Health Records Management System Project: Detailed Task Tracking Sheet

Team Member: Jack

- **1.1 Design Interface Layout**
  - [x] ~~Define overall page layout and structure using HTML.~~
  - [ ] Design headers, footers, and navigation sections.
  - [x] ~~Plan and implement a color scheme and font choices for readability.~~
- **1.2 Implement Patient Forms**
- **1.2.1 Add Patient Form**
  - [x] ~~Design form fields for patient information (name, DOB, contact details, insurance, etc.).~~
  - [x] ~~Include validation for required fields and format (e.g., email validation).~~
  - [x] ~~Add submit and reset buttons with visual feedback for successful submission.~~
- **1.2.2 Update Patient Form**
  - [x] ~~Design form to display existing patient data for edits.~~
  - [x] ~~Ensure updated details replace previous data accurately.~~
  - [x] ~~Include confirmation message for successful update.~~
- **1.2.3 Delete Patient Form**
  - [x] ~~Create a button or link to delete records with a confirmation prompt.~~
  - [ ] Ensure error messages appear if the record cannot be deleted.
- **1.2.4 Search Patient Functionality**
  - [x] ~~Design search bar or filter options to locate records by criteria (name, DOB, etc.).~~
  - [ ] Display results in a user-friendly table or list with pagination, if necessary.
- **1.3 Ensure Responsiveness**
  - [ ] Test and adjust layout for different screen sizes (desktop, tablet, mobile).
  - [ ] Apply media queries in CSS for a smooth user experience on all devices.

- **1.4 Usability Testing and Feedback**

  - [ ] **Conduct usability testing with sample users to gather feedback on navigation and layout.**
  - [ ] **Document usability issues, if any, and apply fixes to improve user experience.**

- **1.5 Final Adjustments**

  - [ ] **Re-check for consistency in fonts, colors, and layout across pages.**
  - [ ] **Perform cross-browser testing to ensure compatibility (Chrome, Firefox, Safari).**

Team Member: Husam

- **2.1 Project Setup**

  - [ ] **Create the NodeJS project structure (folders for routes, controllers, etc.).**
  - [ ] **Install required dependencies (Express.js, body-parser, MySQL connectors, etc.).**

- **2.2 Route Setup for CRUD Operations**

- **2.2.1 Add Patient**

  - [ ] **Define a route to handle new patient record creation requests.**
  - [ ] **Validate input data before inserting it into the database.**
  - [ ] **Send confirmation response to the frontend upon successful insertion.**

- **2.2.2 Update Patient**

  - [ ] **Define a route to process updates to existing patient records.**
  - [ ] **Verify the record exists before allowing updates.**
  - [ ] **Confirm successful updates back to the frontend.**

- **2.2.3 Delete Patient**

  - [ ] **Define a route to handle deletion requests for patient records.**
  - [ ] **Confirm deletion with the database before sending a success message.**
  - [ ] **Handle errors if the record cannot be found or deleted.**

- **2.2.4 Retrieve/Search Patient Records**

  - [ ] **Set up a route for fetching specific records based on search criteria.**
  - [ ] **Ensure efficient query handling for large datasets.**
  - [ ] **Return results in a structured format (e.g., JSON) to the frontend.**

- **2.3 Error Handling and Data Security**
  - ☐ **Implement server-side validation for all input data.**
  - ☐ **Set up error-handling middleware to manage unexpected errors gracefully.**
  - ☐ **Sanitize data inputs to prevent SQL injection and cross-site scripting attacks.**
- **2.4 Integration Testing**
  - ☐ **Test each route independently to ensure CRUD functionality works as expected.**
  - ☐ **Test end-to-end functionality by integrating with Jack's frontend forms.**

Team Member: Abdullah

- **3.1 Design ER Diagram and Relational Model**
  - ☐ **Sketch an ER diagram to represent the data structure for patient records.**
  - ☐ **Determine relationships (e.g., 1:1, 1:M) and define primary and foreign keys. With making assumptions**
  - ☐ **Convert the ER diagram into a relational model for MySQL.**
- **3.2 Database Table Creation**
  - ☐ **Write SQL scripts to create tables with appropriate columns for patient data (name, DOB, etc.).**
  - ☐ **Define primary keys for unique identification of records.**
  - ☐ **Set up foreign keys and constraints as necessary.**
- **3.3 SQL Query Development**
- **3.3.1 Insert Query**
  - ☐ **Write an SQL query for adding new patient records into the database.**
  - ☐ **Test query with sample data to confirm data insertion works as expected.**
- **3.3.2 Update Query**
  - ☐ **Develop an SQL query to modify existing patient records (e.g., update discharge date).**
  - ☐ **Test update functionality to verify that changes are saved correctly.**
- **3.3.3 Delete Query**

- ☐ **Write an SQL query to delete specific patient records.**
- ☐ **Ensure constraints prevent accidental deletion of non-related records.**

· **3.3.4 Select Query**

- ☐ **Write queries for retrieving patient records based on various criteria.**
- ☐ **Optimize queries to handle potential large datasets efficiently.**

· **3.4 Database-Backend Integration**

- ☐ **Test SQL queries within NodeJS to verify smooth interaction between backend and database.**
- ☐ **Implement error handling for database connection issues.**
- ☐ **Validate that database responses are sent back accurately to the backend.**

· **3.5 Database Optimization**

- ☐ **Index frequently queried columns to improve retrieval speed.**
- ☐ **Review and optimize queries to ensure database performance is high.**