**MSE800 Assessment II**

Hengpan He

Wen Liang

Arnold Aristotle Tayag

1. **Introduction**

As the tourism industry grows rapidly, online tour booking has become essential for tourists and operators. To address this, we are developing a Tour Booking Management System using Agile methodology. 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 Python will handle backend logic. Two-week sprints 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 versions, gradually adding features at each stage.

**Key Features:**

* **Tour Browsing:** View detailed information (description, price, schedule, ratings).
* **Manage Online Booking:** Secure booking, email confirmations, real-time status updates.
* **Tour Review:** Tourists can rate and review tours.
* **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.

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

1. Team Structure & Governance
2. Project Kick-off Meeting
3. 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.
4. 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.
4. Product Backlog Prioritization & Refinement
5. **Release Planning / Sprint Planning**

**Release Plan**

Release 1:

1. Tour browsing -

- description about the tour -1

- price -1

- schedule (place, time, activities) -1

2. Online booking

- send booking email confirmation -1

- booking cancellation -1

- online secure payment -1

- booking status: booked, paid, cancelled, modify -1

4. Admin Interface

- manage user accounts

user types: customer, system users (user, admin) -1

activities: customer + user creation, change privilege -1

6. Customer Support

- send email for clarifications -1

Release 2:

1. Tour browsing -

- show computed average rating -2

- tour management -2

3. Tour review

- 1 rating from 1 to 5 -2

- customer comments/recommendations -2

4. Admin Interface

- report generation -2

5. System Monitoring & Reporting

- system performance -2

- monitor error logs -2

- booking report and analytics -2

Release 3:

7. System integration

- API for third-party integrations -3

**Sprint Plan for Release 1**

|  |  |  |  |
| --- | --- | --- | --- |
| User Story | Estimation | Acceptance Criteria | Sprint Number |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **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

1. Sign–Off
2. Reflection Report

#################################

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

1. User Registration and Profiles

* Users can create accounts to manage bookings, save favorite tours, and receive personalized recommendations

1. Tour Search and Filter Options

* Users can search for tours based on criteria such as destination, price range, duration, and user ratings

1. Booking Management

* Users can easily book tours, make payments, and receive confirmations.
* Options for group bookings and individual itineraries

1. Tour Operator Dashboard

* A dedicated interface for tour operators to add new tours, manage availability, and view bookings and customer feedback

1. Payment Processing

* Integration with payment gateways to facilitate secure online transactions.

1. Reviews and Ratings

* Users can leave reviews and rate their experiences, helping future customers make informed decisions

1. Customer Support

* Live chat support and a FAQ section to assist users with their queries.

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

Alignment with Assessment Requirements

* Agile Methodology: The project can be developed iteratively & incrementally, with feedback loops from both users and operators to refine features
* Cultural Considerations: Incorporating local cultural elements into tour offerings (e.g., guided cultural experiences) will align with the principles of Te Tiriti o Waitangi, emphasizing partnership and respect for Māori perspectives
* Collaboration: The project will require collaboration among diverse team members, potentially including stakeholders from various cultural backgrounds to ensure inclusivity

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

This Tour Booking Management System not only meets the requirements of the assessment but also serves as a valuable tool for promoting tourism while respecting and integrating local cultures.

Project Plan

**System Environment**

Backend: Python

Frontend: HTML, CSS, JavaScript (React.js/Vue.js)

Database: SQLite

Hosting: Local server or cloud-based

Tools: Git for version control, Jira for project management

Project Roles

Product Owner / 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.

Project Roles:

* Product Owner / Business Users / Stakeholders / Sponsor
* Agile Facilitator
* Project Team

Product Backlog

|  |  |  |
| --- | --- | --- |
| Epic / Feature / User Story |  |  |
| User Story 1 | User Management | As a user, I want to register an account to manage my bookings. |
| User Story 2 | User Management | As a user, I want to log in and set my language preference (e.g., Māori). |
| User Story 3 | Tour Management | As an admin, I want to add, update, and delete tour information. |
| User Story 4 | Tour Management | As a user, I want to search and filter tours based on various criteria.  Epic 3: Booking and Payment. |
| User Story 5 | Booking and Payment | As a user, I want to book a tour and make secure payments online. |
| User Story 6 | Booking and Payment | As an admin, I want to view all bookings and manage availability.  Epic 4: Reviews and Support. |
| User Story 7 | Reviews and Support | As a user, I want to leave a review and rate my tour experience. |
| User Story 8 | Reviews and Support | As a user, I want to access customer support through a live chat or FAQ section. |

Sprint Planning

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint | User Story |  |  |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

Database Tables

1. Users
2. Bookings
3. Payments
4. Reviews
5. Roles
6. The
7. The

Database creation 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 bookings (

ID int primary key not null,

...

ENTERED\_BY char(50) not null,

ENTRY\_DATE text not null

);

create table payments (

ID int primary key not null,

...

ENTERED\_BY char(50) not null,

ENTRY\_DATE text not null

);

create table reviews (

ID int primary key not null,

...

ENTERED\_BY char(50) not null,

ENTRY\_DATE text not null

);

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()));