**MSE800 Assessment II**

Hengpan He

Wen Liang

Arnold Aristotle Tayag

**Introduction**

As the tourism industry grows rapidly, an online tour booking system has become essential for tourists and operators. To address this, we are developing a Tour Booking Management System. This system will help users quickly search and book tours while providing operators with tools to manage their offerings. The frontend will use HTML, CSS, and JavaScript, while Node.js and Python will handle backend logic. Agile methodology will be used and a duration of one–week sprint for development is adopted to allow flexibility in meeting user requirements.

The goal of this project is to create a user-friendly platform for tourists to find tour information, book, pay, cancel reservations, and leave reviews. Operators and administrators will have tools to manage tours, user accounts, generate reports, and monitor system performance to ensure efficiency.

The project focuses on New Zealand tours, including browsing information, online booking, reviews, and monitoring. The development will be divided into three releases. A minimum viable product (MVP) will be available in the first release and gradually adding features at each stage.

The main goal is to deliver a comprehensive and user-friendly online booking platform to meet both tourists' needs and enhance operators' business efficiency.

**Project Overview**

The Tour Booking Management System will allow users to search, book, and manage tours while providing tour operators with tools to create and manage tour offerings. The system will include various features that enhance user experience and operational efficiency.

**Key Features:**

* **Tour Browsing:** View detailed information (description, price, schedule, ratings).
* **Manage Online Booking:** Secure booking, email confirmations, real-time status updates.
* **Payment Processing:** Integration with payment gateways to facilitate secure online transactions.
* **User Registration and Profiles:** Users can create accounts to manage bookings, save favorite tours, and receive personalized recommendations
* **Tour Review:** Users can leave reviews and rate their experiences, helping future customers make informed decisions.
* **Admin Interface:** Manage user accounts, generate reports, and monitor system performance.
* **System Monitoring & Reporting:** Centralized performance and error log tracking.
* **Customer Support:** Email support for inquiries.
* **System Integration:** API for third-party integrations.

**Benefits:**

* User-Friendly Interface: Intuitive design ensures users can easily navigate the system and find suitable tours
* Streamlined Operations: Helps tour operators manage bookings and customer interactions efficiently
* Cultural Sensitivity: By including tours that focus on local Māori culture and history, the system can respect and promote indigenous perspectives
* Increased Visibility: Operators can showcase their tours to a wider audience, increasing bookings and revenue

**Potential Extensions:**

* Mobile Application: A companion mobile app for users to book and manage tours on the go
* Social Sharing Features: Allow users to share their experiences on social media to promote the tours and attract more customers
* Multilingual Support: The system can support multiple languages, catering to international users
* Calendar Integration: A calendar feature that shows available dates for each tour, allowing users to see availability in real time

**Project Plan**

**System Environment:**

* Backend: Node.js + Python
* Frontend: HTML, CSS, JavaScript (React/Vue)
* Database: SQLite
* Hosting: Local server or cloud-based
* Tools: Git for version control, Jira for project management

**Development Approach:**

* Agile methodology (Scrum + XP)
* 1-week sprint
* Fibonacci estimation

**Stakeholders:**

* Product Owner / Business Users (Tourists/Customers/Tour Operators/Tour Owners) / Stakeholders / Sponsor: Provides requirements and feedback on the product's functionality.
* Agile Facilitator: Oversees project progress, facilitates meetings, and ensures agile practices are followed.
* Project Team: Responsible for designing, developing, and testing the system.

Team Structure & Governance

Project Kick-off Meeting

**Scope of Work**

The system focuses on providing tour booking services exclusively for tours within New Zealand. It will cover the following key features:

1. **Tour Browsing**: Displays detailed information such as tour descriptions, prices, itineraries, and ratings.
2. **Online Booking**: Supports email confirmation, payment processing, and tracking of booking statuses.
3. **Tour Review**: Allows users to provide feedback and ratings after the tour.
4. **User Management**: Enables administrators to manage user accounts and permissions.
5. **System Integration**: Offers APIs for integrating with third-party payment services and tour information providers.

**Out of Scope**

The following items are beyond the scope of this project:

1. Offline travel arrangements or services outside of New Zealand.
2. **Travel insurance management**: The system will not handle or process insurance requests or claims.
3. **Multi-currency support**: The platform will only process payments in New Zealand dollars (NZD) without exchange rate calculations.

**System Design**

**System Architecture**

1. **Frontend Technologies:** The frontend will be developed using HTML, CSS, and JavaScript, ensuring a responsive and user-friendly interface for tourists and administrators.
2. **Backend Technologies:** The backend logic will be handled using a combination of Python and Node.js to efficiently manage business logic, API calls, and data processing.
3. **Database:** The system will use SQLite as the primary database to store data such as tour details, user information, bookings, and feedback.

**Module Breakdown**

1. **Tour Browsing Module:** Loads available tours from the database and displays relevant information to the users based on their preferences.
2. **Online Booking Module:** Processes bookings and payments, updates booking status, and handles email confirmations.
3. **Admin Interface:** Provides system administrators with the ability to manage users, process booking requests, and generate reports.
4. **Tour Review Module:** Collects and displays feedback from users, including ratings and comments.
5. **System Integration Module:** Integrates with third-party services through APIs, such as payment gateways.

**System Interfaces**

1. **Tour Listing Page:** Displays all available tours and allows users to filter based on preferences like destination, date, and tour type.
2. **User/Admin Login Page:** Provides separate login interfaces for tourists and administrators.
3. **Booking Page:** Facilitates the booking process by collecting traveler details, confirming availability, and processing payments.

**Data Flow and Communication**

1. **Frontend and Backend Communication:** The frontend will communicate with the backend using REST APIs to ensure smooth data exchange. Node.js will handle API requests, while Python will manage the business logic and data processing.
2. **Database Operations:** SQLite will store all necessary data, including user accounts, tours, bookings, and reviews. The backend will manage data queries and ensure synchronization between modules.
3. **Scalability Considerations:** Although SQLite is a lightweight database suitable for the initial development phase, the system architecture allows for future migration to more robust databases (e.g., MySQL or PostgreSQL) if needed.

**Product Backlog Prioritization & Refinement**

After many meetups and discussions with various stakeholders, below features were identified as critical to meet the business requirements:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Story # | Functionality | User Story | Acceptance Criteria | Prioritization |
| 1 | For Tourists, Customers | As a tourist, I want to browse available tours, so that I can find options that interest me | Users can filter tours by category, location, and price | Release 1 |
| 2 | For Tourists, Customers | As a tourist, I want to view detailed information about each tour, so that I can make informed decisions | Each tour page includes a description, itinerary, duration, price, and reviews | Release 1 |
| 3 | For Tourists, Customers | As a tourist, I want to book a tour online, so that I can secure my spot easily | Users can select dates, number of participants, and complete payment through a secure gateway | Release 1 |
| 4 | For Tourists, Customers | As a tourist, I want to receive confirmation emails after booking, so that I have all the details at hand | A confirmation email is sent immediately after booking, containing all relevant details | Release 1 |
| 5 | For Tourists, Customers | As a tourist, I want to be able to cancel or modify my booking, so that I can manage my plans flexibly | Users can cancel or modify bookings within a specified time frame, with a clear refund policy | Release 1 |
| 6 | For Tourists, Customers | As a tourist, I want to leave reviews for tours I’ve taken, so that I can share my experiences with others | Users can rate tours and provide written feedback after the tour is completed | Release 2 |
| 7 | For Tourists, Customers | As a user, I want to log in and set my language preference (e.g., Māori) | Users can set preferred language to any language supported by the system | Release 1 |
| 8 | For Tour Operators, Administrators | As a tour operator, I want to create and manage tour listings, so that I can keep my offerings up to date | Operators can add, edit, or delete tours and manage details like pricing and availability | Release 2 |
| 9 | For Tour Operators, Administrators | As a tour operator, I want to view booking reports and analytics, so that I can assess the performance of my tours | Operators can access dashboards showing sales, cancellations, and customer feedback | Release 2 |
| 10 | For Tour Operators, Administrators | As a tour operator, I want to manage customer inquiries, so that I can provide timely responses | Operators have a messaging system to communicate with customers regarding inquiries or concerns | Release 2 |
| 11 | For Tour Operators, Administrators | As a tour operator, I want to set seasonal pricing and promotions, so that I can attract more customers | Operators can schedule discounts and special offers that apply to specific dates or tours | Release 2 |
| 12 | For Tour Operators, Administrators | As a tour operator, I want to manage payment processing, so that I can receive payments securely | The system integrates with payment gateways, allowing for secure transactions and providing operators with transaction reports | Release 1 |
| 13 | For Admins | As an admin, I want to manage user accounts, so that I can maintain the integrity of the platform | Admins can create, edit, and deactivate user accounts for both customers and operators | Release 1 |
| 14 | For Admins | As an admin, I want to monitor system performance and error logs, so that I can ensure the system runs smoothly | Admins have access to performance metrics and can view logs for troubleshooting | Release 2 |
| 15 | For Admins | As an admin, I want to enforce policies and regulations for tour operators, so that the platform maintains quality standards | Admins can set guidelines for tour operators and review compliance | Release 3 |
| 16 | For Admins | As an admin, I want to generate reports on user activity and bookings, so that I can analyze system usage and revenue | Admins can generate customizable reports on various metrics | Release 2 |
| 17 | For Developers | As a developer, I want to ensure the system is responsive, so that users can book tours on any device | The booking platform functions seamlessly on mobile, tablet, and desktop | Release 1 |
| 18 | For Developers | As a developer, I want to implement secure payment processing, so that user financial information is protected | The payment system complies with industry standards for security (e.g., PCI DSS) | Release 1 |
| 19 | For Developers | As a developer, I want to create an API for third-party integrations, so that the system can connect with other applications | The API provides endpoints for tour listings, bookings, and user management | Release 3 |

**Release Planning / Sprint Planning**

**Release Plan:**

**Release 1:**

1. Tour browsing -

- description about the tour

- price

- schedule (place, time, activities)

2. Online booking

- send booking email confirmation

- booking cancellation

- online secure payment

- booking status: booked, paid, cancelled, modify

4. Admin Interface

- manage user accounts

user types: customer, system users (user, admin)

activities: customer + user creation, change privilege

6. Customer Support

- send email for clarifications

**Release 2:**

1. Tour browsing -

- show computed average rating

- tour management

3. Tour review

- 1 rating from 1 to 5

- customer comments/recommendations

4. Admin Interface

- report generation

5. System Monitoring & Reporting

- system performance

- monitor error logs

- booking report and analytics

**Release 3:**

7. System integration

- API for third-party integrations

**Sprint Plan for Release 1**

|  |  |  |
| --- | --- | --- |
| User Story # | Estimation (Fibonacci) | Sprint Number |
| 1 | 2 | 1 |
| 2 | 2 | 1 |
| 3 | 5 | 1 |
| 4 | 5 | 2 |
| 5 | 5 | 2 |
| 7 | 5 | 2 |
| 12 | 4 | 3 |
| 13 | 5 | 3 |
| 17 | 13 | 4 |
| 18 | 4 | 4 |

**Sprint Plan for Release 2**

|  |  |  |
| --- | --- | --- |
| User Story # | Estimation (Fibonacci) | Sprint Number |
| 6 | 3 | 1 |
| 8 | 5 | 1 |
| 9 | 8 | 2 |
| 10 | 5 | 2 |
| 11 | 5 | 3 |
| 14 | 8 | 3 |
| 16 | 8 | 4 |

**Sprint Plan for Release 3**

|  |  |  |
| --- | --- | --- |
| User Story # | Estimation (Fibonacci) | Sprint Number |
| 15 | 8 | 1 |
| 19 | 13 | 1 |

**Database Schema:**

**Tables:**

1. Users
2. Roles
3. Bookings
4. Booking\_status
5. Payments
6. Reviews

**DB Scripts:**

create database TourBooking.db;

CREATE TABLE users (

ID int primary key not null,

FNAME char(50) not null,

LNAME char(50) not null,

USERID char(50) not null,

PASSWORD char(50) not null,

ROLE char(10) not null,

ENTERED\_BY char(50) not null,

ENTRY\_DATE text not null

);

insert into users values (1, 'admin', 'admin', 'admin', 'admin@123', 1, 'admin', strftime('%d/%m/%Y', date()));

CREATE TABLE roles (

ID int primary key not null,

ROLE\_NAME char(50) not null,

ROLE\_DESC char(50) not null,

ENTERED\_BY char(50) not null,

ENTRY\_DATE text not null

);

insert into roles values (1,'admin', 'admin', 'admin', strftime('%d/%m/%Y', date()));

insert into roles values (2,'user', 'ordinary user', 'admin', strftime('%d/%m/%Y', date()));

insert into roles values (3,'customer', 'customer', 'admin', strftime('%d/%m/%Y', date()));

CREATE TABLE booking\_status(

ID int primary key not null,

STATUS\_NAME char(50) not null,

STATUS\_DESC char(50) not null

);

CREATE TABLE bookings (

ID integer primary key autoincrement,

TOUR char(50) not null,

NAME char(50) not null,

EMAIL char(50) not null,

DATE char(50) not null,

STATUS int not null

);

create table payments (

ID int primary key not null,

BOOKING\_ID integer not null,

AMOUNT real not null,

ENTERED\_BY char(50) not null,

ENTRY\_DATE text not null

);

create table reviews (

ID int primary key not null,

BOOKING\_ID integer not null,

RATING integer not null,

DESCRPTION char(50) not null,

ENTERED\_BY char(50) not null,

ENTRY\_DATE text not null

);

**Costing/Budget**

Assumptions:

1. Sample budget outline (baseline only) for a Tour Booking Management System, broken down into development, infrastructure, and operational expenses

2. Budget assumes a small to mid-sized tour agency is building a custom system

3. Actual cost will vary based on system complexity, user volume, and additional features, such as AI-driven personalization or advanced reporting

1. Development Costs

a. Planning & Design

* Market Research & Feasibility Study: $1,500 - $3,000
* Requirement Gathering and Documentation: $1,500 - $3,000
* System Architecture Design: $1,500 - $3,000

b. Software Development

* Frontend Development (HTML, CSS, Javascript (React/Angular/Vue)): $5,000 - $10,000
* Backend Development (Node.js/Django): $10,000 - $20,000
* Database Setup (SQLite): $1,000 - $2,000
* System Integration: $5,000 - $10,000
* Payment Gateway Integration: $2,000 - $5,000

c. Testing & Quality Assurance

* Automated & Manual Testing: $3,000 - $6,000
* User Acceptance Testing (UAT): $1,000 - $3,000

2. Infrastructure & Licensing Costs

a. Hosting & Server Costs

* Cloud Hosting (AWS, Azure, or Google Cloud): $100 - $500/month
* Domain Name Registration: $10 - $30/year
* SSL Certificate: $50 - $200/year

b. Software Licensing & Subscriptions

* Booking & CRM Software Integration: $500 - $2,000/year
* Payment Gateway Fees (Stripe, PayPal, etc.): 2.9% + $0.30 per transaction
* Analytics Tools (Google Analytics): $10 - $100/month

3. Operational Costs

a. Content Creation & Marketing

* Graphic Design: $1,000 - $2,500
* SEO & Marketing: $300 - $1,000/month

b. Ongoing Maintenance & Updates

* System Maintenance & Bug Fixes: $1,000 - $2,500/month
* Feature Updates & Enhancements: $2,000 - $4,500/quarter

c. Staff Training & Support

* Training Sessions for Admin & Staff: $500 - $1,500
* Documentation Creation: $500 - $1,500

Total Estimated Cost (1st Year):

Initial Development & Setup Costs: $36,000 - $81,000

Ongoing Monthly Costs: $2,000 - $5,000

Total Yearly Operational Cost: $26,000 - $66,000

**Sign–Off**

**Reflection Report**

The Tour Booking System is a project that has its origin from the increasingly felt need in the tourism market for an effective, user-friendly online booking system targeting tours around New Zealand specially focus on Auckland. In this group project, we have worked toward developing a system that would give tourists an enhanced feeling during booking on one hand and operators a set of tools badly needed to operate tours with maximum efficiency on the other hand. Agile methodology allowed us to adapt flexibly in the face of evolving requirements and user feedback during all stages of the project's lifetime. This report reflects on the stages of demand gathering, requirement analysis, code implementation, testing, and release, focusing on the lessons learned at each stage. Here are the following details of the reflection for each stage.

**Demand Gathering**

This is the demand-gathering phase, during which we have had limited stakeholder interactive engagement in the form of surveys and interviews with potential users: tourists, tour operators, and administrators. Of course, these roles are acted by each of our team members respectively and virtually. Role of tourists had a number of suggestions on how to make their booking simpler, having full information about the tour, and reviewing options. One important challenge was to strike a balance between the seemingly conflicting requirements presented by different needs, and these needs were carefully documented and categorized and had to be balanced.

**Requirements Analysis**

We prioritized requirements into features such as "Must-have," "Should-have," "Could-have," and "Won't-have." The reason for this methodology is that it will enable us to stick to core functionalities while still accommodating some functionalities that can be part of future enhancement plans. Hence, at this stage, the user demands were mapped into functional and non-functional requirements. This helped us outline the project scope for each version and helped plan our Agile sprints in that respect.

**Coding**

The coding was divided into two-week sprints in which we would develop specific features based on the prioritized requirements. We went with Agile methodologies which allowed us to fix our development process as we went and to perform feedback intake or any other changes on the go right away. After each sprint, we had a working prototype, reflecting the incremental approach towards our product in its entirety.

**Testing**

Testing forms one of the very key features related to the project, being done through various stages of the project in order to ensure the quality of the system. We started unit testing, verifying that all the components were behaving as anticipated. Then we moved on to integration testing and following UAT. But due to the time limitation we skipped performance test in the first release and planned to implement in the second stage for the reason that at the earlier period of online services, the number of registered costumers will be less.

**Challenges and Lessons Learnt**

We faced various types of challenges while working on this project; each taught us something important. An important challenge was coordination and communication within the team during different phases of the project. The overall cooperation among each one of the team members taught us the importance of clear communication protocols and collaboration tools, which maintained the alignments and reduced misunderstandings.

Other challenges were technical integrations because rigorous testing and fixing needed to be done to ensure a smooth flow of data across these components, especially when real-time updates occurred in the booking status. This only reinforced the idea of modular and robust documentation since these allowed us to fix problems much faster.

Finally, the time management was an issue, too, especially in the closing stages after the coding phases. In balance with the thoroughness of testing, timewise for the project forced us to prioritize which features were most vital and which we could get away with not focusing on for each sprint. This taught us the importance of setting realistic goals and timelines for each cycle of development, since project timelines have to be respected because one wants to deliver a quality product within expected times.

The challenges we have gone through really imparted lessons on communication, technical integration, time management, and user-centered development. Such insight from our group project will surely help us to develop anything even more effectively and efficiently in our future projects.