# Problem statement

Suzzane wants to visit Japan for her holidays in December but doesn’t want the manual hassle of going into her browser, making multiple search queries for hotels, food spots, modes of transport she can use, activity areas with good reviews and finally, choosing from her searches. Additionally, she doesn’t know a rough estimation of how much she should budget for the holiday vacation in Japan.

# Proposed solution

To solve her issue, this web application will compile a budget for her vacation. Based on her vacation destination, she will specify what type of spender she is. Based on the destination and her spender type, the application will retrieve a selection of accommodation, food spots, common modes of transportation, and attraction places to visit. Once she is satisfied with her selection, the application will compile a plan detailing the hotel, food spots, transportations, and attractions places and provide an estimated budget for her vacation. Additionally, it will suggest how much she can save per month before her vacation to meet the estimated budget.

# Functional requirements

For users:

* Create, update, or delete their account
* Add, view, update, or delete their budget plans
* Add, or remove their friends on their budget plans
* View budget plans they have been added into as a friend
* View their dashboard for their budget plans
* Report errors found on the application
* Send suggestions or feedback for the developers

For the application:

* Ask users about the progress regarding their budget plans
* Send users reminders about their budget plan goals

For the administrators

* View the dashboard for the entire application
* View and resolve errors
* View and add suggestions to suggestion board

# Non-functional requirements

The application will use JSON Web Tokens (JWTs) and it will hash the user’s passwords before they are stored in the database for enhanced security. It will also be a responsive application to enable accessibility for desktop, Android and iOS devices.

# Feasibility study

## Market feasibility

There are several vacation planner apps with budgeting features. For example, Stippl offers itinerary planning, travel budgeting, and an AI feature that generates tailored itineraries based on your destination. It also helps find accommodation, transportation, restaurants, and activities. Its budget planner lets users set limits, track purchases, and split expenses among co-travellers.

My solution is similar in that it compiles a list of accommodations, food spots, transportation options, and attractions to visit. However, unlike existing planners, it uniquely provides users with an estimated total expense for their entire vacation before the trip begins—a feature not offered by current apps like Stippl. This budgeting capability is especially helpful for individuals visiting unfamiliar places or those who want to know in advance how much they should save. As a result, users can focus on enjoying their trip without the need to track each individual expense during their vacation.

## Technical feasibility

The proposed solution will make user of ReactJS, CSS, HTML, and JavaScript for the development of the frontend, MySQL and SQL for the database, and ASP.NET Core Web API to allow the frontend to make requests to the backend. The solution will be a web-application to make it accessible to desktop, Android and iOS devices. (what cloud storage will it be deployed to?)

## Financial feasibility

By complexity, the ReactJS web application needs to make http requests to an ASP.NET Core web API to add, update, retrieve or delete data in the MySQL database. The frontend development costs R90 000, the backend development costs R120 000, and the Database & APIs costs R170 000. Finally, testing and quality assurance costs R 60 000. Therefore, the total cost of developing this web application is R440 000.

## Schedule feasibility

The project scope:

* User sign up page
* User login page
* User account page
* User home page
* Add budget plan page
* Update budget plan page
* Budget plan dashboard page
* Error reporting page
* Suggestion page
* Admin home page
* Admin dashboard page
* View errors page
* Update errors page
* View suggestions page
* Added suggestions page

Project estimated effort:

* Write documentation detailing problem, solution, use cases, requirements (2 – 5 day)
* Design Entity Relational Database Diagram (5 – 7 days)
* Develop Backend with test data (7 – 10 days)
* Develop ASP.NET Core web API (5 - 10 days)
* Develop frontend (10 - 15 days)

|  |  |
| --- | --- |
| Task | Deliverable deadline |
| Write documentation | 24 September - 28 September 2025 |
| Design ERD Diagram | 28 September - 4 October 2025 |
| Develop Backend | 4 October - 12 October 2025 |
| Develop API | 12 October - 19 October 2025 |
| Develop frontend | 19 October - 26 October 2025 |