Workout Tracker Application

Hosted Application: https://hilarious-wasp-kilt.cyclic.app/
Repository:https://github.com/AhubSajid/AE1_AD_Final

Introduction

My whole life I have struggled with my weight and have always felt hopeless when it came to getting in-shape. Like many others I've had my moments where I started excercising more and eating less in an attempt to change this however often I felt like my routine was all over the place and questioned whether I was actually achieving anything. In 2022, I started going to the gym and st first my aim was to do unplanned workouts, mostly involving cardio, until I lost some pounds and then switch to a more focused approach. I am now at that point where I can focus on building muscle and getting in better shape so this was the main inspiration for my project.

I have decided to create a tracker for my workouts so that I can keep note of what workouts I have done and keep track of my progress. As well as this benefitting me I will also give access to my gym friends to boost the competition we have between each other on who can get the best physic by the summer as there is a lot of money on the line.

System Overview

My project is built using Model-View-Controller(MVC) architecture which uses Node.js for all the backend code and then using MongoDB databases via EJS to create the views. Node.js is a tool that makes developing code easier and faster which was the main reason I have used it alongside EJS as it meant I was able to preview the changes as I do them as oppose to waiting for the program to compile and run again.

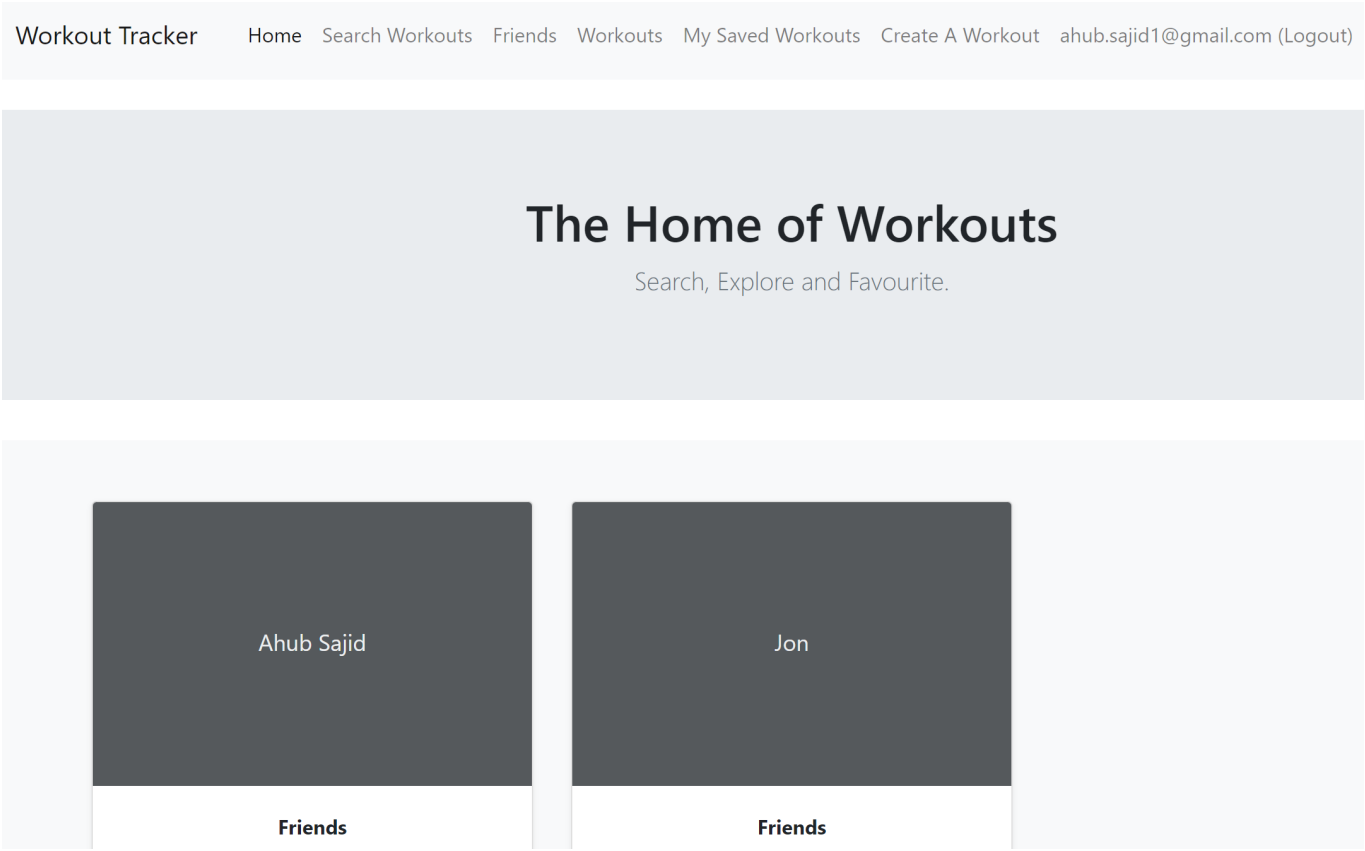
The module revolved around the use of MongoDB so it seemed clear that that was going to be the technology that I was going to use to interact with the databases. Of course, MongoDB uses non-realtional databases which can have their pros and cons. The advantages of using non-relational databases is the scalabilty due to the lack of structure in the data and the simplicity of the code required for it. On the other hand, there is a big disadvantage as non-realtional databases can lead to a compromise in data integrity and consistency, however in the scenario of my project this is not a problem.

In terms of the architecture of the system, the reason for choosing to use MVC is the fact that it is very easy to maintain the system no matter the size and this has been proven as a lot of very big websites such as GoDaddy.com or Ancestry.com use this technology.

To make the system serverless and widely usable I have used Cyclic. The benefits of using Cyclic to deploy the application is Cyclic apps never sleeps so all front-ends and back-ends are ready on-demand at all times. Cyclic apps can be run worldwide without any overload as on Cyclic, serverless functions are allocated to each individual request on demaind, making it possible for apps to hyper scale. [2]

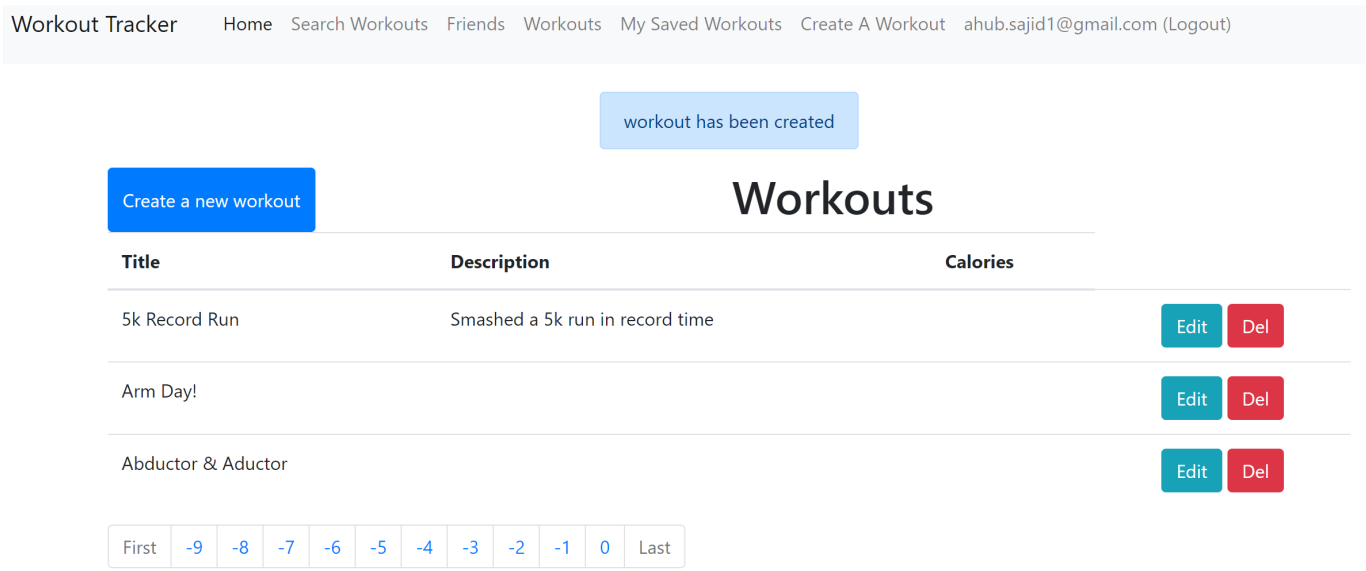
Home

The home page is the most important page for a website as it is the first view the user gets of your site. My home page includes a main header that states the functionality of the website and for added looks I have also decided to display the friends in boxes allowing the user to view each friend by clicking on them. I have also included both a header and footer. The header is the navigation bar for the website and includes links to all pages as well as displaying the login information of the user and gives them the ability to register, login and then logout once logged in.



Workouts

The workouts page shows a list of all workouts that have been entered into the system. I have also added pagination which means that the pages load a lot faster as there is less data that needs to be loaded per page. The workouts are displayed using their title, a description and friend name and the user is given the option for each workout to edit or delete the workout which will then re-direct them to the appropriate page. As well as this, the user can also add a workout.

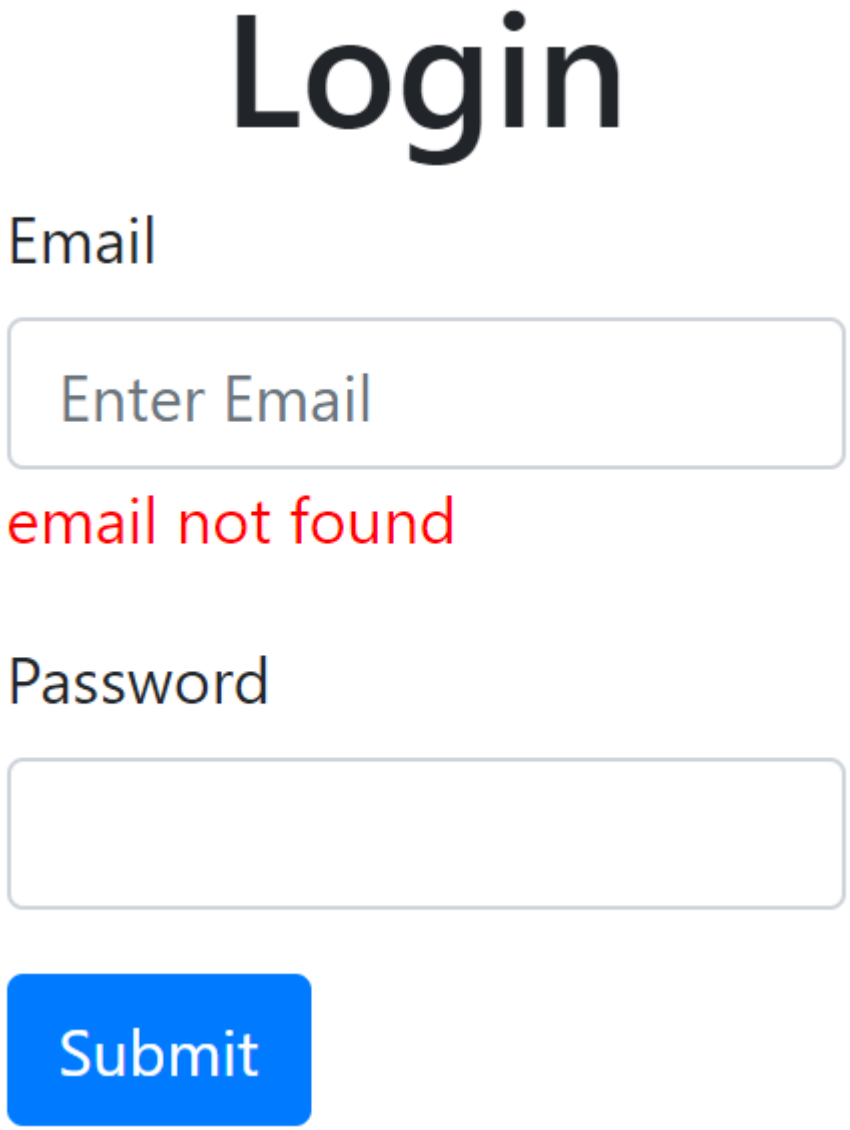


Friends

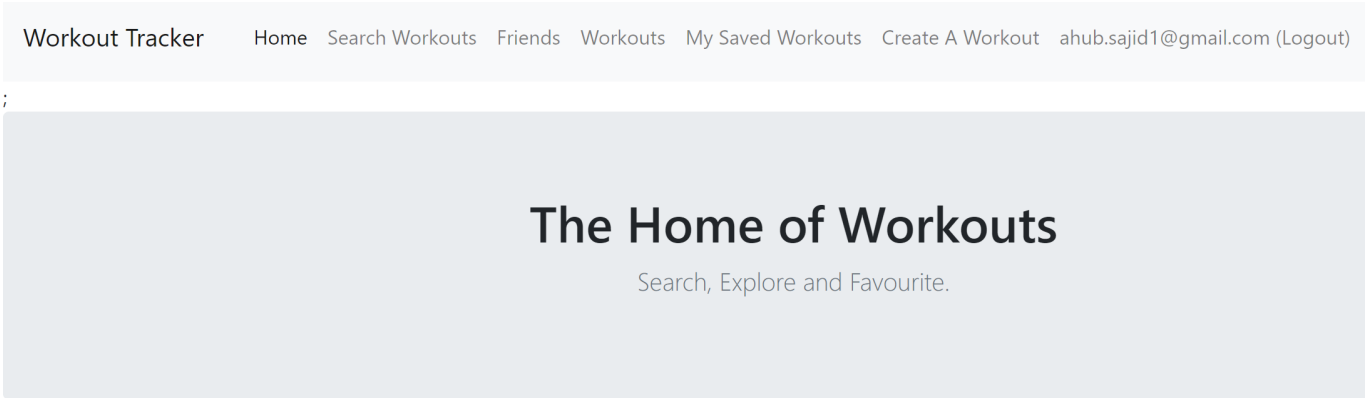
The friends page is used to display a list of gym friends that have been retrieved from the database. The friend's ID, name and number of workouts is displayed as well as buttons that allow the user to edit or delete the friend. The user also has the ability to add a new friend from this page.

Login and Register

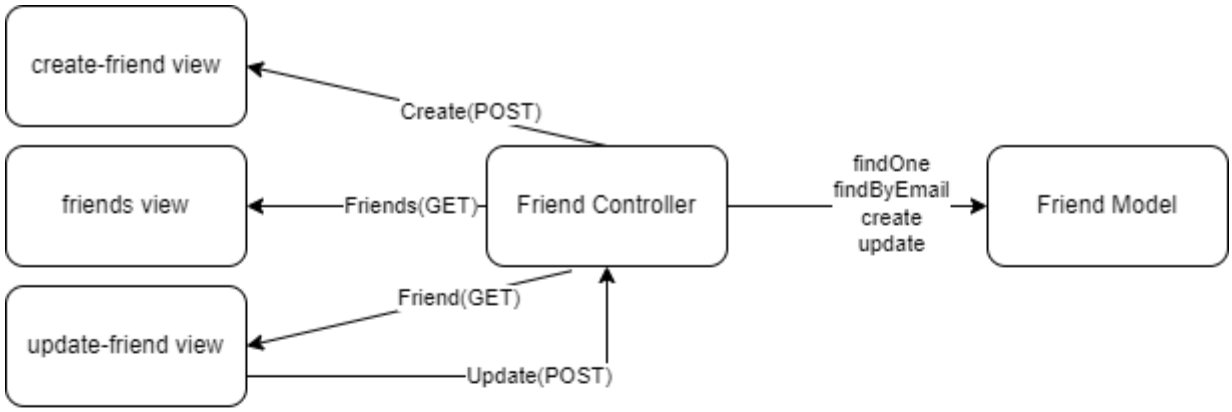
For a user to perform certain tasks such as viewing saved workouts or creating a new workout they must login. This is only possible if the user has previously registered otherwise they will encounter an error as they won't be found in the database. I have inserted a picture to show what happens when a user who isn't registered tries to log in.

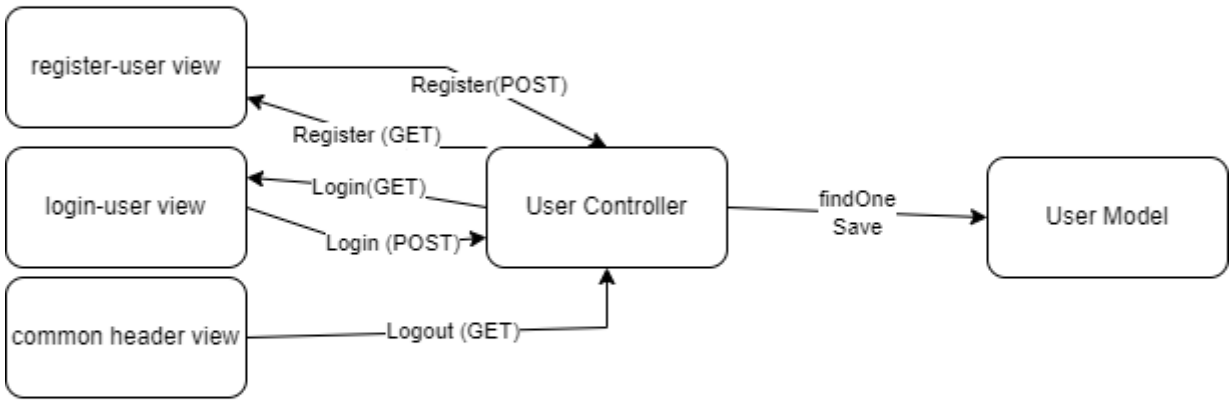


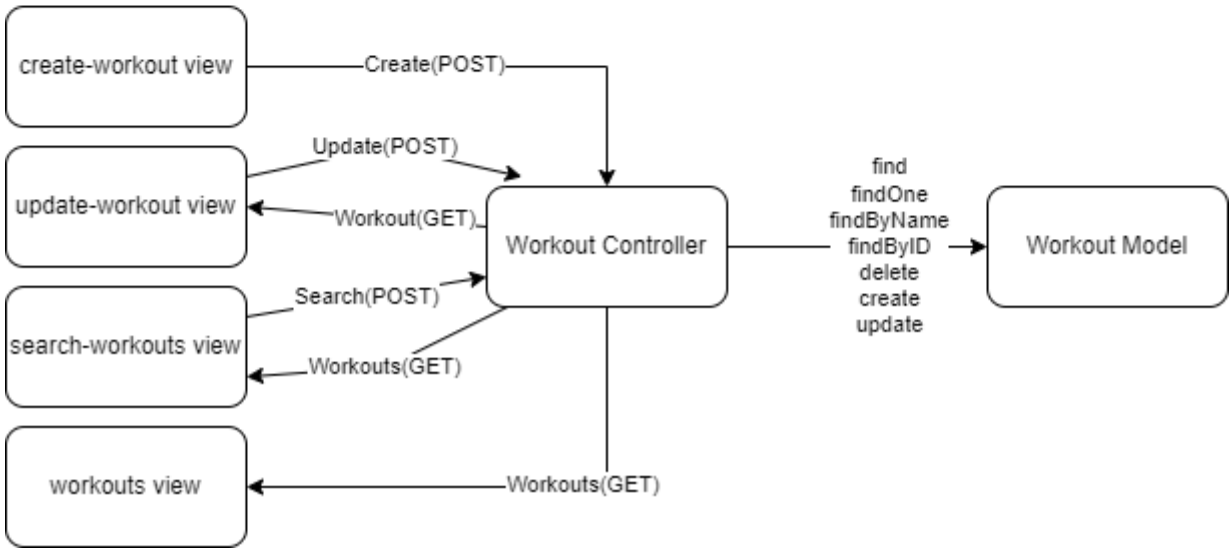
If a new user wants to register they must enter their email and a password into a form which then sends this information to the database. Once they have done this they will be able to login and then perform all tasks as the header displaying the navigation will change to display all tasks and also display the users email as well as a logout button. For added security, the program doesn't save the users password directly but it uses BCrypt in order to hash the password. The user is then signed in and this allows the application to identify the current user from within every controller as the user seesion is set.



Below is a system diagram to see how each layer of the MVC app works together:







Key Design Decisions

Database Design

This application consists of a production and development MongoDB database located in MongoDB atlas. Each of the databases have the following 3 main collections:

- Users
- Workouts
- Friends

And 3 secondary collections:

- Categories
- Reps
- Times

Users

The users collection stores the login information of each user, allowing workouts to be associated with a user. An example of a user from the collection can be seen below:


```
{
  "_id": {
    "$oid": "63c02d4848b15b2134dcc39e"
  },
  "saved_tastings": [],
  "email": "ahub.sajid1@gmail.com",
  "password": "$2b$10$AbSsvIwB0TmaWi9hdamFJ.Q.jMfo0DMKuJoZnWbOihPt/7KVugLYi",
  "createdAt": {
    "$date": "2023-01-12T15:54:48.859Z"
  },
  "updatedAt": {
    "$date": "2023-01-12T15:54:48.859Z"
  },
  "__v": 0
}
```


As you can see the password us hashed as opposed to being plaintext for added security.

Workouts

The workouts collection stores the workout information for each workout created. This include all information such as title, reps, category etc. An example of this can be seen below.

```
{
  "_id": {
    "$oid": "63c02d08f5c482251cb45823"
  },
  "points": "87",
  "title": "5k Record Run",
  "description": "Smashed a 5k run in record time",
  "friend_name": "Ahub Sajid",
  "category": "cardio",
  "reps": "5km",
  "time": "21 mins",
  "friend_id": {
    "$oid": "63c02d08f5c482251cb4582e"
  },
  "categories": [
    null
  ],
  "calories": null
}
```


Friends

The friends collection stores the information for each friend which includes data such as the friend's name, ID, how many workouts they've completed and when the friend was last updated. An example of this can be seen below:

Categories, Reps and Times

There are three secondary collections which are categories, reps and times which are mainly used to represent workouts or friends. The categories collection can be useful as if the program were to be developed further the user can be given the option to view workouts by category where this collection would then be useful. For example if the user wanted to perform cardio workouts only they can get a list of workouts that are from the cardio category to choose from. This also applies to reps and times.

Security and Scalability

Security

Security is a very integral part of any sort of website as companies must make sure the users data is completely safe in case of any sort of security breach or hackers. The only real sensitive information that is included in this system is the users password and in order to improve security the database only stores the password as a hashed value. The hashing is done with BCrypt which means the salting of the password is included as well. Salting protects the password against Rainbow Table attacks and makes sure even identical passwords give a unique output. As well as this BCrypt is a very slow algorithm which increases security as the time taken to guess the password using brute force also increases.

One possible way that security could be improved in the system is by using third-party security systems such as Google using OAuth. This could be used in the login / register sections and the benfit of this is the system will no longer need to store the users password.

Deployment & Scalability

The way that my applictaion deploys means that it is automatically build and released whenever it is merged with the master branch. This means that the code can be tested in quick iterations.

Scaling in MongoDB can be done in many different ways compared to normal databases that are normally limited due to the cost of implementation. In the application I have used horizontal scaling which brings in additional nodes to share the load of the data. This is difficult to do with relational databases as it is hard to split related data across multiple nodes however, as I am using a non-relational database this is simpler as the collections are self-contained and not coupled. This means that the nodes don't have to be 'joined' together. [1]

One way that I have increased scalability is by using a serverless design by having the application hosted by Cyclic.

Conclusion

Looking back, the prototype includes most of the features that I set out to make however I feel like it is a long way from being a finished and polished product. It fulfills the original goals that I set out to make which were the ability to add and view workouts to track my progress in the gym.

One way I could improve the applictaion in future development is thinking about making the application mobile friendly as well. This will be easy enough as the backend code is the same however there are little adjustments that need to be made to the views. The changes would be made to improve design consistency across all devices.

References

1. ◦ MongoDB. "How to Scale MongoDB." MongoDB, https://www.mongodb.com/basics/scaling
2. ◦ Cyclic. "What's different about cyclic?" Cyclic, https://docs.cyclic.sh/