is, which day of the week the flight is, etc. Thus, finding the cheapest ticket by searching manually can be cumbersome. Furthermore, it is necessary to optimally find the best purchase date for a particular route within the traveller's' budget.

With the current technologies, it is possible to track airfares, get notifications in case of a drop or increase in price and even get rough predictions if the price will increase or decrease for a particular route. Nevertheless, the existing methods do not provide user specific airplane predictions. Therefore, in order to purchase the ideal cheapest ticket, travellers will need to use separate services. Moreover, if they were to look for different travelling plans, the number of different routes they need to check will increase making things even more time consuming and complicated.

Aerocast aims to minimize the efforts put into finding the best airfare deal available. It will be a web application, which collects the available data scattered through different web services and analyzes them in a user specific way. Therefore, it will assist travellers to find the best airfare in the minimum possible time. The users can search and track their flight of interest months before the departure and can get elusive intelligence. Regarding to the personal preferences, users can provide preferences such as carrier type or cabin class, Aerocast will list the most related results.

In the report, a description of the system decomposition as well as the system architecture is provided. Initially, the purpose of the system and the design goals are specified. Then, the details of the system architecture and the differences between the current architectures and the proposed architecture is discussed. Later, the subsystem decomposition of the system and the detailed description of the subsystem components is provided. Also, the data management and security decision are explained along with the hardware/software choices made in the design of the architecture. Finally, the subsystem services are specified in this report.

1.1. Purpose Of The System

AeroCast aims to provide an effective tool for finding the optimum purchase and departure dates in order to buy flight tickets. For this purpose, AeroCast provides an forecasting system to the users, which enables the user to obtain the departure dates having the best purchase dates with the lowest prices. Also, the AeroCast aims to enable users to track the changes in the prices and predictions of their search results. In addition to the forecasting features, the system provides a visualization of the flight prices to the users in order to give them more insight about the changes in the flight prices. Therefore, AeroCast aims to provide an efficient way of forecasting and tracking the airfares as well as visualizing the flight data.

1.2. Design Goals

1.2.1. Availability

AeroCast aims to provide continuous service to its users except the scheduled maintenance times, which will be between 20.00 and 22.00 on the first Monday of each month. Therefore, AeroCast aims to be an available and operable system by decreasing the probability of crashes and failures with the help of reliable and supportable components.

1.2.2. Maintainability

AeroCast aims to enable the modules of the system to work independently from each other and the decomposition of the system into subsystem tries to provide modularity between different servers and components of AeroCast. Therefore, by having this kind of modularity, AeroCast should be able to enable the addition of new functionalities to the system and the changes in the current functionalities of the system by experiencing little or no effect on the rest of the system. Also, when one of the components crashes, the modularity of the system enables AeroCast to preserve the working state of its other components.

1.2.3. Response Time

AeroCast should be highly responsive and it should respond to the requests of the system as quickly as possible. Therefore, AeroCast aims to keep response time of the system under five seconds for its prediction system and under two seconds for the remaining functionalities of the system.

1.2.4. Scalability

AeroCast aims to be a scalable system which can support more than 10,000 users. Also, AeroCast system needs to store a continuously increasing amount of data which requires the system to be highly scalable in order to meet the increasing data storage requirements of the project. Hence, the AeroCast system aims to provide scalable database as well as maintaining the consistency and accuracy of the outputs of the system.

1.2.5. Security

The AeroCast system aims to protect the user information and flight data stored in its the databases. Therefore, AeroCast uses hashing with SHA-256 to protect the password information of the users and it limits the number of daily requests to 200 to protect the flight data from malicious actions. Also, AeroCast aims to protect its web servers by using the Cloudflare Services.

1.2.6. Reliability

It is aimed that the AeroCast system is highly reliable and unlikely to face failures. Therefore, the design of the system architecture and the decomposition of the system into subsystems is done such that the failure or the probability of failure in one of the core functionalities such as forecasting the airfares will not affect the working state of the system. Also, the AeroCast system targets to provide reliable forecasts and accurate flight data to the users by collecting data from various sources and making predictions based on the high amount of collected data.

1.2.7. Usability

Ease of Use is one of the most important factors that affects the design and development of AeroCast system. The user should be able to use the AeroCast system without requiring any kind of tutoring or assistance. Therefore, the AeroCast system aims to

provide an understandable interface in which the user is required to supply only a small number of inputs to use the most of the AeroCast's functionalities. Other than the user profile information, the user is able to use most of the AeroCast functionalities such as forecasting flight prices, tracking prices, getting notifications and viewing graphs only by specifying the flight search criteria for the specific flight he wants to search and by clicking the appropriate button. Also, AeroCast provides an interactive graph to the user to make the system more efficient to use. Hence, the AeroCast system aims to provide a neatly shown and user friendly interface to the users.

1.2.8. Portability

The system shall run in desktop and mobile web platforms.

1.2.9. Extensibility

AeroCast aims to make it easy to add new components and functionalities by providing modularity in its architecture and subsystems. Therefore, AeroCast system should be open to the future growth and it should be able to extend its current system and architecture to enable the addition of new components and functionalities.

1.3. Definitions, Acronyms and Abbrevations

API: Application Programmer Interface.

RESTful API: API with representational state transfer.

Client: The application that initiates the connection with server.

Server: The program that provides the services to the application.

Open API specification: Standartized interface description for REST APIs.

MongoDB Wire Protocol: Application layer protocol for the database connection.

HTTP: Application Layer Protocol.

1.4. Overview

AeroCast is a web application that aims to enable the users to find the optimum time to buy the flight tickets. AeroCast makes purchase date predictions which tries to determine when the prices of the flights are predicted to be the lowest and it makes departure date forecast that tries to find which departure dates are expected to have low flight prices. In addition to the purchase and departure date forecasts, the users see the expected prices for the predicted purchase dates. Also, AeroCast shows the history of flight prices to the users and it enables the users to track the flights. The users can get notifications when the predictions change and they can give feedback to the system as well as the particular predictions. By using the AeroCast system, the users are able to get comprehensive insight about the flights based on the date forecasts and flight statistics.

AeroCast gets various specifications from the users such as origin, destination, departure date interval, cabin class and carrier code to make forecasts for the specified search of the user. Based on the criteria specified by the user, AeroCast determines the best departure dates having the lowest prices and for these departure dates, AeroCast predicts the purchase dates having the expected lowest price. Therefore, unlike the current systems, AeroCast enables the users to get a two staged prediction regarding the departure and purchase dates along with the expected price of the flight. Also, AeroCast provides the option of making predictions for multiple departure dates, which makes the system more flexible. AeroCast provides filtering, sorting and tracking options to the users and it enables the users to receive notifications for the changes in the flight prices. Unlike most of the current systems, the users not only see the current prices and the future predicted prices of the flights but they also see the past price data of the flights. AeroCast visualizes the history of price data along with the predicted prices of the future purchase dates on an interactive graph. AeroCast also continuously collects flight data from various sources and it updates the airfare forecasts based on the newly collected data. Therefore, AeroCast provides a

dynamic service to the users that offers a comprehensive amount of past and future data and makes flexible multi-staged airfare forecasts.

2. Current Architectures

There are various applications providing services regarding the airfares and flights. Two kind of applications show some similarities with the proposed system of AeroCast and two main applications are discussed as current systems in this report.

1. Applications, providing flight information to users. These applications enable the user to search, buy and track the current flights for various destinations, origins, departure dates and carrier codes. However, they do not make forecasts for airfares.

   Skyscanner[1]:

   - It enables the users to search the current flight prices based on the criteria specified by the user.
   - The user can display the prices of the flights in different sort orders based on the price or duration of the flights.
   - Skyscanner enables the users to track their flights and it alerts the users when the prices of the flights change.
   - The users need to register into the skyscanner system to get alerts on the flights.
   - The skyscanner also let the users choose multiple destinations for their flight searches.
   - Skyscanner does not provide any forecasting system and it does not show the previous flight data.

   AirfareWatchDog[2]**:**

   - It enables the users to find current cheap flight tickets based on the criteria specified by the user.

- The users are able to track flights and get alerts by providing their email address.
- AirfareWatchDog does not provide any forecasting system and it does not show the previous flight data.
- AirfareWatchDog enables the users to compare cheap flights from different sources.

Expedia[3]**:**

- Similar to the above applications, Expedia enables the users to search, track and filter the current flight prices.
- The users can create lists of favorite flights and trips.
- The users need to log into the system to track the flights and create the lists.
- Expedia also provides the users the option of searching for flights and hotels together.
- Expedia does not provide any forecasting system and it does not show the previous flight data.

TripAdvisor[4]**:**

- TripAdvisor also enables the users to search for the current cheap flight prices.
- The users can save the searches and see their lastly searched flights.
- TripAdvisor provides the option of comparing its results with the results of Expedia, in which TripAdvisor mostly gives worse results than Expedia.
- TripAdvisor does not provide any forecasting system and it does not show the previous flight data.

2. Applications providing forecasts for the flight prices. These applications enable the user to buy the flight tickets at the time that the flight prices are going to be low. Mostly, these applications do not provide flight data and make only purchase date predictions.

Hopper[5]:

- Hopper is a mobile application that aims to predict when the users should buy the flight tickets.
- Hopper shows the current cheap flight price to the user and enables the user to track the flights.
- Hopper does not require the users to log into the system to use its functionalities.
- Hopper makes predictions as buy now or wait.
- It does not make specific purchase date predictions and it does not show expected prices for the flights.
- Hopper does not provide previous flight data to the users.
- Hopper sends notifications to the users when the prices for one of the tracked flights change.

Fairfly[6]:

- Fairfly makes airfare predictions for corporate travel.
- Fairfly only serves to the travel management companies and enterprises. It is not open to the public.

Kayak[7]**:**

- It enables the users to search the current flight prices based on the criteria specified by the user.
- Kayak enables the user to track, filter and sort the search results based on multiple criteria.
- Kayak make predictions as the prices of the flights are expected to decrease or increase in the next week. Kayak make forecasts only for seven days.
- It does not make specific purchase date or departure date predictions and it does not make any prediction for the dates later than one week.
- It does not show the expected prices.
- Kayak has an explore option that shows the previous flight price data on a map. The user can specify the origin and date interval in the explore option

and Kayak displays the past prices of the flights found in the specified date interval and having the specified origin. The past prices are shown on top of the destinations in a world map.

- The user can interactively use the map by clicking on the destinations in order to search for current cheap airfares and get predictions for the next week.

3. System Architecture

3.1. Overview

System Architecture section gives detailed information about the decomposition of the system into components. It also explains the decisions made in the design of the system architecture. Firstly, the subsystem decomposition is given. The subsystem decomposition is supported by the subsystem diagrams that show the main components and the architecture of the system. Secondly, hardware/software mapping is specified along with the deployment diagram. In addition to these specifications, persistent data management, access control and security decisions are discussed in detail. Finally, the software control over the system and boundary conditions of the system are explained.

3.2. Subsystem Decomposition

The system uses the Client/Server architecture, which contains one client and two servers. In addition to the client/server architecture, the system uses 3-tier architecture. The client side consists of the presentation tier which manages the connection between client and server. For the Client - Server connection, the REST API is used and the specification of the API is written by using the OpenAPI specifications [8]. The server side contains two servers and each server includes one logic tier, which is responsible from the execution of the application operations and one data tier that is responsible from the storage and management of the data. The client side is responsible from the graphical user interface of the system and it initiates the connection between the application server and client. The

application server is responsible from managing all information related to the user which includes the notifications, profile information and feedbacks. The application server receives the requests of the client. The application server receive the searches of the user from the client and forwards this information to the prediction server in order to get the forecast information. The machine learning server is responsible from producing the forecast for the user's searches. It also keeps the data that is collected from the flight APIs and it regularly collects new data by using its scrapers. Figure - 1 below shows the subsystem decomposition of the application in terms of the tiers and subsystems. Figure - 2 below shows the subsystem decomposition of the subsystem in a detailed manner including the subsystem classes and services.

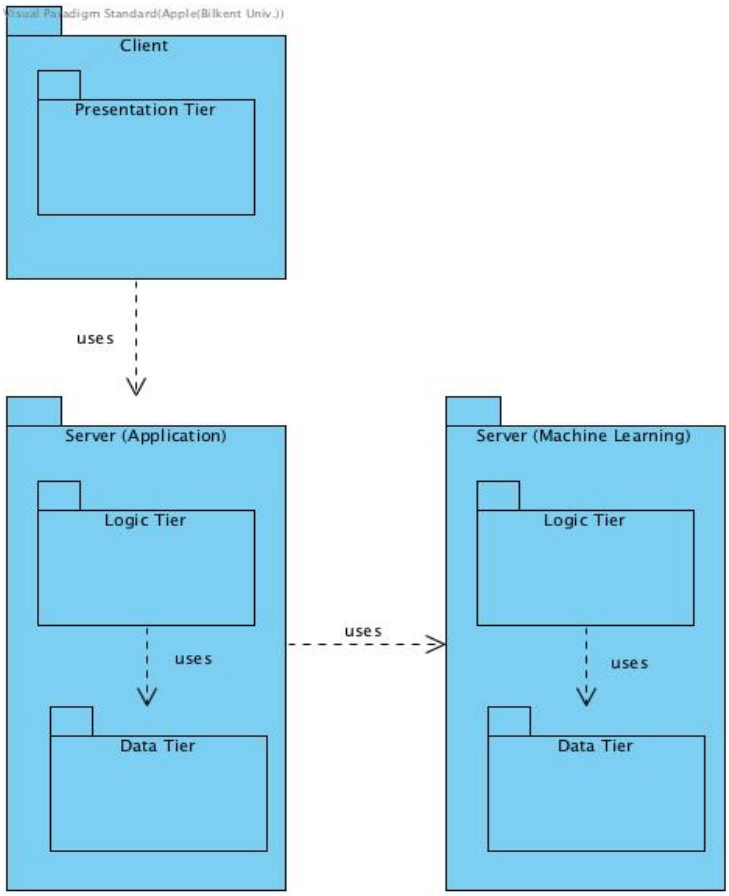

Figure 1 - Subsystem Decomposition

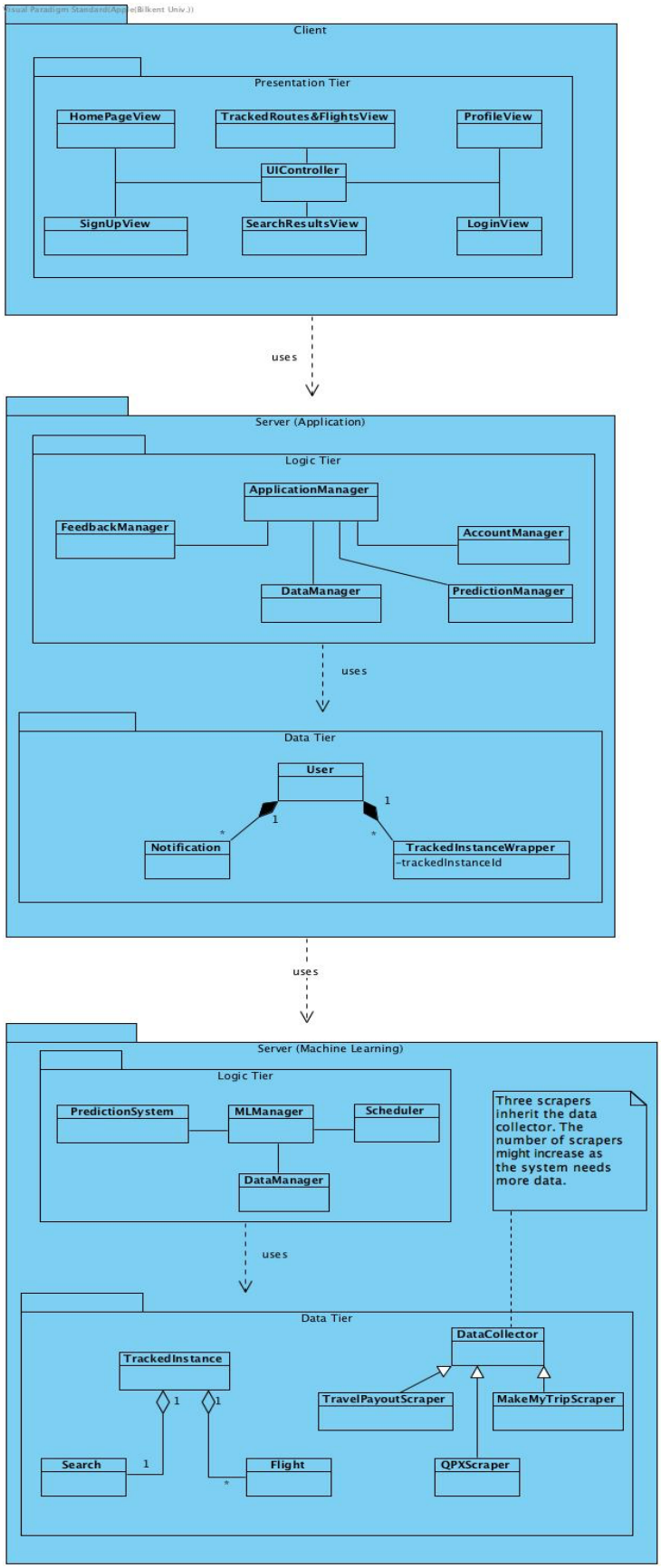

Figure 2 - Subsystem Decomposition with Classes

3.3. Hardware/Software Mapping

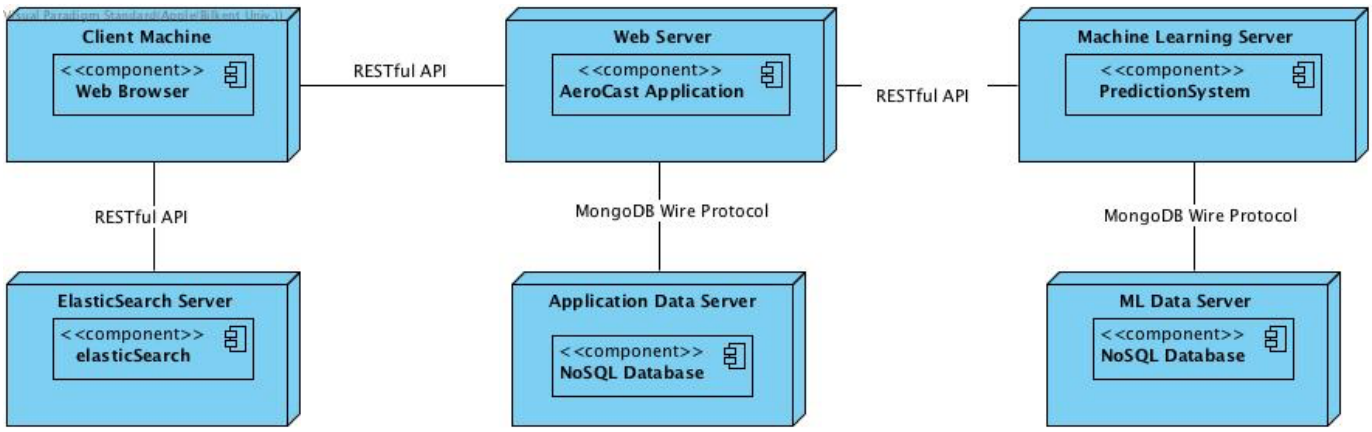

Figure 3 - Deployment Diagram

The client of the system corresponds to the web application. The user uses the web browser of his/her computer to access the application and to perform specific tasks. Client application connects to the ElasticSearch Server and AeroCast web server via RESTful API, which uses HTTP. Client uses the AeroCast Web server to handle the user requests and it uses the ElasticSearch Server to perform fast and efficient searches as the user types departure and destination airport inputs.

AeroCast web server consists of the AeroCast application that is responsible from handling the tasks related to the user profile. The AeroCast application forwards the tasks related to the prediction of Airfares to the Machine Learning Server. This forwarding allows the AeroCast application to run faster by not waiting for the prediction tasks to complete, which might take long periods of time. AeroCast Application uses RESTful API to connect to the Machine Learning Server. Also, AeroCast Web Server connects to the Application Data Server to handle the management of the data. Application Data Server uses MongoDB as NoSQL database for the data storage and AeroCast Application connects to the database via MongoDB Wire Control protocol.

Machine Learning Server is responsible from handling the forecasting tasks and it contains the prediction system of the application. Machine Learning Server returns the prediction results to the AeroCast Web Server which returns the results to the corresponding

user in the Client Server. Machine Learning Server uses MongoDB Wire Control Protocol to connect to its own MongoDB database, which handles the storage of the collected flight data and the prediction results.

3.4. Persistent Data Management

We will store airports, airlines, countries, cities, old flight data as object which will never change. Only exception is that as days progress, new flight data will be added as “old flight data”. Also users, users’ tracked flights and routes, users’ feedback and notifications will be stored but they can be updated as users want.

Most importantly, we will store the predictions we made and they can change as time progresses and new “flight data” comes. We will continuously check the predictions and update them according to “old flight data”. To make data management persistent, we will use MongoDB.

3.5. Access Control And Security

The users can search a flight and get the detailed information of the flight such as price graph and flight number. They can reach the flight and route data without any restriction but these data cannot be modified by the users. The users can only be able to modify data that are their personal information and their tracked route and flight list. The users that did not log in cannot reach profile and tracked flight and route data. The users that logged in can get and modify the tracked route and flight and profile data. They can track the route or any specific flight and remove routes and flights from their track list. They can update their personal information such as password and name. The users cannot see data belongs to the other users.

The users login to the system with their email and passwords.Communication with the server for logging secured by cryptographic communication protocols like SSL. This means the server-client communication is secure and even somebody eavesdrops the connection they cannot get any information. In order to secure the data in the database, our

system stores the encrypted data. Any data from the system database, either it is user information or flight information, is not shared third parties.

3.6. Global Software Control

For the software control, the system uses the event-driven type. The main software controls are:

As the user registers, the credentials, email and password will be checked. When the user enters the email, the system will check the database if it exists or not. As for the password check, the client side will check whether it is in boundaries such as minimum 8 characters, maximum 32 characters, at least 1 upper-case, 1 lower-case, 1 number.

As the user tries to login with email and password, the system will check the credentials by reaching to database. If the credentials are correct, system will give permission to login.

When the user wants to search for the best date for purchase ticket (the prediction system), the user enters the options for the flight such as dates, number of passengers, origin and destination.

3.7. Boundary Conditions

Initialization

The user needs to have a web browser for desktop or mobile. In order to use some of the functionalities like “Track”, “Profile”, “Notification” user needs to register and create an account. As user can login with the registered mail and password, he/she can also login with Google or Facebook Account. If an error is encountered during “Register” or ”Login” the “Register” and ”Login” pop-ups will appear again. As in any website, it is required to have an internet connection.

Termination

When the web browser is closed, the user automatically logs out from the system. In addition, if the user clicks “Log Out”, the user logs out from the system.

Failure

In case the internet connection is lost, the user will not be able to do anything other than to look at the current web page. If the user is editing the profile and does not click “Submit” button, the information on the form will be lost in case the web page is closed. Likewise, if the user is filling the feedback form and does not click “Submit” button the information in form will be lost. Moreover, if the internet connection gets lost, browser will not save the page content like forms.

4. Subsystem Services

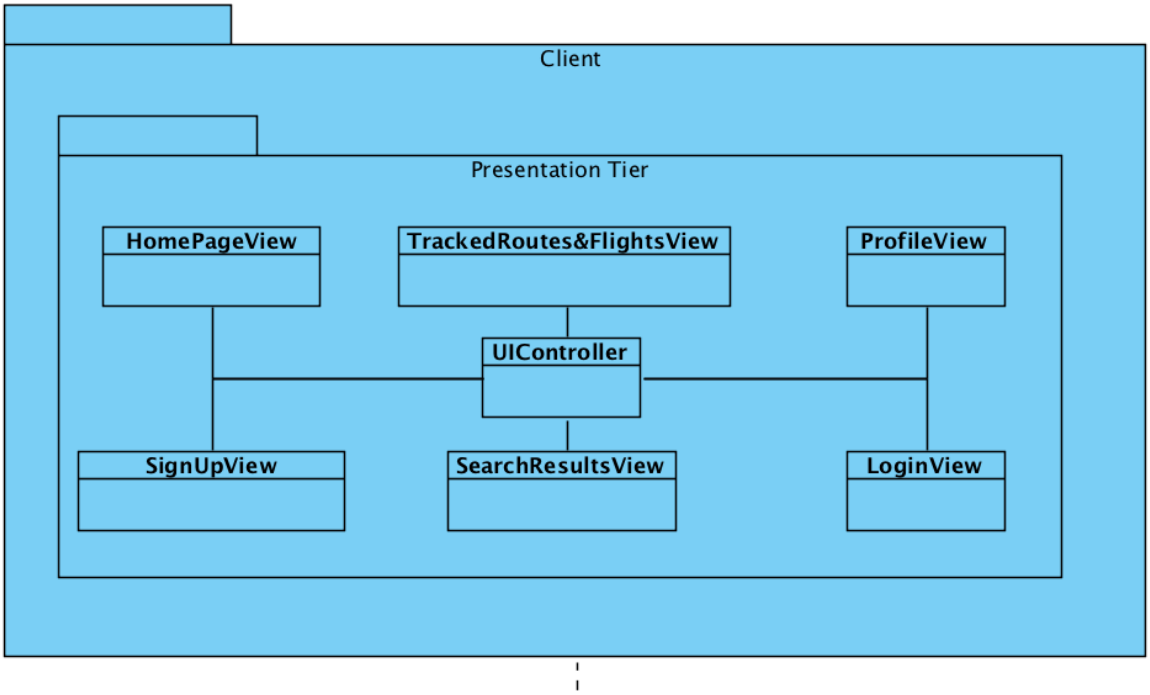

Figure 4 - Client Subsystem

The client corresponds to the web application of our system. The client is the presentation layer of the system. The user creates an account or logs into the system via the

client. The client requests login access from the server. The client is responsible from presenting the data it retrieves from server to user.

The client consists of only the presentation tier. This tier is responsible for the interface operations. The required data will be fetched from the server using specified REST API interface. Therefore all the inputs which is taken from the user should be validated. This tier is also responsible for initiating necessary communication between client and server such as checking user's credentials.

4.1.1. Presentation Tier

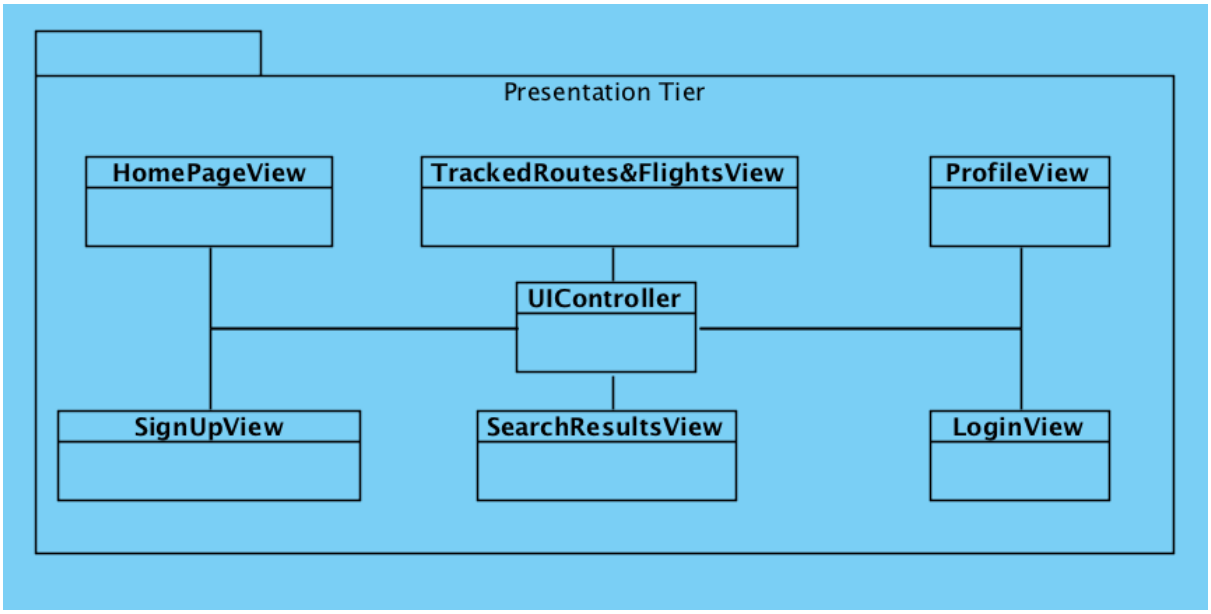

Figure 5 - Presentation Tier

Presentation tier handles all user interface related operations.

UIController: Main control class that arranges the task among the view classes.

SignupView: Class for arranging the account view.

HomepageView: Class that is responsible from the presentation of all components in the homepage of the user.

TrackedRoutes&FlightsView: Class that controls the presentation of tracked routes and flights page.

ProfileView: Class for arranging the profile view.

LoginView: Class that controls the presentation of login page.

4.2. Application Server

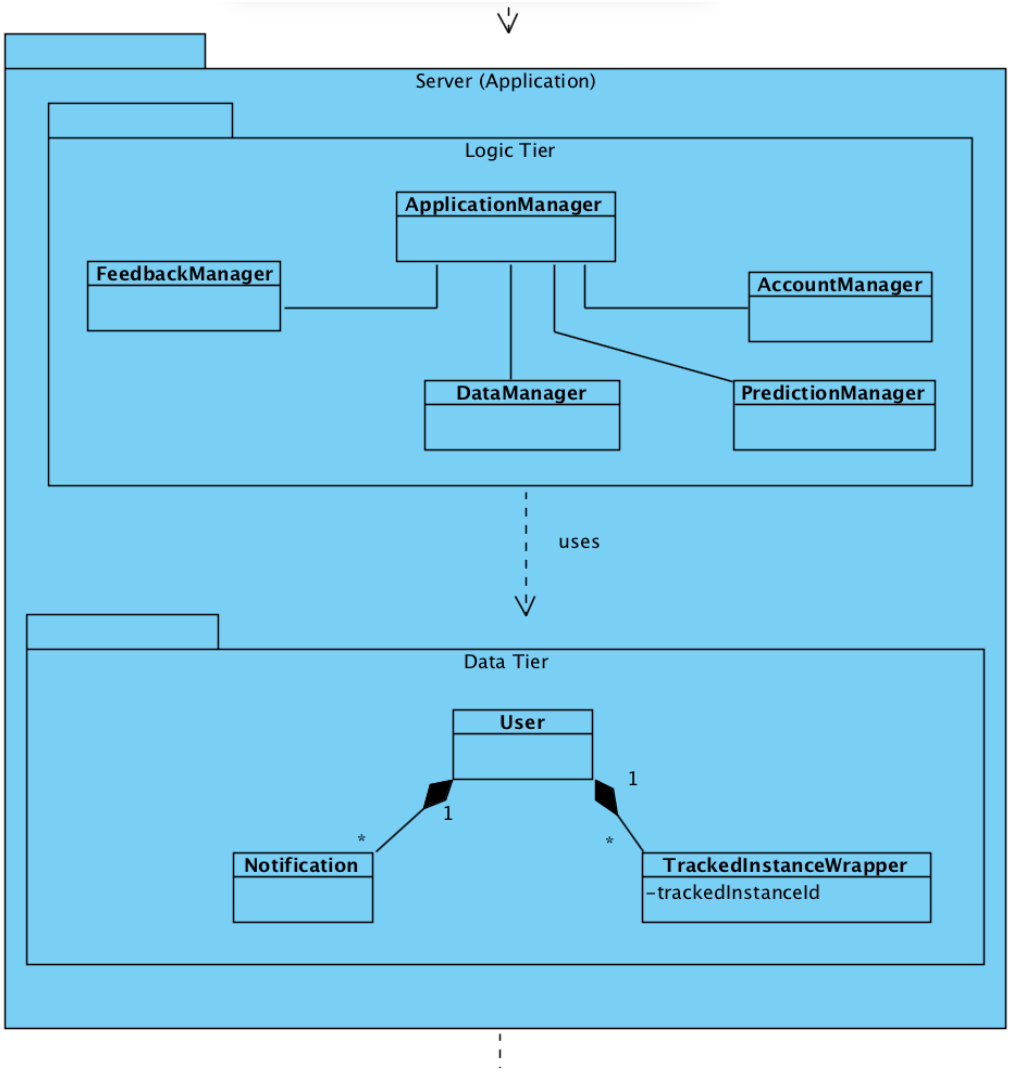

Figure 6 - Application Server Subsystem

Application Server is one of our two servers. It is the main server of the system. It manages databases, front-end requests, the general structure flow of the system. When client send user requests such as registering, logging in and searching, application server

gets the request and handles it. Then it responds to the client and updates the database if necessary or forwards the request to the machine learning server if necessary. Server decides what to do according to the user request.

Application Server has two layers which are Logic Tier and Data Tier. Logic Tier is the one that initially gets the requests and manages them. It decides which operations should be done and what responses should be returned. Logic Tier controls other parts of the system. Data Tier is the one that responsible from the stored data that user sees such as notifications, tracked flights etc.

4.2.1. Logic Tier

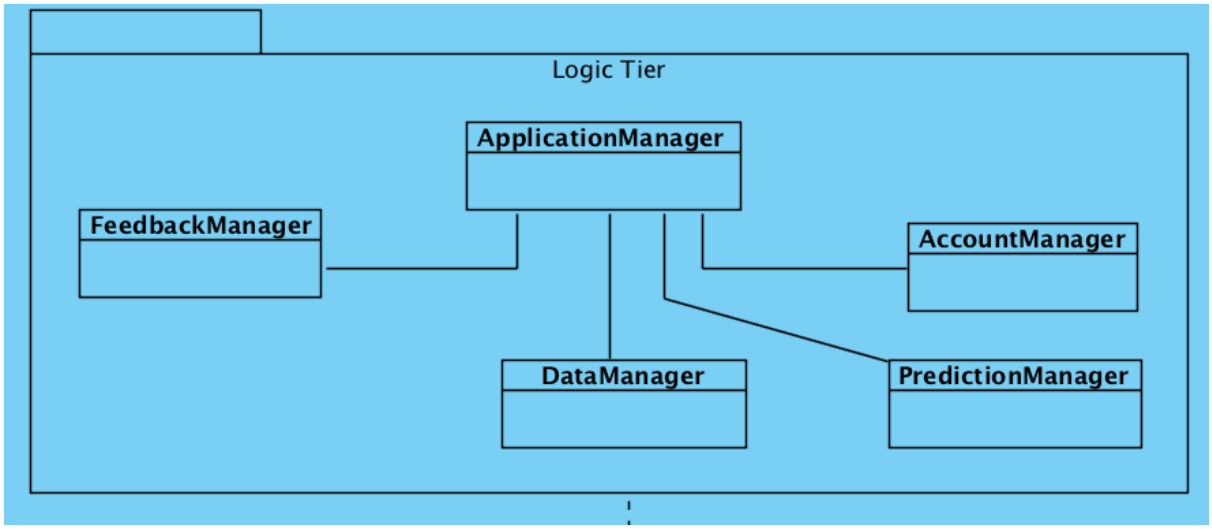

Figure 7 - Logic Tier Of Application Server

Logic Tier is the decision mechanism of the system. It has connections with all other managers and ensures that the system works without any problem.

ApplicationManager: Application Manager class is the main class of the Logic Tier. It manages the general system processes. It is the connection point of other managers. It is the main decision mechanism that charge other managers when a request comes.

FeedbackManager: Feedback Manager class is the data class that connects to feedback database. It manages the feedback requests and work with other managers to store the feedback.

DataManager: DataManager class is data class that manages all database related requests. When it gets the request it connects to database and modifies it according to the request.

AccountManager: AccountManager class manages the account related processes. It controls modification requests of accounts, login and register processes. It works with other managers to store the new accounts into the database or update the database with the modifications.

PredictionManager: Prediction Manager class is the connection point of Application Server and Machine Learning Server. It manages the forecast functionality which needs the Machine Learning Server work.

4.2.2. Data Tier

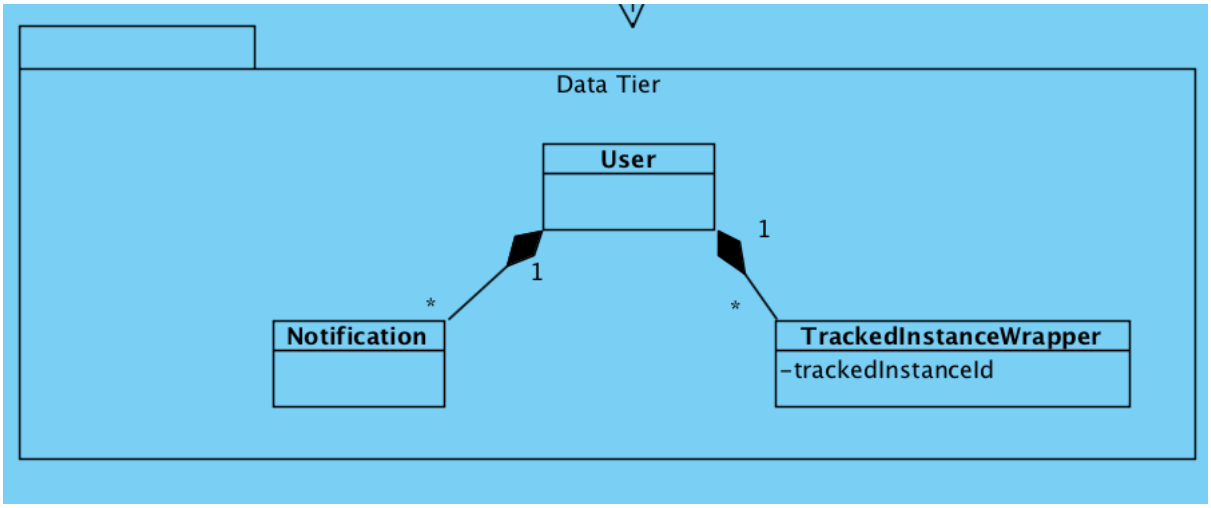

Figure 8 - Data Tier Of Application Server

Data Tier manages data that user can see. It connects to the database whenever a request comes.

User: User class is data class that manages user instance. It contains properties of user information such as name, email, password etc.

Notification: Notification class is the data class that manages notification instance. It contains notification properties.

TrackedInstanceWrapper: Tracked Instance Wrapper class is data class that manages tracked flight or route instance. It contains tracked instance id.

4.3. Machine Learning Server

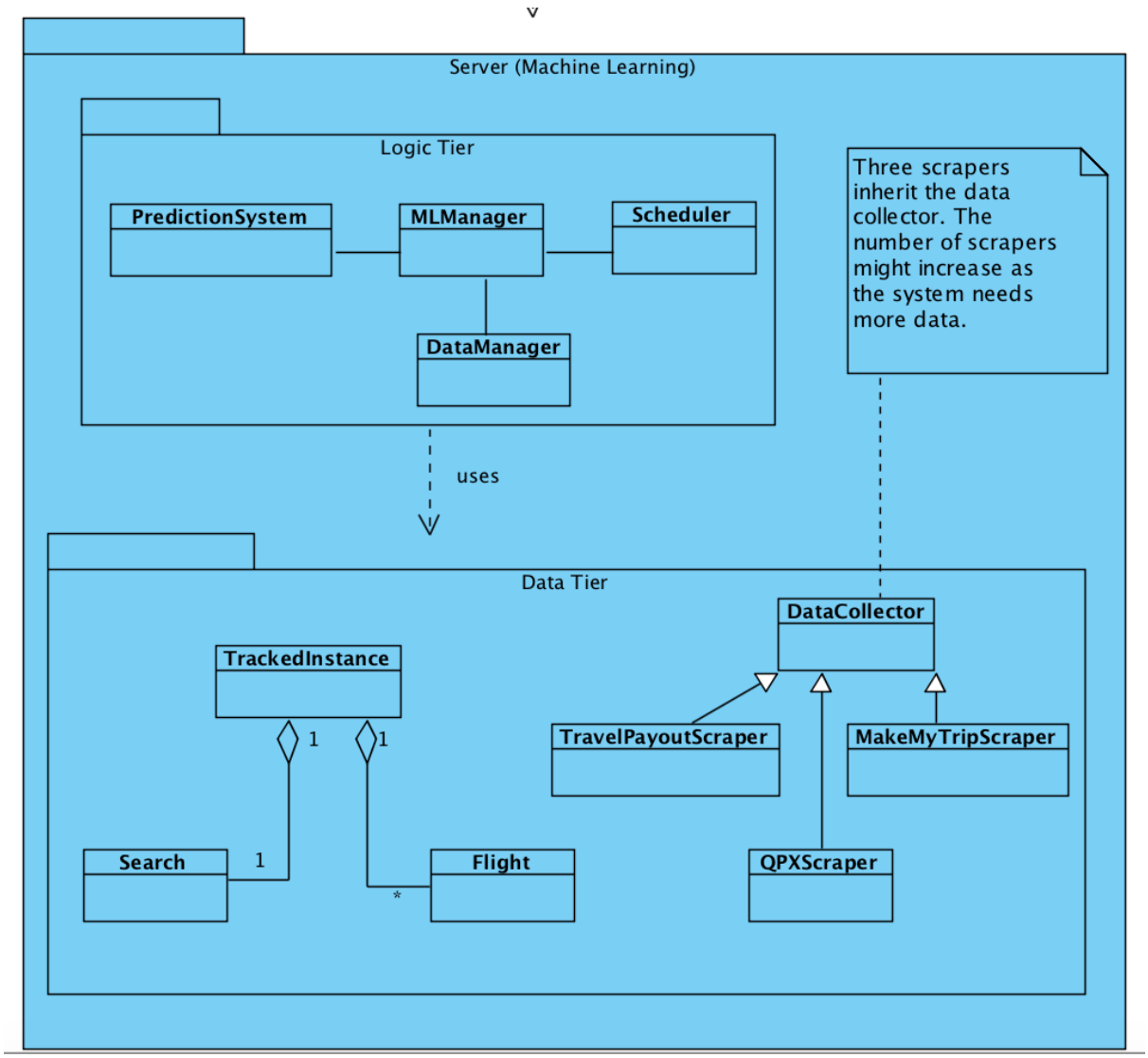

Figure 9 - Machine Learning Server Subsystem

Machine Learning Server is the second server which comes after Application Server. It is the server of the system for Machine Learning Training, Testing and Predicting. It manages and handles the predictions of the system which is requeste d by client. When there is a new data from scrapers or new requests from the user this server gets it and handles it. Then it responds to the application server to handle the request.

Machine Learning Server has two layers which are Logic Tier and Data Tier. Logic Tier is the one that handles all Machine Learning Prediction. It also controls when the prediction will be renewed. Logic Tier controls the existing data. Data Tier is responsible from data collection and it also stores Tracked Instances of the user.

4.3.1. Logic Tier

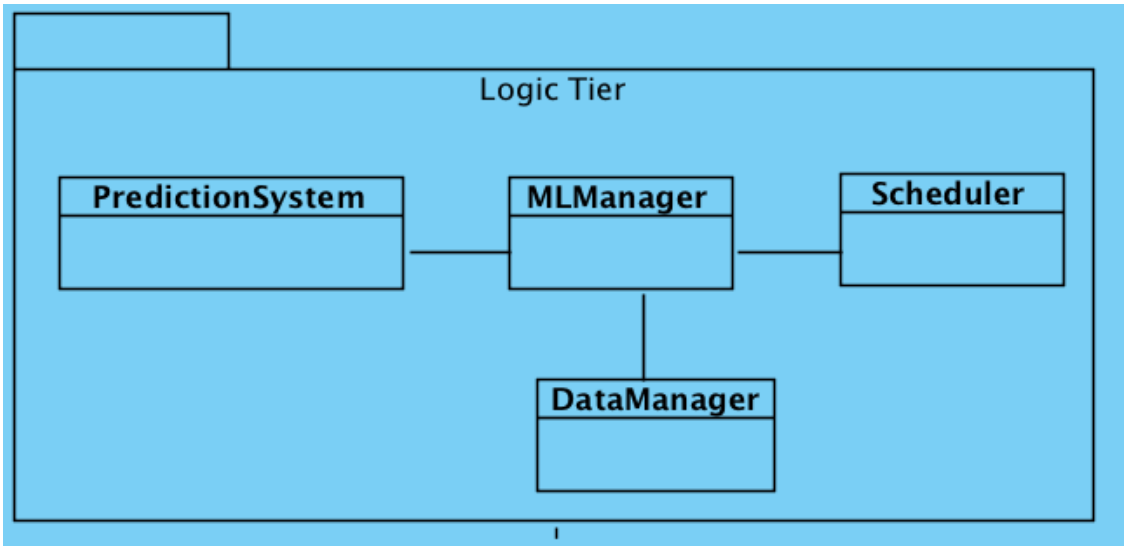

Figure 10 - Logic Tier Of Machine Learning Server

Logic Tier is the training, testing and predicting mechanism of the system.

PredictionSystem: Prediction Manager class is the connection point of Application Server and Machine Learning Server. It responds to the forecasts of the Machine Learning to the Application Server.

MLManager: Main class in this tier that does the all Machine Learning work such as making a model, training, testing and predicting.

DataManager: Data Manager class is data class that manages all database related tasks of Machine Learning. When there is a request or new data coming, it handles the updating of other parts.

Scheduler: Scheduler is the class that determines when the Machine Learning algorithms should work and predict. It controls the timing of the updates of the predicted data.

4.3.2. Data Tier

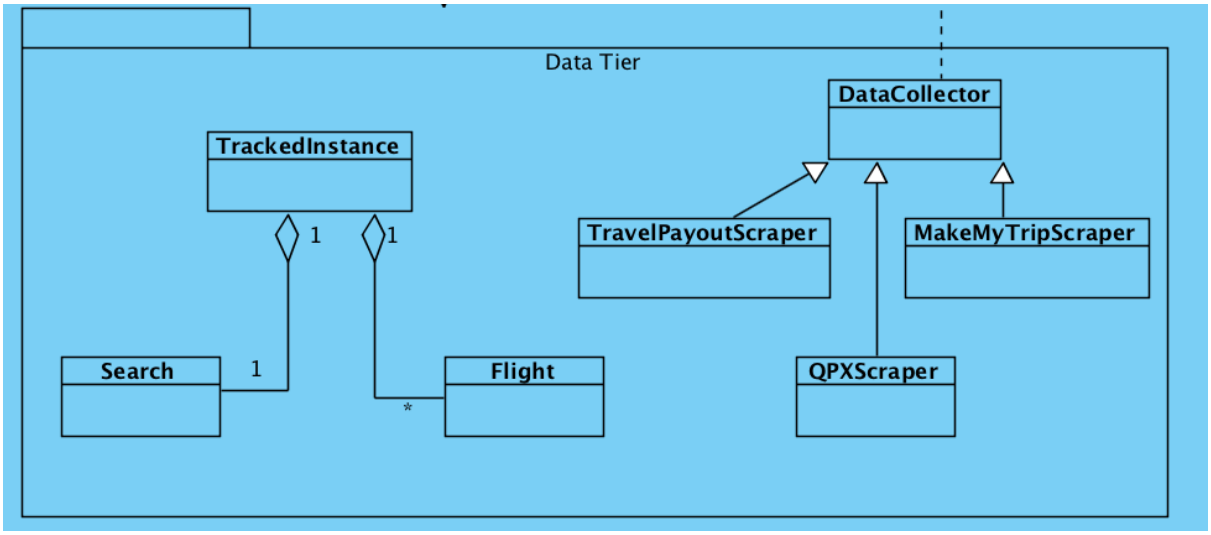

Figure 11 - Data Tier Of Machine Learning Server

Data Tier has two functionality. One is for keeping the tracked instances and second one is the control of the data collection for the predictions.

TrackedInstance: Tracked Instance class is data class that manages tracked flight or search instances. It contains tracked instance properties.

Search: It contains the Tracked Searches.

Flight: It contains Tracked Flights.

DataCollector: It makes the connection between Logic and Data Tier. Also it includes the data from Scraper.

TravelPayoutScraper: Class for managing the scraping from Travel Payout.

MakeMyTripScraper: Class for managing the scraping from Make My Trip.

QPXScraper: Class for managing the scraping from QPX API.

5. Glossary

Flight : An entity, which is created by using the user's input and has an origin, destination, departure date, carrier code, purchase date forecast and expected price.

Route : Result of the user search that consists a list of flights. It created with the chosen constraints.

Track : Following a flight or route including getting updates and notifications for the tracked flight or route.

Scraper: Scripts used for collecting data from various websites.

6. References

[1] "About Skyscanner". https://www.skyscanner.net/aboutskyscanner.aspx. [Accessed: Nov 05, 2017].

[2] "AirFareWatchDog". https://www.airfarewatchdog.com. [Accessed: Nov 01, 2017].

[3] "Expedia Customer Service". https://www.expedia.com/service. [Accessed: Nov 03, 2017].

[4] "TripAdvisor Help Center". https://www.tripadvisorsupport.com/hc/en-us. [Accessed: Oct 25, 2017].

[5] "Hopper About". http://www.hopper.com/corp/about.html. [Accessed: Oct 30, 2017].

[6] "FairFly Corporate Travel". http://www.fairfly.com/corporate-travel. [Accessed: Oct 14, 2017].

[7] "Kayak". https://www.kayak.com. [Accessed: Nov 02, 2017].

[8] "The OpenAPI Specification". https://github.com/OAI/OpenAPI-Specification. [Accessed: Dec 01, 2017].