University of Sheffield

Design and Implementation of an Airline Reservation System - Scheduling System

Xin Wang

Supervisor: Georg Struth

A report submitted in fulfilment of the requirements
for the degree of MSc in Advanced Computer Science

in the

Department of Computer Science

September 10, 2019

Declaration

Name: Xin Wang

Signature: Xin Wang

Date: September 10, 2019

Abstract

Scheduling subsystem of an airline reservation system is described, it filters out the flights that the user needs with information entered by the user, and adjusts the ranking of flights by the recommendation system. This system is a multi-function scheduling system. In addition to the basic system and search functions, there are features such as map visualisation, stitching flights and automatic queuing to buy tickets. The system has strong robustness, maintainability and extensibility, which applies a low-coupling SSM framework, structured for a classic MVC architecture.

Acknowledgements

I want to show my gratitude to my project supervisor, Georg Struth, for his meticulously support and guidance for the preparation and research when writing this dissertation. His prompt and precise feedback as well as appropriate suggestions throughout the lifecycle of the project also contributes a lot for my study.

Also, I want to show my appreciation to anyone who participated and contributed towards this dissertation.

Content

Introduction

With the vigorous development of tourism, civil aviation industry, and the improvement of people's consumption level, the number of new routes has reached a new high, so do people’s opportunities to book tickets. Whenever people book a ticket, they are often trapped in filtering through many flights, wasting a great body of time. Such late booking often lets the right flight slip. The spring-up of numerous airline reservation systems due to the ever-renewing web technology, still cannot address these problems. This is because most airline reservation systems merely sort flights by fare, and require users to enter several filters before searching for flights. It is extremely inconvenient for users to select flights, which even indicates that the system is cumbersome. In addition, tickets desired are sold out, but the person who purchased the ticket might refund it when people who need this ticket is offline. In this case, people can only miss the ticket they need.

1.1 Aims and Objectives

In response to these issues, the object of this project is to develop a scheduling subsystem in an airline reservation system. The subsystem includes search/fuzzy search, filter/sort, map visualisation, the feature of combining different flights, and automatic queuing and buying tickets. Some basic features such as login/register/logout, book/refund, check tickets, and so on are also contained.

1.2 Technical contributions

Implementing these features requires several technologies:

- Recommendation Algorithm
- SSM(Spring+SpringMVC+MyBatis) Framework
- Google Maps Platform

Recommendation Algorithm means using some mathematical algorithms to predict tickets interested by users, and ranking based on predicting the probability of user purchasing. As a result, tickets with high purchasing rate often ranked high. Users will no longer need to turn the page several times for desired tickets. As a difficult part of the entire project, the recommendation algorithm has to solve many mathematic problems. Due to flaws in each algorithm, this project utilises a variety of algorithms to fix defects. Besides, a new algorithm is developed by the developer for this project to solve the problem that visitors and new users do not have historical data.

SSM Framework is the combination of three frameworks: Spring, SpringMVC and MyBatis.

Spring, as an open-source framework that manages the entire project, is mainly designed for Java development (Java is the main development language of this project). Spring has the feature of Inversion of Control(IoC), Aspect-Oriented Programming(AOP)and Dependency Injection(DI). In simple words, programmers can hand control of allocating memory to spring, and divide code such as logging, performance statistics, and security controls from business logic code. This allows programmers to emphasise the logic of the program. SpringMVC is a type of MVC(Model-View-Controller) framework that is used with Spring. MyBatis is a framework used to access database and mapping POJO(Plain Ordinary Java Object). Frameworks will be demonstrated in detail in the section of Literature Review.

Google Maps Platform is Maps programming API provided by Google for developers. Developers are capable of calling API to embed Google map data into the website without setting up their own map server. This technology is required in this project to realise map visualisation.

1.3 Overview of the Report

The structure of the whole report is as follows:

- **Literature Review:** Investigate, research and analyse relevant technologies.
- **Requirements and analysis:** Consider the specific objectives of the project and the method of assessment, use case diagram will be used to represent requirements.
- **Design:** Technologies selected are expected to be introduced, and the reasons for choosing these technologies will be explained. A set of UML diagrams represents the database design and project structure design.
- **Implementation:** Explain the process and principles of development and testing. Also, this section will emphasise the novelty and working principle of the algorithm.
- **Testing:** In addition to testing the functionality and robustness testing of the program, Benchmark testing will also be conducted for the value of weight between algorithms.
- **Results and Discussion:** This section analyses the results of the entire project and compares the original project objectives with self-reflection.
- **Conclusions** – summary of key points

Literature Review

2.1 JSP (Java Server Pages) and Servlet

JSP(Java Server Pages)is defined as a dynamic web page technical standard created and led by Sun Microsystems. Deploying on a network server, JSP can respond to requests sent by client-side and dynamically generate Web pages of HTML, XML or other format documents based on the content of the request before returning them to the requester (Gammeter, Gassmann, Bossard, Quack & Van Gool, 2010). Java-scripted JSP serves HTTP request from users and handles complex business requirements with other Java programs on the server. To generate part of the content dynamically using static pages as templates, JSP embeds Java code and specific changes into static pates. An XML tag called "JSP action" was introduced by JSP to invoke built-in functionality. Also, JSP tag libraries can be created where tags can be used like standard HTML or XML tags. In this case, functionality and server performance of JSP can be enhanced by these libraries without worrying about cross-platform problems. JSP files are converted into more primitive Servlet code by their compilers during running time (Bergman & Chopra, 2001). The JSP compiler can compile JSP files into Servlets written in Java code and then compile them into quickly executed binary machine code or directly compiled into binary code (Hunter and Crawford, 2010).

Servlet(Server Applet)is a server-side program written in Java. The main function is to browse and modify data interactively before generating Web content. Servlet runs on a Java-support application server. As for the implementation, the Servlet could have been able to respond any type of request (Pursnani, 2001), but Servlet can only be used to extend the Web server based on HTTP protocol in most cases.

Strengths

- Standard HTML output.
- Supporting Java Code.
- Open Source

Weaknesses

- Errors discovered can hardly be traced because of translated and compiled JSP.
- Time-consuming when the first visit.

2.2 PHP

As a server-side scripting language, PHP(Hypertext Preprocessor) is designed for web development and commonly used as a general-purpose programming language. Its grammar draws lessons from features of popular computer language such as C language, Java and Perl and is easy for developers to learn. PHP code might either be implanted into HTML code or combine with many web template systems, web content management systems and web frameworks (Paul, 2018).

PHP can execute dynamic pages faster than CGI or Perl. Compared with other programming languages for dynamic pages, PHP embeds programs into HTML documents to execute, which means it enjoys higher efficiency than those who fully generate HTML tags such as CGI. PHP can also execute compiled code. Compilation can achieve encryption and optimise the running of the code, which accelerates code's operating process.

Strengths

- Cross-platform
- Easy-to-use
- Open source and Powerful library support
- Fast speed
- Stable because many developers maintain it

Weaknesses

- Difficult to maintain because it is not extremely modular
- Low-security level due to its open-source feature

2.3 JavaScript

Flanagan proposed that JavaScript(JS), complying with ECMAScript specification, is a premium-level and interpreted scripting language (Flanagan, 2011). It supports object-oriented, imperative, and functional programming, which means it integrates the best of both object-oriented and functional programming. JavaScript, for example, has the features of object-oriented programming including easy-to-expand, high-cohesion, and low-coupling. In web development, it can directly operate HTML elements as objects, and the changes of HTML elements will not affect the relevant codes in JavaScript. Meanwhile, JavaScript can pass a function as a parameter to other methods, which not only optimises operating speed but also reduces a great body of repetitive code.

JavaScript consists of three parts - ECMAScript, DOM, BOM, in which ECMAScript describes the syntax and basic objects of the language. BOM(Browser Object Model), a browser object model, gives JavaScript the ability to "talk" to the browser. With DOM(Document Object Model), a document object model, all elements in an HTML document can be accessed. As an explanatory language, JavaScript has a speedy terminal response and can be interpreted by most browsers. This makes JavaScript an excellent Front-end language. In traditional web development, it is the back-end that handles data processing and logical code and then passes data, static resources and HTML elements to the front-end, which is a cumbersome developing model in developing user interface, not to mention complicated code. JavaScript, however, solves this problem. If programmers want to conduct fuzzy search in an input box, the system will dynamically give the user keyword prompt. The code with JavaScript used can be concise and efficient in this case. This feature is critical in practical development because some gorgeous effects will have to be discarded for better performance if the dynamic display of the front-end is completely tackled by the back-end.

Besides, it has a powerful literal notation called JSON(JavaScript Object Notation) which can exist independently without the language. Its self-descriptive and concise grammatical rules make it an ideal data exchanging format among different languages (Tilkov and Vinoski, 2010). This explains why passing a portion of data with JSON can enhance flexibility in most web programs.

Strengths

- Fast terminal reaction
- Simplicity
- Versatility

Weaknesses

- Safety
- Dependence on browsers
- Replaceable, and there are other techniques that can replace it, such as JQuery.

2.4 Framework

Web Framework is a designed standard to serve web development, which can be interpreted as semi-finished software. An increasing number of mature frameworks emerge thanks to the ever-renewing web technology. This section will examine several popular web frameworks to increase developing efficiency of this project.

2.4.1 Spring

Spring is an open-source framework, and a lightweight Java development framework pervading in 2003 which derives from part of the ideas and prototypes outlined in Rod Johnson's book named Expert One-On-One J2EE Development and Design (Johnson, 2004). It was created to simplify enterprise application development. One of the main merits of the framework lies in its layered architecture that enables user to select which component to use while provides an integrated framework for J2EE application development (Arthur and Azadegan, 2005). Spring uses basic JavaBean to accomplish things that were previously only possible by EJB (Fisher, 2012). Besides from server-side development, Spring is beneficial for all Java applications in terms of simplicity, testability and loose-coupling. The core of Spring includes Inversion of Control (IoC), Aspect-Oriented Programming (AOP) and Dependency Injection (DI). In simple words, Spring is a tiered JavaSE/EE full-stack, lightweight and open-source framework.

Core

- Inversion of Control, or an object created by Spring. Instead, it is not produced by programmers but is configured by Spring. In other words, giving control of the object to Spring.
- Aspect-Oriented Programming: AOP is a continuation of object-oriented programming and a derivative model for Functional programming. It isolates all parts of the business logic to reduce the coupling among all parts, improve program's reusability and promote developing efficiency (Laddad, 2010).
- Dependency Injection: injecting values into the properties of an object based on the IoC.

Strengths

- **Easy decoupling, simplified development (high-cohesion and low-coupling)**

  Spring is a large factory(container) that can create and maintain all objects before leaving them under Spring management. Spring factory is used to produce bean.

- **Support from AOP programming**

  Spring provides face-oriented programming, which can easily implement functions such as permission intercepting, operation monitoring and so on (Gupta and Govil, 2010).

- **Support of declarative transactions**

  Transaction management can be finished merely with configuration, without manual programming (Aswani, 2014).

- **Testing programs conveniently**

  Spring supports Junit4 and can test its program with annotation easily.

- **Easy integration of excellent frameworks**

  Spring does not close off to a variety of high-quality open-source frameworks, supporting various excellent frameworks (e.g. Struts, Hibernate, MyBatis, Quartz, etc.) directly.

- **Reduce the difficulty of using the JavaEE API**

  Spring provides encapsulation for some of the most difficult API (JDBC, JavaMail, remote calls, etc.) in JavaEE development, making them much easier to use.

Weaknesses

- The controller is overly flexible, and a common controller is needed.

- Not suitable for distributed development
- Code is difficult to split.

2.4.2 Spring MVC

Spring MVC is a kind of MVC (Model-View-Controller) framework, belong to the presentation layer. As one part of Spring framework, it is released after Spring 3.0.

Strengths

- Multi-view shares a model that greatly improved code reusability
- MVC's three modules are relatively independent, which is a loosely coupled architecture (Dandan Zhang, Zhiqiang Wei and Yongquan Yang, 2013).
- Controllers increase the flexibility and configurability of applications.
- It is beneficial for software engineering management.

Weaknesses

- Complicated principles.
- Increased complexity in system structure and implementation.
- Inefficient access to model data for viewers.

2.4.3 MyBatis

MyBatis is a Java persistence framework that uses XML descriptors or annotations to couple objects with saved procedures or SQL statements. Unlike the ORM framework that maps Java objects to database tables, MyBatis rather map them to SQL statements (Reddy, 2013). This means high code reusability and avoids repetitive code. Coordinating with spring's management of the database connection pool, the DAO layer's code can be completely replaced by XML and interfaces (Ho, 2012). JDBC has to control database connection or uses c3p0 connection pool to manage connections when operating database every time. Unlike JDBC, MyBatis' approach is more concise and efficient.

Compared to fully automated Hibernate, semi-automated MyBatis requires developers to write statements and define maps, which adds programmers' operations, yet brings flexibility in design (Dandan Zhang, Zhiqiang Wei and Yongquan Yang, 2013). Some features of hibernating, such as lazy loading, caching, mapping and etc. are also supported by it. MyBatis is less compatible with the database than hibernate and lacks portability, but the programming of flexible and high-performance SQL statements should not be a problem.

Strengths

- Easy to maintain and management

- Replace writing logical code with labels
- Auto-mapping
- High flexibility

Weaknesses

- Hard to debug
- Difficult to stitch complex SQL statement
- High reliance on SQL statement

2.4.4 Bootstrap

Bootstrap is a set of open-source front-end frameworks for website and web application development, including HTML, CSS and JavaScript frameworks (Hesterberg, 2011). It provides font typography, forms, buttons, navigation and various other components and JavaScript extensions, aiming to make the development of dynamic web pages and web applications easier (Mark Otto, 2019). In addition to providing a variety of off-the-shelf styles, Bootstrap has the advantage of a grid system that allows web pages to be displayed at different resolutions or on different devices. In other words, it can adjust the layout according to different screen sizes. At a time when the number of smartphone users is huge, the grid system meets the need for people to switch back and forth between PCs and mobile devices, which is one of reasons why it ranks third on GitHub. With internet access, there is also no need to import Bootstrap source code, which saves a lot of storage occupied by front-end projects.

However, the problem about the compatibility of Bootstrap toward IE should not be neglected. For example, Bootstraps sets all element box models as border-box, which however is the box model in IE promiscuous model. A large number of H5 labels and CSS3 syntax are also used which all have difficulty in their compatibility. Even if there are many methods to become compatible with IE, other files which will take up a large amount of storage need to be introduced. This definitely leads to slower loading speed and affects the user experience.

Strengths

- Beautiful components provided, coupled with easier and faster development, allow back-end developers to easily make beautiful UI Grid systems (Patel, 2017)
- Grid systems adapting to varying sizes of equipment
- Being imported by link with an internet connection

Weaknesses

- Compatibility issues for some browsers

- API and Tools
- Combining with the background of the project, some API and tools are required, which will be studied and explored in this section

2.5 Tools and APIs

2.5.1 Maven

Maven, as a project hosting tool, is mainly used for building automation of Java projects. The Maven Project Object Model (POM) can use an XML document to describe the dependence, libraries and Jar packages needed by a project (Redmond, 2019). Apart from the ability of program-building, Maven also provides advanced project managing tools. Maven and Ant address two different aspects of the building problem. Ant offers cross-platform build tasks for Java technology development projects. Maven itself describes the advanced part of the project, borrowing most of the build tasks from Ant.

2.5.2 Junit

Junit, founded by Kent Beck and Erich Gamma, is a unit testing framework of Java language, which has its own Junit extended ecosystem (Massol and Husted, 2004). Most Java developments' environments have integrated Junit as a tool for unit testing. Junit testing is a programmer test, or white box test because the programmer knows how and what functions performed by the software under test.

2.5.3 Google Map Platform

Google Map Platform is Maps programming API provided for developers by Google (Google, 2019). It enables developers to embed GoogleMaps' map data into their websites without setting up their own map servers, thereby embedding the mapping service applications into GoogleMaps and providing location services to users with GoogleMaps map data.

2.5.4 Bcrypt

Bcrypt, as a password hashing feature, devised by Niels Provos and David Mazieres, relying on the Blowfish cypher and showed at USENIX in 1998 (Wikipedia, 2019). Working as an adaptive function, Bcrypt can still reject brute-force search attacks of higher computation power, even though the iteration count is able to decelerate Bcrypt.

2.6 Recommendation Algorithm

Recommendation system has gradually become a crucial research content of major websites due to the developing in e-commerce and machine learning (Li, Zhang and Wang, 2013). A preeminent recommendation system can be employed to recommend products, accurate advertisements delivery and information push, and so on. Combining with background of this project, some recommendation algorithms will be studied in this section, in the way of overviewing, studying and analyzing.

2.6.1 Similarity Calculation

In the field of machine learning and data mining, the similarity is a numerical measure for the degree of difference between two objects (Ahn, 2008). In some recommendation algorithms, similarity calculation between users and items is needed. There is three commonly applied similarity calculating methods, including Euclidean Distance, Pearson Correlation Coefficient and Cosine Similarity.

2.6.1.1 Euclidean Distance

Euclidean Distance is a relatively simple way to calculate similarity (Ahn, 2008). It represents the similarity between two objects by calculating the distance between two objects in the coordinate system (the smaller the distance value is, the higher the similarity becomes). As shown in the following formula, the distance between two points is calculated first and then substituted into similarity formula to get the result (Robinson, 2019).

$$\mathrm{d}(\mathrm{x},\mathrm{y}) = \sqrt{(\sum (x_i - y_i)^2)}$$

(Technical Whitepaper, 2005)

2.6.1.2 Pearson Correlation Coefficient

Pearson Correlation Coefficient is a linear correlation coefficient, which is a statistic that reflects the degree of linear correlation between the two variables (Benesty, Jingdong Chen and Yiteng Huang, 2008). The greater its absolute value is, the stronger the correlation becomes. Based on the Cauchy-Schwarz inequality, it owns a value from +1 to -1, in which 1 is the total positive linear correlation, 0 means no linear correlation, and -1 indicates total negative linear correlation, which is commonly applied in the sciences.

As shown below, the molecule is the covariance of x and y, and the denominator is the standard deviation of x and y.

$$\mathrm{p(x,y)} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum {x_i}^2 - (x_i)^2}\sqrt{n\sum {y_i}^2 - (y_i)^2}}$$

(Benesty et al., 2009)

Thus, the formula can be represented as follows:

$$\mathrm{p(x,y)} = \mathrm{corr(X,Y)} = \frac{cov(X,Y)}{\sigma x \sigma \mathrm{y}}$$

(Benesty et al., 2009)

2.6.1.3 Cosine Similarity

Cosine Similarity evaluates their similarity by calculating the angled cosine values of two vectors. Cosine similarity plots vectors into vector space, such as the most common two-dimensional space, according to the coordinate value (Ye, 2011).

Cosine similarity measures the similarity between two vectors by scaling the cosine value of the angle of the two vectors. The cosine value of 0-degree angle is 1, while the cosine values of any other angle are no more than 1, and the minimum value is -1.

Thus, the cosine value of the angle between two vectors determines whether the two vectors point roughly in the same direction.

When two vectors have the same direction, the value of cosine similarity is 1, the value of cosine similarity is 0 when the two vectors are 90 degrees, and when two vectors point in the exact opposite direction, the value of cosine similarity is -1. The result only depends on the direction of the vector and is independent of the vector length. Since cosine similarity is typically used in positive space, the given value is between -1 and 1.

The cosine value of two vectors can be derived by using the Euclidean dot product formula.

$$\mathrm{a \cdot b} = \mathrm{\|a\|\|b\|} \cos\theta$$

(Perone, 2013)

Two attribute vectors: A and B, whose cosine similarity is given by the dot product and vector length, which is as follows:

$$\mathrm{T(x,y)} = \frac{\sum x_i y_i}{\sum x^2 \sum \mathrm{y}^2}$$

(Perone, 2013)

2.6.2 Content-based Recommendation

Content-Based Recommendation (CB) is a kind of machine learning algorithm, which makes recommendations based on content information of the project without relying on the user's comments on the project (Pazzani and Billsus, 2007). Instead, there is more need to use machine learning methods to obtain information of user interest, from cases about feature description of content.

In a content-based recommendation system, the feature properties of an item (or object) will be defined. The system learns users' interest relying on users' evaluations and features of objects, and analyzes the matching degree between the user and projects to be predicted. In general, the CB process is divided into three main steps:

1. **Item Representation:** some features(contents of the item) are extracted for each item to represent it.

2. **Profile Learning:** learning features (profile) of a user's preferences by utilising characteristic data of a user's likes (and dislikes) in the past.

3. **Recommendation Generation:** recommending a group of most relevant items via comparing features of a user's profile obtained in the former step with the candidate item (Sato et al., 2017).

The user's data model depends on the learning algorithm applied, such as decision tree, linear classification algorithm and Naive Bayes algorithm (Belsare and Deshmukh, 2018). Content-based user data needs historical data of the user, and the user profile model might change with users' preferences.

Advantages

- **User Independence:** As each user's profile is based on his or her own preferences for the item, it is irrelevant to others' behaviour. CF(Collaborative Filtering), on the other hand, has to take advantage of other users' data. The user independence of CB prevents some cheating behaviours, such as using multiple accounts to improve recommendation ranking of an item (Pereira and Varma, 2016).

- **Transparency:** The algorithm of CB is easy to understand for users and ordinary people (Meteren and Someren, 2000). Thus, it is better explained than other algorithms. When a user asks why these products are recommended, they will realise that certain attributes match his or her interests.

- **New Item Problem:** If a new item is added to the database, it will be immediately recommended. The opportunity for recommendation is the same between new and old item because CB does not need any records about user purchasing or following for the new item. Compared to CB, a pure CF recommendation system will never recommend a new item.

Disadvantages

- **Limited Content Analysis:** If the item in the system documents (such as personalized reading), it is easier to extract the feature of items “more precisely” using the method in the information retrieval. However, in most cases, accurate characteristics describing items can hardly be extracted from them, such as the movie recommendation item or social software recommendations. In fact, almost all item features we extract can only embody one item partly, rather than comprehensively. One problem with this is that characteristics extracted from two items are totally identical, which means that CB is completely incapable of distinguishing between them (Lops, de Gemmis and Semeraro, 2010). For instance, if only actors or directors can be extracted from a movie’s information, then two films with same actors and directors will become indistinguishable for CB.
- **Over-specialization:** since CB’s recommendation only relies on users’ previous preferences toward certain items, recommendations generated by it are similar to those that users used to like. CB, thus, cannot dig up the potential interests of a user. Moreover, the preferences of users keep changing (Pinela, 2017). For example, a user purchased an airline ticket in one company but changed to another airline company for the next time. This is because he or she is unsatisfied with the previous service. It is evidently inappropriate to recommend the company that the user purchased before.
- **New User Problem:** the lack of history for preferences in new users means that the system will hardly be able to recommend according to the behaviour records, and so does CF (de Campos et al., 2010).

2.6.3 Collaborative Filtering Recommendation

Collaborative Filtering Recommendation is one of the earlies the most successful technologies applied in the recommendation system (Linden, Smith and York, 2003). It generally utilizes the near-neighbour technology to calculate the distance between users via the user’s histories about preferences information (Zhi-Dan Zhao and Ming-Sheng Shang, 2010). Afterwards, it uses the target user’s nearest neighbour users’ weighted evaluation value of commodity comments to predict the target user’s preference degree for a particular commodity, so as to conduct recommendations based on this degree. The benefit of this lies in the fact that unrealized interests of users can be unearthed. To illustrate, the system might recommend a science fiction book according to the similarity calculated by the CF algorithm, after a user purchased an algorithm book. Chances are that the user will be interested in the recommended book. Generally, the CF recommendation is divided into three categories: user-based, item-based and model-based CF.

User-based CF mainly considers the similarity between users. As long as users’ preferred items and estimate targeted users’ rating toward them are discovered, several items with the highest rating can be recommended to users (Herlocker et al., 2004).

Item-based CF, partly similar to user-based CF, needs to find the similarity among items. Only if target users' ratings about some items are found can similar items with high similarity be predicted (Su and Khoshgoftaar, 2019)? These items with the highest rating can be recommended to users. Different from CB algorithm, Item-based CF focuses on the relationship between items and users and does not require properties of items.

Model-based CF addresses problems using the idea of machine learning to model (Koren, 2008). For instance, a matrix of m*n can be listed with m users and n items, however, only part of users and data have ratings and the other part is blank. At this point, some sparse data will be used to predict the rating relationships between blank items and data and find the highest rating items for users.

Advantages

- The ability to handle unstructured and complex objects, such as movies and people
- Ability to avoid incomplete and inaccurate content analysis due to all user data sharing
- Being capable of finding users' potential or unrealized preferences
- Ability to effectively utilize feedback information from other similar users, reduce the amount of user feedback and accelerate personalized learning (Cacheda et al., 2011)

Disadvantages

- Sparse data
- A huge amount of computation
- Changeable user preferences

2.6.4 Association Rule-based Recommendation

Association Rule-based Recommendation, on the basis of association rules, regards purchased goods as the rule header and the ruling body as the recommendation object. Association rule mining can find the relevance of different goods during sales process, which has been successfully applied in retail industry (Mobasher et al., 2001). And Association Rule-based Recommendation is widely used in areas with large amounts of data, especially e-commerce platforms. For example, the user who buys products on Amazon will receive recommendations of other items bought by other customers who also purchased these products.

The association rule is to count the percentage of transactions of purchasing the commodity set Y in transactions of buying the commodity set X in a trading database (Sandvig, Mobasher and Burke, 2007). In simple words, the degree of intention for users to buy other products when they are purchasing some products. For example, 60% of customers who buy bread also buy milk.

The common application of associate rules is to recommend a single item with a single item (Zhang and Jiao, 2007). In this case, only frequent item sets are needed. Goods are not all equally sold, which means that combination, bundle sale, sales with the present, enterprise purchase and other others will affect the generation of frequent item sets. Using support degree only to measure associations between items can easily lead to spurious associations.

In the association rules, since Support indicates the percentage of purchasing A and B simultaneously, Confidence represents the credibility of A recommends B. Thus, the formula of Upgrade = Support* Confidence can be applied to indicate that A recommends B and the rate of buying A and B at the same time. Compared to simply using Support, this formula is more comprehensive and avoids the elimination of association rules with medium Support and Confidence.

Advantages

- Intelligible and interpretable Algorithm (Yan and Li, 2006)
- Ability to dig out potential combinations of products that users might need
- Ability to handle unstructured and complicated objects

Disadvantages

- Time-consuming algorithm
- The synonym of commodity names is the difficulty of association rules (Tan, Guo and Li, 2008)
- Requiring a great body of data.

2.6.5 Hybrid Recommendation

Regarding the fact that each recommendation approach has its pros and cons, Hybrid Recommendation is often adopted in practical situations (Albadvi and Shahbazi, 2009). The combination of content-based recommendations and collaborative filtering recommendations is the most researched and applied one. The simplest approach is to use a content-based approach and a collaborative filtering recommendation method to produce a predicted result of a recommendation, and then combine the result with a certain method. Although there are many recommendations combining methods, it does not always work out in a specific problem. The most important principle of Hybrid Recommendation is that this can avoid or compensate for weaknesses in each recommendation technology (Nilashi, Ibrahim and Ithnin, 2014).

There are seven combination methods (Lucas et al., 2013):

- **Weight:** weighted results of multiple recommendation technologies.
- **Switch:** applying different recommendation technologies according to the background of the problem and the actual situation or requirements.
- **Mixed:** a variety of recommendation technologies are adopted simultaneously to given several recommendations for users.
- **Feature combination:** combining characteristics from different recommendation data sources before being used by another recommendation algorithm.
- **Cascade:** based on a rough result produced by one recommendation method, the second recommendation technology is applied to conduct a more accurate recommendation.
- **Feature augmentation:** embedding additional characteristic information produced by one technology in the characteristic input of another recommendation technology.
- **Meta-level:** using model generated by one recommended method to become the input of another recommendation method.

2.7 Methodologies

In order to increase the possibility of successful completion of the project, a systematic approach has been adopted. This approach needs to comply with software development methodology. In software engineering, a system development methodology refers to the framework for structuring, planning and controlling the development process of the information system. Over the years, a variety of frameworks like this have evolved and each has their strengths and weaknesses (CMS Office of Information Service, 2008). Therefore, there should be some good reasons to choose a suitable method.

In this section, four software development models are introduced, and their advantages and disadvantages are also discussed.

2.7.1 Waterfall Model

The first software development model was the waterfall model proposed by W.Royce in 1970 (Royce, 1987). The model gives a fixed order of transition from one stage to the next stage of life activity. It divides the software lifecycle into seven stages: system requirements; software requirements; analysis; program design; coding; testing; and operations (Royce, 1987), as depicted in Figure 2.1. The method of the waterfall model helps to eliminate many of the difficulties that have been encountered in the software project (Boehm, 1988). However, the basic scheme of the waterfall model has encountered some more fundamental difficulties, all of which have led to the formulation of the alternative process model (Boehm, 1988). According to Petersen et al (2009), there are three problems in the waterfall model. First, the division of each stage is completely fixed, and a large number of documents are generated between stages, which greatly increases the workload. Moreover, the user can only wait until the end of the whole process to see the development results, thereby increasing the risk of development. In addition, early errors may have to wait until the test phase can be found, which will lead to serious consequences (Petersen, Wohlin and Baca, 2009).

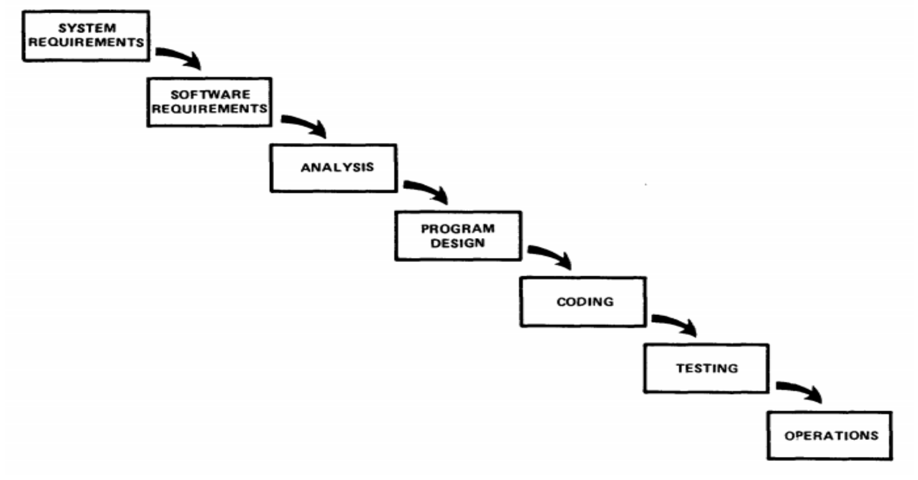


Figure 2.1 (Boehm, 1988)

Advantages

- Easy to manage with simple structures
- Good work efficiency for small projects where user requirements are clear
- Only need to focus on the current stage when the previous stage was completed

Disadvantages

- The huge workload for big projects
- Not adapting to changes in user needs
- The expensive cost of changing the requirements and uncertainty
- High levels of risk because user feedback is only available at the end of development

2.7.2 Incremental Model

The incremental model can adapt to the changes in user requirements to overcome the shortcomings of the waterfall model (Singh & Kaur, 2017). It can be understood as the iteration of multiple waterfall models. As shown in Figure 2.2, the product is decomposed into multiple modules according to its function, and each module through the demand, design, implementation, and testing phases. Thus each module can be used when it is completed, and the incremental model can avoid a long development time. In addition, this model helps mitigate the adverse effect of introducing new requirements (Basics et al., 2019). However, there are some problems with this model; for example, each new module must be integrated with the previous module and any existing system at the beginning (Basics et al., 2019). This is the reason why the incremental model leads to high cost.

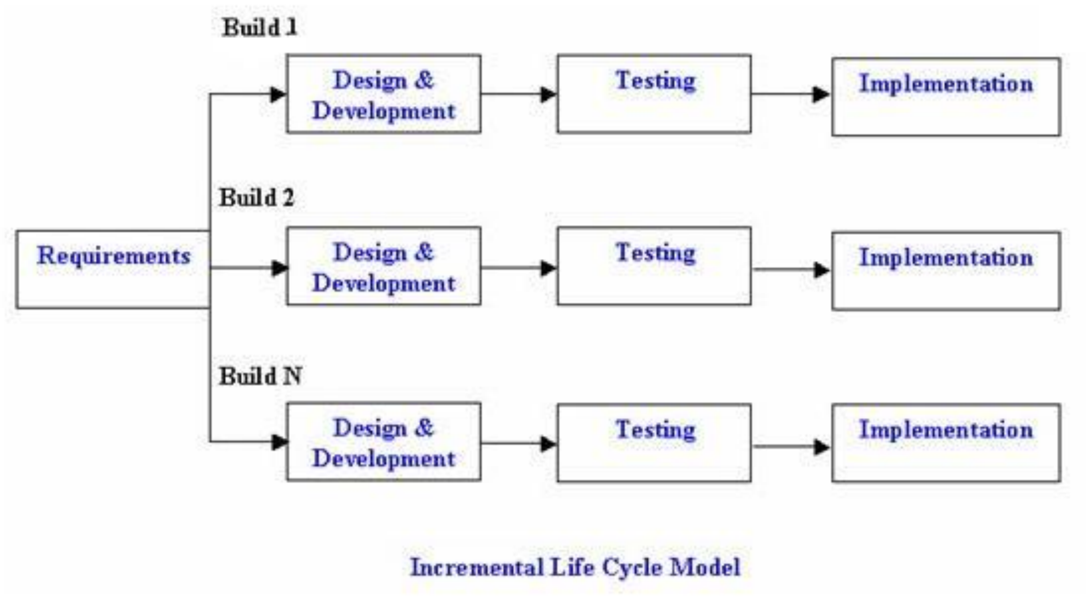


Figure 2.4 (Singh & Kaur, 2017)

Advantages

- Users can understand the progress of software projects in a timely manner
- Developing a component-based unit reduces the risk of software development
- Flexible development order
- Debugging and testing becomes easier because the product is decomposed according to function

Disadvantages

- High development costs caused by decomposing projects
- Only applicable to modules that can be modularized
- The implementation of incremental functionality may affect the architecture

2.7.3 Spiral Model

In 1988, B.W. Boehm proposed the spiral model (depicted in Figure 2.3). It combines the waterfall model with the rapid prototyping model, which emphasizes the risk analysis ignored by other models, and is particularly suitable for large and complex systems. Compared to the waterfall model and the incremental model, the concept of prototyping and risk assessment are proposed in the spiral model (Laplante, 2007). In the spiral model, the first prototype may be without any output, but it is used to define the direction of the entire software development process. On the other hand, the way to achieve risk assessment is to introduce a very strict risk identification, risk analysis and risk control before every development stage of the waterfall model, which decomposes software project into small projects. Each small project identifies one or more major risks until all major risk factors are determined.

The spiral model consists of four stages: planning, risk analysis, project implementation and user assessment. Similar to the incremental model, software projects iterate over these phases in the spiral. Requirements are collected at the planning stage. In the risk analysis phase, a process is needed to identify risks and to change the solution, and a prototype is generated at the end. Based on the generated prototypes, development and testing are carried out during the project implementation phase. Finally, the customer is allowed to give an assessment before the next spiral. "In the spiral model, the angular component represents progress, and the radius of the spiral represents cost".

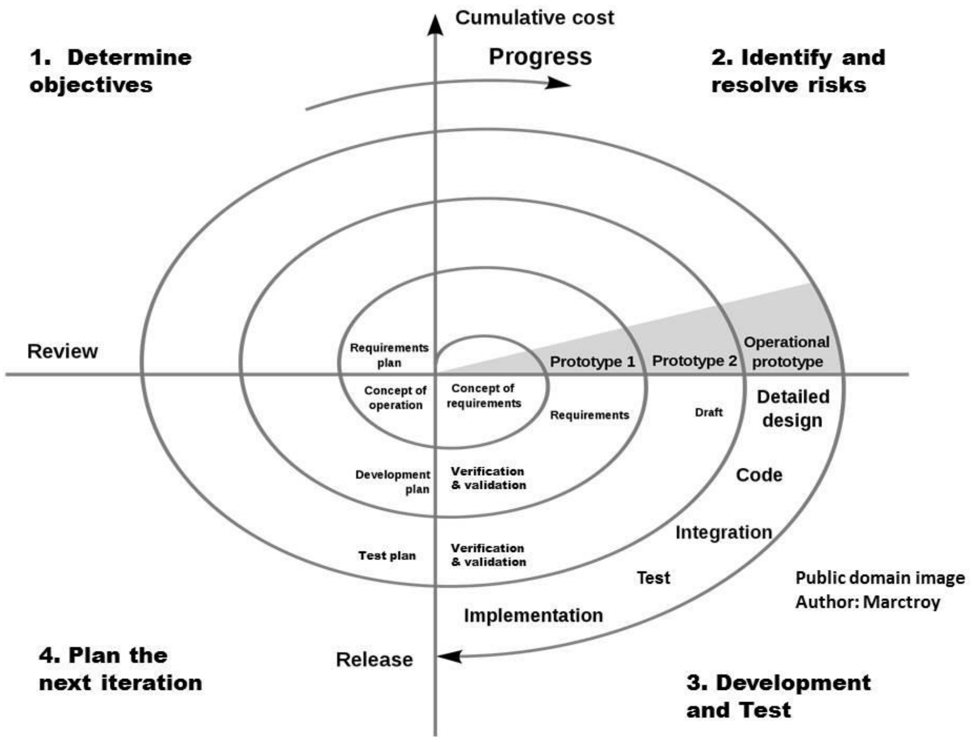


Figure 2.3 (Boehm, 1988)

Advantages

- The development of software has a clear direction at the beginning of each iteration
- The risk is maximally reduced

- The development process is flexible and can be easily changed
- The user evaluation phase ensures the implementation of the user requirements

Disadvantages

- The high cost of risk management
- Risk analysis requires a high degree of expertise
- The success of the project relies heavily on risk analysis

2.7.4 Agile Software Development

Agile software development is a software development approach that has received wide attention since the 1990s. Agile software development emphasizes efficiency and is at the core of people. As Jim Highsmith said, agile method allows user requirements to change at any time (Highsmith, 2002). In agile software development, the software project is divided into many parts and is iteratively developed by individual developers. It requires that the number of developers is not a lot and focuses on communication.

“Agile processes are designed to capitalize on each individual and each team’s unique

strengths” (Cockburn & Highsmith, 2001). Agile development solves the problem of coordination among developers. Every developer's work will not be influenced by other people and will not rely on others. This allows each developer in the team to work efficiently (Dingsøyr et al, 2017).

Agile software development does not require a clear understanding of the system being developed up-front. Developers can start with a little planning and in-depth details of a whole system are not required. Small functions that are divided from software projects are allowed to be developed first and any changes in each function do not affect the other parts.

Advantages

- Developers, testers and customers do not interact with each other
- High work efficiency
- Easy to change
- Good design caused by timely feedback from users

Disadvantages

- Lack of necessary documents
- If the project is too large or too much, the final result may not be the same as expected.
- It is hard to communicate if the number of teams is too large

Requirements and analysis

The goal of this project is to develop an airline reservation system to address issues such as international flight booking and flexible route planning. As a multiplayer team project, it is divided into 5 subsystems – User profile, Booking, Scheduling, Rescheduling and Payment subsystem, which is the responsibility of five people with no need of integrating. The subject of this development and research, and the scope of the report, is the scheduling subsystem in the airline reservation system. To make it easier for readers to understand the content of this project, this section will describe and analyze the requirements of the scheduling system, and uses a use case diagram to indicate what has been done with the entire subsystem.

3.1 Requirements analysis

In terms of the scheduling system, in addition to the display of flight information and search function, there are other functions needed to simplify the operating of this system. In this regard, potential features should be researched and analyzed. A simple questionnaire is proposed to obtain data for supporting the feasibility of these functions. The total number of people surveyed is 26. Due to certain limitations, the number of people in this survey is small, meaning that data is one of the references. Specific questionnaires and results can be found in Appendix C.

Since Booking and Rescheduling subsystem is the responsibility of others, this study will not include additional functions such as booking, rescheduling, aeroplane delays and route changing. However, in order to test the main function and the compatibility of the entire system, basic functions of User Profile, Booking and Rescheduling subsystem are listed in requirements.

In the survey, 61.53% of people think that existing apps still need a lot of time to screen proper tickets, while fewer alternatives for some routes are proposed by 46.15% of them. Also, 65.38% of people believe that they often miss out on proper tickets. Based on the above data, certain solutions have been initially determined, such as recommendation system, route combination function, etc.

In addition to questionnaires, more effective method to understand user requirements of this type of system is to investigate similar apps. The survey selects two targets: Expedia and Trip.com.

Expedia.com

Expedia.com is a travel booking website where users can book air tickets, hotels, etc. In its ticket reservation system, users are required to enter many screening conditions, such as departure time, direct flight or not and class, etc. The input of these conditions allows users to find the tickets they want more precisely and quickly, but the operation is too cumbersome. In other words, the system does not look that smart. The fuzzy search function in the website's ticket search feature, not only prompts airports by letter but also indexes by distance. The advantage of this is that users no longer need to go online to check the names of the surrounding international airports, just enter their locations. The system will automatically prompt the nearest international airport to users. For instance, if Sheffield without an international airport is entered, then the system will prompt Manchester Airport for the user. However, the problem is that the system does not have a pop-up window to choose the airport when the mouse focus is on the airport input box, which means that the user must type at least a few letters before the prompt appears instead of being prompted at the beginning of the input.

The site's ticket searching results are simply arranged by fare. Searching for international flights from Manchester to Beijing, for example, the top of the list are all flights, with low fares but with a flight time of up to 32 hours and two transfers. If a user wants to find flights with short layovers and cheaper price, it will be time-consuming. To compensate for this, the website has set up a number of filters for the user to screen, allowing users to find suitable tickets effectively.

Trip.com

Trip.com, just as Expedia.com, is a travel booking website. Besides functions similar to Expedia.com, it can combine routes of different airlines and mark them out. The lowest fares and the shortest direct flights are also marked out, which makes it easy for users to find a target flight.

Its dashboard is still tedious and requires a large number of conditions for user to enter for searching. However, it has a recommendation system, instead of simply ranking tickets based on fares, which means that users can find the most suitable flight conveniently. The importance of recommendation system lies in the fact that not all users are willing to look through a few pages of flight information and enter a bunch of filters. There are some intelligent recommendation algorithms that enables the system to understand and attract users.

Results

According to the above survey and analysis, in addition to the basic login, registration, booking, refund and search functions, the system also needs the following functions.

- **Optimizing the search function**
- **Conditional search:** in the dashboard (or home page), users do not need to enter a large number of criteria for ticket searching, and the filtering of the criteria will be placed on the page after the search.
- **Fuzzy search:** users can search for airports or flights with city name instead of airport name. as the number of international airports is not immensely large, pop-up windows will appear for users to select the city when the mouse focus is in the input box. The spelling mistake will also be corrected automatically to give the right results.

- **Filters:** filters will refilter and sort out existing search results, rather than research the data in the database, which speeds up the operation.
- **Queuing function:** Due to the limited amount of tickets for flights, many users think they miss out on suitable tickets. Meanwhile, refunded tickets will be resold, because many users have to refund tickets for changed itinerary. If a user wants this ticket, queuing function allows the user to enter the ticket queue, and buys the refunded tickets for the user automatically once the refunded ticket appears. The priority of purchase should base on the time of entry.
- **Scheduling system:** Scheduling system needs to show the ranking of tickets intelligently, understand users' preferences and recommend a ticket with high possibility of purchasing, which not only saves time for users, but also increases the rate of buying tickets. The scheduling system will draw conclusions relying on a series of algorithms and a benchmarking system. This is the difficulty of the project, which will be covered detailly in the following statement.
- **Map visualization:** Flights and airports will be visualized on a world map, allowing users to choose flights directly by clicking on the coordinates on the map. Clicking on the airport coordinates will pop up he specific information about the airport, and the lowest fares from the nearest airport to this one. Clicking on the airport name or picture on the pop-up window again will show all flights. The departure place of flights is determined by the user's current location.

3.2 Requirements Definition

The MoSCoW method indicates the meaning of Must have, Should have, Could have and Won't have. Considering pronunciation, "O" is put in the center. This name was put forward by Dai Clegg initially (Clegg, 1994). To reach an agreement with stakeholders toward the great value installed by them about requirement's handing over, the MoSCoW method, being named as MoSCoW prioritization or MoSCoW analysis, becomes the preferential technology for managing, business analyzing, project management and software development;

Regarding long-term solutions, several requirements are distributed into two parts, including function requirements and non-functional requirements. With regard to the first one, they will be utilized in the building of a basic framework to assist functions like computation or data operation. Concerning non-functional requirements, they however prefer to consider application performance and using. Moreover, the development of the application has been distributed into small tasks effectively using MoSCoW method. Each task owns certain technical goal and enables developers to allocate resources, time and efforts to vital requirements of projects.

3.2.1 Functional Requirements

The application **MUST**:

- Require an internet connection
- Require users to be logged in before purchasing tickets
- Require users to enter departure city, arrival city and departure date
- Allow users to log in/register
- Allow users to view their ticket purchasing history
- Allow users to search for fights
- Allow users to book tickets
- Allow users to refund tickets
- Allow users to view maps
- Show users specific flight information
- Label and distinguish between direct and connecting flights
- Label and distinguish different classes of travel
- Combine flights with different airlines

The application **SHOULD**:

- Allow users to filter tickets using filters
- Allows users to sort tickets with different criteria
- Automatically put users in the ticket queue, and buy tickets when others refund sold-out ticket
- Recommend good flights based on user's behavioral record
- Recommend the right flight to new users with no behavioral records
- Allows users to search for tickets by map coordinates

The application **COULD**:

- Support multiple languages and allow users to set up languages
- Allow users to change passwords

The application **WON'T**:

- include payment system
- include hotel reservation system and car rental system

3.2.2 Non-Functional Requirements

The precedence of Non-Fictional requirements normally utilize produced application rather than function, so does MoSCoW prioritization technology.

The System **Must**:

- Run without error
- Guaranteed fast running/calculation speed
- Plan the servlet pool and database pool properly
- Encrypting users' password
- Allow multiple users to access at the same time
- Strong robustness, no obvious loopholes
- Deny unlogged users' access to the user's personal interface and ticket booking

The System **Should**:

- Use different recommendation algorithm combinations to obtain the optimal solution
- Defend against write-in of malicious JavaScript

The System **Could**:

- Defend against injection of SQL
- Keep up-to-date

The UI **Must**:

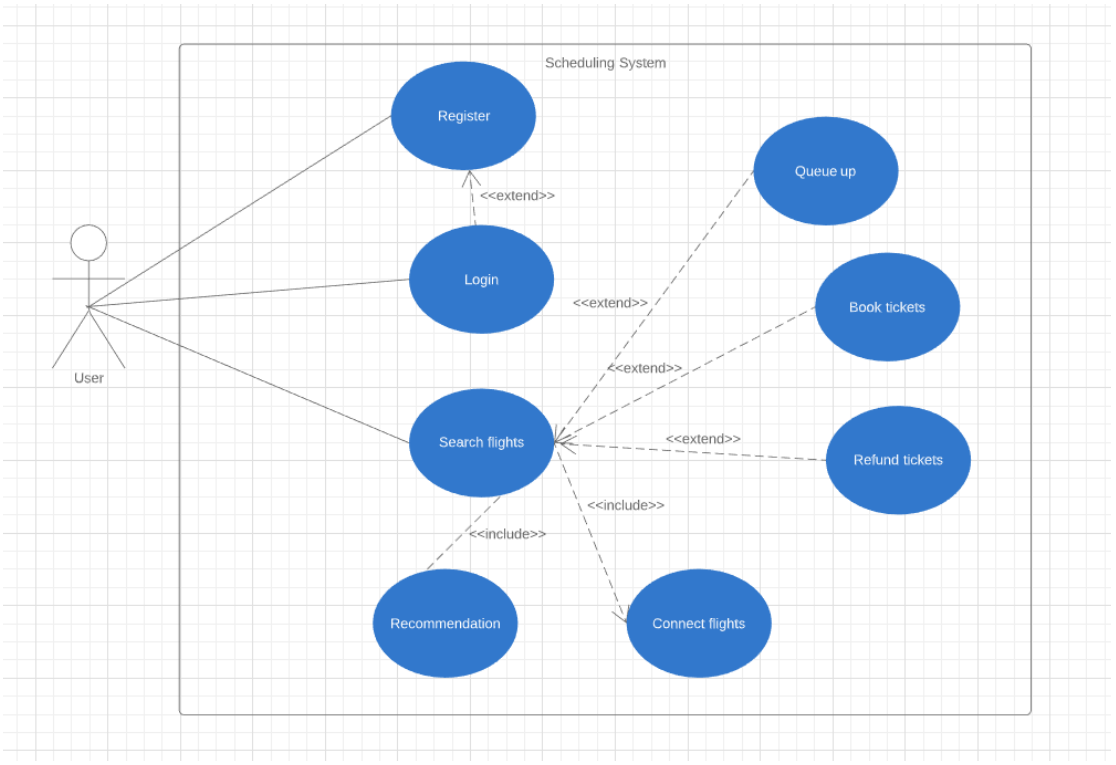
- Support responsive layout, different resolutions and varying devices
- Clear layout, allowing users to operate independently without relying on any instructions
- Include map visualization

The UI **Should**:

- Delicate styles and layouts
- Dynamic windows and buttons
- Consistent style and theme

3.3 Use Case Diagram

A use case diagram can mirror the relationship between use cases and users directly and clearly in the system. Requirements and use cases of the scheduling system are defined below.



Design

The system needs to be designed before development. This section will specify technologies that will be used, based on previous literature review and analysis. UI design, database design will also be included in this section. UML class diagram will be applied to define certain class and interface.

4.1 Programming Language and Web Technology

As the primary development language for this project, Java will be responsible for back-end development and work as a Servlet. The front-end pages will be developed with the combination of HTML, CSS and JS. The HTML page will be changed into JSP page when the front-end and back-end are consolidated. Since JSP will be compiled into Servlet written in Java code before eventually compiling into binary code, each JSP pate can be regarded as the small Servlet. JSP statements can insert Java code into HTML pages to dynamically display the page, which however seems to be outdated. This project thus will utilize JSTL (JavaServer Pages Standard Tag Library) for background data display and some simple logical relationship processing. JavaScript will be applied to develop more dynamic page effects.

Reasons for choosing Java are listed below:

- Java is a kind of strong type language, which will not cause memory conflicts.
- Java has many full-fledged web frameworks.
- Java is easier to maintain and scalable than scripting languages such as PHP, although its development is relatively slow.
- High code reusability
- Thread pool and connect pool of Java is convenient for managing, and this project needs frequent database operation and asynchronous operation.

4.2 Framework

The back-end development of this project will adopt SSM framework – Spring+ SpringMVC+MyBatis. Spring's features of controlling inversion and dependency injection can help developers to manage projects effectively, which allows developers to focus more on developing logic and algorithms. Besides, with Spring's AOP feature, it is easy to control transactions, logs and permissions. SpringMVC, shares the same feature of high cohesion and low coupling with other MVC, and is often used with Spring. MyBatis will be applied to develop Data Access Object (DAO) speedily, and all SOL statements are written in the XML file, which decouples from the code.

The reasons for choosing SSM framework are as follows:

- Give dependencies between objects to Spring, which is convenient for decoupling.
- Easy control of transactions, logs and permissions.
- Provide integrated support for other excellent open-source frameworks
- SpringMVC's feature of high cohesion and low coupling.
- High code reusability, avoiding a lot of duplicate code
- Flexibility of MyBatis
- MyBatis can quickly develop DAO and map POJO (Plain Ordinary Java Object)

About the front-end framework, Bootstrap, an open-source kit proposed by Twitter, will be used, which provides a number of callable styles to help developers save time on page aesthetic design. More importantly, Bootstrap supports the responsive layout and adaptive resolution features, enabling pages developed to be compatible on both PC side and phone side. This feature is necessary for an airline reservation system because users often use mobile phones to view the trip.

Reasons for selecting Bootstrap are as follows

- Bootstrap offers well-developed styles

- The front-end developed with Bootstrap can be self-adaptable in devices of various solutions.

4.3 IDE (Integrated Development Environment)

IntelliJ IDEA is commonly recognized as one of the best Java development tools. Due to the fact that it supports multi-module development, it is more suitable for the development of this project than Eclipse. The reason is that this project is more scalable (e.g. many features can be added and maintenance will be needed in the future). Thus, IntelliJ IDEA, which supports multimodule development, is more suitable as the IDE for this project.

This project, as a subsystem in this project, will be integrated into the entire system in the future, even though integration is not currently required.

4.4 RDBMS (Relational Database Management System)

MySQL is a better choice of RDBMS this project is an open-source, multi-platform and optimized SQL of MySQL. In terms of the version selection, 5.7.20 is selected for this subject.

4.5 Server

Web development projects need to be deployed on the server for testing. Tomcat server is a free and open-source web application server that is a lightweight application one, which is commonly used in many situations, such as small and medium-sized systems and a few concurrent access users. It is the optimal option for developing and debugging JSP program. In this project, the Tomcat version used is 8.53.

4.6 Methodology

The agile development method is believed to be the most preferred choice, concerning various contexts of the project. As the waterfall model have to string along the sequence of planned requirements, analysis, designing, coding and testing, the flexibility of it and the whole project is decreased, which explains why the waterfall model is ruled out. Oppositely, the agile method concerns the inchoate application of existed functions and keeps on changing in the whole project cycle. The disorientation in the project’ s requirements makes the agile development a suitable method.

Furthermore, with regard to the length of the development cycle, agile development has a competitive edge over the incremental and spiral models, since its cycle is shorter than either of them, meaning that it can complete more functions with a shorter period.

Regarding the project, agile software developing principles include multiple stages

1. Requirements definition: pivotal depicting of the anticipated prototype

2. Design: decide the software structure and devise the user interface

3. Development: develop with suitable tools relying on defined requirements and architecture

4. Testing: to test whether all functions have been adopted smoothly in the software developed.

The sequence of chapters is flexible as the project keeps to the agile development approach. The alterations of requirements will lead to modifications in the design and development process.

4.7 Algorithm

In accordance with previous literature review and analysis, each recommendation algorithm has its pros and cons, so the recommendation system will be implemented with Hybrid Recommendation. The principle of content-based recommendation algorithm is that features and keywords of item will be extracted, whose frequency of occurrence will also be calculated. However, lack of a great body of sample data and low frequency in features and keywords (limited number of air tickets) means that CB algorithm can never be utilized for accurate recommendation in this project. Also, it is unnecessary for CB to have features extracting procedure because feature of air tickets is relatively simple, basically including the quantifiable fare and flight time.

Therefore, this project will use the recommendation algorithm after co-filtering and apply a weighted method to combine the other algorithm. For the weight of both methods, an initial value will be given first, and then several groups of benchmarks are made to get the optimal weight. Among them, the similarity calculation in the CF recommendation algorithm will apply Pearson Correlation Coefficient.

In addition, this project will utilize the item-based CF instead of user-based CF algorithm. Reasons are as follows.

1. The sparseness of the user-based CF algorithm is more serious. Since the average user does not buy tickets on a daily basis while new flights are available every day, the listed matrix will be a sparse one, with most of the value at 0, when it is imperative to compare the similarity between two users. The results calculated in this situation are meaningless.

2. The item-based CF only compares the ticket bought by users and flight lists checked by users, because other flights without user data are not meaningful. However, the user-based CF will go through all users. It this is a real airline reservation system,

data of large amounts of user groups will increase the computation and burden the system.

3. The indefiniteness of users should be considered. Users' likes will change as time goes by. For instance, a user will buy direct flight next time, even though he or she bought a transferred flight this time, as the user might feel that transferring flight was troublesome. In this case, using the user's data for similarity calculation is unreasonable. Comparatively, item-based CF algorithm applies the data of items, which means that the flight data will not change.

However, both recommendation algorithms are undesirable for new users or for unlogged visitors. In this case, another algorithm thus is sought. The algorithm, called "initialisation recommendation algorithm", is designed only for this project other than a popular one. The principle is that the system sorts the ticket list based on the two most important attributes (fare and flight time) of the ticket. The lower the ticket price and flight time are, the better, so the lower the ticket price and flight time are, the higher the ranking becomes. However, the existence of two attributes means that the ranking based merely on numerical value is impossible. The Euclidean Distance from the point of fare and flight time (p,t) to the origin on the axis needs to be calculated. Different weights of two factors should also be given (the impact of fare and flight time in the user's mind is not equivalent). For the weight calculation, purchasing records of all users in the system are required.

The following are the working steps of the algorithm:

1. Callout all users' purchasing records to get the ticket price and flight time of each one, and project the component vector onto the coordinate axis. At this point, the angle of each vector represents the weight of fare and flight time among users.

2. Gain the mean by calculating the angle, and calculate average weight for the subsequent steps.

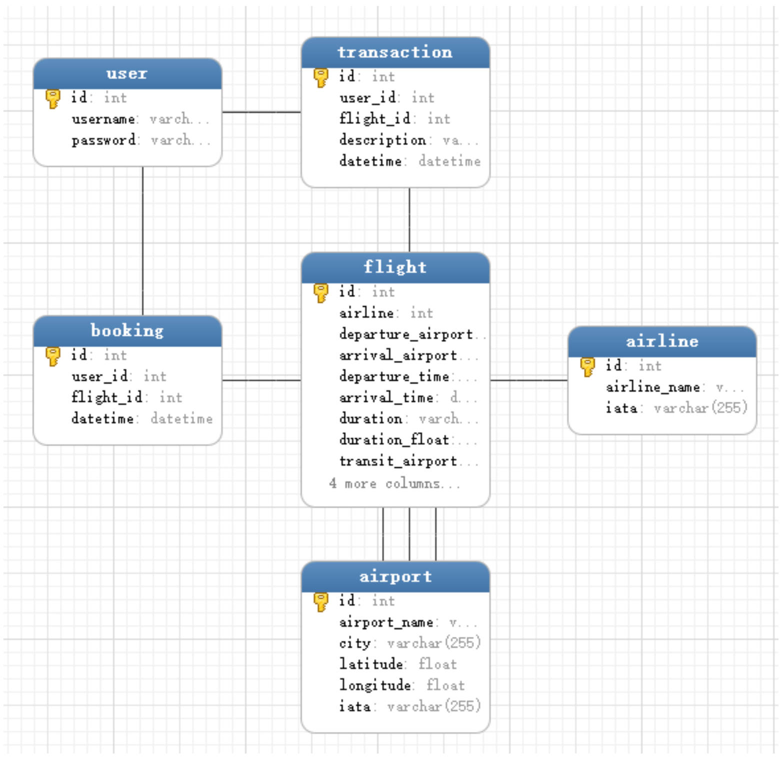
3. Calculate Euclidean Distance of each ticket queried by the current user, adding weights to the factor, with the following formula.

4. Sequence the controller in the system by Euclidean Distance of each ticket, in ascending order.

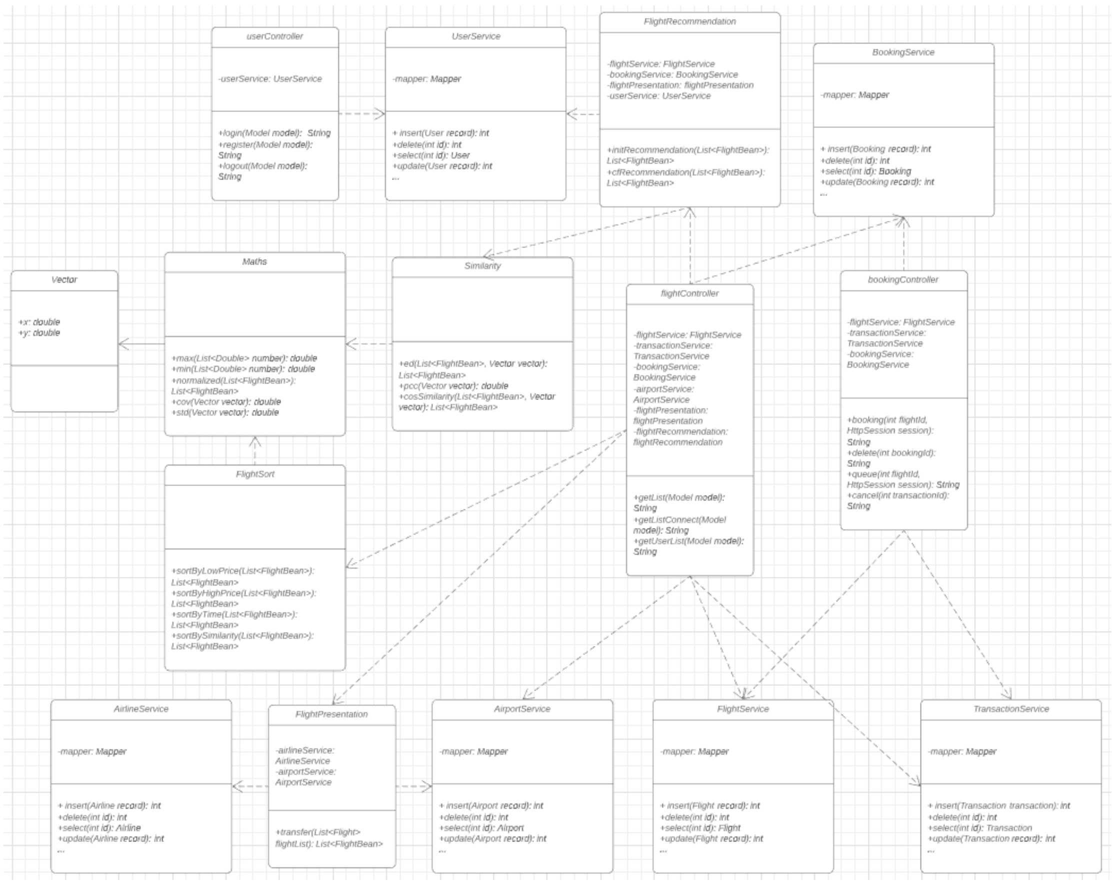
This algorithm, in conjunction with the Collaborative Filtering algorithm, also needs to be combined via weighting. The weights will be derived by the benchmark. The process will be described specifically in the implement and Testing section.

4.8 Database Design

The database design is shown in the E-R diagram below.



4.9 Class Diagram

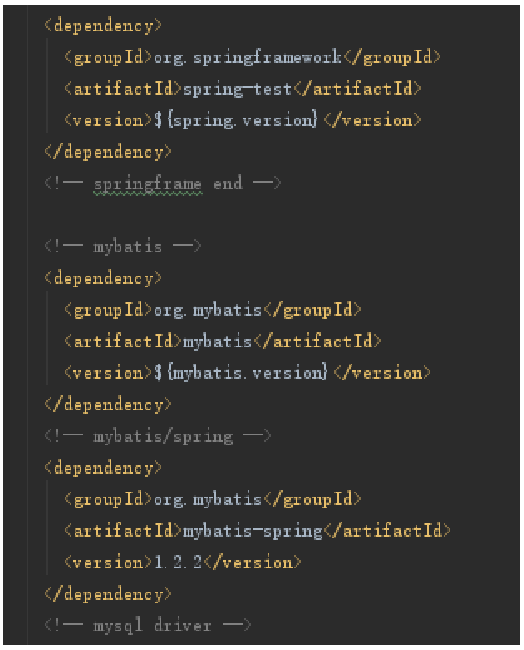

userController
-userService: UserService
+login(Model model): String
+register(Model model): String
+logout(Model model): String
UserService
-mapper: Mapper
+ insert(User record): int
+delete(int id): int
+select(int id): User
+update(User record): int
...
FlightRecommendation
-flightService: FlightService
-bookingService: BookingService
-flightPresentation: flightPresentation
-userService: UserService
+initRecommendation(List<FlightBean>): List<FlightBean>
+cfRecommendation(List<FlightBean>): List<FlightBean>
BookingService
-mapper: Mapper
+ insert(Booking record): int
+delete(int id): int
+select(int id): Booking
+update(Booking record): int
...
Vector
+x: double
+y: double
Maths
+max(List<Double> number): double
+min(List<Double> number): double
+normalized(List<FlightBean>): List<FlightBean>
+cov(Vector vector): double
+std(Vector vector): double
Similarity
+ed(List<FlightBean>, Vector vector): List<FlightBean>
+pcc(Vector vector): double
+cosSimilarity(List<FlightBean>, Vector vector): List<FlightBean>
flightController
-flightService: FlightService
-transactionService: TransactionService
-bookingService: BookingService
-airportService: AirportService
-flightPresentation: flightPresentation
-flightRecommendation: flightRecommendation
+getList(Model model): String
+getListConnect(Model model): String
+getUserList(Model model): String
bookingController
-flightService: FlightService
-transactionService: TransactionService
-bookingService: BookingService
+booking(int flightId, HttpSession session): String
+delete(int bookingId): String
+queue(int flightId, HttpSession session): String
+cancel(int transactionId): String
FlightSort
+sortByLowPrice(List<FlightBean>): List<FlightBean>
+sortByHighPrice(List<FlightBean>): List<FlightBean>
+sortByTime(List<FlightBean>): List<FlightBean>
+sortBySimilarity(List<FlightBean>): List<FlightBean>
AirlineService
-mapper: Mapper
+ insert(Airline record): int
+delete(int id): int
+select(int id): Airline
+update(Airline record): int
...
FlightPresentation
-airlineService: AirlineService
-airportService: AirportService
+transfer(List<Flight> flightList): List<FlightBean>
AirportService
-mapper: Mapper
+ insert(Airport record): int
+delete(int id): int
+select(int id): Airport
+update(Airport record): int
...
FlightService
-mapper: Mapper
+ insert(Flight record): int
+delete(int id): int
+select(int id): Flight
+update(Flight record): int
...
TransactionService
-mapper: Mapper
+ insert(Transaction transaction): int
+delete(int id): int
+select(int id): Transaction
+update(Transaction record): int
...

Implementation and testing

This section illustrates the whole project's procedure in depth, relying on the preceding survey and design findings. This chart will explain development process for the project, and it is also expected to finish testing program and demonstrating results. Thanks to the agile development method applied, this is a rational sequence. The code and the diagram of result will be shown partly in this chapter regarding the length. Specific display of results and testing table can be found in Appendix A and B.

5.1 Framework building and project structure.

The first step of the project is framework building, which uses SSM framework. Therefore, Spring, SpringMVC and JAR Packages of Mybatis should be imported. The process of importing JAR Packages can be hosted by Maven, as shown in Figure 5.1.

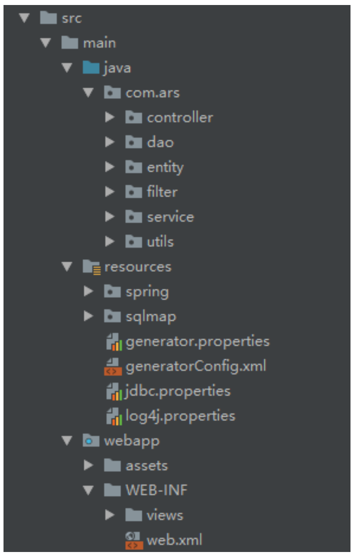


```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-test</artifactId>
  <version>${spring.version}</version>
</dependency>
<!-- springframe end -->

<!-- mybatis -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>${mybatis.version}</version>
</dependency>
<!-- mybatis/spring -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>1.2.2</version>
</dependency>
<!-- mysql driver -->
```

Figure 5.1

After importing the dependency of the required JAR packages in Maven's configuration file, Maven automatically imports these JAR packages into the project. All Maven dependencies come from MavenRepository.

The constructed project structure is shown in figure 5.2.



```
▼ src
  ▼ main
    ▼ java
      ▼ com.ars
        ▶ controller
        ▶ dao
        ▶ entity
        ▶ filter
        ▶ service
        ▶ utils
    ▼ resources
      ▶ spring
      ▶ sqlmap
        generator.properties
        generatorConfig.xml
        jdbc.properties
        log4j.properties
    ▼ webapp
      ▶ assets
      ▼ WEB-INF
        ▶ views
          web.xml
```

Figure 5.2

- **controller:** Controller files, logic for connecting the front-end and back-end, and of scheduling business.
- **dao:** interface to database operation
- **entity:** POJO, object mapping to the database, containing get and set methods.
- **filter:** Used to filter some access, such as part of pages that are not allowed to access without logging in.
- **service:** Service files, belonging to the business layer, and implementing specific business requirements.
- **utils:** Tools, include Maths, Similarity and Vector which contain function of supporting math calculation, also have FlightPresentation, FlightRecommendation and FlightSort which are responsible for logic of recommendation system and data display.
- **spring:** Spring's configuration file, applicationContext.xml, all spring beans will be declared here. Spring dynamically creates and allocates memory for declared controllers, services, components, etc. This is called control inversion, in which the programmer gives control of allocating memory to spring.
- **sqlmap:** Mybatis mapping file, which can be interpreted as an xml form of sql.
- **assets:** static file at the front-end, such as images, CSS and JavaScript files
- **views:** front-end JSP file, the UI of the project.

In the utils class, Maths class contains the methods that calculate the maximum and minimum, normalization, covariance and standard deviation. The similarity class includes the calculation methods of Euclidean Distance, Pearson Correlation Coefficient and Cosine Similarity respectively. In FlightSort is the bubble sorting algorithm that is sorted by different criteria. FlightRecommendation includes two recommendation algorithms. FlightPresentation is a character used to transform data from a database into visualization. For example, airlines and airports stored in flight table are both ids which need to be converted into the name before being transmitted to the front end.

Moreover, there are other configuration files such as generator.properties and generatorConfig.xml that are used to configure the plug-in of myBatis generator configuration files. Jdbc. Properties and log4j.properties are properties of some jdbc and logs, such as the username and password of database. If this project is debugged on other devices, jdbc.properties need to be checked and changed because the username and password of database might be different.

5.2 CURD

CURD(Create, Update, Retrieve and Delete) is the foundation of a web project, and the back-end operates on the database relying on the users' actions at the front-end. In this project, CURD is based on the MyBatis framework, which works by reading xml mapping files written by programmers, and build up SqlSessionFactory (SqlSessionFactory is thread-safe). Subsequently， SqlSessionFactory's instance will open a SqlSession directly. It then will gain the Mapper object through SqlSession instance, and run the SQL statement of Mapper mapping for completing the CRUD and transaction committing to the database, and finally close the SqlSession.It must be noted that SqlSession, as a single-threaded object, is non-thread-safe and the exclusive object of persistent operation. Similar to the Connection in jdbc, the underlying of SqlSession encapsulates the jdbc Connection.

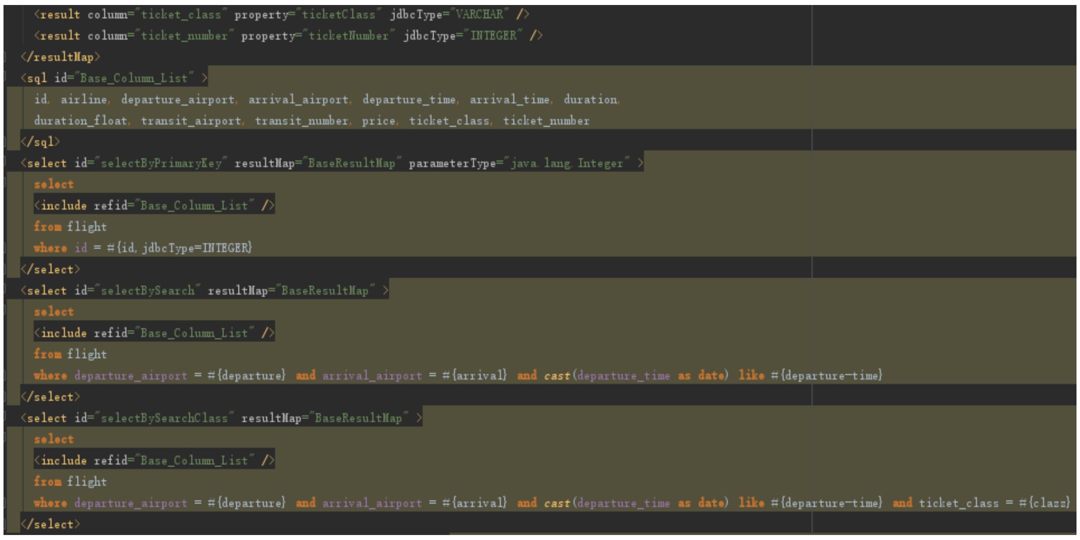
The detailed workflow is as follows:

- Load mybatis global configuration file (data source, mapper mapping files, etc.), parse configuration files, MyBatis generates Configuration based on XML configuration files, and each mappedStatement (including parameter mapping configurations, dynamic SQL statements, result mapping configurations), which correspond to <select | update | delete | insert>Label items (Figure 5.3).
- SqlSession Factory Builder generates SqlSession Factory through Configuration objects to turn on SqlSession.
- SqlSession objects complete interaction with the database:
    a) The user program calls the mybatis interface layer api (i.e. the method in the Mapper interface).
    b) SqlSession finds corresponding MappedStatement object via calling api's Statement ID.

c) The MappedStatement object is parsed, the sql parameter is conversed, dynamic sql is stitched and jdbc Statement object is generated with Executor (the generation of dynamic SQL and maintenance of the query cache).

d) JDBC executes sql.

e) Transform the returned results into HashMap, JavaBean, etc. storage structures with the result mapping relationship of MappedStatement, and then return.

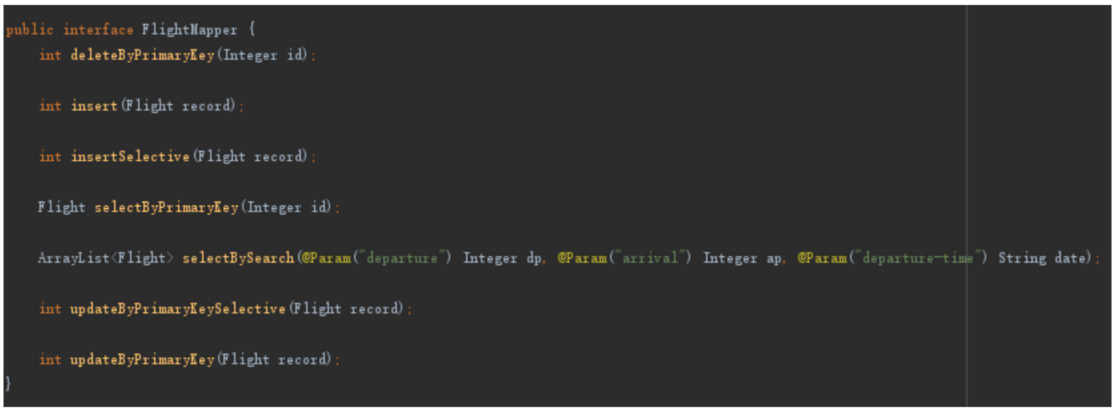


```
    <result column="ticket_class" property="ticketClass" jdbcType="VARCHAR" />
    <result column="ticket_number" property="ticketNumber" jdbcType="INTEGER" />
  </resultMap>
  <sql id="Base_Column_List" >
    id, airline, departure_airport, arrival_airport, departure_time, arrival_time, duration,
    duration_float, transit_airport, transit_number, price, ticket_class, ticket_number
  </sql>
  <select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Integer" >
    select
    <include refid="Base_Column_List" />
    from flight
    where id = #{id,jdbcType=INTEGER}
  </select>
  <select id="selectBySearch" resultMap="BaseResultMap" >
    select
    <include refid="Base_Column_List" />
    from flight
    where departure_airport = #{departure} and arrival_airport = #{arrival} and cast(departure_time as date) like #{departure-time}
  </select>
  <select id="selectBySearchClass" resultMap="BaseResultMap" >
    select
    <include refid="Base_Column_List" />
    from flight
    where departure_airport = #{departure} and arrival_airport = #{arrival} and cast(departure_time as date) like #{departure-time} and ticket_class = #{clazz}
  </select>
```

Figure 5.3

For instance, selectBySearch in the figure searches the database for eligible flights based on the departure city, arrival city and departure date entered by the user, which will return a collection in the form of Flight JavaBean.

As shown in Figure 5.4, the SelectBySearch interface is declared in the FlightMapper interface under dao directory, which is implemented at runtime. It receives the result set returned by Mybatis using ArrayList <Flight> type. This interface will then be called by the Service layer which will be called by the Controller, and finally the visual result will be passed to the front end and displayed.



```
public interface FlightMapper {
    int deleteByPrimaryKey(Integer id);

    int insert(Flight record);

    int insertSelective(Flight record);

    Flight selectByPrimaryKey(Integer id);

    ArrayList<Flight> selectBySearch(@Param("departure") Integer dp, @Param("arrival") Integer ap, @Param("departure-time") String date);

    int updateByPrimaryKeySelective(Flight record);

    int updateByPrimaryKey(Flight record);
}
```

Figure 5.4

5.3 Front-end

According to previous UI design, HTML+CSS+JavaScript and BootStrap will be utilized to develop the front-end. The difficulty is the calling of Google Maps API (as shown in Figure 5.5). This requires learning of Google Maps Platform's JS interfaces

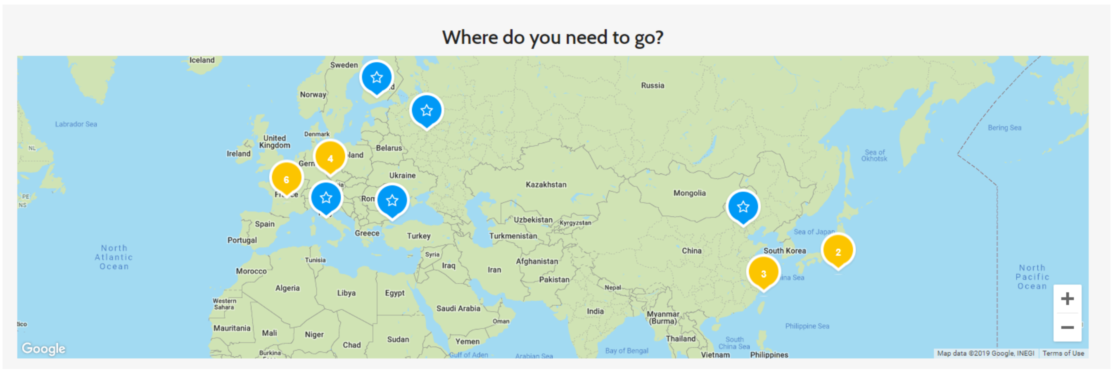


Figure 5.5

All airports in the database are marked on the map, and the nearby airport is shrunk into a separate yellow marker, which can enlarge the map and turn it to blue with a click. Also, city information and the lowest fare from the nearest airport to the area will pop up by clicking on marker (as shown in Figure 5.6), clicking on the airport name or logo automatically searches for flights from nearby airports.

See appendix A for more details and results for operating.

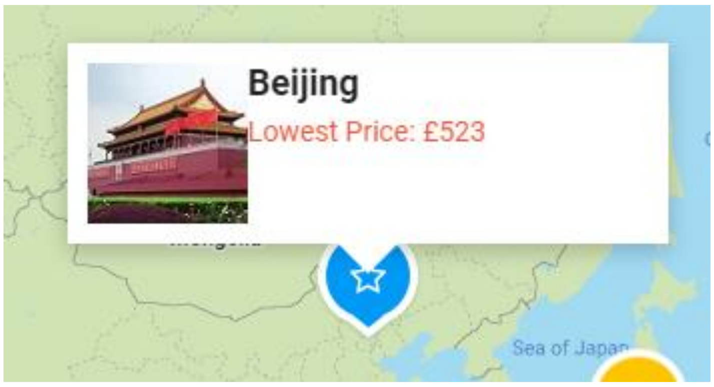


Figure 5.6

The actual implementation of this function is to write the longitude and latitude of the marker, icons, and pop-up content (HTML statement) in the form of JSON, before calling the method in the API to generate map. The latitude and longitude of the airport can be queried from the database and passed through ajax to JavaScript's function.

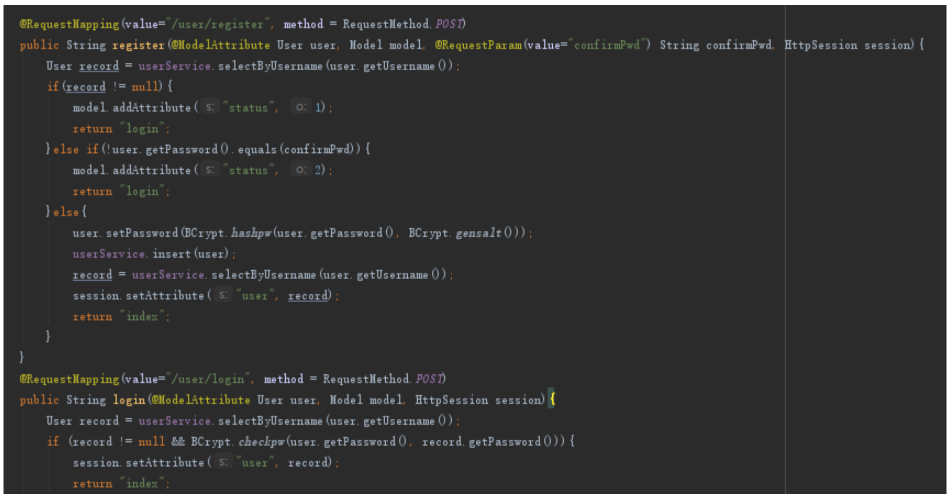
5.4 Back-end

The logical implementation of back-end is the focus of the project. This section will lay emphasis on logic rather than retell the content of the framework and CURD. This section will describe separately in functional modules, but not all will be mentioned considering the length.

5.4.1 User-related features

User's related features include login, register, logout and check users' tickets, which are some basic functions instead of the core for the project. However, the core code needs to call them, which means that their implementation is imperative.

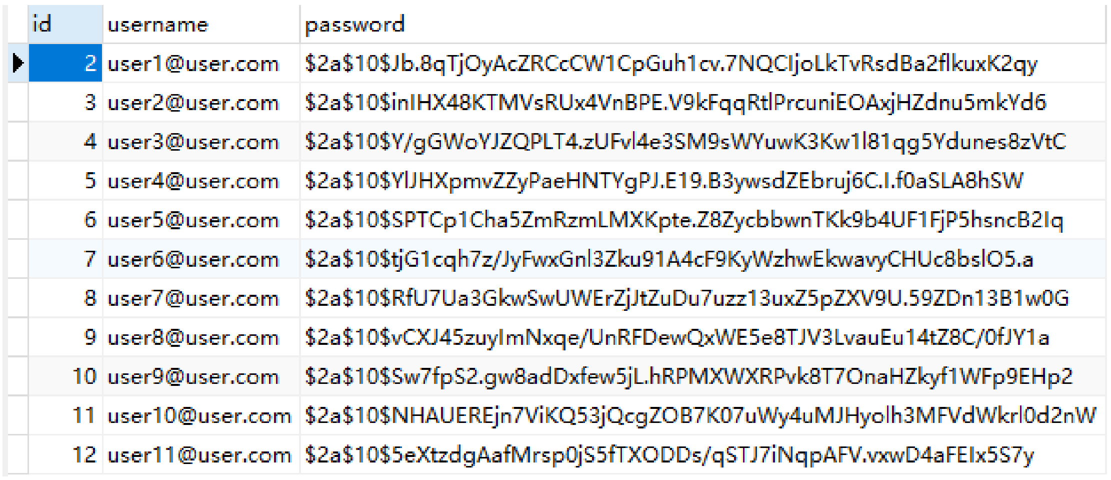
User-related features are controlled by userController, as shown in Figure 5.7.



```java
@RequestMapping(value="/user/register", method = RequestMethod.POST)
public String register(@ModelAttribute User user, Model model, @RequestParam(value="confirmPwd") String confirmPwd, HttpSession session){
    User record = userService.selectByUsername(user.getUsername());
    if(record != null){
        model.addAttribute( s: "status",  o: 1);
        return "login";
    }else if(!user.getPassword().equals(confirmPwd)){
        model.addAttribute( s: "status",  o: 2);
        return "login";
    }else{
        user.setPassword(BCrypt.hashpw(user.getPassword(), BCrypt.gensalt()));
        userService.insert(user);
        record = userService.selectByUsername(user.getUsername());
        session.setAttribute( s: "user", record);
        return "index";
    }
}
@RequestMapping(value="/user/login", method = RequestMethod.POST)
public String login(@ModelAttribute User user, Model model, HttpSession session){
    User record = userService.selectByUsername(user.getUsername());
    if (record != null && BCrypt.checkpw(user.getPassword(), record.getPassword())){
        session.setAttribute( s: "user", record);
        return "index";
```

Figure 5.7

The implementation of this part relies on basic CURD. User password adopts BCrypt encryption. Since BCrypt is a one-way encryption technology, it cannot be compared by decryption intelligence, which conforms to the fact that the password does not need to be displayed in plain text with only the comparison feature. The encrypted password is shown in Figure 5.8. 11 users' password is all "123456", but the password text is completely different.

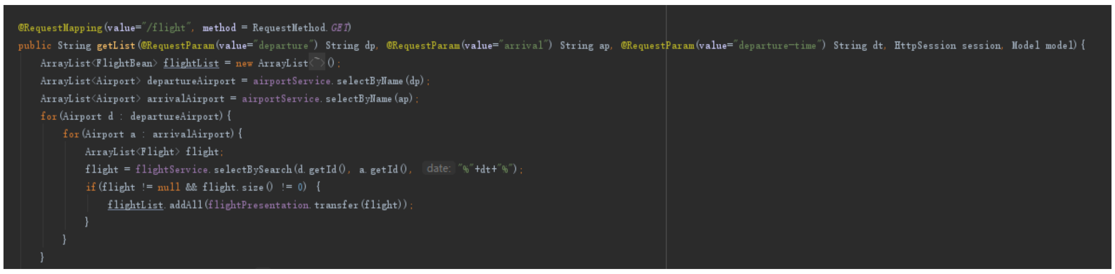


id	username	password
2	user1@user.com	$2a$10$Jb.8qTjOyAcZRCcCW1CpGuh1cv.7NQCIjoLkTvRsdBa2flkuxK2qy
3	user2@user.com	$2a$10$inIHX48KTMVsRUx4VnBPE.V9kFqqRtlPrcuniEOAxjHZdnu5mkYd6
4	user3@user.com	$2a$10$Y/gGWoYJZQPLT4.zUFvl4e3SM9sWYuwK3Kw1l81qg5Ydunes8zVtC
5	user4@user.com	$2a$10$YlJHXpmvZZyPaeHNTYgPJ.E19.B3ywsdZEbruj6C.I.f0aSLA8hSW
6	user5@user.com	$2a$10$SPTCp1Cha5ZmRzmLMXKpte.Z8ZycbbwnTKk9b4UF1FjP5hsncB2Iq
7	user6@user.com	$2a$10$tjG1cqh7z/JyFwxGnl3Zku91A4cF9KyWzhwEkwavyCHUc8bslO5.a
8	user7@user.com	$2a$10$RfU7Ua3GkwSwUWErZjJtZuDu7uzz13uxZ5pZXV9U.59ZDn13B1w0G
9	user8@user.com	$2a$10$vCXJ45zuyImNxqe/UnRFDewQxWE5e8TJV3LvauEu14tZ8C/0fJY1a
10	user9@user.com	$2a$10$Sw7fpS2.gw8adDxfew5jL.hRPMXWXRPvk8T7OnaHZkyf1WFp9EHp2
11	user10@user.com	$2a$10$NHAUEREjn7ViKQ53jQcgZOB7K07uWy4uMJHyolh3MFVdWkrl0d2nW
12	user11@user.com	$2a$10$5eXtzdgAafMrsp0jS5fTXODDs/qSTJ7iNqpAFV.vxwD4aFEIx5S7y

Figure 5.8

5.4.2 Search/Fuzzy Search

As for the scheduling system, the most vital and basic function is the Search/Fuzzy Search, which needs to filter out eligible flights based on the criteria entered by the user. The realisation of search is shown in Figure 5.9.



```
@RequestMapping(value="/flight", method = RequestMethod.GET)
public String getList(@RequestParam(value="departure") String dp, @RequestParam(value="arrival") String ap, @RequestParam(value="departure-time") String dt, HttpSession session, Model model){
    ArrayList<FlightBean> flightList = new ArrayList<~>();
    ArrayList<Airport> departureAirport = airportService.selectByName(dp);
    ArrayList<Airport> arrivalAirport = airportService.selectByName(ap);
    for(Airport d : departureAirport){
        for(Airport a : arrivalAirport){
            ArrayList<Flight> flight;
            flight = flightService.selectBySearch(d.getId(), a.getId(), date: "%"+dt+"%");
            if(flight != null && flight.size() != 0) {
                flightList.addAll(flightPresentation.transfer(flight));
            }
        }
    }
```

Figure 5.9

As shown in the figure, as there is only the airport id without city field in the flight table in the database, and users normally enter the city name instead of entering an airport name, the system needs to automatically query the airport id associating with the city entered by the user from other tables. The whole search process is listed below.

Search airport from the airport table, the condition is that the city airport locates equals to the city entered by the user

Search eligible flights on the basis of the available departure airport id, arrival airport id and date. Fuzzy search should be applied for querying date because the flight time is accurate to minutes and users only enter the date.

Convert found flight information into a visual string.

The function of fuzzy search is also reflected in the front-end, as Figure 5.10 shows.

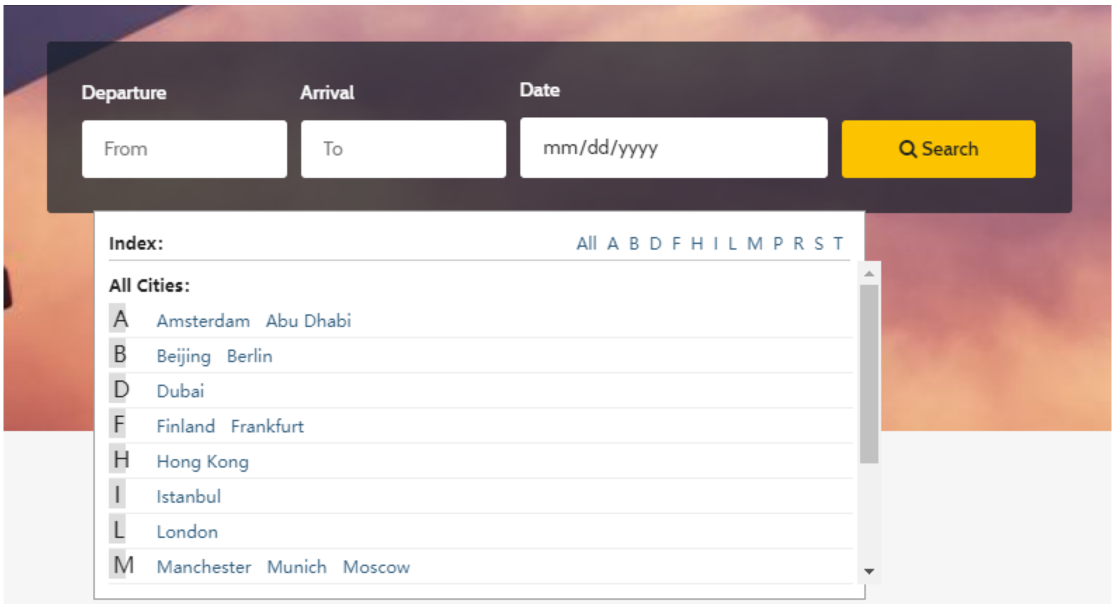


Figure 5.10

Since the number of cities owning international flights is limited, it is possible to give users tips with pop-up windows written by JavaScript when users are entering them. Just clicking on the city name is equivalent to entering the full name, which avoids potential spelling mistakes when the user enters. In addition, as the SQL statement for querying city names is like statement, the right city can still be queried even with spelling errors, so that the effect of fuzzy search can be realized.

5.4.3 Filter/Sort

When users are not satisfied with the order and filtering of the flight list, the system provides user with the ability to manually filter and sort, as shown in Figure 5.11.

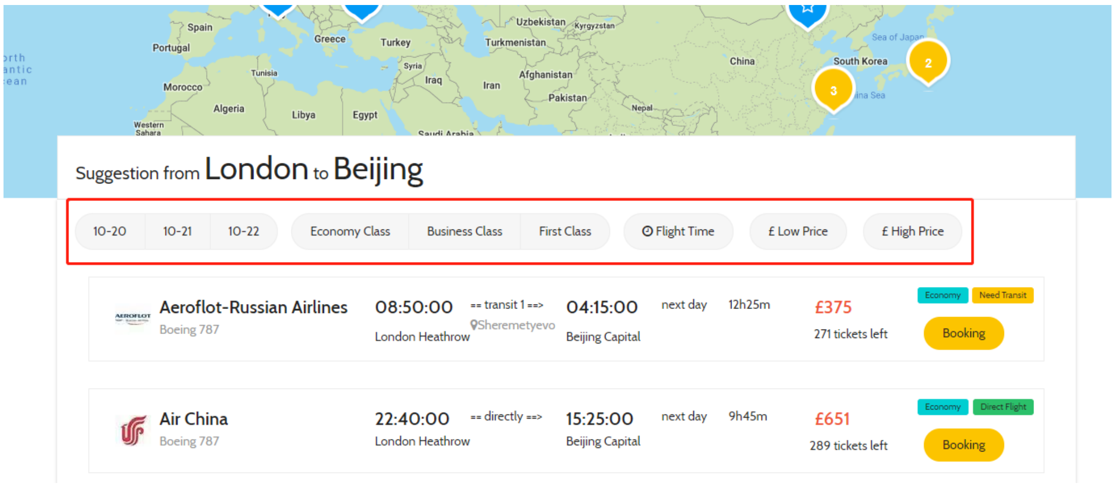
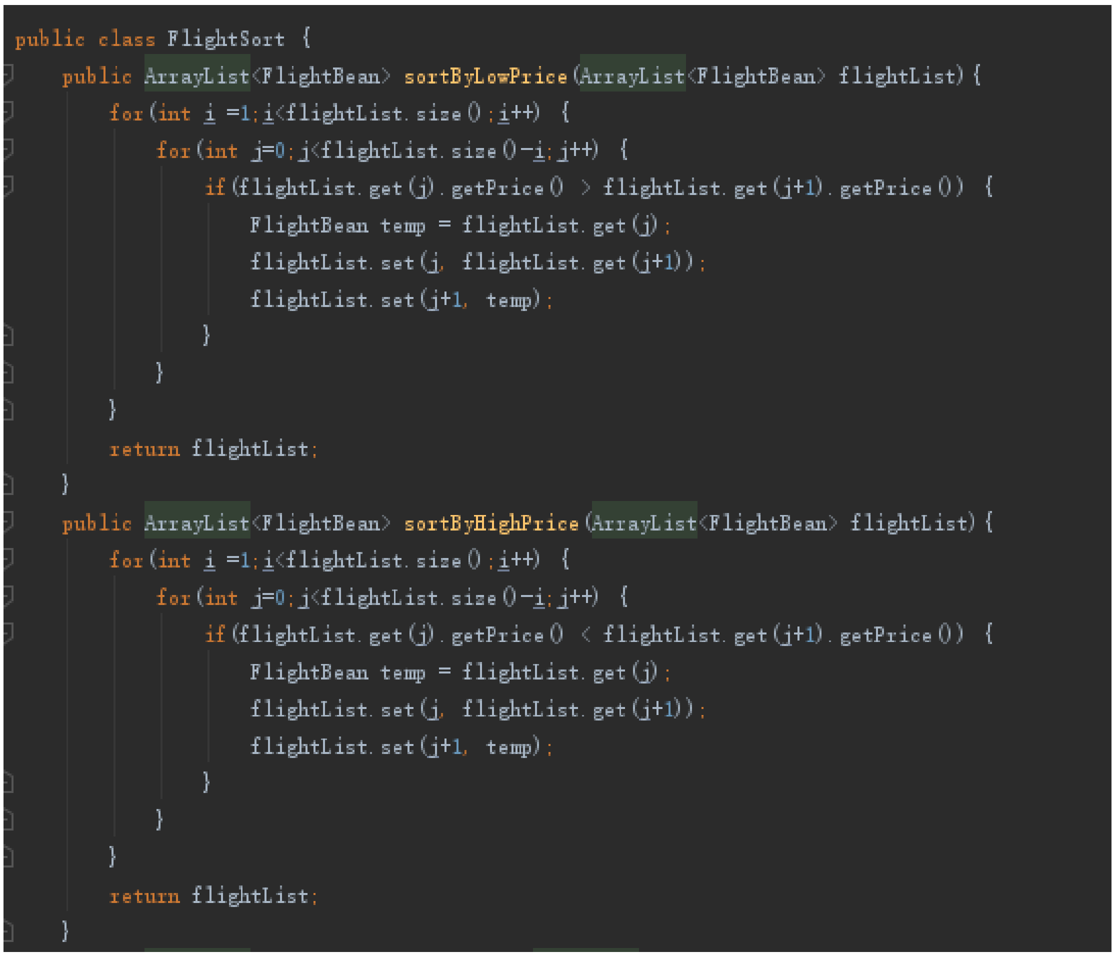


Figure 5.11

This row button above the flight list, can adjust the date, class, time and price of flight respectively. Since the testing data in database only contains three days of data, only up to three-day adjustment can be conducted. Users can choose to display Economy Class, Business Class and First Class only, and can select to sort flights by time, ascending price or descending price. The practical realization is to bubble sort the queried, stitched and transferred flight list with methods in the Flightsort tool class, as shown in Figure 5.12.



```
public class FlightSort {
    public ArrayList<FlightBean> sortByLowPrice(ArrayList<FlightBean> flightList){
        for(int i =1;i<flightList.size();i++) {
            for(int j=0;j<flightList.size()-i;j++) {
                if(flightList.get(j).getPrice() > flightList.get(j+1).getPrice()) {
                    FlightBean temp = flightList.get(j);
                    flightList.set(j, flightList.get(j+1));
                    flightList.set(j+1, temp);
                }
            }
        }
        return flightList;
    }
    public ArrayList<FlightBean> sortByHighPrice(ArrayList<FlightBean> flightList){
        for(int i =1;i<flightList.size();i++) {
            for(int j=0;j<flightList.size()-i;j++) {
                if(flightList.get(j).getPrice() < flightList.get(j+1).getPrice()) {
                    FlightBean temp = flightList.get(j);
                    flightList.set(j, flightList.get(j+1));
                    flightList.set(j+1, temp);
                }
            }
        }
        return flightList;
    }
```

Figure 5.12

5.4.4 Recommendation Algorithm

Before the user manually adjust the sort, the system will give a default sorting scheme. This sort scheme merely relying on price or time is not flawless, because the lower the fare and flight time are, the better. however, the weight of fare and flight time is difficult to balance. For instance, trade-off between a flight with lower price but longer flight time and a flight with higher price but shorter flight time can never be achieved. Therefore, users' data are required to determine the weight of two attributes.

Undeniably, for the testing data to be more meaningful, all real flight information gains from a real Chinese travel website-Ctrip.com. because flight information is public data, it can be utilized. However, there is no source of user’s data, all user’s data is simulated.

Furthermore, in addition to two quantifiable attributes: fare and flight time, some text information, such as the airlines, direct flight or not and class, that may affect user’s behavior is included. These are hard to represent in numbers, so Collaborative Filtering(CF) recommendation algorithm is needed.

As described in the previous design stage, this project applied the item-based CF recommendation system. however, this algorithm relies on the data of logged-in users, and the CF algorithm will not work with visitors or new users who do not have any booking records. In response to this problem, a new algorithm called “initialisation recommendation algorithm ” is designed for this project, the implementation steps of this algorithm are listed below (some code implemented is shown in Figure 5.13).

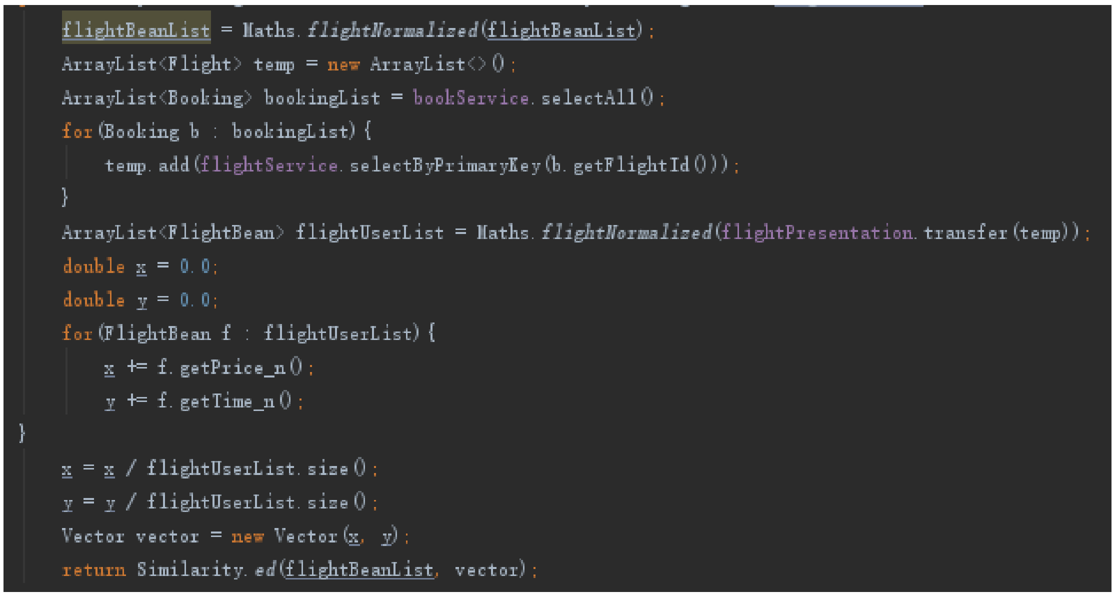
First of all, fares and flight time of all searched flights (i.e. flights that need to be sorted) are normalized

The fare and flight time of purchased air tickets are normalized and averaged. This is the idea of a popularity-based recommendation algorithm, which recommends for uncertain user groups in the direction of common interest.

Compare the gained two averages to get weights.

Calculate Euclidean Distance from origin to normalized fare and flight time of flights that need to be sorted on the axis, during which the fare and flight time are weighted respectively.

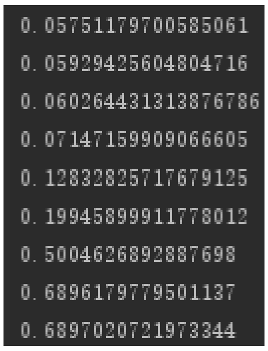
Finally, the flight is sorted in ascending order by the calculated results.



```
    flightBeanList = Maths.flightNormalized(flightBeanList);
    ArrayList<Flight> temp = new ArrayList<>();
    ArrayList<Booking> bookingList = bookService.selectAll();
    for(Booking b : bookingList){
        temp.add(flightService.selectByPrimaryKey(b.getFlightId()));
    }
    ArrayList<FlightBean> flightUserList = Maths.flightNormalized(flightPresentation.transfer(temp));
    double x = 0.0;
    double y = 0.0;
    for(FlightBean f : flightUserList){
        x += f.getPrice_n();
        y += f.getTime_n();
}
    x = x / flightUserList.size();
    y = y / flightUserList.size();
    Vector vector = new Vector(x, y);
    return Similarity.ed(flightBeanList, vector);
```

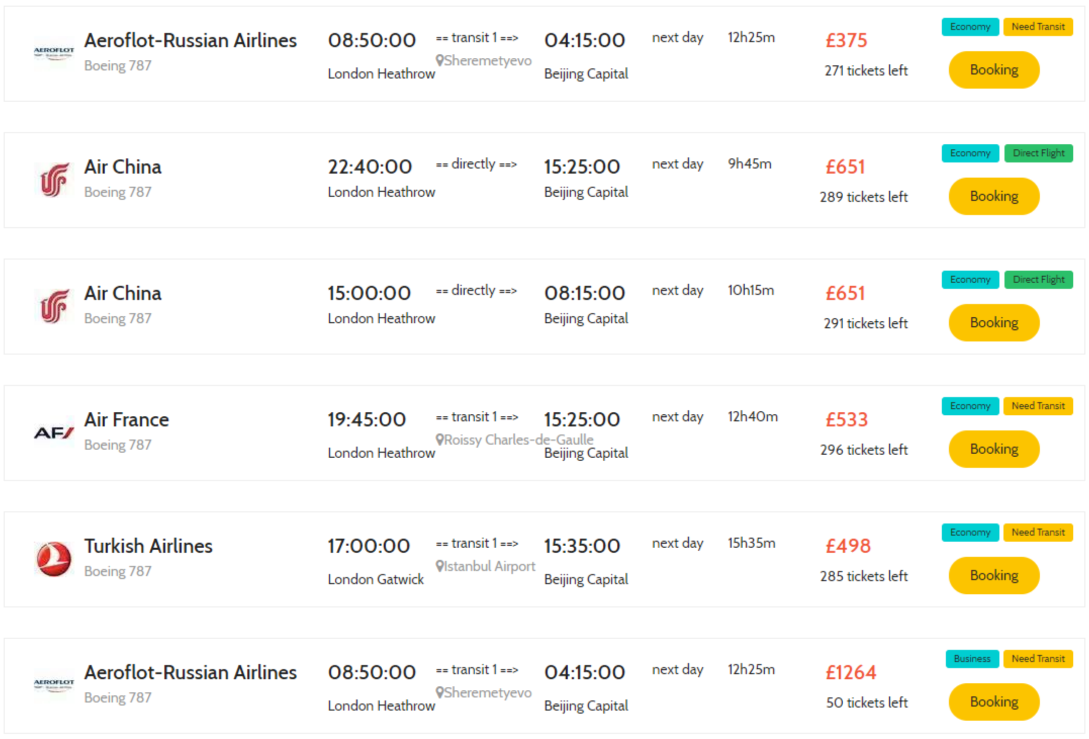
Figure 5.13

Through testing, the results of the calculation are shown in Figure 5.14, and the list of recommended flights is shown in Figure 5.15.



```
0.05751179700585061
0.05929425604804716
0.060264431313876786
0.07147159909066605
0.12832825717679125
0.19945899911778012
0.5004626892887698
0.6896179779501137
0.6897020721973344
```

Figure 5.14



Aeroflot-Russian Airlines Boeing 787	08:50:00 London Heathrow	== transit 1 ==> Sheremetyevo	04:15:00 Beijing Capital	next day	12h25m	£375 271 tickets left	Economy · Need Transit	Booking
Air China Boeing 787	22:40:00 London Heathrow	== directly ==>	15:25:00 Beijing Capital	next day	9h45m	£651 289 tickets left	Economy · Direct Flight	Booking
Air China Boeing 787	15:00:00 London Heathrow	== directly ==>	08:15:00 Beijing Capital	next day	10h15m	£651 291 tickets left	Economy · Direct Flight	Booking
Air France Boeing 787	19:45:00 London Heathrow	== transit 1 ==> Roissy Charles-de-Gaulle	15:25:00 Beijing Capital	next day	12h40m	£533 296 tickets left	Economy · Need Transit	Booking
Turkish Airlines Boeing 787	17:00:00 London Gatwick	== transit 1 ==> Istanbul Airport	15:35:00 Beijing Capital	next day	15h35m	£498 285 tickets left	Economy · Need Transit	Booking
Aeroflot-Russian Airlines Boeing 787	08:50:00 London Heathrow	== transit 1 ==> Sheremetyevo	04:15:00 Beijing Capital	next day	12h25m	£1264 50 tickets left	Business · Need Transit	Booking

Figure 5.15

According to results calculated by the "initialisation recommendation algorithm", flights are not just sorted by fare or flight time. The system will trade off fare and flight time based on weights determined by user data. The testing results indicate that the algorithm is successful.

For users who have been logged in, the system adds the results of the item-based CF recommendation algorithm on top of the above algorithm. The purpose of item-based CF recommendation algorithm is to predict the user's rating of the ticket by matching the similarity between flights. Considering that the system has not yet developed a scoring system, a simulated score for testing is given—Rating of no purchase is 0, rating of one purchase is 1, rating of more than 2 purchases is 2, and by parity of reasoning, the rating is capped with 5.

The ratings of flights that need to be sorted can form a matrix, such as

User

Flight

	1	2	3	4	5	6	7	8	9	10	11
1	3		5		5		3		3		3
2		3		3		3					
3	3		3		3	3			3	3	3
4	3		3		5	5	5	3	3	5	5
5	3		3		3		5		3		
6		3		5		3	5		5	5	5

Table 5.16

$$\mathrm{sim}(\mathrm{f}_3, \mathrm{f}_1) = 0.48$$

$$\mathrm{sim}(\mathrm{f}_3, \mathrm{f}_2) = -0.56$$

$$\mathrm{sim}(\mathrm{f}_3, \mathrm{f}_3) = 1$$

$$\mathrm{sim}(\mathrm{f}_3, \mathrm{f}_4) = 0.56$$

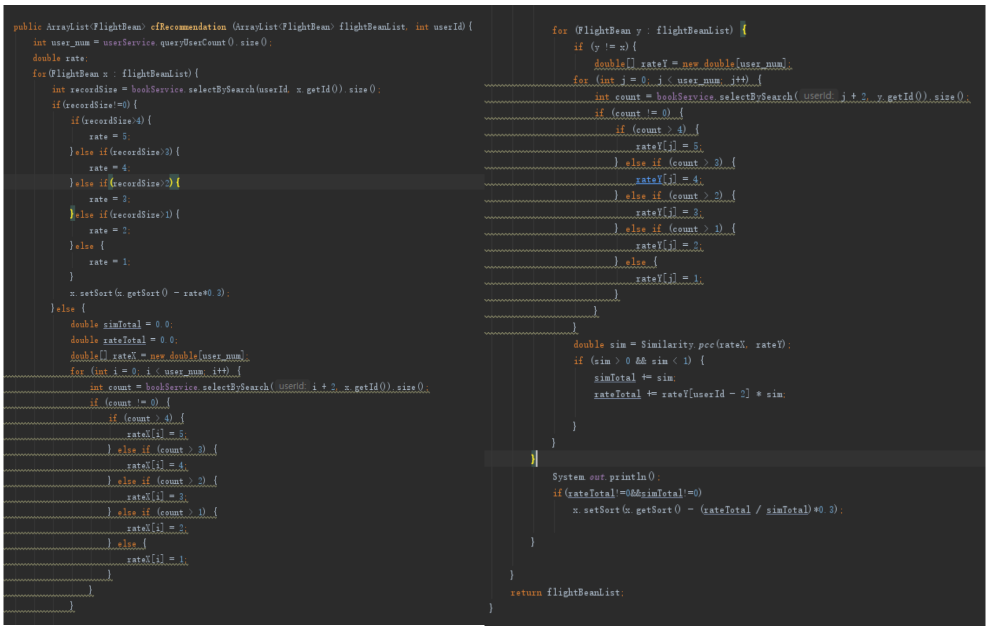
$$\mathrm{sim}(\mathrm{f}_3, \mathrm{f}_5) = 0.13$$

$$\mathrm{sim}(\mathrm{f}_3, \mathrm{f}_6) = -0.14$$

As shown in Table 5.16, if user7 logs in the system, user's rating should be added to the results of the previous algorithm with a certain proportion. However, user7 has never purchased tickets for flight 2 and flight 3, so their value is empty, in which CF recommendation algorithm is needed to complete the rating. If user7's rating about flight 3 is needed, the similarity of each flight to flight 3 has to be calculated first. A threshold then is set, for instance, if the similarity is greater than 0.3 (except1, 1 represents the flight itself), it is relatively similar. The calculated item thus can be used. In this case, only flight 1 and flight 4 are similar. The forecast rating = Sigma(sim*yn)/ Sigma(sim), i.e. the predicted rating of user7 for ticket 3 is (0.56*5+0.48*3)/(0.56+0.48)=4.08.

Therefore, after filling the rating with this method, which is added into the results of the previous algorithm with a certain percentage, Hybrid Recommendation can integrate advantages of both algorithms. The "initialisation recommendation algorithm" makes up for the defect that the CF recommendation algorithm cannot recommend to visitors and new users. Meanwhile, CF recommendation algorithm compensates for the monotony of using a mere of two attributes in the "initialisation recommendation algorithm".

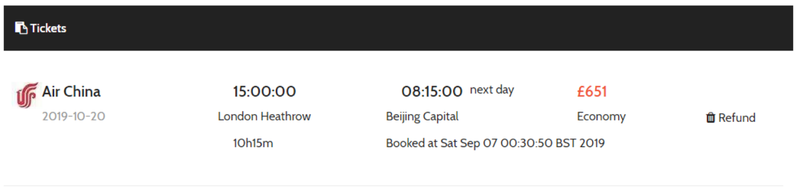
In the project, part of the code implementation of item-based CF recommendation algorithm is shown in Figure 5.17.



```java
public ArrayList<FlightBean> cfRecommendation (ArrayList<FlightBean> flightBeanList, int userId) {
    int user_num = userService.queryUserCount().size();
    double rate;
    for(FlightBean x : flightBeanList) {
        int recordSize = bookService.selectBySearch(userId, x.getId()).size();
        if(recordSize!=0) {
            if(recordSize>4) {
                rate = 5;
            }else if(recordSize>3) {
                rate = 4;
            }else if(recordSize>2) {
                rate = 3;
            }else if(recordSize>1) {
                rate = 2;
            }else {
                rate = 1;
            }
            x.setSort(x.getSort() - rate*0.3);
        }else {
            double simTotal = 0.0;
            double rateTotal = 0.0;
            double[] rateX = new double[user_num];
            for (int i = 0; i < user_num; i++) {
                int count = bookService.selectBySearch( userId: i + 2, x.getId()).size();
                if (count != 0) {
                    if (count > 4) {
                        rateX[i] = 5;
                    } else if (count > 3) {
                        rateX[i] = 4;
                    } else if (count > 2) {
                        rateX[i] = 3;
                    } else if (count > 1) {
                        rateX[i] = 2;
                    } else {
                        rateX[i] = 1;
                    }
                }
            }
            for (FlightBean y : flightBeanList) {
                if (y != x) {
                    double[] rateY = new double[user_num];
                for (int j = 0; j < user_num; j++) {
                    int count = bookService.selectBySearch( userId: j + 2, y.getId()).size();
                    if (count != 0) {
                        if (count > 4) {
                            rateY[j] = 5;
                        } else if (count > 3) {
                            rateY[j] = 4;
                        } else if (count > 2) {
                            rateY[j] = 3;
                        } else if (count > 1) {
                            rateY[j] = 2;
                        } else {
                            rateY[j] = 1;
                        }
                    }
                }
                double sim = Similarity.pcc(rateX, rateY);
                if (sim > 0 && sim < 1) {
                    simTotal += sim;
                    rateTotal += rateY[userId - 2] * sim;

                }
            }
        }
            System.out.println();
            if(rateTotal!=0&&simTotal!=0)
                x.setSort(x.getSort() - (rateTotal / simTotal)*0.3);

        }

    }
    return flightBeanList;
}
```

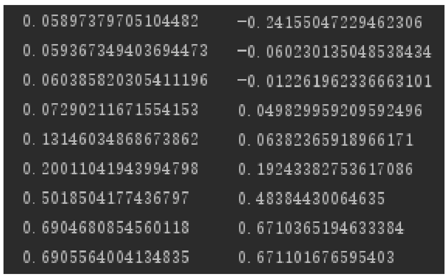
Figure 5.17

Finally, comparing the difference in results after users log in, to test the effect of CF recommendation algorithm. The logged-in user only purchased an Air China's 3 o'clock pm ticket, as shown in Figure 5.18. Testing results are shown in Figure 5.19 and Figure 5.20.



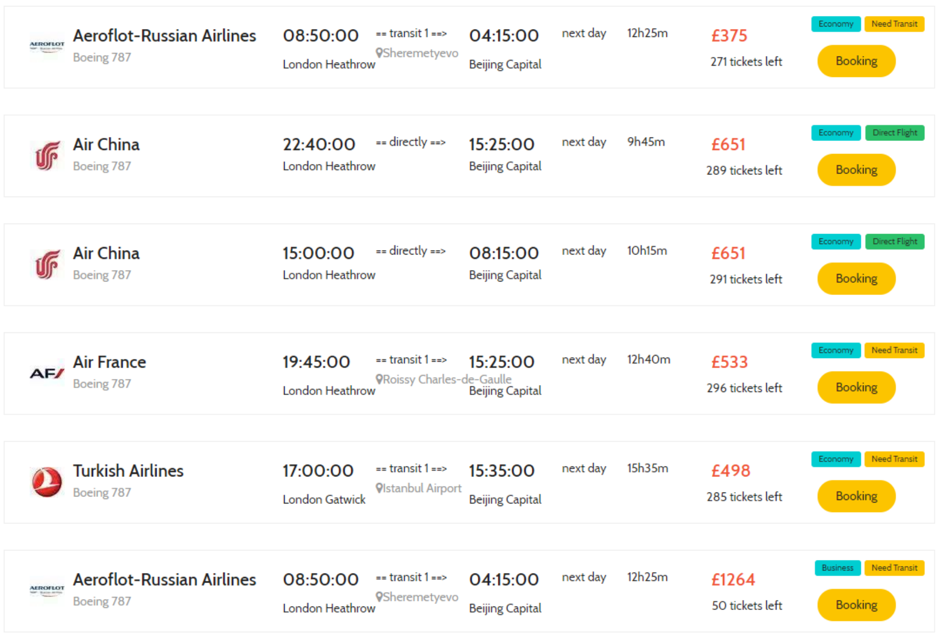
Air China	15:00:00	08:15:00 next day	£651	
2019-10-20	London Heathrow	Beijing Capital	Economy	Refund
	10h15m	Booked at Sat Sep 07 00:30:50 BST 2019		

Figure 5.18

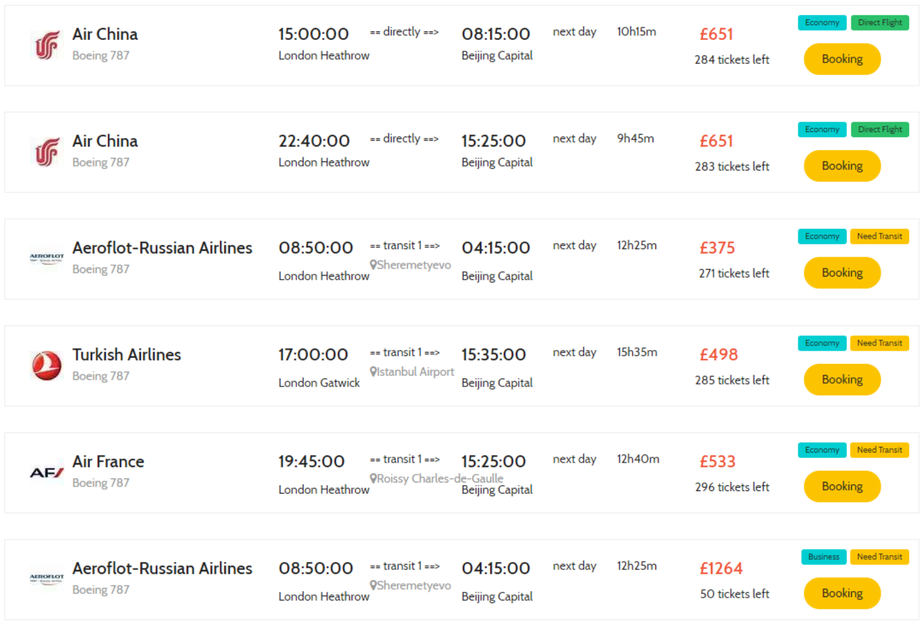


```
0.05897379705104482    -0.24155047229462306
0.059367349403694473   -0.060230135048538434
0.060385820305411196   -0.012261962336663101
0.07290211671554153    0.04982995920959249 6
0.13146034868673862    0.06382365918966171
0.20011041943994798    0.19243382753617086
0.5018504177436797     0.48384430064635
0.6904680854560118     0.671036519463338 4
0.6905564004134835     0.671101676595403
```

Figure 5.19



Aeroflot-Russian Airlines Boeing 787	08:50:00 London Heathrow	== transit 1 ==> Sheremetyevo	04:15:00 Beijing Capital	next day	12h25m	£375 271 tickets left	Economy · Need Transit	Booking
Air China Boeing 787	22:40:00 London Heathrow	== directly ==>	15:25:00 Beijing Capital	next day	9h45m	£651 289 tickets left	Economy · Direct Flight	Booking
Air China Boeing 787	15:00:00 London Heathrow	== directly ==>	08:15:00 Beijing Capital	next day	10h15m	£651 291 tickets left	Economy · Direct Flight	Booking
Air France Boeing 787	19:45:00 London Heathrow	== transit 1 ==> Roissy Charles-de-Gaulle	15:25:00 Beijing Capital	next day	12h40m	£533 296 tickets left	Economy · Need Transit	Booking
Turkish Airlines Boeing 787	17:00:00 London Gatwick	== transit 1 ==> Istanbul Airport	15:35:00 Beijing Capital	next day	15h35m	£498 285 tickets left	Economy · Need Transit	Booking
Aeroflot-Russian Airlines Boeing 787	08:50:00 London Heathrow	== transit 1 ==> Sheremetyevo	04:15:00 Beijing Capital	next day	12h25m	£1264 50 tickets left	Business · Need Transit	Booking



Air China Boeing 787	15:00:00 London Heathrow	== directly ==>	08:15:00 Beijing Capital	next day	10h15m	£651 284 tickets left	Economy · Direct Flight	Booking
Air China Boeing 787	22:40:00 London Heathrow	== directly ==>	15:25:00 Beijing Capital	next day	9h45m	£651 283 tickets left	Economy · Direct Flight	Booking
Aeroflot-Russian Airlines Boeing 787	08:50:00 London Heathrow	== transit 1 ==> Sheremetyevo	04:15:00 Beijing Capital	next day	12h25m	£375 271 tickets left	Economy · Need Transit	Booking
Turkish Airlines Boeing 787	17:00:00 London Gatwick	== transit 1 ==> Istanbul Airport	15:35:00 Beijing Capital	next day	15h35m	£498 285 tickets left	Economy · Need Transit	Booking
Air France Boeing 787	19:45:00 London Heathrow	== transit 1 ==> Roissy Charles-de-Gaulle	15:25:00 Beijing Capital	next day	12h40m	£533 296 tickets left	Economy · Need Transit	Booking
Aeroflot-Russian Airlines Boeing 787	08:50:00 London Heathrow	== transit 1 ==> Sheremetyevo	04:15:00 Beijing Capital	next day	12h25m	£1264 50 tickets left	Business · Need Transit	Booking

Figure 5.20

The comparison of two groups of figures reveals that both sort value and ranking have changed. Due to the fact that the higher the predicted rating is, the lower the value of the sort becomes, the final sort value can be gained by using sort value minus the weighted rating. The testing weigh is set to 0.3, the benchmark will be tested and analyzed to conclude the optimal setting.

It is obvious that ranking of three tickets have increased, especially another Air China ticket, although only one ticket is purchased. This is because a ticket similar to the purchased one is found through item-based CF recommendation algorithm. Depending on their similarity, the predicted ranking of each ticket varies. Ticket with higher similarity will receive more effects.

From the testing results, it is consistent with the theoretical expectations, proving the successful work of the item-based recommendation algorithm.

5.4.5 Flex Combo

It is universally acknowledged that airlines will connect two or more routes to form a new one, thus airlines will offer flights that require transfer. Generally, routes combined by airlines often belong to one airline or the same airline alliance. If a flight ticket needed by a user is sold out, the user might have to find two routes (from different airlines) on their own. Hence, the system should provide the function to connect flexibly with various airlines.

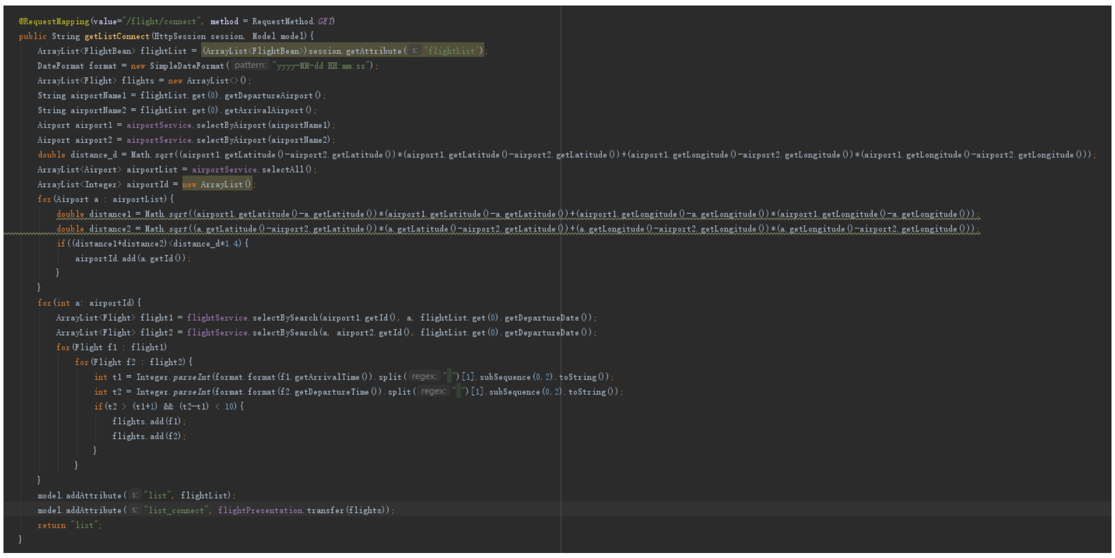
The following three conditions must be satisfied for a flight stitched with this feature:

The arrival airport on the previous flight must be the departure airport of the last flight.

The departure time of the latter flight must be the arrival time of the previous flight plus one hour at least. The transfer time should top 10 hours.

The total flight distance of two routes should not exceed 1.4 times of the straight-line distance from the starting point to the end point.

Specific ode implementations are shown in Figure 5.21.

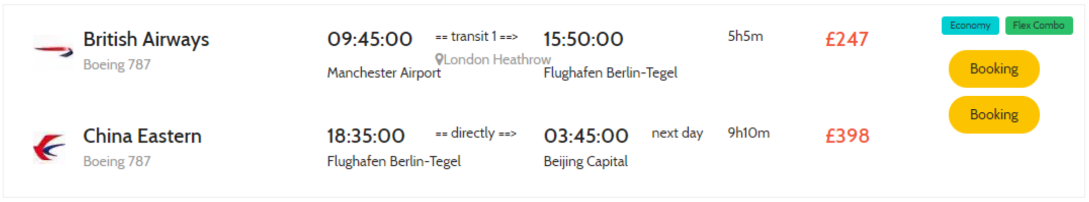


```java
@RequestMapping(value="/flight/connect", method = RequestMethod.GET)
public String getListConnect(HttpSession session, Model model){
    ArrayList<FlightBean> flightList = (ArrayList<FlightBean>)session.getAttribute( s: "flightList");
    DateFormat format = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");
    ArrayList<Flight> flights = new ArrayList<>();
    String airportName1 = flightList.get(0).getDepartureAirport();
    String airportName2 = flightList.get(0).getArrivalAirport();
    Airport airport1 = airportService.selectByAirport(airportName1);
    Airport airport2 = airportService.selectByAirport(airportName2);
    double distance_d = Math.sqrt((airport1.getLatitude()-airport2.getLatitude())*(airport1.getLatitude()-airport2.getLatitude())+(airport1.getLongitude()-airport2.getLongitude())*(airport1.getLongitude()-airport2.getLongitude()));
    ArrayList<Airport> airportList = airportService.selectAll();
    ArrayList<Integer> airportId = new ArrayList();
    for(Airport a : airportList){
        double distance1 = Math.sqrt((airport1.getLatitude()-a.getLatitude())*(airport1.getLatitude()-a.getLatitude())+(airport1.getLongitude()-a.getLongitude())*(airport1.getLongitude()-a.getLongitude()));
        double distance2 = Math.sqrt((a.getLatitude()-airport2.getLatitude())*(a.getLatitude()-airport2.getLatitude())+(a.getLongitude()-airport2.getLongitude())*(a.getLongitude()-airport2.getLongitude()));
        if((distance1+distance2)<distance_d*1.4){
            airportId.add(a.getId());
        }
    }
    for(int a: airportId){
        ArrayList<Flight> flight1 = flightService.selectBySearch(airport1.getId(), a, flightList.get(0).getDepartureDate());
        ArrayList<Flight> flight2 = flightService.selectBySearch(a, airport2.getId(), flightList.get(0).getDepartureDate());
        for(Flight f1 : flight1)
            for(Flight f2 : flight2){
                int t1 = Integer.parseInt(format.format(f1.getArrivalTime()).split( regex: " ")[1].subSequence(0, 2).toString());
                int t2 = Integer.parseInt(format.format(f2.getDepartureTime()).split( regex: " ")[1].subSequence(0, 2).toString());
                if(t2 > (t1+1) && (t2-t1) < 10){
                    flights.add(f1);
                    flights.add(f2);
                }
            }
    }
    model.addAttribute( s: "list", flightList);
    model.addAttribute( s: "list_connect", flightPresentation.transfer(flights));
    return "list";
}
```

Figure 5.21

As shown in the Figure, the flight combination suiting the first condition is found by traversing. Splitting out time by converting datetime type to the String type, and converting them into numbers for minus, which can filter out a flight combination meeting the second condition. Finally, the distance among airports can be calculated by the latitude and longitude of the airport and Euclidean Distance formula (i.e. from

A to B, from B to C, from A to C). flight combinations meeting three conditions will be transmitted and displayed at the front-end. As shown in Figure 5.22, the first flight is from Machester Airport to Flughafen Berlin-Tegel Airport, and the second flight is from Flughafen Berlin-Tegel Airport to Beijing Capital Airport.



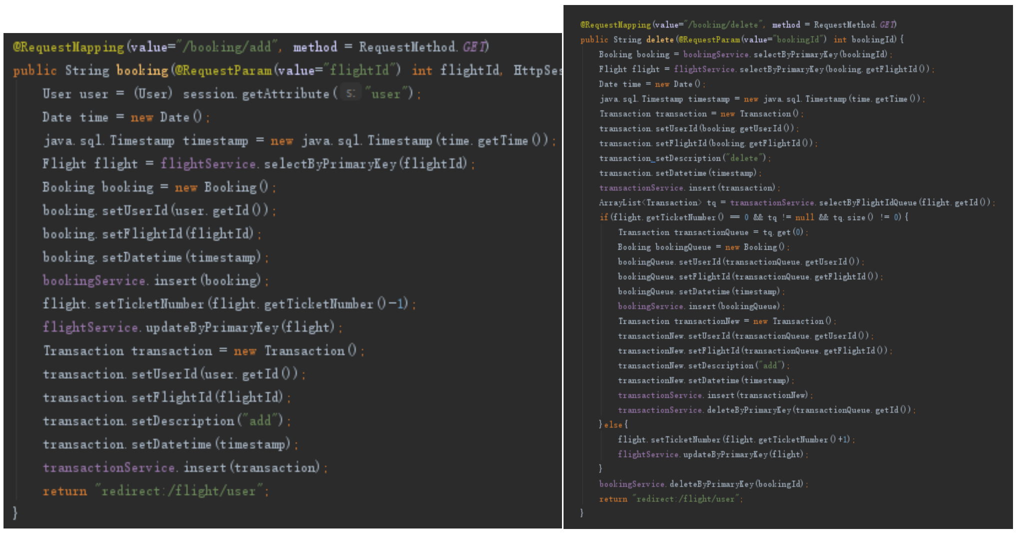
British Airways
Boeing 787
09:45:00 == transit 1 ==> 15:50:00 5h5m £247
London Heathrow
Manchester Airport Flughafen Berlin-Tegel

China Eastern
Boeing 787
18:35:00 == directly ==> 03:45:00 next day 9h10m £398
Flughafen Berlin-Tegel Beijing Capital

Figure 5.22

5.4.6 Queue up

As previously stated, the system requires a queuing function. Entering the queue once tickets are sold out, and buying tickets automatically for the first user in the queue, as soon as the needed ticket is refunded by other users. The code implementation is shown in Figure 5.23, which is integrated with the ticket refund function. The system tracks queue status via transaction table in the database, and the addition of the transaction table will be triggered by each booking, refund, queue and un-queue operation. Also, in the delete method, once a user has triggered this method, the delete method will determine the refunding ticket. If the number of ticket is 0 currently, and transaction table indicates that someone is in a queue, the id of first user in the queue will be searched out. The insert method in BookingService will be called to book tickets for users automatically. At the same time, FlightService will be called to modify the number of ticket if the number has not reached 0 or no one is in the queue, during the ticket booking or refund operation.



```java
@RequestMapping(value="/booking/add", method = RequestMethod.GET)
public String booking(@RequestParam(value="flightId") int flightId, HttpSes
    User user = (User) session.getAttribute( s: "user");
    Date time = new Date();
    java.sql.Timestamp timestamp = new java.sql.Timestamp(time.getTime());
    Flight flight = flightService.selectByPrimaryKey(flightId);
    Booking booking = new Booking();
    booking.setUserId(user.getId());
    booking.setFlightId(flightId);
    booking.setDatetime(timestamp);
    bookingService.insert(booking);
    flight.setTicketNumber(flight.getTicketNumber()-1);
    flightService.updateByPrimaryKey(flight);
    Transaction transaction = new Transaction();
    transaction.setUserId(user.getId());
    transaction.setFlightId(flightId);
    transaction.setDescription("add");
    transaction.setDatetime(timestamp);
    transactionService.insert(transaction);
    return "redirect:/flight/user";
}
```

```java
@RequestMapping(value="/booking/delete", method = RequestMethod.GET)
public String delete(@RequestParam(value="bookingId") int bookingId) {
    Booking booking = bookingService.selectByPrimaryKey(bookingId);
    Flight flight = flightService.selectByPrimaryKey(booking.getFlightId());
    Date time = new Date();
    java.sql.Timestamp timestamp = new java.sql.Timestamp(time.getTime());
    Transaction transaction = new Transaction();
    transaction.setUserId(booking.getUserId());
    transaction.setFlightId(booking.getFlightId());
    transaction.setDescription("delete");
    transaction.setDatetime(timestamp);
    transactionService.insert(transaction);
    ArrayList<Transaction> tq = transactionService.selectByFlightIdQueue(flight.getId());
    if(flight.getTicketNumber() == 0 && tq != null && tq.size() != 0) {
        Transaction transactionQueue = tq.get(0);
        Booking bookingQueue = new Booking();
        bookingQueue.setUserId(transactionQueue.getUserId());
        bookingQueue.setFlightId(transactionQueue.getFlightId());
        bookingQueue.setDatetime(timestamp);
        bookingService.insert(bookingQueue);
        Transaction transactionNew = new Transaction();
        transactionNew.setUserId(transactionQueue.getUserId());
        transactionNew.setFlightId(transactionQueue.getFlightId());
        transactionNew.setDescription("add");
        transactionNew.setDatetime(timestamp);
        transactionService.insert(transactionNew);
        transactionService.deleteByPrimaryKey(transactionQueue.getId());
    }else{
        flight.setTicketNumber(flight.getTicketNumber()+1);
        flightService.updateByPrimaryKey(flight);
    }
    bookingService.deleteByPrimaryKey(bookingId);
    return "redirect:/flight/user";
}
```

Figure 5.23

5.5 Testing

During the system development, JUnit has been used for unit testing (i.e. white box testing).

However, after the entire system is completed, the overall function still needs testing. Also, a few sets of benchmark testing are required for the weights that needs to be set in Hybrid Recommendation Algorithm.

5.5.1 Black-box testing

The section will illustrate the applied testing strategies and the studying of how the developed application satisfy the fixed demands based on the testing results. Black-box testing is the strategy applied in the testing. Ultimately, testing results will be utilized to verify the accomplished project. With consecutive testing, the fluent operation of development function can be guaranteed, which can avoid bugs in the beginning via testing. Meanwhile, an application without mistakes thus can be ensured. The black-box testing was utilized to test whether the application's verifying function is effective or not, and bug-free function, which is tested to have poor performance in the early testing. Please see detailed testing process in Appendix B

Two bugs and their solutions are presented below.

The first bug is that the flex combo of the ticket shows an exception, and that the second ticket information cannot display properly.

Solutions: firstly, a unit test is used to test the correctness of ticket information returned by the getListConnect method before testing the front-end JSTL statement.

Results: changing the loop.count+1 in the front-end JSTL loop statement to loop.count. The error has been resolved successfully.

Analysis: the cause of the error is the subscript overflow. Since loop.count is counted from 1 instead of 0, the subscript of the second flight ticket does not need to be added by 1.

The second bug is that the co-filtering recommendation algorithm is enabled after the user login, but the sort value and flight ranking keep unchanged.

Solutions: testing unit by unit in the formula's calculation order.

1.Covariance 2. Standard deviation 3. Pearson Correlation Factor

4. Prediction Score 5. sort Value

Results: increasing the purchase history of each user, assuming that several users have the same tendency for choosing and simulating their purchase data. Sort value and flight ranking begin changing with the user's behaviour.

Analysis: due to the fact that users' purchase data is generated randomly without planning, and that the number of data is limited, the algorithm cannot find relevant items. Since the value of negative similarity is filtered out during the implementation of the algorithm, all items will be excluded if previous data is used for testing. After adding purchasing data of users designedly, the similarity among items becomes a positive number. Item-based CF recommendation algorithm thus can work out normally. It is noteworthy that planned addition of user data is in accordance with the data simulation rule, as in real life, there should be some people who prefer cheap tickets while others turn to certain airline companies. such simulation data is closer to the real data.

5.5.2 Benchmark testing

This section will benchmark the weights between the two recommended algorithms. The user account being tested only has a small amount of data, meaning that the sort value should be increased after the user logs in to affect the ranking but not too much, which is optimal. The domain of definition for the result of the "initialisation recommendation algorithm" is [0, $\sqrt{2}$], and the Pearson coefficient defines the domain as (0,1]) (since sim≤0 is filtered out, the actual defined domain is (0,1)). Moreover, most people buy only one ticket (i.e. rate =1), so the sim rate is similar to the defined domain of the Initialisation Recommendation Algorithm. Thus, the plan is to test the value of weight from 0.1 to 1. Here are the tests for different weight groups.

Weight = 0.1 CF recommendation algorithm affects ranking

Results:

```
-0.04155047229462307
0.01838768694059755
0.02920389139304567
0.06382365918966171
0.09129581293930127
0.19243382753617086
0.48384430064635
0.6710365194633384
0.671101676595403
```

Since the CF still works when the weight value is set to a minimum of 0.1, so the next step is to test a value smaller than 0.1.

Weight = 0.05 CF recommendation algorithm does not affect ranking.

Results:

```
0.04993681825790006
0.05769659793516555
```

```
0.05844952770537694
0.06382365918966171
0.11202873980415566
0.19243382753617086
0.48384430064635
0.6710365194633384
0.671101676595403
```

Weight = 0.07 CF recommendation algorithm affects ranking.

Results:

```
-0.011550472294623068
0.03018036023896795
0.03542376945250199
0.06382365918966171
0.09751569099875759
0.19243382753617086
0.48384430064635
0.6710365194633384

0.671101676595403
```

Weight = 0.06 CF recommendation algorithm affects ranking.

Results:

```
-0.0015504722946230592
0.034111251338424756
0.03749706213898743
0.06382365918966171
0.09958898368524302
0.19243382753617086
0.48384430064635
0.6710365194633384

0.671101676595403
```

Currently, weight=0.06 is the optimal solution, and then test different users and flight list.

Flight list is changed from London-Beijing to Manchester-Beijing

```
0.06143675260362754
0.10931280030010453
0.32724250978566743
0.37542761348622167
0.41185333440097416

0.6798377089141071
```

London-Beijing after user changed

```
-0.010063181742099934
-0.0023034020648344464
-0.0015504722946230592
0.00382365918966171
0.05202873980415566
0.19243382753617086
0.48384430064635
0.6710365194633384
```

0.671101676595403

Manchester-Beijing after user changed

```
-0.06
0.0014367526036275433
0.10931280030010453
0.26724250978566744
0.37542761348622167
0.41185333440097416
```

0.6798377089141071

Considering the data, the curve is flattening, and there has been no fitting situation, so the weight=0.06 is the optimal solution.

Analysis

The reason for the initial miscalculation of the weight range is that a similar definition domain is simply used as a basis. The real factor affecting weight is the user data used for testing. In the initialisation algorithm, if the user data favours one of the attributes, such as price, the calculation result will become lower. Besides, if the gap of two attributes between flights in database is not large enough, the difference between results will be tiny.

For the item-based CF recommendation algorithm, the size of results depends on the value of the rate. If there are more flights with high rates, the calculation results will become larger. This can be avoided by dividing the average rate. Hence, in the test of this project, weight should not be a fixed value, but ought to change with user behaviour. However, the main reason for this situation lies in the limited number of samples for user's data. Creating a new user and executing operations might affect weight. if the user's data is huge, and the preference of the public has been stabilized, the weight will approximately be a fixed value.

Hence, the large user's data in actual Airline Reservation System will stabilize the weight. moreover, for the calculation of weight, machine learning can be adopted, which will be represented in future work.

Results and discussion

At this point, an airline reservation system's scheduling subsystem has been completed, which includes login/register, search/fuzzy search, filter/sort, book/refund, check tickets, map visualization, flex combo, queue up and other functions. In addition, the collaborative filtering recommendation algorithm was used in the flight system, and new algorithms for visitors and new users have been designed. The recommendation system is a challenging part of the system, as solving many mathematic problems in the project is difficult.

Reviewing the whole project process, the expected goals have been completed perfectly and even exceeded. The challenge of the project lies in the adopting of unknown technologies. To begin with, the project kept modifying and reconstructing in the early stage due to the changing demand. This is because the developer lacks understanding about requirements and starts developing blindly without analyzing demands and planning tasks at each stage. The consequence is that code must be constantly modified, reducing development efficiency, which might cause that projects results can hardly meet the clients' requirements.

After the project technology is determined, short-term requirement analysis is finished immediately once the framework is set up. The long-term project object is divided into small tasks before development testing. Analysis of requirements is completed afterwards. The cycle begins again. Since requirements vary according to actual development, requirements analysis is an extremely important stage. it ensures the smooth development process and avoids the error of ignoring requirements analysis. The short-term iterative approach to develop also increase development efficiency.

Choosing agile development method means that no design is fixed, and its merits can be displayed obviously, especially when unknown technologies are used. Unknown technologies represent potential difficulties and challenges, and long-term requirements are vague. Thus, each function module is preferably independent of each other. The low coupling of the SSM framework ensures that the entire system will not be affected, even if the developer fails to complete the implementation of one function, or need to modify a feature.

Since the developer, as a beginner for SSM framework, never utilize SSM (Spring + SpringMVC + MyBatis) framework before, there are many mistakes with project construction, configuration and debug. For instance, the understanding of Inversion of Control in Spring was not very profound at the beginning of the project development, errors of allocating memory to the object manually, often occur. This leads to strange null-pointer errors.

Although the Spring framework is open-source, it is impossible to read the source code of the framework meticulously and gain insight into the framework due to constraints in time. Due to this, Spring and SpringMVC frameworks are only regarded as a tool to help declare and manage object memory in project development. Thereby, many Spring’s excellent features such as Aspect-Oriented Programming (AOP) are neglected. In the future, @Aspect annotation will be applied to separate codes such as Logging, performance statistics, security controls, transaction processing, exception processing, etc. from the business logic code. Currently, the focus of this project is on business logic.

Additionally, because of the unfamiliarity with the mapping configuration of SpringMVC and web.xml files, loading errors of static resources are found after the UI is consolidated into the project. Even though the problem has been solved, the understanding of servlet and SpringMVC needs to be strengthened. The SSM framework might be a new knowledge for the developer, yet the project is based on Java web development. the developer’s theoretical understanding and proficiency of Java has been improved in the past year’s study for master degree, which has helped the project development immensely. Skilled Java technology ensures that developers are able to focus on logic without worrying about programming language rules and data structures, especially when writing algorithms.

For developers who skill at SQL, the MyBatis framework is easy to learn. MyBatis' logical code is in the XML-style mapping file, sharing identical syntax with SQL. The project has a myriad of CURD code, such as login, registration, display flights, search flights, stitch flights, booking, refund, check orders and queue. All these features are perfectly implemented, and developers minimize the number of accessing the database and accelerate the operating speed of the program. Specifically, the data that users need to query is placed into session or cookie at once, which can be read by the system directly when needed. Furthermore, the developer configured the database connection pool, ensuring that the system processes multiple requests simultaneously without blocking the database.

The entire project has two difficulties: map visualization and recommendation system. the realization of map visualization requires the use of Google Maps API, which urges the developer to learn API’s JavaScript functions. Due to limited time, it is a challenging task to grasp the function and develop with it. The map features that have been developed include airport marking and automatic searching. All airports in the database will be marked on the map, and the lowest fare from the nearest airports to the area will be shown with a simple click. Automatic searching for flights by the system for users can be activated by entering city names and pictures. These developments have been completed successfully without any bugs. The current flight list page and the map on the home page are the same. In the future, the map of the flight list page will be changed to marked airports with routes on the basis of tickets searched by users.

The recommendation system is the most crucial part of the whole project. As developers lack mathematic background, algorithm learning can be a great challenge. In this project, only the collaborative filtering recommendation algorithm is applied after studying several recommendation algorithms, based on the background of this project. However, for visitors and new users, these algorithms are not suitable except for popularity-based recommendation algorithm, because most of them rely on user historical data (so-called “personalised recommendations”).

The popularity-based recommendation algorithm is only based on circulation (sales volume in this case), which is evidently too simple to be suitable for this project. This leads to fewer tickets being sold faster, but the number of tickets is limited, unlike virtual products such as video or music. Under the circumstance that no user historical data are found, recommendation definitely needs flight information. It is the two attributes: the fare and flight time that can affect users’ choice and be quantized. The popularity-based algorithm might not be applicable, yet certain inspirations can be drawn from it. In other words, the interest of the public can be used to predict the preferences of visitors and new users, which has a greater chance of success. A new algorithm thus is designed by the developer to address problem that visitors and new users do not have the user historical data, which has far exceeded the expected goal. Not only can the developer grasp recommendation algorithm, but also the developer is able to use the recommendation algorithm developing system to design a new one.

However, the amount of computation of both algorithms is huge, thus a large number of purchase records, user information and flight information should be traversed. Also, some of the information should be calculated as a factor in the formula. The operating speed is fast due to limited number of testing data. If the system is put into practical use, the speed of the system will be greatly reduced because of a great body of data. The usage experience will be affected as each search is calculated. The solution is to model and develop distributed systems with machine learning methods, which will be reflected in future work.

The algorithm of Flex combo is complicated. In order to ensure the rationality of the stitched route, the distance is calculated by the latitude and longitude of the airport, and to meet the time, the algorithm conducted a lot of string operations. Multiple iterations and calculations result in some time being spent identifying eligible routes in more than 600 testing data. The solution is that the system backstage calculates qualified routes at set intervals, and adds the routes that meet the requirements to a table in the database. When the user queries, the system only needs to run the query method. However, database, method and interface need to be modified in this case. This part has not been realized because of the limited time, which will be implemented in the future.

The most creative and innovative feature in the project should be the queuing system. The completion of this feature, similar to a ticket refund, is basically due to CURD. Because the developer is skilful at logical code, the realization of related features is smooth. Threading issues should be noted. To be specific, one user point queuing button when another user refunds the ticket, which might lead to errors. In this situation, some determinations need to be added to the front-end page or a synchronization lock should be added to the back-end page. However, due to limited time, it is not complemented yet. In the future, related functions will be achieved.

Besides from unit test (white-box testing) and black-box testing, the developer did some benchmarks to test the different performance of the system under the weight of varying recommendation algorithms. During testing, the developer realized that weights are not necessarily fixed values as they were initially affected by testing data, which has led to a new research area – the return of weights in Hybrid Recommendation using machine learning. In the future work, this could serve as a new direction for researching, and machine learning deserves in-depth learning.

Conclusions

In conclusion, the original objectives of the project were successfully accomplished and exceeded expectations.   A reservation system's scheduling subsystem was fully developed. The entire project process went through investigation, requirements analysis, design, testing and reflection. In addition to basic function, other functions include search/fuzzy search, filter/sort, map, visualization, connecting tickets for different flights and automatic queuing to buy tickets have been completed. Moreover, a detailed study of two recommendation algorithms in the flight system was conducted, and their results were verified with benchmark testing. The results of the project are summarized as listed below.

- User login/register
- Book/refund
- User check tickets
- Search/Fuzzy Search
- Filter/Sort
- Recommendation Algorithm
- Map Visualization
- Flex Combo
- Queue up

The experience gained throughout the project process is valuable. Since the developer had no previous experience about SSM framework, recommendation algorithm and Google Maps API, a lot of learning was required before the project begins. From beginners to professionals in using a variety of techniques to develop similar systems, the accumulated experience can be well used in future work. Moreover, there had been some Java-related courses and Java web development experience in the developer's master programs. The knowledge and experience had played a role in the development of the project, which had been consolidated.

I also looked for some relevant works in the course of the project to get some inspiration, such as S.E.Levinson and K.L. Shipley’s A conversational-mode airline information and reservation system using speech input and output (Levinson and Shipley, 1980). This research was dedicated to describing a conversational-mode. The speech-understanding system is a good idea, but not the focus of this project. It could be part of future work. Furthermore, references were also made to some of the related applications, such as Expedia and Trip.com. After thoroughly analyzing their advantages and disadvantages, the feasible idea was referred to.

Due to limited time, some good ideas had not been included in the project objectives. However, the development of the system is promising, and the development of the scheduling system is only the beginning of airline reservation system. Thus, the prototype is expected to be expanded to a varying extent. The planned additional features are as follows:

- Optimise the recommendation algorithm
- Conversational system
- Phonetic system
- Use web service technology to gain data from airlines instead of getting them from a database.
- Develop naïve application in phone’s end, and sync with the server on the web end.
- Improve user information encryption

References

Ahn, H. (2008). A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. Information Sciences, 178(1), pp.37-51.

Albadvi, A. and Shahbazi, M. (2009). A hybrid recommendation technique based on product category attributes. Expert Systems with Applications, 36(9), pp.11480-11488.

Arthur, J. and Azadegan, S. (2005). Spring Framework for Rapid Open Source J2EE Web Application Development: A Case Study. Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05).

Aswani (2014). Exploring spring framework advantages and disadvantages. [online] One Stop Blog. Available at: http://www.aksindiblog.com/spring-framework-advantages-disadvantages.html [Accessed 19 Aug. 2019].

Basics, Q., QA, T., Testing, A., Ghahrai, A., Ghahrai, A., Policy, P. and Use, T. (2019). Incremental Model - Advantages and Disadvantages. [online] Testing Excellence. Available at: https://www.testingexcellence.com/incremental-model/ [Accessed 3 Sep. 2019].

Belsare, R. and Deshmukh, D. (2018). Employment Recommendation System using Matching, Collaborative Filtering and Content Based Recommendation. International Journal of Computer Applications Technology and Research, 7(6), pp.215-220.

Benesty, J., Chen, J., Huang, Y. and Cohen, I. (2009). Pearson Correlation Coefficient. Noise Reduction in Speech Processing, pp.1-4.

Benesty, J., Jingdong Chen and Yiteng Huang (2008). On the Importance of the Pearson Correlation Coefficient in Noise Reduction. IEEE Transactions on Audio, Speech, and Language Processing, 16(4), pp.757-765.

Bergman, N. and Chopra, A. (2019). Introduction to JavaServer Pages. [online] Ibm.com. Available at: https://www.ibm.com/developerworks/java/tutorials/j-introjsp/j-introjsp.html [Accessed 9 Aug. 2019].

Boehm, B. (1988). A spiral model of software development and enhancement. Computer, 21(5), pp.61-72.

Cacheda, F., Carneiro, V., Fernández, D. and Formoso, V. (2011). Comparison of collaborative filtering algorithms. ACM Transactions on the Web, 5(1), pp.1-33.

Cockburn, A. and Highsmith, J. (2001). Agile software development, the people factor. Computer, 34(11), pp.131-133.

Dandan Zhang, Zhiqiang Wei and Yongquan Yang (2013). Research on Lightweight MVC Framework Based on Spring MVC and Mybatis. 2013 Sixth International Symposium on Computational Intelligence and Design.

de Campos, L., Fernández-Luna, J., Huete, J. and Rueda-Morales, M. (2010). Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. International Journal of Approximate Reasoning, 51(7), pp.785-799.

Dingsøyr, T., Rolland, K., Moe, N. and Seim, E. (2017). Coordination in multi-team programmes: An investigation of the group mode in large-scale agile software development. Procedia Computer Science, 121, pp.123-128.

Fisher, M. (2012). Spring Integration in Action. Manning Publications.

Flanagan, D. (2011). JavaScript. Sebastopol: O'Reilly Media, Inc., p.2.

Gammeter, S., Gassmann, A., Bossard, L., Quack, T. and Van Gool, L. (2019). Server-side object recognition and client-side object tracking for mobile augmented reality.

Google (2019). Overview | Maps JavaScript API | Google Developers. [online] Google Developers. Available at:

https://developers.google.com/maps/documentation/javascript/tutorial [Accessed 1 Sep. 2019].

Gupta, P. and Govil, M. (2010). MVC Design Pattern for the multi framework distributed applications using XML, spring and struts framework. [ebook] International Journal on Computer Science and Engineering. Available at: https://www.researchgate.net/profile/Praveen_Gupta25/publication/49619227_MVC_Design_Pattern_for_the_multi_framework_distributed_applications_using_XML_spring_and_struts_framework/links/5672564e08ae54b5e462aac5.pdf [Accessed 22 Aug. 2019].

Herlocker, J., Konstan, J., Terveen, L. and Riedl, J. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems, 22(1), pp.5-53.

Hesterberg, T. (2011). Bootstrap. Wiley Interdisciplinary Reviews: Computational Statistics, 3(6), pp.497-526.

Highsmith, J. (2006). Agile software development ecosystems. Boston: Addison-Wesley.

Ho, C. (2012). Using MyBatis in Spring. Pro Spring 3, pp.397-435.

Hunter, J. and Crawford, W. (2010). Java Servlet Programming. Sebastopol: O'Reilly Media, Inc.

Johnson, R. (2004). Expert One-on-One J2EE Design and Development. Hoboken: John Wiley & Sons.

Koren, Y. (2008). Factorization meets the neighborhood. Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08.

Laddad, R. (2010). AspectJ in action: Enterprise AOP with Spring Applications. Greenwich, Conn.: Manning.

Laplante, P. (2007). What every engineer should know about software engineering. Boca Raton: CRC Press.

Levinson, S. and Shipley, K. (1980). A Conversational-Mode Airline Information and Reservation System Using Speech Input and Output. Bell System Technical Journal, 59(1), pp.119-137.

Li, H., Zhang, S. and Wang, X. (2013). A Personalization Recommendation Algorithm for E-Commerce. Journal of Software, 8(1).

Linden, G., Smith, B. and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing, 7(1), pp.76-80.

Lops, P., de Gemmis, M. and Semeraro, G. (2010). Content-based Recommender Systems: State of the Art and Trends. Recommender Systems Handbook, pp.73-105.

Lucas, J., Luz, N., Moreno, M., Anacleto, R., Almeida Figueiredo, A. and Martins, C. (2013). A hybrid recommendation approach for a tourism system. Expert Systems with

Applications, 40(9), pp.3532-3550.

Mark Otto (2019). Introduction. [online] Getbootstrap.com. Available at: https://getbootstrap.com/docs/4.3/getting-started/introduction/ [Accessed 26 Aug. 2019].

Massol, V. and Husted, T. (2004). JUnit in action. Greenwich, Conn.: Manning.

Meteren, R. and Someren, M. (2000). Using Content-Based Filtering for Recommendation. [ebook] Available at: http://users.ics.forth.gr/~potamias/mlnia/paper_6.pdf [Accessed 6 Sep. 2019].

Mobasher, B., Dai, H., Luo, T. and Nakagawa, M. (2001). Effective personalization based on association rule discovery from web usage data. Proceeding of the third international workshop on Web information and data management - WIDM '01.

Nilashi, M., Ibrahim, O. and Ithnin, N. (2014). Hybrid recommendation approaches for multi-criteria collaborative filtering. Expert Systems with Applications, 41(8), pp.3879-3900.

Patel, N. (2017). What are The Pros & Cons of Foundation and Bootstrap?. [online] The Blog - Webby Monks. Available at: https://webbymonks.com/blog/what-are-the-pros-cons-of-foundation-and-bootstrap/ [Accessed 26 Aug. 2019].

Paul, D. (2018). Advantages and Disadvantages of PHP. [online] Bigcheaphosting.com. Available at: https://bigcheaphosting.com/advantages-and-disadvantages-of-php/ [Accessed 11 Aug. 2019].

Pazzani, M. and Billsus, D. (2007). Content-Based Recommendation Systems. The Adaptive Web, pp.325-341.

Pereira, N. and Varma, S. (2016). Survey on Content Based Recommendation System. [ebook] International Journal of Computer Science and Information Technologies. Available at: https://pdfs.semanticscholar.org/5ed8/f58c3de37fc80016ed12c1be5b0af605f2c3.pdf [Accessed 6 Sep. 2019].

Perone, C. (2013). [online] Blog.christianperone.com. Available at: http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/ [Accessed 5 Sep. 2019].

Petersen, K., Wohlin, C. and Baca, D. (2009). The Waterfall Model in Large-Scale Development. Lecture Notes in Business Information Processing, pp.386-400.

Pinela, C. (2017). Content-Based Recommender Systems. [online] Medium. Available at: https://medium.com/@cfpinela/content-based-recommender-systems-a68c2aee2235 [Accessed 6 Sep. 2019].

Pursnani, V. (2019). An introduction to Java servlet programming.

Reddy, K. (2013). Implementing MyBatis. Birmingham: Packt Publishing.

Redmond, E. (2019). Maven – Maven in 5 Minutes. [online] Maven.apache.org. Available at: https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html [Accessed 29 Aug. 2019].

Robinson, A. (2019). How to Calculate Euclidean Distance. [online] Sciencing. Available at: https://sciencing.com/how-to-calculate-euclidean-distance-12751761.html [Accessed 3 Sep. 2019].

Rovce, W. (1987). Managing the development of large software systems: concepts and techniques. [ebook] Available at: https://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf [Accessed 3 Sep. 2019].

Sandvig, J., Mobasher, B. and Burke, R. (2007). Robustness of collaborative recommendation based on association rule mining. Proceedings of the 2007 ACM conference on Recommender systems - RecSys '07.

Sato, S., Hasegawa, A., Kumagai, Y. and Kamiyoshi, U. (2017). Content-based Instruction (CBI) for the Social Future: A Recommendation for Critical Content-Based Language Instruction (CCBI). L2 Journal, 9(3).

Singh, A. (2017). A SIMULATION MODEL FOR INCREMENTAL SOFTWARE DEVELOPMENT LIFE CYCLE MODEL.

Su, X. and Khoshgoftaar, T. (2019). A Survey of Collaborative Filtering Techniques.

Tan, H., Guo, J. and Li, Y. (2008). E-learning Recommendation System. 2008 International Conference on Computer Science and Software Engineering.

Technical Whitepaper (2005). Euclidean Distance. [ebook] Available at: https://www.pbarrett.net/techpapers/euclid.pdf [Accessed 3 Sep. 2019].

Tilkov, S. and Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing, 14(6), pp.80-83.

Wikipedia (2019). Bcrypt. [online] En.wikipedia.org. Available at: https://en.wikipedia.org/wiki/Bcrypt [Accessed 1 Sep. 2019].

Yan, L. and Li, C. (2006). Incorporating Pageview Weight into an Association-Rule-Based Web Recommendation System. Lecture Notes in Computer Science, pp.577-586.

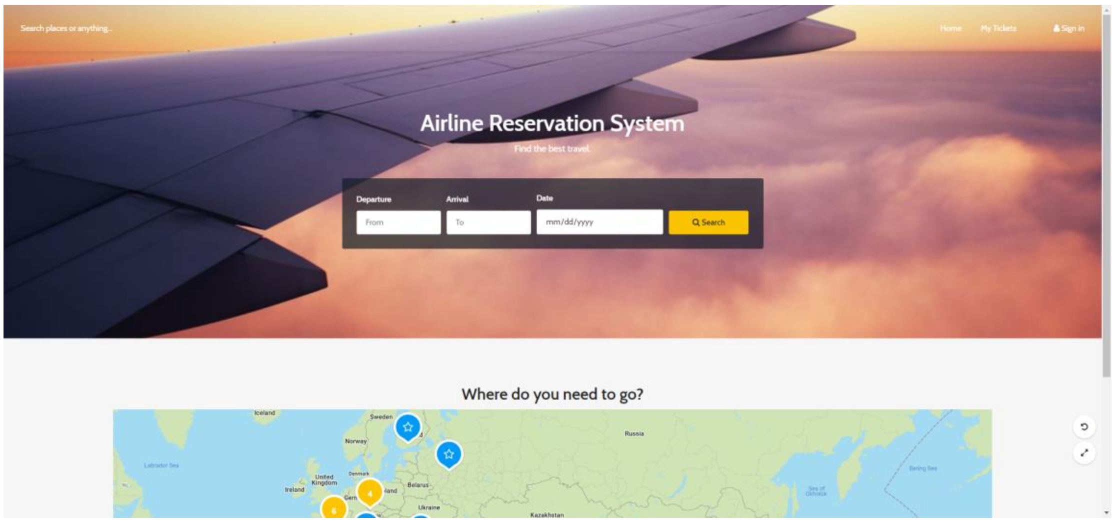
Ye, J. (2011). Cosine similarity measures for intuitionistic fuzzy sets and their applications. Mathematical and Computer Modelling, 53(1-2), pp.91-97.

Zhang, Y. and Jiao, J. (2007). An associative classification-based recommendation system for personalization in B2C e-commerce applications. Expert Systems with Applications, 33(2), pp.357-367.

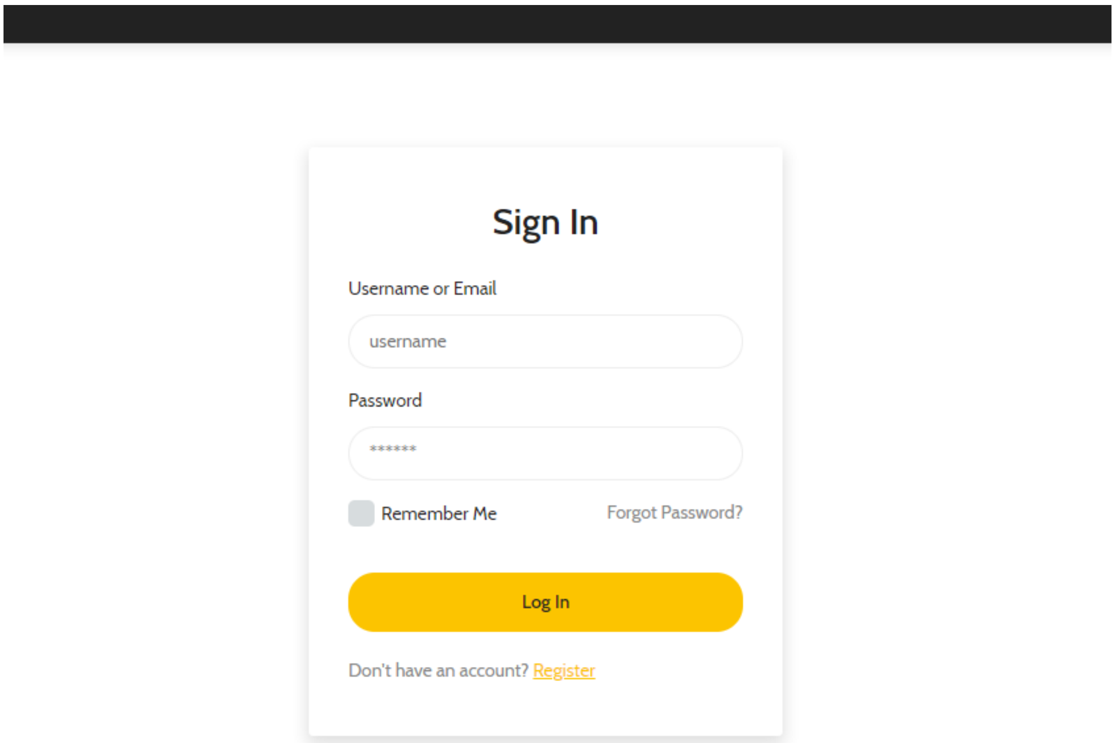
Zhi-Dan Zhao and Ming-Sheng Shang (2010). User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop. 2010 Third International Conference on Knowledge Discovery and Data Mining.

Appendices

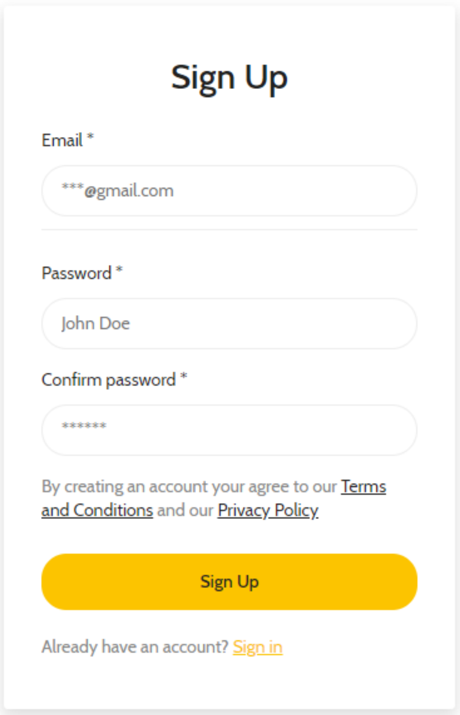
Appendix A



Dashboard



Login



Sign Up

Email *

***@gmail.com

Password *

John Doe

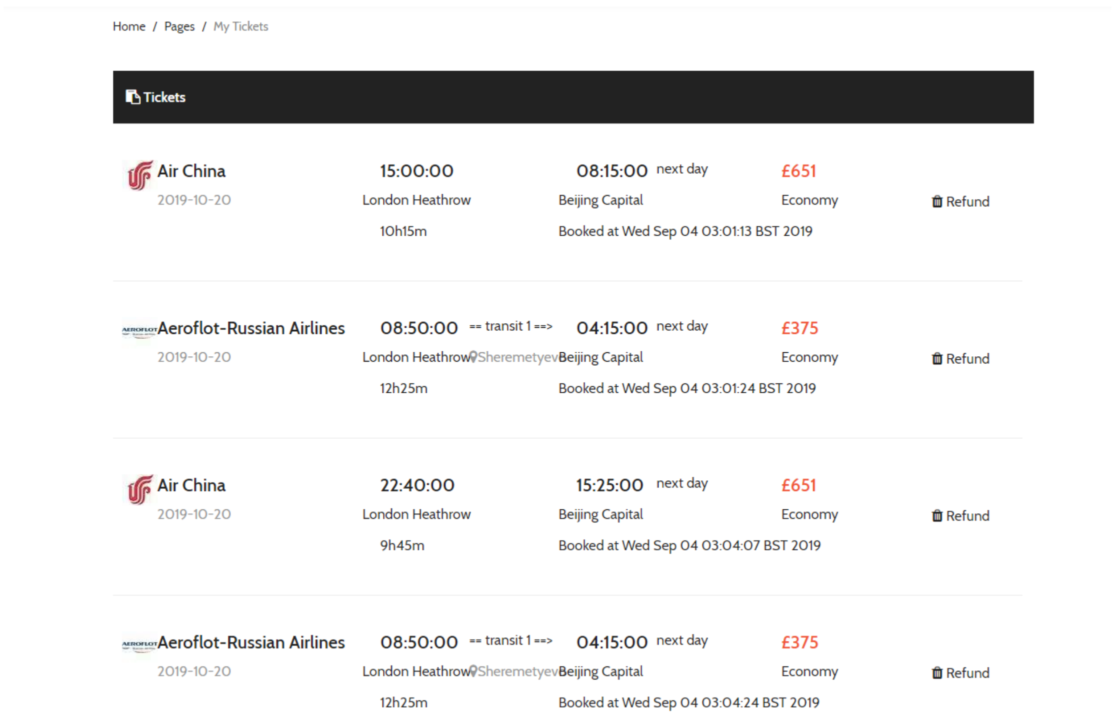
Confirm password *

******

By creating an account your agree to our Terms and Conditions and our Privacy Policy

Already have an account? Sign in

Register

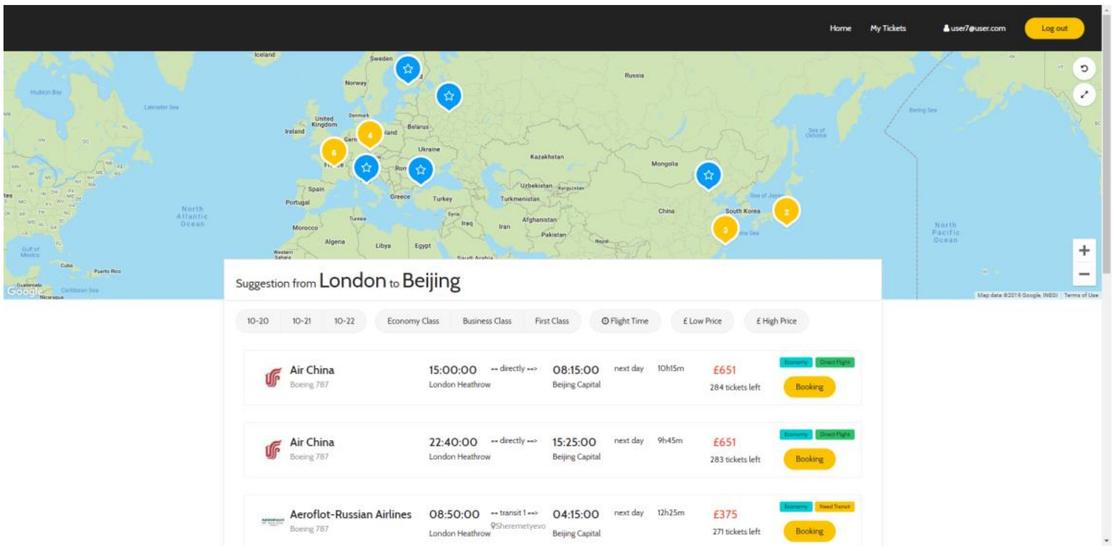


Home / Pages / My Tickets

Tickets

Air China 2019-10-20	15:00:00 London Heathrow 10h15m		08:15:00 next day Beijing Capital Booked at Wed Sep 04 03:01:13 BST 2019	£651 Economy	Refund
Aeroflot-Russian Airlines 2019-10-20	08:50:00 London Heathrow 12h25m	== transit 1 ==> Sheremetyev	04:15:00 next day Beijing Capital Booked at Wed Sep 04 03:01:24 BST 2019	£375 Economy	Refund
Air China 2019-10-20	22:40:00 London Heathrow 9h45m		15:25:00 next day Beijing Capital Booked at Wed Sep 04 03:04:07 BST 2019	£651 Economy	Refund
Aeroflot-Russian Airlines 2019-10-20	08:50:00 London Heathrow 12h25m	== transit 1 ==> Sheremetyev	04:15:00 next day Beijing Capital Booked at Wed Sep 04 03:04:24 BST 2019	£375 Economy	Refund

My Tickets

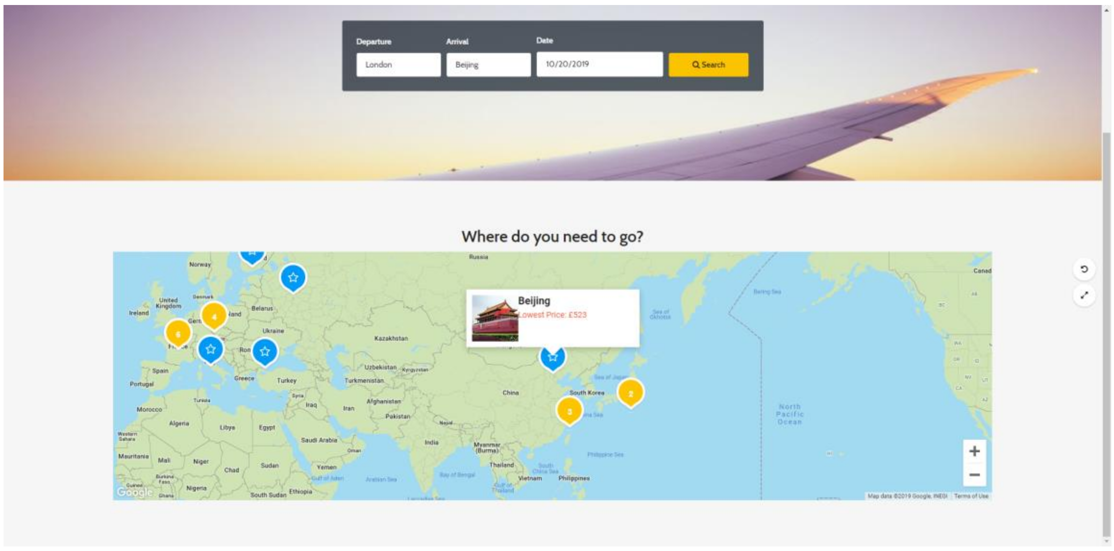


Suggestion from London to Beijing

10-20 | 10-21 | 10-22 | Economy Class | Business Class | First Class | Flight Time | £ Low Price | £ High Price

Air China Boeing 787	15:00:00 London Heathrow	-- directly -->	08:15:00 Beijing Capital	next day	10h15m	£651 284 tickets left	Economy Direct Flight	Booking
Air China Boeing 787	22:40:00 London Heathrow	-- directly -->	15:25:00 Beijing Capital	next day	9h45m	£651 283 tickets left	Economy Direct Flight	Booking
Aeroflot-Russian Airlines Boeing 787	08:50:00 London Heathrow	-- transit 1 --> Sheremetyevo	04:15:00 Beijing Capital	next day	12h25m	£375 271 tickets left	Economy Need Transit	Booking

Flight List

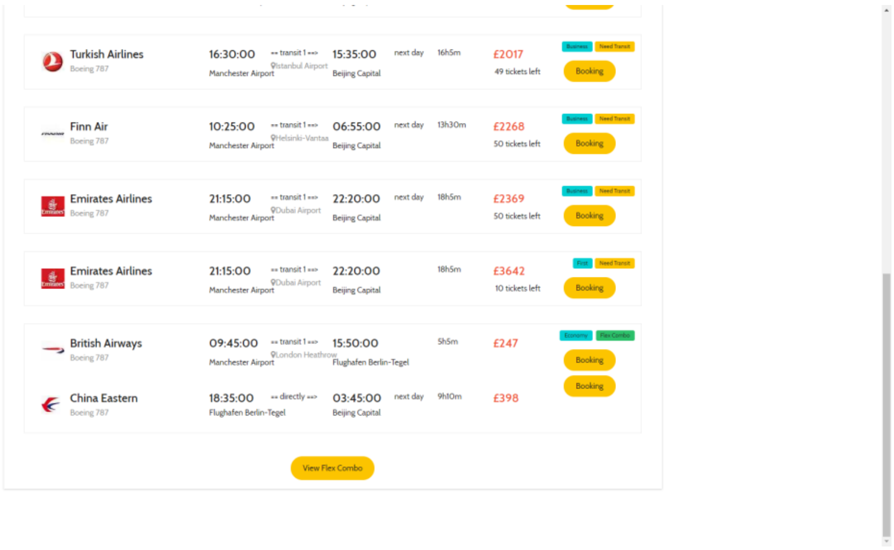


Departure: London
Arrival: Beijing
Date: 10/20/2019
Search

Where do you need to go?

Beijing
Lowest Price: £523

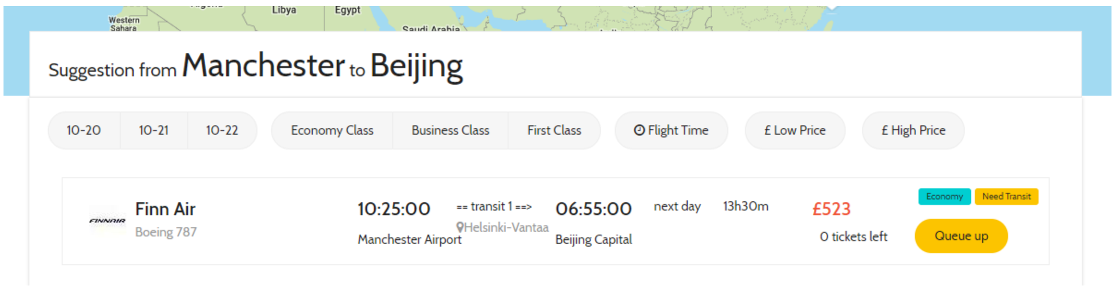
Map Visualization



Turkish Airlines Boeing 787	16:30:00 Manchester Airport	-- transit 1 --> Istanbul Airport	15:35:00 Beijing Capital	next day	16h5m	£2017 49 tickets left	Business Need Transit	Booking
Finn Air Boeing 787	10:25:00 Manchester Airport	-- transit 1 --> Helsinki-Vantaa	06:55:00 Beijing Capital	next day	13h30m	£2268 50 tickets left	Business Need Transit	Booking
Emirates Airlines Boeing 787	21:15:00 Manchester Airport	-- transit 1 --> Dubai Airport	22:20:00 Beijing Capital	next day	18h5m	£2369 50 tickets left	Business Need Transit	Booking
Emirates Airlines Boeing 787	21:15:00 Manchester Airport	-- transit 1 --> Dubai Airport	22:20:00 Beijing Capital		18h5m	£3642 10 tickets left	First Need Transit	Booking
British Airways Boeing 787	09:45:00 Manchester Airport	-- transit 1 --> London Heathrow	15:50:00 Flughafen Berlin-Tegel		5h5m	£247	Economy Flex Combo	Booking
China Eastern Boeing 787	18:35:00 Flughafen Berlin-Tegel	-- directly -->	03:45:00 Beijing Capital	next day	9h10m	£398		Booking

View Flex Combo

Flex Combo

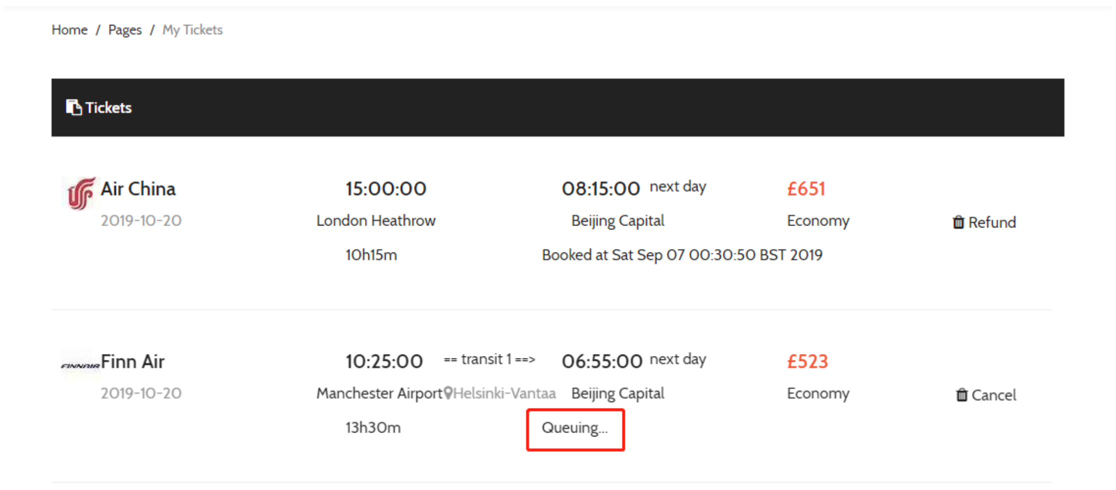


Suggestion from Manchester to Beijing

10-20 10-21 10-22 | Economy Class Business Class First Class | Flight Time | £ Low Price | £ High Price

Finn Air
Boeing 787
10:25:00 == transit 1 ==> 06:55:00 next day 13h30m £523 Economy Need Transit
Manchester Airport Helsinki-Vantaa Beijing Capital 0 tickets left Queue up

Queue up



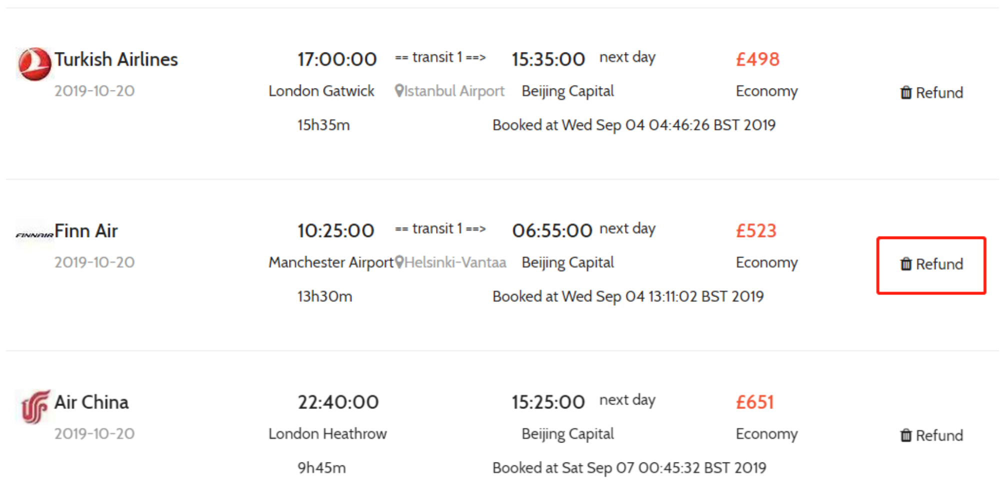
Home / Pages / My Tickets

Tickets

Air China
2019-10-20
15:00:00 08:15:00 next day £651
London Heathrow Beijing Capital Economy Refund
10h15m Booked at Sat Sep 07 00:30:50 BST 2019

Finn Air
2019-10-20
10:25:00 == transit 1 ==> 06:55:00 next day £523
Manchester Airport Helsinki-Vantaa Beijing Capital Economy Cancel
13h30m Queuing...

After Queuing

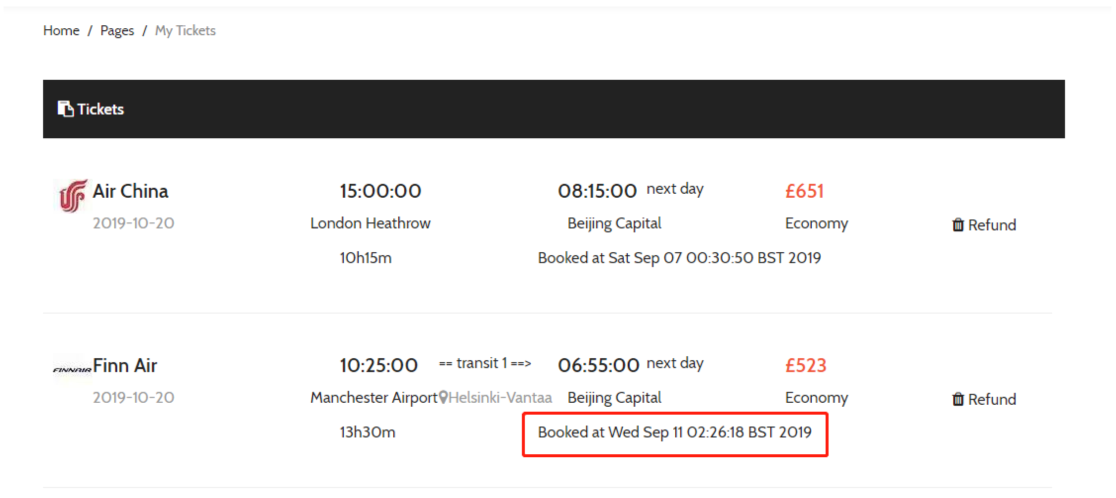


Turkish Airlines
2019-10-20
17:00:00 == transit 1 ==> 15:35:00 next day £498
London Gatwick Istanbul Airport Beijing Capital Economy Refund
15h35m Booked at Wed Sep 04 04:46:26 BST 2019

Finn Air
2019-10-20
10:25:00 == transit 1 ==> 06:55:00 next day £523
Manchester Airport Helsinki-Vantaa Beijing Capital Economy Refund
13h30m Booked at Wed Sep 04 13:11:02 BST 2019

Air China
2019-10-20
22:40:00 15:25:00 next day £651
London Heathrow Beijing Capital Economy Refund
9h45m Booked at Sat Sep 07 00:45:32 BST 2019

Another User Refund


Home / Pages / My Tickets

Tickets

Air China 2019-10-20	15:00:00 London Heathrow 10h15m	08:15:00 next day Beijing Capital	£651 Economy	🗑 Refund
		Booked at Sat Sep 07 00:30:50 BST 2019		
Finn Air 2019-10-20	10:25:00 == transit 1 ==> Manchester Airport Helsinki-Vantaa 13h30m	06:55:00 next day Beijing Capital	£523 Economy	🗑 Refund
		Booked at Wed Sep 11 02:26:18 BST 2019		

Book Automatically

Appendix B

Test Plan 1

Test No.	Description	Expected Result	Result (Pass / Fail)	Comments
1.1	Launch application	Application will launch and the dashboard will be displayed	Pass	
1.2	Login	Navigate to Login Screen, navigate to the dashboard after login	Pass	
1.3	Register	Login automatically	Pass	
1.4	Logout	Logout and navigate to the dashboard	Pass	
2.1	Check Tickets	The current user’s flight tickets display	Pass	
2.2	Book Tickets	Book successfully and navigate to MyTickets	Pass	

		page		
2.3	Refund Tickets	Refund successfully and navigate to MyTickets page	Pass	
3.1	Search	Display the flights recommended according to the conditions	Fail	The list of flights did not change after login. Users with historical data should see their own list of flights recommended.
3.2	Fuzzy Search	Pop up the correct prompt	Pass	
4.1	Filter	Filter out non-conforming flights	Pass	
4.2	Sort	Sort flights by the order	Pass	
4.3	Flex Combo	Connect flights according to the conditions	Fail	The second flight information can not be displayed normally
5.1	Map Visualization	Display the map and markers. Search flights automatically if click the city name or picture on marker window	Pass	
6.1	Queue up	Jion in the line and buy tickets automatically if the flight is refunded	Pass	
6.2	Cancel queue	Leave the queue	Pass	

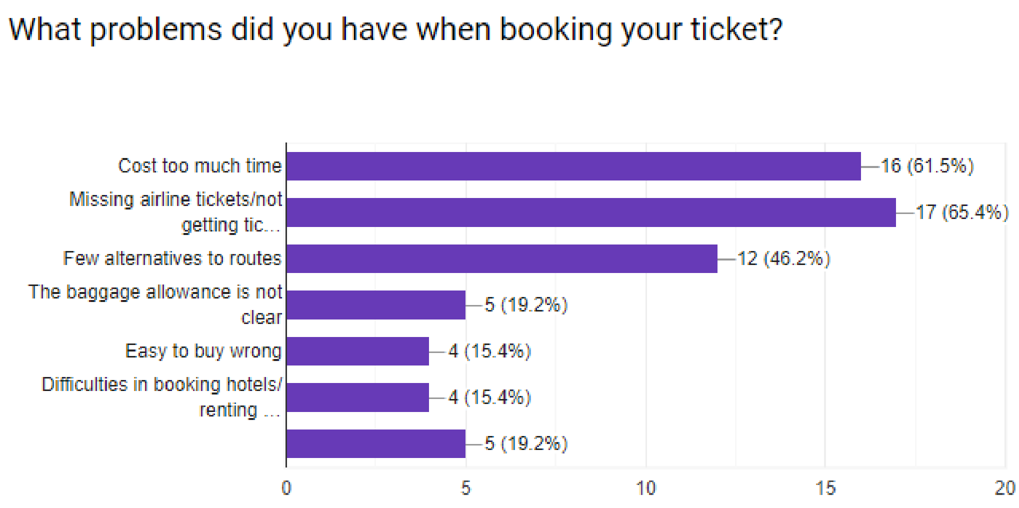
Test Plan 2

Test No.	Description	Expected Result	Result (Pass / Fail)	Comments

1.1	Launch application	Application will launch and the dashboard will be displayed	Pass	
1.2	Login	Navigate to Login Screen, navigate to the dashboard after login	Pass	
1.3	Register	Login automatically	Pass	
1.4	Logout	Logout and navigate to the dashboard	Pass	
2.1	Check Tickets	The current user’s flight tickets display	Pass	
2.2	Book Tickets	Book successfully and navigate to MyTickets page	Pass	
2.3	Refund Tickets	Refund successfully and navigate to MyTickets page	Pass	
3.1	Search	Display the flights recommended according to the conditions	Pass	
3.2	Fuzzy Search	Pop up the correct prompt	Pass	
4.1	Filter	Filter out non-conforming flights	Pass	
4.2	Sort	Sort flights by the order	Pass	
4.3	Flex Combo	Connect flights according to the conditions	Pass	

5.1	Map Visualization	Display the map and markers. Search flights automatically if click the city name or picture on marker window	Pass	
6.1	Queue up	Jion in the line and buy tickets automatically if the flight is refunded	Pass	
6.2	Cancel queue	Leave the queue	Pass	

Appendix C


What problems did you have when booking your ticket?
Cost too much time
Missing airline tickets/not getting tic...
Few alternatives to routes
The baggage allowance is not clear
Easy to buy wrong
Difficulties in booking hotels/ renting ...
16 (61.5%)
17 (65.4%)
12 (46.2%)
5 (19.2%)
4 (15.4%)
4 (15.4%)
5 (19.2%)