**A Brief Introduction and Overview of the Selected Features**

**Project Title:** Liam’s Visit Korea – Interactive Tour Experience

1. Project Overview:

The project **“Liam’s Visit Korea”** is a dynamic, feature-rich tourism website designed to provide users with an immersive, modern travel discovery experience. The site highlights curated tour packages across South Korea’s four seasons and cultural themes, while leveraging advanced front-end and back-end techniques to improve interactivity, data-driven rendering, and user experience.

To fulfill the task requirements of SIT774 Task 10.4HD, a set of advanced features were selected, implemented, and integrated into the website. These features address both **technical depth (backend + routing)** and **user engagement (front-end + UI/UX interaction)**.

1. Selected Features and Purpose:
2. **Application of Extended User Interface Styling & Interaction Effects.**

To deliver a compelling and intuitive user experience, the project incorporates multiple advanced user interface (UI) styling and interaction enhancements that go beyond default Bootstrap usage. These enhancements were deliberately designed to support the theme of cultural exploration while ensuring accessibility, responsiveness, and engagement across devices.

Key implementations include:

* **Scroll-triggered animations:** Using the IntersectionObserver API combined with CSS transitions (opacity, translateY), elements such as testimonials, tour cards, and promotional sections animate into view as the user scrolls. This creates a sense of progression and narrative flow, reducing visual fatigue and encouraging exploration.
* **Custom-styled Bootstrap components:** Default Bootstrap buttons, modals, pagination, and forms were extended with custom classes (e.g., .search-button) to match the site’s pastel aesthetic and Korean tourism brand identity. This includes rounded-pill buttons, custom hover effects, and tailored form validation styling.
* **Video banner carousel:** The homepage hero section features an autoplaying background video carousel, layered with call-to-action overlays. This provides immediate immersion and emotional resonance, supporting the storytelling aspect of the site.
* **Interactive feedback and microinteractions:** All user actions—such as form submissions, hover events, and scroll triggers—provide smooth, immediate visual responses. This includes inline form error messages, button hover highlights, and a responsive “Back to Top” button.
* **Consistent responsive design:** The layout leverages Bootstrap’s grid system (col-md-4, col-md-6, etc.) and utility classes (text-center, fw-bold, etc.) to ensure that elements maintain their hierarchy and clarity across all screen sizes.

Together, these interface and interaction effects significantly enhance the overall usability and polish of the site, aligning with the project’s objective of delivering a modern, engaging, and culturally enriched digital tourism experience.

**2. Customized Bootstrap Components**

Several Bootstrap elements were extended beyond their default styles:

• The Search button and Subscribe button use a .search-button class, which adds a pastel-pink background, soft hover effects, and rounded-pill borders to match the theme and brand tone.

• Pagination controls (.page-link) for seasonal tours are customized with active state tracking and soft color indicators.

• Modals for Login and Signup are not just Bootstrap defaults — they’re styled with spacing, color contrast, and error message handling that mimic real-world applications.

1. Implementation of Routing and Error Handling using Express.js

The project leverages the Express.js framework on the backend to build multiple routes that serve dynamic content and respond to user requests in real time. Custom routes were created for actions such as user registration (/signup), user login (/login), dynamic tour search (/search?keyword=), and popular tour retrieval (/top-tours).

To ensure stability and reliability, server-side error handling has been implemented at each route. For instance:

• The /signup route returns a specific HTML error page when required fields are missing or when the email already exists.

• The server uses status codes (e.g., 400, 500) and structured error messages (via res.status().send(...)) to inform the frontend of different failure scenarios.

• These error messages are dynamically interpreted by the frontend and rendered below form inputs, providing instant user feedback.

This routing architecture and error-handling approach mirrors practices in production-ready web apps and provides a scalable base for future feature additions.

1. Implementation of Secure User Authentication

Security is a top priority in user interactions. All authentication flows in this project are handled securely using bcrypt hashing on the backend and protected data transmission practices on the frontend.

Key implementations include:

• Passwords are hashed using bcrypt before being stored in the SQLite database, ensuring no plain text credentials are retained.

• Client-side JavaScript collects user inputs, validates them (password match, email format, etc.), and sends them via fetch() with Content-Type: application/json.

• Credentials are never passed via query strings or URLs, reducing exposure risks.

• Session-based tracking using express-session helps manage login state (optional depending on full scope).

This authentication model not only follows best practices, but also demonstrates a responsible handling of sensitive user information in a student-developed full-stack web application.

1. Dynamic Content Rendering from JSON Data

One of the standout features of the project is its use of JSON-based dynamic rendering to simulate real-world data-driven interactions. In my web, I did:

• The tour search form uses a fetch() call to query the Express route /search, which returns matching tours from the SQLite database in JSON format. The frontend parses and renders the results live into a responsive Bootstrap grid layout.

• The “Some Favourite Tours” carousel is populated via server-side JSON (/top-tours) and dynamically inserted into the DOM using JavaScript.

• A seasonal image and title panel is updated dynamically every few seconds using an internal categoryData array, mimicking live content rotation.

This approach showcases asynchronous communication, DOM manipulation, and separation of content from presentation, all core skills in modern JavaScript development.

1. Persistent Data Storage Using SQLite

The backend system uses a lightweight, server-side SQLite database to persistently store user data, messages from contact forms, and available tours.

• The init.js script initializes the database with three tables: users, messages, and tours, and pre-populates them with seed data.

• Backend logic uses db.run() to insert data (e.g., on signup or message submission) and db.all() to retrieve filtered data (e.g., in search).

• Unlike static JSON files, this structure enables ongoing updates and real-time querying.

This fulfills the requirement of implementing alternate permanent storage and also introduces students to SQL-based backend development, balancing simplicity with real-world applicability.