



Bilkent University

Department of Computer Engineering

# Senior Design Project

*AeroCast*

## Final 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

Final Report

May 3, 2018

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

## **Index**

<b>Introduction</b>	<b>3</b>
<b>System Overview</b>	<b>4</b>
<b>AeroCast Algorithmic Design</b>	<b>5</b>
Predictive Models	5
DataCollection	5
Date Range and Price Prediction	6
<b>Final Architecture and Design</b>	<b>7</b>
Subsystem Decomposition	7
Overview	7
Subsystem Decompositions	7
Subsystem Service	10
Client	10
Application Server	11
Logic Tier	12
Data Tier	12
Prediction Server	13
Logic Tier	14
Data Tier	14
<b>Impact of Engineering Solutions</b>	<b>15</b>
Global Impact	15
Economic Impact	15
Social Impact	16
<b>Engineering Solutions and Contemporary Issues</b>	<b>16</b>
<b>Tools and Technologies Used</b>	<b>19</b>
Library Resources	21

APIs Used	21
<b>Appendix A User Manual</b>	<b>20</b>
Header	22
Login Pop-up	23
Register Pop-up	24
Forget Password Pop-up	25
Footer	25
Application	25
<b>Appendix B Class Diagrams</b>	<b>34</b>
<b>References</b>	<b>54</b>

## Table of Figures

Figure - 1: Prediction System Diagram	6
Figure - 2: The Subsystem Decomposition	10
Figure - 3: Client Subsystem	11
Figure - 4: Application Server	12
Figure - 5: Logic Tier	13
Figure - 6: Data Tier	13
Figure - 7: Prediction Server	14
Figure - 8: Logic Tier	15
Figure - 9: Data Tier	23
Figure - 10: Header Without Log-in	23
Figure - 11: Header With Log-in	24
Figure - 12: Login Pop-Up	25
Figure - 13: Register Pop-up	26
Figure - 14: Forget Password Pop-Up	26
Figure - 15: Footer	27
Figure - 16: Home Page	28
Figure - 17: Search Result Page	29
Figure - 18: Profile Page	30
Figure - 19: Tracked Routes & Flights Page	30
Figure - 20 Tracked Routes	31
Figure - 21 Tracked Flights	31
Figure - 22 Contact Page	32
Figure - 23 Feedback Page	33
Figure - 24 Notification Page	34
Figure - 25 AeroCast System Class	35
Figure - 26 Client Class Diagram	36
Figure - 27 Application Server Logic Tier	39
Figure - 28 Application Server Data Tier	43
Figure - 29 Prediction Server Logic Tier	47
Figure - 30 Prediction Server Data Tier	48

## **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.

## **2. System 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.

### 3. AeroCast Algorithmic Design

#### 3.1. Predictive Models

AeroCast core scrapes data from different Online Travelling Agency (OTA) sites and store them in database. For each route in dataset, Aerocast core uses predictive models to forecast best purchase date and price. It has two major parts: data collection, purchase date and price prediction. System overview is shown in Figure 1.

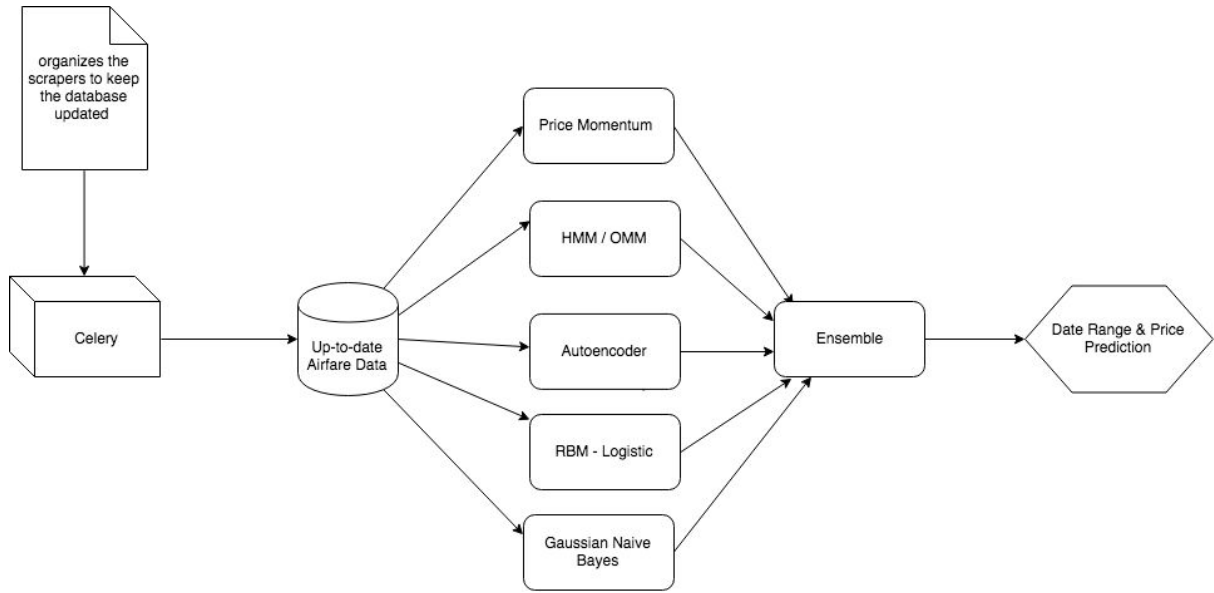


Figure 1: Prediction System Diagram

##### 3.1.1. Data Collection

The price data is collected mainly from 3 different sources, being makeMyTrip, Travelpayout and QPX Express API. The tasks that collect the data is organized in the celery task queue and these tasks are sent to the available workers as the workers become idle. Since the amount of data collected daily is high, some of the task mentioned above takes long time to complete. Therefore, the input of the tasks are divided into subparts where each list of origin and destination pairs is sent to a separate worker and executed concurrently. Although multiple tasks are executed at the same time, the parallel execution of the tasks are limited due to the restrictions of the APIs and web pages we use. For instance, the makeMyTrip does not allow us to access multiple webpages in less than a couple of seconds. Therefore, a task needs to request the content of a makeMyTrip page while another task waits for the components of the web page to load. Hence, the tasks are connected to the makeMyTrip one at a time but loaded and processes the components of the page concurrently. The data collected is

stored in the mongoDB and the destination, origin and departure date fields are indexed to enable fast access to the database in order to increase the response time.

### **3.1.2. Date Range and Price Prediction**

In this project, we developed a prediction system that estimates the purchase date having the lowest airfares. The prediction system makes predictions based on the previously obtained data and the data we collected from our scrapers. Before the predictive models are used by the prediction system, the prediction system extracts features such as the deviation, average or percentage increase of the prices on a specific route. After the features are extracted, the various predictive models are trained by using these features. The Scikit learn library, Tensorflow and Keras are used for the implementation and the training of the models used by the AeroCast system. The models used in the system are listed below.

- HMM / OMM: Hidden / Observable Markov Models that models the system as Markov processes with hidden / observable states.
- Autoencoder: An artificial neural network that obtains a representation of the data.
- Price Momentum: The rate of acceleration for the price change of the flights.
- Gaussian Naive Bayes: A probabilistic classifier that uses the Bayes theorem along with strong independence and Gaussian distribution assumption for the data.
- Restricted Boltzmann Machine: An artificial neural network that infers hidden features from the input features. RBM is used along with the Logistic Regression in a pipeline where the output of the RBM is fed to the Logistic Regression model.
- Logistic Regression: A classifier that uses logit model.

There are three different ways that are considered for the predictive models to make predictions for a specific purchase date in the future. The first approach listed below is used in the AeroCast system.

- The model can estimate the price for the next day based on the time-series data it has. Then, this new estimate can be added to the time-series data and the model can predict two days later by using the previous data and the new price estimate together. The model can continue making estimation for the next day and adding these estimations to the time-series data until it reaches the desired specific date in the future. After that point, the model can obtain the estimate for the desired date by using the time-series data generated from all previous estimations.



- Different models can be trained that predicts specific dates in the future. For instance, one model can predict the price for three days later whereas another model can predict the price for one week later by using the same time-series data. In this approach, each model is responsible from predicting the price of a purchase date having a specific time gap with the time-series data.
- The time-series data can be divided into subparts by using a sliding window. The price data in the first window can be used to make predictions for first purchase date in the future whereas the second window can be used to make predictions for the second purchase date in the future. In this case, the  $n$ th window will be used to make predictions for the purchase date, which is  $n$  days away from the current date.

## **4. Final Architecture and Design**

### **4.1. Subsystem Decomposition**

#### **4.1.1. Overview**

In the subsystem decomposition, the structure of the Aerocast system is described in detail. This part of the report analyzes the subsystems of our system and describes the services they provide in detail. The real structure of the project is much more complex to be directly modeled so the system is decomposed into two main subsystems.

#### **4.1.2. Subsystem Decompositions**

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.

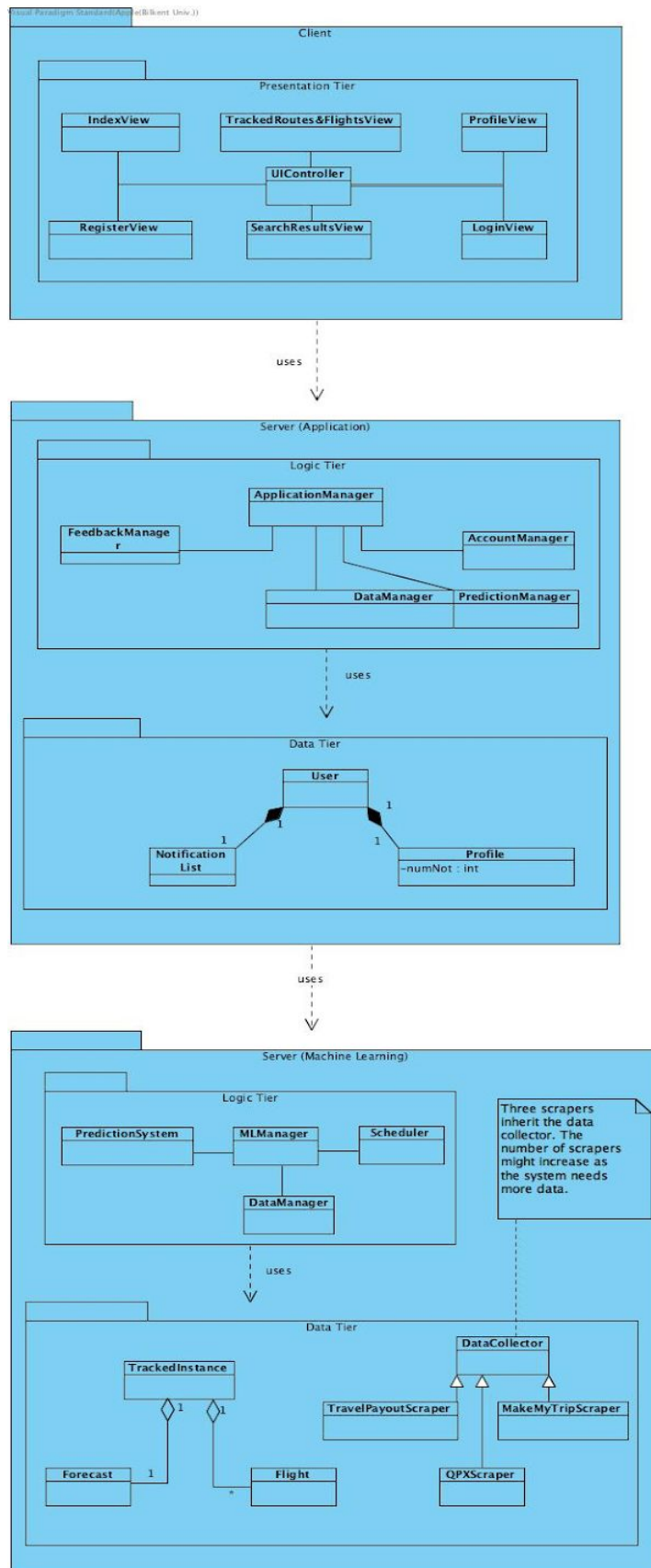


Figure 2: The Subsystem Decomposition

## 4.2. Subsystem Service

This chapter of our report explains the services implemented by the AeroCast system. Mainly, the services provided by the system can be divided into three groups which are the services related to the user account including the messages, feedbacks and notifications, the services related to the predictive models that we use and the services used for the user interface.

## 4.3. Client

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 RESTful 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.

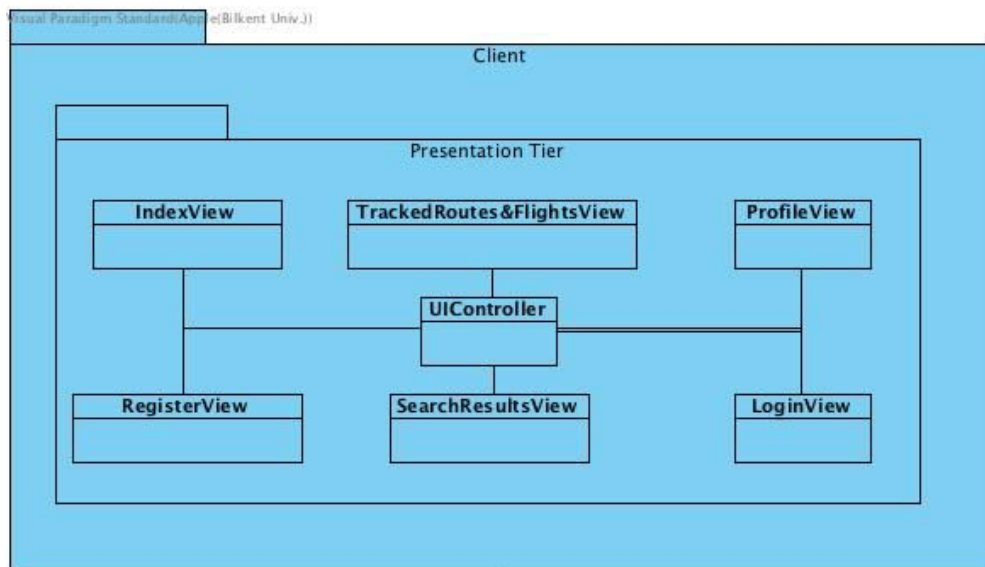


Figure 3: Client Subsystem

## 4.4. Application Server

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.

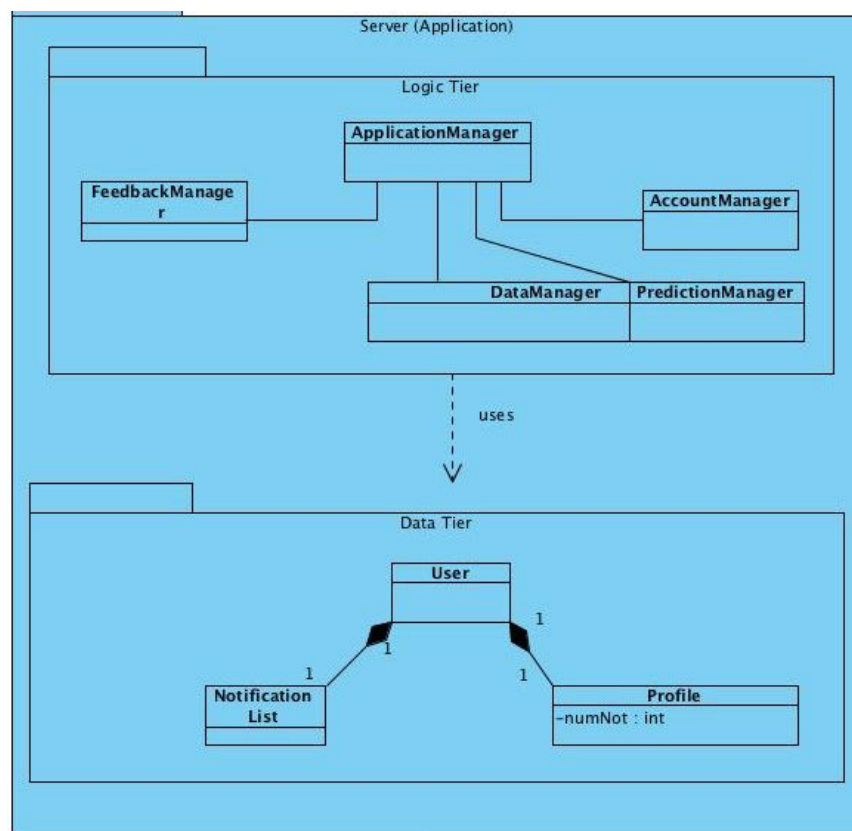


Figure 4: Application Server

#### 4.4.1. Logic Tier

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.

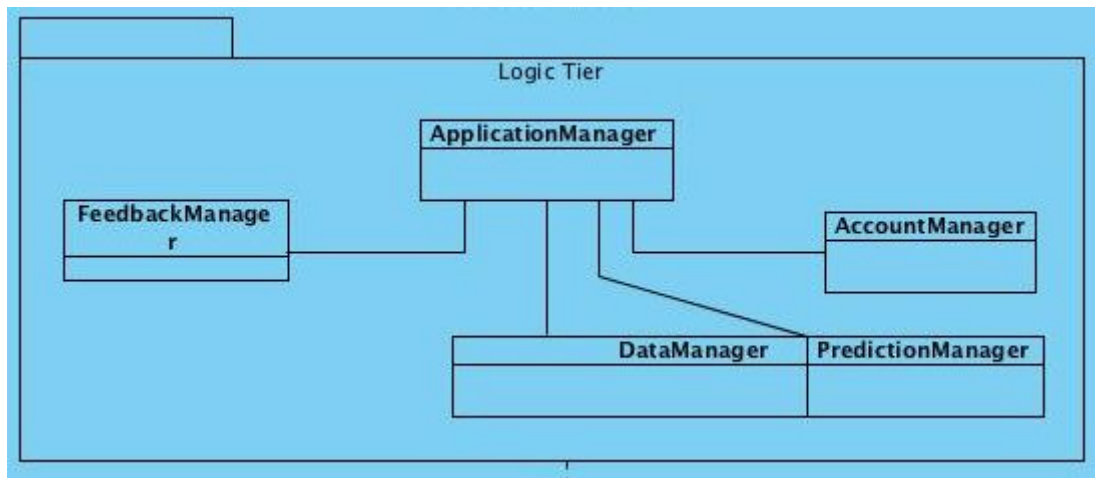


Figure 5: Logic Tier

#### 4.4.2. Data Tier

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

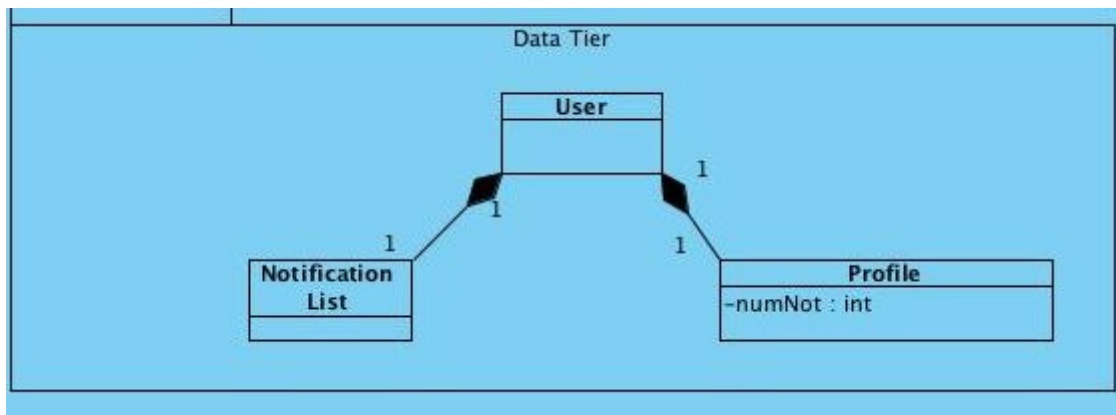


Figure 6: Data Tier

#### 4.5. Prediction Server

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 requested 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.

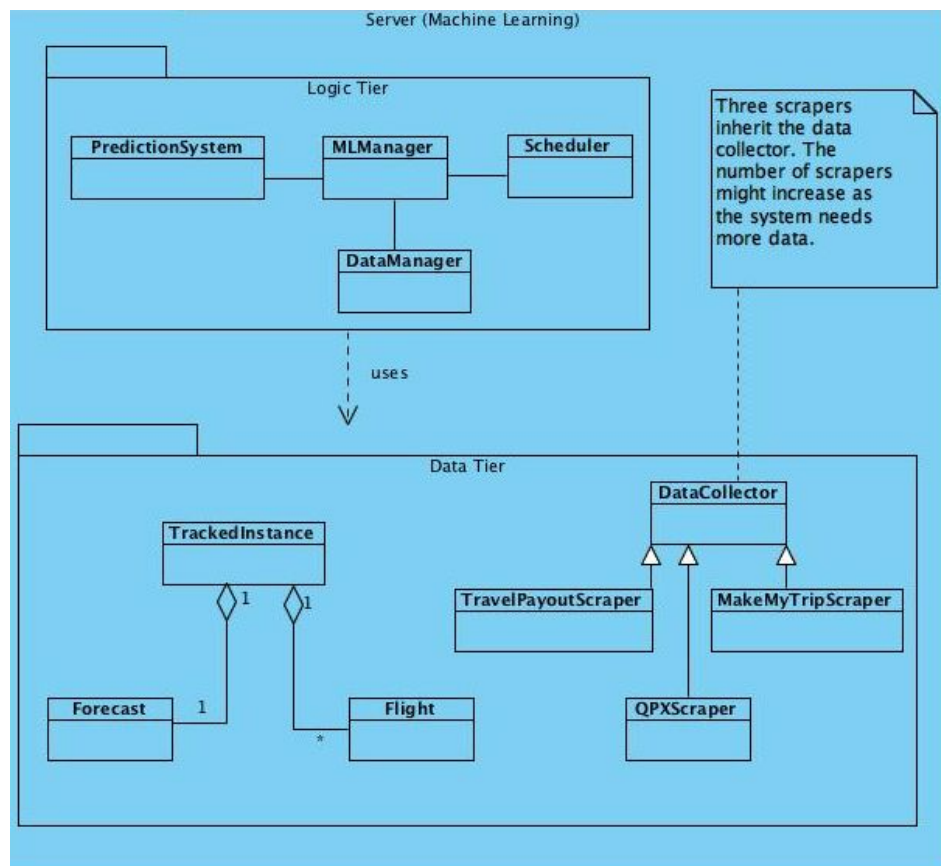


Figure 7: Prediction Server

#### 4.5.1. Logic Tier

Performing prediction, scheduling scrapers to collect data periodically, serialize & deserialize the flight data are among responsibilities of this particular package.

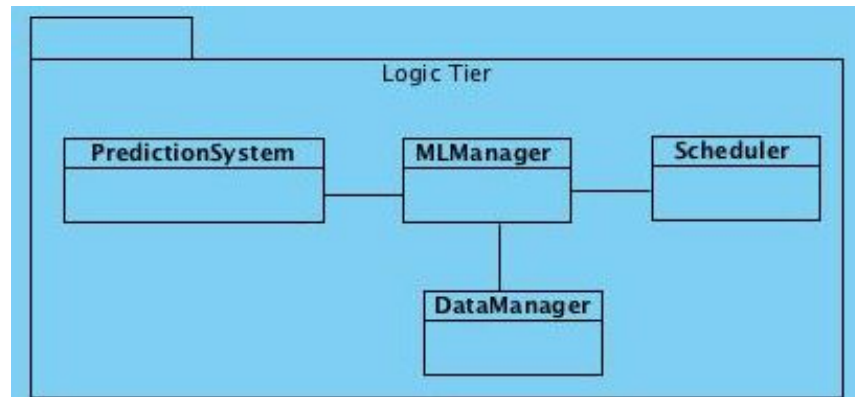


Figure 8: Logic Tier

#### 4.5.2. Data Tier

Data Package is responsible from the management and storage of the flight and forecast related data. It also uses its web scrapers to collect data from websites. All entities of the machine learning server is managed by this package.

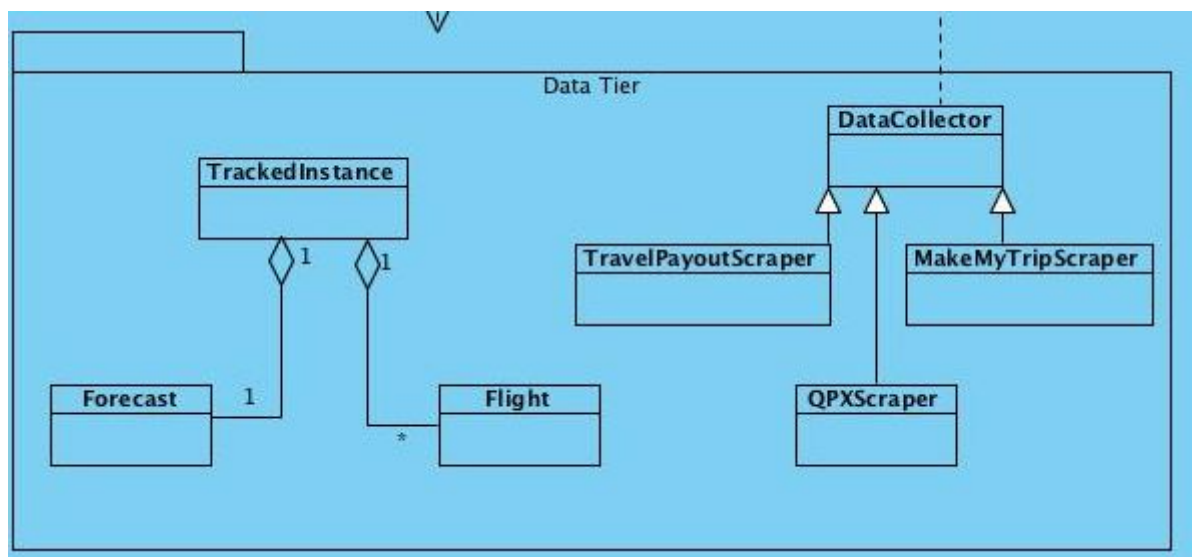


Figure 9: Data Tier

## 5. Impact of Engineering Solutions

The various impacts of the engineering solutions implemented in the AeroCast System are given as follows.



### **5.1. Global Impact**

The AeroCast system aims to provide an efficient way of finding the best purchase date for various flights along with the options of tracking the flight price, visualizing the price data as a graph and receiving notifications based on the changes on the flight prices. The system is meant to make predictions for different flight routes from various regions such as Europe, East Asia and Turkey. Therefore, AeroCast does not only aim for the users living in a specific region but it aims to provide predictions to various customers living in different parts of the world. It tries to use the flight data coming from various regions of the world and it also aims to make predictions for a large range of departure dates along with multiple flight options. Hence, not only airports and flights on some local routes but different airports and flights encapsulating routes in a global scale is targeted in the AeroCast system.

Also, AeroCast application targets a variety of users. Since air travel is widely used by the people in today's world, almost everyone in the world is a potential customer of the AeroCast system. By enabling people to find cheap flight tickets, AeroCast is trying to maximize the profit of the user. Therefore, AeroCast system is serving as a tool that enables more people to attain cheap flight tickets, which consequently can lead to an increase in the number of people using airplanes frequently. This increase in the number of people can also benefit the air industry in return. Hence, AeroCast system is trying to reach a variety of customers and it is trying to provide a service that encapsulates the flight routes in a global scale.

### **5.2. Economic Impact**

The main purpose of the AeroCast system is to enable people to spend less money on the airfares. Therefore, the AeroCast system is intended to have a vast economic impact on a large range of customers as well as the airline industry by changing how much money people spend on the flight tickets. AeroCast aims to increase the number of people who can buy cheap flight tickets by making accurate predictions based on the data collected from the web. AeroCast system does not sell flight tickets but it enables users to track the changes in the prices of the flight tickets and it also helps users to get more insight about how the prices of the flights change throughout the time. Besides, in the future, the AeroCast system might

direct the users to the relevant web pages selling the flight tickets that the users are interested in. In this way, AeroCast system can leave an economic impact on the industry by increasing the number of customers buying the flight tickets and it also can impact the customer's economy by providing them a way of attaining cheap flight tickets.

Since AeroCast is a prediction system for finding the cheap airfares that includes multiple airlines, airport, flight routes and regions, it is likely that AeroCast will have an impact on the economy of the users of the system as well as the airline company and travelling agency who sell flight tickets to the users. Therefore, economic impact of the AeroCast system can help improving the transportation industry altogether.

### **5.3. Social Impact**

AeroCast aims to provide cheap flight tickets to the users which can enable a large part of the population to obtain flight tickets more easily and the ability to buy cheap flight tickets can cause users of the system to fly more frequently than before. In return, this increase in the number of people who are travelling with airplanes can increase the interactions between the societies. Consequently, the impact of the AeroCast on transportation and economy can affect the social structure of the populations. Therefore, the AeroCast can have an indirect social impact by making the transportation more affordable for the people.

## **6. Engineering Solutions and Contemporary Issues**

In Aerocast, English is selected as the official service language due to the fact that English is the most commonly spoken language in the world. This allows us to target around 1.5 billion English speaking users worldwide [9]. In future, Turkish will also be supported for courtesy of the country of founders.

When users are in Aerocast front page, they should see a simple but yet informative layout to have an enjoyable and smooth user experience. One of the most important functionality of the system is search and users should be able to do real-time search in both airport and airline databases. This means that, each time user enters a letter, Aerocast should search the most relevant results from its database and return the results as a list in a split second. To solve this problem, we have used an industrial standard software named

Elasticsearch which does this process efficiently. It stores an index table to return the results fast and reliable.

Aerocast platform is a web service and we target majority of smart devices e.g. smartphone, tablet, mac, pc, etc. in the market. This means platform should have a cross-platform support. To achieve this goal we use novel front-end tool namely ReactJS and Bootstrap in order to implement the face of the platform. ReactJS and Bootstrap ensure functionality and front-end components work as promised respectively.

As the time is a precious resource and people's attention is shortened further, Aerocast has to return responses to all of the requests of its users in at most two seconds. To achieve such speed in service time, only necessary content is updated instead of loading same front-end components multiple time, each server handles requests in parallel; it is implemented as efficient as possible, task scheduling is used to utilize computational resources and data is stored in a hierarchical database to reduce the fetching time. Here is the brief explanation for each solution:

- To decrease the loading time for pages, we have used tools that only loads specific parts of front-end pages where there is a change. Consequently, response time is shortened by reducing the amount of data required to transfer from back-end to front-end. Name of the library is ReactJS. It is build to develop single-page web applications and maintained by Facebook.
- Handling requests in parallel works as follows. In Application Server where users' accounts are managed, we use multiprocessing features of Spring framework. In Prediction Server where data is scraped from web and predictions are made, we use uWSGI server multi-thread feature. The reason being is that Python server has GIL (Global Interpreter Lock) and it prevents usage of multi cores in the system. When necessary, uWSGI runs multiple Python interpreters which can run on different cores so we achieve multi-processing performance.
- Writing efficient code means that we try to use libraries and tools that are compliant with industrial standards. As a result we get quite efficient code running for the Aerocast platform.
- Task scheduling is used to utilize computational resources. In order to keep Aerocast platform updated and as accessible as possible, it has to scrape new data from different online travelling agency websites and respond to all the requests sent by

users. As a solution to this problem, we schedule tasks such as scraping data from web and updating prediction models to less busy moments during the day so we utilize computational resources. Celery task management tool is used for scheduling the task to the workers.

- Finally, all data is stored in a hierarchical database software namely MongoDB. Since there is no need for complex database queries in Aerocast platform, we prefer to use a NoSQL database. Relational databases perform joins for related tables, this creates an unnecessary overhead for platforms that needs simple database operations. That is why we use MongoDB and it contributes to Aerocast to meet its response time requirements.

Due to broad range of potential users in online travelling industry, it is important to maintain the service quality no matter how many customer using it. As the number of online users increase, the server would have to respond millions of requests in a matter of seconds and manage huge amount of data. To meet with such needs, AeroCast platform should be scalable by design. Therefore, Aerocast has a modular design meaning that prediction and accounting are separated from each other into different machines and encapsulated with Docker containers. Likewise, it uses NoSQL database that is widely used for scalable systems. Moreover, Docker allows us to create exact copy of the existing servers in either the same machine or different machines due to excessive load of requests. As a result, we can increase the number of servers for each system to reply all the users. Due to the limited project budget, we have had limited computation power so we have only been actively using the parallel execution of requests. In future, system can be moved to computationally powerful machines so Docker can be used to increase scalability.

While suggesting anything to its users, Aerocast has to consider many things in its mind. As the number of online travelling agencies supported increase, the amount of data fetch increases. In order to manage the data fast and efficiently, data collection interface of Aerocast should scrape minimum amount of data while achieving highest possible coverage and precision. To solve this problem, data crawlers fetch only necessary amount of data per route to make an accurate prediction. It also uses previously fetched data to make prediction more accurate.

Aerocast aims to make price and purchase date range predictions for specific origin, destination and departure date values. While making these predictions, Aerocast system needs to consider multiple constraints such as the carriers, departure dates, origin and destination airports the user choose. In addition to the wide range of constraints, the scarcity of data on some routes, the missing values in the data and the discrete nature of the data results in the difficulty of making accurate predictions. Furthermore, since the Aerocast system needs to make estimations for multiple dates in future and it needs to choose the date range that has the minimum prices, the models that the system uses has to be sophisticated enough to handle multi date predictions with time gaps in between.

Aerocast system implements several solutions to handle the above cases. Rather than relying on a single model, It uses multiple models to make predictions for the specified constraints of the users and ensembles the results of the multiple models to get the final result. Also, it feeds the outputs of some of the models to other models as input and uses the final results of this pipeline of multi-models as the estimation. It uses linear interpolation to fill the missing values on the data and to make some discrete inputs continuous. Also, the system daily collects data and updates the predictions of the models based on the newly collected data to decrease the effect of the scarcity of data on some routes. Finally, for each request of the user Aerocast makes a series of prediction where each prediction corresponds to a purchase date in the future and the system chooses the purchase date having the lowest price among all these predictions by using the methods mentioned in the Algorithmic design chapter of the report.

## 7. Tools and Technologies Used

- **WebStorm:** A JavaScript IDE. Since we used ‘React’ JavaScript Library to build interface of the web site, we needed a JavaScript IDE. WebStorm was the most suitable choice in terms of git connection, user-friendliness, efficient coding performance.
- **NodeJS:** An open source server environment. We used it as a front-end server. It is very compatible with React library and many libraries(modules).
- **Git:** A version control system that we used in the AeroCast system. We have two different Git repository one for the front-end development (JavaScript) and one for the backend development (Java). Both local coding environments and server coding environments were connected to Git repositories, so that we can synchronize our jobs.

- **ElasticSearch:** A distributed RESTful search engine capable of performing search operations with high speed on a large dataset. In AeroCast system, elasticsearch is used to enable the users to query the airport codes, names and locations that are indexed and stored in the elasticsearch database system.
- **Swagger (OpenAPI v2.0):** An open API specification for designing the RESTful API between the client and the application Server.
- **Nginx:** A proxy server that load balances the requests and distributes them among the servers.
- **ApacheBench:** A command line tool that enables the developers to measure the performance of the servers. It can test how many requests the server can handle concurrently. ApacheBench is used for testing the performance of the Application and Prediction servers of the system.
- **Spring:** An application framework that handles the requests, manages the user sessions and provides access control for the database. In this project, Spring is used to handle the user related request and responses. It keeps the session and profile information of the user and manages the transactions.
- **Apache Maven:** A dependency management tool that injects the necessary dependencies of the project.
- **MongoDB:** A NoSQL database used by the prediction and application servers of the system. AeroCast system has two separate mongoDB collections for storing the application and prediction related data. The mongoDB stores the user profile and session information as well as the airfare predictions and the flight data that is scraped from the web.
- **Docker:** An operating-system-level virtualization containerization technology used by developers in order to overcome challenges for cross operating system dependencies in the development stack. It is used for making AeroCast scalable.
- **Flask:** A light-weight framework used for implementing Python applications that can handle RESTful requests. Due to its lightweight nature, Flask applications are simple but highly extensible. In AeroCast system, Flask is used for handling the request and responses that are relevant to the prediction system.

- **uWSGI:** A self-hosted server that manages the Flask application and handles the connection between the Nginx and Flask. It can increase the number of server instances running on the machine as needed.
- **Celery:** A task management tool. It schedules the tasks for execution, adds the tasks to the task queue and runs the tasks as the workers become available. Celery uses multiple workers for concurrent execution of the tasks. It uses a message broker to receive the task from the task scheduler and it adds the received task to the task queue in the order that they arrive. In AeroCast system, rabbitMQ is used as the message broker that sends the tasks to the Celery task queue.
- **Selenium:** A testing tool mostly used for testing the web applications. It can load the web pages and perform operations on the components of the web pages. In the AeroCast system, the selenium is used to scrape the flight data from the web pages by loading the page and getting the content. Selenium uses the PhantomJS webdriver to connect to the web servers. The flight data obtained from these web pages are stored in the mongoDB database.

### 7.1. Library Resources

- **React:** A single page application and Javascript library used for designing web user interfaces.
- **Bootstrap:** A responsive, front-end component library. We used it to make the website responsive.
- **PyMongo:** A PPython library used for connecting the Python application to the mongoDB.
- **Sci-kit Learn:** A PPython machine learning library used for making predictions for the airfares.
- **Tensorflow:** A low level Machine Learning library written in Python and used to design and build statistical models.
- **Keras:** A high level Machine Learning library written in Python. It is mainly used to develop Deep Learning models.

## 7.2. APIs Used

- **QPX Express:** Google's API for accessing the flight data.
- **TravelPayouts:** An online travelling agency site, where an API is provided for accessing travel related data.
- **Google Maps Javascript API:** Google Map API to place Google Map to the Contact Page.

## 8. Appendix A User Manual

### 8.1. Header

Header will stay in the top of the page in every page of the website. The fixed parts are the "AEROCAST" and the currency dropdown. When the user clicked "AEROCAST", page goes to the main page. If the user clicked the currency, he/she can choose another (TL or EURO) currency from the dropdown list. Rest of the Header changes according to LogIn situation.

#### 8.1.1. Not Logged In Header



Figure 10: Header Without Log-in

When the user is not logged in, there are two components besides the fixed ones: Sign Up and Sign In. Sign Up opens a modal for Registration and Sign In opens a modal for Login. User clicks the "Sign Up" button to register the website through the register modal, and clicks the "Sign In" button to login the website through the login modal.

#### 8.1.2. Logged In Header

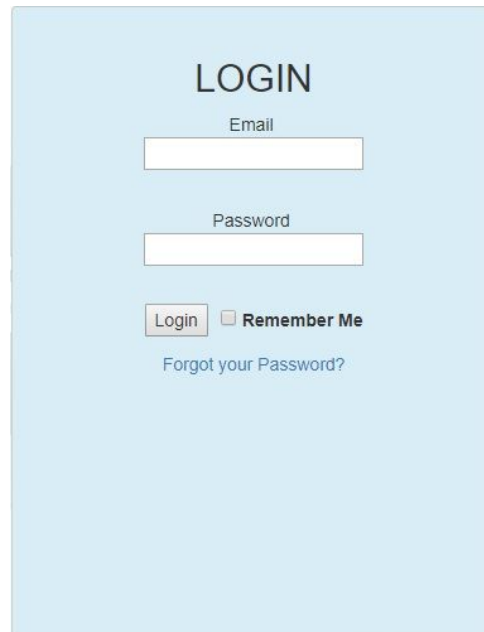


Figure 11: Header With Log-in



When the user logged in, instead of Sign Up and Sign In buttons, there are “Notifications” and user dropdowns. The user can click notifications to see them or they can choose to go to “Profile”, “Tracked Routes&Flights” from user dropdown. Also the user can logout from the user dropdown.

## 8.2. Login Pop-up



LOGIN

Email

Password

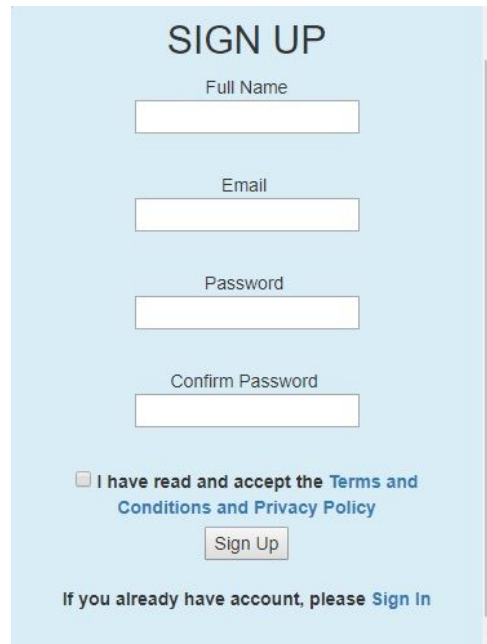
Login ☐ Remember Me

[Forgot your Password?](#)

Figure 12: Login Pop-Up

Login popup is accesible through the “Sign In” button. By entering the email and password the user can login. Also there is a remember me check box which keeps the user logged in unless user clears the browser cache. If the user forgets the password, he/she can click the “Forgot your Password?” and go to the forgot password popup.

### 8.3. Register Pop-up



**SIGN UP**

Full Name

Email

Password

Confirm Password

☐ I have read and accept the [Terms and Conditions and Privacy Policy](#)

If you already have account, please [Sign In](#)

Figure 13: Register Pop-up

Register popup is accessible from the “Sign Up” button. After entering the full name, email, password and re-password, the user can open an account for himself/herself. Email has to be an email type and no other user uses it and also passwords need to match. Before submitting user has to check the box of Terms and Conditions and Privacy Policy. Also register popup gives the opportunity to directly open the Login popup at the very end. After sign up user gets some link to activate his/her account.

#### 8.4. Forget Password Pop-up

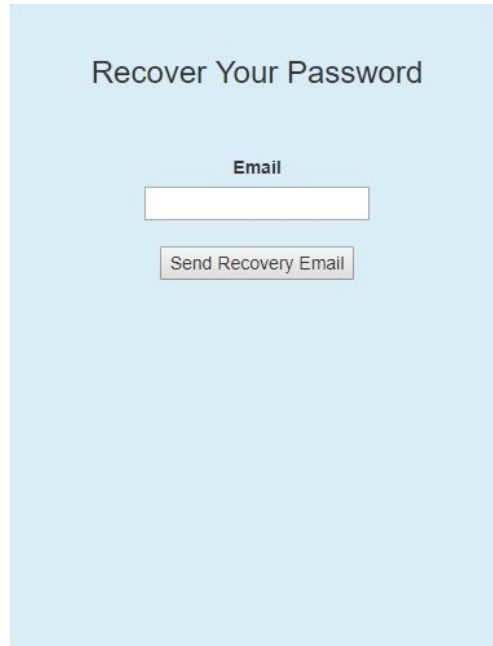
A light blue rectangular pop-up window titled "Recover Your Password". Inside the window, the word "Email" is centered above a white text input field. Below the input field is a button with the text "Send Recovery Email".

Figure 14: Forget Password Pop-up

Forgot password popup is simple and only takes one input, email. User enters the email and clicks the “Send Recovery Email” button to send new password to his email.

#### 8.5. Footer



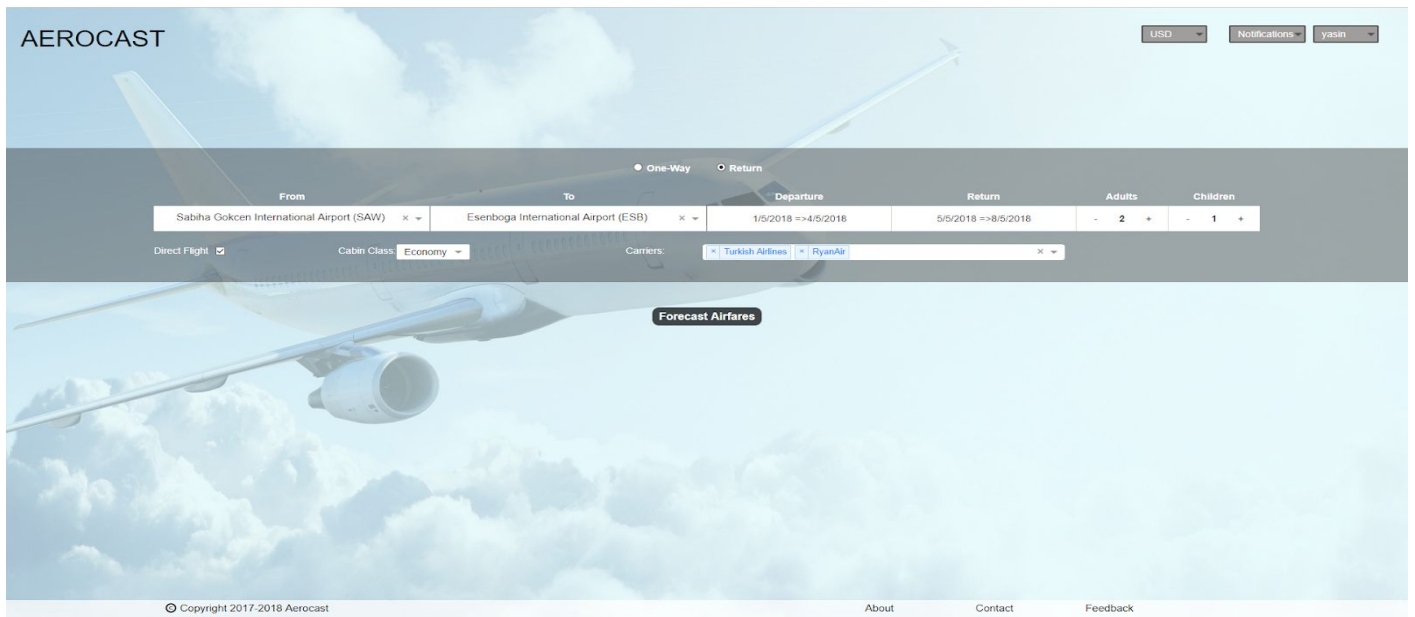
Figure 15: Footer

Footer is at the bottom of all pages. There are three links: “About”, “Contact”, “Feedback”. The user can go to project website in a new tab from About. “Contact” and “Feedback” will turn the page into Contact Information page and Feedback page.

## 8.6. Application

This is the whole webpage except header and footer. Rest of the pages are included in the application.

### 8.6.1. Home Page



The screenshot displays the AEROCAST website's home page, which features a flight search interface. The background is a light blue sky with white clouds and a faint image of an airplane. The search form is centered and includes the following fields and options:

- From:** Sabiha Gokcen International Airport (SAW)
- To:** Esenboga International Airport (ESB)
- Departure:** 1/5/2018 => 4/5/2018
- Return:** 5/5/2018 => 8/5/2018
- Adults:** 2
- Children:** 1
- Direct Flight:** ☒
- Cabin Class:** Economy
- Carriers:** Turkish Airlines, RyanAir
- Buttons:** One-Way (selected), Return, Forecast Airfares
- Top Right:** USD, Notifications, yasin
- Footer:** Copyright 2017-2018 Aerocast, About, Contact, Feedback

Figure 16: Home Page

This page with the functionality of “searching” is the main page of the website. This is the only page that can be accessed through web link. <http://domain/anything> will direct user to this page. After choosing “One-Way” or “Return” for the flight(Default is One-Way), user can choose Origin Airport and dates; Destination Airport and dates; number of adult and child; whether it is a direct flight or not, cabin class and Carriers(Empty means all carriers). To select date user can click on the component and choose starting point and then ending point. To choose one day, just click on the same day for starting and ending. Pressing “Forecast Airfares” will lead the page to result of the search. As constraints, the user has to choose both airports and is not allowed to choose same airports for origin and destination. Also start and end of the return dates needs to be after the start and end of the departure dates. Otherwise, the user will be warned by browser.

### 8.6.2. Search Result Page

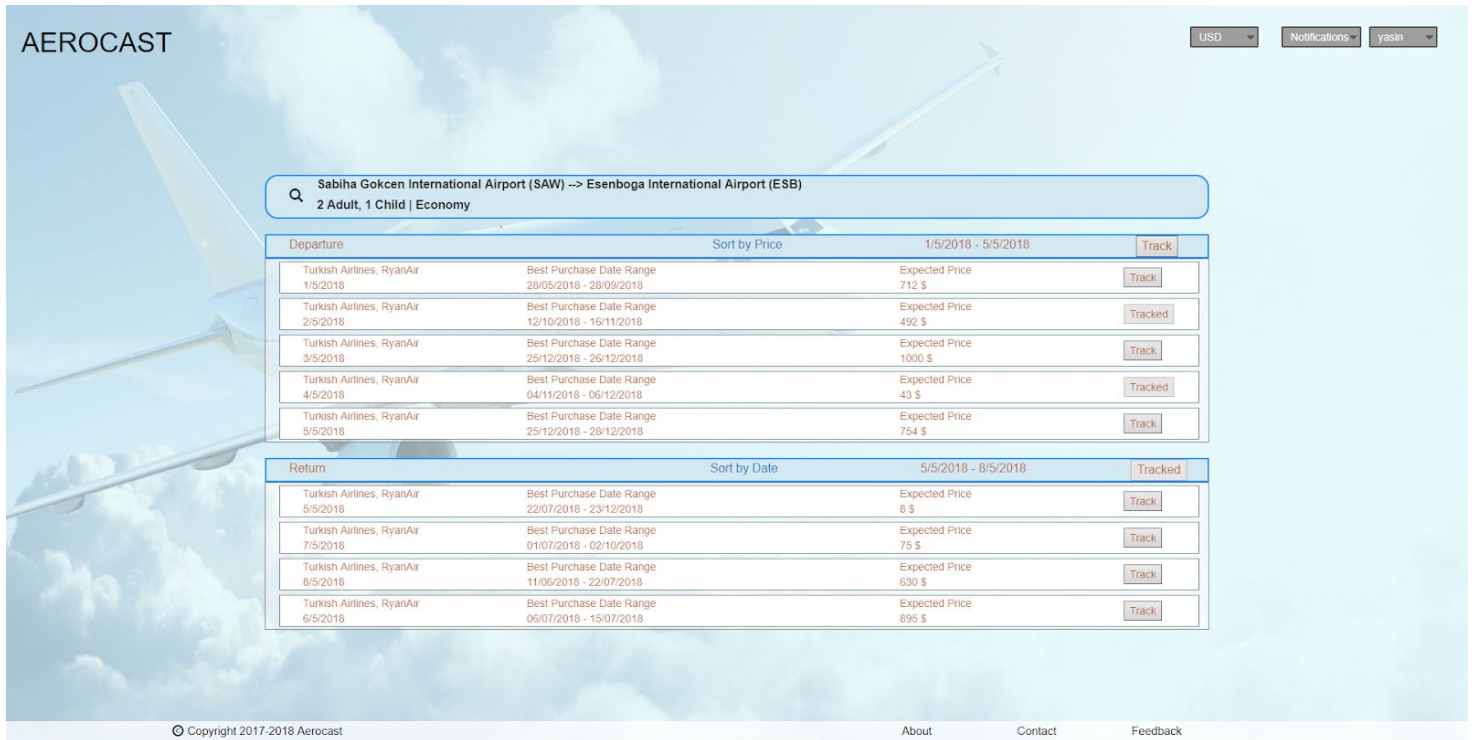


Figure 17: Search Result Page

This page returns as the results of the search. At the top, there are information on airports, number of passengers and the cabin class.

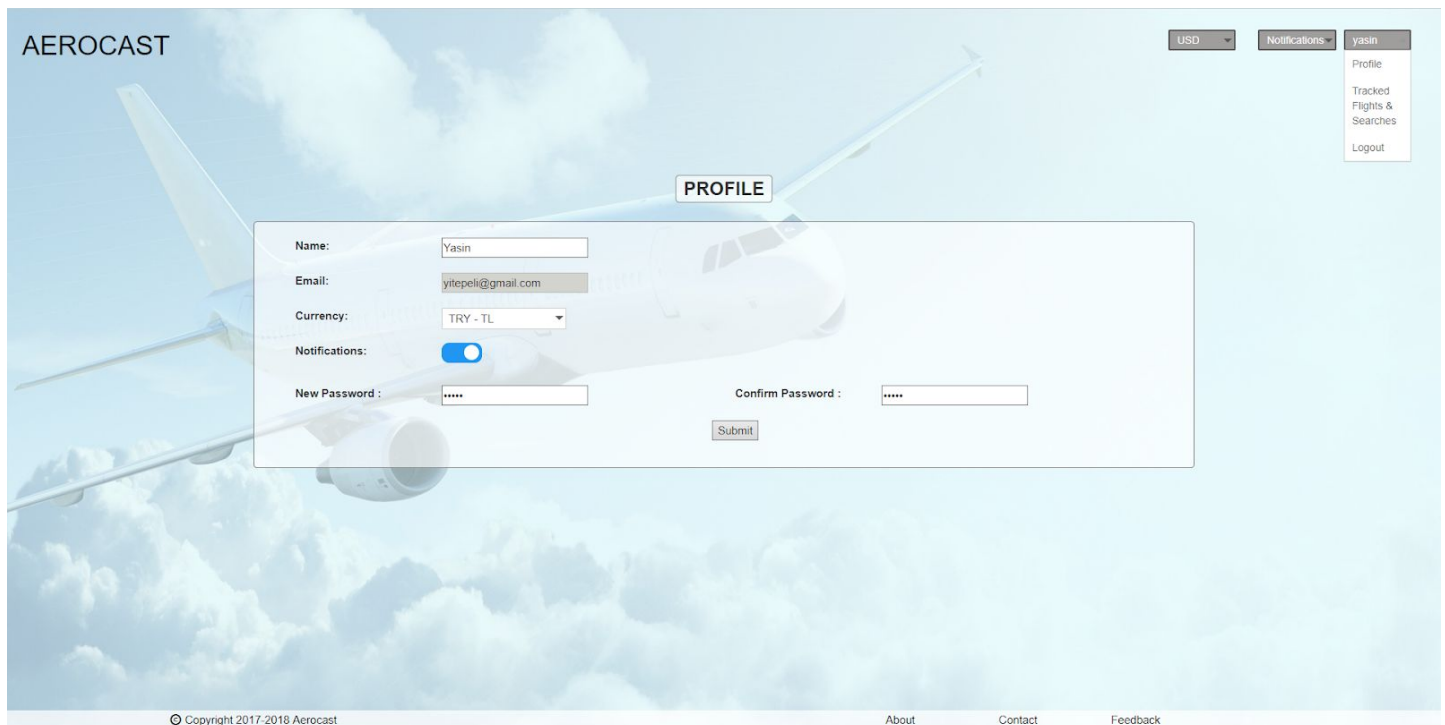
The first part is always Departure part of the flight. If the return has been chosen, then there will be second identical part for the return flights. There is one predicted result for each day.

Initially, all results are ordered by date, the user can change it to ordered by the expected price by clicking “Sort by price” (The text turns to “Sort by date”). If the user wants to go with date ordering, he/she also can click “Sort by Date” to order it back to by date. Also there is the date range and “Track” button on the title of the departure or return. The user can track the whole search (Departure and Return are separated.) by clicking the “Track button” at the title.

When the user clicks anywhere of a flight, the user can see the graph about that flight and prediction. If there is no data on that flight, there will be a warning saying that “No data to draw graph”

Inside the result parts (Flight for each day), user can found flight's exact departure date, the carriers, best purchase date range, the expected price. Also the user can track the one-day flight by clicking the “Track” button near the flight. If user is not logged in, when he/she pressed the track button, the Login pop up will appear.

### 8.6.3 Profile Page



The screenshot displays the AEROCAST website's profile page. The background features a light blue sky with white clouds and a faint image of a white airplane. In the top left corner, the word "AEROCAST" is written in a bold, sans-serif font. In the top right corner, there are three buttons: "USD" with a dropdown arrow, "Notifications" with a dropdown arrow, and a user profile button labeled "yasin". The "yasin" button has a dropdown menu with four options: "Profile", "Tracked Flights & Searches", and "Logout". Centered in the upper part of the page is a white rectangular box with a thin border, titled "PROFILE" in a small, bold, sans-serif font. Inside this box, there are several form fields: "Name:" with a text input containing "Yasin"; "Email:" with a text input containing "yitepeli@gmail.com"; "Currency:" with a dropdown menu showing "TRY - TL"; "Notifications:" with a blue toggle switch that is turned on; "New Password:" with a text input containing six asterisks; and "Confirm Password:" with a text input containing six asterisks. Below these fields is a "Submit" button. At the bottom of the page, there is a footer with four links: "© Copyright 2017-2018 Aerocast", "About", "Contact", and "Feedback".

Figure 18: Profile Page

Profile page is accessed by clicking the “Profile” at the right top corner. It includes the settings of the user such as Name, Email, Currency, Notification Decision. The user change these except the email and also can change the password. User can leave the password empty does not want to change it. When submit button is clicked, profile will be updated.

## 8.6.4 Tracked Routes & Flights

Routes / Flights			
THU YAV	RyanAir 3/5/2018	Best Purchase Date Range 05/06/2018 / 22/10/2018	Expected Price 128 \$
THU YAV	RyanAir 4/5/2018	Best Purchase Date Range 18/05/2018 / 25/08/2018	Expected Price 593 \$

Figure 19: Tracked Routes & Flights Page

Tracked Routes and Flights consist of 2 sections: Tracked Routes and Tracked Flights. The user can change the section by clicking the “Routes” or “Flights” button at the top of the page.

## 8.6.5 Tracked Routes

Routes / Flights			
THU YAV	RyanAir 3/5/2018	Best Purchase Date Range 05/12/2018 / 28/12/2018	Expected Price 656 \$
THU YAV	RyanAir 4/5/2018	Best Purchase Date Range 07/05/2018 / 27/05/2018	Expected Price 697 \$

Figure - 20 Tracked Routes

Tracked Routes will give the user the searches (either Departure or Return) which the user tracked before. One route consists of flights. User can untrack the Tracked Route by clicking the button.

When the user clicks anywhere of a flight, the user can see the graph about that flight and prediction. If there is no data on that flight, there will be a warning saying that “No data to draw graph”.

### 8.6.6 Tracked Flights

Routes / Flights				
THU YAV	RyanAir 4/5/2018	Best Purchase Date Range 24/04/2018 - 24/12/2018	Expected Price 200 \$	<a href="#">Untrack</a>
YAV THU	RyanAir 9/5/2018	Best Purchase Date Range 02/11/2018 - 25/11/2018	Expected Price 799 \$	<a href="#">Untrack</a>
YAV THU	RyanAir 10/5/2018	Best Purchase Date Range 25/12/2018 - 27/12/2018	Expected Price 129 \$	<a href="#">Untrack</a>

Figure - 21 Tracked Flights

Tracked Routes will give the user the specific flights which the user tracked before. User can untrack the Tracked Flight by clicking the button.

When the user clicks anywhere of a flight, the user can see the graph about that flight and prediction. If there is no data on that flight, there will be a warning saying that “No data to draw graph”

### 8.7 Contact

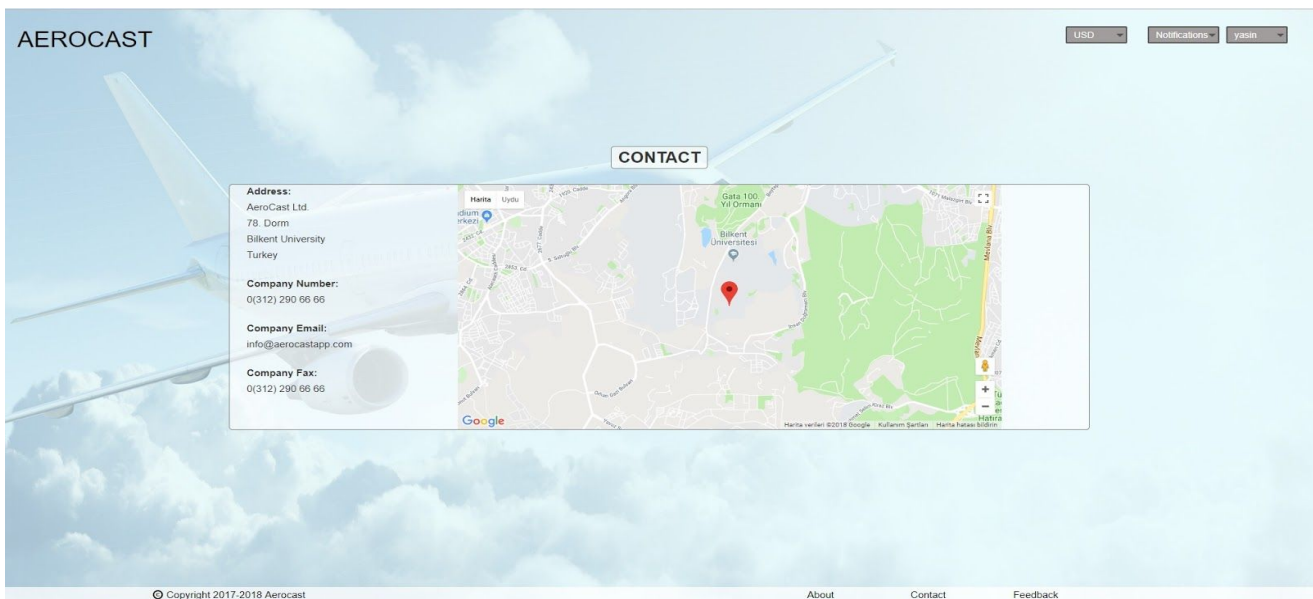
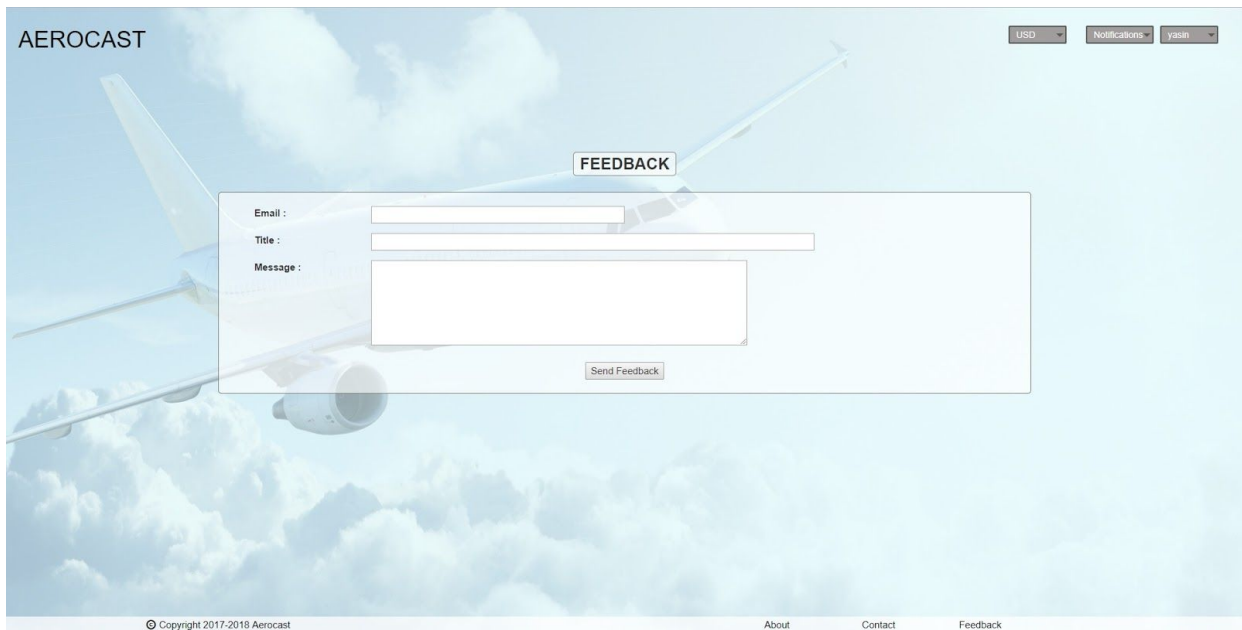


Figure - 22 Contact Page

Contact page includes the company address, phone, mail and fax information. Besides these, there is also a Google Map which shows the location of the company.



## 8.8 Feedback



The screenshot displays the AEROCAST website's feedback interface. The background features a light blue sky with white clouds and a faint image of a white commercial airplane. In the top left corner, the word "AEROCAST" is displayed. In the top right corner, there are three small, dark rectangular buttons labeled "USD", "Notifications", and "yasin". Centered in the upper half of the page is a white rectangular box with a thin black border. At the top of this box, the word "FEEDBACK" is written in a small, black, sans-serif font. Below this title, the form contains three input fields: "Email :" followed by a single-line text box, "Title :" followed by a single-line text box, and "Message :" followed by a larger, multi-line text box. At the bottom right of the form box is a small button labeled "Send Feedback". The footer of the page is a light blue bar containing the text "© Copyright 2017-2018 Aerocast" on the left and three links labeled "About", "Contact", and "Feedback" on the right.

Figure - 23 Feedback Page

The user can send a feedback through “Feedback” page. It is enough to fill the email, title and message parts and click the submit button.

## 8.9 Notification

**AEROCAST**

USD See All Notifications Yasin

**Notifications**

Prediction for your tracked flight from THU to YAV on 4/5/2018 has changed. The expected best price for your flight is 809\$ between 20/08/2018 and 28/10/2018	Delete
Prediction for your tracked flight from YAV to THU on 9/5/2018 has changed. The expected best price for your flight is 321\$ between 22/11/2018 and 27/11/2018	Delete
Prediction for your tracked flight from YAV to THU on 10/5/2018 has changed. The expected best price for your flight is 606\$ between 23/11/2018 and 25/12/2018	Delete
Prediction for your tracked flight from THU to YAV on 2/5/2018 has changed. The expected best price for your flight is 94\$ between 07/06/2018 and 11/09/2018	Delete
Prediction for your tracked route from YAV to THU has changed. The expected best prices for your route are the following: For the flight on 8/5/2018, best price is 676\$ between 16/06/2018 and 16/11/2018 For the flight on 9/5/2018, best price is 189\$ between 19/11/2018 and 19/11/2018 For the flight on 10/5/2018, best price is 118\$ between 24/04/2018 and 26/09/2018	Delete
Prediction for your tracked route from THU to YAV has changed. The expected best prices for your route are the following:	Delete

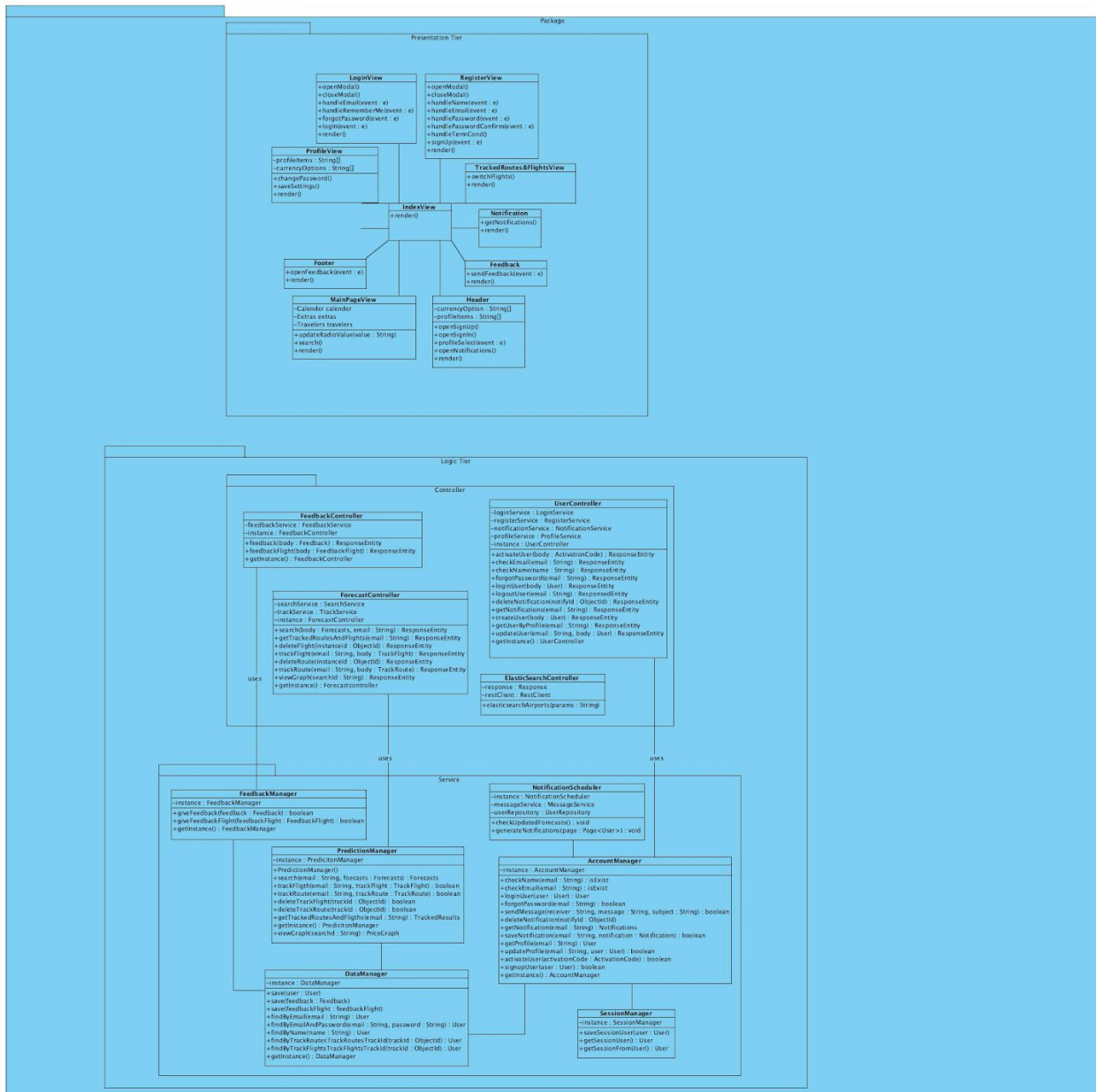
© Copyright 2017-2018 Aerocast About Contact Feedback

Figure - 24 Notification Page

Notification page lists notifications of tracked flights and routes. User will see the list as the new one on the top, and can delete the notification by clicking the “Delete” button on the right of the notification. User can also see the update of flights in the tracked route by clicking on the route notification. It will expand the notification and shows flights update of the route.

## 9. Appendix B Class Diagrams

The AeroCast system consists of Client, Application Server and Prediction Server.  
The complete class diagram for the AeroCast system is provided below.



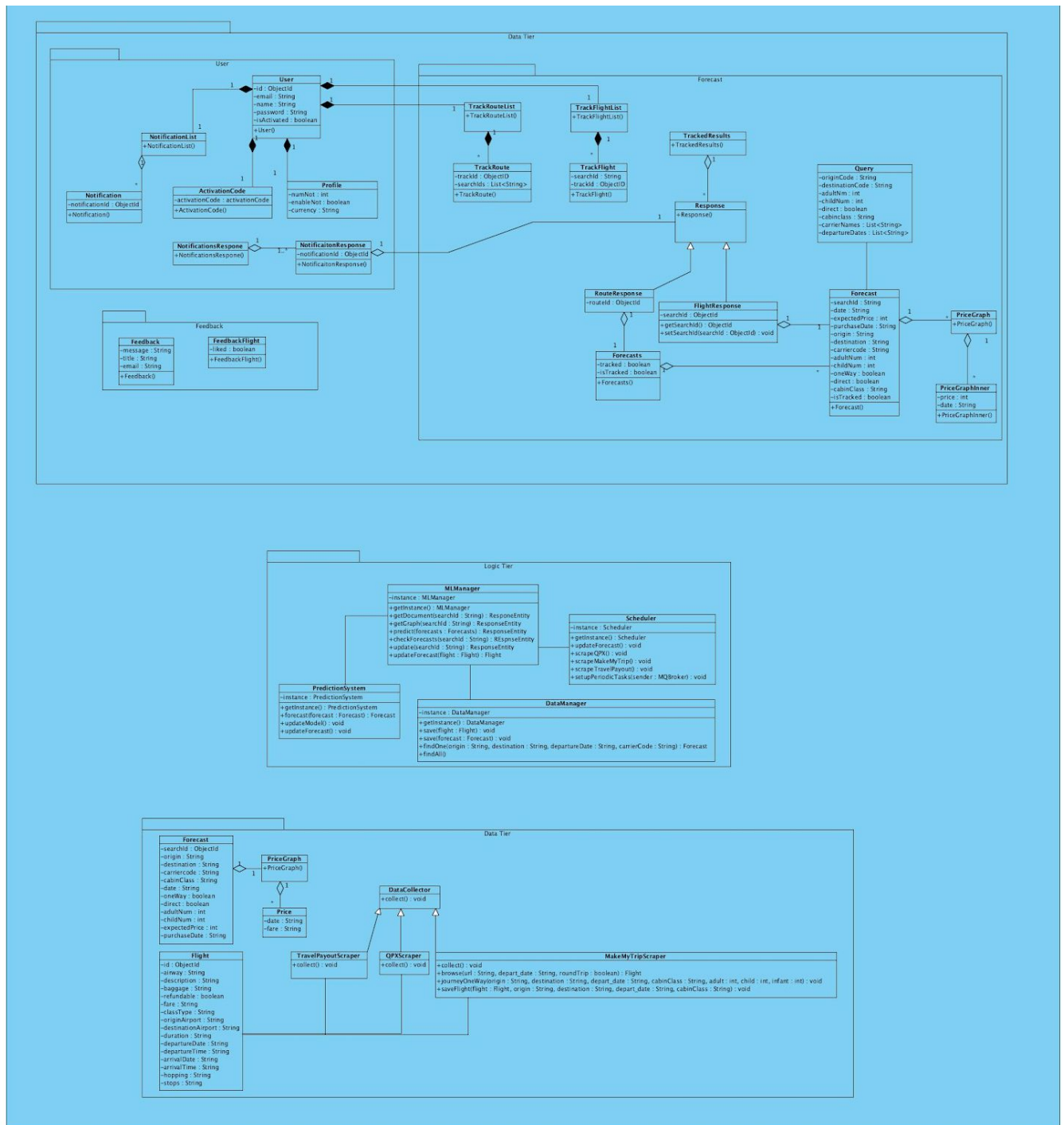


Figure 25 - AeroCast System Class Diagram

## Client Class Diagram

Client is responsible from all user interfaces and communication between the user and the system. It renders pages, gets user inputs, sends user requests to application server, receive the response of the application server, and re-renders the page according to the response. It has a single package which is Presentation Package. Presentation Package has two responsibilities. First, it handles the graphical user interface, rendering pages, and second, it handles user input and server responses. Presentation Package classes has both controller and view codes therefore client does not need any additional server side controller package.

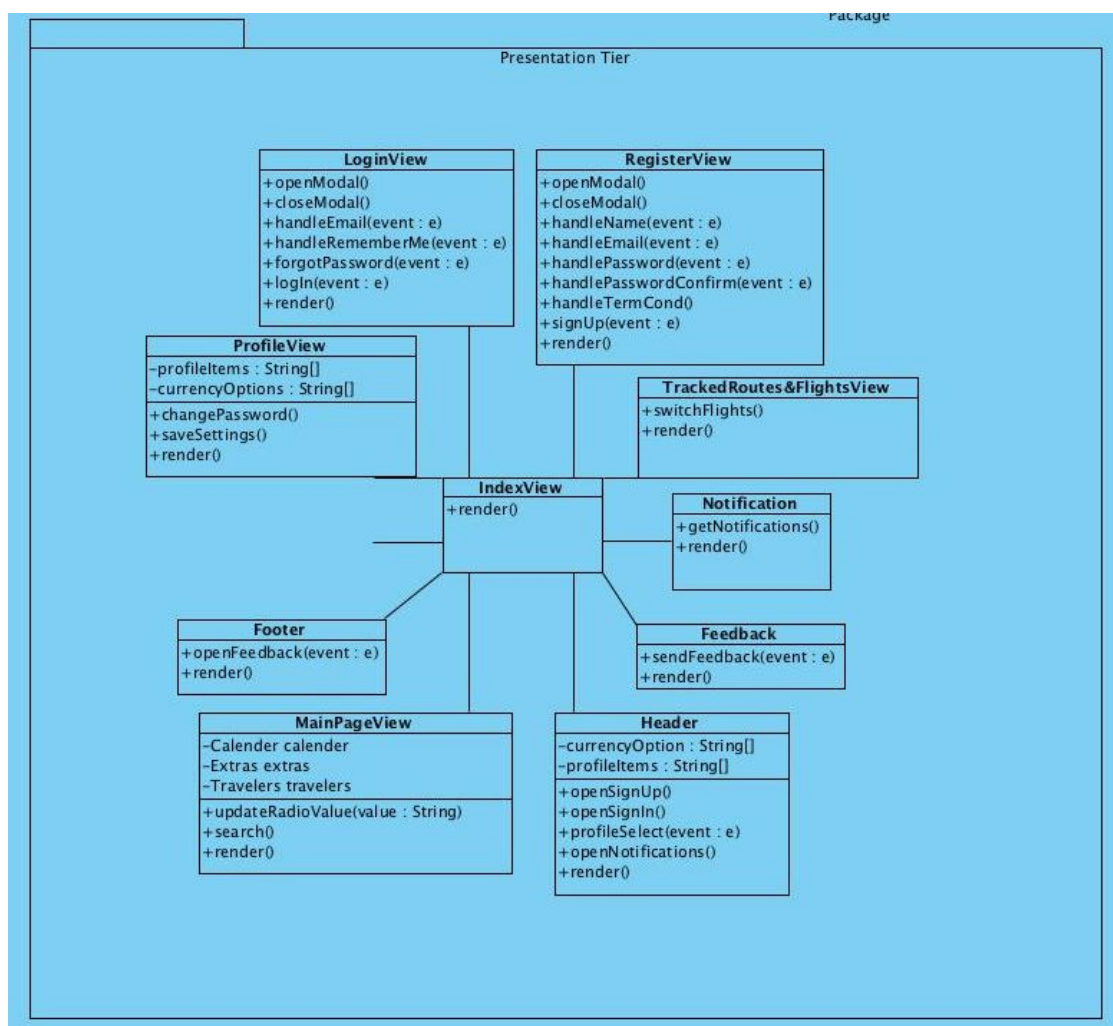
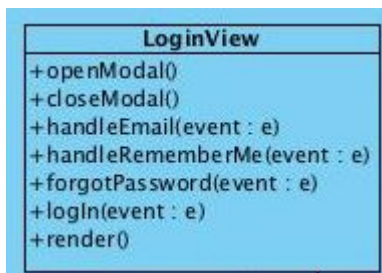
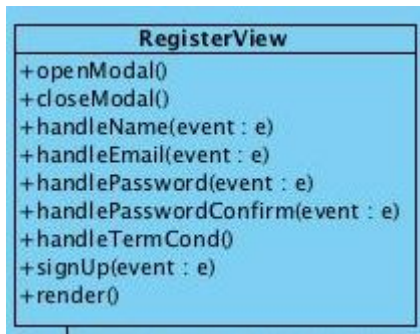


Figure - 26 Client Class Diagram

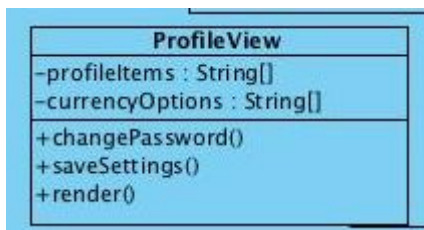
## Client Classes



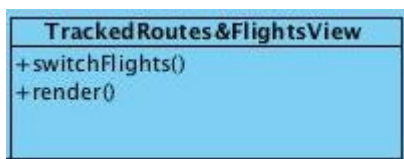
LoginView handles the login operation.



RegisterView handles the registration of the user.



ProfileView receives and updates the user profile

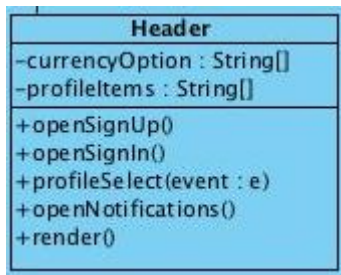


TrackedRoutes&FlightsView displays the tracked routes and flights.

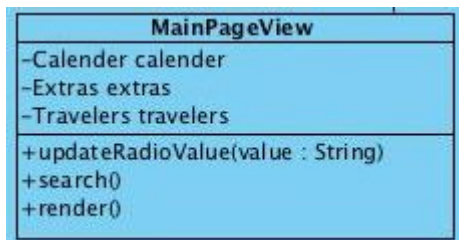


Notification displays the notifications.





Feedback sends and receives feedback



MainPageView displays the main page of the application.

### Application Server

Application Server is responsible from receiving the user requests coming from the client and handling them with appropriate operations. The application server has two components, which are the logic package and the data package. Logic package handles all communication between client and application and it manages the entities of the application, which are residing in the data package. The data package consists of the entity objects. In addition to these responsibilities, the application server communicates with the prediction server to handle the forecast related operations. The application server is the main component of the system that contains the most of the functionalities.



## Application Server Logic Tier Class Diagram

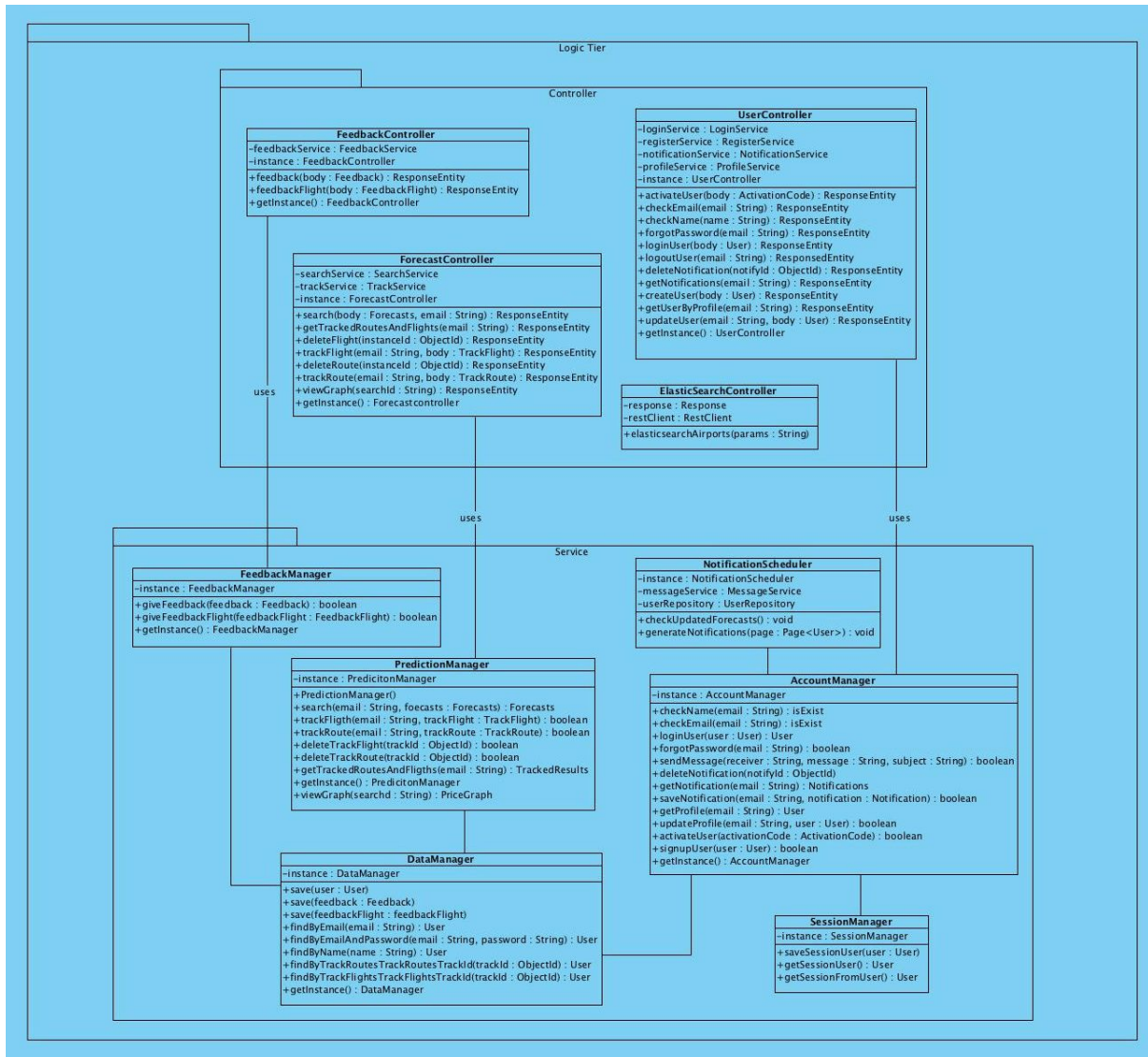
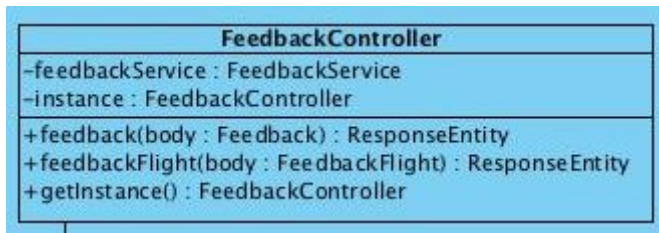


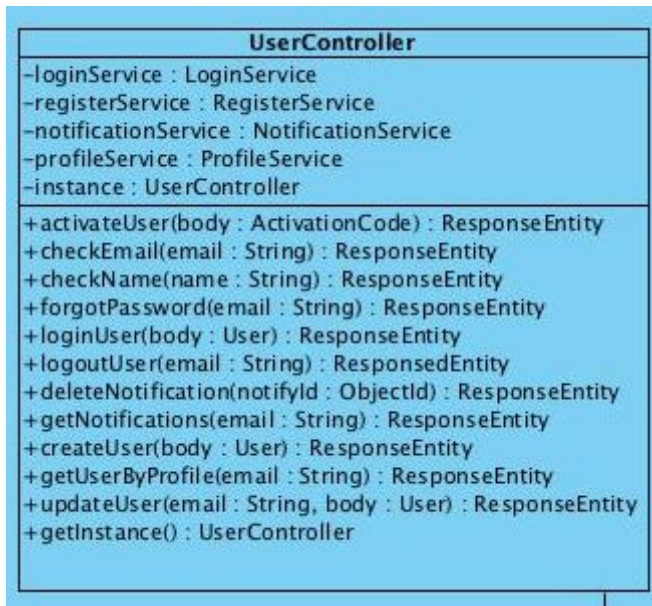
Figure - 27 Application Server Logic Tier Class Diagram



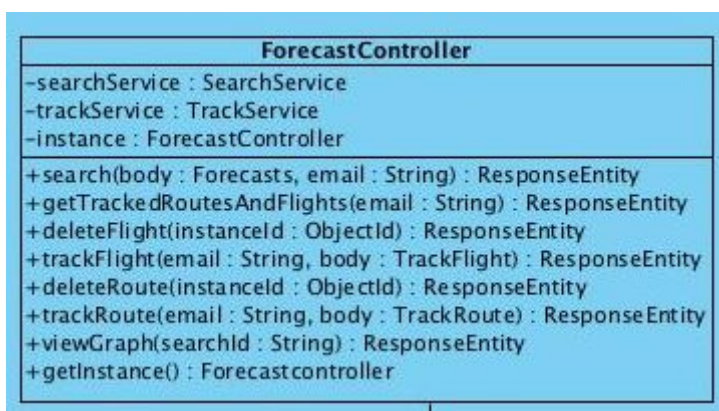
## Application Server Logic Tier Classes



FeedbackController enables communication between server and client for the feedback operations.



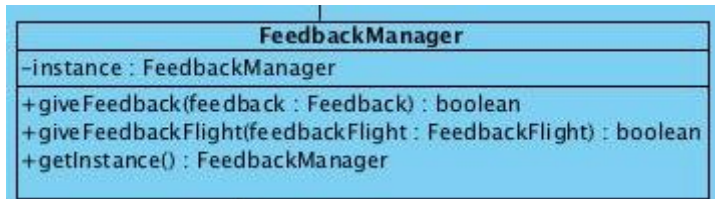
UserController provides support for the communication between the client and server for the user related operations.



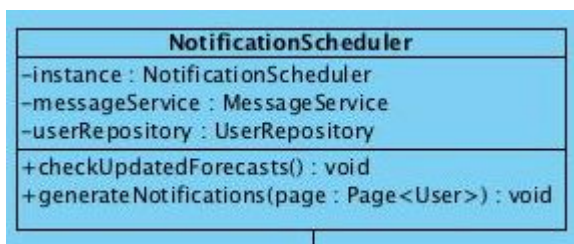
ForecastController receives forecast related inputs from the user interface.



It handles the request coming to the elasticsearch server.



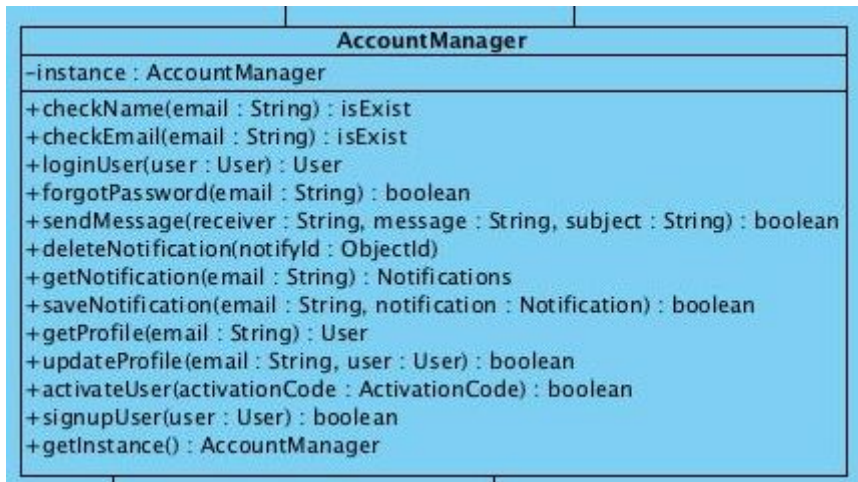
It manages the communication between the server and the database for the feedback operations.



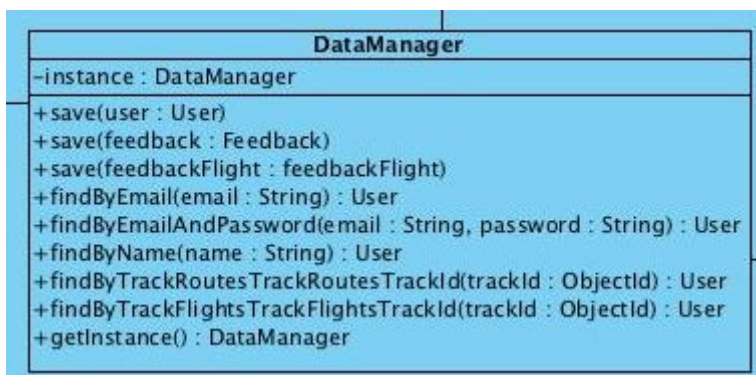
NotificationScheduler periodically checks tracked flights and routes and generates new notifications.



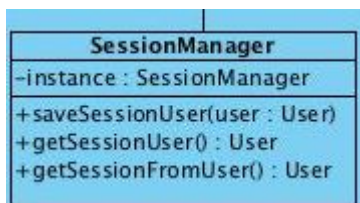
PredictionManager handles the communication between the Application server and the prediction server. It directs the request to the prediction server and receives responses.



AccountManager handles the communication between the Application server and the database for the user related operations. It also sends messages to the users.



DataManager determines the API for communicating with database repositories.



SessionManager keeps the session information of the user.

## Application Server Data Tier Class Diagram

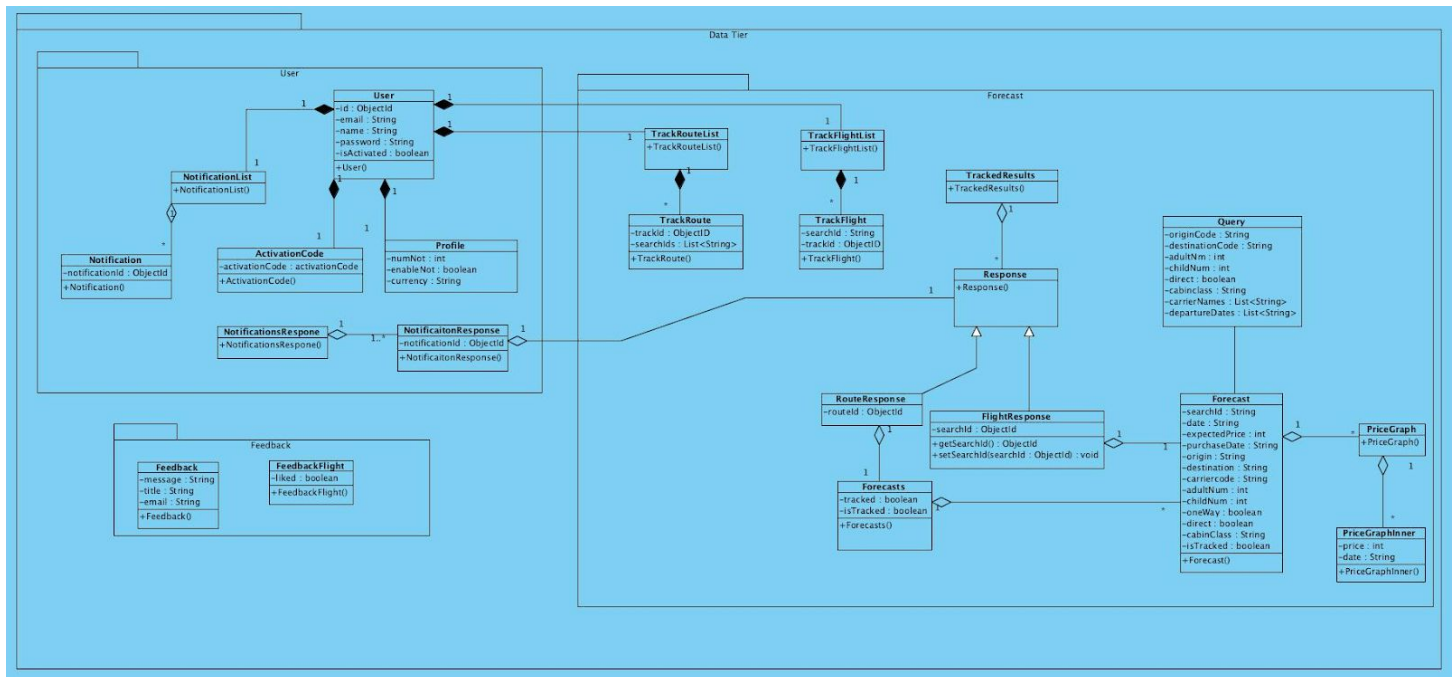
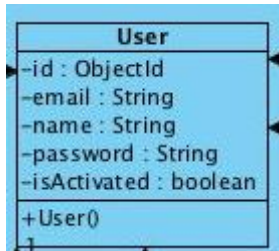


Figure - 28 Application Server Data Tier Class Diagram

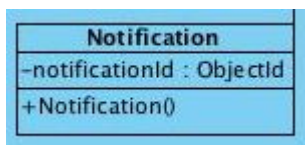
## Application Server Data Tier Classes



User class represents the user object in the system.

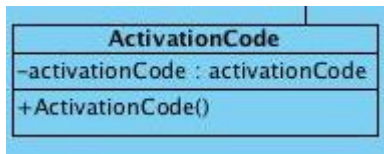


NotificationList keeps a list of notifications to return.

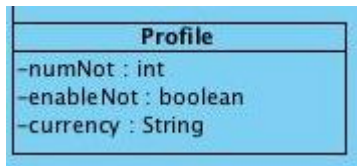


Notification class represents the notification objects of the system.

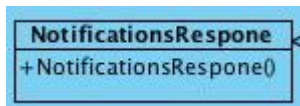




ActivationCode keeps the activationCode of the user.



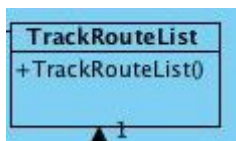
Profile class represents the profile object of the user.



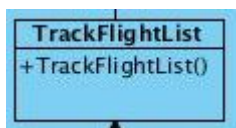
NotificationsResponse returns the notification list as response.



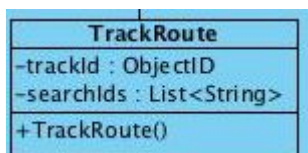
NotificationResponse represents the response object for the notifications.



TrackRouteList keeps a list of routes.



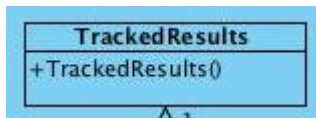
TrackFlightList keeps a list of flights.



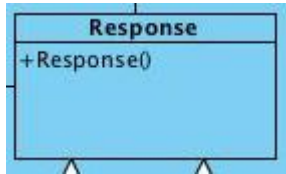
TrackRoute keeps the ids of the flights to be tracked.



TrackFlight keeps the id of the flight to be tracked.



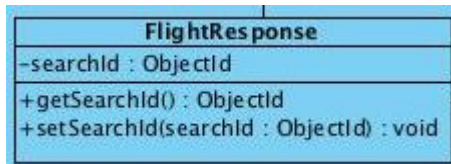
TrackedResults return the tracked routes or flights as a response.



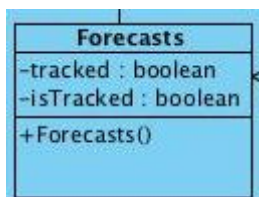
Response is the superclass for all of the responses return by the system.



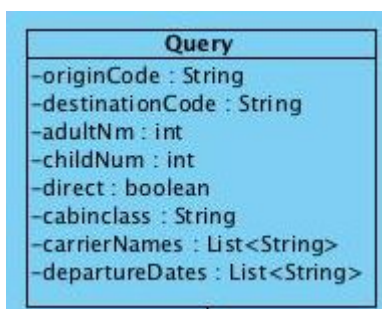
RouteResponse returns a route as a response.



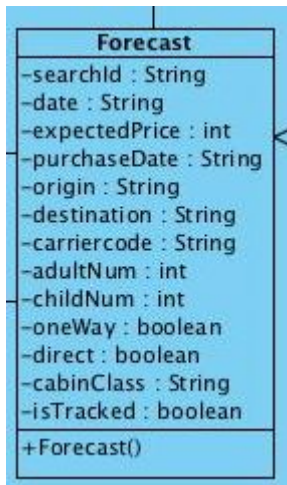
FlightResponse returns a flight as a response.



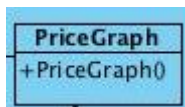
Forecasts keep a list of forecast objects.



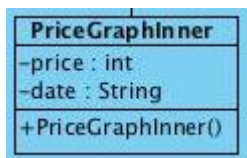
Query class represents the input received from the user.



Forecast class represents the response returned by the prediction server.



PriceGraph represents the list of date, price pairs for the graph.



PriceGraphInner keeps a date and price pair.

## Prediction Server

Prediction Server is responsible from generating, updating and returning forecasts as well as collecting and storing flight information. The prediction server has two components, which are the logic package and the data package. Logic package handles all communication between the application server and the prediction server and it makes airfare predictions based on the flight data by using the machine learning algorithms. The data package is responsible from collecting the data that is used by the logic package and it also contains the entity objects of the prediction server. The prediction server is the component of the system that handles all machine learning operations.

## Prediction Server Logic Tier Class Diagram

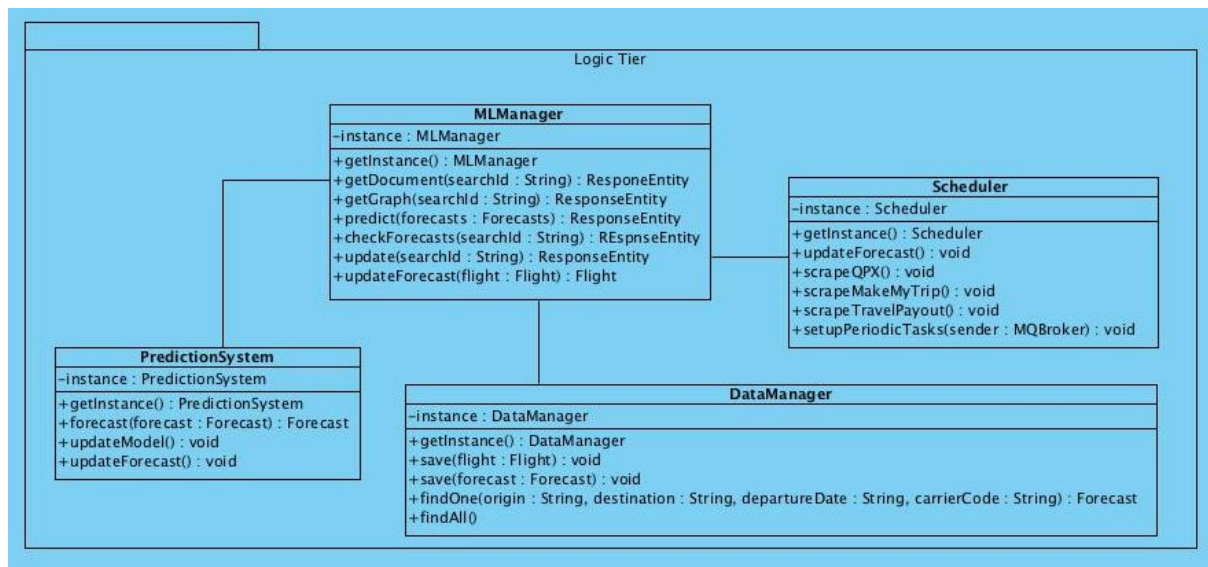
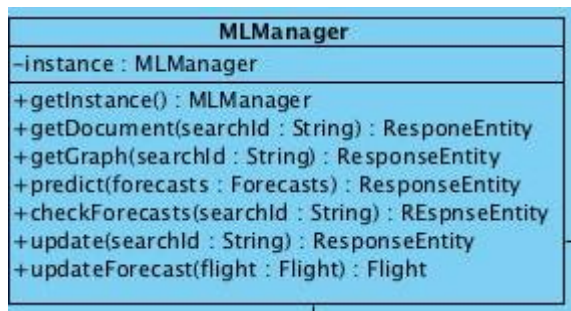
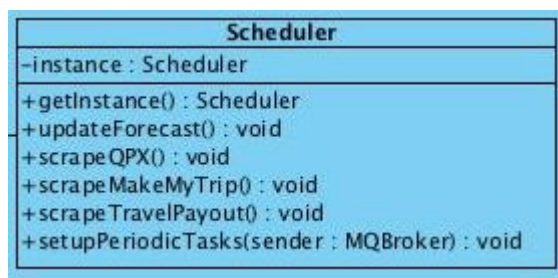


Figure - 29 Prediction Server Logic Tier Class Diagram

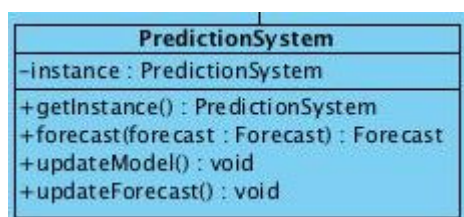
## Prediction Server Logic Tier Classes



MLManager manages the communication between machine learning module and the python application.

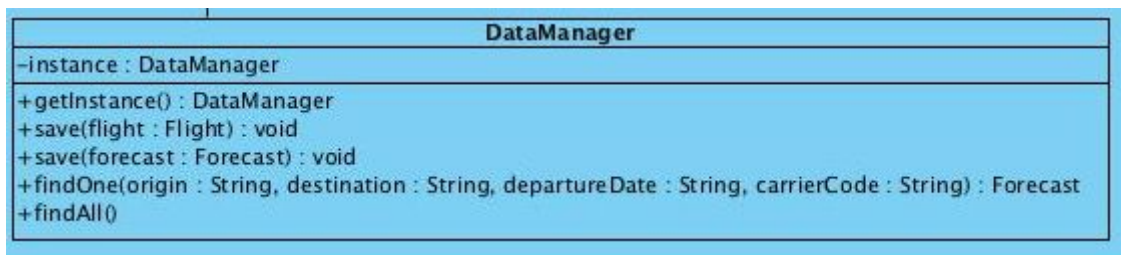


Scheduler schedules the data scrapers.





PredictionSystem handles the machine learning related tasks.



DataManager manages the communication between the database and the prediction server.

## Prediction Server Data Tier Class Diagram

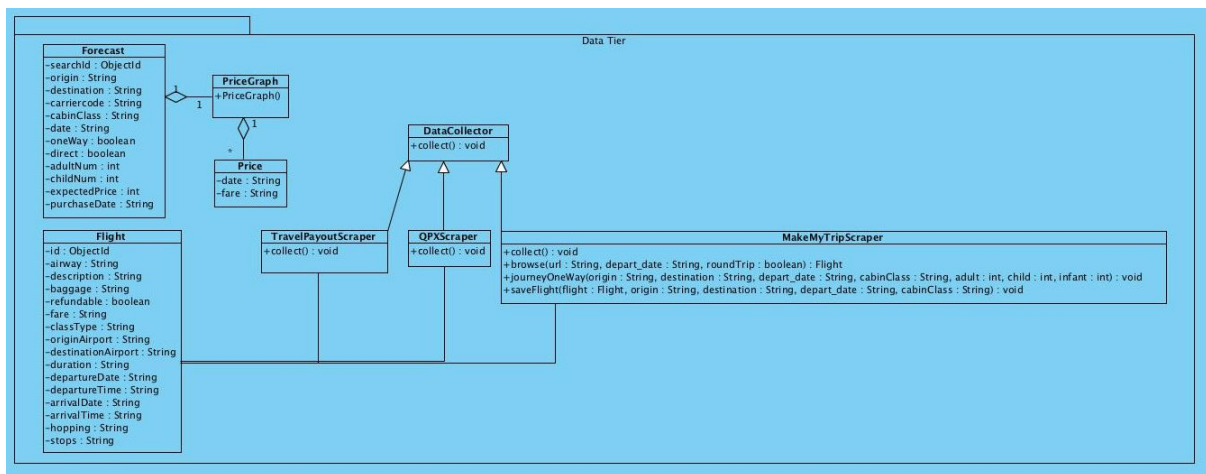
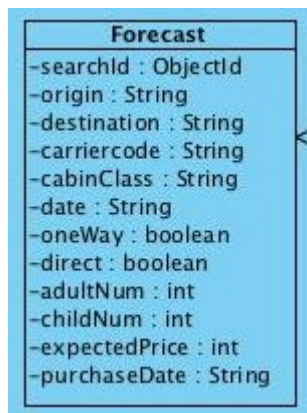


Figure - 30 Prediction Server Data Tier Class Diagram

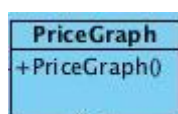
## Prediction Server Data Tier Classes



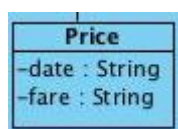
Forecast represents the forecast object of the prediction server.



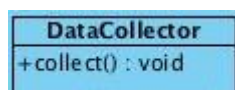
Flight class represents a flight object of the Prediction Server.



PriceGraph represents the graph information generated from the data.



Price keeps a date and price pair.



DataCollector is the super class for the data scrapers.

TravelPayoutScraper
+collect() : void

TravelPayoutScraper obtains data from the Travelpaout API.

QPXScraper
+collect() : void

QPXScraper obtains data from QPX Express API.

MakeMyTripScraper
+collect() : void
+browse(url : String, depart_date : String, roundTrip : boolean) : Flight
+journeyOneWay(origin : String, destination : String, depart_date : String, cabinClass : String, adult : int, child : int, infant : int) : void
+saveFlight(flight : Flight, origin : String, destination : String, depart_date : String, cabinClass : String) : void

MakeMyTripScraper collects data from makeMyTrip.

## 9. 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.tripadvisor.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].
- [9] Mikael Parkvall, "Världens 100 största språk 2007" (The World's 100 Largest Languages in 2007)