SC2006: Software Engineering

Lab 4 Deliverables

WanderMap

Ang Yvette (U2220779L)
Chua Sym Eyan (U2221430B)
Tan Hui Ling (U2222280A)
Dan He (N2304782D)

Content Page

Introduction

Wandermap is a web application that facilitates planning of travel itineraries around different countries. Wandermap is a one stop platform for users to plan itineraries, review itineraries and get inspiration from others’ itineraries for their travels.

Working Application Prototype

Our working application prototype will be demonstrated live.

Source code

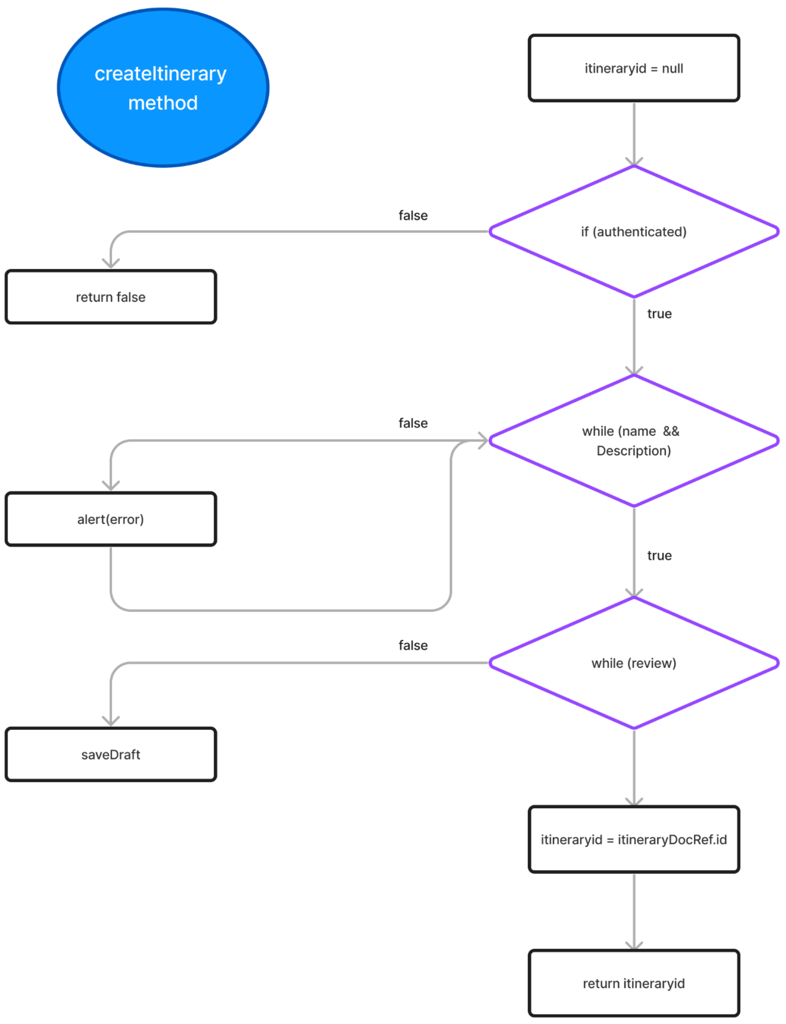
Our source code can be found at:
https://github.com/ey4n/wandermap-final

Test Cases and Testing Results

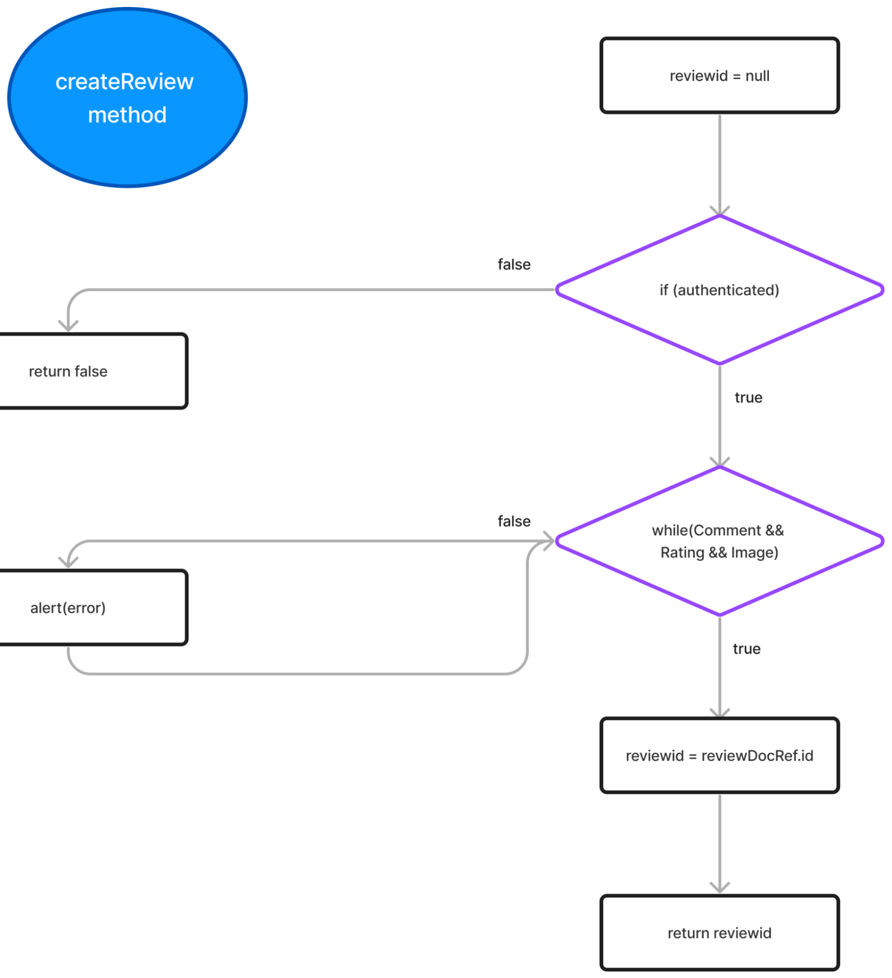
Test Cases

Basis Path Testing

createItinerary



createReview



Equivalence class and boundary value testing

Create itinerary feature

Title Control Class

Valid Equivalence Class	Invalid Equivalence Class
{Any string that is between 1 - 50 characters}	{Empty strings or strings with more then 50 characters}

Description Control Class

Valid Equivalence Class	Invalid Equivalence Class
{Any string that is between 1 - 250 characters}	{Empty strings or strings with more then 250 characters}

Attraction Control Class

Valid Equivalence Class	Invalid Equivalence Class
{Any selection of 1 or more attractions provided in the list of attractions}	{A selection of 0 attractions}

Review itinerary feature

Rating Control Class

Valid Equivalence Class	Invalid Equivalence Class
1 2 3 4 5	{Any real number that is not an integer between 1 and 5 (inclusive)}

Review Control Class

Valid Equivalence Class	Invalid Equivalence Class
{Any string that is between 1 - 250 characters}	{Empty strings or strings with more then 250 characters}

Image Control Class

Valid Equivalence Class	Invalid Equivalence Class
{Any image file in the form of a png or jpg}	{Any files not in the form of a png or jpg}

Test Results

Equivalence class and boundary value testing

Create itinerary feature

Title	Description	Attraction	Expected Output	Actual Output
‘’	‘test description’	‘Singapore Zoo’, ‘Merlion’	Invalid	Invalid
‘Lorem ipsum dolor sit…’ (over 50 characters)	‘test description’	‘Singapore Zoo’, ‘Merlion’	Invalid	Invalid
‘test title’	‘’	‘Singapore Zoo’, ‘Merlion’	Invalid	Invalid
‘test title’	‘Lorem ipsum dolor sit…’ (over 250 characters)	‘Singapore Zoo’, ‘Merlion’	Invalid	Invalid
‘test title’	‘test description’	‘’	Invalid	Invalid
‘test title’	‘test description’	‘My House’	Invalid	Invalid
‘test title’	‘test description’	‘Singapore Zoo’, ‘Merlion’	Accepted	Accepted

Review itinerary feature

Comment	Rating	Image	Expected Output	Actual Output
‘test comment’	6	png file	Invalid	Invalid
‘test comment’	2.5	png file	Invalid	Invalid
‘’	4	png file	Invalid	Invalid
‘Lorem ipsum dolor sit…’ (over 250 characters)	4	png file	Invalid	Invalid
‘test comment’	4	pdf file	Invalid	Invalid
‘test comment’	4	png file	Accepted	Accepted

<u>Demo Script</u>

<Slide >
Good afternoon everyone! We are Team One, and today we are here to present our web application Wandermap.

<Slide >
Our team was inspired by one of our friends on exchange. He kept complaining that it was so difficult to find attractions in Singapore, and that it is hard to plan out itineraries to travel around Singapore. As such, we decided to help our friend by creating Wandermap.

<Slide >
Wandermap is a crowdsourced travel itinerary planner offering a wide range of user generated itineraries. It helps make travel in Singapore easy and fuss free with its built in routes generator created through Google Maps API. It is also highly customisable, allowing users to pick and choose their ideal itinerary.

<Slide >
For users of Wandermap, we anticipate that we can split them into 2 categories. Users looking for ideas on travel itineraries, and users who want to submit their own travel itineraries for others to view.

While planning our web application, we especially focused on the needs of these 2 groups to decide on our use cases.

<click>
For users looking for ideas on travel itineraries, they are able to browse all published itineraries on the website. After choosing an itinerary, these users would also be able to leave reviews for other users to see, provided that they had created an account.

For users who want to submit their own travel itineraries, they are able to create their itineraries and either publish them immediately, or save them as drafts. Drafts can be edited before they are published, while published itineraries cannot be edited. Users will also be given the option to delete their itineraries if they become outdated, or are otherwise unhappy with it.

<Slide >
Now with our use cases in mind, we can see how they come together in our web application through our use case diagram.

Our use case diagram has 3 main actors: the user, google maps and prisma database.

<click>
For users, the user accounts must be authenticated through login to access all the functionalities.

(click)
For itinerary viewers, they can browse and view all itineraries without logging in. When they view routes in each individual itinerary, our application will call the Google Maps Places and Routes API to return a route to different attractions.

(click)
For itinerary creators, they will be required to login to create an itinerary. The creator will also be able to browse his created itineraries, edit and or delete them. Logged in users will also be able to review itineraries. These changes will be updated in the database.

<Live Demo>
We will now move on to our live demo portion.

Upon entering the website url, users are brought to a home page where they are prompted to either ‘Login’ or to ‘Browse All Itineraries’.

Users who just want to get travel itinerary ideas can go straight to the ‘Browse All Itineraries’ page without having to log in. From here, they can see all the currently published itineraries in our database. To help users find the itinerary that suits their taste the most, they are given the option to filter the published itineraries based on these 4 filters: categories that catch their interest, lets say Nature, highest upvotes, most recent posts, and cheapest options.

Once the user finds an itinerary they like they can click the ‘Show Details’ button, which would redirect them to the itinerary information page. From here they can view the full description of the itinerary, pictures associated with the itinerary, reviews left by other users, budget of the itinerary, and the order of attractions visited. If the user further wants to view the route between each attraction, they can click the ‘View Route to next attraction’. From here they can see a google maps route between the two attractions. They can then choose if they wish to drive or to take public transport, and then view a list of instructions on how to get from one attraction to another.

Now if a user wants to use the other functions of our webpage, they will be required to log in. They can do this through the navbar on the side of the web page, whereby they will be prompted to sign in through Google's O-Auth. After logging in, users will be greeted by a user home page. They will be able to access all functionalities through the nav bar.

Let's go back to the ‘Browse All Itineraries’ page. The user can choose an itinerary to review by clicking the ‘Review’ button, which would redirect them to the review page. Here they are prompted to write a comment of no more than 250 characters, select a rating from 1-5, and upload an image. Upon a successful review, they would be shown a confirmation page and then be redirected back to the home page. The user can then click back into the itinerary they reviewed to confirm that their review has been published.

Now, let's go back to the ‘Browse All Itineraries’ page again. As you can see, now that the user is logged in, they are given the option to bookmark or upvote an itinerary through the icons attached to each itinerary card. By clicking the upvote icon, we can see the upvote count of that itinerary going up, and the icon changing color. Next we can try bookmarking a few itineraries, also through clicking the icon. Users can now navigate to the bookmarked itineraries page through the navbar. At this point the user should see the itineraries they bookmarked appear here.

Now, let's say the user is sufficiently inspired, and wishes to make their own itinerary. Lets navigate to the create itineraries page through the navbar. Here, users are prompted to fill in the title of their itinerary which has a character limit of 50, and a description of their itinerary which has a character limit of 250. As you can see, if users try to write a title that is more than 50 characters, they will be unable to do so. They will also be prompted to select the attractions they wish to visit, and the order they wish to visit them in.

Now lets fill in this itinerary. We can select a filter to narrow down the list of attractions by selecting a few categories. Lets select Culture and Scenic. Now we can select the attractions that interest us. Lets choose Marina Bay Sands, Merlion and Gardens By the Bay. Then we can slide the attractions into the order we want to visit them in. Finally, let's fill in the title and description.

Users now have a choice to submit their itinerary or to save it as a draft. Lets try submitting first. Users will be automatically redirected to the review page of their newly created itinerary so that they may leave a review for their itinerary. This review page includes a comments section, a star rating, and also an option to upload images. This image they submit will be displayed in the browse page after posting. Lets click submit. After this, their itinerary is successfully published.

Let's follow the same few steps again, but this time saving the itinerary as a draft. This time, lets say that we want to go on a food trip. Lets just fill in the title and description. Lets click save as draft. Upon submission the form will clear and the user will receive a confirmation banner.

Now lets go to the navbar again. By going to ‘saved drafts’, users can view the itineraries they had saved as drafts. Next, by going to ‘view all your itineraries’, users can view all the itineraries they have created, both the published and drafts.

Lets try to edit our draft itinerary by clicking on the edit button. From here, we can edit the title, description and order of attractions of the itinerary. Lets select food. Lets select the attractions we want to visit now. Now we can try publishing this draft. We have to fill in the review form again, then we are done!

Now if we return to ‘view all your itineraries’ we can see both our published itineraries.

Finally, let's say the user changed their minds on one of the itineraries. The user can click into the show details of the itinerary. If they are the creator of the itinerary, a delete button will show on the bottom of the page. If we click it, we will see a confirmation pop up, before fully deleting the itinerary

We can then go to the ‘Browse All Itineraries’ page to ensure that our remaining itinerary has successfully be published, and that our deleted itinerary is indeed deleted.

Now that the user is done with our application, they can log out through the nav bar.

<Slide > - system architecture diagram

Now, let us bring you through the system architecture diagram.

<Slide>
Firstly, we have the tools used, which include React for the user interface, Mantine UI for UI Library and React Router for the client side routing.

<Slide>
Next, applications. Our application will have 7 static routes and 4 dynamic routes. All the dynamic routes will include those that are determined by the different [itineraryId]s and attraction number

<Slide>
Lastly, we have the application framework. We made use of utility managers to manage interactions between our application and Vercel and Google Maps.

<Slide >
We adopted elements of agile development techniques, specifically extreme programming practices.

Incremental planning:
The project requirements were added incrementally throughout development. This allowed for greater flexibility as we were able to implement new features or reorganise the priorities of different functionalities to adapt to the changing project needs.

Simple Design:
We focused on keeping our design simple so that it delivered on functionality without being overly complicated. This made our code more understandable and easier to debug when issues arise.

Pair programming:
We practiced pair programming whereby we worked in pairs when developing our application. This allowed for us to bounce ideas off each other, to seek clarification when in doubt, and also to check each other's code in real time.

Sustainable pace:
We regularly met in person to discuss the progress of our project, decide on what parts of the work should be prioritized, and the delegation of work. This allowed the work to be spread out evenly so that no singular person had to work large amounts of overtime.

<Slide>
We also followed Scrum principles in this project through sprints. We determined our sprint duration to be 2 weeks, and spread out the work amongst 3 sprints. In the first sprint, we set up

the NextJS project, developed UI for all pages and implemented authentication. In the second sprint, we implement database service and itinerary CRUD. Finally, in the last sprint, we implemented the review feature and integrated the Google Maps API.

<Slide>
Moving on to our tech stack, we used Next.js as our web development framework as it automatically configures tooling needed for React, like bundling and compiling, allowing us to easily use React to build our user interfaces.

<Slide>
As for our design decisions, we chose to use a UI library, as it provides greater design consistency in order to provide a professional and user-friendly user experience. It also allows for faster UI prototyping.

In addition, we implemented custom client-side form validation, allowing us to improve overall user experience and maintain the data integrity of our application.

<Slide>

Next, we used an industry standard combination of git, github and conventional commits to systemise our version control. This allows for easy continuous integration and continuous deployment, and increases the traceability of our application's history.

<Slide>
Our data is stored in PostgreSQL and the images are stored in vercel blob store

<Slide>

To efficiently manage these 2 databases, we will be using Prisma as the persistence layer to store and retrieve data. Prisma allows us to manage both databases efficiently and easily migrate data as we continuously develop the app. Furthermore, functions such as Prisma accelerate and Prisma pulse can be used in future to provide support to handle increased traffic and data as application grows

<Slide>

Last but not least, we utilised ESLint and Prettier to govern and manage our code style. This ensures greater code consistency across our codebase, and allows for better code readability.

<Slide>

These good practices set us up better for these future improvements – Firstly, we can add a discussion forum for users to add a response to the different reviews or comments. Secondly, we can add more complex routing capabilities, thirdly, we can add a feature to copy and modify other date ideas, and lastly we can implement automated testing.

<Slide_>

Now let us deep dive into one of our use cases – Create Itinerary and how we implemented it. Create itinerary involves 2 of our actors – User and Database. This use case is for users to upload their itineraries into our database for others to view. These itineraries include several information: title, description, list of attractions and published or draft status. Each itinerary also includes several attractions which each comprises information such as name, description, budget, image and category.

<Slide_>
To implement the Create itinerary, we had 2 considerations. Given the large number of user inputs, we also had to ensure form validation for each data field. We also wanted to offer a draft functionality where users can save their ideas without publishing for future editing.

<Slide > Let me now bring you through the class diagrams and sequence diagrams for creating an itinerary.
Firstly, we have our class diagram.

<Slide>
Our CreateItineraryPage class has a controller class that interacts with our database.

<Slide>

The controller class is mapped to an itinerary idea. This itinerary comprises several attractions. Each itinerary idea also includes a one to many mapping of itinerary reviews.

<Slide>

And this is our sequence diagram.

<Slide_>
Moving onto our testing, we first used black box testing to test our requirements. We used equivalence classes testing on these fields to ensure that our inputs were reacting as expected. For these fields, we are mainly checking for the presence of input. Specifically, both title and description are checking for text input, with a character limit of 50 characters and 250 characters respectively

<Slide_>
Next, we have our control flow tests. The method that we tested on is our handleSubmit function that creates our itinerary. Our flow test has 3 conditions – the user must be logged in, itinerary name and itinerary description cannot be empty.
We first initialized our itinereary Id to be null and check if the user is authenticated. Then, if the itinerary has a valid name and description, we would return the itineraryId as generated by the database, else the appropriate errors will be thrown.

<Slide_>
Finally, to execute Create itinerary, we implemented our design using Mantine client side validation library. This allows us to achieve 2 benefits.

Firstly, we can enjoy improved user experience. Client-side validation provides immediate feedback, allowing users to correct errors in real time, without submitting the date idea first.
Secondly, we enjoy enhanced data integrity. We can catch validation errors at its source and ensure data submitted to server is more likely to be valid.

<Slide_>
We have come to the end of our presentation. Thank you!