**Name: Rajvee Vaghela**

**Databases and the Web**

Mini Project Assignment Documentation

1. **An outline describing the application you have built (max 200 words)**

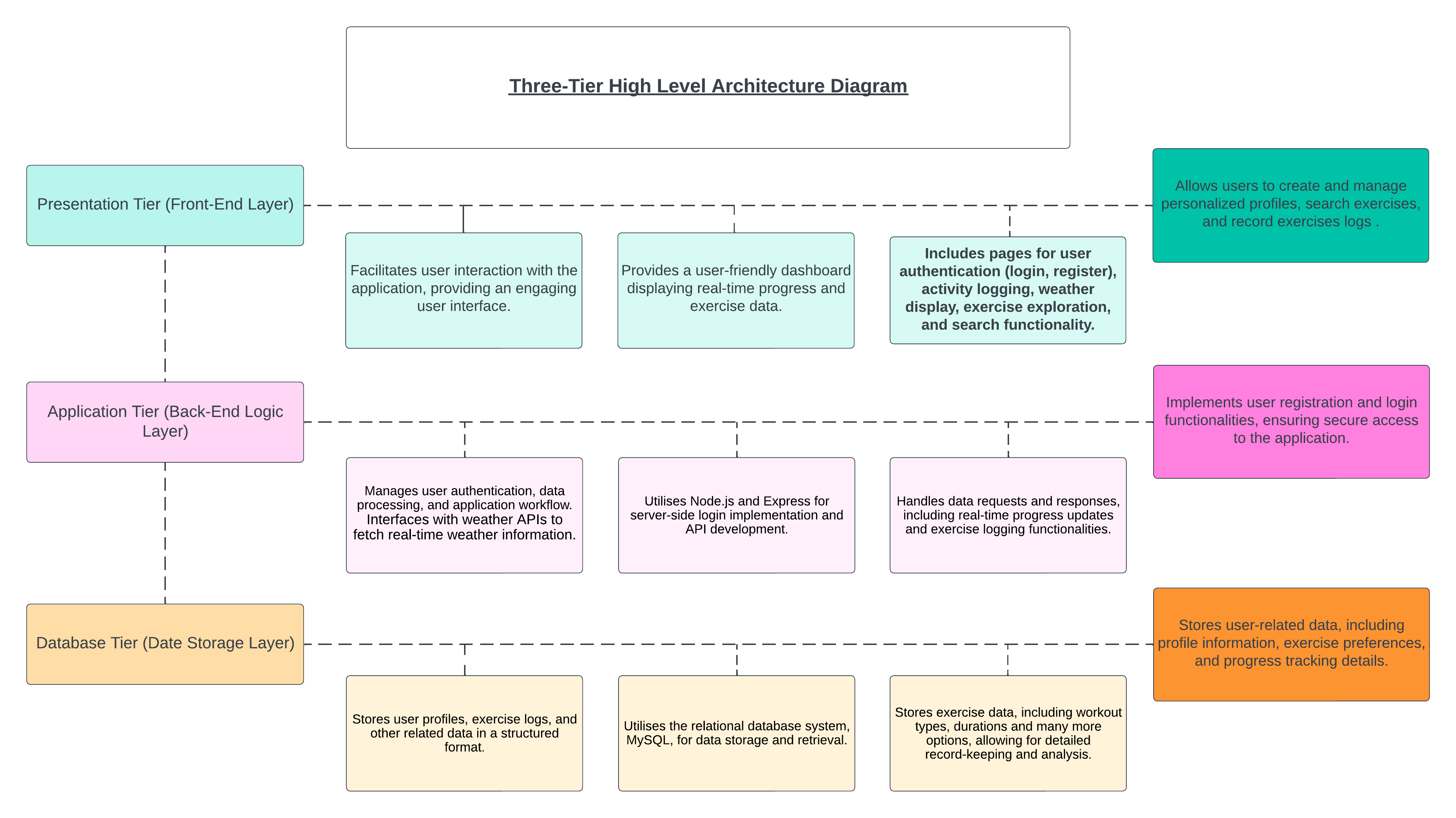
The name of my application is Worth the Weight, a health and fitness tracker.

The application includes a user authentication system, allowing users to log in securely, register new accounts, and seamlessly log out. I chose to maintain personalized sessions by displaying the user's name on pages where it is relevant.

The forum's core functionality is to allow the user to maintain a log and keep track of the exercise they perform over time. Users can log their exercises, including details such as exercise name, date, and duration. The system ensures data integrity by validating usernames before recording exercise logs. Additionally, users can view and edit their logged activities, a feature I decided to incorporate in the application to enhance the application's usability.

The application incorporates a weather-based external API to fetch real-time weather information. This information is dynamically presented on the forum, providing users with current temperature, weather conditions, wind speed, precipitation, and a time stamp. I also decided to add a personalised recommendation feature based on the weather information, suggesting to the user whether outdoor activities are favourable.

Furthermore, the forum includes pages for exploring and listing exercises, along with a search functionality for finding specific exercises.

1. **A high-level architecture including a diagram and description (max 100 words) describing what technologies and components you have used in your application tier and data tier**

In the application tier, I employed Express.js as the web server framework to handle HTTP requests.

User authentication is implemented securely with bcrypt for password encryption and Express-validator for input validation during user registration.

The data tier relies on MySQL as the relational database management system.

User details, exercise logs, and forum discussions are stored in distinct tables.

External APIs are integrated to fetch real-time weather data, enhancing user experience by dynamically presenting weather information.

1. **A data model including a diagram and description (max 100 words)**

A diagram of a computer program

Description automatically generated

The data model consists of a ‘**users’** table, a table to search exercises, ‘search\_exercise’, and a table called ‘exercise\_log’.

Each user is uniquely identified by an auto-incremented user\_id. The table captures user details, including username, firstname, lastname, email, gender, age, and a securely hashed password.

The username column is indexed for optimized retrieval.

In the ‘**search\_exercise**’ table each exercise is uniquely identified by an auto-incremented exercise\_id.

The table facilitates the organization and retrieval of a diverse range of exercises, supporting user exploration and selection.

The ‘**exercise\_log**’ table is responsible for recording individual exercise sessions logged by users.

Each log entry has a unique identifier (log\_id) and is associated with a specific user through the username field, establishing a relationship with the ‘users’ table.

This table enables users to track and review their exercise history.

1. **A description of the user-facing functionality of your application, adding screenshots to help explain (max 500 words)**

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated | The **index/home page**, integrated with all the buttons of the application. |
|  | **Button: Record Activity**  This page is to create exercise logs.  The user will only be able to make an entry after logging in and using a registered username. |
|  | **Button: Activity Logs**  After a user succeeds in creating a log entry, it is displayed on this page.  Using the **Edit** button, the user also has the option to edit the log, except for the username. |
|  | **Button: Search Activity**  This page is a search for an activity saved in the exercises list database. |
|  | Button: Exercise List  This page displays pre-stored exercises the users can search. |
|  | **Button: About**  The about page. |
|  | **Button: Weather**  This page displays information from the Weather API and recommends whether the user should exercise indoors or outdoors. |

1. **A description of how you have addressed security risks in your application (max 500 words)**

List of ways I addressed security risks in my application:

* **Password Hashing:**

User passwords are securely hashed using the bcrypt hashing algorithm before being stored in the database. This ensures that even if the database is compromised, sensitive user passwords remain protected.

* **User Input Validation:**

Express-validator is employed to validate and sanitize user inputs, preventing common security vulnerabilities such as SQL injection and cross-site scripting (XSS). This helps mitigate the risk of malicious input compromising the application.

* **Cookie Security:**

User sessions are managed securely using HTTP cookies. Sensitive information is stored in cookies with appropriate security attributes, such as the 'HttpOnly' flag, to prevent client-side script access.

* **HTTPS Protocol:**

The application is configured to use HTTPS to encrypt data during transmission, safeguarding against potential eavesdropping and man-in-the-middle attacks.

* **Database Prepared Statements:**

Prepared statements are utilized when interacting with the database to prevent SQL injection attacks. Parameterized queries ensure that user inputs are treated as data, not executable code.

* **User Authentication:**

Robust user authentication mechanisms are implemented, including secure password storage, session management, and secure transmission of authentication data.

* **Limited Access to Resources:**

Access controls and restrictions are enforced to ensure that users can only access resources and functionalities for which they have the appropriate permissions, minimizing the risk of unauthorized access.

* **External API Security:**

When interacting with external APIs, secure practices such as validating API responses and ensuring API keys are kept confidential are followed to prevent data manipulation and unauthorized access.

These security measures collectively create a defence-in-depth strategy, addressing various potential risks and enhancing the overall security posture of my application.

1. **Details of how to use your API**

I created my API file in a folder called API. The name of the file is app.js.

In this simple example of how to use my API, I created the following data collections:

A screenshot of a computer program

Description automatically generated

And below are the three routes to print this information in the browser:

A screenshot of a computer code

Description automatically generated