

Fitness Tracker Web Application

by

CS 157A

Nam Bui ID#015794363
Alvin Lee ID#016377491
Andy Liu ID#016057782
Karthik Renuprasad ID#0216216278

Email:

nam.bui@sjsu.edu
alvin.lee02@sjsu.edu
andy.k.liu@sjsu.edu
karthik.renuprasad@sjsu.edu

Goal and Description

Goal:

Our goal is to develop a comprehensive fitness tracker web application that integrates into users' daily routines and empowers them to achieve their health and fitness objectives. We aim to provide a user-friendly platform that motivates individuals to lead healthier lifestyles by tracking their physical activities and monitoring their progress. We aspire to create a robust digital companion that fosters accountability, encourages goal setting, and helps users reach their fitness aspirations, whether they aim to lose weight, improve endurance, or simply maintain a balanced lifestyle.

Description:

Our web application empowers users to set personalized fitness goals tailored to their aspirations and preferences. Whether aiming to lose weight, increase endurance, or build muscle, users can define specific targets and milestones within the application's intuitive interface. Tracking progress becomes easy as users update their achievements, monitoring their advancements toward these objectives.

Not limited to mere activity logging, our application offers a comprehensive overview of users' fitness endeavors. With the ability to track various metrics including duration, intensity, and calories burned for each activity session, users gain invaluable insights into their workout effectiveness and overall fitness journey. Whether it's a morning run in the park or an after-work workout, users can effortlessly record their activities and corresponding locations, providing relevant information for their fitness growth.

Functional Requirements & Application

Architecture

Functional Requirements:

1) User Account Management

a) Users can create an account

- i) The account creation process should include fields for username, password, first name, last name, email, and any additional information

b) Users can log in and out of their account

2) Activity Management

a) Users can create new activities

- i) Fields for users to input activity information include the activity name, the duration of the activity, and the calories burnt from the activity

b) Users can track activities previously done

3) Location Management

a) Users can view all locations they use for activities

b) Users can add locations where they do physical exercises or activities

- i) Fields for users to input location information include the Street Address, the City, the State, and any other additional information

4) Fitness Goal Management

a) Users can view their fitness goals

- b) Users can create Fitness Goals
 - i) The input for the creation of a fitness goal includes a textbox for a description of the goal
- 5) Profile Management
 - a) Users can view their profile page
 - b) Users can write Milestones in the Fitness Journey
 - i) The input for the creation of Milestones includes a textbox for a description of the milestone
 - c) Users can view comments made about the user's profile
 - d) Users can comment on other users' profile
 - i) The input for the comment includes a textbox for the comment

Application Architecture:

The Fitness Tracker web application uses the Node.js runtime environment and Express.js frameworks. The application adopts a Model-View-Controller (MVC) design pattern to separate concerns and provide a maintainable codebase. At the heart of the architecture lies the Model layer, which handles data manipulation and interactions with the MySQL database. This layer encapsulates the logic for managing user profiles, exercising SQL queries, and ensuring data integrity and consistency. The View layer, implemented using HTML, CSS, and templating engines like Express.js, dynamically renders user interfaces, providing an intuitive and responsive user experience. Finally, the Controller layer acts as an intermediary between the Model and View, orchestrating user requests, processing business logic, and rendering appropriate responses.

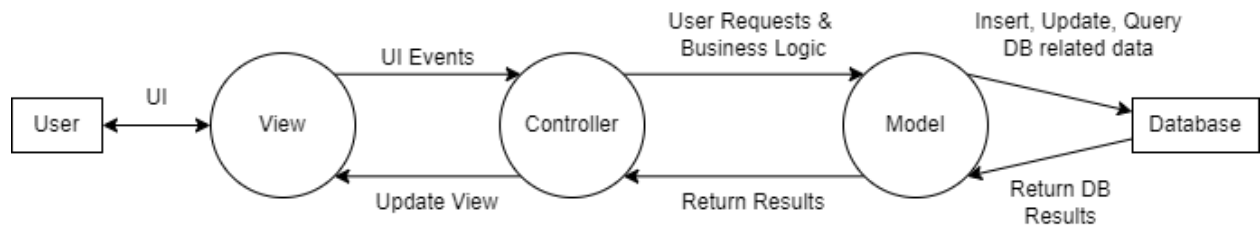


Figure 1: MVC Diagram of Fitness Tracker Application

ER Data Model

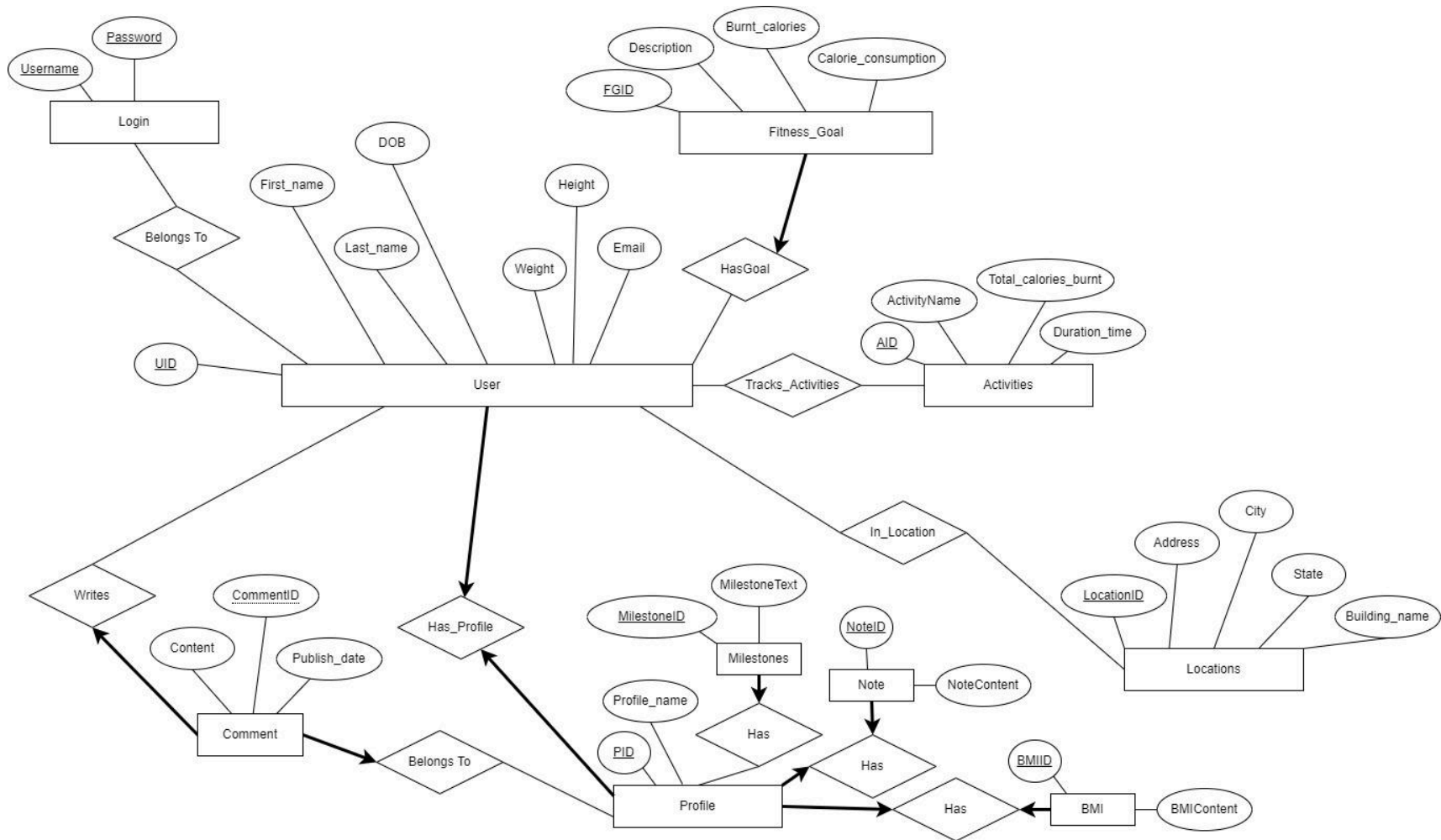


Figure 2: ER Diagram of Fitness Tracker Database

Database Design

Relational Schema:

User(Integer: UserID, String: FName, String: LName, Date: DOB, String: Weight, String: Height, String: Email)

Login(String: Username, String: Password, Integer: UserID)
Foreign Key: UserID referencing User.UserID

FitnessGoal(Integer: FitnessGoalID, Integer: UserID, String: Descriptions, Integer: CaloriesBurnt, Integer: CaloriesConsumed)
Foreign Key: UserID referencing User.UserID

Location(Integer: LocationID, String: Address, String: City, String: State, String: BuildingName)

Activities(Integer: ActivityID, String: ActivityName, Integer: DurationTime, Integer: TotalCaloriesBurnt)

TracksActivities(Integer: UserID, Integer: ActivityID)
Foreign Key: UserID referencing User.UserID
Foreign Key: ActivityID referencing Activities.ActivityID

ActivityLocation(Integer: UserID, Integer: LocationID)
Foreign Key: UserID referencing User.UserID
Foreign Key: LocationID referencing Location.LocationID

Profile(ProfileID, String: ProfileName)
Foreign Key: ProfileID referencing User.UserID

Milestones(Integer: MilestoneID, Text: Milestone)

ProfileHasMilestones(Integer: ProfileID, Integer: MilestoneID)
Foreign Key: ProfileID referencing Profile.ProfileID
Foreign Key: MilestoneID referencing Milestones.MilestoneID

BMI(Integer: BMIID, Text: BMIContent)

```
CREATE TABLE Location (
```



```
LocationID INT AUTO_INCREMENT PRIMARY KEY,  
Address VARCHAR(255),  
City VARCHAR(255),  
State VARCHAR(255),  
BuildingName VARCHAR(255)  
);
```

```
CREATE TABLE Activities (  
    ActivityID INT AUTO_INCREMENT PRIMARY KEY,  
    ActivityName VARCHAR(255),  
    DurationTime INT,  
    TotalCaloriesBurnt INT  
);
```

```
CREATE TABLE TracksActivities (  
    UserID INT,  
    ActivityID INT,  
    PRIMARY KEY(UserID, ActivityID),  
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE,  
    FOREIGN KEY (ActivityID) REFERENCES Activities(ActivityID) ON DELETE  
CASCADE  
);
```

```
CREATE TABLE ActivityLocation (  
    UserID INT,  
    LocationID INT,  
    PRIMARY KEY (UserID, LocationID),  
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE,  
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID) ON DELETE  
CASCADE  
);
```

```
CREATE TABLE Profile (  
    ProfileID INT PRIMARY KEY,  
    ProfileName VARCHAR(255),  
    FOREIGN KEY (ProfileID) REFERENCES User(UserID) ON DELETE CASCADE  
);
```

```
CREATE TABLE Milestones (  
    MilestoneID INT AUTO_INCREMENT PRIMARY KEY,
```

```

    Milestone TEXT
);

CREATE TABLE ProfileHasMilestones (
    ProfileID INT,
    MilestoneID INT,
    PRIMARY KEY(ProfileID, MilestoneID),
    FOREIGN KEY (ProfileID) REFERENCES Profile(ProfileID) ON DELETE CASCADE,
    FOREIGN KEY (MilestoneID) REFERENCES Milestones(MilestoneID) ON DELETE
    CASCADE
);

CREATE TABLE BMI (
    BMIID INT PRIMARY KEY,
    BMIContent TEXT,
    FOREIGN KEY (BMIID) REFERENCES Profile(ProfileID) ON DELETE CASCADE
);

CREATE TABLE Note (
    NoteID INT PRIMARY KEY,
    NoteContent TEXT,
    FOREIGN KEY (NoteID) REFERENCES Profile(ProfileID) ON DELETE CASCADE
);

CREATE TABLE Comment (
    CommentID INT AUTO_INCREMENT PRIMARY KEY,
    ProfileID INT,
    Content TEXT,
    UserID INT,
    PublishDate DATE,
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE,
    FOREIGN KEY (ProfileID) REFERENCES Profile(ProfileID) ON DELETE CASCADE
);

```

Analysis of the Functional Dependencies

User Table

- Attributes: UserID, FName, LName, DOB, Weight, Height, Email
- Primary Key: UserID
- Functional Dependencies:
 - UserID → FName
 - UserID → LName
 - UserID → DOB
 - UserID → Weight
 - UserID → Height
 - UserID → Email

Login Table

- Attributes: Username, Password, UserID
- Primary Key: (Username, Password)
- Functional Dependencies:
 - Username, Password → UserID

FitnessGoal Table

- Attributes: FitnessGoalID, UserID, Descriptions, CaloriesBurnt, CaloriesConsumed
- Primary Key: FitnessGoalID
- Functional Dependencies:
 - FitnessGoalID → UserID
 - FitnessGoalID → Descriptions
 - FitnessGoalID → CaloriesBurnt
 - FitnessGoalID → CaloriesConsumed

Location Table

- Attributes: LocationID, Address, City, State, BuildingName
- Primary Key: LocationID
- Functional Dependencies:
 - LocationID → Address
 - LocationID → City
 - LocationID → State
 - LocationID → BuildingName

ActivitiesTable

- Attributes: ActivityID, ActivityName, DurationTime, TotalCaloriesBurnt
- Primary Key: ActivityID
- Functional Dependencies:
 - ActivityID \rightarrow ActivityName
 - ActivityID \rightarrow DurationTime
 - ActivityID \rightarrow TotalCaloriesBurnt

TracksActivities Table

- Attributes: UserID, ActivityID
- Primary Key: (UserID, ActivityID)
- Functional Dependencies: no dependencies because it is a junction table

ActivityLocation Table

- Attributes: UserID, LocationID
- Primary Key: (UserID, LocationID)
- Functional Dependencies: no dependencies because it is a junction table

Profile Table

- Attributes: ProfileID, ProfileName
- Primary Key: ProfileID
- Functional Dependencies:
 - ProfileID \rightarrow ProfileName

Milestones Table

- Attributes: MilestoneID, Milestone
- Primary Key: MilestoneID
- Functional Dependencies:
 - MilestoneID \rightarrow Milestone

ProfileHasMilestones Table

- Attributes: ProfileID, MilestoneID
- Primary Key: (ProfileID, MilestoneID)
- Functional Dependencies: no dependencies because it is a junction table

BMI Table

- Attributes: BMIID, BMIContent
- Primary Key: BMIID
- Functional Dependencies:
 - BMIID \rightarrow BMIContent

Note Table

- Attributes: NoteID, NoteContent
- Primary Key: NoteID
- Functional Dependencies
 - NoteID \rightarrow NoteContent

Comment Table

- Attributes: CommentID, ProfileID, Content, UserID, PublishDate
- Primary Key: CommentID
- Functional Dependencies:
 - CommentID \rightarrow ProfileID
 - CommentID \rightarrow Content
 - CommentID \rightarrow UserID
 - CommentID \rightarrow PublishDate

ALL TABLES FOLLOW BCNF

Major Design Decisions

Throughout the development of the Fitness Tracker Application, many decisions made a critical impact on the project. The first major decision was to switch from JavaFX to Node.js and Express.js. This change happened due to a requirement of the project that needed it to be based on a web application. As the decision to use JavaFX was made a period before the discovery of the missed requirement, our team had spent time setting up our workspace to be able to use JavaFX. The sudden change to Node.js and Express.js. forced us to learn the web application framework as we developed our Fitness Tracker Application.

The second major decision was on the ER diagram. A final review of our ER diagram made us rethink the relations in the diagram. It ended with the removal of some entities and relations along with the addition of new entities and relations. Another design change was with the database design. As we were developing the application, we realized some of the relations and connections between relations were illogical or did not fit the application's requirements and needs. As such the changes made to the database tables had to be reflected in the relational schema and even the ER diagram.

Implementation

The Fitness Tracker Application is built on a robust tech stack comprising Node.js, Express.js, and MySQL for functionality and efficient data management. Node.js was used for server-side operations and Express.js for streamlined routing to ensure swift and responsive performance for the application. MySQL was used as the database management system and enables secure and reliable storage and retrieval of user data. The coding process was done through Visual Studio Code, offering a comprehensive and user-friendly environment for our team to craft clean and maintainable code.

In the Fitness Tracker Application, accessing the MySQL database through Express.js is streamlined and efficient, thanks to the implementation of the `mysql2` npm package. This process begins with establishing a connection to the MySQL database using the `mysql2` module within the Express.js application. Once the connection is established, Express.js routes are configured to handle various HTTP API requests, such as GET, POST, PUT, and DELETE, corresponding to CRUD operations. When a request is made to retrieve or manipulate data, Express.js routes utilize SQL queries to interact with the MySQL database. These queries are structured to fetch or modify the desired data based on the request parameters. The results from these queries are then processed and returned to the client as HTTP responses.

The Fitness Tracker Application uses multiple indexes on several tables to optimize the performance of the application's queries. We have decided to create indexes for tables User, FitnessGoal, Location, Activities, Profile, and Milestones, which act on UserID, FitnessGoalID, LocationID, ActivitiesID, ProfileID, and MilestoneID

respectively. The type of index we used is a B+ tree because the application is expected to have thousands of tuples in each table and they are queried multiple times.

Demonstration

Fitness Tracker

[Profile](#)[Activities](#)[Goals](#)[Locations](#)[View Profile](#)[Login/Logout](#)

User Login

Username:

Password:

[Create Account](#)

Login

First, we click “Create Account”.

User Information Form

Username:

Password:

First Name:

Last Name:

Date of Birth:

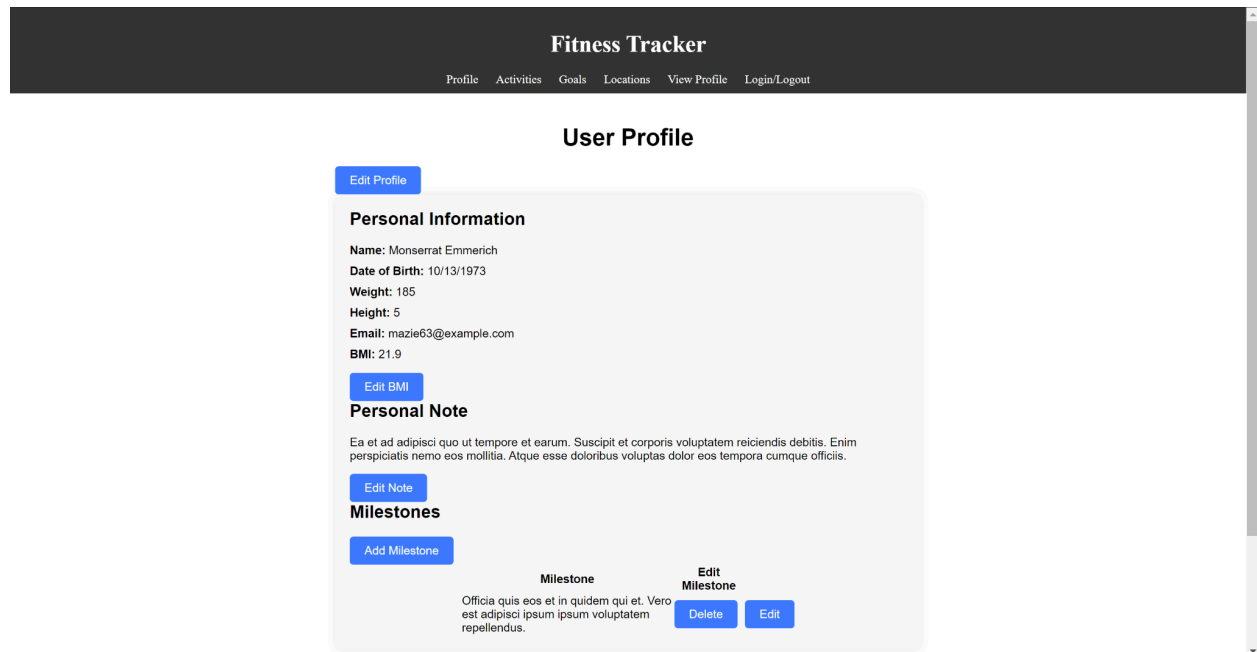
Weight:

Height:

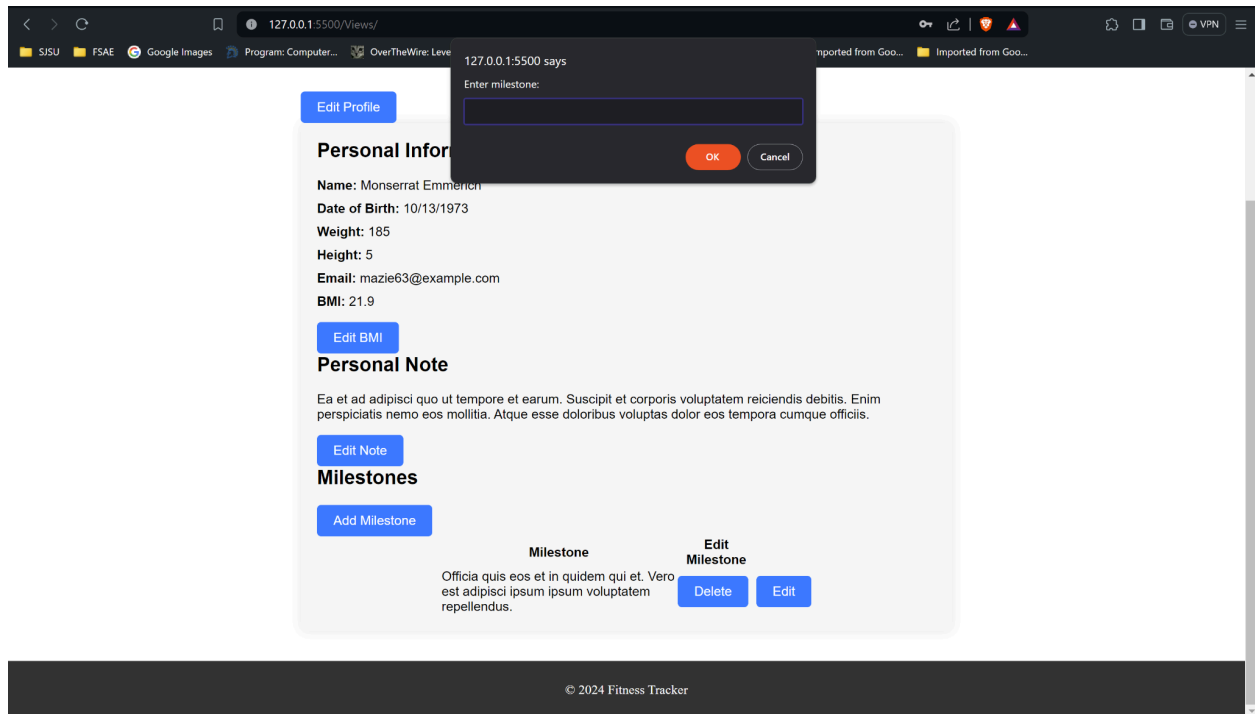
Email:

Create Account

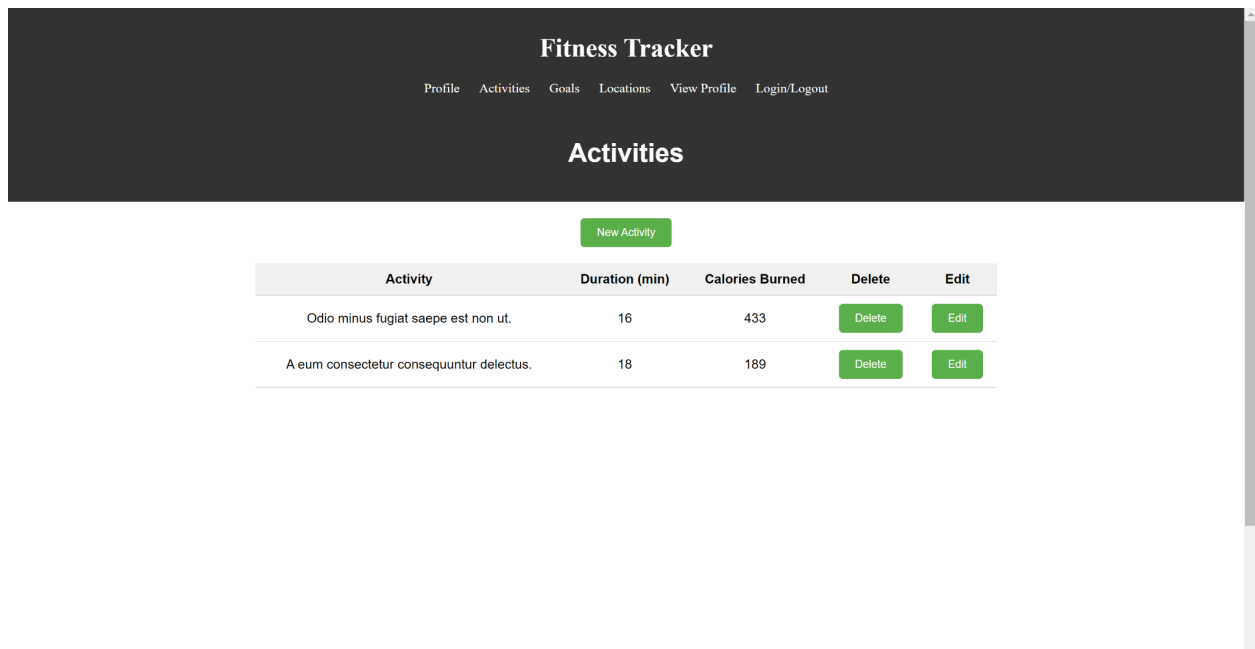
Here you can create an account, and then you can close this tab to return to the login page.



We use an example user to log in. Then, we land on the Profile Page where we can Edit Profile, Edit BMI, Add Milestone, Delete Milestones, and Edit Milestones.



Here, I have clicked “Add Milestone”, and every add and edit button click will cause a pop-up to appear in which you will be asked to add/edit rows.



After clicking “Activities” in the header, we come to the activities page. Here we can add, edit, and delete activities. Adding and Editing are similar to Milestones.

Fitness Tracker

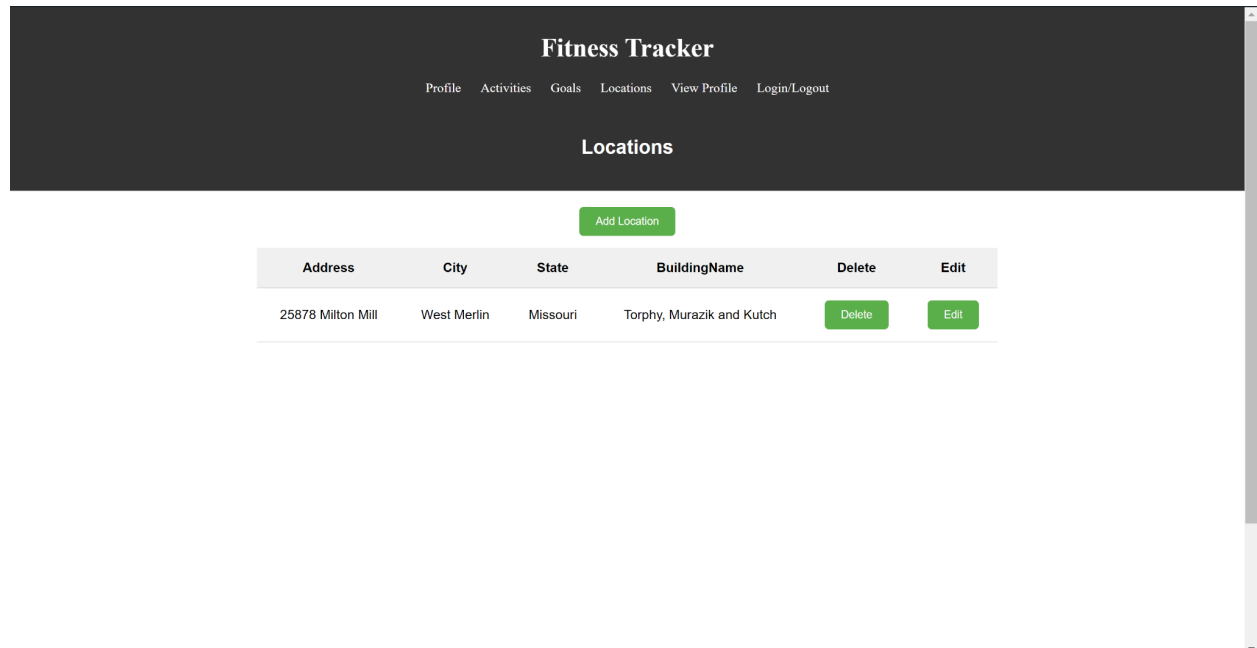
[Profile](#)[Activities](#)[Goals](#)[Locations](#)[View Profile](#)[Login/Logout](#)

Fitness Goals

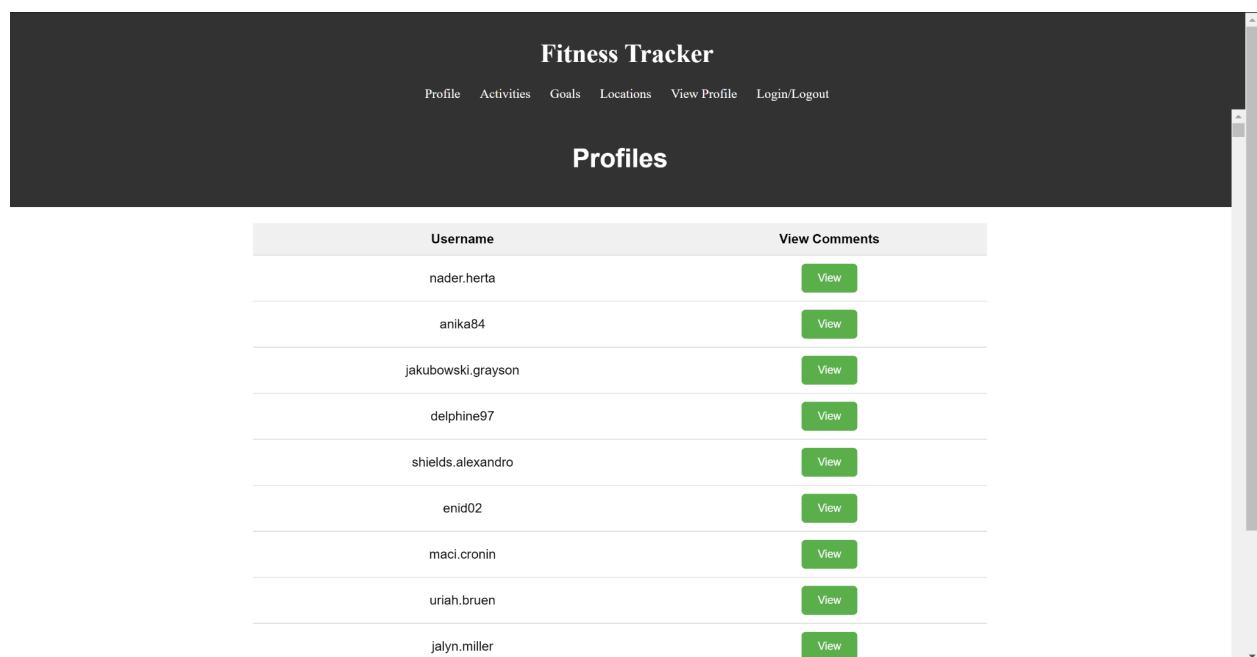
Add Fitness Goal

Description	Calories Burn Goal	Calories Consumption Goal	Delete	Edit
Commodi illo vero reprehenderit deserunt animi perspicatis facere. Ellgendi rem magni alias itaque id cumque excepturi veniam. Molestias dolor iure quaerat sit voluptatibus.	652	1931	Delete	Edit

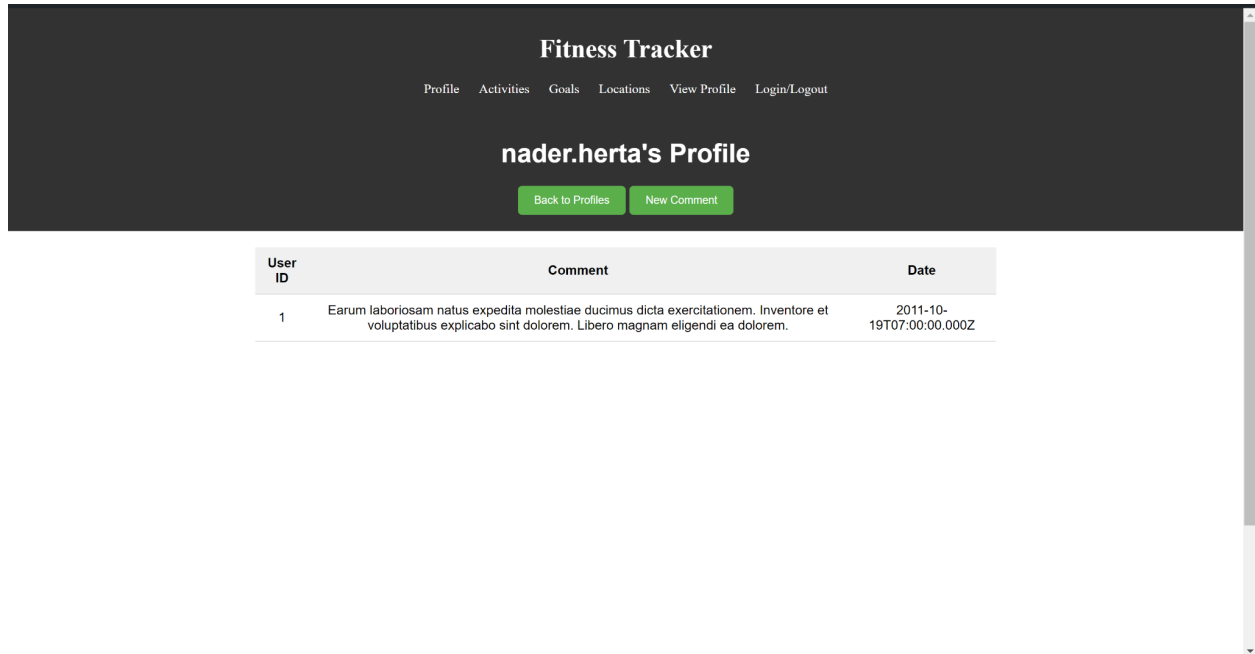
After clicking “Goals” in the header, we come to the Fitness Goals page. Here we can add, edit, and delete goals. Adding and Editing are similar to Milestones.



After clicking “Locations” in the header, we come to the Locations page. Here we can add, edit, and delete Locations. Adding and Editing are similar to Milestones.



After clicking “View Profile” in the header, we come to the Profiles Page. Here we can click on View Comments for a specific Profile.



The screenshot displays the 'Fitness Tracker' application interface. At the top, a dark header contains the title 'Fitness Tracker' and a navigation menu with links: Profile, Activities, Goals, Locations, View Profile, and Login/Logout. Below the header, the page title 'nader.herta's Profile' is centered. Underneath the title are two green buttons: 'Back to Profiles' and 'New Comment'. A table with three columns—User ID, Comment, and Date—is positioned below the buttons. The table contains one row of data. The User ID is 1, the Comment is a long Latin sentence, and the Date is 2011-10-19T07:00:00.000Z.

User ID	Comment	Date
1	Earum laboriosam natus expedita molestiae ducimus dicta exercitationem. Inventore et voluptatibus explicabo sint dolore. Libero magnam eligendi ea dolore.	2011-10-19T07:00:00.000Z

After clicking user nader.herta’s View Comment, we come to nader.herta’s profile’s comments. Here we can add a new comment similar to how we add new milestones. We can also click “Back to Profile” to return to the previous page.

Fitness Tracker

ProfileActivitiesGoalsLocationsView ProfileLogin/Logout

User Login

Username:

Password:

Create Account

Login

After you click Login/Logout, you return to the Login page.

Conclusion

In conclusion, the Fitness Tracker Web Application provides users with a platform for tracking their fitness goals and progress using Node.js, Express.js, and MySQL. Visual Studio Code was used as the development environment to provide efficiency and maintainability throughout the coding process. With the integration of the mysql2 npm package, the application effortlessly accesses and manages data in the MySQL database through SQL queries.

Throughout the development process, the Fitness Tracker Application encountered several challenges, including a sudden change in project design, time constraints, and a learning curve with Node.js and Express.js. However, these challenges provided valuable lessons learned. We discovered the importance of carefully reviewing project requirements and gained proficiency in utilizing Node.js and Express.js for web development.

Looking to the future, improvements for the application include enhancing the user interface to elevate the user experience, implementing pre-built exercise plans for user convenience, integrating activity location tracking, and adding features for users to monitor their BMI, height, and weight. Additionally, we plan to strengthen the application's security by enhancing the login page to encrypt user data and adhere to modern security practices. These improvements aim to enrich the functionality and usability of the Fitness Tracker Application, ensuring it continues to meet the evolving needs of fitness enthusiasts.