**Project Goal:** A simple website where users can see a list of movies currently playing, click on a movie to see its details and showtimes, and perhaps click a "Book" button (we'll keep the actual booking very simple for now, maybe just showing a confirmation message).

**Technology Stack Overview:**

1. **Node.js (Backend):** This is your server environment. It will run JavaScript code to handle requests from users (like asking for the movie list), interact with the database, and send back the web pages. We'll use the **Express.js** framework on top of Node.js because it makes building web servers much easier.
2. **MongoDB (Database):** This is where you'll store your data, like movie titles, descriptions, image URLs, and showtimes. It's a NoSQL database, which means it's flexible with how data is structured. We'll use **Mongoose**, a library that makes working with MongoDB from Node.js simpler.
3. **Bootstrap (Frontend):** This is a CSS framework that provides pre-built components (like buttons, cards, navigation bars) and a grid system to make your website look good and be responsive (work well on different screen sizes) without writing lots of custom CSS from scratch. You'll use this with **HTML** to structure your pages.
4. **Templating Engine (e.g., EJS):** Since you're using Node.js, you'll need a way to dynamically insert data (like movie titles from the database) into your HTML pages before sending them to the user. EJS (Embedded JavaScript) is a popular and easy-to-learn choice that works well with Express.

---

**Phase 1: Learning & Setup (Estimate: 1-3 Weeks)**

- **Goal:** Get your tools ready and understand the fundamentals. Don't rush this phase!
- **Tasks:**
    1. **Install Tools:**
        - **Node.js & npm:** Download and install from the official Node.js website (https://nodejs.org/). npm (Node Package Manager) comes with it. Verify installation using `node -v` and `npm -v` in your terminal/command prompt.
        - **MongoDB:** Install MongoDB Community Server following instructions on the MongoDB website (https://www.mongodb.com/try/download/community). Also, install MongoDB Compass (a graphical tool to view your database). Alternatively, you can use MongoDB Atlas (a free cloud-hosted option) which might be easier initially as you don't need to manage the server yourself.
        - **Text Editor/IDE:** Use a code editor like Visual Studio Code (VS Code) (https://code.visualstudio.com/) - it's free and has great support for web development.
        - **Git & GitHub (Highly Recommended):** Learn basic Git commands (`init`, `add`, `commit`, `push`) and create a GitHub account. This helps you track changes and back up your code.
    2. **Learn Basics:**
        - **HTML & CSS:** Understand how to structure a webpage (HTML) and style it (CSS).
            - *Resources:* MDN Web Docs (HTML) (https://developer.mozilla.org/en-US/docs/Web/HTML), MDN Web Docs (CSS) (https://developer.mozilla.org/en-US/docs/Web/CSS), FreeCodeCamp, W3Schools.
        - **JavaScript (Fundamentals):** Variables, data types, functions, loops, conditional statements, basic DOM manipulation (though we'll minimize this initially with EJS).
            - *Resources:* MDN Web Docs (JavaScript) (https://developer.mozilla.org/en-US/docs/Web/JavaScript), FreeCodeCamp, Eloquent JavaScript (online book).

        - **Bootstrap:** Learn how to use its grid system, components (Cards, Buttons, Navbar), and utilities.
            - *Resources:* Official Bootstrap Documentation (https://getbootstrap.com/docs/5.3/getting-started/introduction/) (check for the latest version).
        - **Node.js & Express.js:** Understand basic server concepts, routing (handling URLs), request/response cycle.
            - *Resources:* Node.js Docs (https://nodejs.org/en/docs/), Express.js Website (https://expressjs.com/), Net Ninja or Traversy Media tutorials on YouTube.
        - **MongoDB & Mongoose:** Understand collections, documents, basic queries (find). Learn how Mongoose uses Schemas and Models.
            - *Resources:* MongoDB Manual (https://docs.mongodb.com/manual/), MongooseJS Docs (https://mongoosejs.com/docs/guide.html), MongoDB University (free courses).

---

**Phase 2: Project Setup & Backend Basics (Estimate: 1-2 Weeks)**

- **Goal:** Create the basic server structure and connect to the database.
- **Tasks:**
    1. **Create Project Folder:** Make a new folder for your project (e.g., `movie-booking-app`).
    2. **Initialize Node.js Project:** Open your terminal in the project folder and run `npm init -y`. This creates a `package.json` file to manage project details and dependencies.
    3. **Install Dependencies:** Run `npm install express mongoose ejs dotenv`.
        - `express`: The web framework.
        - `mongoose`: To interact with MongoDB.
        - `ejs`: The templating engine.
        - `dotenv`: To manage environment variables (like your database connection string) securely.
    4. **Basic Server Setup:** Create a main file (e.g., `server.js` or `app.js`). Write basic Express code to create a server that listens on a port (e.g., 3000).
    5. **Database Connection:** Use Mongoose to connect to your local MongoDB instance or your MongoDB Atlas cluster. Put your connection string in a `.env` file (and add `.env` to your `.gitignore` file!).
    6. **Define Data Structure (Schema):** Create a Mongoose Schema for your movies (e.g., `models/Movie.js`). Define fields like `title` (String), `description` (String), `posterUrl` (String), `showtimes` (Array of Strings or Dates).
    7. **Seed Database:** Manually add a couple of sample movies to your MongoDB database using MongoDB Compass or create a simple script to insert some data using your Mongoose model.

---

**Phase 3: Displaying Movies (Estimate: 1-2 Weeks)**

- **Goal:** Fetch movies from the database and display them on the homepage.
- **Tasks:**
    1. **Setup EJS:** Configure Express to use EJS as the view engine. Create a `views` folder.
    2. **Create Homepage Route:** In `server.js`, create a route for the homepage (`/`). In this route handler:
        - Use your Mongoose `Movie` model to find all movies in the database (`Movie.find({})`).
        - Render an EJS template (e.g., `views/index.ejs`), passing the fetched movie data to it.
    3. **Create Homepage Template (`index.ejs`):**

        - Include Bootstrap CSS/JS.
        - Use EJS syntax (`<% %>`, `<%= %>`) to loop through the movie data passed from the server.
        - Use Bootstrap Cards to display each movie (poster, title). Make each card link to a details page (e.g., `/movies/MOVIE_ID`).

---

**Phase 4: Movie Details Page (Estimate: 1 Week)**

- **Goal:** Show details and showtimes for a specific movie.
- **Tasks:**
    1. **Create Details Page Route:** Create a dynamic route like `/movies/:id`. In this route handler:
        - Get the movie `id` from the URL parameters (`req.params.id`).
        - Use Mongoose to find the specific movie by its ID (`Movie.findById(id)`).
        - Render a details EJS template (e.g., `views/movie-details.ejs`), passing the single movie's data.
    2. **Create Details Template (`movie-details.ejs`):**
        - Include Bootstrap.
        - Display the movie's title, poster, full description, and list its showtimes.
        - Add a "Book" button next to each showtime (for now, these buttons won't do much).

---

**Phase 5: Basic "Booking" Simulation (Estimate: 3-5 Days)**

- **Goal:** Simulate a booking by showing a confirmation message.
- **Tasks:**
    1. **Create Booking Confirmation Route:** Create a route (maybe a `GET` route for simplicity first, like `/booking-confirmation/:movieId/:showtimeIndex`).
        - Fetch the movie details using `movieId`.
        - Get the specific showtime using `showtimeIndex` (or pass the actual time).
        - Render a confirmation template (e.g., `views/confirmation.ejs`).
    2. **Create Confirmation Template (`confirmation.ejs`):**
        - Display a message like "Success! You've 'booked' a ticket for [Movie Title] at [Showtime]."
    3. **Update Details Page:** Make the "Book" buttons link to this new confirmation route, passing the necessary movie ID and showtime information in the URL.

---

**Phase 6: Testing & Refinement (Ongoing)**

- **Goal:** Make sure everything works and fix any bugs.
- **Tasks:**
    - Test all pages and links.
    - Check console logs in the browser and terminal for errors.
    - Add basic error handling (e.g., what if a movie ID doesn't exist?).
    - Maybe add some simple custom CSS to personalize the look.

---

**Beginner-Friendly Timeline Estimate:**

- **Total:** Roughly **4 to 8 weeks**, assuming you're working on this part-time alongside studies and spending significant time on the initial learning phase.

Youtube links

1. nodejs : https://www.youtube.com/watch?v=RLtyhwFtXQA&list=PLWKjhJtqVAbmGQoa3vFjeRbRADAOC9drk (https://www.youtube.com/watch?v=RLtyhwFtXQA&list=PLWKjhJtqVAbmGQoa3vFjeRbRADAOC9drk)
2. boostrap : https://www.youtube.com/watch?v=-qfEOE4vtxE (https://www.youtube.com/watch?v=-qfEOE4vtxE)
3. mongodb : https://www.youtube.com/watch?v=_7UQPve99r4 (https://www.youtube.com/watch?v=_7UQPve99r4)