SCRIPT

**Start: Full-Screen Introduction (0:00 – 0:30)**

**What to Show:** Homepage (home.html) in browser  
**Say:**

Hello, I’m Lam, and this is my Task 10.4 project: *Liam’s Visit Korea*. It’s a full-stack tourism website built with HTML, CSS, Bootstrap, JavaScript, Node.js, and SQLite. Let’s walk through the key features I’ve implemented.

**🔹 Scroll Animations + UI Effects (0:30 – 2:00)**

**Show:** Scroll slowly down homepage  
**Say:**

As we scroll, content fades in smoothly using IntersectionObserver and .fade-on-scroll CSS transitions.

**Switch to:** script.jsA screen shot of a computer program

AI-generated content may be incorrect.

In script.js, I use an observer that adds .fade-in when an element enters the viewport.A screen shot of a computer program

AI-generated content may be incorrect.

In style.css, .fade-on-scroll sets opacity and pushes the element down. When .fade-in is added, it slides up and becomes visible.

**🔹 Dynamic Tour Search (2:00 – 3:30)**

**Khi thao tác trên trình duyệt (gõ “Jeju”):**

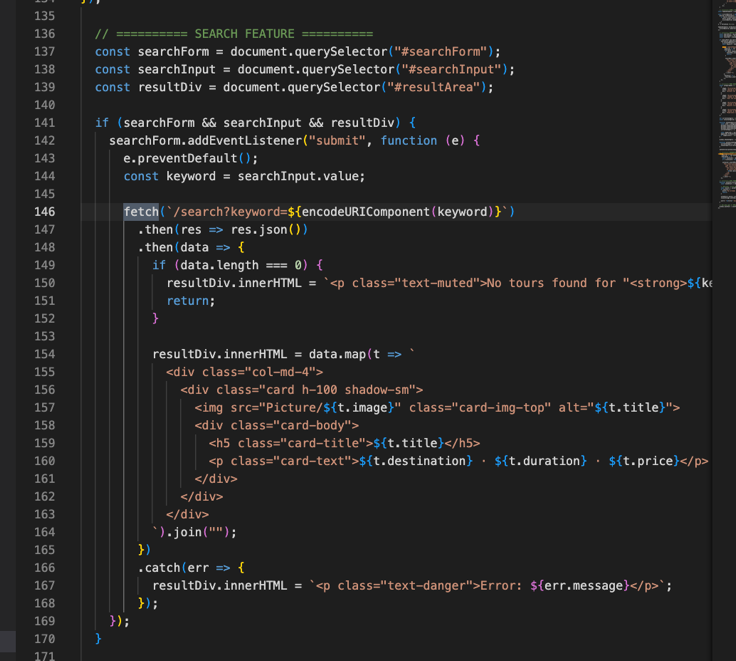
**“Let’s try searching for a tour — I’ll type ‘Jeju’ here and hit Search.**

**<PAGE SEARCHING và thao tác search trên đó>**

**Actually, The page now sends a request to the backend using fetch, and it receives matching tour data in JSON format from the server.**

**As you can see, the results are dynamically displayed as tour cards without reloading the page.”**

**Chuyển sang script.js (chỉ đoạn fetch(/search?keyword=...)):**

****

“Now here’s how this works in the frontend.

In script.js, the search form listens for submit, then sends a fetch() request to /search, passing the keyword as a query parameter.

When the server responds with JSON, I use .map() to inject each tour into the page as a Bootstrap card.”

**Chuyển sang index.js, chỉ route /search:A computer screen shot of text

AI-generated content may be incorrect.**

“On the backend, this /search route takes the keyword from the query string, adds wildcards using %, and performs a case-insensitive SQL query.”

**Mở terminal chạy:**

Here’s the actual query behind the scences:

sqlite3 mydb.sqlite

SELECT \* FROM tours WHERE destination LIKE '%Jeju%';

“So the /search route in my Express server simply performs this exact SQL query behind the scenes using db.all() in JavaScript, and then returns the result as JSON to the frontend.”

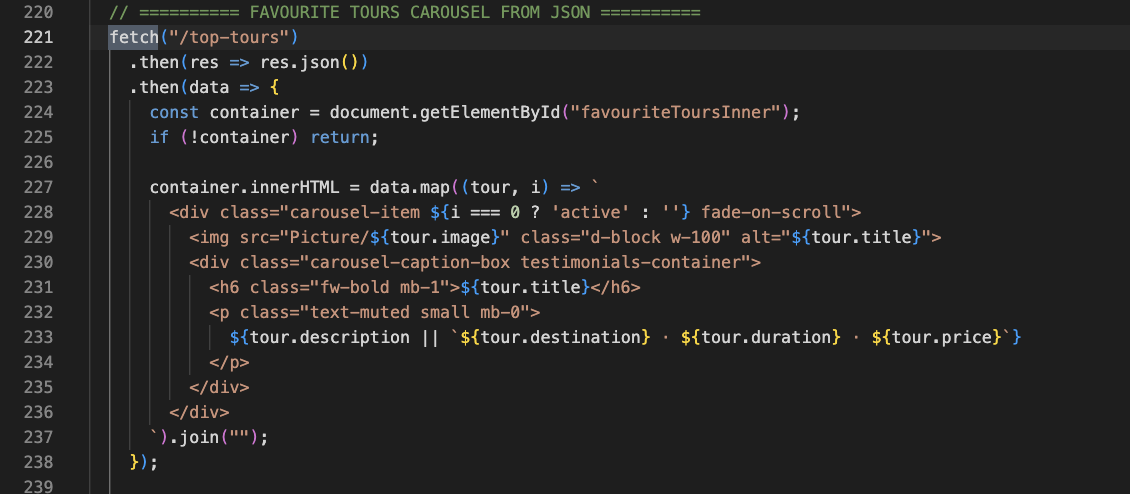
**Favourite Tours Carousel (3:30 – 4:30)**

1. **Mở trình duyệt → kéo xuống tới mục  “Some Favourite Tours”**

Let’s move to Favoutire tours carousel section, This carousel is dynamically built.

In script.js, I fetch /top-tours to load 3 popular tours. Each slide is rendered using JS and injected with .carousel-item.

**2. Chuyển sang  script.js → chỉ đoạn fetch("/top-tours")**

****

“This fetch() call retrieves the top 3 featured tours from the server.

The data is returned as a JSON array, and this block of code maps over each item to render it as a Bootstrap carousel slide.

Notice that the first item is conditionally marked as active so the carousel initializes properly.”

**3. Chuyển sang  index.js→ chỉ route /top-tours**

**A computer screen shot of text

AI-generated content may be incorrect.**

**“On the backend, this is the /top-tours route. It runs a simple SQL query to select the top 3 tours from the database.**

**The data is returned as JSON, which the frontend uses to build the carousel slides.”**

“This dynamic carousel shows how I use asynchronous JavaScript to load and render content from the server in real time. It improves performance and reduces hardcoded HTML.”

**🔹 Signup & Secure Auth (4:30 – 5:30)**

1. **Trình duyệt → Click “Sign Up” → điền form → bấm submit**

“Let’s register a new user.

Let’s register a new user. When I fill out the form and submit, the data is validated on the client side and then sent to the backend.

*(Chờ thông báo “Sign Up Successful” xuất hiện)*

“As you can see, the registration succeeds and we’re redirected to the homepage.”

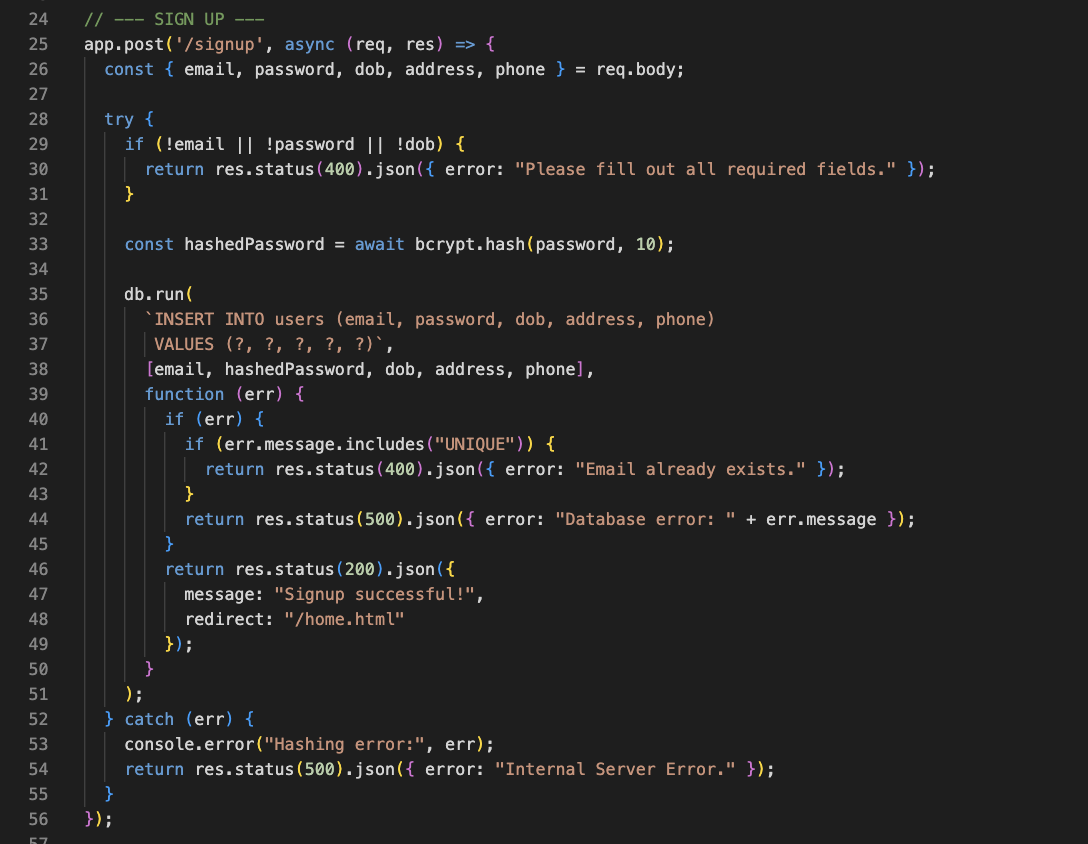
1. **Chuyển sang script.js→ show phần validation form**

**It** show form validation logic “I check that emails are valid, passwords match, and all fields are filled. Then I send the form data using fetch().”

**A computer screen shot of a program

AI-generated content may be incorrect.**

**3. Chuyển sang index.js → show route /signup**

****

Here, I hash the password using bcrypt with 10 salt rounds before saving it in SQLite.”

<LOGIN.HTML>

“I’ll now log in using the same credentials. On submit, the server compares the stored bcrypt hash to verify identity, and if successful, a session is created with express-session.”

**I will Switch to:** index.js, as u can see in login route “Here, I query the user by email and verify the password using bcrypt.compare().”

A computer screen shot of text

AI-generated content may be incorrect.

<MOVE TO CHAT BOX>

One of the standouts features I wanted to include is this chatbot.

It’s built using Firebase Realtime Database for message storage, and the backend uses OpenAI’s Chat Completion API to generate smart replies.”

“Here I’ll type ‘Hello?’ and send it.”

(gõ và bấm Send – xuất hiện phản hồi mặc định như ảnh)

“Currently, the chatbot returns a default fallback response — because my OpenAI API quota has been exceeded.

But under normal conditions, this would reply with real travel tips from ChatGPT.”

📂 Switch: chatbotServer.js → highlight đoạn fetch()

“This is the backend logic in chatbotServer.js. When a user sends a message, the server makes a request to OpenAI’s API and sends back the assistant’s reply.”

A screenshot of a computer program

AI-generated content may be incorrect.

📂 Switch: script.js → highlight child\_added

“And here on the frontend, Firebase’s child\_added listener detects new messages and displays them in real time — creating a live chatbot experience.”

A screen shot of a computer code

AI-generated content may be incorrect.

**🔹 Error Handling in Express (5:30 – 6:30)**

1. **Trình duyệt → Thử đăng ký lại với email đã dùng trước đó**

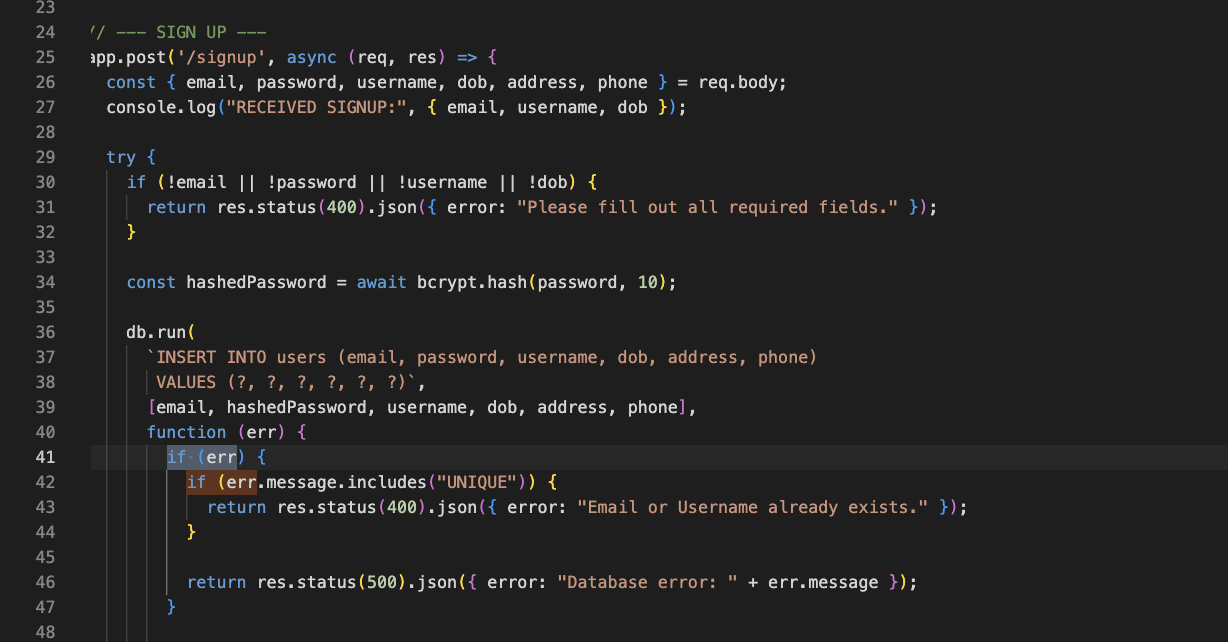
“Now I’ll try signing up again using an email that already exists in the database.

When I click Submit, the server detects the duplication and returns a custom error response.”

*(Trang hiện thông báo lỗi: “Email or Username Already Exists”)*

“This feedback is returned as an HTML page — or as JSON if the form uses fetch — depending on how the request is sent.”

**2. Chuyển sang  index.js → show đoạn  /signup và  res.status(400).send(...)**



“Here in the /signup route, if the SQLite error contains the word UNIQUE, that means the email or username is already taken.

I return a 400 status code and a JSON message with a custom error.”

**3. Chuyển sang script.js → show phần nhận JSON lỗi**

A screen shot of a computer program

AI-generated content may be incorrect.

“On the frontend, I check if the server response fails and immediately show the error message under the signup form — without reloading the page.

This improves usability by giving users real-time feedback.”

“Each backend route like /signup, /login, or /search is modular and returns clear JSON or HTML responses depending on the request.”

“I also use try/catch in routes like /signup to handle unexpected errors and return a 500 fallback, making the server more reliable.”

**🔹 SQLite Integration (6:30 – 7:30)**

“Let me now show you how the project’s database was initialized and structured.

In this init.js file, I define and create three database tables: users, messages, and tours.”

A screenshot of a computer program

AI-generated content may be incorrect.

### 2. Chỉ đoạn insert tour data

A screen shot of a computer code

AI-generated content may be incorrect.

“This section inserts sample tour data into the tours table.

Each record includes the tour’s title, destination, price, duration, and image path.

These are the same tours that appear in the homepage, the carousel, and the search feature.”

**DRAP UP:**

“To wrap up, beyond the main features I’ve demonstrated, I also included several enhancements to improve usability and interactivity.”

“For example, the homepage uses a sticky navigation bar with scrollspy — so users always know where they are.”

“The seasonal tour section auto-rotates between categories like Spring and Winter using JavaScript and timed intervals.”

“I also added a newsletter subscription form at the bottom. It includes client-side email validation to provide instant feedback before submission.”

“To help assessors and developers, there’s a Developer Wiki section summarizing what features were implemented and why — acting as built-in documentation.”

“And if enabled, I also experimented with using localStorage to save a user’s favourite tour locally.”

“These smaller touches help polish the overall experience, and demonstrate how I’ve applied extended UI interactions, data handling, and full-stack logic across this project.”

**This is all my presentation, thank for listening!!!**