System Document

Version 2.3.1

Abstract

The purpose of this document is to provide a comprehensive and detailed information of Tripster, a flask web application developed for a local travel agent. This document provide analysis on requirements; system design including software architecture design, databse design, UI/UX design and UML diagrams; explainations on technical implementations on environment, source code, APIs, development tools and algorithms; depolyment and configurations; security measures; testing, maintainance and help guide. The document as well introduces the plan and process during the development stage of Tripster, including project management approach, work division and team-work. In the final section, the users can find our outlook on future work on upgrades, troubleshooting and new features.

March 18, 2025

Contents

1 Introduction

1.1 Background

The COVID-19 pandemic has had a significant impact on businesses worldwide, including the travel industry. Our client, a family-owned local Travel Agent has also been affected by the pandemic-induced shift toward online sales due to the need for social distancing. Thus, the agent has decided to adopt a digital online platform in response to the situation. This is where our software engineering team comes in. We designed a user-friendly, secure and robust website, that enables the local Travel Agent to provide their customers with a simple and intuitive interface that guides users to browse, plan and book for their trips, and the staffs with a friendly interface to manage operations in the company.

1.2 Intended Audience and Reading Suggestions

This system documentation is intended for

- **Website/software developers** who are building their own applications or integration based on the structure, design, service or technology provided in our website.
- **Graphic/UI/UX Designers** who are to modify the designs of the website for their use of interests.
- **Quality Assurance Testers** who are testing the functionality and usability of our website to ensure the highest possible quality.
- **System Administrators** who hold the responsibility for website management and maintainace.
- **End-users** who uses our website and are interested in the technical details and implementations of the service our product provide.
- **Product managers** who will be making business decisions and analyses based on our website product.

Since it is a system documentation, prior knowledge regarding html based front-end development and Flask back-end development is required to fully understand all the contents.

1.3 Requirement Analysis

1.3.1 Functional requirements

The website requires two portals to fulfill the functional requirements for customers and staffs of the website.

- **Customer requirements**
  The **customer portal** should allow the functionalities illustrated below:
  - Browse the travel information, including destination, accommodations, attractions
  - Search for travel information, including destination, accommodations, and attractions based on keywords.
  - Browse and add reviews to the destinations, attractions, and accommodations
  - Instant chatting with staff
  - Display system notifications

  - Add/delete attractions and accommodations into a plan cart for further use in the planning tool
  - A planning tool to create/modify a tour plan from the plan cart.
  - A booking tool to book for transportation, attractions and accommodation within a created plan.

- **Staff requirements**
  The **staff portal** should allow the functionalities illustrated below:
  - Browse travel information, including destination, accommodations, attractions
  - Add destinations. Browse destinations, attractions, and accommodations details in admin mode.
  - Supervise(including search), organize, modify, prioritize, update and keep track of reservations.
  - Instant communication with customer
  - Browse visualized statistics of the website

- **Website/System Requirements**:
  - Present interactive and user-friendly interface
  - Provide detailed user and system documentations
  - Provide responsive layout and mobile adaptance.
  - Support language switch between Chinese and English
  - Provide content(travel information) recommendations utilizing machine learning techniques
  - Provide an accurate and fast dispatch mechanism for staffs to communicate with customers

1.3.2 Non-functional requirements

- **Performance**: The website should ensure stable operation under high concurrency conditions. It should provide fast response times with a maximum response time of 1.5 seconds per request. It should be able to process 1200 transactions per second with a maximum latency of 90 milliseconds with a data accuracy rate of 99.9 percent and a data storage capacity of 1 terabyte.

- **Security**: The website should prevent information leakage and utilize encryption and secure authentication mechanisms to ensure data privacy while preventing unauthorized access. It should comply with relevant data privacy regulations in China, such as the Cybersecurity Law and the Personal Information Protection Law.

- **Reliability**: The website should ensure normal operation under various abnormal conditions with unexpected errors or exceptions like network failure and system downtime. It should also handle high traffic volumes and provide reliable service with minimal downtime and possesses a backup and automatic recovery mechanism in case of system failure. The system should be able to recover from errors or crashes.

- **Availability**: The website should be *deployed* on a cloud server, being accessible anywhere on earth via a web browser. The website should have a backup and automatic recovery mechanism in case of system failure. The system should be able to work on low-bandwidth connections and slow internet speeds, with a maximum load time of 4.2 seconds per page.

- **Compatibility**: The website should provide consistent user experience across different devices (e.g. mobile phones, iPad) and browser platforms (e.g. Edge, Google browser).
- **Usability**: The website should have a user-friendly and intuitive UI/UX with clear interaction logic. Navigation of the website should be straightforward, with clear labels for menus and links. The layout and design should be organized and visually appealing, maintaining consistency within color, theme, and typography. Forms and input fields should be easy to understand, with clear instructions and placeholders. The website should be accessible to all users, including those with disabilities. The website should also provide a bilingual interface for users from different backgrounds.

2 System Design

2.1 Software Architecture Design

2.1.1 Overall Architecture

Our website is based on the Model View Template (MVT) software design pattern.

- The **Model** is a data layer that is responsible for domain logic and persistence, in other words, the required fields and behaviors of the data. It also allows data-related mechanisms like adding, updating, retrieving, deleting data in the database, as well as possessing data manipulation techniques such as custom methods.
- The **View** is a controller function that performs application/business logic(defines application APIs) by interacting with the **model** and executing requests to external services and eventually renders a **template**. In View, HTTP requests are handled, such as accepting/processing HTTP requests and returning the HTTP response.
- The **Template** is a presentation layer that contains HTML text files(static/dynamic) that renders data, handling the User Interface aspect.

Advantages of MVT:

- **Separation of responsibility**: MVT separates the web application into three components regarding data management, presentation and application logic, through which it reduces coupling within the application framework and presents better scalablity and flexibility. The separation also promotes better code organization and reduces the complexity of the codebase and makes it easier to maintain, update, test and debug, since each functional unit is restricted to a specific area of the codebase. In this case the efficiency in development is also improved.
- **Scalability/Extensibility**: MVT pattern allows high scalability through separtion of responsibilities. The addition of new features(e.g. modifyng templates, adding new models or routes) in one layer will not impact other layers nor the quality and reliability of the overall application system.
- **Reusability**: MVT framework eliminates the need for code replication while creating consistent and stable user interface and adding new features through providing reusable templates for web pages. It also allows ease on applying global modifications to the User Interface.
- **Efficiency in development**: Separation of responsibility and resuability boosts the speed of the development process of the website. It also promotes work efficiency of the developers by providing pre-built components such as login forms and existing templates.

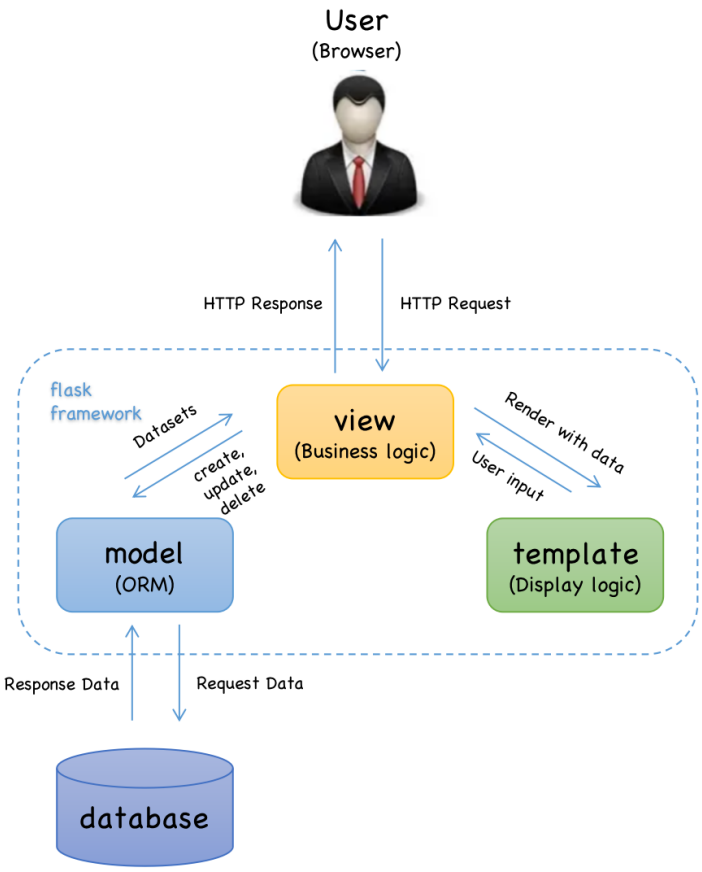


Figure 1: MVT Architecture

2.1.2 Architectural Technical Specifications and reasons

The application framework of our web app *Tripster* is illustrated in Figure2. **Flask** is a micro web framework that allows us to create and run web applications with Python. It is a set of tools and libraries that help us handle common tasks such as routing, templating, security, and database access. Furthermore, Flask features well-documented and fully-supported. Whenever a programming problem occurs, the solution can always be found in the documentation.

Flask is based on two subsystems, Werkzeug WSGI toolkit,and Jinja2 template engine.

- **WSGI**: Web Server Gateway Interface (WSGI) is a standard for Python web app development that stipulates the specifications regarding the universal communication interface between a web server and a web/client application. **Werkzeug** is a WSGI toolkit that manages functionalites such as requests and response of objects, providing a foundation to build a web framework upon it.
  Some advantages of this choice includes:
  - Flexibility: The flexibility lies in the freedom in modification of web stack components, allowing users to separately choose a application/framework and a web server.
  - Scalability and Performance: Compared to frameworks, WSGI servers can serve thousands of requests concurrently and best route them within the process of application framework, suggesting higher performance. Such separation in functional domain not only sclaes the web traffic but also lowers the possibility of DDoS attack.
- **Jinja2**: Jinja2 is a template engine for Python, which enables dynamic generation/render of different template-based document. It transforms python code to HTML files and handles the interaction between them. advantages of Jinja2 are as follows:

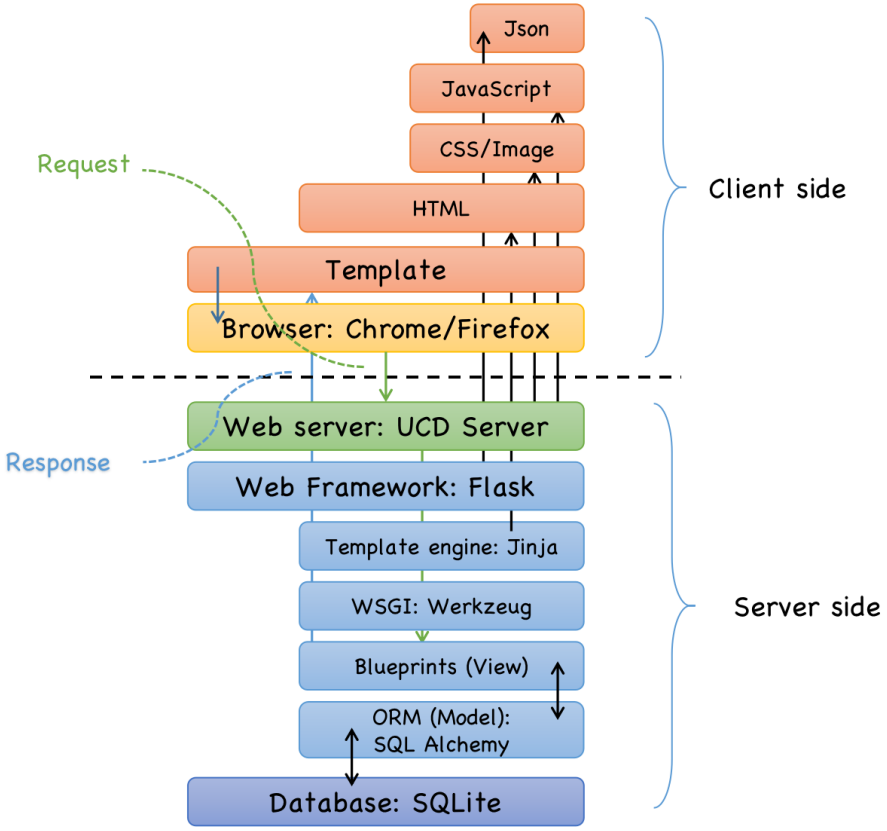


Figure 2: General layers and interactions of Tripster

- Security: Jinja2 allows automatic HTML Escaping that protects the website against Cross-site Scripting(XSS Attack).
- Efficiency: Better performance mainly lies in faster rendering algorithms and techniques. Jinja2 supports asynchronous rendering of templates. It also utilizes pre-processing tools to optimize the raw html and css in advance to rendering process. Caching is also applied, through which compiled templates are chached and could be only parsed once on subsequent requests, reducing parsing and compiling overhead.

Other technical implementations in layers:

- **SQLAlchemy**: SQLAlchemy is a Python library that provides an object-relational mapper (ORM) and a SQL toolkit. An ORM is a tool that allows us to interact with a database using Python objects instead of writing raw SQL queries. A SQL toolkit is a tool that helps us construct and execute SQL statements programmatically.
- **Blueprints**: The flexibility and modularity of flask framework allow us to use blueprints, where the route methods are categorized based on the work package it belongs to. The routes acts as the View in the MVT structure.
- **SQLite**: Among the multiple database engines that SQLAlcemy supports, such as SQLite, PostgreSQL, MySQL, and Oracle, we choose SQLite to be the database of the website. SQLite is a lightweight database with high performance in reading and writing operations. It allows easy extension in future modifications in application and also preserves backwards compatibility.

2.2 UML Diagrams

Domain Model:

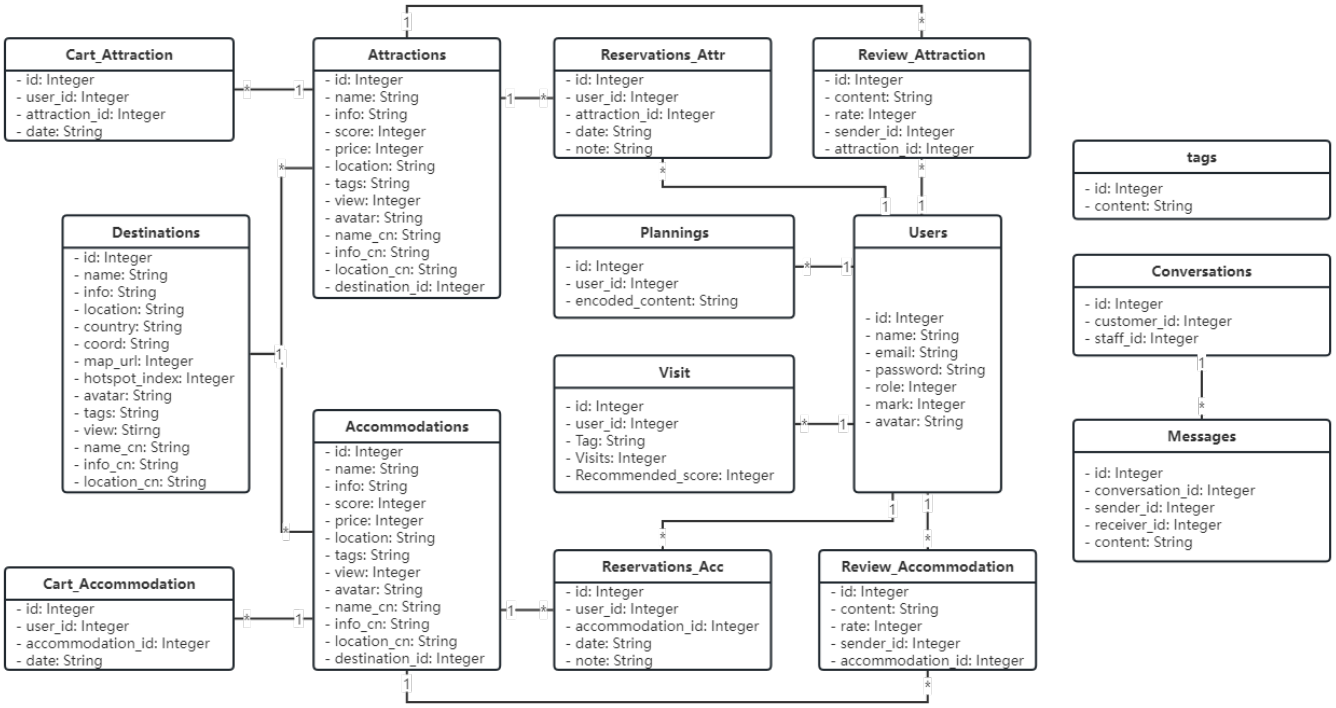


Figure 3: Domain Model

2.3 Database Design

ER diagram:

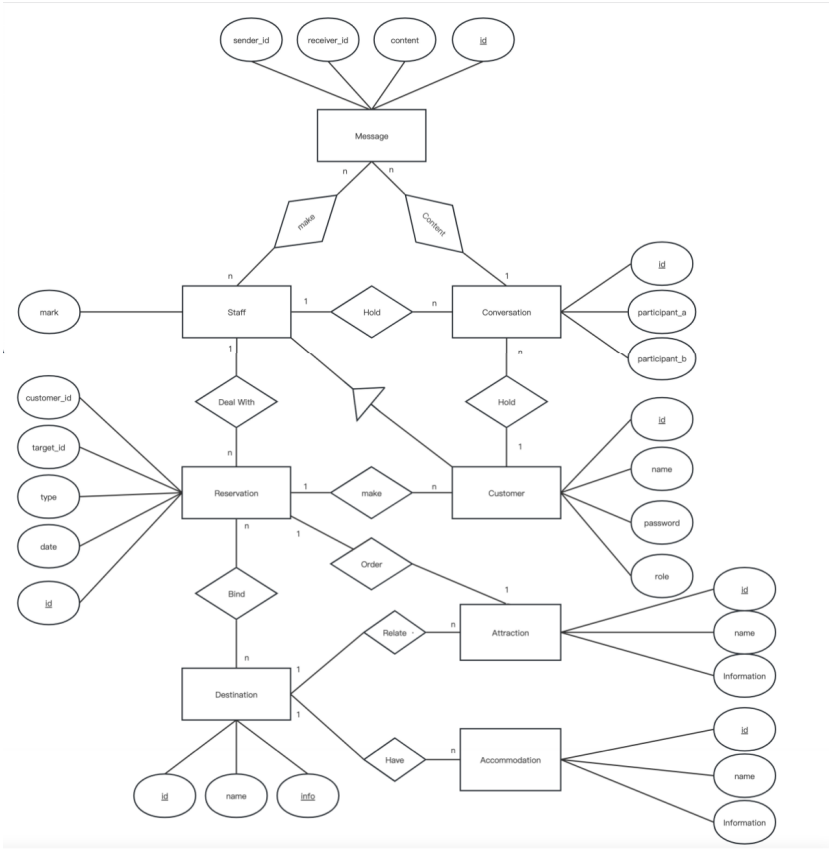


Figure 4: ER Diagram

2.4 UX/UI Design

2.4.1 User Experience

Considering that more and more people are using mobile devices to browse travel websites, we have placed a lot of emphasis on responsive design, adapting the website to different screen sizes, and ensuring a good browsing experience on mobile devices. Moreover, VR experience for certain attractions is also provided to enhance user experience. Users can interact with the VR scene by dragging and immersing in the 3D video using VR equipment.

Figure 5: Virtual Guide Shizuku

In addition, we added **Shizuku AI - Virtual Guidance** to the site, using AI to help users interact better with the site and meet their more diverse needs. The eyes of the virtual guide Shizuku will follow the movements of the user's mouse. Considering the problems that people with **disabilities** encounter in using the site, we also provide voice interaction in addition to keyboard input and mouse click sections. When a user asks a question to Shizuku, the website uses voice output to answer it. At the same time, we designed the character form for AI to enhance the site interaction fun.

We also offer **drag-and-drop planning panels**. It provides optional places on the left and the user's planning panel on the right. After the user selects a destination, the system automatically displays the scenic spots owned by the destination. In this case, the user can plan freely and ensure the order of the plan panel. This is a new way of interaction on travel websites, and we hope to enhance the user experience in this way.

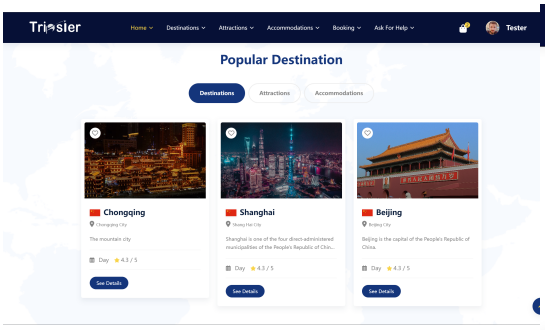


Figure 6: Unified interface design style

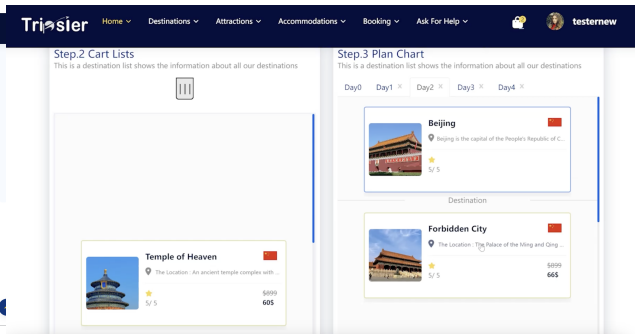


Figure 7: Draggable Plan Panel

We have set up many aids on the website. With the help of Google Maps, we have inserted different interactive maps(drag and zoom in/out with scroll) to give users a better understanding of the areas they are interested in and to help them plan their travel routes. In the case of 3D Earth, travel destinations are marked on the Earth and connected with blinking 3d lines. Figures 91014 contains details of interactive maps.

2.4.2 Interface Design

Our website focused on overall style unity. Considering the tourism-related attributes of the website, we used large matching images and a blue and white color scheme to highlight the key points related to tourism information. In terms of visual presentation, we borrowed from Google's design concept and used a lot of geometric shapes to simplify the site. To improve the visual quality of the site, we used straight line shadows for each component and eliminated gradient or curve shadows. We also used high-resolution images to promote an immersive user experience.

3 Technical implementation

3.1 Environment Specifications

- **Operating Environment**:
  A laptop, desktop computer, tablet, or smartphone with a compatible web browser like google chrome, Mozilla firefox, apple safari, and Microsoft edge is required. No extra plugins are required for our website. A stable internet connection and fully functioning I/O interaction devices like a mouse or keyboard are required.

- **Development Environment**:
  Along with the operating environment, the computer should contain at least 4 GB ram and a dual-core processor. In terms of software, python should be installed, a virtual environment should be created based on requirements.txt and an IDE like Pycharm or a text editor is also a must.

3.2 Front-end

3.2.1 Functional Division(Source code)

- **Visitor Section**:
  - **Index**: Presenting the tour information, search bar, team members, and ways of contacting us (index-03-new.html).
  - **View all Destinations, Attractions, or accommodations**: User can navigate to these pages to see all destinations, attractions, or accommodations (all_destinations.html, all_acco.html)
  - **Destination Details**: Page for exhibiting the description, location, VR experience, prices, and reviews of a chosen destination (detail_destination.html).
  - **See Reviews**: Browse the reviews sent by registered users (detail_destination.html, detail_attraction.html, detail_accommodation.html).
  - **Check Company Profile**: User can browse the information about the travel agency (about-us.html).
  - **Ask AI for help**: The website provide virtual guide with GPT3.5 language model. Visitors can ask AI for suggestions and information (baseHTML.html).
  - **Login & Register**: Login or register to be a member of the website to enjoy our services (login.html).

- **Customer Section**:
  - **Visitor Section**: All functionalities included in the visitor section.

- **Send Reviews**: The customer can express their ideas after the trip (detail_destination.html, detail_attraction.html, detail_accommodation.html).
- **Add to Cart**: Customers can add attractions and accommodations to the cart for later planning (detail_attraction.html, detail_accommodation.html).
- **Plan & Book Trip**: Users can customize their travel by dragging and dropping the card-like blocks (drag_drop.html).
- **Check My Tour**: After booking, customers can have a quick view of their tour (mytours.html).
- **Communicate with Staffs**: Customers can ask staff for help through the communication function (chat_room.html).

- **Staff Section**:
  - **Visitor Section**: All functionalities included in the visitor section.
  - **Communicate with Customers**: Staff is responsible for answering queries given by customers (chat_room.html).
  - **Add Destination**: Staff can add new tour destination to the website (add_destination.html).
  - **Supervise Website Traffic**: Staff can check the visually presented data on the supervision page (staff_supervise.html).
  - **Manage Reservation Made**: Staff needs to alter part of the reservation that is temporarily inaccessible (staff_cancel_reservation.html).

3.2.2 Core Sub-systems

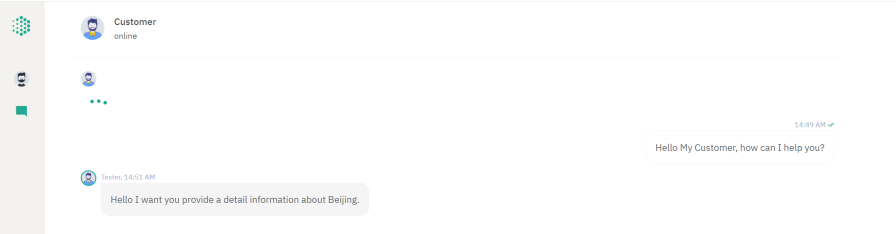


Figure 8: Chat-room Sub-System

- **Chat-room Sub-System** The website allows instant communications between customers and staff within a chat-box-based chat room(Figure 8). It is implemented using*flask-socket-io* in the following section.
- **Maps Sub-System** Figure 9, 10, and 14 contains details of our map subsystems.
  - **JVectorMap** JVectorMap open-source JavaScript-based mapping library that enables the display of vector maps and data visualizations on web pages, as well as allowing developers to create and customize interactive maps around the world. The map is displayed in Figure 15.
  - **Google map API** The Google map allows a dynamic map starting around the travel attraction/accommodation where users can interact using buttons and mouse to drag and scroll. The map is displayed in Figure 19.
  - **EChartsJS-3D earth**
    Apache ECharts is a powerful, interactive charting and data visualization library for browsers. The globe is illustrated in Figure 10. The details of the configurations are displayed in Figures 11 and 12.

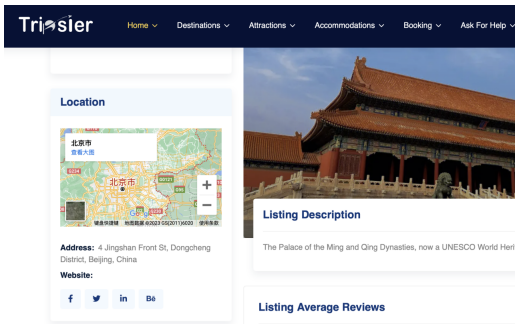


Figure 9: Interactive map in attraction/accommodation details

Figure 10: 3D earth with dynamic 3D lines connecting destinations

```
globe: {
    top: '5%',
    globeRadius: 130,
    baseTexture: baseImg,
    displacementScale: 0.05,
    displacementQuality: 'high',
    shading: 'realistic',
    realisticMaterial: {
        roughness: 0.2,
        metalness: 0
    },

    postEffect: {
        enable: true,
        depthOfField: {
            // enable: true
        }
    },
    light: {
        ambient: {
            intensity: 1
        },
        main: { // 主光源
            intensity: 0,
            shadow: false
```

Figure 11: globe configurations

```
var backgroundImg = new Image();
backgroundImg.src = "../static/galaxy.png";
backgroundImg.width = '100%';
backgroundImg.height = '100%';
var baseImg = "../static/earth_new.png"; // 背景纹理贴图
var scanImg = "../static/scan.png"; // 扫描光影效果
var myChart = echarts.init(document.getElementById("showC"));
var config = { // 扫描线条配置
    lineWidth: 0.5, // 扫描线条宽度
    color: '#01CADE', // 线条颜色
    levels: 1,
    intensity: 3, // 强度
    threshold: 0.01
}
var canvas = document.createElement('canvas');
canvas.width = 4096;
canvas.height = 2048;
context = canvas.getContext("2d");
context.lineWidth = config.lineWidth;
context.strokeStyle = config.color;
context.fillStyle = config.color;
context.shadowColor = config.color;
```

Figure 12: background, canvas and line configurations

- **Admin Sub-System**

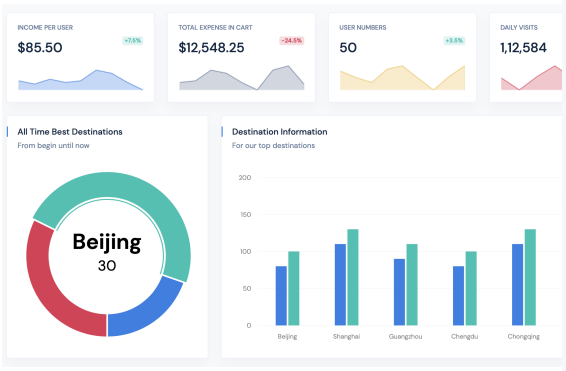


Figure 13: Charts displayed for admins

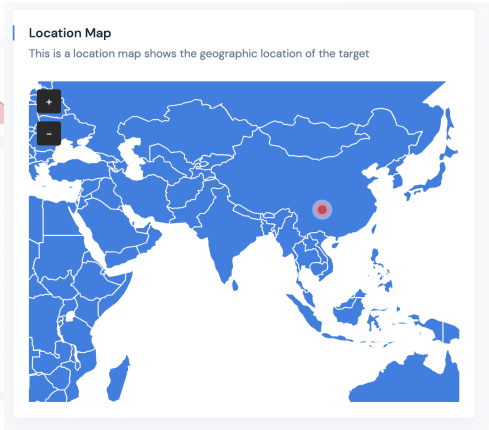


Figure 14: Map in travel information details displayed for admin side

– **Chart.js** is a free and comprehensive JavaScript library that enables the creation of HTML-based charts, being one of the simplest JavaScript visualization libraries. ChartJS allows charts like in Figure 13 for the admin pages.

- **Shizuku AI - Virtual Guidance Sub-System**
  As displayed in Figure 15, users can ask the virtual guide Shizuku using text and the text will be sent to the ChatGPT API and further obtain responses from the Open AI Server. The response token will be sent back to two output end-points: TTS (Responsive Voice) and text in the textbox of Shizuku.

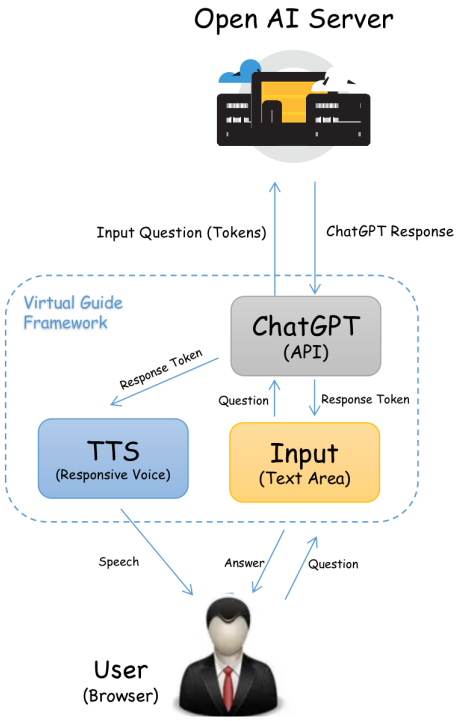


Figure 15: Virtual Guide Framework

```
var data = JSON.stringify({
  "prompt": document.getElementById("chat-gpt-input").value,
  "max_tokens": 2048,
  "temperature": 0.5,
  "top_p": 1,
  "frequency_penalty": 0,
  "presence_penalty": 0,
  "model": "text-davinci-003"
});
console.log(data);
await printMessage('Thinking...');
await xhr.send(data);
```

Figure 16: Current GPT Configurations

  - **Live2D Widget**
    Live2D widget is a tool for displaying interactive character animations in projects such as web pages, apps, or games. It uses Live2D technology, which can show 2D character animation more lifelike and smoothly. With Live2D widgets, we dynamically display characters on the web as our virtual guidance as displayed in Figure 4.
  - **ResponsiveVoice**
    ResponsiveVoice is an online text-to-speech tool that provides voice synthesis services for web pages. With the help of ResponsiveVoice, we can translate text contents on a web page to natural, fluid speech output.

```
responsiveVoice.speak(response,\$('#voiceselection').val())
```

  - **GPT3.5**
    GPT3.5 is a neural network machine learning model that generates any type of text from internet data. The Virtual Guide system is developed based on GPT-3.5, and it can allow the shizuku character to respond to questions from the users in an accurate and effective manner. It can be regarded as a extremely advanced auto-complete, where the model processes users' text prompt and tries to predict what's most likely to come next. Figure 16 contains details on current GPT configurations in our website.

- **VR-Virtual tour**
  3D VR video is embedded on the web page using the code displayed in Figure 17 and examples in 18.

```html
<div id="embedded-video">
    <button class="close-video-btn" type="reset" id="close-video-btn">
        <svg xmlns="http://www.w3.org/2000/svg" class="h-6 w-6" fill="none" viewBox="0 0 24 24" stroke="currentColor" stroke-width="2">
            <path stroke-linecap="round" stroke-linejoin="round" d="M6 18L18 6M6 6l12 12"></path>
        </svg>
    </button>
    <iframe id="video-iframe"  src="https://www.youtube.com/embed/jZIr7PL1SDU" title="YouTube video player" frameborder="0"
        allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>
</div>
```

Figure 17: 3D video embedding in html

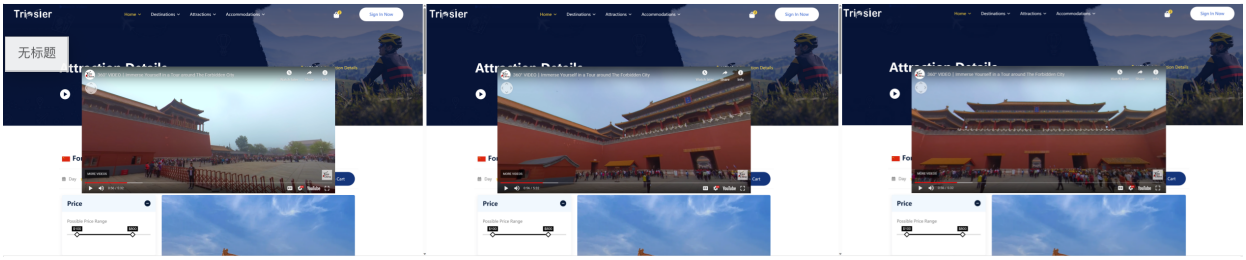


Figure 18: Virtual tour in attraction

3.2.3 Other Development Technologies

- **Bootstrap**
  Bootstrap is a front-end development framework aimed to help developers conveniently construct powerful **responsive** website. We apply Bootstrap to the project to satisfy the need of the client's requirements that the website should be correctly shown on different size screens.

- **JQuery**
  JQuery is a JavaScript library that provides many convenient features and methods to make it easier for developers to manipulate HTML documents, handle events, perform animation effects, etc. We choose to use JQuery to assist our development process.

3.3 Back-end

3.3.1 Core APIs

- **flask-socket-io**
  SocketIO is utilized in the website to achieve the instant and fluent communication and information exchange between customers and staffs through chat box.
- **Numpy**
  Numpy is a Python library that provides a powerful and efficient way to work with numerical data. It enables us to create and manipulate multidimensional arrays, which are the core data structure of Numpy.
- **SkLearn**
  SkLearn is a Python library that offers a rich and user-friendly interface to a variety of machine learning algorithms and tools. SkLearn enables us to perform different stages of a machine learning pipeline, such as data preprocessing, feature engineering, model training, and deployment.
- **flask-babel**
  Flask-Babel is a Flask extension which enables internationalization (i18n) and localization (l10n) support to any Flask web application.

– **Baidu Translation API**
Baidu Translate API is a free and unlimited machine translation API developed by Baidu that supports 201 languages. The details are shown in Figure 16.

```python
def translate(query):
    salt = random.randint(32768, 65536)
    sign = __make_md5(APPID + query + str(salt) + APPKEY)

    # Build request
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}
    payload = {'appid': APPID, 'q': query, 'from': FROM_LANG, 'to': TO_LANG, 'salt': salt, 'sign': sign}

    # Send request
    r = requests.post(URL, params=payload, headers=headers)
    result = r.json()

    resp = (json.dumps(result, indent=4, ensure_ascii=False)).replace("\n", "")
    return json.loads(resp)["trans_result"][0]["dst"]
```

Figure 19: Send request to Baidu API

3.3.2 Algorithms

- **Dispatch mechanism**
The dispatch system of staff communication is realized through the usage of staff workload table. When a staff logged in to this system, the staff's id is first to the table and initialized with a workload value of 0. When staff log out from this system, the value is updated to *inf_num (99999999)*. The workload value is updated when a staff communicates with a customer, adding by 1 when the communication starts and minus 1 when the conversation ends. Eventually, just as displayed in the code below, the most suitable staff is selected based on the minimum workload value.

```
        def get_suitable_staff_id():
    """
    get the best staff id for the customer
    :return: the staff id
    """
    min_value = inf_num
    for id,value in staff_dict.items():
        if(min_value>value):
            min_value = value
    if(min_value == inf_num):
        # No available Staff now
        return "No"
    else:
        return min_value
```

- **KNN-based travel information recommendation**:
In this project, we use Numpy to build a KNN network to accomplish the recommending system, which can provide various information on the homepage for customers. The recommendation request cycle is displayed in Figure 20. The process is basically as follows. A visit is created to every tag and recommendations are generated with 15 highly-recommend, 10 medium-recommend, and 6 random separated in three sub-pages. In this process, the recommendation index is first read and added to the recommendation group. The updation of the recommendation index is proceeded with first reading the tag of the visit and the view numbers of the tag as training data (giving weight to the first element as 10). The order will be disrupted to simulate random recommendations. Eventually, the tags are read and a recommendation is completed.

Part of the code is presented as follows:

```
    def knnTrain():
    feature_group = []
    score_group = []
    visit_all = db.session.query(Visit).filter(Visit.user_id == current_user.id).all()
    for visit in visit_all:
        f = []
        tags = visit.Tag.split(",")
        f.append(int(tags[0])*10)
        f.append(int(tags[1]))
        f.append(int(tags[2]))
        f.append(int(tags[3]))
        feature_group.append(f)
        score_group.append([visit.Visits])
    knn = KNeighborsClassifier(n_neighbors = 1)
    knn.fit(feature_group, score_group)
    return knn

    def knnPredict(knn,tag):
        ## Read the tags of Visit, update the recommending index
        l = []
        tags = tag.split(",")
        l.append(int(tags[0])*10)
        l.append(int(tags[1]))
        l.append(int(tags[2]))
        l.append(int(tags[3]))
        l = np.array([l])
        recommened_point = int(knn.predict(l))
        return recommened_point
```

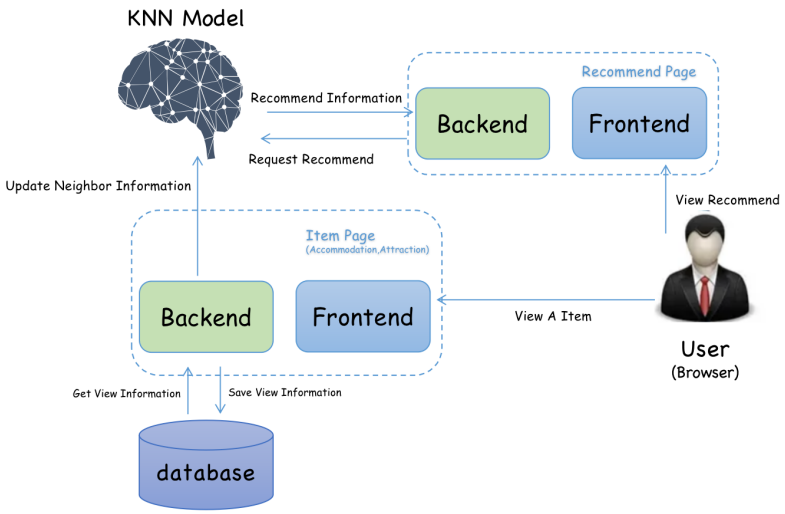


Figure 20: Recommendation with KNN

3.4 Deployment and Configuration

The Dublin-based virtual server provided by UCD is used to deploy our project. The website is running on the Flask framework and using the shell command as follows to run it in the background.

```
FLASK\_APP=main.py flask run
```

3.5 Security Measures

- **SSL certificate**:
  Enabling *https* by installing SSL certificates and configuring them with your web server will allow the website to create an encrypted and trusted connection between servers and now all the messages have a low possibility leak, preventing any man-in-the-middle attacks.

- **Secure authentication**:
  Apply longer password policy (6 digits). Apply role-based access control: authenticates a user and then grants or restricts access based on their position or role with required data encryption techniques.

- **Input validation**:
  Validate using wtforms.validators and email validator when logging in and registering. Input validation ensures that user input is properly sanitized before it reaches the application's back-end. this helps prevent various kinds of attacks, such as sql injection and cross-site scripting.

- **Rate limit**:
  Through limiting requests for sensitive calls like login and password reset and API usage, etc. brute-force attacks could be prevented. By limiting the number of requests a user can make in a certain period, you can slow down attackers who may try to guess passwords or exploit vulnerabilities.

- **Logging and monitoring**:
  Utilize flask-logging to create logging files. Monitor system resource usage, such as memory using memory profiler. This will help identify potential bottlenecks or performance issues.

- **Cross Site Request Forgery**:
  The form validation API flask-wtf will apply CSRF validation through *form.validate_on_submit()*.

3.6 Testing

3.6.1 Functionality Test

Objective: Tests can check that the site meets the user's needs and that there are no errors that could affect the user's use

Approach: Black Box Test (Manually)

Standards: There are no any issues when a user operating all the functionalities in the ***user manual*** instructions. The website also has responsive layout based on screen size and mobile adaptive layout.

3.6.2 Concurrency Test

Objective: The purpose of concurrent testing is to test whether a website can effectively respond to user requests during peak times

Approach:
Siege (Concurrency Testing software)

Standards: We use Siege to simulate there are 20 users send 50 POST requests to our website's index simultaneously and check if there are lots of requests not been receive successfully or the response time is unacceptable to the users.

Result: After the test, our system receives 1000 hits completely and the availability is 100% which means we could receive all the requests sent by the users. The concurrency is 3.30, elapsed time is 43.15s in total.

```
Transactions:                 1000 hits
Availability:               100.00 %
Elapsed time:                43.15 secs
Data transferred:          3427.88 MB
Response time:                0.14 secs
Transaction rate:            23.18 trans/sec
Throughput:                  79.44 MB/sec
Concurrency:                  3.30
Successful transactions:      1000
Failed transactions:             0
Longest transaction:          0.45
Shortest transaction:         0.02
```

Figure 21: The Testings Results

3.6.3 Security Test

Objective: The main purpose of security testing for websites is to discover and identify potential security loopholes and weaknesses, so as to ensure that the website can maintain security and stability during use.

Approach: Manual Test

Standards: There are no issues when a user operates these tests.

- **Penetration Testing**
  - A user whose role is customer cannot simulate the operation of the staffs (e.g. access management page, edit other customer's plan or chat with other customers.)
- **SQL Insertion Testing**
  - A user's input with SQL statement will be identified and not be accepted by our database For example, user input with

    $$'\mathbf{OR}'\mathbf{1}'='\mathbf{1}$$

    Some databases could see it as a part of SQL statement and execute it like this:

    $$\mathbf{SELECT} * \mathbf{FROM\ users\ WHERE\ name} =''\mathbf{OR}'\mathbf{1}'='\mathbf{1}'$$

    which would return a list of username.

3.7 Maintenance and help guide

Maintenance: We suggest developers to monitor logs, errors and system resource; backup vital data incluing source code, database, etc.; weekly review application performance based on the requirement section; check for broken links, outdated content; upgrade APIs and perform necessary updates in security or technical measure based on new technologies; monthly run comprehensive tests based on the testing section.

Help: Please contact (+86)153****4303 if any unsolvable error occurs or you find any mistakes in code/documents.

4 Team work and collaboration

4.1 Project Management Approach

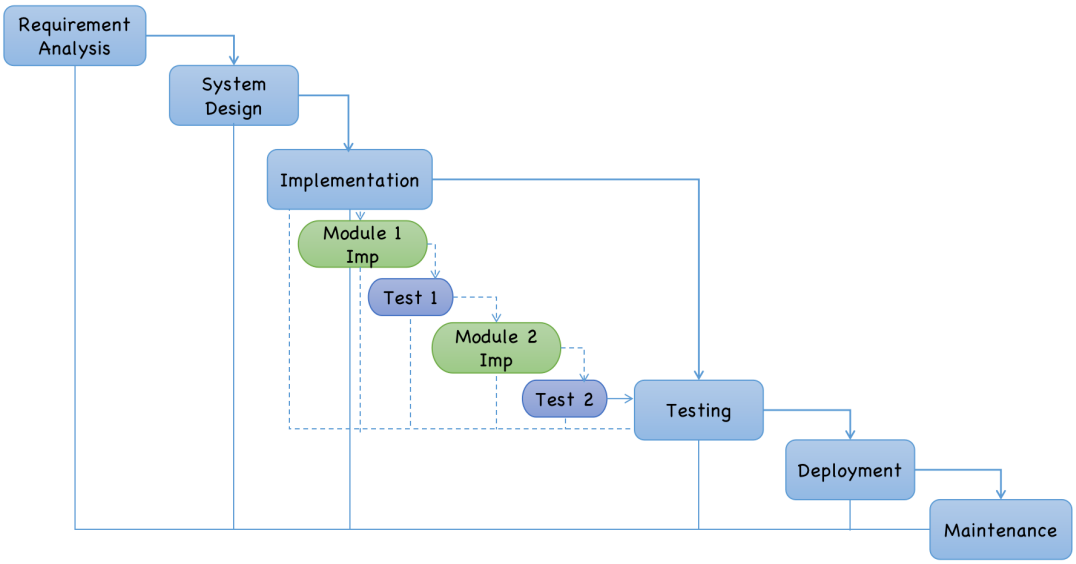


Figure 22: The Waterfall Model-based approach in our development process

Our software development team adopts the waterfall project management method (Figure 22), which is a linear-sequential life-cycle model, where all the specifications are planned at the beginning and developed with structured phases. Each phase of the project is planned in an order in advance and is only moved forward after the completion of the previous one's requirements. Since functional and technical specifications are already determined within the scope of work required, this model illustrates much more effectiveness compared to other methods. In addition, the workload will be appropriately distributed based on each member's skills and strengths, ensuring sufficient motivation and manpower allocation for each key responsibility, as well as the effectiveness of the overall teamwork.

4.2 Equality, Diversity and Inclusion

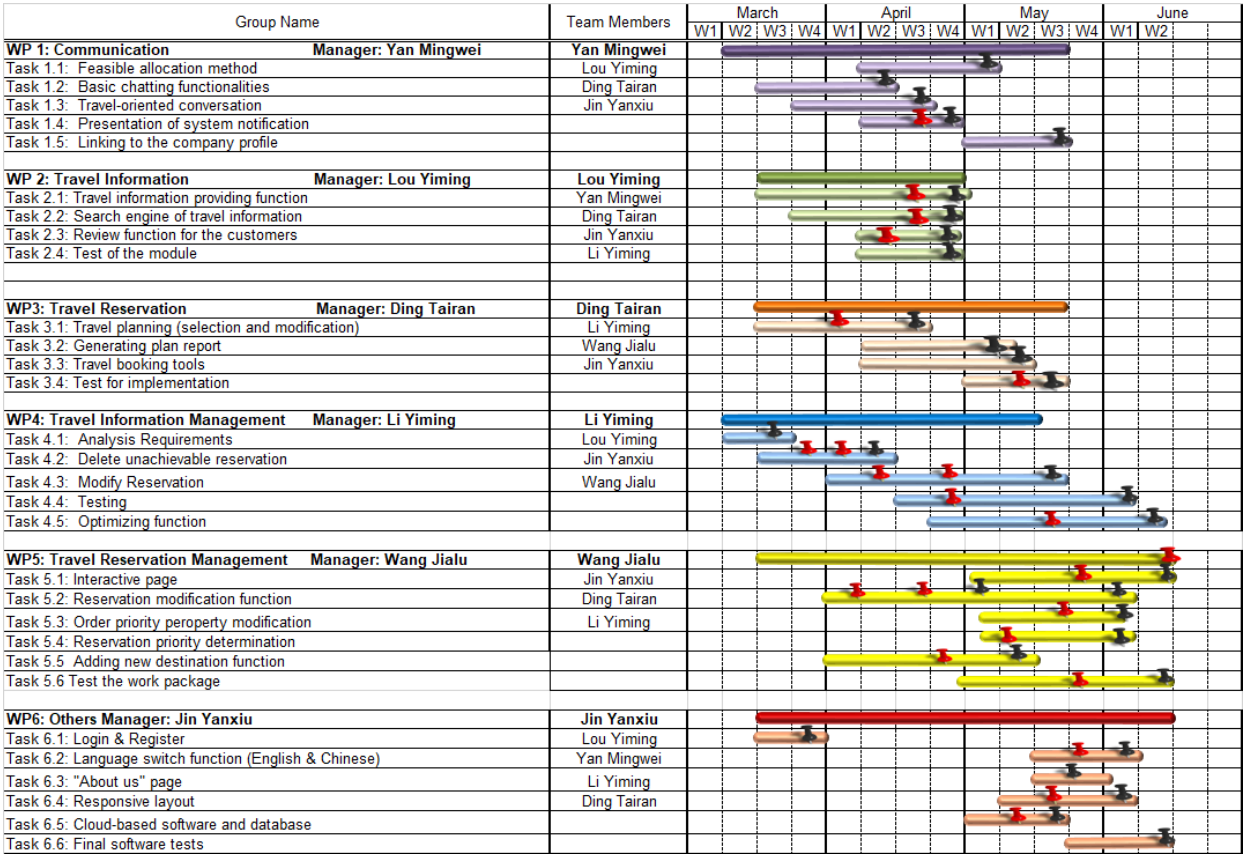
Our software development team encourages equality, diversity and inclusion.

- **Equality in team-work** We provide equal job opportunities and fairness to all the members, through which each member can select work loads based on their interests and skills.
- **Diversity** We understand, accepting and value differences between people. We have learned about discrimination laws all-together and our team strictly forbade any means of bullying, harassment or discrimination. We embraced members from different ethnicities backgrounds (we have a member who is a Korean nationality Chinese), from different regions (our members are from Shanxi, Beijing, Shandong, etc.) and from different religions.
- **Inclusion** We strove hard to guarantee an safe and comfortable working atmosphere where all the members are valued, heard, respected, and supported and will not hesitate or be feared to come up with new/different ideas and raise problems and issues they've witnessed. No members in our team ever felt afraid of facing retaliation for a complaint and all of the members feel a sense of belongingness.

4.3 Plan and Process

4.3.1 Deadlines

The project timeline is as follows: The customer portal(mainly WP1, WP2 and WP3) is due completed on week 8, and the staff portal(mainly WP4 and WP5) is due on week 14. Work package 6 is operated throughout the whole process of the project. Functional test and the performance test of the first deliverable was settled in week 8 as soon as it was completed and the final software testings takes place in week 14 along with optimizations such as improvement on data structure to enhance the website's performance on each testings.



Group Name	Team Members
WP 1: Communication Manager: Yan Mingwei	**Yan Mingwei**
Task 1.1: Feasible allocation method	Lou Yiming
Task 1.2: Basic chatting functionalities	Ding Tairan
Task 1.3: Travel-oriented conversation	Jin Yanxiu
Task 1.4: Presentation of system notification	
Task 1.5: Linking to the company profile	
WP 2: Travel Information Manager: Lou Yiming	**Lou Yiming**
Task 2.1: Travel information providing function	Yan Mingwei
Task 2.2: Search engine of travel information	Ding Tairan
Task 2.3: Review function for the customers	Jin Yanxiu
Task 2.4: Test of the module	Li Yiming
WP3: Travel Reservation Manager: Ding Tairan	**Ding Tairan**
Task 3.1: Travel planning (selection and modification)	Li Yiming
Task 3.2: Generating plan report	Wang Jialu
Task 3.3: Travel booking tools	Jin Yanxiu
Task 3.4: Test for implementation	
WP4: Travel Information Management Manager: Li Yiming	**Li Yiming**
Task 4.1: Analysis Requirements	Lou Yiming
Task 4.2: Delete unachievable reservation	Jin Yanxiu
Task 4.3: Modify Reservation	Wang Jialu
Task 4.4: Testing	
Task 4.5: Optimizing function	
WP5: Travel Reservation Management Manager: Wang Jialu	**Wang Jialu**
Task 5.1: Interactive page	Jin Yanxiu
Task 5.2: Reservation modification function	Ding Tairan
Task 5.3: Order priority peroperty modification	Li Yiming
Task 5.4: Reservation priority determination	
Task 5.5 Adding new destination function	
Task 5.6 Test the work package	
WP6: Others Manager: Jin Yanxiu	**Jin Yanxiu**
Task 6.1: Login & Register	Lou Yiming
Task 6.2: Language switch function (English & Chinese)	Yan Mingwei
Task 6.3: "About us" page	Li Yiming
Task 6.4: Responsive layout	Ding Tairan
Task 6.5: Cloud-based software and database	
Task 6.6: Final software tests	

Figure 23: The Gantt Chart

4.3.2 Issues and solutions

- **Cooperation**: Due to the Covid-19 during May, the members were unable to meet in reality. Out team arranged virtual meeting with tencent meeting.
- **Conflicts**: We had a conflict on "Start coding in week 3 or spending more time on the design phase?". We initiated a face-to-face meeting in the seminar room where everyone exchanged their thoughts to the other team members. Any question or opinion was addressed after the speaker finished his/her/their speech.
- **Technical Challenges:**
  - **Network latency:** The API servers used are located in different regions (e.g. ChatGpt API server in the US and Baidu translation API server is in China). Website needs to download JavaScript files from a remote server. If the server is not performing well or overloaded, it can't respond to requests in a time. *Solutions*: Make some APIs invisible to users,

not allowing them to use them directly, but letting the server preload them. Download JavaScript files locally to reduce the number of requests to the server and the download time. Split the JavaScript code into multiple modules allows the page to continue rendering while the JavaScript file is being downloaded.

- **Unique id for each module**: Due to the complexity of planning page functionality, different modules need to be given unique identifiers, but since the number of modules comes from database, the conflict occurs between the generation of elements by the js loop and adding them to the html and the generation of element ids. *Solutions*: Declare a counter variable in the JavaScript code. Within the loop, each time an element is generated, the value of the counter is used as part of the unique identifier (id) and the value of the counter incremented by 1.
- **Chat message refreshing automation**: Initially we wanted to use ajax to achieve it, but found that asynchronous refreshing is not good enough. *Solutions*: Utilized socketio with ajax for local refreshing, instead of database, since socketio can realize a two-way communication between server and client, encapsulating the complex TCP/IP communication protocol, which can greatly simplify the workload of chat room communication design.

5 Future Work

5.1 Upgrades/Troubleshooting

- **Visit outside from Ireland**: All API calls made up until this point have been directed to a server in Ireland. The user in China or other countries will be able to load considerably more slowly than in Ireland, even if loading is unsuccessful, thanks to this setup. Additionally, we only offer English as the default input.
- **Cloud database**: For further updates in database and data management, a cloud database could be a wise choice. It supports **Object-Oriented Data Storage** which enables complete freedom to create, read, update, or remove any type of data since all of them between the front and back end are abstracted into objects. In this case, the database is scaled automatically and there will be no more concerns on data size and volume of traffic generated.
- **Front-end Resource Overload**: When front-end resources are too large and complex, it may lead to slower page loading, poorer user experience and reduction in portability. Developers are able to reduce HTTP requests by merging CSS and JavaScript files, using technologies such as CSS sprites and HTTP2. CSS sprites reduce the number of browser requests to HTTP by combining multiple images into a single image, which in turn reduces page load times. HTTP/2 allows multiple requests on a single TCP connection. CDNs (Content Delivery Networks) can also be used to reduce requests to the server by caching static resources.

5.2 New Features

- Real-time pricing and location updates for attractions and accommodations
- Provide membership service, in which customers can have access to a set of exclusive resources or opportunities with fees
- Add a section where users can upload travel blogs
- Integrate some of the functionalities with social media, including but not limited to creating a social login option and a sharing link.