Practical 3
Runnify

University of St. Andrews

School of Computer Science

Student IDs:
230001378
230029279
230000868
230017667
220034529

Module:	CS5003
Date of Sub:	05.04.2024
Word Count:	1810 words

1 Overview

We believe that our app 'Runnify' meets all of the requirements in the specification. By kicking off the project with low-fidelity wireframes, and carefully designing the user experience, we have been able to create a fun and interactive experience for the user. Most of the styling of our user interface relied heavily on CSS as well as some external libraries which will be discussed in a later section of the report. Our application also relies on a server built on node.js and Express for exchanging data between the client and the server. We have also provided a stand-alone script to initialise the database and populate it with sufficient starter data, so that the website can be used directly. The credentials for the users and how to start the server can be found in the ReadME file.

2 Teamwork

Disclaimer: We were a group of five people, which led to an increased workload for the individual team members.

To begin, we had an initial meeting to talk about expectations and work patterns going forward. After this, a low-fidelity wireframe of the application was created to ensure we shared a mutual vision of the project goal. We discussed the API routes necessary to achieve this goal, as well as the distribution of work between members. Frequent communication continued through a WhatsApp group, while progress was tracked through Trello. As tasks were created people were assigned tickets to complete those. This system worked well as members could see who was working on what task with a clear overview of the remaining ones and the current stage of the project. Although teammates had some tasks assigned exclusively to them, collaboration was a huge part of the success of this project. Pair programming was used on an ad-hoc basis and helped members work through problems more efficiently. This approach led to an increased learning curve.

Task	Emma	Harry	Phil	Reanne	Teja
Running Feed (huge part)		x			
Testing Server Endpoints	x				
External weather API			x	x	
Graphs Statistics					x
Calendar for Runs	x				
Map API			x		
Algorithm			x		
Database Setup	x		x	x	
Register Page			x	x	
Login Page			x	x	
Styling of Pages	x	x		x	x
Create endpoints on server	x	x	x	x	
Sessions				x	
Helper Functions			x		

Table 1: Task Assignment

3 Technologies & Resources

The resources that were used in creating the project were accessed to make our project as dynamic, fun, and interactive as possible. As we had accessed external libraries and sources in the hopes of achieving a successful project, the majority of our sources are a result of external library documentation and inspiration for making the page interactive. Resources, as well as what they were specifically used for, are provided in the table below:

Resource	**Description**
openWeather API	Predict the weather for a specific day
GeoCoding API	Retrieve longitude and latitude for the starting point
MapBox	Plotting the route on the map
D3	Creating the graphs in the statistic page
SweetAlert2	Creating the pop-overs
MongoDB	Used for storing the data
Sessions	Track the user over the pages
Tutorial & Lecture Slides	Set up the database
Mocha & Chai	Testing the Code

Table 2: Resources used for the Project

4 Coding Design

Modularisation was used for HTML. The helperFunctions.js file exports functions to help build HTML elements using JavaScript more efficient. Modularisation of code was also used extensively throughout the project to maintain a good structure and increase maintainability. Furthermore, the project was split into the into various files on the client side. These are listed below:

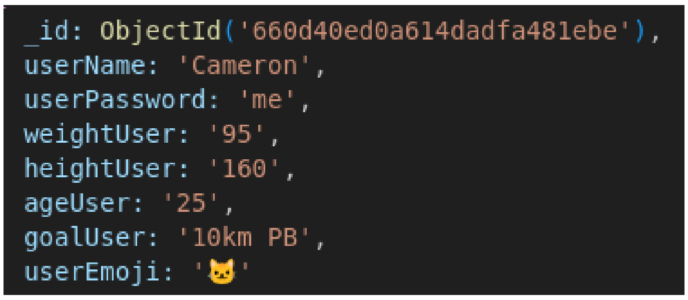
- Log-In
- Registration
- Homepage/Feed
- Create run
- Profile/Statistics

Each file contains its own Javascript & CSS file. The split was based on functionality as it made working more efficient and also merge conflicts were decreased. In addition, the server (main.js) connects to MongoDB to handle the session of the users and is responsible for the logic, as well as the retrieval and insertion of data while acting as an intersection between the client and the database. It contains the following endpoints:

API Route	**Description**
app.post("/checkUserCredentials")	Authenticates users at log in. Creates a new session for users at log in. Compares user-inputted credentials with those in the database and sends back status codes to the client (e.g. 401 Username and Password do not match).
app.post("/registerUser")	Checks that username is not taken and sends back status code. If successful inputs user data into the database.
app.get("/fetchUserInfo/:userName")	Returns data from 'users' collection based on provided username.
app.post("/createruns/:CITY")	Inserts user created runs into the createRuns collection of the database. Sends relevant status codes back to the client.
app.get("/api/fetchRuns")	Fetches entries from the createRuns collection and sends data back to the client.
app.get("/updateToPastRun")	Sorts createRuns entries by date. Marks runs whose date are in the past to 'isFinished'.
app.post("/api/updateRun")	Updates objects in createRuns collection when user likes, joins or comments on posts in the main feed.
app.post("/updateUserProfile")	Updates entries in the user collection when a user makes changes on their profile page.
app.get("/logout")	Destroys user session when clicking logout button and sends back status code to client.

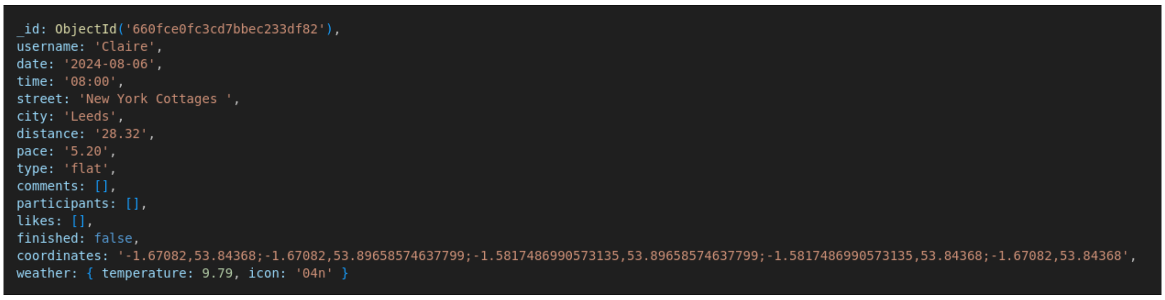
Table 3: API Endpoints

These endpoints are being used to communicate directly with the database, which contains two collections inside with the following structure:



```
_id: ObjectId('660d40ed0a614dadfa481ebe'),
userName: 'Cameron',
userPassword: 'me',
weightUser: '95',
heightUser: '160',
ageUser: '25',
goalUser: '10km PB',
userEmoji: '🐱'
```

Figure 1: Structure collection users



```
_id: ObjectId('660fce0fc3cd7bbec233df82'),
username: 'Claire',
date: '2024-08-06',
time: '08:00',
street: 'New York Cottages ',
city: 'Leeds',
distance: '28.32',
pace: '5.20',
type: 'flat',
comments: [],
participants: [],
likes: [],
finished: false,
coordinates: '-1.67082,53.84368;-1.67082,53.89658574637799;-1.5817486990573135,53.89658574637799;-1.5817486990573135,53.84368;-1.67082,53.84368',
weather: { temperature: 9.79, icon: '04n' }
```

Figure 2: Structure collection createRuns

4.1 Signing up/Logging in

The landing page for the user is the log-in page. Entering incorrect or unrecognised credentials will trigger a pop-up to the user prompting them to try again or register as a new user. If the user successfully logs in a new session is created for them for the server to track their journey through the app. The user's username is added to the session to track which data they add. Clicking the logout button destroys the user's session and redirects them to the login page. Users without a session are unable to navigate to the main app. Directly entering the file path redirects unverified users back to the login page, to exclude users without a session as this would break the logic of our code. The registration page requires user inputs which are then stored in the database and reused throughout the application. Error alerts are triggered for invalid inputs e.g. mismatching passwords or having a username already in use.

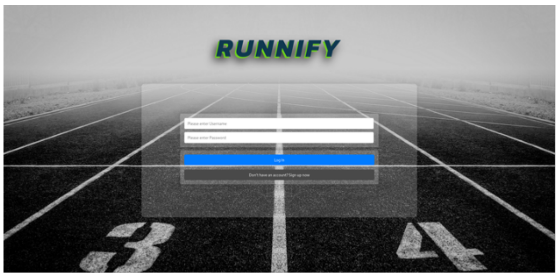


Figure 3: Login Page

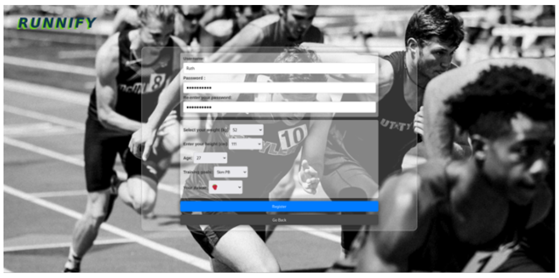


Figure 4: Register Page

4.2 Running Feed

The running feed shows important information about all runs, like date, meeting point, pace, and the route on a map with all the 'rest' points along the run. As all routes are circular, the first point is coloured red to indicate the start and end points of the run. The shape was chosen because mostly a run starts and ends on the same location. More time given we could have implemented more options for the user to choose from. Each card has four pieces of functionality: joining a run which adds it to a user's upcoming runs in their profile, liking a run which makes the like (heart) counter increase, commenting, and viewing other participants attending the run. Additionally, we have implemented a filter function to sort the list by runs that occur the soonest and runs that have been recently posted. Additionally, you can filter by pace or distance. Finally, we have implemented a refresh button for a user to see new posted runs or to check updates when other users perform new interaction actions.

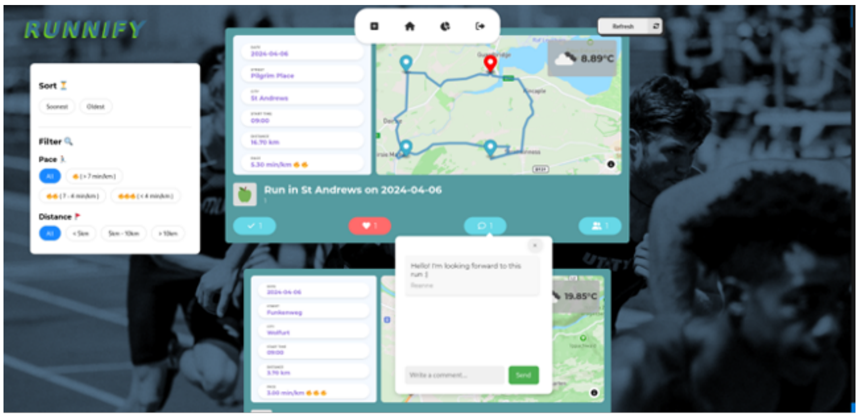


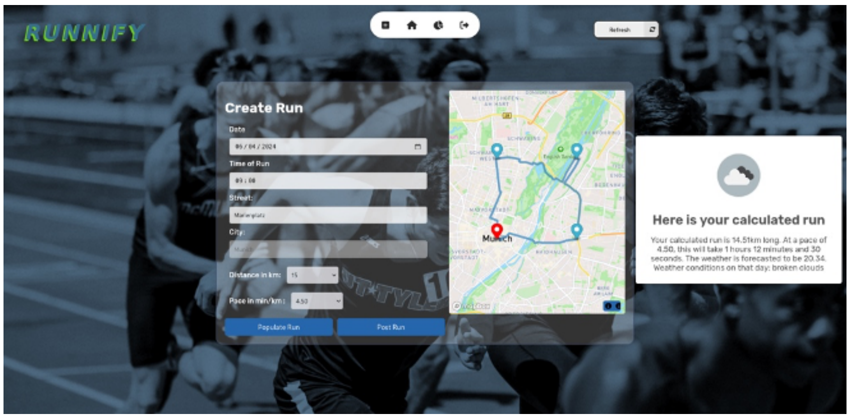
Figure 5: Running Feed Page

4.3 Create Run

The 'Create Runs' page allows the user to create and post a new run. The user inputs the time and date, which are restricted to future dates and are used to calculate if weather forecasts are possible. The inputs related to the location are connected to mapBox to retrieve actual street names and get valid inputs from the user. Furthermore, the city is disabled to reduce the risk of invalid street names. This is important because those are the inputs for the geolocation API of Openweather, which returns the coordinates of the given address. This was essential as the weather forecast uses longitude and longitude as parameters. Furthermore, these are the inputs to the algorithm, which calculates waypoints for a circular route, also based on the distance chosen by the user. The mapBox API can calculate a route between these way points using these coordinates. Furthermore, these are also used as way points afterwards. If, due to limitations of the algorithm, no route can be calculated that matches the user's desire a popup will inform them to handle errors properly. This means that either the starting point or distance has to be adjusted, which would lead to a different result

4.4 Profile and Statistic Page

The statistics page allows the user to update their fitness goals and keep track of their activity trends. Depending on the fitness goal the user selects they will be provided with an informative video on how to improve their running and achieve that goal. Additionally, there are two charts created using D3. These include a bar chart displaying the total distance ran in each month and a scatter plot that plots each run's pace and distance. The graphs are dynamic with each row in the Past Runs table and will change the visualisation on both axes to modify the range when necessary. A small table is also provided to make a user's longest run, fastest pace, and average distance visible. These



Date: 05 / 04 / 2024
Time of Run: 09 : 00
Street: Marienplatz
City: Munich
Distance in km: 15
Pace in min/km: 4.50

Here is your calculated run

Your calculated run is 14.51km long. At a pace of 4.50, this will take 1 hours 12 minutes and 30 seconds. The weather is forecasted to be 20.34. Weather conditions on that day: broken clouds

Figure 6: Create Run Page

functions calculate the longest run and fastest pace using a for-loop of their respective object in the database, while the average distance calls to an external function as it requires a calculation that totals and divides the distance by the number of runs. Finally, two more tables show the user their past and future runs. The past runs are filterable by a dropdown selection of location and month attributes once the user submits their selection.

5 Testing

Our testing strategy was designed to ensure the robustness and reliability of both client and server components. We used a blend of automated and manual testing techniques, focusing on comprehensive coverage of functionalities and user interactions. Our objective was to identify and rectify potential issues early in the development process, thereby enhancing the overall quality of our web application.

Automated server-side testing with mocha and chai

We opted for mocha as our JavaScript framework for writing unit tests and chai as our assertion library. They were chosen due to their ease of use and support with asynchronous operations. Although these frameworks were compatible with both the frontend and server, they were only used to test the server. This was due to the application's inconsistency in importing packages in the frontend. For example, map features using mapbox-gl had been imported into the html page so they could not be accessed by mocha and chai in the terminal. Due to the widespread use of maps across this application and the integration of numerous other libraries in this fashion, it was not possible to create automated tests for many features on the frontend.

Manual browser testing

Complementing our automated tests, manual testing through the browser was integral for evaluating the client-side experience. Firefox was the primary browser used for this due to its comprehensive developer tools and widespread use. This phase involved simulating user interactions within the application, such as joining/ posting runs, interacting / filtering posts on the feed page, and viewing user-specific information on the user page.

Test scenario	Expected output	Actual output as expected
Three users join the same run	Three users should be added as participants of the run.	Y
Five users attempt to join the same run	Five users should be added as participants of the run. Run will appear on each of their profiles.	Y
User attempts to like post on run feed twice	User should be restricted from liking the post more than once.	Y
User attempts to unlike a post	Like count on post should decrease by one when they click the like button again.	Y
One user cancels run with four other participants	User should be removed from the participants list. Three users should remain. Run will be removed from the users profile that left.	Y
User attempts to update profile, changes goal on drop down menu but presses back instead of submitting.	User profile should and recommended video should not change.	Y
User schedules run for location that does not exist.	User should be restricted from posting run.	Y
User initially schedules run with a correct location, however, they remove the street input and replace it with a street name that does not exist.	User should be restricted from posting run	Y
User attempts to schedule run at time 25:00	User should be restricted from typing 25:00 into the time field.	Y
User attempts to schedule a run before today's date.	User should be restricted from inserting past dates into the date field.	Y
User attempts to schedule run without including date	User should be restricted from posting the run.	Y
User attempts to schedule run without including location	User should be restricted from posting the run.	Y
User attempts to schedule run without including time	User should be restricted from posting the run.	Y
User attempts to schedule run without including distance	User should be restricted from posting the run.	Y
User attempts to schedule run without including pace	User should be restricted from posting the run.	Y

User attempts to schedule a run for the morning when it is currently the afternoon.	User should be restricted from posting the run.	Y
User filters feed page for runs less then 5km in distance but has not scheduled/completed any runs for this distance	No runs should appear on the feed page.	Y
User attempts to access create runs page from URL without logging in	User should be restricted from accessing the page. They are directed to the login page.	Y
User attempts to access profile page from URL without logging in	User should be restricted from accessing the page. They are directed to the login page.	Y
User attempts to access feed page from URL without logging in	User should be restricted from accessing the page. They are directed to the login page.	Y
User participated in a run that was scheduled yesterday. Time has passed and it is now the next day.	Run should appear in past runs table and be included in the user statistics.	Y
User participated in a run that was scheduled for the morning. Time has passed and it is now the afternoon.	Run should appear in past runs table and be included in the user statistics.	Y
On the feed page, the user filters the pace for faster than 4 min/km and distance longer than 10km. They sort runs to see the newest first.	Only runs faster than4 min/km in pace and longer than 10km in pace should appear on the page. Runs are sorted newest to latest.	Y
On the feed page, the user filters the pace faster than 4 min/km and distance for greater then 10km. They sort runs to see the soonest first.	Only runs faster than 4 min/km in pace and greater than 10km in pace should appear on the page. Runs are sorted from soonest to newest.	Y

Table 4: Test scenarios and outcomes

6 Evaluation

Without time constraints, we would have liked to personalise a user's experience more within the app. Some examples of this would have been to allow a user to set their training goal to 'marathon' for instance, then we could have implemented a feature where longer runs that they could join were pushed to the top. Additionally, we would have liked to include graphs that allow a user to compare their statistics and running trends with other runners training for marathons. This would then be an inspirational and goal-setting feature that would incentives users to run more and achieve higher results as a cause. Furthermore, one feature of our project that is slightly finicky is the algorithm that has to do with locations. If we had more time, we could create an algorithm that provides more precise locations when given different distances and running routes.
Overall, we are proud of the work we were able to achieve with five team members. As our group had started with a disadvantage, most of us were unsure about how having one less group member would affect us. However, we were able to persevere and work past this challenge despite our previous concerns. One of the things that helped us accomplish our goals was that we shared a common vision for the project. Even though we had different views of what the project might look like, our team was quite flexible and open to new ideas. In addition, our creation of a rough wireframe had set us up for success as we were able to all visualise the same finished project and allocate certain tasks more effectively. We also communicated well as a group which enabled us to move quickly and get help when necessary.

References

[1] Basic scatter plot in d3.js. https://d3-graph-gallery.com/graph/scatter_basic.html. Last accessed: 2024-04-02.

[2] Distance calculation. https://en.kompf.de/gps/distcalc.html. Last accessed: 2024-03-28.

[3] headshot-placeholder-400×267. https://uolcareers.co.uk/about/headshot-placeholder-400x267/. Last accessed: 2024-04-02.

[4] How to add a place autocomplete search widget to your website. https://www.youtube.com/watch?v=c3MjU9E9buQ. Last accessed: 2024-04-02.

[5] Most basic barplot in d3.js. https://d3-graph-gallery.com/graph/barplot_basic.html. Last accessed: 2024-04-02.

[6] Numbers on red running track. start and finish point of a race track in a stadium(black and white photo). https://www.shutterstock.com/image-photo/numbers-on-red-running-track-start-1759475468. Last accessed: 2024-04-01.

[7] Popular png categories. https://www.vecteezy.com/free-png. Last accessed: 2024-04-04.

[8] Sessions vs tokens: How to authenticate in node.js. https://rrawat.com/blog/sessions-vs-tokens-authentication. Last accessed: 2024-04-02.

[9] Trendy route map. https://www.vecteezy.com/vector-art/17309960-trendy-route-map. Last accessed: 2024-04-04.

[10] Weather api. https://openweathermap.org/api. Last accessed: 2024-03-29.

[11] Web services apis. https://docs.mapbox.com/api/overview/. Last accessed: 2024-03-29.