System Model and Design: Tripedia

Yulin Zhang	Yuhang Chen	Jiaxi Yang	Boyang Wang
7th Group	*7th Group*	*7th Group*	*7th Group*
Software Engineering	*Software Engineering*	*Software Engineering*	*Software Engineering*
Montreal, Canada	Montreal, Canada	Montreal, Canada	Montreal, Canada
silveralex2023820@gmail.com	yuhang.chen@mail.concordia.ca	yjxyang2@outlook.com	wangboyang0626@outlook.com

I. THE AGILE PROCESS

A. *Team Setup*

In our group, personnel are assigned as follows:

Yuhang Chen - Sprint 1: Product Owner, Sprint 2: Scrum Master, Sprint 3: Architect, Sprint 4: Development

Jiaxi Yang - Sprint 1: Development, Sprint 2: Product Owner, Sprint 3: Scrum Master, Sprint 4: Architect

Yulin Zhang - Sprint 1: Architect, Sprint 2: Development, Sprint 3: Product Owner, Sprint 4: Scrum Master

Boyang Wang - Sprint 1: Scrum Master, Sprint 2: Architect, Sprint 3: Development, Sprint 4: Product Owner

Our roles rotate every sprint.

B. *Association between User Stories and Sub-requirements*

According to the following User Stories which we mention at delivery1.

1) The traveler opens the travel website without logging in, the website will collect his location, so that the website can show local people's favorite place to visit which provides a reference for the user.
2) The traveler registers an account and logs in to the Tripedia website.
3) The traveler finishes a questionnaire so that the website can recommend appropriate information for users based on the questionnaire content.
4) The traveler has no idea about the where are the best places for him. He finds the data analysis diagram panel for more information to make a decision.
5) The traveler searches for his destination and then the website recommends related information (e.t. hotel, transport function, Landmark Tickets, scenic spot review) so that he can make plans by reference.
6) The traveler checks suitable hotel timings so that he can book an appropriate hotel.
7) The traveler checks suitable flight or train timing so that he can book an appropriate transport function.
8) The traveler selects his favorite hotel and then clicks to jump to the booking screen so that he can make payment.

We have extracted the following sub-requirements

1) Login&Register
2) Account Edit
3) Searching data from database
4) Crawler searching
5) The analysis Diagram

C. *The Scrum Sprint Cycle*

1) Review Work to be done: Currently, our system infrastructure is completed. In the next scrum sprint cycle, we will focus on the incremental functions to boost user experience, such as updating user interfaces, and destination recommendations.

2) Sprint Plan: We have experienced two sprint cycles right now. We are designing the requirements for the next two sprint cycles(the 3rd and 4th sprint cycles).

* The first sprint cycle: 30th September - 12th October
* The second sprint cycle: 13th October - 28th October
* The third sprint cycle: 28th October - 20th November

3) Product Backlog & Sprint Backlog: As shown in Figure 1, the requirements shown in the table are the product backlogs. And the Sprint Backlogs in Sprint 1,2,3 are clear in this table.

4) Sprint Process: In the first sprint cycle, we have developed a ser ies of basic features with MUST priority, such as Home Page, Diagram Display, Crawler, etc. In this stage, we focus on the development work of basic infrastructure, such as the crawler, and database interface encapsulation. Based on the stable architecture, we have developed an initial version system and satisfied the requirements of the ICDE.

In the second and third sprints, we increased the functions to boost the user experience. For example, users can edit their accounts in the second version. Meanwhile, users can search for hotels and tickets in the second system. Moreover, the data analyzer not only provides the analyzed results of the recommendation but also tells users why the recommendation is the best. At the same time, we have fixed a great number of bugs in our system.

Sprint 1

Meeting Minutes: Scrum Sprint Cycle Review
Date: 2023.09.30
Time: 10:00 AM
Location: EV Building
Meeting Objective: Determine the main structure and content of the project
Attendees: Yuhang Chen, JiaxiYang, Yulin Zhang, Boyang Wang
Agenda:

Requirement	Sub-Requirement	Task	man/hour	Sprints	Priority	Status
Web Page	Web Page Design	The arrangement of web pages.	15	1	Must	Done
	Web Page Impl	Users wanna a clear page to do operations, such as search web page, home page, account page, etc.	20	2	Must	Done
Account	Login & Validation	User logins his/her own account.	5	1	Must	Done
	Register & Validation	User create his/her own account.	5	1	Must	Done
	Security Questions	Add security question to increase the safety.	8	3	Could	Done
	Verification Code	System provide more safe method for security.	5	3	Could	To Do
	Account Edit	User can edit his/her own account information.	10	2	Could	Done
Search	Search Hotel	User wanna search appropriate hotels information.	15	1	Must	Done
	Search Destination	User wanna search search destination information with data analysis.	5	3	Must	In Progress
	Search Tickets	User wanna search tickets information.	5	2	Could	Done
	Crawler	Capture information from internet.	10	1	Must	Done
	Recommend Items	Recommend items when user search sth.	5	3	Could	To Do
	Auto Complete	The search block can auto complete users' search item.	5	2	Could	Done
Diagram Analysis(ICDE)	Data Analysis	Design a flexible algorithm for data analysis.	10	1	Must	Done
	Diagram Display	Display the data analysis result.	5	2	Must	Done
	Questionnaire	Capture the user preference data.	10	2	Must	Done
	Analysis Introduction	Users can know the reason for recommendation.	10	3	Could	Done
Data Base Deployment	Data Middleware	System provide stable data operation methods.	8	1	Must	Done
	Cloud Deployment	Users can access the website anywhere.	2	2	Must	Done
User Interface	Interface	Update the user interface for a better user experience	10	3	Could	To Do

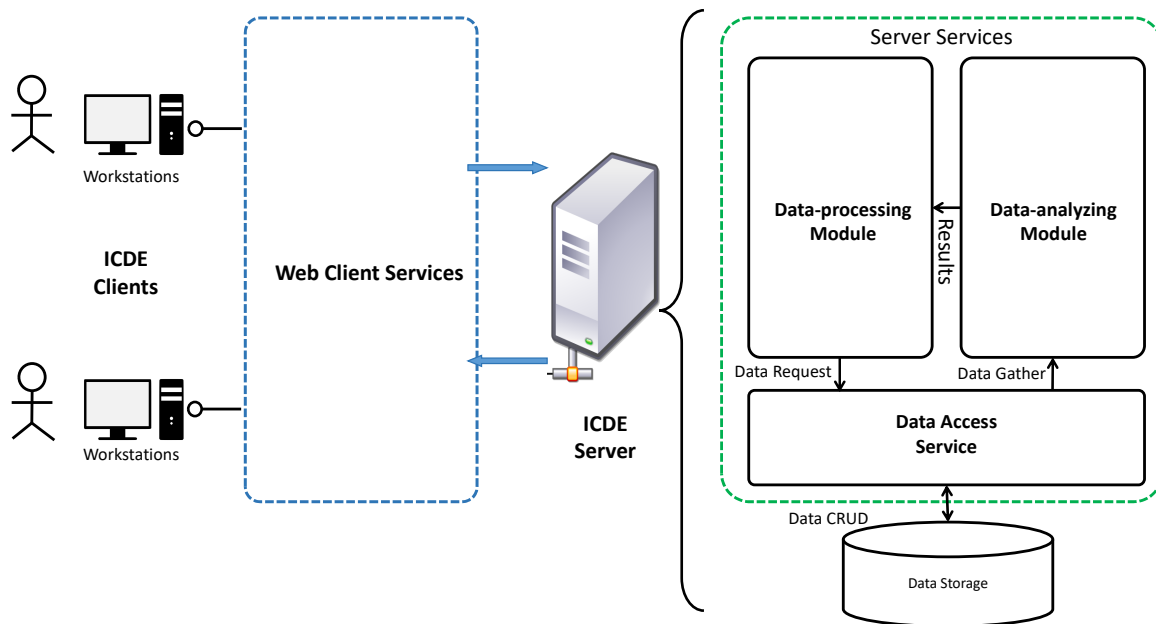Fig. 1. The development process of our system.

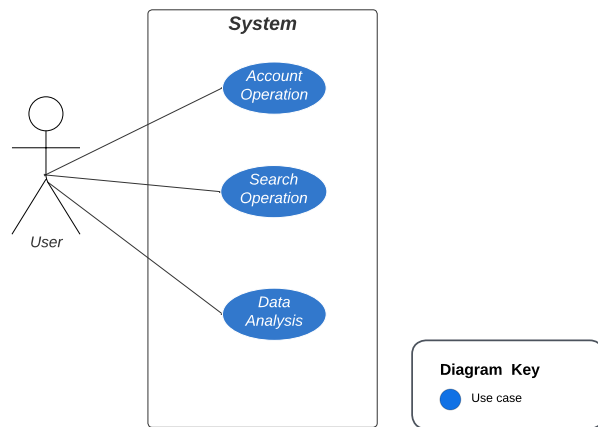Fig. 2. The high-level architecture graph of Tripedia.

Fig. 3. The overall level architecture graph of Tripedia.

Identify data sources for the project.

Determine the name of the project.

Determine the structure of the web interface

Determine what charts are displayed on the website.

Determine what each person is responsible for.

Discussions:

Data sources [Booking.com, Expeida.com]

Name of the project.[Tripedia]

Structure of the web interface[Our project has 4 main pages, HomePage, Login/Register Page, SearchPage, Analysis Page]

Charts [Pie Charts and Bar Charts shows Recommend City]

Responsible Section:

- SearchPage: Yuhang Chen
- Login/Register Page: Jiaxi Yang
- Analysis Page: Yulin Zhang
- Home Page: Boyang Wang

Action Items:

[SearchPage], Responsible: [Yuhang Chen], Due: [2023.10.10]

[Login/Register Page], Responsible: [Jiaxi Yang], Due: [2023.10.10]

[Analysis Page], Responsible: [Yulin Zhang], Due: [2023.10.10]

[Home Page], Responsible: [Boyang Wang], Due: [2023.10.10]

Next Meeting:

Date: [2023.10.15]

Meeting Adjourned: [10:00 AM]

Sprint 2

Meeting Minutes: Scrum Sprint Cycle Review

Date: 2023.10.13

Time: 10:20 AM

Location: EV Building

Meeting Objective: Design the database structure and determine implementation methods for various functionalities of the project.

Attendees: Yuhang Chen, Jiaxi Yang, Yulin Zhang, Boyang Wang

Agenda:

Design the database structure.

Determine the implementation methods for login and registration functionalities.

Decide on the utilization of survey questionnaires for the ICDE intelligent recommendation system.

Determine the implementation methods for the hotel search functionality.

Discussions:

Database Structure

A comprehensive discussion was held to design the database structure to support the project's functionalities effectively.

Login and Registration Functionality

Various methods were explored, and a final approach was determined to implement the login and registration functionalities. + ICDE Intelligent Recommendation System

It was decided to use survey questionnaires to implement the ICDE intelligent recommendation system.

Hotel Search Functionality

Discussion on the methods to be used for implementing the hotel search functionality, and a method was finalized.

Responsible Section

- Design Database: Boyang Wang
- Login/Register Function: Jiaxi Yang
- Intelligent Recommendation Function: Yulin Zhang
- Search Hotel Function: Yuhang Chen

Action Items: Design the database structure, Responsible: Boyang Wang, Due: [Oct.14.2023]

Develop the login and registration functionalities, Responsible: Jiaxi Yang, Due: [Oct.19.2023]

Implement the ICDE intelligent recommendation system, Responsible: Yulin Zhang, Due: [Oct.19.2023]

Create the hotel search functionality, Responsible: Yuhang Chen, Due: [Oct.19.2023]

Next Meeting:

Date: [2023.10.30]

Meeting Adjourned: [11:50]

Sprint 3

Meeting Minutes: Scrum Sprint Cycle Review

Date: 2023.10.28

Time: 10:20 AM

Location: EV Building

Meeting Objective: Design the database interface, search ticket function, edit account requirements, analyze introductions, and review the functions that have been developed.

Attendees: Yulin Zhang, Yuhang Chen, Jiaxi Yang, Boyang Wang

Agenda:

Design the unique database interfaces for usability.

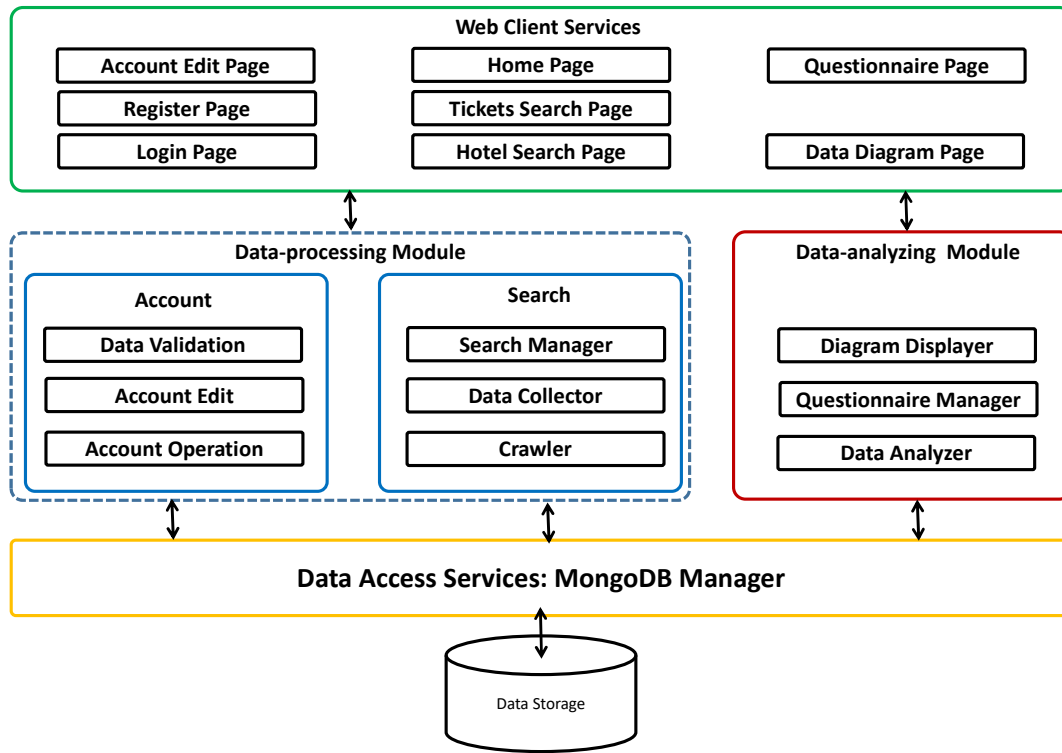Determine the introduction of the analysis description.

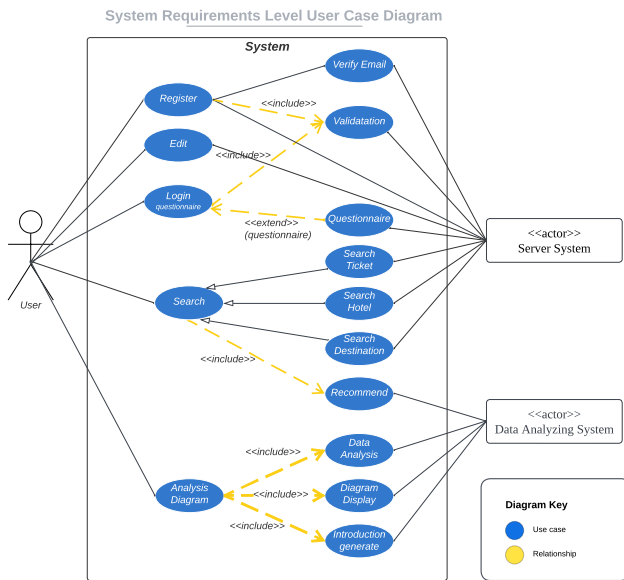Fig. 4. The system-level architecture graph of Tripedia.

Fig. 5. The requirements level architecture graph of Tripedia.

Decide on the account incremental function.

Determine the implementation of ticket search and initial recommendations.

Discussions:

(1) Account Incremental Function

The account edit function is essential for the user to change their account information. It is important to boost user experience.

(2) Search Incremental Functions

A comprehensive discussion was held to add a search ticket function. It is a basic function for users to check the tickets of the destination.

(3) ICDE Intelligent Recommendation System Incremental Function

The recommended result of the data analysis system may make users confused. Considering this issue, we decided to add an auto description of our recommendation.

Responsible Section

- Account Function: Jiaxi Yang
- Analysis Description: Yulin Zhang
- Search Ticket Function: Yuhang Chen
- Database Interface Development: Yulin Zhang, Jiaxi Yang, Yuhang Chen, Boyang Wang

Action Items:

Develop the Account Component which integrates login, register, and edit, Responsible: Jiaxi Yang, Due: [Nov.7.2023]

Develop the analysis descriptions and rebuild the processing steps: Yulin Zhang, Due: [Nov.9.2023]

Create the ticket search functionality, Responsible: Yuhang Chen, Due: [Nov.8.2023]

Design the database interface, Responsible: Yulin Zhang, Jiaxi Yang, Yuhang Chen, Boyang Wang, Due: [Nov.10.2023]

Next Meeting:

Date: [2023.11.13]

Meeting Adjourned: [11:50]

5) Sprint Review: Above all, the development process of Goupe 7 is steady and fast. We try our best to use the software engineering technique to boost our development process, which has a surprising result right now.

II. ARCHITECTURE

A. Architecture Design in Overall Level

As shown in Figure2, the high-level ICDE system architecture mainly contains two parts: client and server. Based on the limited server hardware equipment, we implement the data analysis logic as a single module in the server, which stands for ICDE 3rd Party Applications.

In the client part, we implement Tripedia's client logic based on the web. The web client service mainly contains user assistance services and data capture services. The web pages provided by the web services are the operation interface for users.

On the server side, we will develop three modules to implement the backbone of the ICDE system: Data-processing Module, Data-analyzing Module, and Data Access Service. The Data Access Service, as the median layer for database access, provides a unified database interface (deletion, modification, query, etc) for the Data-processing Module and the Data-analyzing module. The Data-processing Module is responsible for the basic logic of the server, such as registration, login, search, data preprocessing, and other interaction logic with the client. The Data-analyzing Module accesses the data in the database through the Data Access Service to perform data analysis and returns the analysis results to the Data-processing Module.

The Figure 3 shows the overall level user case. This figure shows the visible functions for users. As for the user, the services provided by the system are clear, such as account operation, search Operation, and Data Analysis.

To sum up, Tripedia's architecture is a standard ICDE system. The Client part includes a Data Capture Service to collect user behavior and information and transmit relevant data to the server. The Data-processing Module in the Servers section is responsible for implementing the interactive logic of the server; the Data Access Service provides a unified data access interface; the Data-analyzing Module serves as the core of the ICDE system to analyze user data and feedback the analysis results to the user through the server to provide User Assistance Service.

B. Architecture Design in User Requirements Level

The system-level architecture is shown in Figure 4. The main layout of the system-level architecture diagram is similar to the overall level architecture diagram. The system-level architecture shows the functions in each Module in detail. Considering the features of our system, we design our system-level architecture based on Layer Information System Architecture and Processing Application Architecture.

The web client services contain a series of web pages, which provide operation interfaces for users.

The data-processing module can be divided into two parts: Account and Search. The Account part focuses on the requirements of account operation. Meanwhile, the Search part is responsible for searching data from both networks and the database.

The data-analyzing module is responsible for data analysis and diagram displaying, which play the role of data processing in the ICDE system.

The Figure 5 shows the requirements level user case. This figure shows the visible functions for users in detail. Furthermore, the user case of the system actor is shown in this figure, which is invisible to users. We also consider the relationships among these user cases.

The MongoDB Manager is a database operation mediate layer, which increases the reusability of the database interface.

III. MODELING AND DESIGN

In this section, we will discuss the design of the tasks of the Tripedia. For a clearer structural presentation, we consider the choice of task description granularity. A small task granularity can not take on the task of showing interactions among multiple objects. A large task granularity may lose the meaning of the task design. Therefore, we merged some sub-requirements into a single task. We try to show the relationships among objects as much as possible while ensuring appropriate granularity.

A. Task: Login & Register

1) Task Context :

This class diagram meticulously outlines the architecture of a user login and registration system, representing a coalescence of strategically organized classes - the 'user,' 'UserDAO,' and 'Database(MongoDB)' - each playing a pivotal role in safeguarding and proficiently managing user information.

The 'user' class is the cornerstone, embodying essential attributes such as username, password, and email alongside crucial functionalities that facilitate user login and registration processes. It acts as the primary interface, allowing users to interact with the system ensuring their credentials are accurately captured and securely managed.

The 'UserDAO' class emerges as a vital intermediary, bridging the 'user' class and the 'Database(MongoDB)' class. It encapsulates many user-related database operations, including user creation and data retrieval, fostering a seamless and efficient management of user data stored within the database.

Lastly, the 'Database(MongoDB)' class is dedicated to direct interactions with the MongoDB database, ingrained with essential attributes and functionalities required for initiating and managing connections, emphasizing the robustness and reliability of the database operations.

Together, these classes form a cohesive system aimed at simplifying the user login and registration process while

prioritizing the integrity and security of user data. Through precise role allocations and harmonious class collaboration, the system is empowered to process user requests proficiently, ensuring a streamlined and secure user login and registration experience.
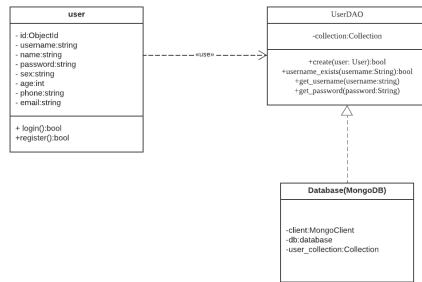
Incorrect details will circulate the user back to re-enter the correct credentials, while a successful validation will grant the user access, redirecting them to the homepage.

Throughout this process, each object and decision node plays a critical role in managing the data flow and ensuring secure and correct user authentication. The precise coordination and interaction among these objects and decision nodes are pivotal for a seamless and secure user experience accessing the system. The process guarantees that only registered and validated users gain access, preserving the integrity and security of the system and its data.
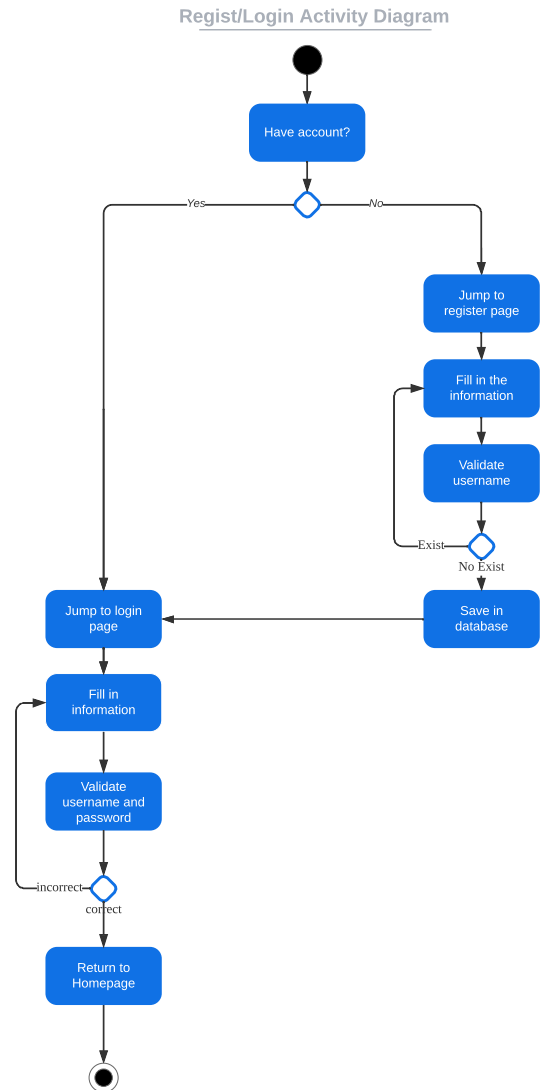

Fig. 6. Class diagram of Login and Register

2) Structural Context :

According to the login and registration activity diagram, we elucidate the structural context and task data flow, mapping the journey of a user through the system and delineating the roles and interactions that various elements undertake during user authentication processes.

Initially, the user encounters a decision node, questioning the existence of an account. Lacking an account directs the user towards the registration process, where they populate necessary fields with relevant information, initiating a verification process and ensuring the uniqueness of the username within the system database. Once validated, the user's data is secured in the database, concluding the registration. If the user possesses an account, they navigate towards the login page. Here, the user enters essential login credentials - username and password. The system then validates these credentials, checking their conformity with the stored data.

Fig. 7. Activity diagram of Login and Register

3) Interaction Context:
Registration Process:

The actor (user) begins registering by entering their information on the registration page.

The Register Page, upon receiving the user's information, communicates with the Server to check whether the username

already exists in the Database.

The Server accesses the Database to validate the uniqueness of the username. It returns a Boolean value, indicating the existence or non-existence of the username. If the username is available, the Server saves the user's information into the Database. A confirmation message is returned to the Register Page, informing the user of the successful registration, and the user is redirected to the login page.
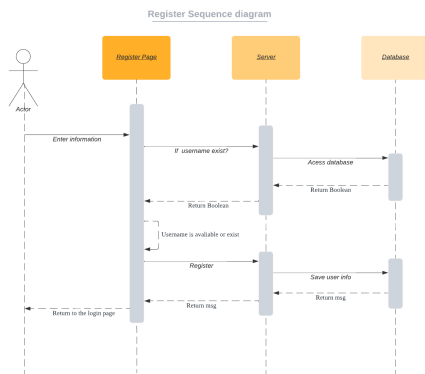
validates the information. After successful registration, the user logs in with their credentials, which are verified by the system for authentication.
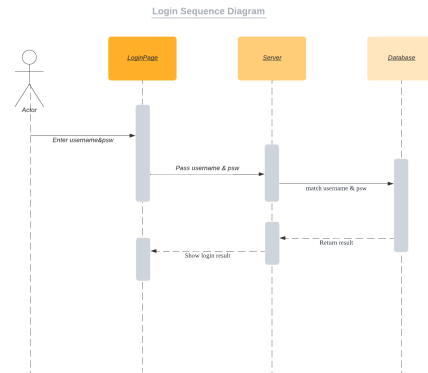

Fig. 9. Sequence diagram of Login

Fig. 8. Sequence diagram of registration

Login Process:

For the login process, the actor enters their username and password on the Login Page.

The Login Page then passes the entered credentials to the Server for validation. The Server cross-references the provided username and password with the stored information in the Database. The Database returns a result, confirming or denying the match of the credentials.

Depending on the result, the Server informs the Login Page of the success or failure of the login attempt, and the corresponding message is displayed to the user. In case of successful login, the user gains access to their account.

This context describes a two-part interaction where the user first registers by entering personal details, and the system

4) Behavior Context:

The state diagrams vividly encapsulate the user journey during the registration and login processes. Let us delve into the behavior context of these tasks:

Registration:

Enter Register Page: The journey begins with the user entering the registration page, marking the inception of the registration process.

Fill in the Information: Users are required to fill in their necessary information, such as usernames and passwords.

Verify the Username Existence: The system then verifies whether the username already exists in the database.

Username Available: If the username does not exist in the system, it proceeds to register the user successfully.

Username Existed: If the username is already taken, the registration process halts, indicating the unavailability of the chosen username. **Login**:

Enter Username and Password: Users start by entering their username and password, initiating the login process.
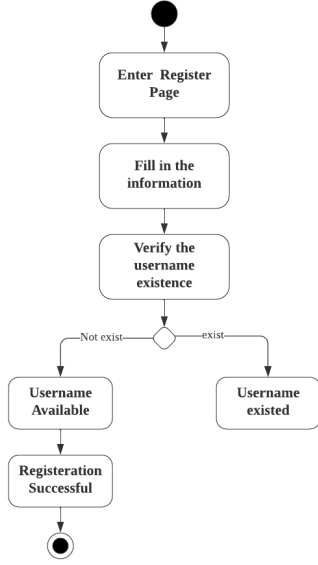
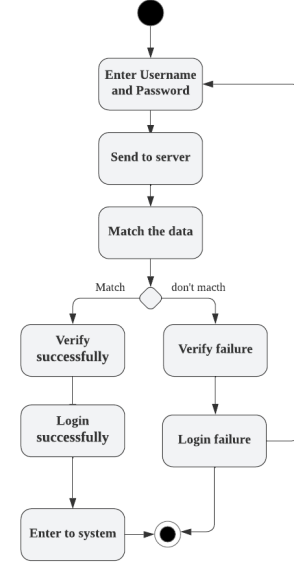Fig. 10. State diagram of registration

Fig. 11. State diagram of Login

Send to Server: The entered credentials are then sent to the server for verification.

Match the Data: The server matches the entered credentials against the stored data in the database.

Verify Successfully: If the credentials match, the user is verified successfully, marking a successful login.

Verify Failure: If there is a mismatch in the credentials, the verification fails, and the login process is interrupted.

In both diagrams, the transition of states is clear and systematically laid out, guiding users through each phase of registration and login, ensuring clarity and precision in navigating the website's authentication processes.

B. Task: Account Edit

1) Task context :

The class diagram in the provided figure illustrates the relationships and interactions among various entities in a user profile management system, specifically focusing on the functionalities of editing personal information and changing passwords.

Central to the diagram is the 'User' class, containing essential personal attributes such as username, name, password,

and contact details. This class is fundamental and acts as a data structure where individual user information is stored and manipulated. The 'User' class has operations like 'editPersonalInfo()' and 'changePassword()' to allow users to update their information and credentials. Collaborating closely with the 'User' class is the 'Profile Manager'. The 'Profile Manager' acts as an intermediary facilitating various user operations. It holds a reference to the current 'User' and encompasses methods such as 'loadUserInfo(),' 'saveUserInfo(),' and 'validatePassword().' These methods are crucial for retrieving, updating, and validating user information against the database.

The 'MongoDB connector' class is instrumental in interfacing with the database. It encapsulates basic database operations such as connect, find, insert, update, and delete, ensuring the application can interact seamlessly with the MongoDB database. This design promotes modularity and separation of concerns. The 'User' class focuses on user data, the 'Profile Manager' manages user interactions, and the 'MongoDB connector' deals with database communications. By having a dedicated class for MongoDB operations, the design ensures that changes in database technologies or schemas have minimal impact on the user management functionalities,

thus enhancing maintainability and flexibility. It facilitates the easy adaptation of the system to meet evolving requirements or technologies, promoting robustness and reusability of the system components in managing user profiles effectively.
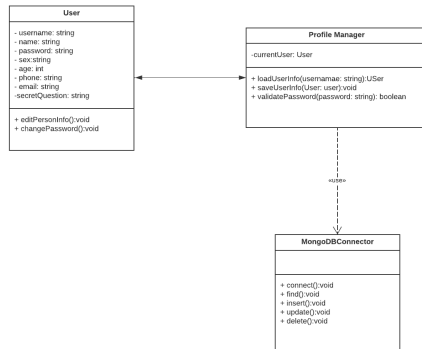

Fig. 12. Class diagram of Account Edit

2) *Structural context :*

In the discussion here, we delve into the data flow processing depicted in the modified activity diagram for user account modification, encompassing both the changing of personal information and passwords.

The user initiates the process by opting to modify the account. They are presented with two paths: one for password modification and another for editing personal information, each comprising a series of systematic steps facilitated by the system's user account management functionalities.

Password Modification Pathway: The user navigates to the 'Change password page,' where they are prompted to enter their username and answer a secret question as a security measure. The system validates the information entered. If it doesn't match the existing records, the user is prompted to re-enter the correct details. Upon successful validation, the user is allowed to enter a new password, which then gets saved as part of the user's account information. Personal

Information Modification Pathway: Users who wish to modify their personal information are directed to the 'Edit Info page.'

Here, users can input new information, like updating addresses or contact details, which the system captures and saves, ensuring the user's account information remains current and accurate.

In this structured context, each component (Modify Account, Change Password Page, and Edit Info Page) plays a pivotal role in orchestrating the flow of information, ensuring that user preferences are accurately captured, validated, and updated as per the user's inputs. The cohesive interaction between these components ensures a secure and user-friendly environment for account modification tasks, safeguarding user data while allowing flexibility for necessary updates and modifications.
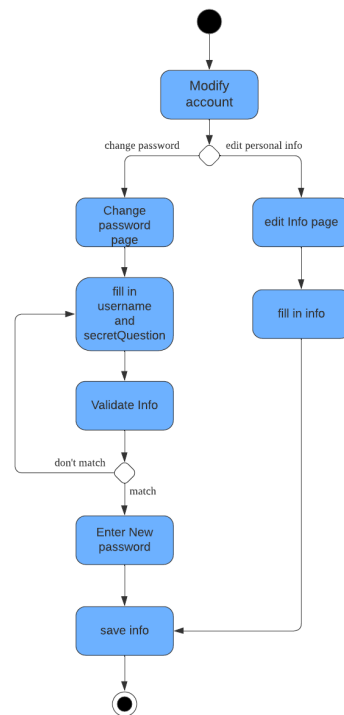

Fig. 13. Activity diagram of Account Edit

3) *Interaction context :*
Password Modification Process:

User Interaction: The actor (user) initiates the password modification by entering their username and new password and answering a secret question on a specific webpage.

Verification and Validation: The webpage, upon capturing the user's details, interacts with the profile manager to verify

the user's identity by matching the username and secret question with the records in the MongoDB database.

Password Updating: If the verification is successful, the profile manager facilitates the updating of the password in the MongoDB database. A confirmation of this update, whether successful or failed, is communicated back to the user through the webpage. **User Information Editing Process**:
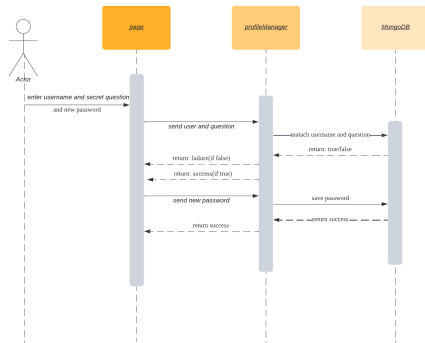
This comprehensive interaction context illustrates a dual-functional process. It involves a password modification procedure followed by a user information editing activity, each with unique steps and interactions among various components like the user, webpage, profile manager, and MongoDB database.
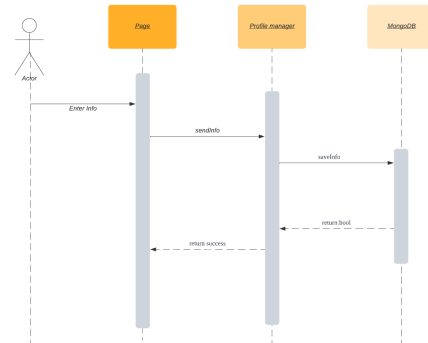

Fig. 14. Squence diagram of Password Modification

Fig. 15. Sequence diagram of User Information Editing

Information Entry: The actor engages in editing their user information by inputting the necessary details on a specific webpage.

Information Transmission: The webpage takes responsibility for forwarding the modified or newly entered information to the profile manager to commence the editing process.

Database Interaction: The profile manager cooperates with MongoDB to secure the updated information, receiving a confirmation about whether the information was successfully saved or not.

Completion Confirmation: The result of the information editing process, either a success or failure, is communicated back to the actor through the webpage, concluding the interaction for editing user information.

4) Behavior context :
Password Modification
Initialization: The process begins when the user intends to modify the password, initializing the system to receive the necessary information.

Enter Username and Secret Question: The user provides their username and answer to a secret question to confirm identity.

Verification: The system then verifies the information. If the information is correct, the process proceeds; otherwise, the user is prompted to re-enter the details.

Password Entry and Saving: Upon successful verification, the user can enter a new password. The system will then save this new password, completing the process.

User Profile Editing
Initialization: The process commences with the system ready to receive updated user information.
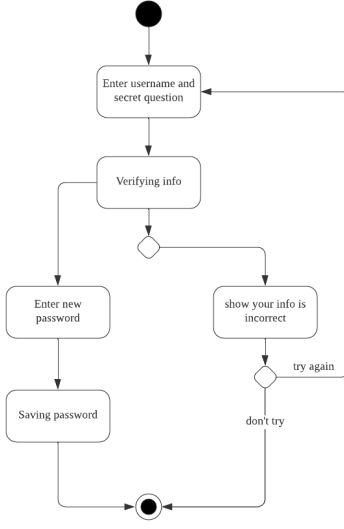
Fig. 16. State diagram of Password Modification

Fig. 17. State diagram of User Profile Editing

Enter UserInfo: Users enter or edit their personal information in the provided fields.

Information Transmission: The edited information is sent to the server for processing and validation.

Database Interaction: After successful validation, the information is saved in the database, and a confirmation is generated.

Completion: Finally, a success message is displayed to the user, indicating that the edits have been successfully made and saved.

Each state transition is systematic and user-friendly, ensuring that the users find the process intuitive and straightforward. Both processes are designed to secure user data and ensure that changes are made correctly and effectively.

C. Task: Searching Data from Database

1) Task context :

The class diagram in Figure 18 shows the relationships among the searcher and database manager.

The DBManager is a base class in the database interface, which encapsulates the Mongo Database interface. The

DBManager can search hotel data and ticket data from the database.

The Searcher class is responsible for catching the keywords that users want to search for. Then call DBManager to find the data in the database based on the keywords combined into a query statement.

Above all, we have designed the context of the diagram display task.

2) Structural context :

In this section, we discuss the data flow processing by Figure 19. Meanwhile, we show the roles these objects play in the data flow processing, which describes the structural context of the Searching Data from Database task.

The user enters the keyword on the search page. Then the service combines the keyword into a query statement. DBManager searches the data from the database based on the query statement. Finally, the Searcher shows the data in the front end.

3) Interaction context :

The sequence diagram in Figure 20 shows the interaction context of this task.
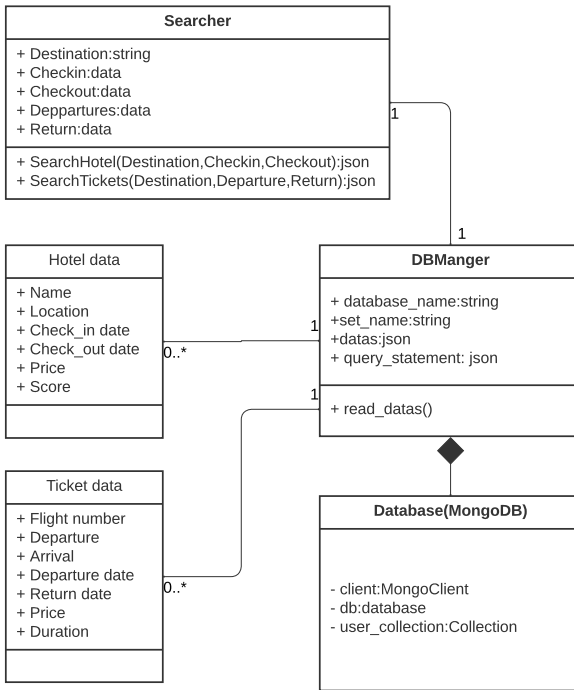
Fig. 18. Class diagram of Searching data from database

The user inputs the keyword and calls the SearchHotel function or Search Ticket function of the Searcher object. The Searcher object calls the read_data function provided by the DBManager object to get the related data from the database.

In the end, the Searcher Object displays the data to the user.

4) Behavior context :
The state diagram in figure 21 shows the behavior context of this task.

The user enters the search web page. After inputting the keywords, the system searches data from the database. Subsequently, the website displays the data as a list.

D. Task: Crawler searching

1) Task context :
The class diagram in Figure 22 shows the relationships among the searcher, crawler and database manager.

The DBManager is a base class in the database interface, which encapsulates the Mongo Database interface. The DBManager can save hotel data and ticket data into the database.

The Searcher class is responsible for catching the keywords that users want to search for. When the searcher receives the keywords, it will call the Crawler to search information from the internet.

The Crawler class is responsible for catching data from the internet. The Crawler can be generalized as the HotelCollector and TicketsCollector. They have extended with some class functions related to searching hotel or ticket data from the internet. When they get hotel and ticket data, they will call the DBManager to save the data into the database.
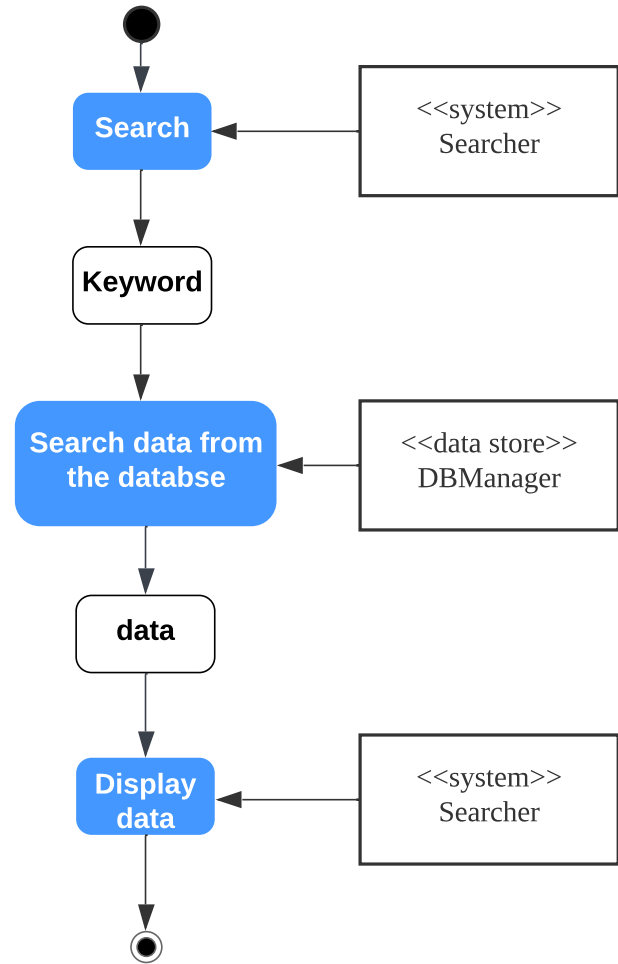

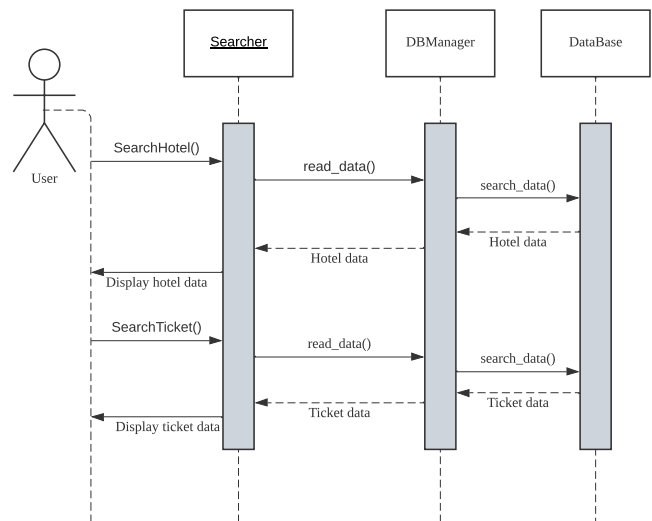Fig. 19. Activity diagram of Searching data from database

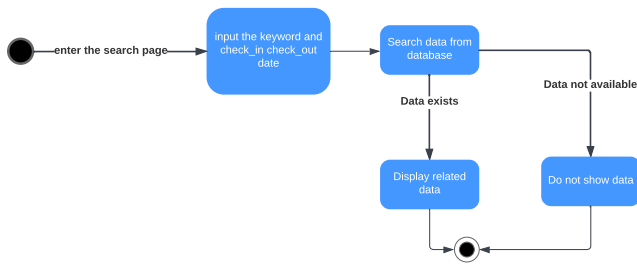Fig. 20. Sequence diagram of Searching data from database

Fig. 21. Statement diagram of Searching data from database

2) Structural context :

In this section, we discuss the data flow processing by Figure 23. Meanwhile, we show the roles these objects play in the data flow processing, which describes the structural context of the Crawler searching task.

The user enters the keyword on the search page. Then the service calls the Crawler to search data from the internet. Finally, the Searcher shows the data in the front end, meanwhile, the DBManager saves the data into the database.

3) Interaction context :

The sequence diagram in Figure 24 shows the interaction context of this task.

The user inputs the keyword and calls the SearchHotel function or Search Ticket function of the Searcher object. The Searcher object calls the get_hotels pr get_tickets function provided by the Crawler object to get the related data from the Internet.

The crawler first gets the searchid from the website, and then uses the searchid to get the required data. When the Crawler receives the data, it will call the DBManager to insert it into the database and return the data to the Searcher Object.

In the end, the Searcher Object displays the data to the user.

4) Behavior context :

The state diagram in figure 25 shows the behavior context of this task.

The user enters the search web page. After inputting the keywords, the system will call the crawler to search data from the internet. Subsequently, the website displays the data as a list and saves the data into the database.

E. Task: The Analysis Diagram

1) Task context :

The class diagram in Figure 26 shows the relationships among the analyzer, questionnaire, displayer, and database manager.

For high efficiency and reusability, we designed a mediate layer for accessing the database. The MongoDBManager is a base class in the database interface, which encapsulates the Mongo Database interface. The Mongo DB Manager can be generalized as the Cities Information Collection and Preference Information Collection. The Cities Information

Collection has extended with some class functions related to the city information. Meanwhile, the Preference Information Collection has extended with some class functions related to user preference information.

The Questionnaire class is responsible for catching user's preference information. A global Preference Infor Collection class instance provides the database interface for the data processing function in the Questionnaire class. Hence, the Preference Information Collection is associated with the Questionnaire.

On the other side of the class diagram, the diagram displayer performs the issues about analysis data displaying, which is based on the data analyzed by the KNNDataAnalyzer. The KNN Data Analyzer compares each city data entry with the user's preference and decides which city is the most appropriate recommendation. In this process, the KNN Data Analyzer calls the database interfaces provided by the Cities Information Collection and Preference Information Collection.

Above all, we have designed the context of the diagram display task.

2) Structural context and Task Data Flow:

In this section, we discuss the data flow processing by Figure 27. Meanwhile, we show the roles these objects play in the data flow processing, which describes the structural context of the Analysis Diagram task.

The user fills out the questionnaire on the questionnaire web page with the help of the questionnaire system. The Preference Information Collection saves the user's preference information for further analysis. Subsequently, the user enters the analysis page, which calls the Get Data Functions of Preference Information Collection and Cities Information Collection simultaneously. The preference data and city data are sent to the Data Analyzer for data analysis. Finally, the Diagram Displayer shows the data diagram based on analyzed data.

3) Interaction context :

The sequence diagram in Figure 28 shows the interaction context of this task.

The user fills out the questionnaire and calls the PostQuestionnaire function of the Questionnaire object. The Questionnaire object calls the StorePreferenceData function provided by the Preference Infor Collection object to save the preference data into the database.

Then the user clicks the 'recommend' button on the home page and calls the EnterAnalysisPage function, which is provided by the Diagram Displayer object. The Diagram Displayer object makes a request to the Data Analyzer for the recommendation information. The Data Analyzer gets the user preference data from the Preference Infor Collection object according to the UID. Meanwhile, all city data is gained from the Cities Infor Collection object. The Data Analyzer processes a recommendation algorithm based on these data and feeds back the recommended city information and popular city information. The Diagram Displayer shows the data diagram and returns an OK message to the user.
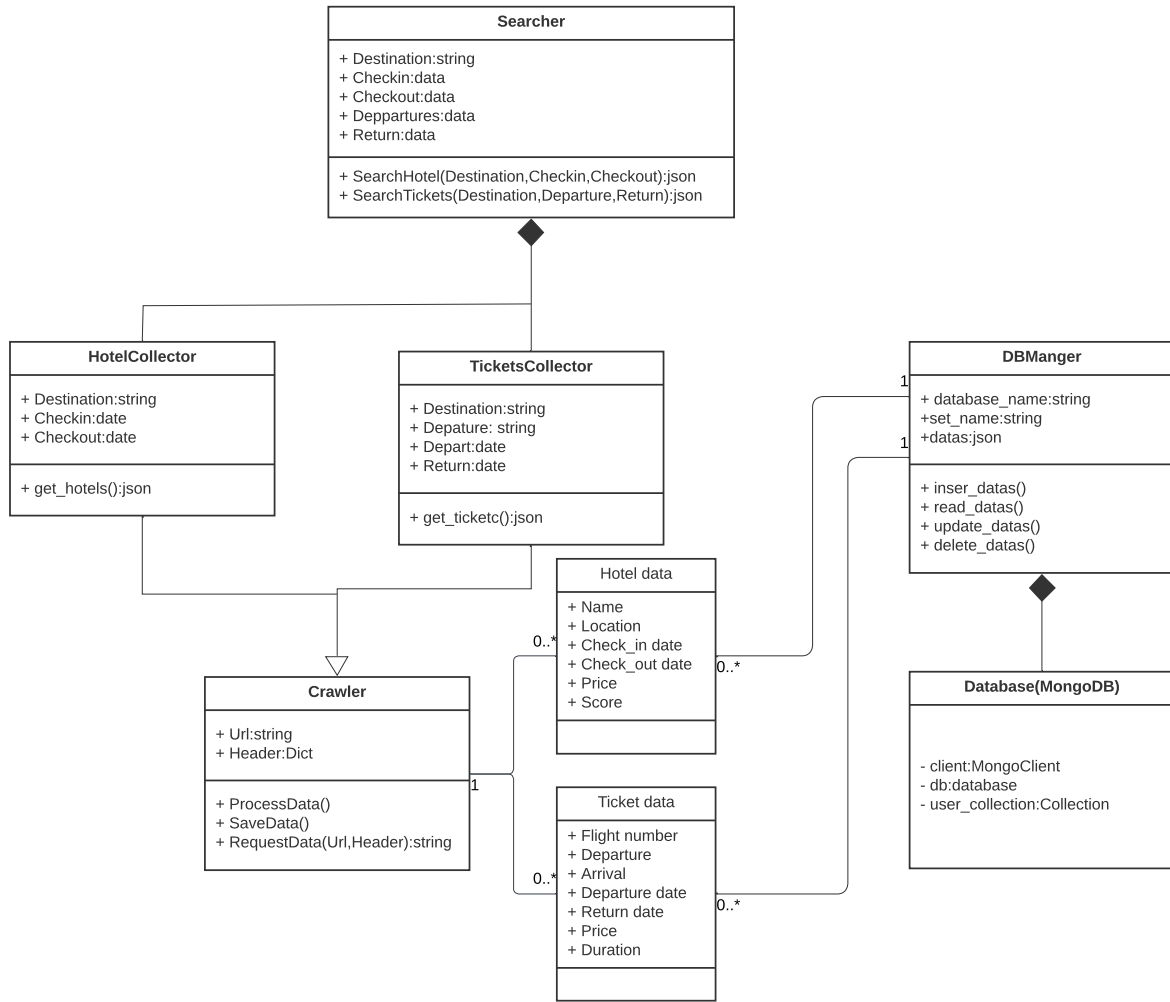
Fig. 22. Class diagram of Crawler searching

4) Behavior context :

The state diagram in figure 29 shows the behavior context of this task. The state transition is simple and most functions in this task process data invisible to users. We divide the state according to the function of the web page. In this section, we only care about the states related to the analysis diagram task.

The user enters the questionnaire web page, the system enters into the Questionnaire State. The Questionnaire State displays the multiple psychological questions for users. After finishing the questionnaire, the system saves the preference data into the database. Subsequently, the website comes back to the home page and the system enters the home state. When the user clicks the 'recommend' button, the website jumps to the Analysis Diagram Page. After the system completes the recommendation, the Analysis Diagram Page displays the data diagram.

IV. UNCOMPLETED TASK DESIGN

1) Task Context : The class diagram illustrates the intricate relationships among various components, such as the desti-

nation data, recommendation system, and database manager (DBManger), focused on facilitating travel recommendations. For optimized performance and enhanced reusability, a meticulously designed interface layer has been integrated to manage database access, epitomized by the DBManger, encapsulating the database interfaces, specifically MongoDB.

DBManger is a pivotal class, coordinating database interactions and ensuring smooth data transitions. It possesses essential attributes such as database name and set name and operational functionalities like reading data, represented by read datas(). It forms the backbone supporting the data-driven operations of the system. Destination data is a comprehensive component, embracing a wealth of information such as City, Travel spot, Transportation ticket, Hotel, Restaurant, and Comment. This aggregated information is instrumental in forming a robust foundation to enhance the recommendation accuracy and user experience. In tandem with the destination data, the Recommendation class emerges as a quintessential component, orchestrating the recommendation processes with functions
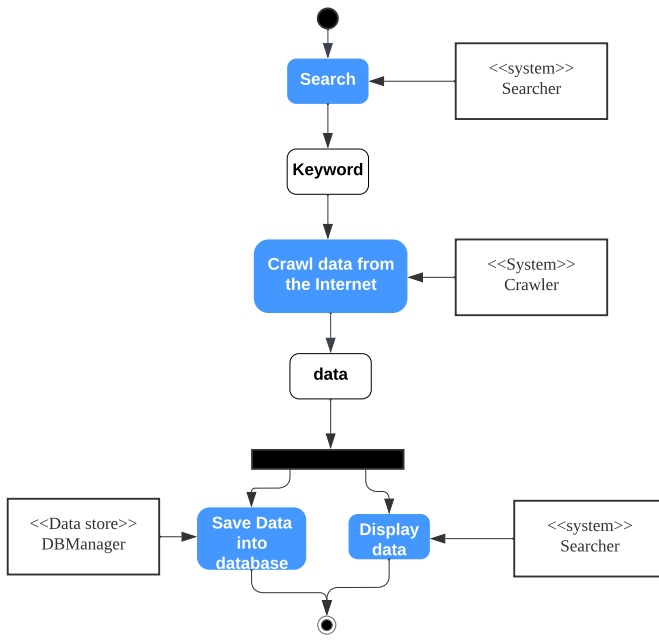
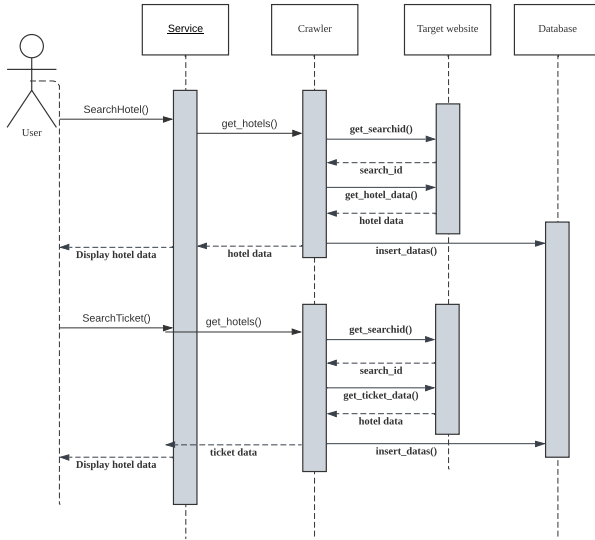Fig. 23. Activity diagram of Crawler searching

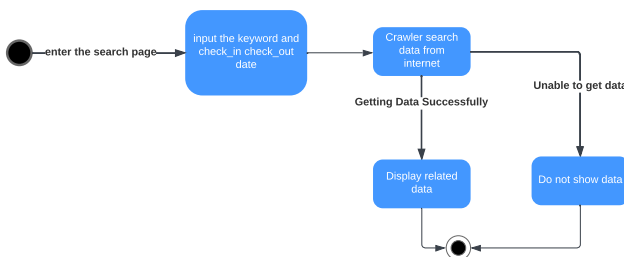Fig. 24. Sequence diagram of Crawler searching

Fig. 25. Statement diagram of Crawler searching

like RecommendSpot(), RecommendTicket(), RecommendHotel(), and RecommendRestaurant(). The class is enriched with attributes such as Keywords and Date, which play a crucial role in customizing and optimizing the recommendations, making them more aligned with user preferences and current trends. Ensuring a seamless and dynamic interaction between these components, the Database (MongoDB) manifests as an essential entity, facilitating robust and efficient data storage and retrieval processes. It hosts vital elements such as client, DB, and user collection, underpinning the database operations and ensuring the system's integrity and functional efficacy.

In conclusion, the designed context of the class diagram illustrates a well-coordinated travel recommendation system, emphasizing modularity, efficient database management, and a user-centric recommendation approach. Through strategic class interactions and functionalities, the system is poised to deliver tailored travel recommendations, enhancing user experience and satisfaction.

2) Structural context : In the data flow processing depicted in the provided activity diagram, we delineate the functional engagement and sequence of operational activities to clarify the structural context of the task.

Initially, users are introduced to the environment by entering the website, marking the commencement of their interaction. Following this entry point, users progress to execute a 'Search destination,' a function integral for navigating through the data or information within the system. Post navigation, there is an execution of 'search data.' This pivotal action is bifurcated into two possible outcomes: the system either finds or doesn't find the related data. A decision diamond illustrates this, symbolizing a decision point where the flow is directed based on the availability of the searched data. In the scenario where data is not found (no finding data), the flow advances towards 'Display related data,' an endpoint that signifies the conclusion of the search with no fruitful results. Conversely, when the searched data is located (find data), the flow navigates towards 'Recommend data.' This section is crucial as it is where data is curated and processed for user-friendly presentation. Finally, 'Display data' is the terminal function, culminating the process by showcasing the refined and recommended data. This presents the users with the output of their initial search and the system's subsequent recommendations, thus completing the structured flow of tasks within this activity diagram.

In conclusion, the activity diagram exemplifies a meticulous order of operations, each functioning collaboratively to streamline the users' experience from entry to the execution of search and display of the correlated data or recommendations.

3) Interaction context : The sequence diagram illustrates the workflow of a recommendation system. An actor, in this case, the user, interacts with the system to get personalized recommendations based on their preferences.

Initially, the actor initiates a search, interacting with the "Recommendation" component of the system. The actor inputs their search destination, and the system forwards this information to the "Search" component for processing. The "Search"
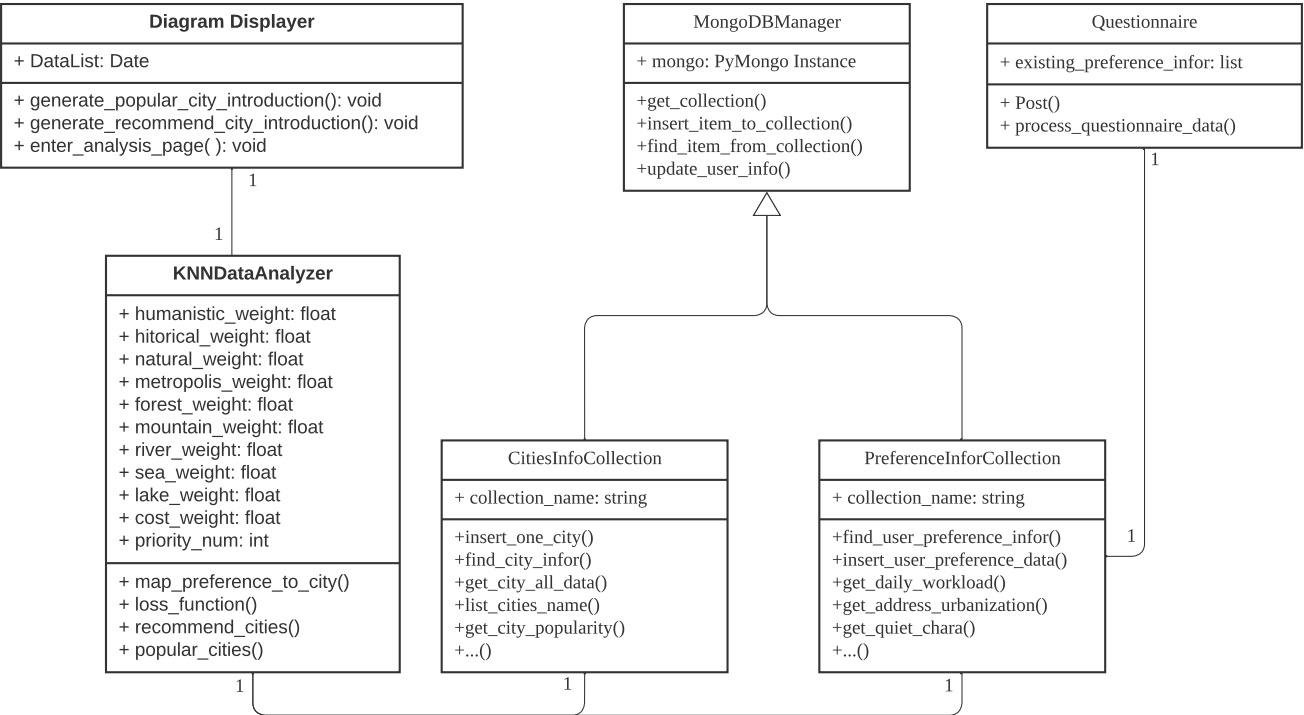
Class Diagram

Diagram Displayer

+ DataList: Date

+ generate_popular_city_introduction(): void
+ generate_recommend_city_introduction(): void
+ enter_analysis_page( ): void

MongoDBManager

+ mongo: PyMongo Instance

+get_collection()
+insert_item_to_collection()
+find_item_from_collection()
+update_user_info()

Questionnaire

+ existing_preference_infor: list

+ Post()
+ process_questionnaire_data()

KNNDataAnalyzer

+ humanistic_weight: float
+ hitorical_weight: float
+ natural_weight: float
+ metropolis_weight: float
+ forest_weight: float
+ mountain_weight: float
+ river_weight: float
+ sea_weight: float
+ lake_weight: float
+ cost_weight: float
+ priority_num: int

+ map_preference_to_city()
+ loss_function()
+ recommend_cities()
+ popular_cities()

CitiesInfoCollection

+ collection_name: string

+insert_one_city()
+find_city_infor()
+get_city_all_data()
+list_cities_name()
+get_city_popularity()
+...()

PreferenceInforCollection

+ collection_name: string

+find_user_preference_infor()
+insert_user_preference_data()
+get_daily_workload()
+get_address_urbanization()
+get_quiet_chara()
+...()

Fig. 26. The class diagram of analysis diagram task.

Structural Conext & Data Flow Activity Diagram

<<data store>> Preference Infor Collection

Fill the Questionnaire → preference data → Preference Data Store → Enter Analysis Page

<<system>> Questionnaire

<<system>> Diagram Displayer

<<data store>> Preference Infor Collection

Get Preference Data

Get Cities Infor Data

<<data store>> Cities Infor Collection

preference data

cities data

<<system>> Data Analayzer

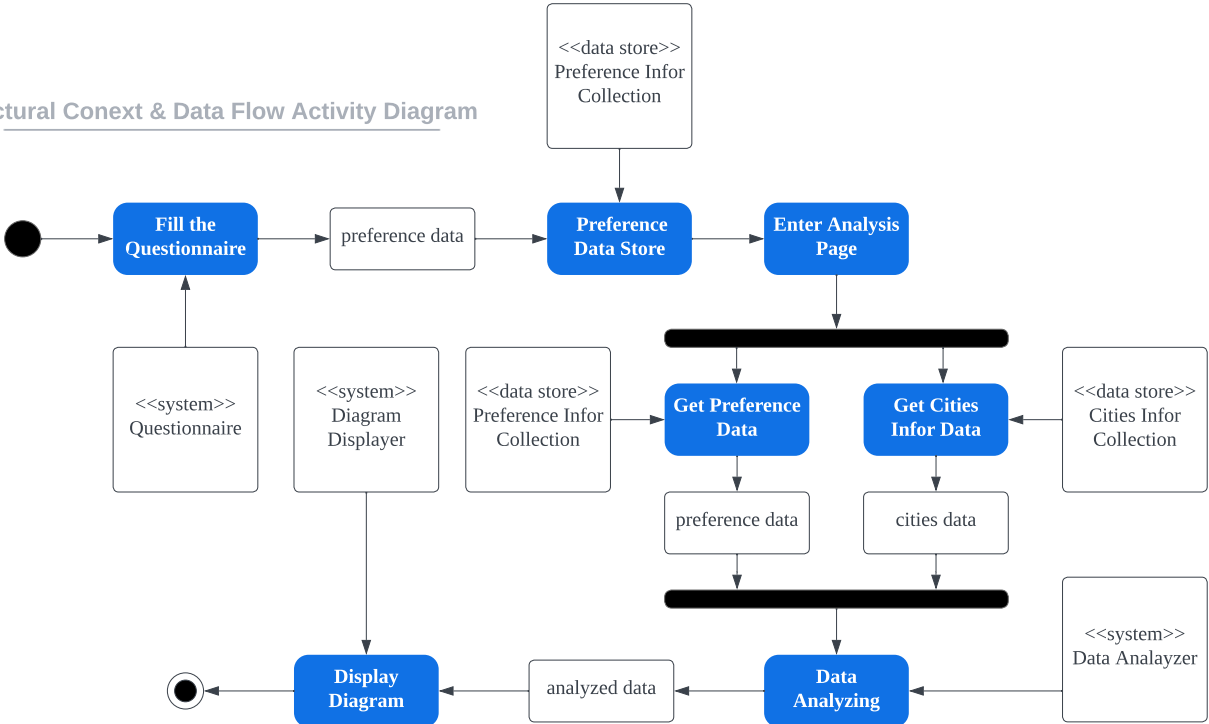Display Diagram ← analyzed data ← Data Analyzing

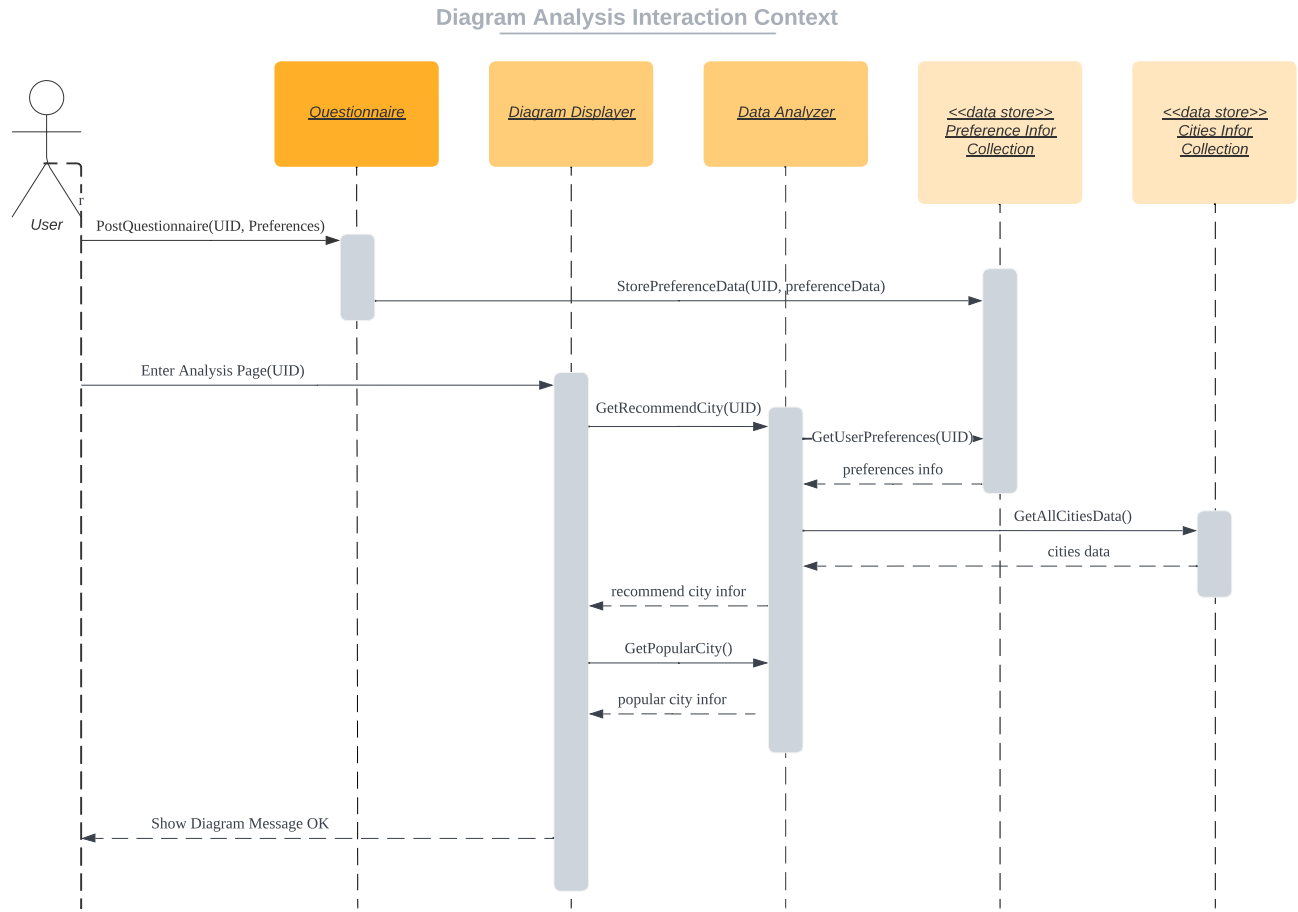Fig. 27. The structural context and data flow of analysis diagram task.

Fig. 28. The interaction context of analysis diagram task.

component interacts with the "Database," fetching relevant data to cater to the actor's search request. The "Search" component handles the data, processing and filtering it based on specific algorithms or criteria, preparing it for recommendation. Once the data is processed and the most relevant information is selected, the "Search" component sends the recommended data back to the "Recommendation" component. The "Recommendation" component then displays this data to the actor, completing the recommendation process. Thus, the actor receives personalized recommendations, enhancing their user experience and providing the most relevant and valuable information based on their initial search destination.

4) Behavior context : The state diagram depicted represents the behavioral context of a data search and recommendation task on a website. The flow is streamlined and primarily focused on backend data processing that is not visible to the users. States are divided based on the functional aspects of the webpage.

Initially, users enter the website, marking the commencement of the interaction. Following this, users input specific keywords that act as queries for the system to search relevant data. In the "Input keywords" state, users actively engage with

the design, providing necessary information to facilitate data retrieval. Upon receiving the keywords, the system transitions into the "search data" state. Here, a systematic search is conducted in the background, striving to find the most relevant data corresponding to the input keywords. Depending on the availability of relevant data, two outcomes are possible. If pertinent data is found, it transitions to the "Display recommend data" state, where users are presented with specific data that the system believes to be the most relevant to the keywords inputted. Conversely, if the system fails to find relevant data, it transitions to the "Display related data" state. In this state, instead of showing recommended data, it provides data that might be closely related or similar to the queried keywords. Each state is crucial for ensuring that the user receives the best possible recommendations or related data based on their input, enhancing the overall user experience on the website. The simplicity of the state transitions also ensures that the process is user-friendly and easily navigable, promoting a seamless interaction between the user and the system.

V. PROGRESS EVALUATION & ANALYSIS

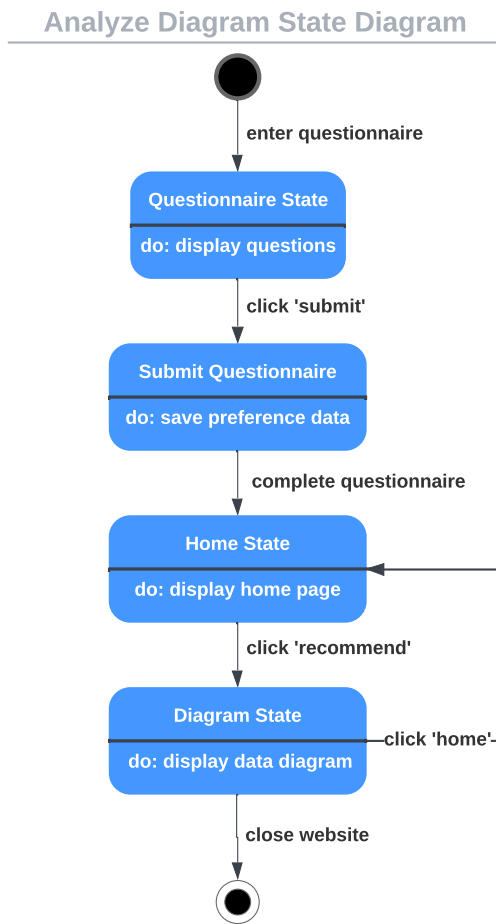As shown in Figure 1, the man/hour about each task is clear.

Fig. 29. The state diagram of analysis diagram task.

We decided to develop the Search Destination requirement in Sprint Cycle 2. However, we face the difficulty of how to use the analyzed result to influence the search result. Hence, we modify this task to the Sprint Cycle 3. Meanwhile, we have a plan to optimize our system user interface. These works will be done in the next stage.
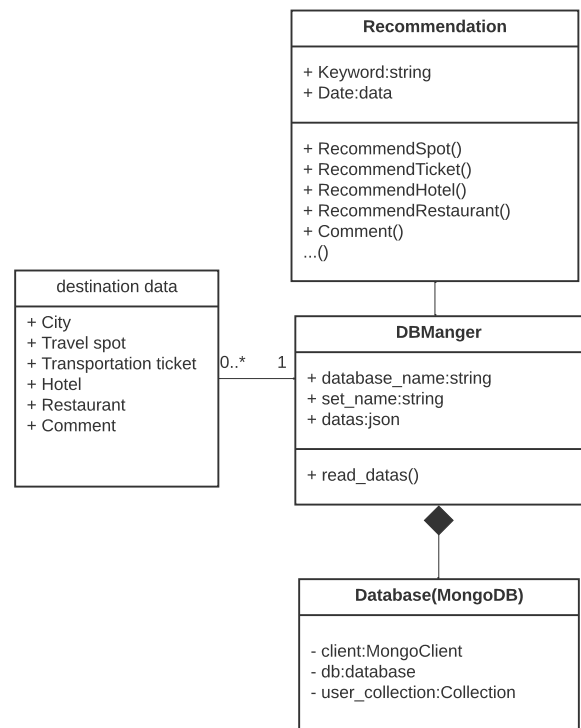

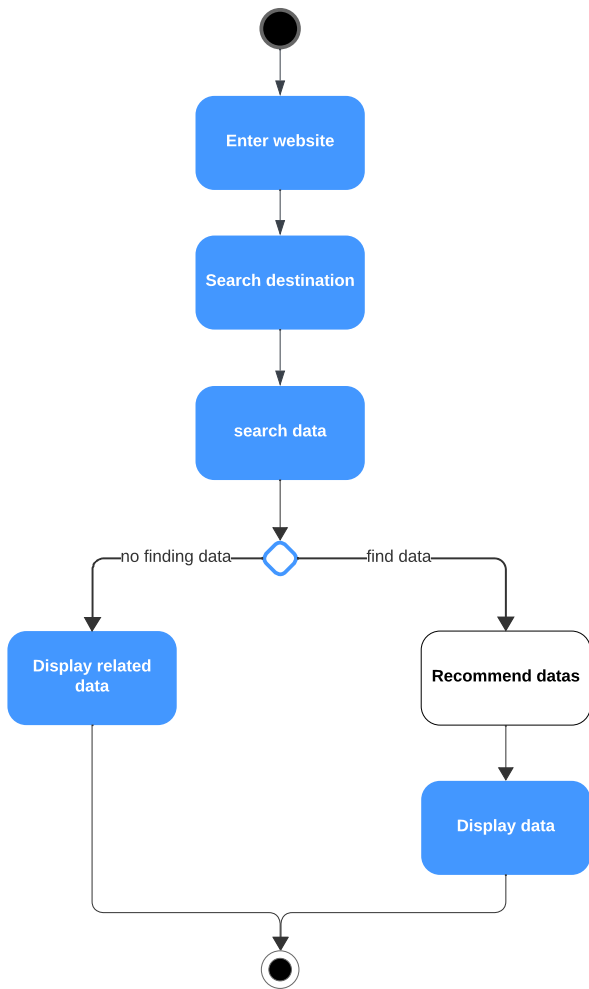Fig. 30. The class diagram of uncompleted function.

Fig. 31. The activity diagram of uncompleted function.
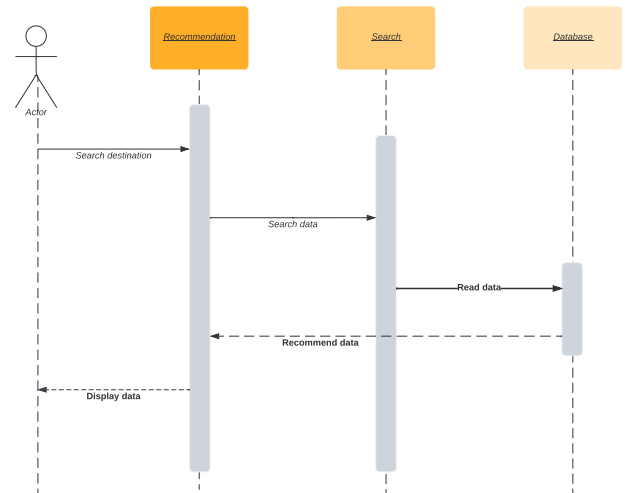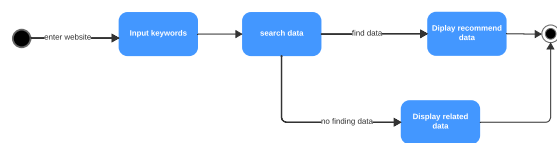

Fig. 32. The sequence diagram of uncompleted function.

Fig. 33. The state diagram of uncompleted function.