Web Final Report

Content

Business Situation Analysis

The flight search and booking process involves several steps. Each step contributes to the overall booking experience.

First, the user logs in to the aggregator or registers if it is their first time using the system. Next, the user searches for flights and selects one, which generates an order number, passenger information, flight information, and booking time. The aggregator then sends this information to the airline for further processing.

Upon receiving the request, the airline generates an AID and updates the seat information for the selected flight. The airline then responds to the aggregator with a booking confirmation, supported payment platforms, and the AID. This ensures that the user has access to the most up-to-date flight and availability information.

Once the user has selected a payment platform and clicked "pay," the aggregator generates an order number, order total price, and selected payment platform. This information is sent to the airline, which sends the order number, AID, total price, and airline company name to the payment platform for processing.

The payment platform generates an invoice with a PID, AID, order number, total price, and key. The invoice is sent to the airline with the key removed. The invoice serves as an official record of the transaction and provides the user with a clear breakdown of the costs involved.

After the user has confirmed the payment, the payment platform sends the order number and its corresponding key to the aggregator. The aggregator then sends the order number, corresponding key, and payment time to the airline for confirmation. This ensures that the payment has been processed correctly and that the user's reservation is confirmed.

Finally, the airline compares the key and sends the order number and confirmation to the aggregator. This provides the user with a final confirmation of their reservation and ensures that any discrepancies or issues are addressed in a timely manner.

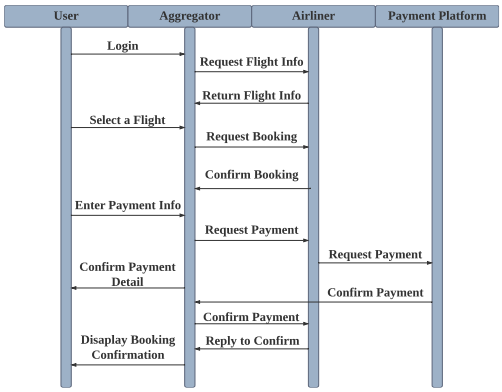

Fig 1 System Sequence Diagram

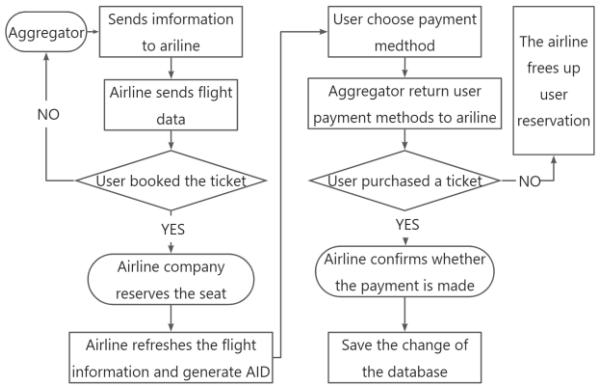

Fig 2 Functional Flow Diagram of Airline

Requirement Documents

Requirement Survey

To meet the diverse requirements of users for air ticket reservation, aggregation and payment, we have investigated and analyzed the current mainstream applications and websites to extract their features. Our requirement survey will also involve gathering user feedback through

surveys and focus groups to identify additional features and functionalities that users would like to see in the platform. This will enable us to develop a comprehensive and user-friendly platform that meets the diverse needs of travelers. Additionally, the project will need to comply with industry standards and regulations to ensure the safety and security of user data and payment information.

Airline

Project Introduction

Project Background and Purpose

To promote their business and to increase the chances of customers finding suitable flights, a group of airlines decided to form an alliance (consortium). One of the aims of this alliance is to develop a distributed web-based system that allows flight aggregator applications (booking engines) to find the most suitable flights for a customer, book seats on a flight, and pay for this online. For this purpose, the alliance also partnered with a number of payment service providers to facilitate online payment for the flights.

Project Scope

1. Search for flight
   Return the corresponding flight information (available seat count, flight number, price, aircraft model) based on the data sent by the aggregator.
2. Undertake booking business.
   Providing detailed flight ticket data to the booking platform, supporting payment methods, and dynamically adjusting the database based on purchase data transmitted from the platform. Verifying payment status with the payment platform.
3. Cancel booking business
   After the user has reserved a seat, if payment has not been made, the reservation can be cancelled, the reserved seat will be released, and the order number and information will be retained. The user's payment will also be automatically canceled after the user has not paid for a period of time.

Functional Requirements

Functional Flow Diagram

Fig2

Airline Terminal Specifications

1. Search for flight

The aggregator sends the user's input restrictions, including time, destination, and departure location, to the airline to query relevant flight information. Upon receiving the request, the airline responds with all flight information that meets the criteria, including details on each flight such as price, remaining seats, and supported payment platforms.

2. Undertake booking business.

After the user places an order, the aggregator sends a message to the airline, and the airline temporarily locks the seat. The user then selects a payment platform and completes the payment.

Upon successful payment, the airline then verifies the payment status and confirms the change to the flight seat availability in the database. If payment fails or times out, the order is canceled and the seat status is restored.

Aggregator

Project Introduction

Project Background and Purpose

The Booking Air Ticket Aggregator project provides a reliable and convenient air ticket booking experience by aggregating information from multiple airline companies into an easy-to-use interface. Users can search for flights and compare prices and schedules across multiple airlines. This all-in-one solution addresses the challenges faced by travelers when booking air tickets, such as comparing prices and schedules to make informed decisions on their travel plans.

Project Scope

1. The application interacts to a web service to help users to search for flights by picking dates, times, and places they want to go.
2. The goal is to simplify the air ticket booking process by aggregating information from multiple airlines in a single platform.
3. Under the payment confirmation of the system, users can safely carry out payment operations.
4. Users can get e-tickets and manage their bookings through the application.

Functional Survey

Functional Flow Diagram

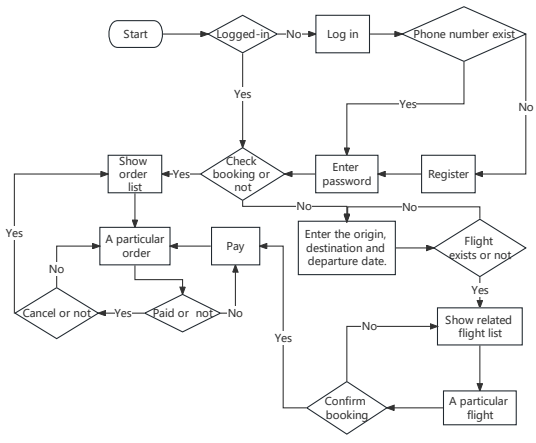

Fig 3 Functional Flow Diagram of Aggregator

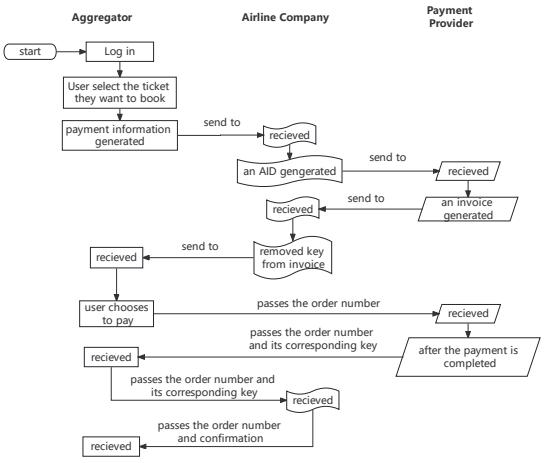

Fig 4 Functional Flow Diagram of Payment Provider

Flight Aggregator Terminal Specifications

1. Users must log in with their mobile phone number to access flight and order information.
2. Flight inquiries require the user to input departure place, destination, and departure date.
3. Paid orders can be cancelled, while unpaid orders can either be paid or cancelled.

Payment Provider

Project Introduction

Project Background and Purpose

A Payment Service Provider acts as a mediator between Booking Air Ticket Aggregator and the Airline Company, making the payment process efficient and seamless for merchants and customers. It validates transaction information and distributes funds to the merchant once a sale is completed. Security of user data, especially payment information, must be strictly guaranteed.

Project Scope

1. basic functions: pay, sign up, sign in, log in
2. deposit: deposit money into your account
3. transfer: transferring money to others
4. balance: check your account balance
5. statement: view all bills

Functional Requirements

Functional Flow Diagram

Fig4

Pay Terminal Specifications

1. Users need to register or login to access the payment function
2. Once logged in the user can pay for previously booked orders or access other functions such as transferring funds, checking balances, viewing statements etc.

Technical Highlights

1. **Overall architecture**
   The system is part of the Django Rest Framework (DRF) project, which uses Django-Python as the back-end framework, along with the RestFul-API architecture to implement three APIs: a flight aggregator, an airline, and a payment service provider.
2. **Database design**
   Django provides built-in support for several types of database backends with built-in support. With just a few lines in our setting.py file, it can support PostgreSQL, MySQL, Oracle or SQLite. we intend to use SQLite, the simplest database, because it can be run with a single file and does not require a complex database installation process. Using SQLite, we designed a database structure listed in later sections. Django imports a modular model to help build the new database model, which will 'model' the characteristics of the data in our database. So we need to provide an explicit name to make it easier for developers, and especially designers, to know what is contained in the template. Technically, the db.sqlite3 file is created when the migration or run server is first run. Using the run server configures the database using Django's default settings, but the migration will synchronize the database with the current state of any database models included in the project and listed in INSTALLED_APPS. In other words, to ensure that the database reflects the current state

of the project, the migration needs to be run (and performed) every time the model is updated when writing this system.

3. **API**
   We used DRF Swagger to design and write the Rest API.
4. **Front-end design**
   Design the front-end interface according to the requirements, including the flight search page, order details page, payment page, etc. The main front-end interface will be run from the command line.
5. **Testing and deployment**
   Writing tests is important as it is the process that can be performed automatically to confirm that the code is working as expected. While in our application we can manually see if the home and about pages exist and contain the expected content. In addition, whenever we make changes to our code - adding new features, updating existing features, removing unused areas of the site - we want to make sure we haven't inadvertently broken other parts of the site. Automated testing allows us to show how we expect a particular part of the project to behave and then let the computer do the checking for us automatically. Django provides robust built-in testing tools for writing and running tests. Since we also need to deal with the database, we use Django.Test.TestCase to perform unit and integration tests during development to ensure system stability and security. After development is complete, the system is deployed to a server to ensure that it is up and running. Deployment can be achieved using container technologies such as Docker.

Working Principle

To develop a Django Rest Framework (DRF) project applied to a system for finding and booking flights, the following steps are followed.

- Integrate three APIs (payment service, airline, flight aggregator) into the system, implement the interface calls and data passing through the routing and view functions provided by Django, and write and implement the Restful-API through the Rest_framework.
- The project is built and applied through django.db to implement the database. The database schema is defined by creating a model using Django's db.models class. Fields can be added, including the data types and default values for the fields.
- Once a model is defined, it can be converted to other data formats such as JSON or XML using DRF's ModelSerializer.
- Views are then created by developing a subclass of DRF's APIView or its corresponding subclasses such as GenericAPIView. These classes are responsible for processing incoming HttpRequest objects and returning a Response object.
- The DefaultRouter provided by DRF can be used to register objects and create the corresponding routing address.
- The DRF Swagger tool provides a visual API interface that can be generated using the get_swager_view() function.
- In addition, the project includes login functionality. django.contrib.auth.urls has a LoginView for implementing login functionality.

Architecture

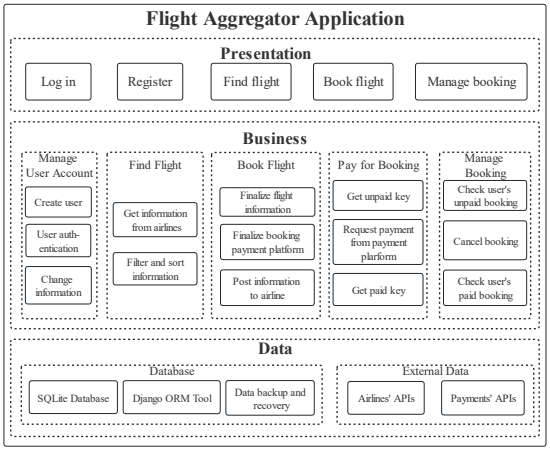

Fig 5 Architecture Diagram of Aggregator

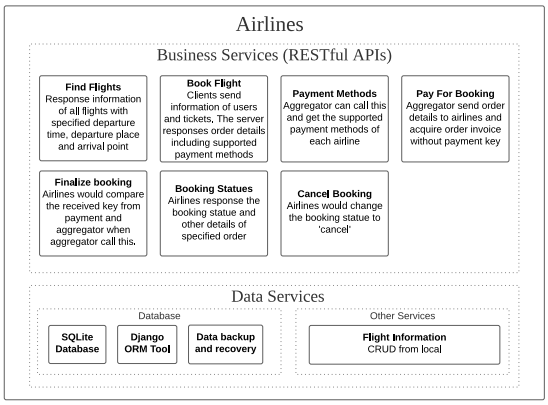

Fig 6 Architecture Diagram of Airlines

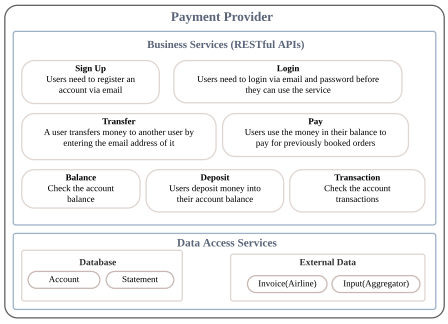

Fig 7 Architecture Diagram of Payment Provider

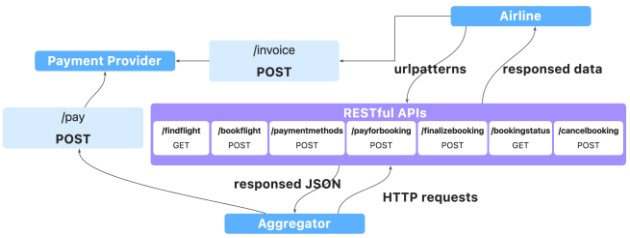

Fig 8 Architecture Diagram of The System

Database

Airline

Table Name	flight		
Description	Airline flight detail information		
Primary Key	**id**		
Field Name	**Description**	**Type**	**Default**
id		INT	Not null
flight_id	Flight number	VARCHAR(10)	Not null
airline	Airline company	VARCHAR(50)	Not null
departure_date	Departure date	DATE	Not null
arrival_date	Arrival date	DATE	Not null
arrival_time	Departure time	TIME	Not null
arrival_time	Arrival time	TIME	Not null
departure	Departure airport	VARCHAR(50)	Not null
arrival	Arrival airport	VARCHAR(50)	Not null
flight_price	Flight price	INT	Not null
seat_number	Available Seats	INT	Not null

Table Name	order		
Description	Airline order detail information		
Primary Key	id		
Field Name	**Description**	**Type**	**Default**
id		INT	Not null
order_id	Order id	INT	Not null
AID	Airline order id	INT	Not null
PID	Payment order id	INT	
flight_id	Flight number	VARCHAR(10)	Not null, FK,refer flight information
order_price	Order price	INT	
ticket_time	Ticketing time	DATETIME	
payment_status	Payment status	TINYINT(1)	0: no pay 1: Paid 2: Cancel paying
key	Electronic payment create key	VARCHAR(50)	
payment_platform	Payment method	VARCHAR(50)	

Table Name	join		
Description	User passenger detail information		
Primary Key	id		
Field Name	**Description**	**Type**	**Default**
id	order and passenger join id	INT	Not null, refer to airline order
order_id	order id	INT	Not null,FK,refer to order
passenger_id	Passenger id	VARCHAR(50)	Not null,FK, refer to passenger

Table Name	passenger		
Description	User passenger detail information		
Primary Key	id		
Field Name	**Description**	**Type**	**Default**
id	Passenger id	INT	Not null
passenger_name	Passenger name	VARCHAR(50)	Not null

Table Name	payment_method		
Description	Airline order detail information		
Primary Key	id		
Field Name	**Description**	**Type**	**Default**
id	Table id	INT	Not null
payment_platform	payment platform	VARCHAR(50)	Not null

Aggregator

Table Name	user		
Description	store the information of user		
Primary Key	id		
Field Name	**Description**	**Type**	**Default**
id	Table id	INT	Not null
phone_number	user phone number	VARCHAR(50)	Not null
status	0: normal 1: unsubscribe	TININT(1)	Not null
password	the password of account	VARCHAR(255)	Not null

Table Name	order		
Description	the user order table		
Primary Key	id		
Field Name	**Description**	**Type**	**Default**
id	Table id	INT	Not null
order_number	order_number	VARCHAR(50)	Not null
airline	airline	VARCHAR(50)	Not null
user_id	user_id	int	Not null
status	0: pay 1:not pay	tinyint(1)	Not null
key	key	VARCHAR(50)	

Paymethod

Table Name	Payment_Account		
Description	Payment information		
Primary Key	id		
Field Name	**Description**	**Type**	**Default**
user_id	User id	INT	Not null
username	Username of User	VARCHAR(50)	Not null
password	Password of User	VARCHAR(50)	Not null
balance	Balance of User	INT	Not null
order_id	Order id	INT	Not null, FK, refer to airline order
name	Real name of user	VARCHAR(50)	Not null
user_id_number	Identification number	INT	Not null
user_phone	Phone number	INT	Not null
user_email	Email address	VARCHAR(50)	Not null

Table Name	Payment_invoice
Description	Payment detail information

Primary Key	PID		
Field Name	**Description**	**Type**	**Default**
PID	Payment id	INT	Not null
AID	Airline id	INT	Not null
Price	Total order price	INT	Not null
Key	Key of ticket	VARCHAR(50)	Not null
order_id	Order id	INT	Not null, FK, refer to airline order
invoice_id	Invoice id	INT	Not null
airline	Name of the Airline	VARCHAR(50)	Not null

Table Name	Payment_order		
Description	Payment detail information		
Primary Key	payment_order_id		
Field Name	**Description**	**Type**	**Default**
user_id	User id	INT	Not null
invoice_time	Time of invoice	DateTime	Not null
invoice_description	Description of invoice	VARCHAR(50)	Not null
Price	Total order price	INT	Not null
status	Income(0) or expenses(1)	INT	Not null

System Interface (API)

Airline

URL	**/findflight**	
Description	Response information of all flights with specified departure time, departure place and arrival point	
Request	GET	
Field	**Description**	**Type**
departure_date	date	Date
departure	departure	String
arrival	arrival	String
Response		


```
{"code": "200", // "503" for fail
 "msg": "successful", // "fail" for fail
 "data": [{“flight_num”: “CA1000”,
          "airline": "Air China",
          "departure_date":"2023/03/08",
          "departure_time":"14:30",
          "arrive_date":"2023/03/08"
          "arrive_time":"23:45",
          "flight_price":2000,
```

"seat_number":999,
"departure":"Chengdu",
"arrival":"London",}]}

URL	/bookflight	
Description	Clients send information of users and tickets. The server responses order details including supported payment methods	
Request	POST	
Field	Description	Type
flight_num	flight number	String
passenger_name	passenger name	List
order_id	order number	String
order_price	order price	Int
ticket_time	ticket time	Time
payment_status	payment status	Int
Response		

{"code": "200", // "503" for fail
"msg": "successful", // "fail" for fail
"data": {“booking_status”: “booking successful”}}

URL	/paymentmethods
Description	Aggregator can call this and get the supported payment methods of each airline
Request	GET
Response	

{"code": "200", // "503" for fail
"msg": "successful", // "fail" for fail
"data": {"payment_platform":["Wechat", "Alipay"],}}

URL	/payforbooking	
Description	Aggregator send order details to airlines and acquire order invoice without payment key	
Request	POST	
Field	Description	Type
payment_platform	payment platform	String
order_id	order number	String
Response		

{"code": "200", // "503" for fail
"msg": "successful", // "fail" for fail
"data": {"AID":"999999999",
"PID":"999999999",
"order_price":"4500",
"order_id":"999999999",}}

URL	/finalizebooking	
Description	Airlines would compare the received key from payment and aggregator when aggregator call this.	
Request	POST	
Field	Description	Type
order_id	Order number	String
key	Payment verification	String
Response		
{"code": "200", // "503" for fail "msg": "successful", // "fail" for fail "data": {"payment_status":"1"}}		

URL	/bookingstatus	
Description	Airlines response the booking statue and other details of specified order	
Request	GET	
Field	Description	Type
order_id	order number	List
Response		
{"code": "200", // "503" for fail "msg": "successful", // "fail" for fail "data": [{"order_id":"4", "payment_status":0, "flight_id": "CA8633", "departure_date": "2023/03/08", "arrive_date": "2023/03/08", "departure_time": "14:30", "arrive_time": "23:45", "departure": "Chengdu", "arrival": "London", "ticket_time":"11:00"},]}		

URL	/cancelbooking	
Description	Airlines would change the booking statue to 'cancel'	
Request	POST	
Field	Description	Type
order_id	order_id	String
Response		
{"code": "200", // "503" for fail "msg": "successful", // "fail" for fail "data"{"booking_status": "booking successfully"}}		

Payment Provider

URL	/signup

Description	User fill in their personal information and create a new account in the payment provider system	
Request	POST	
Field	**Description**	**Type**
userName	real name	String
userIDNumber	identification number	String
userPhone	phone number	Int
userEmail	email	String
userPassword	password for login	String
Response		
{"code": "200", // "503" for fail "msg": "successful", // "fail" for fail "data": {"userId": "1", "userBalance": "0", "username": "Jack", "userIDNumber": "233813200007212421", "userPhone":"13943938738", "userEmail":"3137983@qq.com", "userPassword":"f3fsdd4gdgbfrf5vf5h7$/dadhk03swomdw"}}		

URL	**/signin**	
Description	User can log onto the system by filling in the correct phone number and password.	
Request	POST	
Field	**Description**	**Type**
userPhone	phone number	Int
userPassword	password for login	String
Response		
{"code": "200", // "503" for fail "msg": "successful", // "fail" for fail "data": {"userId": "1", "userPhone": "13943938738", "userPassword":"f3fsdd4gdgbfrf5vf5h7$/dadhk03swomdw", "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ee"}}		

URL	**/deposit**	
Description	User deposits money in the account	
Request	POST	
Field	**Description**	**Type**
depositMoney	deposited money amount	Float
userId	user who deposits	Int
Response		

{"code": "200", // "503" for fail
"msg": "successful", // "fail" for fail
"data": {"userBalance": "200"}}

URL	**/invoice**	
Description	Create invoice based on the order information from airline	
Request	POST	
Field	**Description**	**Type**
orderId	ID number of the order	Int
AID	ID number of the airline	Int
totalAmount	money needs to be paid	Float
airline	airline name	String
Response		

{"code": "200", // "503" for fail
"msg": "successful", // "fail" for fail
"data": {"PID": "999999999",
"AID": "999999999",
"airline": "China Air",
"orderId": "999999999",
"totalAmount": "800",
"payment_status": "1",
"key": "gv7f0sljejks8vm15qpycxb4"}}

URL	**/pay**	
Description	User pay for the invoice	
Request	POST	
Field	**Description**	**Type**
userId	the user who pays	Int
orderId	ID number of the order	Int
Response		

{"code": "200", // "503" for fail
"msg": "successful", // "fail" for fail
"data": {"payment_status": "0",
"payTime": "2023-3-21 18:41:06",
"payDescription": "airline",
"userBalance": "100"}}

URL	**/transfer**	
Description	Transfer money to a specified user's account.	
Request	POST	
Field	**Description**	**Type**
phoneNumber	phone number of the payee	Int
userName	name of the payee	String

userId	the user who transfers	Int
transferMoney	money needs to be transferred	Float
Response		
{"code": "200", // "503" for fail "msg": "successful", // "fail" for fail "data": {"transferStatus": "True", "balance": "100"}}		

URL	**/balance**	
Description	Check account balance	
Request	GET	
Field	**Description**	**Type**
userId	the user who checks balance	Int
Response		
{"code": "200", // "503" for fail "msg": "successful", // "fail" for fail "data": {"balance": "100"}}		

URL	**/statement**	
Description	Look at the bill and the details of each bill.	
Request	GET	
Field	**Description**	**Type**
userId	user who checks statement	Int
Response		
{"code": "200", // "503" for fail "msg": "successful", // "fail" for fail "data": {"payTime": "2023-3-21 18:41:06", "payDescription": "airline", "moneyFlow": "income", // income/expenditure "payAmount": "800" // the amount of this order }}		

Specifications of the Client Applications

The interface between the user and the flight aggregator is described here, defining the ways in which the user is allowed to interact with the system, the inputs and the corresponding outputs.

Register

Description: Allow users to create their aggregator account.

Input: Phone number, password, password again.

Output: Registered successfully (phone number is new to aggregator database and password is confirmed); Phone number already taken; Inconsistent passwords.

Login

Description: Allow registered users to log in the system.

Input: Phone number, password.

Output: Log in successfully (correct phone number and password); Unregistered phone number; Incorrect password.

Find flight

Description: Allow users to find flight with necessary information.

Input: Date, departure location, arrival location.

Output: List of eligible flights

Flight information

Description: Allow users to check the detailed information of the chosen flight.

Input: Chosen flight.

Output: Detailed information.

Book flight

Description: Allow users to book flight from the airline.

Input: Chosen flight, passengers.

Output: Booked successfully; Failed.

Pay for booking

Description: Allow users to pay for booking and confirm booking.

Input: Chosen payment platform.

Output: Successful payment; Fail.

Manage bookings

Description: List the bookings of the user and allow user to manage them.

Input: Nothing but enter the order list.

Output: List of the bookings.

Cancel booking

Description: Allow users to cancel the booking.

Input: Chosen booking.

Output: Cancelled successfully; Failed.

Meeting1

Meeting Date: March 11, 2023
Attending Members:
Group1: Zhao Yimin, Ge Jianhao, lv Haodong, Ren Lingfeng, Zhao Yanbo
Group2: Liu Yunzhu, Chen Xinyi, Ding Yiran, Yao Yifei
Group3: Lou Junhong, Xiao Yang, Zhao Chenhao, Qiu Keyu
Absent Members: None

Main Topics

- Topic 1: Understand the process and grouping

Discussion and Decision

Each member observes domestic and international ticket reservation systems and flight aggregator (including PDF recommendations for British Airways or Turkish Airlines, PayPal or Klarna, Skycanner) to understand the process of searching, booking and paying for tickets, and map them to the subsystem or support service who are responsible for.

There are mainly four parts:

1. An airline web service
2. A payment provider web service
3. A flight aggregator (booking engine) application
4. Supporting services

Members also discussed general user needs:

1. Find the most suitable flights for a journey
2. Book seats on certain flights
3. Use payment services to pay for bookings
4. Receive electronic tickets
5. Manage their bookings

Afterwards, the members were grouped into three teams based on their corresponding systems, which has been written above.

Next Steps

1. Determine the parts that each subsystem contain
2. Determine their corresponding user requirements.
3. Find a suitable document template.

Next Meeting

Meeting Date: March 18, 2023

Topic

- Topic 1: Detailed task assignment for each group

Meeting2

Meeting Date: March 18, 2023
Attending Members:
Group1: Zhao Yimin, Ge Jianhao, lv Haodong, Ren Lingfeng, Zhao Yanbo
Group2: Liu Yunzhu, Chen Xinyi, Ding Yiran, Yao Yifei
Group3: Lou Junhong, Xiao Yang, Zhao Chenhao, Qiu Keyu
Absent Members: None

Main Topics

- Topic 1: Detailed task assignment for each group

Discussion and Decision

We have confirmed the three main responsibilities, and each of them can still be further divided into sub-tasks.

1. Document the process:
   After a brief discussion, we decided to have each person document their tasks to facilitate communication between the different groups. After completion, we will have a discussion between the groups to determine whether tasks can be merged and optimized, potentially increasing efficiency.
2. Learn web system architecture:
   Each group will select several individuals to learn and build the web system architecture. One member from each group will be assigned to search for tutorials and materials to learn about web system architecture. They will first draw a high-level architecture diagram and map their group's assigned subsystems (including their respective small functions) to the architecture diagram. After completion, we will invite members from the other two groups to discuss and finalize the overall architecture diagram.
   (Zhao Yimin, Liu Yunzhu, Ding Yiran, Ren Lingfeng, Qiu Keyu)
   Each group will design the architecture for their required API and database (Zhao Yanbo, lv Haodong, Xiao Yang), and the responsible members from each group will need to communicate with each other to determine the necessary components.
3. Document writing
   Each group should assign one member to learn how to write requirement documents (Ge Jianhao, Yao Yifei, Zhao Chenhao) and another member to learn how to write design documents (Chen Xinyi). During this process, three members responsible for the same task can discuss and generate templates and examples for the requirement and design documents. The requirement document should be completed first so that the group members responsible for building the Web API and database can use it as a reference for their designs.
4. Meeting minutes
   Based on the simple document recorded during the meeting, write the full meeting minutes. (Lou Junhong)

Next Steps

1. Subsystem procedures
2. Determine the overall architecture
3. Determine the document format

Next Meeting

Meeting Date: March 25-28, 2023

Topic

- Topic 1: Determine the main functions of the three subsystem

Payment Provider Web Service Group

Meeting3

Meeting Date: March 28, 2023
Attending Members: Liu Yunzhu, Chen Xinyi, Yao Yifei, Ding Yiran
Absent Members: None

Main Topics

- Topic 1: The main functions of the payment provider system
- Topic 2: Assign tasks

Discussion and Decision

1. Discuss the main functions of the payment provider system
    - Payment provider system should contain functions of register, login, pay, deposit, transfer, check balance and statement.
    - The invoice information can store locally and then call it up when paying.
2. Assign tasks
    - Chen Xinyi is responsible for the templates of design document, and communicate with members in other team who takes charge of the same task to determine the final template.
    - Yao Yifei is responsible for the templates of requirements document, and communicate with members in other team who takes charge of the same task to determine the final template.
    - Liu Yunzhu and Ding Yiran are responsible for the design of architecture of payment system, and contact with other team members to determine the interaction among three systems.

Next Meeting

Meeting Date: April 4, 2023

Topic

- Topic 1: The completion of last task
- Topic 2: Data exchange in payment system
- Topic 3: Assign tasks

Meeting4

Meeting Date: April 4, 2023
Attending Members: Liu Yunzhu, Chen Xinyi, Yao Yifei, Ding Yiran
Absent Members: None

Main Topics

- Topic 1: The completion of last task
- Topic 2: Data exchange in payment system
- Topic3: Assign tasks

Discussion and Decision

1. The completion of last task
    - Finish the search for templates of design and requirements documents, draw a draft of the system architecture.

2. Data exchange in payment system
    - The payment provider system should receive the information of flights from the airline.
    - The system creates invoice and a key consisting of information of the flights, then send it to the airline again.
3. Assign tasks
    - Chen Xinyi completes the writing of design documents and the design of databases.
    - Yao Yifei completes the writing of requirements documents and the design of api.
    - Ding Yiran is responsible for the writing of interface documents and the meeting minutes.
    - Liu Yunzhu is responsible for the system architecture and the workflow of the system.

Next Meeting

Meeting Date: April 11, 2023

Topic

- Topic 1: The completion of last task
- Topic 2: The details in the system
- Topic 3: Assign tasks

Meeting5

Meeting Date: April 11, 2023

Attending Members: Liu Yunzhu, Chen Xinyi, Yao Yifei, Ding Yiran

Absent Members: None

Main Topics

- Topic 1: The completion of last task
- Topic 2: The details in the system
- Topic 3: Assign tasks

Discussion and Decision

1. The completion of last task
    - Finish the design of databases and api, writing of design and requirements documents, drawing of the system architecture.
    - Continue to perfect the content of documents, contact members in other teams to ensure the data change between systems work successfully.
2. The details in the system
    - The payment provider system should store the information of transaction details of each order.
    - The system should contain a function of creating invoice.
3. Assign tasks
    - Yao Yifei completes the requirements document of the payment system and integrate the whole system requirements documents.
    - Ding Yiran integrates the interface documents of the whole system and records the meeting minute.
    - Chen Xinyi extracts technical points and working principles from labs 1-4.
    - Liu Yunzhu completes the architecture diagram involving interaction.

Flight aggregator (booking engine) application group

Meeting3

Meeting Date: March 25, 2023
Attending Members: YangXiao, Chenghao Zhao, Junhong Lou, Keyu Qiu
Absent Members: None

Main Topics

- Topic 1: Determine the format of the requirement document and complete the content of the requirement document
- Topic 2: Determine the format of the design document and complete the content of the design document
- Topic 3: Confirm the format of the architecture diagram and complete the team system architecture diagram

Discussion and Decision

Meeting Discussion

1. Determine the format of requirements and design documents (15 minutes)
    - The host invites Xiao Yang and Zhao Chenhao, the leaders of the requirements document and design document, to introduce the main content and format requirements of the document.
    - Participants discuss and determine the unified format and style of the document, and raise any questions or suggestions.
    - The host summarizes the discussion results and confirms whether there is a need to modify or supplement the document format.
2. Complete the content of the requirements and design documents (30 minutes)
    - The host invites Xiao Yang and Zhao Chenhao, the leaders of the requirements and design documents, to introduce the main chapters and details of the document.
    - Participants jointly modify the content of the document and raise any questions or feedback.
    - The host summarizes the writing situation and confirms whether there is a need to modify or supplement the document content.
3. Confirm the format of the architecture diagram (10 minutes)
    - The person in charge of the architecture diagram asks Ke Yu to introduce the main elements and format requirements of the architecture diagram.
    - Participants discuss and determine the unified format and style of the architecture diagram, and raise any questions or suggestions.
    - The host summarizes the discussion results and confirms whether there is a need to modify or supplement the frame composition format.
4. Complete the team system architecture diagram (30 minutes)
    - The person in charge of the architecture diagram asks Ke Yu to introduce the main components and relationships of the system architecture diagram.
    - Participants use the language sparrow to jointly draw a system architecture diagram and raise any questions or feedback.
    - The host summarizes the drawing situation and confirms whether there is a need to

modify or optimize the system architecture diagram.

Meeting Decision

1. The team successfully determined the format of the requirement document and completed its content.
2. The team successfully determined the format of the design document and completed its content.
3. The team successfully confirmed the format of the architecture diagram and completed the team system architecture diagram.

Next Steps

1. The team starts assigning tasks to each team member
2. Ke Yuqiu will be responsible for overseeing the project and ensuring that it stays on track.
3. The team will meet again in one weeks to review progress and discuss any issues that may have arisen.

Next Meeting

Meeting Date: April 1, 2023

Topic

- Topic 1: Discuss document content and templates with relevant designers from other groups and resolve remaining issues
- Topic 2: Modify the contents of design documents, focusing on API design and database design.
- Topic 3: Review the content of all documents to identify and correct any issues.

Meeting4

Meeting Date: April 1, 2023

Attending Members: YangXiao, Chenghao Zhao, Junhong Lou, Keyu Qiu

Absent Members: None

Main Topics

- Topic 1: Discuss document content and templates with relevant designers from other groups and resolve remaining issues
- Topic 2: Modify the contents of design documents, focusing on API design and database design.
- Topic 3: Review the content of all documents to identify and correct any issues.

Discussion and Decision

Meeting Discussion

1. Discuss document content and templates with relevant designers from other groups and resolve remaining issues (30 minutes)
    - The host invites the responsible person for the content and template of the design document and requirement document to introduce their communication with other groups and the results.
    - Participants discuss and determine unified standards and styles for document content and templates, and raise any questions or suggestions.
    - The host summarizes the discussion results and confirms whether there is a need to modify or supplement the document content and template.

2. Modify the content of the design document (45 minutes)
    - The host invited Xiao Yang, the person in charge of API design and database design, to introduce the main contents of the revised design document, focusing on API design and database design.
    - Participants review the revised design document chapter by chapter and raise any questions or suggestions.
    - The host summarizes the review results and confirms whether further modifications or supplements are needed to the design documents.
3. Review the content of all documents again, identify and correct any issues (30 minutes)
    - The host invites the person in charge of all documents to introduce the issues and corrective measures discovered after their re review.
    - Participants conduct a final review of all documents and raise any questions or feedback.
    - The host summarizes the inspection results and confirms whether final modifications or supplements are needed to all documents.

Meeting Decision

1. The team successfully discussed and modified document content and templates with relevant designers from other groups.
2. The team further revised the content of the design document, focusing on API design and database design.
3. The team reviewed the content of all documents again and identified and corrected any issues.

Next steps

1. The team will continue to work on refining and finalizing all documents.
2. Ke Yuqiu will continue to oversee the project and ensure that it stays on track.
3. The team will meet again in one weeks to review progress and discuss any issues that may have arisen.

Next Meeting

Meeting Date: April 8, 2023

Topic

- Topic 1: Review and Finalize Design Documents: Teams can review design documents to ensure their completeness and accuracy.
- Topic 2: Discuss and solve any remaining issues: The team can discuss and solve any remaining issues or challenges related to the design documents.
- Topic 3: Plan Next Steps: The team can discuss and plan to continue developing the flight aggregator application.

Meeting5

Meeting Date: April 8, 2023

Attending Members: YangXiao, Chenghao Zhao, Junhong Lou, Keyu Qiu

Absent Members: None

Main Topics

- Topic 1: Review and finalize design document.
- Topic 2: Discuss and resolve any remaining issues

- Topic 3: Plan next steps.

Discussion and Decision

Meeting Discussion

1. Review and finalize design documents (45 minutes)
    - The host invites the person in charge of designing the document to introduce the main content of the document.
    - Participants review the document chapter by chapter and raise any questions or suggestions.
    - The host summarizes the review results and confirms whether there is a need to modify or supplement the document.
2. Discuss and resolve any remaining issues (30 minutes)
    - The host invites the person in charge of problem-solving to introduce the problems and challenges they have encountered.
    - Participants share their experience and lessons learned in problem-solving and propose any solutions or suggestions.
    - The host summarizes the issues and confirms whether communication or coordination with other groups is necessary.

Meeting Decision

1. The team successfully reviewed and finalized the design document.
2. The team discussed and resolved any remaining issues or challenges related to the design document.
3. The team planned the next steps for continuing to develop the flight aggregator application.

Next steps

1. The team will continue to develop flight aggregator applications based on the design documents.
2. Ke Yuqiu will continue to supervise the project and ensure that it remains on track.
3. The team will meet again in two weeks to review progress and discuss any issues that may arise.

Airline web service group

Meeting3

Meeting Date: March 24, 2023
Attending Members: Zhao Yimin, Ge Jianhao, lv Haodong, Ren Lingfeng, Zhao Yanbo
Absent Members: None

Main Topics

- Topic1: Requirement specification
- Topic2: Architecture Design
- Topic3: API specification
- Topic4: Database specification

Discussion and Decision

1. Requirement Improvement (Ge Jianhao)
   After confirming that no refunds are required, the refund option was removed.

Following communication with the aggregator, flight queries were changed to a one-time retrieval of all information. The payment process was discussed in detail with the payment platform.

2. Architecture Improvement (Zhao Yimin, Ren Lingfeng)
   The architecture was significantly improved, and the tools required for the database were detailed. Improvements to the API were established, and the use of POST and GET was determined. The payment confirmation process with the payment platform and aggregator was re-confirmed.
3. API Improvement (lv Haodong, Zhao Yanbo)
   The API was combined /findallFlight and /findFlight into one and now sends all information for the queried flight. Each field and type was thoroughly improved, and every possible return value was documented. The get/post relationships of some APIs were changed, and suggestions from other systems were incorporated, like /bookingStatus for confirming and /cancelBooking.
4. Database Improvement (lv Haodong)
   We have changed some of the tables, make some of the detail change, like change the order_id's type. Also, changed the one-to-many to many-to-many relationship, by adding a Join table to connect the order table with the passenger table.

Next Steps

1. Discussions held with other groups; continued improvement was made to achieve the best web architecture design.

Next Meeting

Meeting Date: March 31, 2023

Topic

- Topic 1: Improvement

Meeting4

Meeting Date: March 31, 2023
Attending Members: Zhao Yimin, Ge Jianhao, lv Haodong, Ren Lingfeng, Zhao Yanbo
Absent Members: None

Main Topics

- Topic1: Requirement Improvement
- Topic2: Architecture Improvement
- Topic3: API Improvement
- Topic4: Database Improvement

Discussion and Decision

1. Requirement specification (Ge Jianhao)
   After discussing with other teams, we have identified three main demands from the airlines. 1. Search for flight 2. Undertake booking business. 3. Refund. And we have confirmed the rough process for these two demands.
   The aggregator sends the user's input restrictions to the airline to query corresponding flight information. After receiving the request, the airline will return the corresponding flight information, including detailed information for each flight (price, remaining seats, etc.). After the user places an order, the airline will temporarily lock the seat and send

a payment request to the corresponding payment platform. After a successful payment, the payment platform will return a value to notify the airline and the booking platform, and the airline will confirm the changes to the database. If the payment fails, the order will be cancelled and the seat status will be restored.

2. Architecture Design (Zhao Yimin, Ren Lingfeng)
   We have drafted an initial organizational chart to assist with the development of APIs and the database.
3. API specification (lv Haodong, Zhao Yanbo)
   Based on the requirements, the following APIs have been developed:
   a. Flight search API (/findAllFlight): This API allows the aggregator to send a request to the airline to search for available flights based on user input restrictions.
   b. Flight information API (/findFlight): This API enables the airline to return detailed information about each flight, including price and remaining seats, to the aggregator.
   c. Reservation API (/bookFlight): This API allows the airline to temporarily lock the seat and send a payment request to the corresponding payment platform after the user places an order.
   d. Payment verification API (/paymentMethod /payForBooking): These API enables the airline to verify the payment status with the payment platform and confirm the changes to the database. Also, To make the aggregator aware of the payment methods supported by each flight.
   e. Refund API: This API allows the payment platform to request a refund from the airline, and enables the airline to check if the refund criteria are met and send a refund request to the payment platform.
4. Database specification (lv Haodong)
   We have created a total of 5 tables, which are:
   a. Flight: Used for recording flight message
   b. Order: Used for recording order message
   c. Passenger: Used for recording Passenger message
   d. Payment method: Used for recording Passenger message

The primary and foreign keys for each table have been roughly determined to facilitate communication with other systems. Order data and passenger data are connected using a one-to-many relationship.

Next Steps

1. Reconnect with members from other groups to discuss and improve specific web designs.

Next Meeting

Meeting Date: April 7, 2023

Topic

- Topic 1: Improvement

Meeting5

Meeting Date: April 7, 2023
Attending Members: Zhao Yimin, Ge Jianhao, lv Haodong, Ren Lingfeng, Zhao Yanbo
Absent Members: None

Main Topics

- Topic1: Analyze and change system differences

Discussion and Decision

1. Look for any differences in column names and field names between the database and the API. (Lv Haodong and Zhao Yanbo)
2. First, find out if there is any dissatisfaction in the business process between the aggregator and the airline, and unify the data incoming and outgoing formats of the API. Secondly, the name of the API is unified and determined, and unnecessary data names are removed. Then, the API names that have been identified are unified and sorted out. (Xiao Yang, Zhao Yanbo, Lv Haodong, Qiu Keyu, Zhao Chenhao)
3. On the premise of future code implementation, the relevant format is decided and improved, which is the basis for CW2 development.(Ge Jianhao, Ren Lingfeng)