San Francisco State University

SWE CSC 648

Section-04 Team-05

Milestone 2
updated

05/16/2023

Siddharth Sankar (Team lead)

Muhammed Nafees (Front-end lead)

Hanlei Liu

Dania Dababo (Scrum)

Farah Jamjoum

Yahya Obeid (Github master)

<u>1. Data Definitions V2</u>

User Account (Username):
This is the unique identifier that the user selects to login to their account, usually in the form of an email address.

1. Password:

This is a confidential code that the user selects to secure their account and protect their personal information. The password should be complex and not easily guessable by others.

2. Registration:

This is the process by which an unregistered user becomes a registered user of the TravelMate website. The registration page will typically ask for personal information, such as name, email address, and password, and may also require verification of the user's email address.

3. Registered User:

This is a user who has completed the registration process and has a TravelMate account. Registered users can access all the features of the website, including generating travel itineraries and saving destinations.

4. Unregistered User:

This is a user who has not completed the registration process and does not have a TravelMate account. Unregistered users can still use some features of the website, such as the search function, but they will not be able to generate travel itineraries or save destinations.

5. Destination:

This refers to a place that the user is interested in travelling to and requires assistance in generating a travel plan for. Examples of destinations might include cities, national parks, or popular tourist attractions.

6. Travel Itinerary:

This is a page that is generated by the TravelMate API after the user inputs their travel preferences, such as the duration of their trip, their budget, and their interests. The travel itinerary includes a detailed daily plan for the set duration, including information on where to go, what to do, and where to eat.

7. Day Plan:

This is a component of the travel itinerary that contains information on what activities the user should do on a particular day. The day plan may include recommendations for sightseeing, outdoor activities, or cultural experiences.

8. Description:

This is a detailed plan for each day of the travel itinerary that provides the user with information about what activities they will be doing, such as what to pack, the price of certain activities or restaurants, and other relevant information to make their trip planning easier.

~~9. Map:~~

~~This is a map that shows all the places included in the travel itinerary together for a structured plan for navigation. The map may include pins or markers indicating where the user should go and what they should see.~~

10. User Profile:

This is a page that contains the user's information, such as their name, email address, and saved destinations. The user profile may also include the user's travel preferences and past travel history.

11. Saved Destinations:

This is a page that contains the destinations that are saved in the database by the user for future reference. The user can save destinations they are interested in traveling to and refer to them later when planning future trips.

12. Search Function:

This is a function that allows users to search for popular destinations on the TravelMate website. The search function may allow users to filter destinations by category, such as cities, beaches, or outdoor activities.

13. Popular Destinations:

This is a page that contains the most searched destinations by the userbase on the TravelMate website. The popular destinations may be updated regularly based on user search trends and interests.

14. User Privileges:

User privileges describe the level of access and permissions that registered users have on the TravelMate website. Some examples of user privileges may include:

- The ability to generate travel itineraries
- The ability to save destinations
- The ability to update user profile information

The ability to delete user account information

- User privileges need to be defined carefully to ensure that users have access to the appropriate features and data based on their needs and level of access.

15. Raw Data:

Raw data refers to the original, unprocessed data that is collected by the TravelMate website. Examples of raw data may include user input data, such as travel preferences, travel itinerary data, and user profile data. Raw data needs to be carefully collected, validated, and processed to ensure its accuracy and completeness.

16. Metadata

Refers to the information that describes the characteristics of the raw data. Examples of metadata may include the date and time the data was collected, the source of the data, and the format of the data. Metadata helps to provide context and meaning to the raw data.

17. Supporting Data:

Supporting data refers to any additional data or information that is needed to support the raw data and provide further insights into user behavior and preferences. Examples of supporting data may include user feedback, customer reviews, and data from third-party sources such as travel websites or social media platforms. Supporting data needs to be carefully analyzed and integrated with the raw data and metadata to provide a more complete picture of user behavior and preferences.

2. Functional Requirements V2

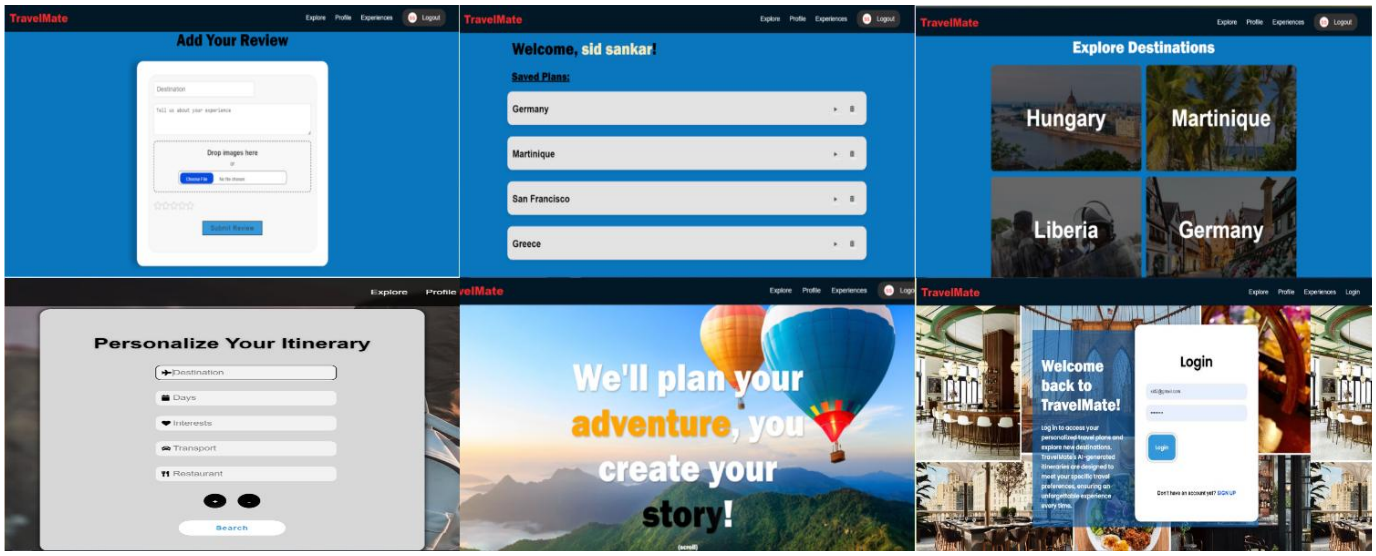
Registered Users should be able to…

1. Create an account with a username and password (Priority - HIGH)
    a. Use email id as username
    b. Login
    c. Logout
2. Enter a prompt about their idea for traveling in order to generate the plan (Priority - HIGH)
    a. Destination
    b. Duration (start to end)
    c. Interests
    d. Preference on Transport
    e. Preference on Resturant
3. See the generated plan in detail (Priority - MEDIUM)
    a. The plan for each day
    b. The plan for each day will include the place, time and description
    ~~c. Map for the routes~~
4. Browse their own personal “Trips” gallery (Priority - LOW)
    a. Create new Trips from a list of Cities/Countries
    b. Delete old Trips
    c. Edit the plan for individual days
5. Save their generated trips to the database for future reference (Priority - HIGH)
6. Read the reviews of other users for their desired destinations (Priority - LOW)
    a. Read reviews
    b. Comment on reviews
    c. Post their own review about the place

7. See most popular destinations Unregistered users must be able to…

    1. Enter a prompt about their idea for traveling in order to generate the plan (Priority - HIGH)
        a. Destination
        b. Total number of travelers
        c. Duration (start to end)
    2. See the generated plan in detail
        a. The plan for each day
        b. The plan for each day will include the place, time,weather and description
        c. Map for the routes
    3. See most popular destination

3. UI Mockups and UX Flows



Log in to access your personalized travel plans and explore new destinations. TravelMate's AI-generated itineraries are designed to meet your specific travel preferences, ensuring an unforgettable experience every time.

4. High level Architecture, Database Organization

DB Organization:

The main database schema is organized into two main tables (collections in MongoDB): "User" and "Destination."

User collection:

username: (String, required, unique) The username of the user.

password: (String, required) The hashed password of the user.

savedPlans: (Array of ObjectIds, references Destination) An array of plans saved by the user.

Destination collection:

title: (String, required) The title of the destination.

description: (String, required) The description of the destination.

dayPlans: (Array of Strings) An array containing the day plans for the destination.

Add/Delete/Search architecture:

API’s used to communicate from the frontend to extract data from the database-

Destination:

Add: Creating a new destination using a POST request.

Search: Searching for destinations based on a query using a GET request.

Delete: Deleting a destination using a DELETE request.

Display: Displaying all destinations or the destinations that match a query using a GET request.

User:

Add: Registering a new user using a POST request.

Display: Displaying the user's data and plans using GET requests.

APIs:

User APIs:

POST /api/users/register: Register a new user.

POST /api/users/login: Log in a user.

GET /api/users: Get the user's data.

GET /api/users/plans: Get the user's plans.

POST /api/users/plans: Save a user's plan.

Destination APIs:

GET /api/destinations: Get all destinations or search for destinations based on a query.

GET /api/destinations/:id: Get a destination by ID.

POST /api/destinations: Create a new destination.

PUT /api/destinations/:id: Update a destination.

DELETE /api/destinations/:id: Delete a destination.

HTTP requests and responses are defined across the backend & frontend using Express.js and Axios.

Backend route functions needed to implement:

User route functions: register, login, getUserData, getUserPlans, saveUserPlan.

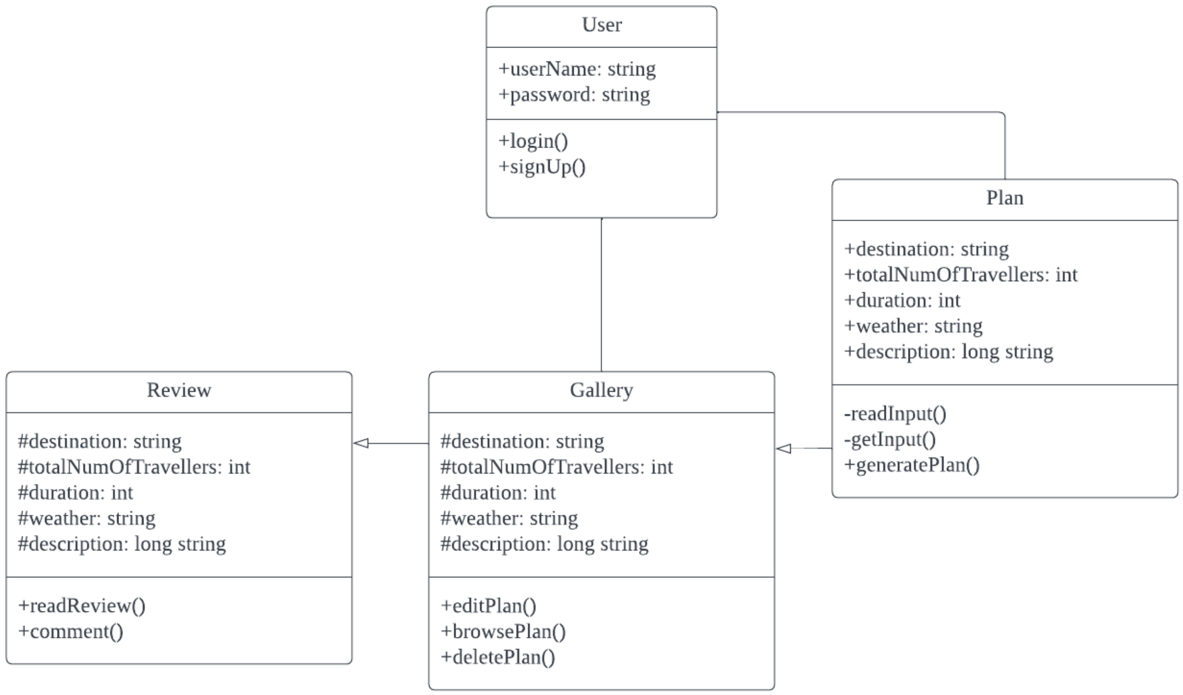
Destination route functions: getDestinations, getDestinationById, createDestination, updateDestination, deleteDestination.

No third-party API, open-source components, or software tools and frameworks changes have been used in the our code for the travelmate vertical prototype.

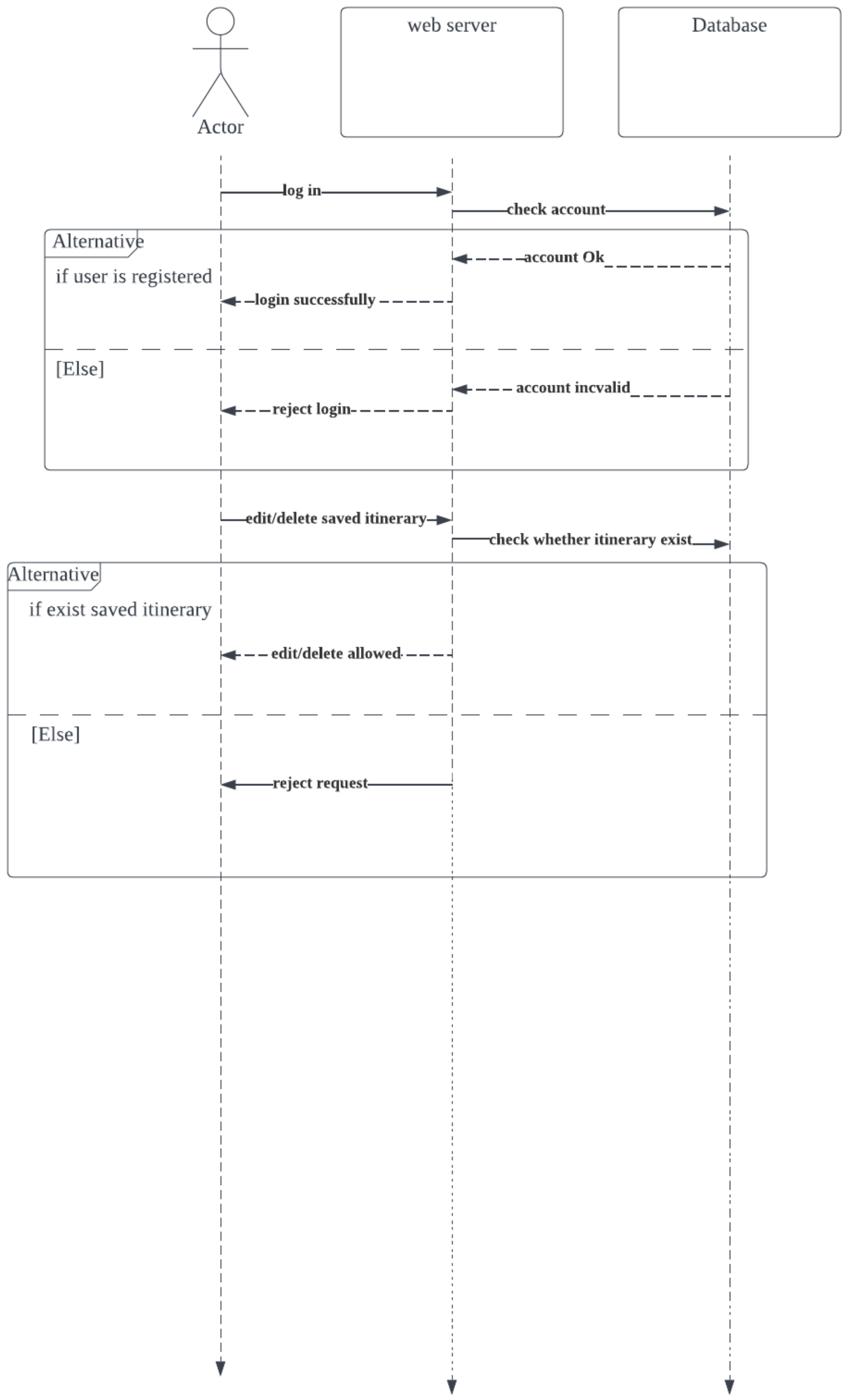
5. High Level UML Diagrams

Familiarize yourself with Unified Modeling Language (UML).
For Milestone 2, provide:

> *High-level UML class diagrams* for implementation classes of core functionality, Focus on main high-level classes only (one or at most two levels deep). This must reflect an OO approach to implementing your web site. For UML, you could find many references including http://edn.embarcadero.com/article/31863.



High-level sequence diagrams: for ~5 major user stories, please develop UML sequence diagram.


Actor
web server
Database
log in
check account
Alternative
if user is registered
account Ok
login successfully
[Else]
account incvalid
reject login
edit/delete saved itinerary
check whether itinerary exist
Alternative
if exist saved itinerary
edit/delete allowed
[Else]
reject request

6 .Identify *actual* key risks for your project at this time

Skills Risks:
Risk: Some team members may lack the necessary skills to complete assigned tasks.

Mitigation: Provide necessary training and resources to team members to improve their skills and encourage cross-functional collaboration.

Schedule Risks:
Risk: The project may take longer than expected to complete, leading to missed deadlines.

Mitigation: Develop a realistic schedule and allow some buffer time for unexpected delays. Regularly review and adjust the schedule as needed, and communicate any changes to team members.

Teamwork Risks:
Risk: Poor communication and collaboration between team members may result in conflicts and delays.

Mitigation: Encourage open communication and regular team meetings to discuss progress and any issues. Assign specific roles and responsibilities to team members to ensure clarity and accountability.

Legal/Content Risks:
Risk: There may be legal issues related to obtaining necessary content and software with proper licensing and copyright

Mitigation: Conduct thorough research on licensing and copyright requirements before using any content or software. Obtain proper licenses and permissions as needed and ensure compliance with all legal requirements.

To address these risks, we will regularly review and update our risk management plan as needed. We will communicate any updates to all team members and ensure that everyone is aware of their roles and responsibilities in mitigating these risks. We will also monitor project progress regularly and take proactive measures to prevent or resolve any issues that arise. Finally, we will ensure that all team members are working together effectively by encouraging open communication and collaboration.

7. Project management

Describe in no more than half a page how your team managed M2 tasks including

- How is each member's progress being shared in scrum meetings? Is it transparently shared?
- Do you have a tool to manage each member's task, such as Notion, Jira? Strongly recommend such tools.

In order to manage tasks effectively, we held regular meetings to discuss upcoming tasks and review completed ones. During these meetings, we would identify any roadblocks or challenges and work together to find solutions. Additionally, we emphasized clear communication and regular check-ins with each other to ensure that everyone was on the same page and aware of any changes to the project. Overall, this combination of software, meetings, and communication helped us stay organized and ensure that tasks were completed on time and to the best of our ability.
At this time, we have not utilized any project management tools to manage our tasks, although it may be helpful to allow us to assign tasks to specific team members, set deadlines, and track progress.

Submission of Milestone 2 *Document* for Review

Formatting instructions for M2 document must be followed precisely, as outlined below. Submission to github M2 folder must be done by the deadline specified; any extension has to be approved ahead of time.

In creating, editing and finalizing Milestone 2 document follow similar team process as outlined for Milestone 1 document

The whole student team submits one milestone document for Milestones 2, as follows (same as M1 submission): Team leads will put the attached file into M2 folder of github.

- First page of Milestone 2 document must include

–"SW Engineering CSC648/848 Section X"
    Project/application title and name (you can use the code name of your choice for your application)
–Team number and name – make it clearly displayed for easy reference
–Names of students (team lead first) -
    Name of all members with his/her role
–"Milestone 2"
–Date
–History table

- The rest of the document must have numbered sections outlined above in "Content and structure for Milestone 2 document for review by institutors". Each section must start on a new page

Vertical SW Prototype

In addition to the Milestone 2 document, the team will create a "vertical SW prototype" to test the infrastructure and chosen frameworks and to jumpstart the coding effort. The vertical prototype is the code that exercises full deployment stack from browser (with simple *test home page*), via middleware, to/from DB, using only your chosen and approved frameworks and SW components. It has to be deployed on your chosen deployment server, the same way the final product will be deployed. The purpose of the vertical prototype is to early and quickly test basic SW components and deployment infrastructure and frameworks as well as the key architecture patterns and thus to serve as a basic "scaffolding" for the final product. It also serves as a "teaching and training" tool to bring the rest of the team up to speed on SW, frameworks etc.

We recommend that back-end team be assigned the task of constructing this vertical prototype, with front end team helping with the front WWW page.

Vertical prototype shall allow one to list your own defined data on *test home page* (simple home page used to test vertical prototype), then get a response from the DB and display the proper content. Every output should be retrieved from the DB.

- UI for *the test home page* can be a simple one and does not have to conform to your UX storyboard yet. Your test home page should provide following operations :
  - #1 : list main data items in your application. For an example of the Resume Review application, list following
    - Resumes with proper attributes
    - Resume Review Instruction with attributes
    - Posted jobs with proper attributes
  - #2 : search
    - Given the DB is populated with the above data items,
    - Search resume with proper search filter (e.g., Name, skills, school name)
    - Show the properties in the test home page
- The DB can have only a few items. The items in the DB shall be encoded with full schema as it is defined by now in M2 document

Vertical prototype serves also to help the rest of the team get "on the same page" in terms of SW development*:* Back end team should also document vertical prototype code well and use it to educate the rest of the team on how to develop the rest of the product. Front end team can make test home page to establish rules for UI development. Back end and front end teams should also agree on common way to connect UI with back end and document it for all.
Vertical SW Prototype delivery/submission
Your team will submit vertical prototype similarly as M0, via e-mail to class TA and the instructor by the deadline. The submission format and process must be followed precisely, as always. Submission must be done by the deadline specified.
e-mail subject line: Must be "CSC648-848 Milestone2 Vertical Prototype Section X Team N" in the subject line (N is a team number).
e-mail body contains:

- some courtesy text explaining what the e-mail is about
- link to vertical prototype home test page so it can be run and tested by TA and the instructor
- link to githiub for vertical prototype

Vertical prototype shall be evaluated by class TA/instructor based on:

- Functionality and correct display data results (be sure to test before sending it)

- Code organization and architecture
- Proper use of frameworks
- Correct deployment on a chosen team server for final delivery