Readme

Team 2

I. General remarks

The application is deployed on http://farm02.ewi.utwente.nl:7004 so it can be easily accessed with variety of devices.

Front-end: styling and layout

One of the requirements for the project is the usage of some CSS framework.

However, our team decided not to use any framework exactly because we know how to use two of the most popular frameworks - Bootstrap and Materialize. More specifically, our decision is based on the following reasoning:

- Frameworks usually provide very default-looking styles meant for fast development without any means for easy customization or overwriting. This usually makes app's own styling sheets more cluttered and gives much less freedom in styling.

- As with the styling, layout possibilities of frameworks are sometimes too limited to satisfy all the needs. Instead, out-of-box CSS3 features flexbox and grid - two layout tools that allow for extremely fine-grained level of layout customization. We used both of these tools to make our application's UI efficient and responsive.

- Frameworks use classes to denote layouts which, as a tradeoff, makes HTML documents too cluttered, especially when there are multiple own classes and ids already assigned to the document's elements.

- By using plain CSS for styles and layout, we wanted to gain more in-depth understanding of the language that is still a foundation for any CSS framework (same reasons as for not using JSP or ORM in the project).

All in all, we decided that we don't need a CSS framework and, as we think, this decision can be justified by the look of our application.

SQL transactions and stored procedures

Although we experimented with SQL transactions, our application in the end result does not use them and there are several reasons for that.

First of all, there are no groups of multiple related queries to be executed as one transaction. Almost all of our queries are self-standing transactions that use only one connection to the database.

Furthermore, as our application's aim is data visualization, the majority of our queries are simple read statements with no remove or write counterparts to cause problems with dirty reads, dirty writes or phantoms which would require aborting the transaction.

To allow faster access to information from the database, our application uses stored procedures (defined in *main.java.utwente.team2.databaseBackup.storedProcedure.sql*).

HTML Templates and multiple ajax calls

As we were told not to use JSP, some code duplication and usage of custom built templates was inevitable as a necessary tradeoff. In some places we solved it by using our custom templates to build HTML with inserted content. But for some files this approach was not suitable as they required multiple lines of inserted content (therefore, for instance, we have separate HTML pages for free and premium dashboards although their content mostly overlaps).

What is more, javascript files contain a lot of ajax requests to populate our HTML files after they are loaded in browser which would otherwise be populated on server with JSP templates.

With JSP all these unnecessary code repetition and workarounds would not be needed.

II. Features

Dashboard

Definitely, the dashboard is the core of our application and contains the features that we are most proud of.

The data that we worked with in our project is mainly consisted of countless rows of values for 20 different variables from on-body sensors for runners. As this is a lot of data to visualize, it may not be very efficient to put all the data together in one single dashboard and clutter it. Therefore, our idea to tackle this problem was to create a flexible and highly customizable dashboard which allows a user to select, arrange or remove visualizations according to user's preferences.

Our "drag-and-drop" dashboard consists of cards of various types. We used a javascript library called "*muuri*" to help us implement a top-left bin-packing algorithm to calculate the positions the cards attain when they are moved around the dashboard.

Our application automatically saves the layout of the dashboard to the database after it detects any change in it (addition, deletion, position change etc.). When the user loads the page again, the layout is restored from the database in the same way as it was left by the user.

Dashboard for each run is accessible form the profile page with buttons on the respective card of this run. To move a card, user just needs to hold the card and drag it to the desired position. Dragging is disabled on smaller screens as our test showed that it does not provide a convenient way to interact with the dashboard on these devices.

Below is a brief description of the functionality of our dashboard.

Adding cards

Dashboard's menu button is located in the bottom left corner of the screen. By clicking on this button the user can expand the group of control buttons.

Users can select cards and place them on the dashboard. To select a card, click on the control button with plus icon. In the popup dialog the user can choose from:

- Individual - these cards show average value for the chosen indicator name and side, as well as the minimal and maximal values.
- Graph - these cards show the graph for the chosen indicator name and side. There is also a blue baseline on each graph which represents an average value for this indicator calculated from all the runs in the database. By clicking on the labels above the graph, the user can disable or enable individual lines.
- Distribution - these cards show the distribution graph for the chosen indicator name and side. With these graphs the user is able to observe for how long the indicator remained in the shown ranges.
- Note - this card gives the user an opportunity to write down comments and remarks about the run. Changes made by the user are saved automatically (red light in the top right corner means that changes are pending, green - changes are saved). Only one note card can be present on the dashboard at the same time.

Removing cards

To remove a card from the dashboard, hover over the card, then button with "x" icon will appear in top right corner of a card. Click on it and the card will be removed.

To remove all cards with one command, press a control button with bin icon on it. It will show a popup dialog where you can confirm your action. All the cards will be permanently deleted from the dashboard afterwards.

Default layout

The user may choose to restore a default layout that was pre-created by us. It consists of 50 cards of various types.

Sorting, searching and filtering

Dashboard also has a top menu for searching, filtering and sorting the cards. To sort the cards by their titles or filter by the type, just select the according property in the menu. To search by the cards titles, enter the search query in the respective field.

Premium features

Premium users have three more features on the dashboard available to them.

Multiple dashboards for one run

If there is a lot of data to analyze and the user wants to arrange it in several convenient ways for further analysis, our application is offering a possibility to create multiple dashboards. Premium users can have up to 5 different dashboards for each run.

Favourite layouts

Premium users can save their favourite dashboard layouts and restore them in other dashboards. For example, if the user likes when there are graphs for push off and braking forces on top and the note with comments below, then the user can mark this layout as favourite and assign a name to it. In a new dashboard for another run the user will be able to select this saved layout and then this dashboard will be arranged according to it.

Export run

Premium users can export general information about the run in two formats. First one is HTML/CSS/SVG webpage 800 by 600px wide, the other one is as a png image.

To export a run, click on export control button and either type in an email to which the image is to be sent or open a web infographic in a new tab.

If a free user is logged in, then these three features’ control buttons are marked with stars. After the user clicks on one of these buttons, a popup dialog appears with a description of a feature and a button to upgrade to premium. A free user may also upgrade to premium by clicking on upgrade button in profile’s navigation bar.

All these premium features are also protected at the server. So even if the user bypasses our application's javascript, there is no way to access the features without a token for a premium user.

Compare runs

One more feature available only to premium users is a possibility to compare indicator graphs for multiple runs. The button to access a comparison page is located under the user's profile picture. The user may select multiple runs to put their indicator curves on a graph for comparison.

Responsive design

Our application is accessible for the users using a variety of different devices and screen sizes. It is properly displayed on laptops and desktops, as well as on smaller devices such as mobiles and tablets. We tried to think about good UI and UX as much as was possible for the project with such a short timeframe.

Profile image upload

The user that just registered in our application will see a default profile picture on the profile page. Any user may change the profile pictures on settings page (accessible from navigation bar) by clicking the button under the picture. Currently, the application does not check for the size or aspect ratio of a new picture, so if an image with different aspect ratio than the default one is loaded, the profile picture container may become less appealing (for example, it may become too small or too big).

Change names

On the settings page the user can also change the first and last names by simply typing in new values and clicking the save button.

Profile page

Profile page shows a general overview of the user's runs with total number of runs, total distance, duration etc. Every run occupies a separate card with general information about the run.

Registration

A new user can register by going to sign up form from the index or sign in page.

All input that is typed in by the user in the registration form is validated according to our requirements. Username and email availability are validated on-fly before the user can submit the form. If some value does not satisfy the requirements, the user does not need to fill the full form again but may just simply change the necessary fields. To prevent the malicious user from bypassing our front-end validation, the same check is performed on the server-side.

After the user registers, the verification email is send to the provided email address to confirm that the user is indeed the owner of this address. To activate the account, the user needs to click the button in the received email. The user will not be able to sign in to the application until the account is activated with the received token.

Password reset

In case the user forgets a password or wants to reset it to prevent the attacker from using this user's stolen credentials (login and password or a valid token), the user may initiate a password reset procedure by clicking on the respective link (currently located only in the bottom of login form). Then the user needs to type in either a username or an email to receive the email with a reset token. After clicking the button in the email, the user may enter a new password and confirm it in the opened window.

More information about the implementation of features related to security can be found in the security report.

III. Default users

Premium user with all features and 8 runs:

- Login: CvdB
- Password: Password7

Free user with limited features and 1 run:

- Login: JF
- Password: Password7

As both of these users have our team's private emails assigned to them, password reset functionality can be tested only by registering a new user with your own email address.

If you want to dump the database and reload it again from excel and csv spreadsheets, just run *main.java.utwente.team2.databaseBackup.DataExporter.* It will clean the data in the database and reload it anew.