Fithub

DM2518 Final Report

Larry Poon(960108-T137)

1. Project Description, background

Fithub is a full-stack web application that allows users to create, edit, copy their own work out schedule or from other users. The idea is based on an existing code-sharing platform, Github, which has been widely recognized by the develop community. The goal of the application is to bring exercise-lovers closer, by providing them space for sharing their work-out schedules, either from copying or editing, or create entirely new schedule. The target audience of this application ranges from normal exerciser to professional body builder. The application will come in handy when you have no idea how to systemically keep your body in a good shape, you could simply seek advice from other users, getting idea from their schedules. Your schedule can be varied day-by-day, week-by-week, changes can be made just under a button. Another highlight of this application is that you could find you best-fit schedule based on the body parts that you would like to work on. For example, Peter wants to grow some abs muscles, he could simply just put “abs” in the search bar, then all the work-out schedule that has relations to “abs” will be shown to his filtered list. Fithub is not just an application, it’s a community for exercise-lovers.

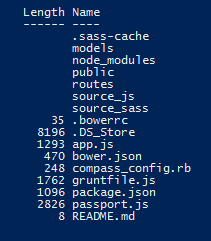
1. Method(s) used

This web application was built in fuill-stack. We have utilized the framework, MEAN.io, which contains MongoDB, Express JS, Augular JS, and Node JS. Here are some short description of those technologies above: MongoDB stores data in flexible, JSON-like document, meaning fields can vary from document to document and data structure can be changed over time, which is suitable to be our database engine of the application because we need to have various type of document for the objects we need to store, such as user profiles, work-out schedule, exercises information, and etc.(See [A] for details). Express JS is a minimal and flexible Node JS web application framework that provides a robust set of features for web applications. Some features that we have used in our application are: Passport Strategy, HTTP server, and JSON parser, these tools have done most of the heavy-lifting for us. We just simply need to call the API functions of those features and understand the use of those tools at a high level. Angular JS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML’s syntax to express the application’s components clearly and succinctly. For this application, we have made use of data binding and dependency injections. We also used Angular JS to wire to application API function to the frontend, so that user will have access to the application data, such as other people’s work out schedules. Angular JS is a powerful tool as it handles to main logic of the application in a systematic way with the feature of routing and module injection. We have also used Postman to test our backend APIs of the application. For the CSS framework, we have chosen Semantic UI. We were aiming to use a new UI framework that we had never used before because we wanted to find out which framework is the best and to expand of knowledge about the feature of each framework so that we would know, in the future, which one would be the best fit to serve our purposes. It turned out Semantic UI had given us all we needed. The framework is convenient, user-friendly, and, most importantly, it supports responsive layouts so that our design won’t crash with devices of different screen sizes.

1. Results, Screen Shots, and Technology overview

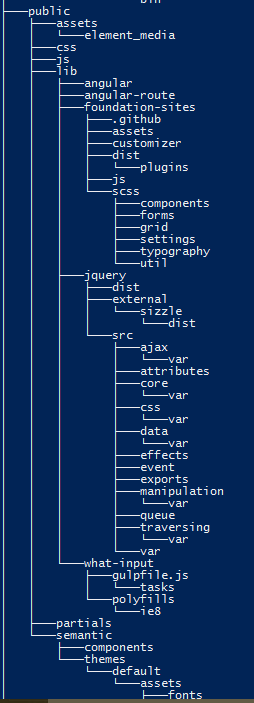
The end-product of the project is a fully functional web application: it has login authentication system, real data stored in database, user functionalities as intended. Within the given time frame, we managed to create 6 pages of HTML. They are the login page, home page, work-out schedule page, create-schedule page, view-schedule page, and profile page. The main functionalities are searching relevant exercise schedules within the list, constructing new work out schedule, and cloning existing work out schedules. They are implemented as intended. The source code of the application can be accessed via this github repository (<https://github.com/lpoon2/KTH-mobile-technologies.git>). To run the application, first, you would need to start the MongoDB engine. After that, you could run $ nodemon app.js to deploy the client to your local network. Finally, open your browser and open the localhost address with port 3000, then you should see the beautiful landing page of our application. The structure of the project can be visualized as follow:

**Main directory:**



*This is the main directory of the application. We define all the document objects, those definitions can be found under /models. We used Bower, Grunt, and Express JS to keep records of the dependencies we needed for the application.*

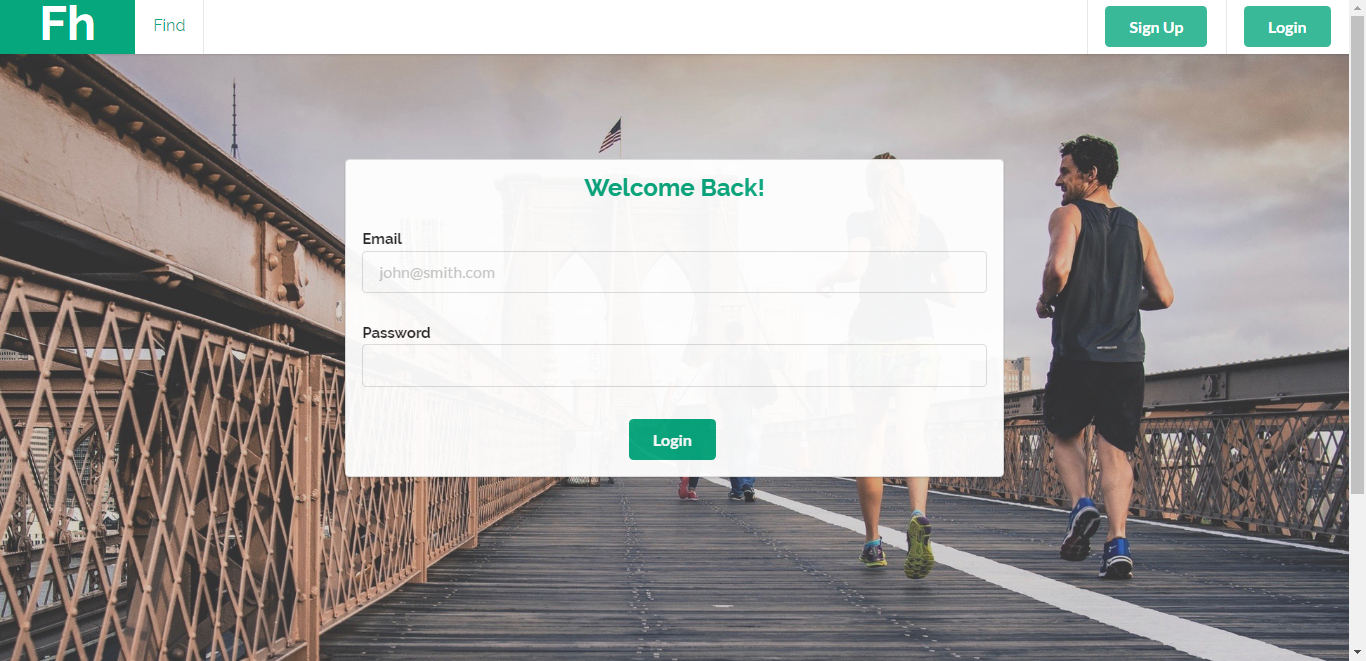
**/public directory:**



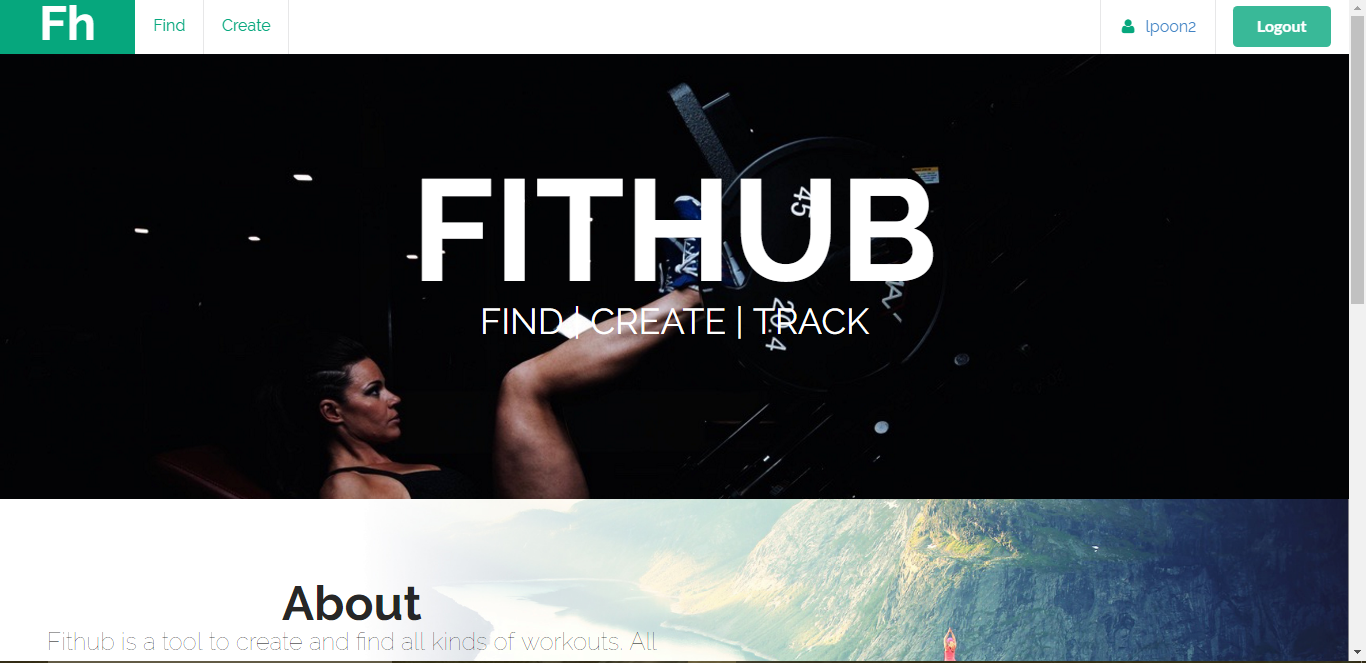
Every user-interface component can be found under the /public directory. The “Asset” folder stores all the physical assets, such as image files. Under “partials”, there are HTML templates. Our web pages are templatized, which means using one HTML as the main frame that contains all the repeated elements appear in every page of the web, like the nav-bars and footer.

The followings are some screenshot of the application:

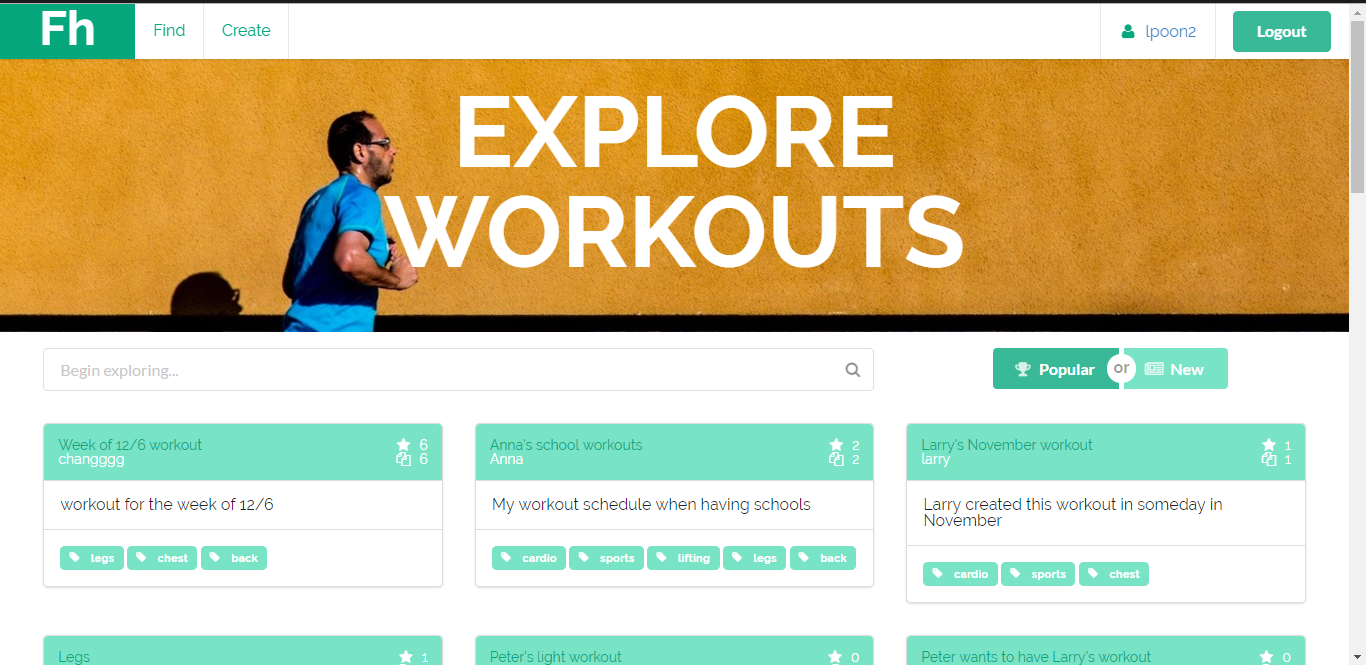
**Login page:**



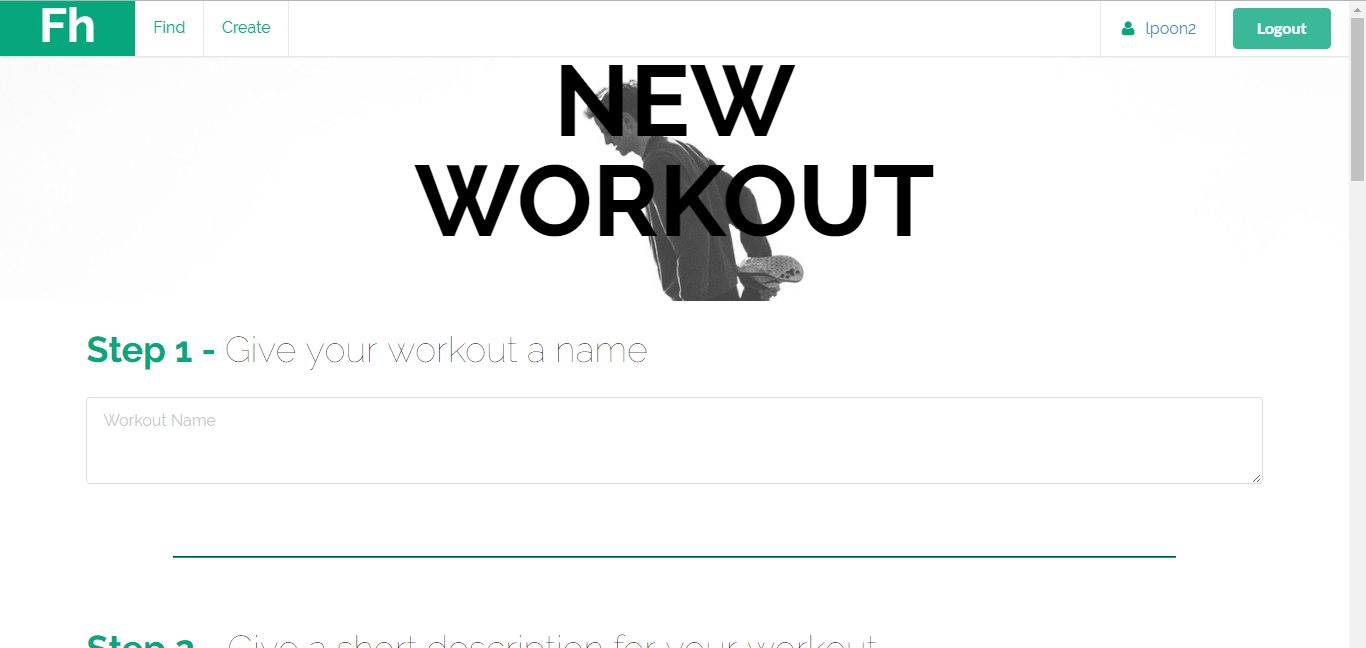
**Home page:**



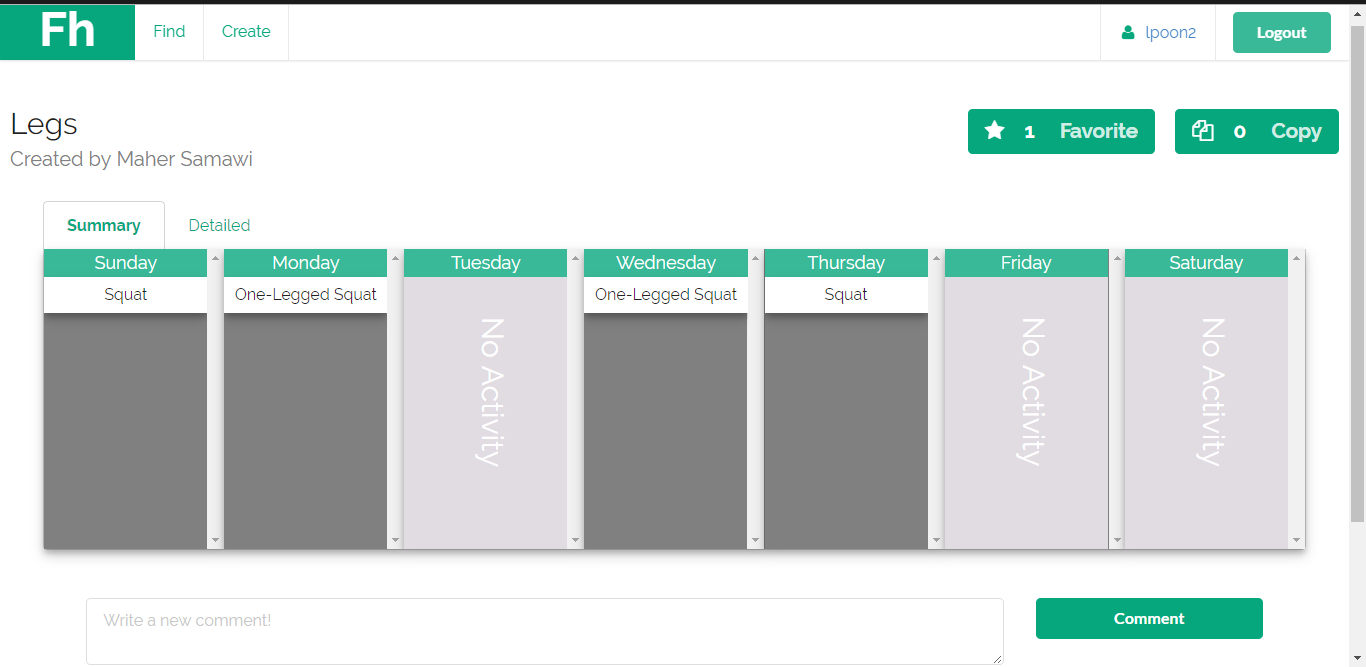
**Work-out list page:**



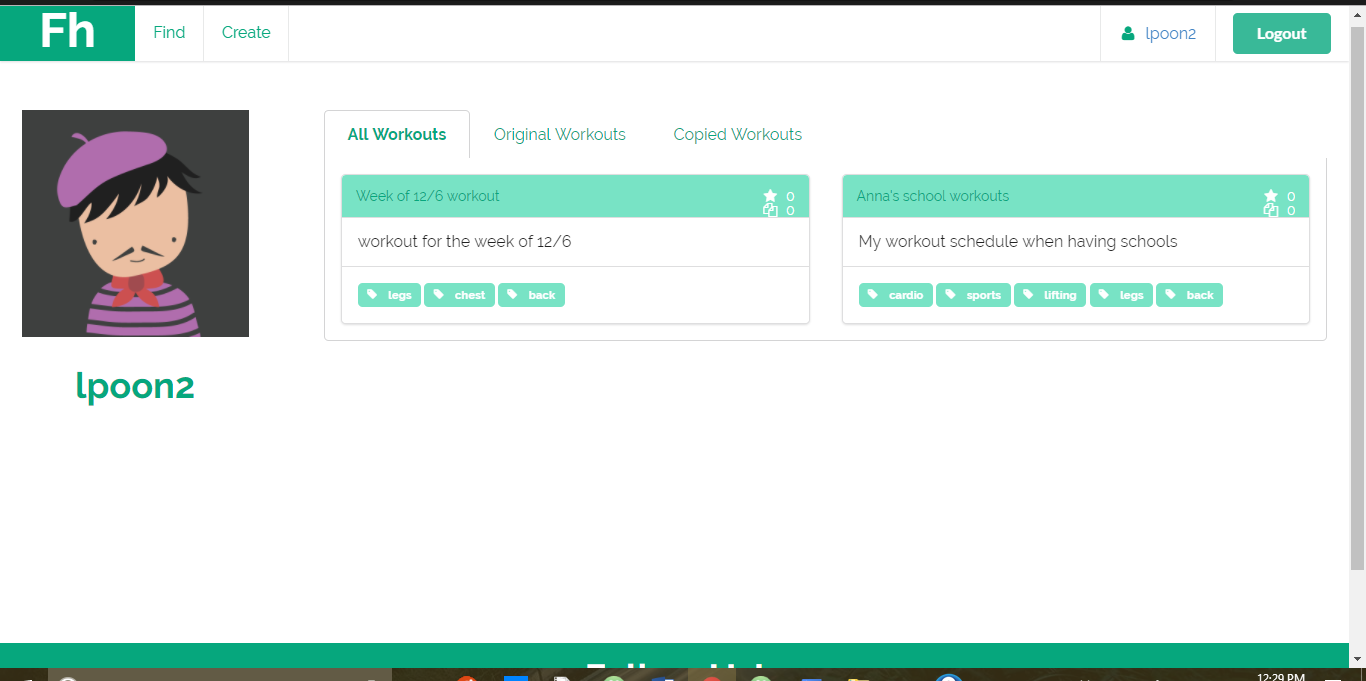
**New schedule page:**



**View-schedule page:**



**Profile page:**



1. Conclusions

With the given time frame, we have done what we have planned, a fully functional web application with some technologies we have learned over this semester. It’s been a great experience understand more different choices of technologies and how they can fit in to our implementation of the application. Those technologies, MEAN.io, are really cool in a way that they can simplify the code greatly, developers only have to know the high-level logic of how to put those components together. I would like to thank Prof. Jonsson and his team for the hard work designing the labs and projects. It’s a fun course, I would definitely recommend this course to my friends in the future.

1. Short segment on future work

There are quite a lot of possible extensions can be added to the application, the application is just a week-long assignment, we couldn’t implement everything with short amount of time. Yet, here are the three main features we have in mind that could be our possible new features.

1. Video-sharing: current users are allowed to add exercise elements to their schedules, however, that seems really plain. For each work-out elements, we could attach short video clips to show how to properly execute each exercise so that users can have a better idea of how’s that exercise look like and which body parts it will involve. Also, during the work-out period, users can live stream their work-out, just to show off their skills or whatever they intend to do.
2. Chat room: while users can share their work out schedule, they should be able to express their opinions. There are some many things users can chat about, such as how to correctly execute the exercises, diet suggestion, how to keep a good body shape, and etc.. The chat can happen under the schedule (the actual calendar), in a user’s profile page, or even the main page of the application.
3. Exercise suggestion: since machine learning is a big thing nowadays, we can add some elements of it in our application as well. Users can enter their body information when they register their account. Then, based on the schedules they have created or observed from the other ones, we can give some suggestions of how and what body parts should the user work on.

Appendix A

The following are the documents for our objects in the application:

**User Profiles:**

var userSchema = mongoose.Schema({

name: String,

email: {type : String, unique : true},

workouts : [String],

Hash\_password : String,

liked\_workouts :[String],

});

**Exercise Details:**

var elemDetail = new mongoose.Schema({

name: String,

index: Number,

elementid: String,

reps: String,

sets: String,

time: String,

distance: String,

});

**Work-out Schedule:**

var workoutSchema = new mongoose.Schema({

name: String,

description : String,

num\_favorite : Number,

num\_copy : Number,

original\_user : String,

original\_workout\_id : String,

current\_user : String,

current\_user\_id :String,

rating : Number,

public: Boolean,

dateCreated: {

type: Date,

default: Date.now

},

comments : [

{

user: String,

content: String,

}

],

tags: [String], // array of tags that pertain to the workout

//elements : [String], //array of element\_ids and fields(ie weight, reps, sets, etc

days : [

{

day:String,

currIndex: Number,

elements: [elemDetail], }]});