A

A1.Activity diagram

A1.1.The New Travel Creation:

- Create a travel plan: The user clicks to enter the new journey, and the new journey is created
- Submit the location: The user agrees to the location requirements and submits their current location
- Select a destination: The user adds a submit travel destination to the address bar or map
- Select the travel mode: the user submits the travel mode to the system
- Confirmation information: If wrong, return the change and confirm again after the change is completed. If correct, start the journey
- Record the route: Record the route and add it to the historical trip before ending the journey

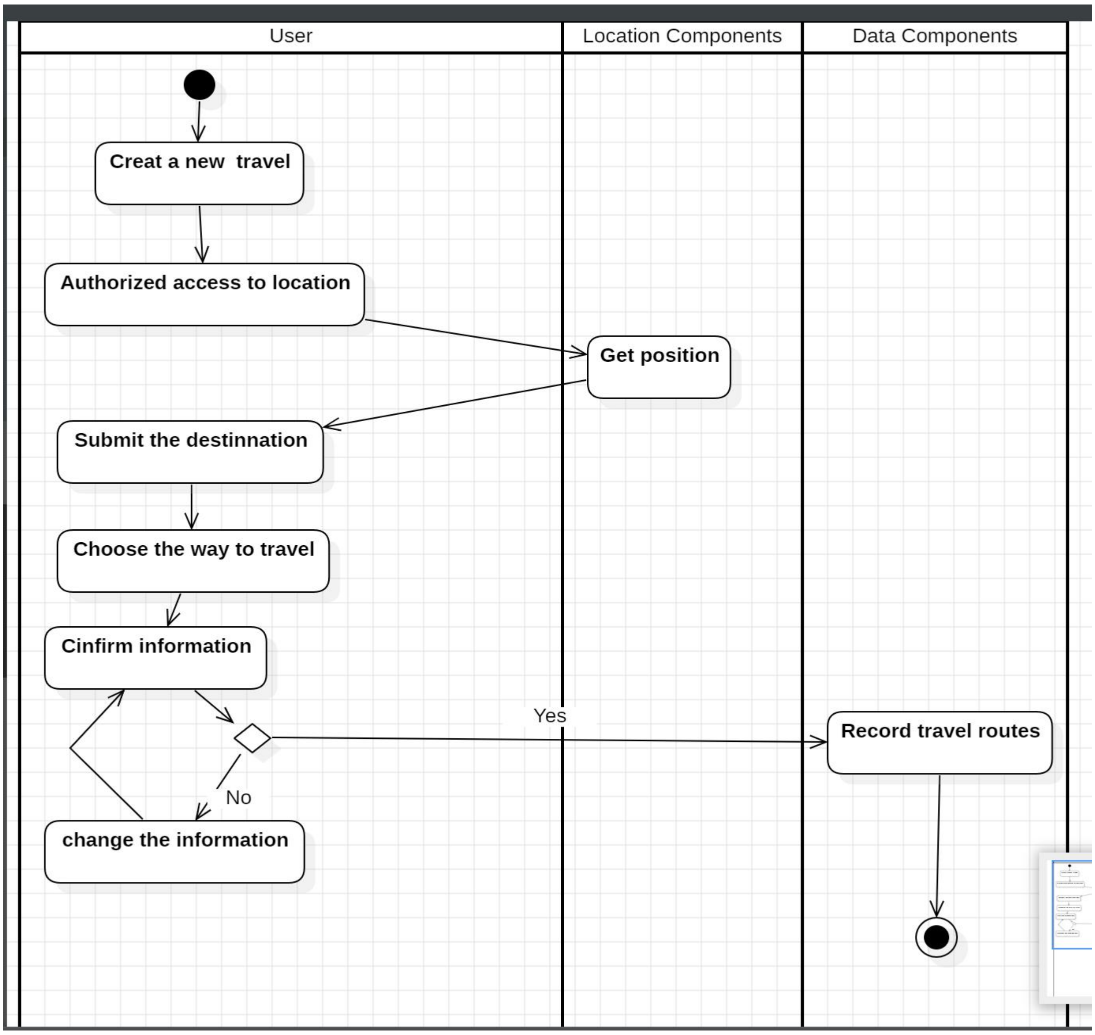

Figure 1.The New Travel Creation

A1.2.Reward System:

- View the number of points: users enter the reward page to view their number of points
- Select prizes: The user selects rewards according to his / her number of points and submits the selection to the system
- Judge whether it complies with the reward rules: the system receives the user's choice, judge whether the user complies with the reward rules, such as whether to participate in relevant activities, whether the number of points is sufficient, etc., if it meets the standards, the reward will be awarded. Otherwise, the prompt does not meet the conditions, and the reward fails.

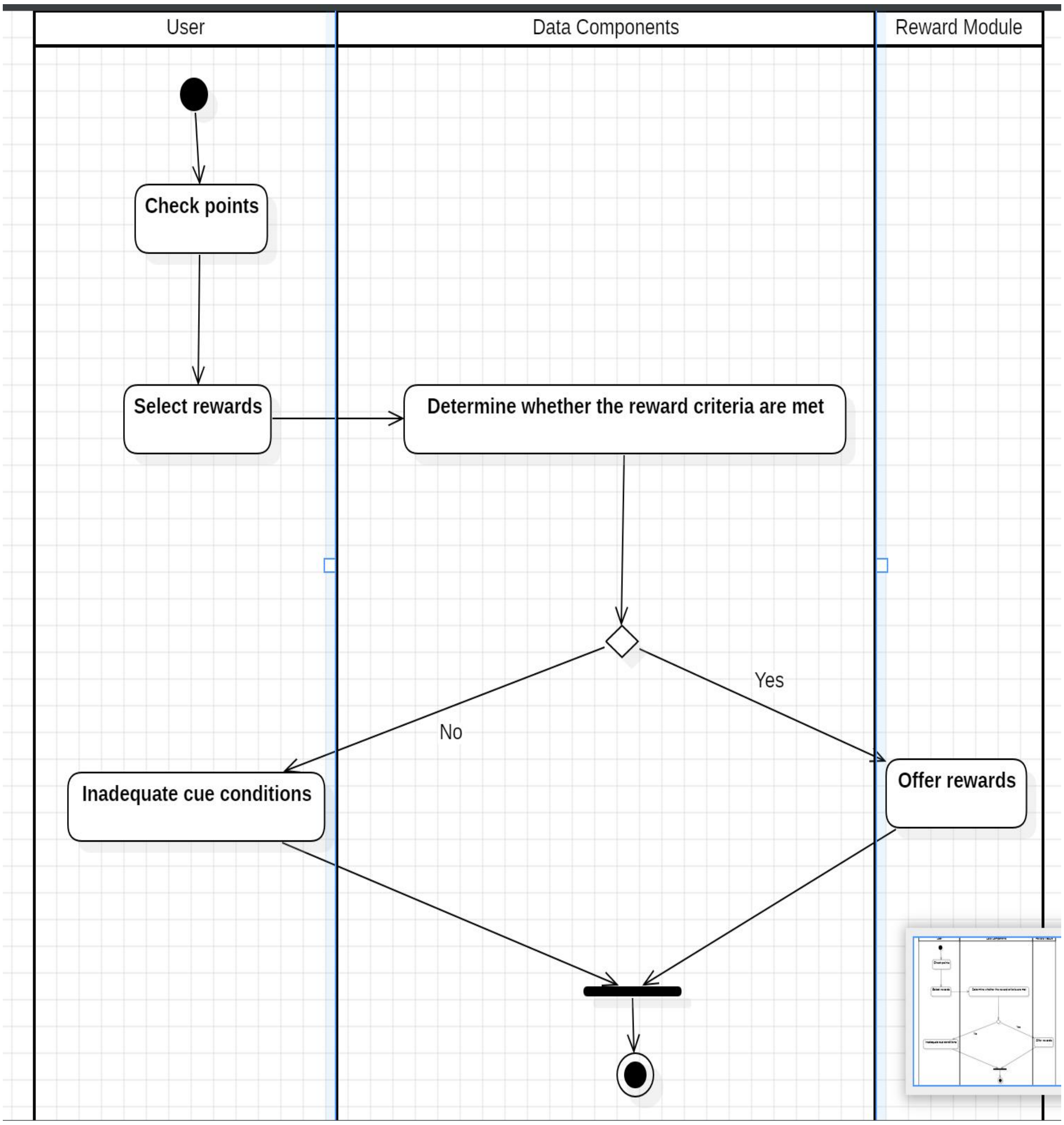

Figure 2.Reward System

A2. Class diagram

A2.1.Noun-verb Analysis

- Use Case 1: User Registration

Flow of Events:

1. The **user opens** the application and **clicks** on the "Register" **button**.
2. The user is prompted to **fill in** details such as full **name**, **age**, **email address**, **phone number** and **password**.
3. User provides necessary **details**.
4. The **system validates** the phone number and email format.
5. The system **checks** if the email and phone number are unique.
6. User **agrees** to the terms and conditions.
7. User **clicks** on "Submit" button.
8. The system **sends** a **verification code** to the provided email.
9. The user enters the verification code.
10. The system validates the code and confirms the user's registration.

Scenario:

Lao, as a **new user**, downloaded a low-carbon travel distance recording software and started to **register** a new **account**. He followed the prompts to enter his e-mail address and password and agreed to the **privacy policy**. After registration, he immediately started using the software to record his travel data.

- Use Case 2: New Travel Creation

Flow of Events:

1. The user clicks on the "**Log a Trip**" option from the **main dashboard**.
2. The user needs to **select a starting point** (his/her current **location** is set as default).
3. The user selects a **destination**.
4. The user selects a mode of transportation (e.g., **car**, **bike**, **walk**, **public transportation**).
5. The system **calculates** the **distance** and **estimated** carbon footprint based on the mode of transportation.

6. The system **displays** the **trip details** and estimated **carbon footprint**.
7. The user **confirms** and **saves** the trip details.
8. The system **stores** the trip and updates the user's **total carbon footprint**.

Scenario:

Chi uses a low-carbon travel distance recording software to start recording a new journey. He **enters** the **start and end locations**, selects the **trip type** and **time**, and **sets up synchronization** to **social media**. He finds the software very useful in helping him to record his journey and generate a low carbon **journey ranking**. The software also calculates distances and estimates carbon footprints based on different **modes** of transportation. After completing the journey, Chi **confirms** and **saves** the trip information and **shares** it on **social media platforms**.

- Use Case 3: Record Travel Activity

Flow of Events:

1. The user **signs up** or **logs into** the system using his/her **credentials**.
2. After logging in successfully, the user navigates to the "**Record Travel Activity**" page. It will be accessible from the dashboard.
3. After "Record Travel Activity" selection, the system presents a user-friendly interface, which is responsible for travel details input. This form includes:

   • **Mode of Transportation**: The user **selects** the transport from the **list**, such as car, bus, bicycle, walking, etc.

   • **Distance Traveled**: The user **inputs** the distance he/she traveled in kilometers or miles.

   • Additional Information: The form may request additional details, depending on the transport, such as fuel type for a car or route information for a bike ride.

4. The user enters accurate and honest travel details into the form. (The application can **recognize** the travel activity by itself too).
5. The system records the travel activity data and **associates** it with the user's **profile**.

6. After travel activity recording, the system recalculates user's carbon footprint based on the new data. This calculation considers **factors**, such as transport and distance traveled. If additional information was provided, the system adds it into the calculation.
7. The user's profile **updates** automatically to reflect the latest carbon footprint information. The new information is presented in a clear and understandable format, such as a graphical representation. It demonstrates recent activity’s effect on user's carbon footprint.
8. After recording completion, the user has several **options**, such as:
    1. He/she may record another travel activity, **returning** to step 3 and **repeating** the process.
    2. The user can return to the dashboard to explore other features and options in the application.

Scenario:

Sarah, a registered user, logs into her account in the application. On the dashboard she notices the "Record Travel Activity" feature and decides to use it. After feature selection, she sees a user-friendly **form**. One day Sarah goes to work by bicycle. She covers a 6 kilometers distance. She chooses "**Bicycle**" as a transport and adds the distance into the form. The system could also **reflect** it automatically. After form completion, she **submits** the **information**. The system records her bicycle ride and associates it with her profile. The **algorithm recalculates** Sarah's carbon footprint and adds her bicycle ride information. From now on, Sarah's profile reflects her recent sustainable travel activity. Her updated carbon footprint displays in a graphical format. Feeling motivated, Sarah decides to record another travel activity. She records her afternoon walk. The system updates her profile again.

- Use case 4: View carbon footprint ranking

Flow of Events:

1. The user clicks on the "**Carbon Footprint Ranking**" option from the main dashboard.
2. The system calculates the user's total carbon footprint based on the **recorded journey**.
3. The system **retrieves** the **carbon footprint ranking** of all users.
4. The system displays the rank of the user and displays the **leader board** of the top user.

5. Users can **filter** rankings based on **daily**, **weekly**, and **monthly metrics**.
6. Users view and compare their rankings with other users to understand areas of improvement.

Scenario:

Ga uses a low-carbon travel distance recording software and finds that he can check his carbon footprint ranking. He compares his ranking with the data of his friends and realizes that his carbon footprint is much lower than others. He is proud that he is able to travel more environmentally friendly and decides to continue to maintain a low-carbon lifestyle and do his part to protect the environment.

- Use case 5: Redeeming rewards.

Flow of Events:

1. User clicks on the "**Rewards**" section from the main dashboard.
2. The system displays the available **rewards** and their **respective point costs**.
3. The user selects the reward they wish to redeem.
4. The system confirms the **redemption** and **deducts** the necessary points from the user's account.
5. The user receives a **confirmation message** and details on how to claim/use the reward.
6. The system **records** the **redemption** for future reference.

Scenario:

Co has accumulated a lot of points by using a low-carbon travel distance recording software. He found that these points can be used to **exchange** for some **desired items**, so he chose an environmental bag. When he successfully exchanged his points for the bag, he was glad that he could do his part for the environment in this way.

- Use Case 6: Viewing and updating travel history

Flow of Events:

1. The user clicks on the "**Trip History**" section from the main dashboard.
2. A **list of recorded trips** is displayed with details such as **date**, **mode of transportation, distance and estimated carbon footprint**.
3. User can **click** on any trip to view the details.
4. The user can **edit** or **correct** the trip details and **save** the changes.
5. The system recalculates the carbon footprint of the updated journey and reflects the changes in the user's total carbon footprint.

Scenario:

Fao uses the Low Carbon Travel Distance Record software to view and update his travel history. By viewing his history, he reviews his previous trips and updates his recent trip data. His friend Fen sees his records and praises him. This feature makes it easier for him to manage and plan his trips and motivates him to travel more environmentally friendly.

● Use Case 7: Calculate Carbon Footprint

Flow of Events:

1. The user is logged into his/her account on the application.
2. The user navigates to the dashboard, where he/she sees and selects the "**Calculate Carbon Footprint**" feature.
3. The system **retrieves** and **compiles** all **travel activities**, which have been previously recorded by the user in the application.
4. The user **chooses** the period of activities that he/she wants to be calculated.
5. The system performs a detailed calculation of the user’s carbon footprint. It is based on the recorded activities, **considering** relevant environmental factors, such as **emissions of various transportation**.
6. The updated carbon footprint is displayed to the user, demonstrating his/her environmental impact. This information is visually presented, including a chart or graph.
7. The user has the option to explore **historical carbon footprint data**, which allows him to **track** his/her progress over time. By selecting "**View Historical Data,**" the user can see how his/her actions have influenced his carbon footprint over weeks or months.
8. After the calculation of the carbon footprint, the user can return to the dashboard.

Scenario:

John, the application user, is **logged** into his **account**. He is interested in **tracking** his environmental impact over a certain period. He notices the "**Calculate Carbon Footprint**" feature on his dashboard.

John selects "Calculate Carbon Footprint" feature. The system **retrieves** all the travel activities he has previously recorded within the app. It includes details, that he provided before, such as the mode of transportation, distances, and other relevant ones. He chooses the period he is interested in. The system performs a thorough calculation over the period that John chooses. The system considers all environmental factors associated with different modes of transportation.

Once the calculation is complete, the updated carbon footprint is **displayed** visually. It provides to a clear **chart or graph** which helps John to understand his environmental impact.

John also explores “View Historical Data” option. It allows him to track his progress over time.

Satisfied with the provided calculation and information, John decides to explore other features on the app, such as participating in challenges and earning points for his sustainable travel.

Word/Phrase	Accepted	Reason
user	**User**	This is a subclass class for all users
button	No	Too Abstract
name	No	It will be the data of user
age	No	It will be the data of user
email address	No	It will be the data of user
phone number	No	It will be the data of user
password	No	It will be the data of account
system	**Database**	The system will automatically verify with registered or logged in users
detail	No	Too broad
verification code	No	This is attribute of Validate()
account	**Account**	Represents a user's account entity or object.

privacy policy	**privacyPolicyText**	Belonging to the attribute in Database, representing privacy policy
open	No	It is the process of users operating the software
click	No	It is the process of users operating the software
fill in	**fill-in()**	As a method in User, fill-in()
validate	No	As a method in User, fill-in()
check	No	A different name for the same function as validate.
agree	No	A different name for the same function as validate.
send	**send()**	A method send() that belongs to system and sends email to the user.
register	**Register**	Classes belonging to the system registration login
Log a Trip	No	A different name for the same function as Create Travel.
main dashboard	**Main Dashboard**	belongs to the homepage of the system and shows the main functions of the system
location	No	An attribute belonging to User
destination	No	Destinations that are part of the trip, one of the attributes of the trip Trip Details
car	No	belongs to the mode of travel and is one of the attributes of the trip Mode_of_Tran
bike	No	belongs to the mode of travel and is one of the attributes of the trip Mode_of_Tran
walk	No	belongs to the mode of travel and is one of the attributes of the trip Mode_of_Tran
public transportation	No	belongs to the mode of travelling and is one of the attributes of the trip Trip Details
distance	No	belongs to the mode of travelling and is one of the attributes of the trip Trip Details

trip details	**Trip Details**	Trip details, which are trip details for each user, are to be used as a class
carbon footprint	No	The CO2 footprint of the trip, which is one of the attributes of Trip Details
total carbon footprint	No	The CO2 footprint of the trip, which is one of the attributes of Trip Details
social media	**Social Media**	A platform for users to share their journeys
select a starting point	**select_starting-point()**	One of the functions belonging to Create Travel, select_starting-point(), allows the user to select the starting point of the trip
calculate	**calculate()**	Methods belonging to the system that count the user's trip information calculate()
estimate	No	Too Abstract
display	**display()**	The method display() that belongs to the system function Create Travel displays the estimated carbon dioxide footprint at the beginning of the recorded trip
confirm	No	Too broad
save	**save()**	The method save() that belongs to the system function Create Travel saves a trip after it has been logged
store	No	A different name for the same function as save
enter	**Input()**	Belongs to the system function Create Travel at the beginning of the logged trip to enter the start and end points of the trip Input method Input()
set up synchronization	No	The method share() belongs to the system function Create Travel to share to Social Media after logging a trip.

share	**share()**	The method share() belongs to the system function Create Travel to share to Social Media after logging a trip.
credential	No	It is the user's credentials for logging into the app, duplicated with account
Record Travel Activity	**Create Travel**	This is the page that records the user's trip data.
Mode of Transportation	**ModeOfTran**	It is the travel mode of the user's trip, one of the attributes recorded on the Record Travel Activity page
Distance Traveled	**Distance_Traveled()**	Methods belonging to Record_Travel for recording the distance travelled by the user
profile	**Profile**	It is a page that displays user's personal information and user's editing personal information.
Additional Information	No	Attributes belonging to Trip Details that can be used to calculate CO2 footprint data.
factor	No	Too broad
option	No	Too Abstract
form	No	Too Abstract
Bicycle	No	belongs to the mode of travel and is one of the attributes of the trip Mode_of_Tran
algorithm	No	Too Abstract
sign up	**sign_up()**	It is a method for users to register a new account, which belongs to User
log into	**login()**	It is the process by which the user logs into the app and belongs to the User's method
select	No	Too Abstract
inputs	**input()**	Methods belonging to Record_Travel that are used by the user to make inputs
recognize	**recognize()**	belonging to the systematic calculation of travelling
associate	No	Too Abstract

update	**update()**	Methods belonging to Record_Travel, where the system automatically updates the user's trip data
returning	No	Too Abstract
repeating	No	Too Abstract
reflect	No	Too Abstract
submit	**submit （）**	Method submit() belonging to Record_Travel, where the user confirms and submits the trip data at the end of the trip
recalculate	**calculate （）**	Methods belonging to Record_Travel that add the new trip data to the previous data
Carbon Footprint Ranking	**CarbonFootprintRanking**	belongs to one of the main functions of the system, calculating the user's CO2 footprint ranking
recorded journey	No	Too Abstract
leader board	No	One of the attributes belonging to Carbon_Footprint_Ranking is the leader board of users
daily	No	One of the attributes belonging to Trip History is the Daily Ranking
weekly	No	One of the attributes belonging to Trip History is the weekly Ranking
monthly metrics	No	One of the attributes belonging to Trip History is the monthly Ranking
retrieves	**retrieves()**	Methods belonging to Carbon_Footprint_Ranking, where the ranking is regularly updated by the system
filter	**filter()**	The method belonging to Carbon_Footprint_Ranking allows users to filter by conditions to see the ranking information
Rewards	**Rewards**	It is the main feature of the system that displays rewards and CO2 footprint points
respective point costs	No	It is the user's CO2 footprint that can be redeemed for rewards
redemption	No	Too Abstract

items	No	Are rewards that are redeemable by the user and are attributes of Reward.
confirmation message	No	Attributes belonging to Rewards, messages sent to users confirming redemption of rewards
desired item	No	Too Abstract
deduct	**deduct()**	It is Rewards' method for calculating the number of points a user has after redeeming a reward
record	No	Same function as deduct()
exchange	No	Too Abstract
Trip History	**Trip History**	Belongs to the main function of the system, recording the history of the user's trips
list of recorded trips	No	Too broad
date	No	Attribute belonging to Trip History, the duration of the trip, inherited from Trip Details
mode of transportation	No	Attribute belonging to Trip History, trip travel mode, inherited from Trip Details
distance	No	Attribute belonging to Trip History, distance travelled, inherited from Trip Details
estimated carbon footprint	No	Too Abstract
click	No	Too Abstract
edit	**edit()**	A method belonging to Trip History that allows the user to edit the history of a trip.
correct	No	Too Abstract
save	No	Too Abstract
Calculate Carbon Footprint	**Calculate Footprint**	It is the main function of the system and is the class that calculates the footprint point
travel activities	No	Too broad
emissions of various transportation	No	It is a property of Calculate_Footprint used to calculate footprint points
historical carbon footprint data	No	It is the user's trip data, repeated

chart	chart()	It is Trip History's method for making the result into a Chart
graph	graph()	It is Trip History's method for making a Graph of the results.
View Historical Date	No	It is one of the main pages of the system, displaying the user's historical trip data, repeated
retrieves	No	Too Abstract
compiles	No	Too Abstract
considering	No	Too Abstract
track	No	Too Abstract

Table 1: Noun-verb Analysis Table

A2.2.Class diagram

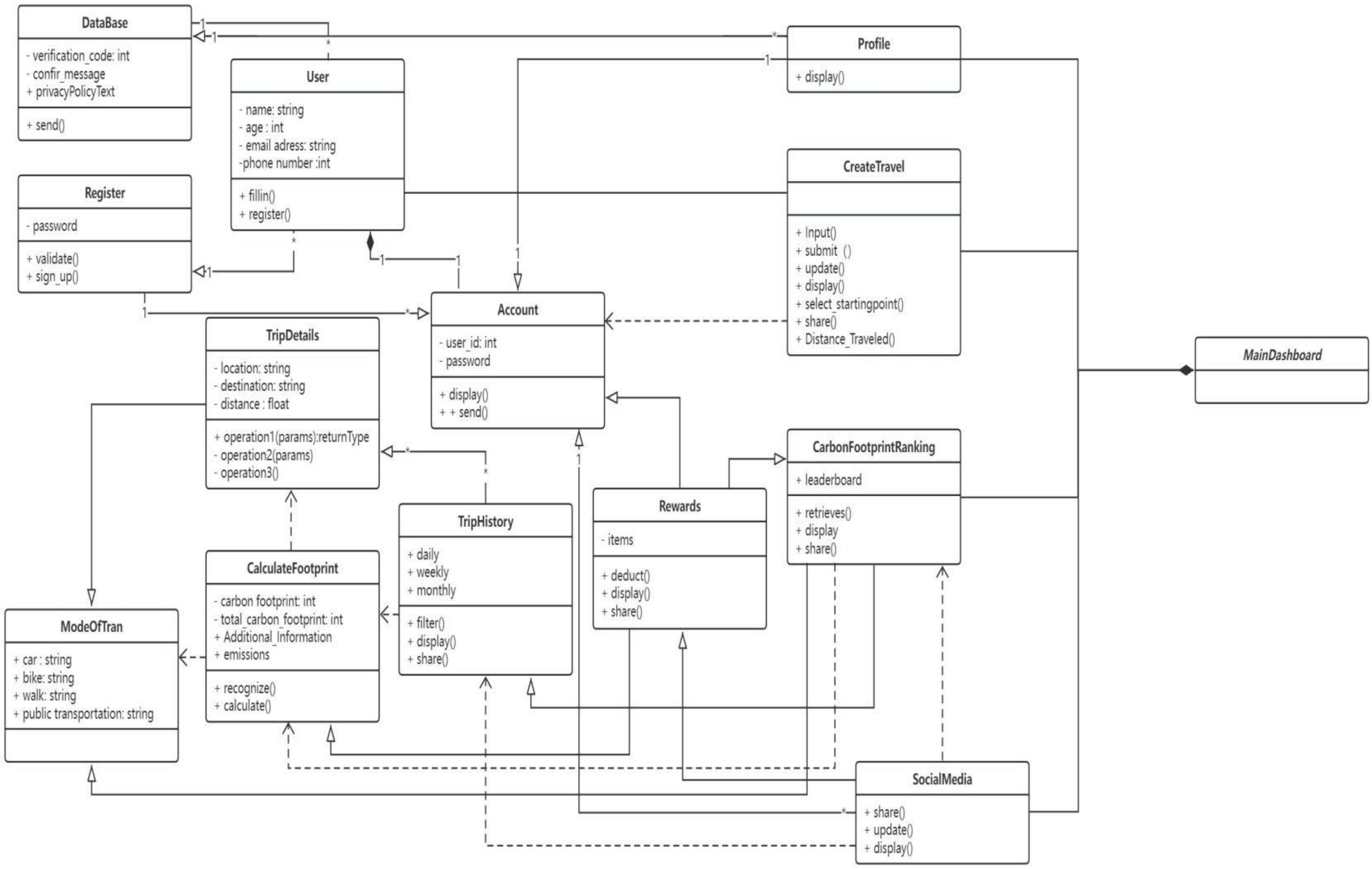

Figure 3.Class diagram

A3. Object diagram

The object diagram illustrates a user-centric scenario within a system designed to track and analyze travel history and environmental impact. The central object, 'John: User', represents a user profile containing personal details such as name, age, email address, and phone number. This user profile is linked to several associated objects, reflecting different aspects of the system's functionality:

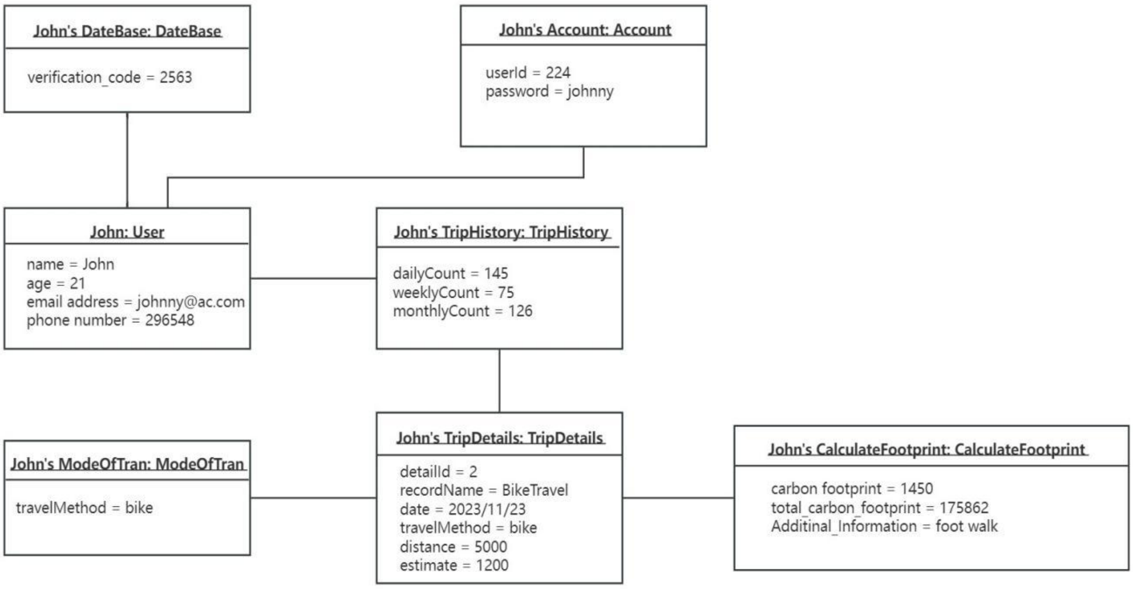

Figure 4.Object diagram

A4. Sequence diagram

A4.1. Record Travel Activity

This is a sequence diagram for the Record Travel Activity scenario. The interaction starts when the user log in or sign up to the system and get access to the dashboard. System checks credentials in the database. Afterwards, the user chooses Record Travel Activity from the dashboard, where the system requires to input relevant information about user’ s travel. Then, the system calculates the footprint based on provided information. Once the calculation is done, the system updates user’ s profile and demonstrates it to the user.

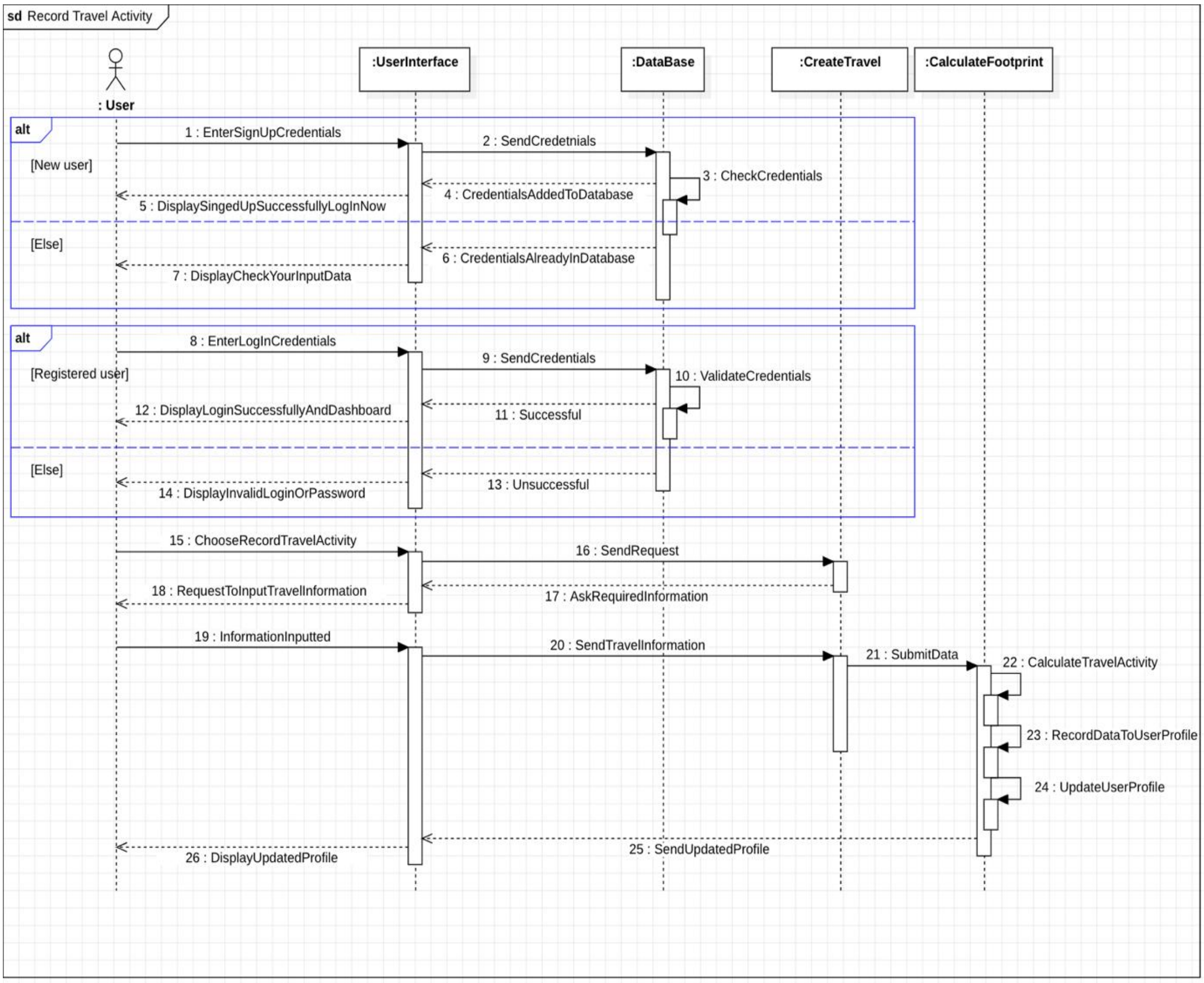
sd Record Travel Activity
: User
:UserInterface
:DataBase
:CreateTravel
:CalculateFootprint
alt
[New user]
1 : EnterSignUpCredentials
2 : SendCredetnials
3 : CheckCredentials
4 : CredentialsAddedToDatabase
5 : DisplaySingedUpSuccessfullyLogInNow
[Else]
6 : CredentialsAlreadyInDatabase
7 : DisplayCheckYourInputData
alt
[Registered user]
8 : EnterLogInCredentials
9 : SendCredentials
10 : ValidateCredentials
11 : Successful
12 : DisplayLoginSuccessfullyAndDashboard
[Else]
13 : Unsuccessful
14 : DisplayInvalidLoginOrPassword
15 : ChooseRecordTravelActivity
16 : SendRequest
17 : AskRequiredInformation
18 : RequestToInputTravelInformation
19 : InformationInputted
20 : SendTravelInformation
21 : SubmitData
22 : CalculateTravelActivity
23 : RecordDataToUserProfile
24 : UpdateUserProfile
25 : SendUpdatedProfile
26 : DisplayUpdatedProfile

Figure 5: Sequence Diagram for Record Travel Activity

A4.2. Calculate Carbon Footprint

This is a sequence diagram for the Calculate Carbon Footprint scenario. The interaction starts when the user get access to the dashboard. Then, the user selects Calculate Carbon Footprint. The system requires to input travel information and relevant period. If the period is wrong, the system will ask to input the correct one. Then, the system calculates the carbon footprint and draws activities chart. Afterwards, the application displays carbon footprint and chart to the user.

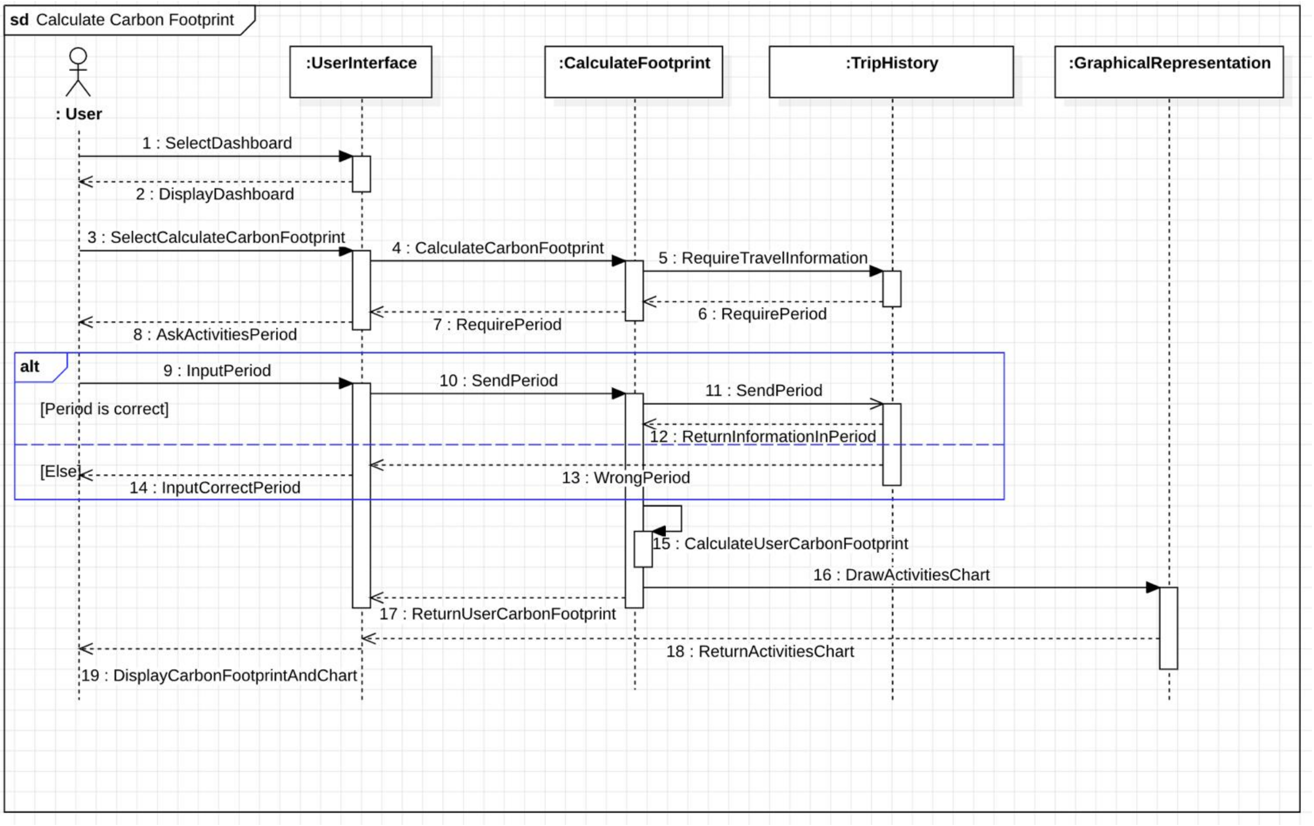

Figure 6: Sequence Diagram for Calculate Carbon Footprint

A5. State diagram

A5.1 User register

This figure is about the status diagram of login app background, the user must first register their personal information, such as name and gender, then click the confirm button, wait for the confirmation information, if the result is correct, judge whether the user information conforms to the rules, if wrong, return to the login interface, if correct, submit the registration form to the database, the submission is successful. The user information is displayed. Registration succeeds.

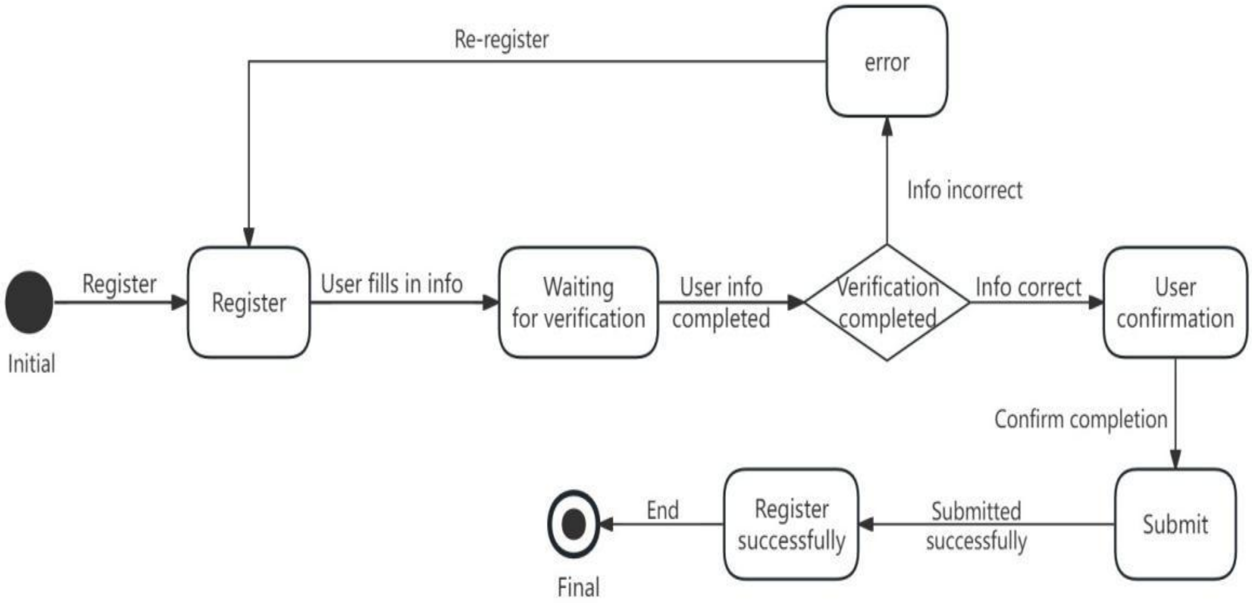

Figure 7: User register

A5.2 The New Travel Creation

This figure is about the activity of the system to record travel records. After successful login, the user chooses the method of recording travel activities and enters the method. After completion, the system judges whether it is the method that has been recorded inside the system, if not, return to fill in again. Personal information update

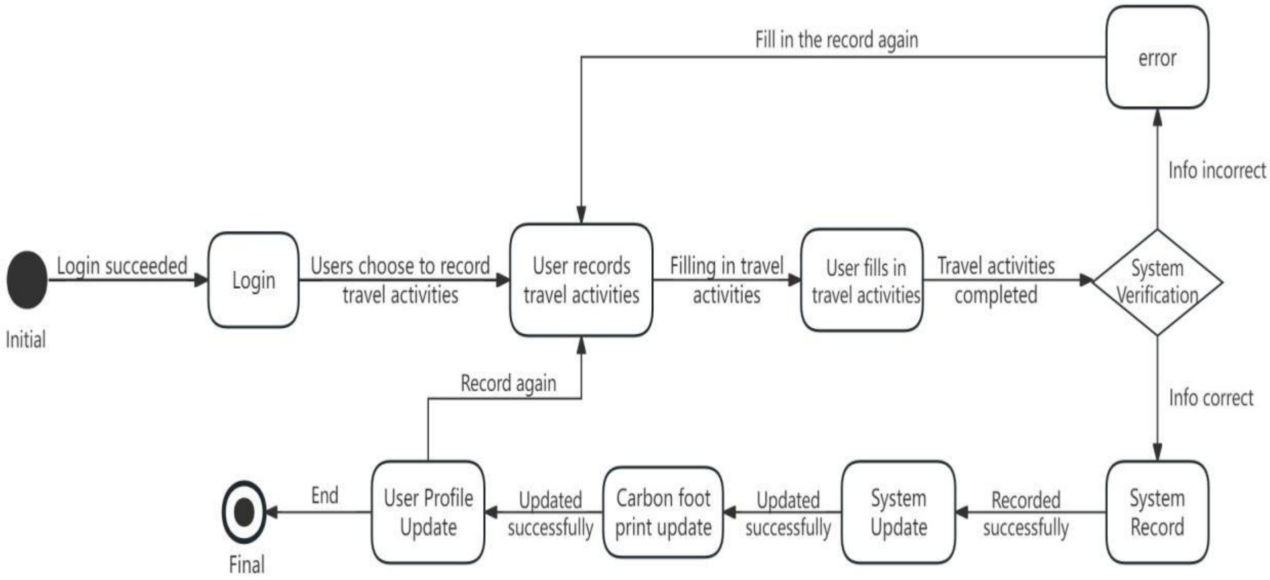

Figure 8: The New Travel Creation

B

B1.Component Diagram

B1.1.Model-View-Controller

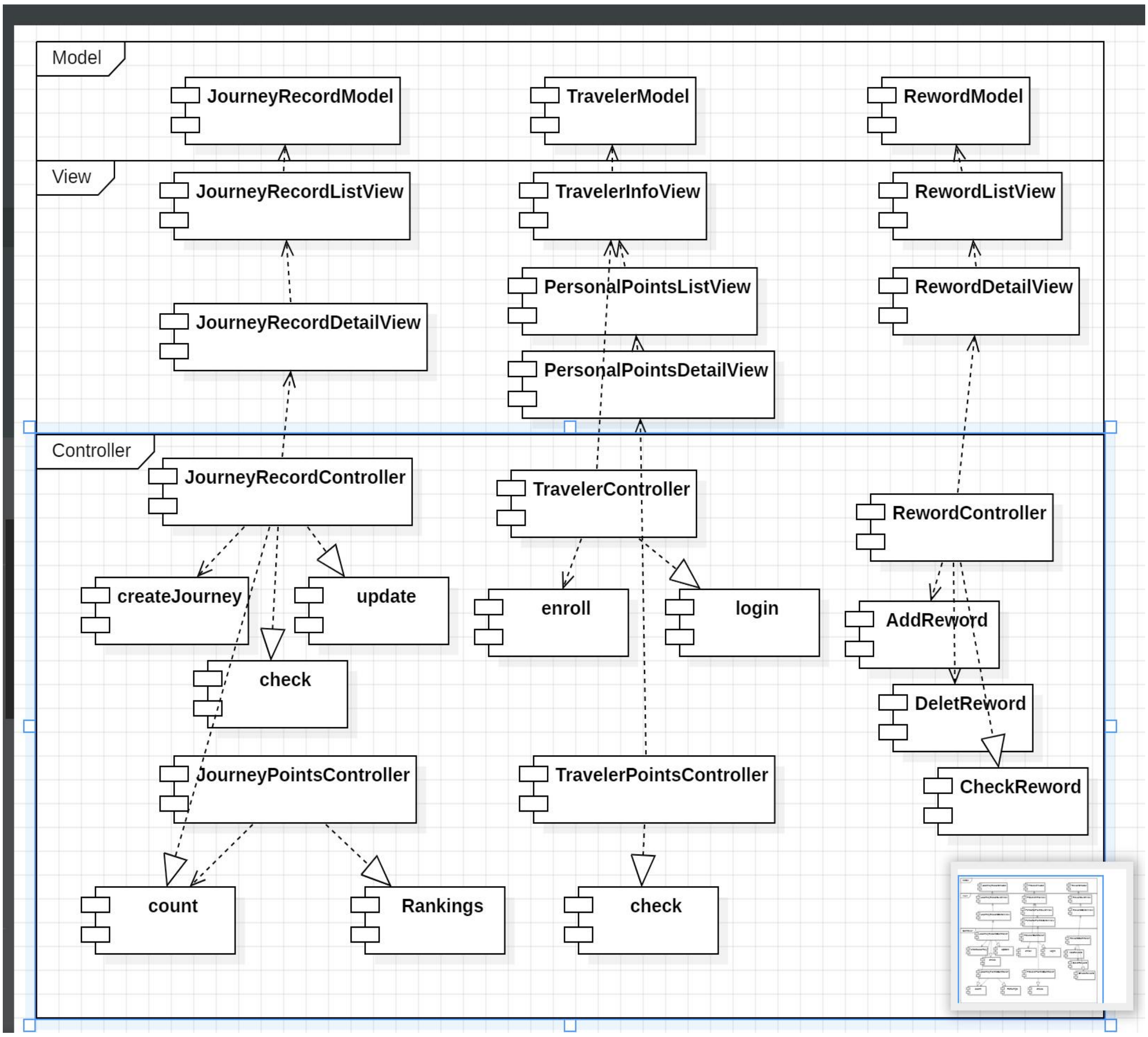

Figure 9: Model-View-Controller

- The model is the core part of the journey recording system, which is responsible for processing the logic related to the data, including data storage, data access, and data processing.
- The view is the user interface part of the journey recording system, responsible for presenting the data to the user.
- The controller is the business logic part of the journey recording system, which handles user requests and calls models and views to process business logic and display data.

B1.2.Client-Server Architecture

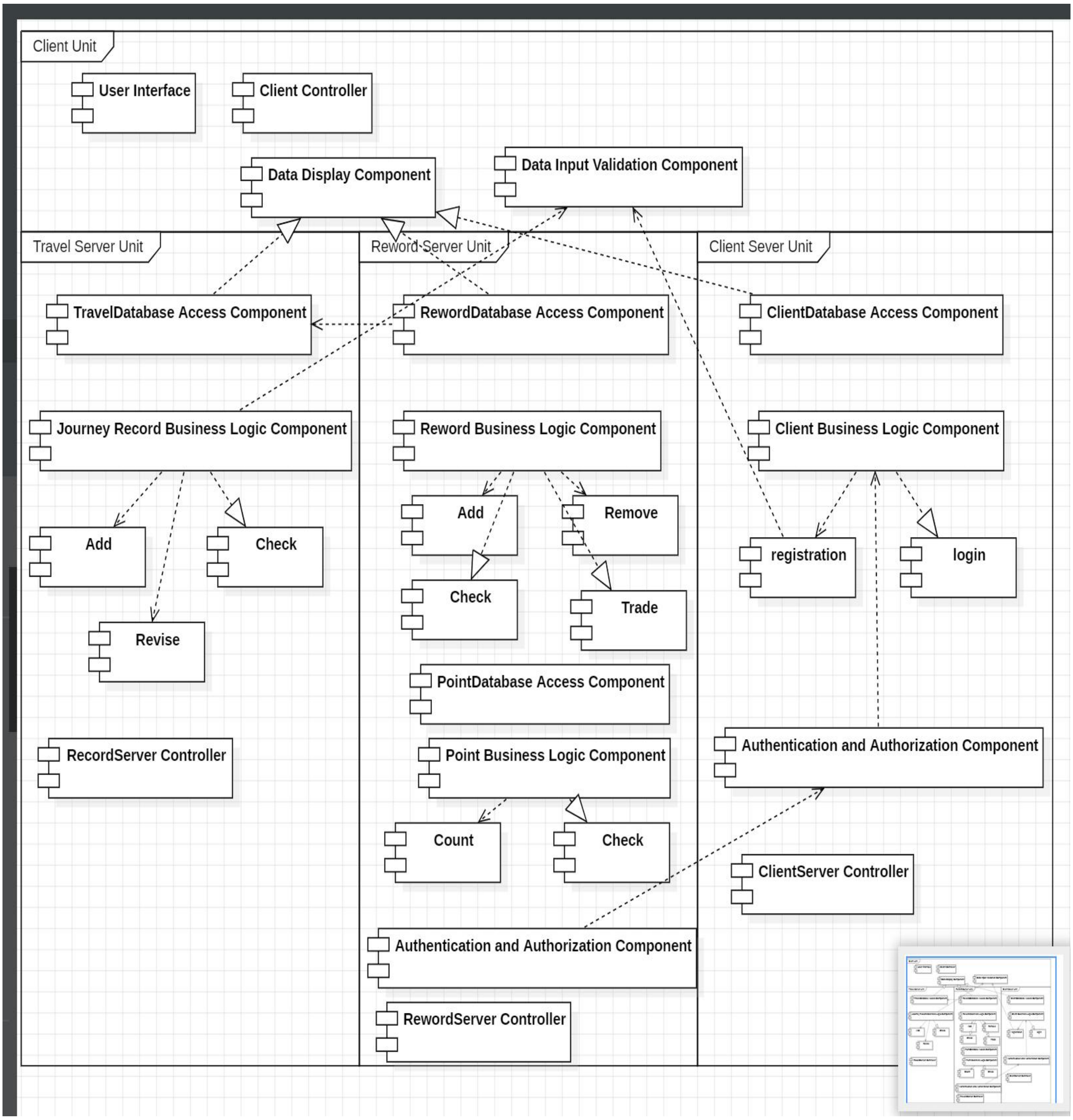
Client Unit
User Interface
Client Controller
Data Display Component
Data Input Validation Component
Travel Server Unit
Reword Server Unit
Client Sever Unit
TravelDatabase Access Component
RewordDatabase Access Component
ClientDatabase Access Component
Journey Record Business Logic Component
Reword Business Logic Component
Client Business Logic Component
Add
Check
Revise
Add
Remove
Check
Trade
registration
login
RecordServer Controller
PointDatabase Access Component
Point Business Logic Component
Count
Check
Authentication and Authorization Component
ClientServer Controller
Authentication and Authorization Component
RewordServer Controller

Figure 10: **Client-Server Architecture**

- The client is the part with which the user interacts directly, and is responsible for providing the user interface, receiving the user's input, and sending requests to the server. Clients typically have a graphical user interface (GUI) to visually access and manipulate data.
- The server is the central component of the system and is responsible for interacting with the database, processing the client requests and executing the corresponding business logic.

B2.Architectural Styles Assessment

In order to decide which **architecture styles between Model-View-Controller and Client-Server Architecture** is appropriate for the project, which can be analyzed among these reasons, includes Real-Time Data Sync and Updates, Handling Large User Base and Data Processing, User Authentication and Data Security, Cross-Platform Compatibility, Scalability and Maintenance, and Performance and Response Time.

- Real-Time Data Sync and Updates

The location of the user and Carbon Footprint need to be updated in time, which requires real-time updates of rankings and travel logs, necessitating timely data synchronization between the server and clients. MVC architecture, primarily focused on separating data, user interface, and business logic. On the opposite, the Client-Server architecture is better equipped to handle such real-time data exchanges and updates.

- Handling Large User Base and Data Processing

The project aims to capture carbon footprints of users, which means dealing with mountains of user-generated data. The Client-Server architecture can effectively manage and process large volumes of data while maintaining good performance.

- User validation and Data Security

User registration and validation are key aspects of the project, requiring absolutely security measures to protect user privacy data. The server side can provide stronger security and encryption measures, which is more suitable for these requirements than the MVC architecture.

- Cross-Platform Compatibility

The project needs to operate on iOS, Android, and windows. As each platform may require different implementations of views and controllers on MVC architecture, which is complexities. Client-Server architecture is more conducive to achieving cross-platform compatibility, as most processing can be done on the server side.

- Scalability and Maintenance

The project aims to provide service for a large number of users. In MVC architecture, updating system could be more complex because changes might need to be implemented across multiple components on different platforms. In contrast, the Client-Server architecture offers better scalability.

- Performance and Response Time

The project aims to show real-time changing user location and information, which needs quick response times. The MVC model may sometimes causes performance bottlenecks, especially if there is heavy client-side processing. However, the server can have powerful computational capabilities in Client-Server architecture, which significantly helps in improving processing speeds and overall performance.

In summary, considering the project's requirements for real-time operation, data processing, data security, cross-platform compatibility, maintenance, and performance, **the Client-Server architecture** is a more appropriate choice.

C

C1: Software Methodology

We will utilize the Scrum framework for our development process which is part of the Agile development approach.

Scrum is an adaptive, iterative, incremental development process that is well suited to managing projects that involve frequent changes and evolving needs.

Here is the rationale behind our choice of Scrum:

- Iterative development and fast delivery:

Scrum prioritizes the achievement of functional product versions through short cycles called sprints. By employing an iterative approach to development, we were able to expedite the delivery of practical functionality, thus allowing consumers to experience and take advantage of the product at an earlier stage.

- Adaptability to transition:

Scrum offers the ability to revise and enhance the product backlog at the conclusion of each sprint..

- User engagement and input:

Scrum places a strong emphasis on active collaboration with stakeholders, allowing users to evaluate and appraise the product during each sprint..

- Collaboration and Openness:

Scrum fosters collaboration and openness among teams through practices that include daily standing-up meetings, sprint reviews and sprint planning.

Our Scrum implementation will involve a 3-week sprint cycle. In addition, we will employ Balkan boards to visually represent tasks and workflows, enabling team members to have a clear understanding of the status and priorities of their work. Common Balkan boards include columns such as To Do, In Progress, Prioritize, Complete and Sprint Length.

Teams commonly establish these parameters through deliberations and consultations. Team members can assess the workload of the assignment by considering their expertise and talents. They can then create a sprint plan that takes into account the time constraints and priorities of the project.

C2. Version Control.

Version control is crucial in software development. There are many options, but the most suitable version control is the Git branching model, which includes multiple repositories and a branching strategy. As a primary source, it is better to maintain a single central repository for our application code-base. Git-flow Workflow will be used as a branching strategy to manage the development life-cycle. It defines branches like “master” for production-ready code, and “develop” for ongoing development. In Git we can use dependency management tools, such as npm (JavaScript) or pip (Python) to handle external libraries and packages. To maintain quality control, it is important to enforce pull requests to review code changes. Git also allows integration of Continuous Integration (CI) and Continuous Deployment (CD) tools, such as Jenkins, Travis CI, or GitHub Actions to automate testing, building, and deployment processes to maintain high code quality and reliability. Our team can use Git tags to note significant releases, and easily identify and mark the development history.

Git version control is a well-structured and organized system, which aligns well with modern software development practices.

D

D1: Test Plan:

1. Experimental endeavor: a software project aimed at documenting carbon footprints.
2. Functional requirements: implementation of user registration and authentication, Creation of a new trip, Management of travel records, Carbon Footprint Ranking and Reward System, Interpersonal communication and cooperative teamwork, Data collection and examination, Dissemination of environmental accomplishments
3. Non-functional requirements:
    - Security: Guarantee the privacy and accuracy of user data.
    - Scalability: managing a high volume of concurrent users
    - Usability: Offers an interface that is both intuitive and user-friendly, resulting in a positive user experience.

- Reliability: Guarantee the steadfastness and dependability of the system.

4. Methods of testing: Unit testing involves doing individual tests on each functional module to verify its appropriate functionality. Integration testing assesses the integration of different functional components and validates the collaborative functionality.

System testing involves doing a thorough examination of the complete system, replicating real-life usage circumstances.

Boundary testing is a highly effective approach for handling complex scenarios. By subjecting input or output values to boundary conditions, one can determine the system's ability to accurately manage such conditions during processing.

For instance, when working with dates, verify if the system can accurately manage leap years or invalid dates.

Exception testing is a highly valuable approach for handling complex scenarios. By conducting tests on the system's response to inaccurate or atypical data, one can determine its ability to effectively manage such scenarios.

When handling user input, it is important to verify if the system can effectively manage invalid or unlawful characters.

Stress testing is a highly valuable approach for addressing complex scenarios.

To evaluate the performance and stability of the system under heavy load, it is possible to simulate a substantial number of users simultaneously utilizing the system.

For instance, while handling a substantial volume of travel records, assess the system's ability to promptly respond and maintain stability.

5. Test cases:

Test Case ID	Test description	Test steps	Test Data	Expected result	Actual result	Pass/ Fail	Test comments
TD-01	Verify the Registration and de-registration	Go to register page Enter valid unregistered User ID Enter valid Password Click Submit	Valid User ID Valid Password	Register Successfully			
		Go to register page Enter valid registered User ID Enter valid Password Click Submit	Valid Registered User ID Valid Password	Failed to registerand prompts that the user is registered			
		Go to register page Enter valid unregistered User ID Enter invalid Password	Valid Unregistered User ID Invalid	Failed to register prompts for an incorrectly formatted pass			

		Click Submit	Password	word		
		Click Exit system	Exit function			
TD-02	Verify the login in and login out	Go to login page Enter valid User ID Enter valid Password Click Submit	Valid User ID Valid Password	Login Successfully		
		Go to login page Enter invalid User ID Enter invalid Password Click Submit	Invalid User ID Invalid Password	Failed to login and received a message indicating that the user name or password as incorrect		
		Enter the wrong password or User ID three times	Invalid User ID Invalid Password	The system will not allow the device to login for the next five minutes and will send a prompt message to the user		
		Click Exit system	Exit function			
TD-03	New trip creation	User login account User clicks on the Record Trip option Choose the destination and mode of trans User saves and confirms System positions and record is of trips	Valid user account Valid destination Valid mode of trans Valid save and confirm Valid position	New Trip Creation Successfully		
		User does not login account User clicks on the Record Trip option	Invalid user accout	Prompts to log in to account		
		User login account User clicks on the Record Trip option No destination and mode of trans selected User saves and	Valid user accout Invalid destination Invalid mode of trans	No location or mode of transportation was selected for the prompt		

		confirms System positions and record is of trips				
		User login account User clicks on the Record Trip option No destination and mode of trans selected User does not saves and confirms System positions and record is of trips	Valid user accout Invalid save Invalid confirm	Prompts need to be saved and submitted		
		User login account User clicks on the Record Trip option No destination and mode of trans selected User does not saves and confirms User does not have location turned on	Valid user accout Invalid position	The system fails to record the trip and indicates that the location is not turned on		
TD-04	Trip records management	User login account User clicks on the Management Trip option User clicks Add Trip Record option User adds trip details User confirms	Valid user accout Valid trip details	Add Trip Successfully		
		User login account User clicks on the Management Trip option User clicks Modify Trip Record option User adds trip details User confirms	Valid user account Valid trip details	Modify Trip Successfully		
		User login account User clicks on the Management Trip option	Valid user account Valid trip	Delete Trip Successfully		

		User clicks Delete Trip Record option User adds trip details User confirms	details				
		User login account User clicks on the Management Trip option User clicks Add/Modify/Delete Trip Record option User adds trip details User confirms	Valid user account Invalid trip details	Failed to Add/Modify/Delete Trip			
TD-05	View Carbon Foot print Rankings	User login account User clicks on the View Carbon Footprint Rankings option Users can filter daily, weekly and monthly rankings Users networking	Valid user account Valid networking	View Carbon Footprint Rankings Successfully			
		User does not login account User clicks on the View Carbon Footprint Rankings option Users can filter daily, weekly and monthly rankings Users networking	Invalid user account Valid networking	Failed to view carbon footprint rankings			
TD-06		User does not login account User clicks on the View Carbon Footprint Rankings option Users can filter daily, weekly and monthly rankings Users does not network	Valid user account Invalid networking	Failed to view carbon footprint rankings			

	Redeem Rewards	User login account User clicks on the Redeem Rewards option Users confirms System confirms the redemption and deducts the points from the user's account	Valid user account Valid user confirm User account points are greater than redemption points	Redeem Rewards Successfully			
		User does not login account User clicks on the Redeem Rewards option Users confirms System confirms the redemption and deducts the points from the user's account	Invalid user account Valid user confirm	Failed to redeem rewards			
		User login account User clicks on the Redeem Rewards option Users confirms System confirms the redemption and deducts the points from the user's account	Valid user account Valid user confirm User account points less than redemption points	Failed to redeem rewards			
		User login account User clicks on the Calculating Carbon Footprint option User selects the period of time for which they wish the activity to be counted Users networking	Valid user account Valid the period of time Valid networking	Calculating Carbon Footprint Successfully			

TD-07	Calculating Carbon Footprint	User does not login account User clicks on the Calculating Carbon Footprint option User selects the period of time for which they wish the activity to be counted Users networking	Invalid user account Valid the period of time Valid networking	Failed to calculating carbon footprint			
		User login account User clicks on the Calculating Carbon Footprint option User selects the period of time for which they wish the activity to be counted Users networking	Valid user account Invalid the trip record Valid networking	Failed to calculating carbon footprint			
		User login account User clicks on the Calculating Carbon Footprint option User does not select the period of time for which they wish the activity t be counted Users networking	Valid user account Invalid the period of time Valid networking	Failed to calculating carbon footprint			
		User login account User clicks on the Calculating Carbon Footprint option User does not select the period of time for which they wish the activity to be counted Users does not network	Valid user account Valid the period of time Invalid networking	Failed to calculating carbon footprint			

Table 2: Test case

D2.Automation

D2.1.Test Cases that Require Automation

- User Registration and Authentication

  User registration and Authentication need to be automatically tested, because, when the software is first released and becomes very popular, people will download and register in droves. What’s more, during specific holidays or travel seasons, there might be a large number of users accessing the app simultaneously, which could lead to server crashes or delayed app responses. Automation testing can simulate scenarios where many users register or use the app at the same time, better testing this functionality of the software. Additionally, there are many combinations of characters that do not meet password format requirements, which can be efficiently tested through automated testing to cover various combinations.

- New Trip Generation

  The creation of new trip involves complex business logic and data processing. Additionally, after creating new trip, complex mathematical calculations are required to convert different trip into corresponding carbon footprint points. There is also the challenge of many users creating trip simultaneously. Automated testing can ensure the stability and reliability of these key functions after each deployment or update.

- Reward System

  The reward system involves complex mathematical calculations. Additionally, some users may accumulate carbon foot-print points over a period before exchanging them for rewards. For manual testing, this would require a very long time to test. Automated testing can help quickly identify errors in algorithms or logic, and the automated testing of these functions is crucial for ensuring the accuracy and consistency of calculations.

D2.2.Test Cases Not Suitable for Automation

- User Feedback and Support

  These tests involve highly personalized and unstructured user feedback, which are difficult to effectively simulate with automated test scripts.

- Email and Notifications

  Although technically feasible, automated testing of email sending functions may become complicated due to frequent triggers of spam filters and similar issues.

E. Software Deployment and Maintenance

- **Maintenance**

The structure of our plan for deploying and maintaining our software project can be outlined as follows:

- Deployment platform:

We plan to deploy the software on a public cloud platform. This approach will allow us to achieve flexibility and scalability.

- (CI/CD) tools:

In order to streamline the process, we will employ Jenkins as our (CI/CD) technology.

Jenkins offers a comprehensive range of functionality that encompasses the automation of the build, test, and deployment processes, as well as a seamless connection to version control systems.

- **Strategy for Maintenance:**

- Periodic Backups:

We will systematically create copies of data and configurations at regular intervals to mitigate the risk of unintentional data loss and ensure prompt system recovery.

- Security Updates:

We regularly upgrade the security of our systems by applying patches to operating systems, databases and applications to protect against known vulnerabilities.

- Monitoring and Alerts:

We will set up a monitoring system to continuously observe the performance and accessibility of the system in real-time. In addition, we will configure alerts to promptly address any irregularities.

- Contingency Plan:

We will implement a comprehensive contingency plan to ensure prompt service restoration and minimal downtime in the event of a system failure or disaster.

- Log Management:

We will establish a log management plan to document significant occurrences and anomalies for the purpose of resolving issues and enhancing system performance.

To ensure high availability, security and maintainability of our software projects, we will utilize a public cloud platform, CI/CD technology and maintenance procedures.. This will build a solid, dependable software ecosystem capable of promptly addressing any concerns or requirements.

F. Software ethics.

Green Travel application is a multipurpose system, which follows ethical and responsible design based on the IEEE/ACM Software Engineering Code of Ethics. The IEEE/ACM Software Engineering Code of Ethics prioritizes the well-being of users, data privacy, and environmental sustainability.

Here's how the system adheres to these principles:

1. **User Privacy and Data Protection:**

   The system ensures the confidentiality of user data privacy by implementing solid security measures, including encryption and secure authentication. User sign up data and sensitive information, such as location data and carbon footprint calculations are handled with care. This information is stored on user devices securely. This approach complies with IEEE/ACM guidelines to protect user privacy and data confidentiality.

2. **Transparency and Accountability:**

   The system promotes transparency. It provides clear information on how users' data is collected, processed, and used. Clients have full control over their data and can update their own preferences at any time. This aligns with the ethical principle of providing users with full disclosure and control over their information.

3. **Environmental Sustainability:**

   The system's primary goal is to encourage low-carbon travel and reduce carbon emissions. It motivates users to use public transportation and adopt sustainable travel habits. So, the system contributes to environmental sustainability. This aligns with ethical considerations related to environmental sustainability as it supports the reduction of greenhouse gas emissions.

4. **User-friendly Design:**

   The system is designed with a user-friendly approach, focusing on simplicity, usability, and accessibility. The intuitive interface is developed to be accessible to users regardless of their technical expertise, adhering to ethical principles that emphasize technology serving the needs of people.

5. **Fairness and Inclusivity:**

The system's automatic identification of travel modes ensures fairness in tracking carbon footprints, preventing data errors. It ensures equal tracking of carbon footprints for all users. This aligns with ethical considerations of fairness and inclusivity in technology.

6. **Regular Maintenance and Testing:**

Regular maintenance and testing are conducted to maintain data accuracy and system effectiveness. This approach adheres to ethical guidelines for ensuring the reliability and integrity of a software system.

In summary, the proposed system exemplifies ethical and responsible design principles as outlined in the IEEE/ACM Software Engineering Code of Ethics. It prioritizes user privacy, transparency, environmental sustainability, fairness, and user-friendly design. Green Travel application contributes positively to society by promoting sustainable travel practices while adhering to ethical standards in software engineering.

G. Remarks:

Yingyi LI	yxl1736@student.bham.ac.uk
Changhao Lin	cxl1067@student.bham.ac.uk
Hongyu Yang	hxy396@student.bham.ac.uk
Damyl	dxb394@student.bham.ac.uk
Jinhua Dai	Jxd555@student.bham.ac.uk

Members/Tasks	A1	A2	A3	A4	A5	B1	B2	C1	C2	D1	D2	E	F	G	Management
Yingyi LI	5					5									5
Changhao Lin		5					5				5				5
Hongyu Yang			5					5				5			5
Damyl				5					5				5		5
Jinhua Dai					5					5				5	5

Table 3: Team member's contribution

Task sorting and error correction：Yingyi Li