CS5500 Project Group 9: Product Delivery

Pradeepti Reddy Tandra
Zachary Sylvane
Sam (Donghyeon) Kim

Overview

Trello Board Link

https://trello.com/invite/b/juyOYrwV/354cac4debdfb8cd945c0b31eaed8cfe/cs5500-project-group-9

Project GitHub Repository

https://github.com/CS5500-Project-Group-9/project

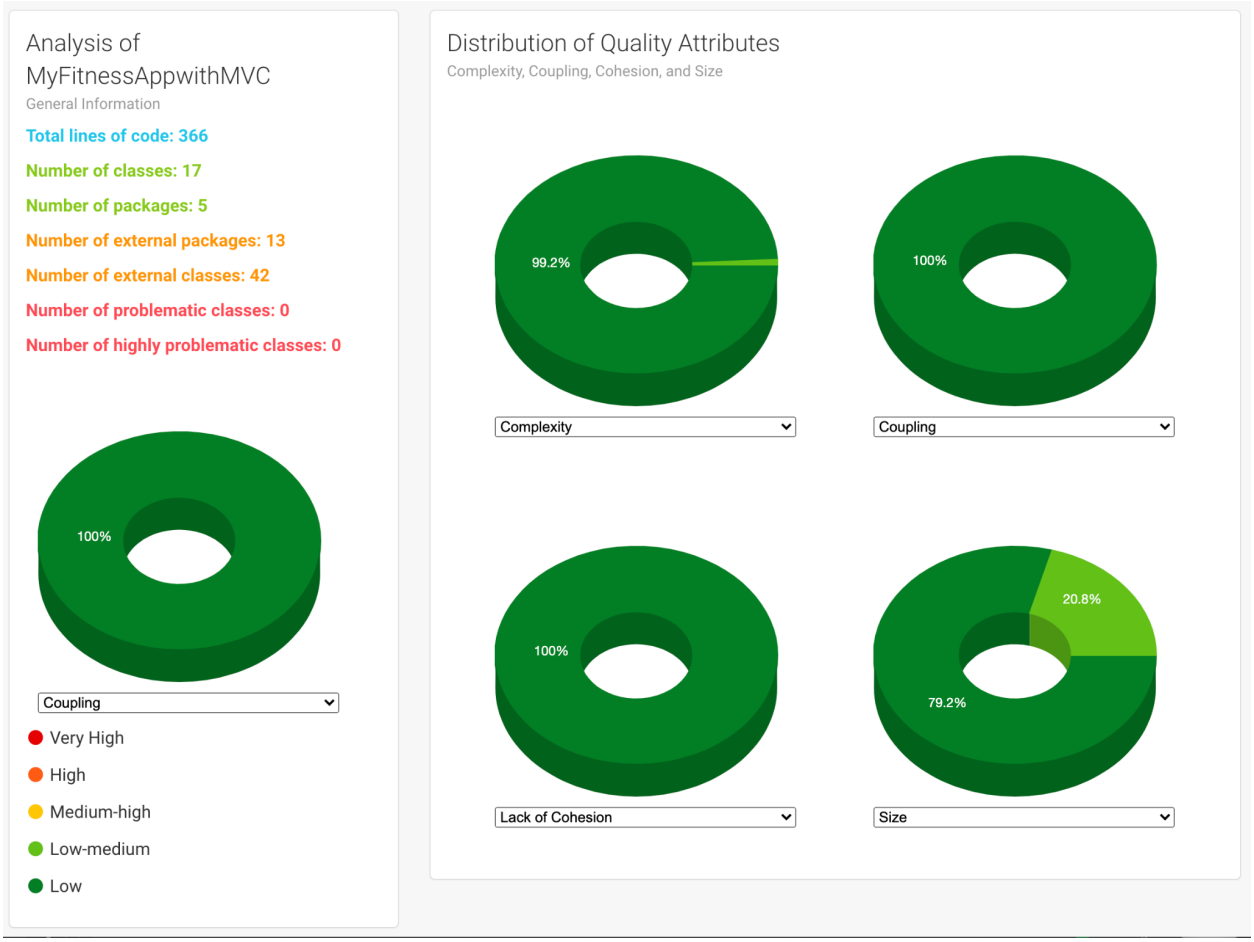
FINAL USER STORIES & FEATURES

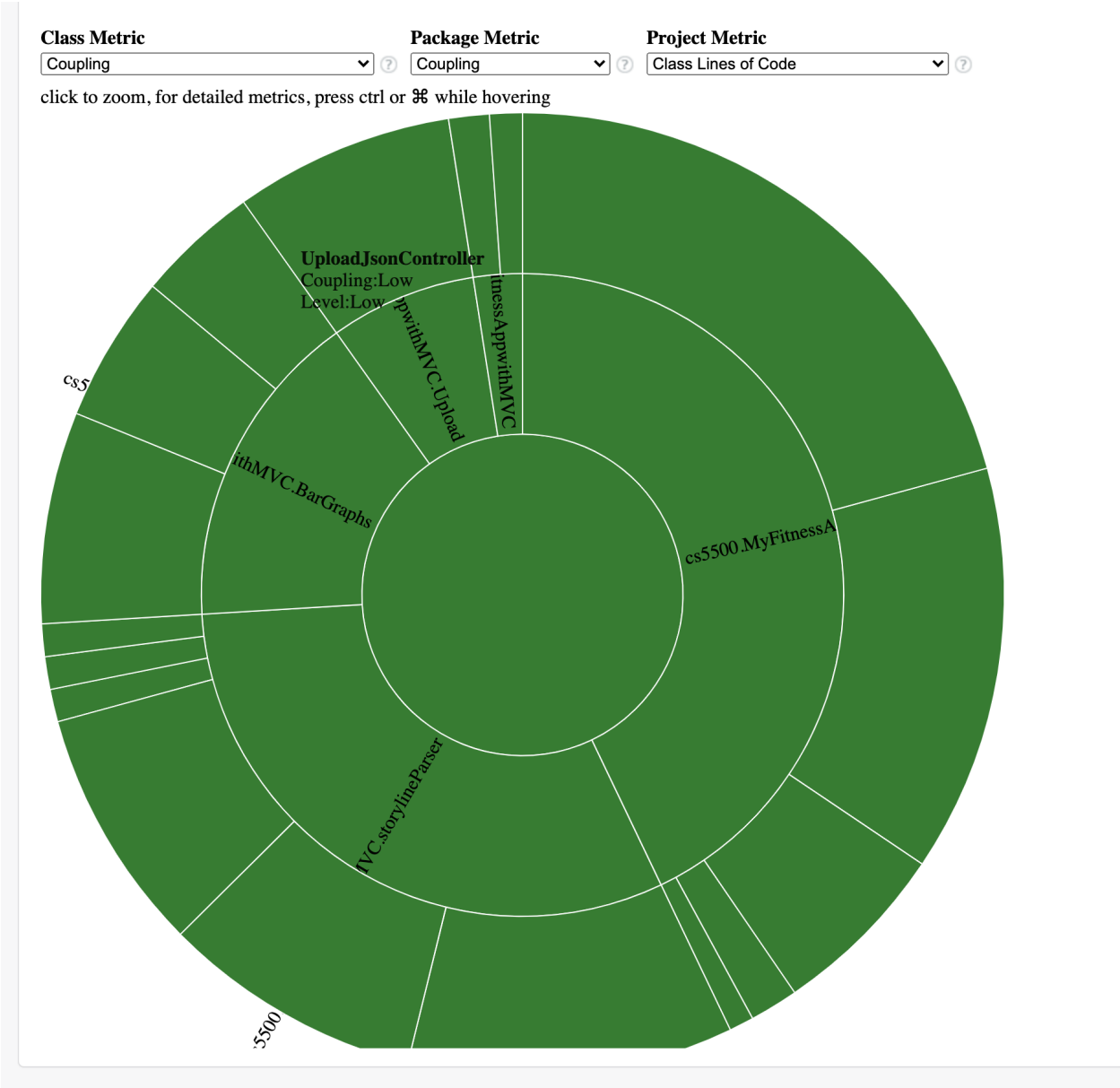
As someone that uses a phone app to track my active lifestyle, I want to be able to directly upload a json file and import my phone data directly on the site UI	*We implemented a basic UI, where a user can click a button and upload a JSON. The file must be named as per the specifications.*
As the CTO, I would like this project to have a REST API, so that it can be accessed and utilized by outside users. (adapt what we have completed to REST API protocols and naming/packaging conventions)	*We correct our endpoints to handle the standard REST API requests.*
As a web app visitor, I want to be able to sort my activities ascending/descending. (this feature can be added to page 1, clicking calorie column will sort by calories for instance.)	*We implemented a sorting query into our database and retrieve that data to the user represented in the UI.*
As a webapp user, I would like the ability to be able to add new Exercise information	*We implemented CRUD operations through a simple GUI.*
As the CTO, I want to see a high-level visualization of the program to see if we can integrate it into our other projects.	*We created a final UML class Diagram of our current working implementation.*

As someone who works in technical sales, I would like the program to be able to provide data visualizations that are appealing to customers.	*We implemented several different data visualizations using Thymeleaf and charting libraries such as Highcharts.*

Code and Test Metrics

Below you will see our screenshots for a static analysis of our code provided by CodeMR. As you can see we have solid results across the board in all categories. This lines up with the modular design of our code, which you can see in this [UML Class Diagram.](#) The utilization of the Model, View, Controller (MVC) pattern was particularly helpful for decoupling and adhering to the solid responsibility principle.





Testing

While we do have some unit tests for our classes, the most obvious and effective testing was done by visually inspecting our UI, to make sure our CRUD operations were functioning as expected. This approach follows the idea (detailed in the book *Software Engineering at Google*), which insisted that testing should focus on behaviors and not methods. We created some unit tests to ensure that our repository was working correctly, but focused on carefully inspecting the UI.

Final Product Documentation

A detailed set of instructions on how to set up and run our program is in our Readme on our Github page. There you will find our source code, how to get up and running and any known issues. I have copied it below for your convenience.

Copied from Github

SprintFit

This app lets you upload and add your fitness activities and view fitness statistics to aid you in better understanding your own health and fitness habits.

Description and Features

SpringFit lets you add, edit, and delete fitness activities. Each activity is comprised of activity name, duration, date, calories burned, and distance. You may add activities manually or upload a JSON of the appropriate format to upload many activities simultaneously. Activities can currently be sorted by calories burned. Four graphs are generated based on the data and can be navigate to via the navigation bar on the UI. The graphs update automatically when any activity is added, deleted, or edited.

Getting Started

Dependencies

Note that you will not beed to install or download any of the following for installation, as they are packaged with the program; see installation section for installation.

- Spring Boot with Maven framework
- JDBC mysql connector jar to connect to database.
- Org.json library for parsing JSONs
- Thyme leaf template for front-end
- Chart.js (this will NOT require a download, as the library is currently implemented via cloud).
- Highcharts.js

Installing

- Simply clone from Github and open with IntelliJ.
- Under the Maven tab, the user may need to click Reload All Maven Projects, Generate Sources and Update Folders For All Projects, as well as Download Sources and/or Documentation. The user should then restart IntelliJ.

Executing program

- You must first create a MySql table to contain the data. It is recommended that the table be named “activities.” The table will need the following columns: id, activity_name, duration, calories, distance, date. Please ensure that the table is constructed exactly as depicted on the MYSQL.png image.
- Navigate to the storylineParser package and open the DAOJDBC class. Under URL, add your MySql Schema, not the name of the table itself. Add user (default root) and password.
- To run the program, right click MyFitnessAppwithMvcApplication class and press run.
- To manually fill the table with a JSON, run Main under the storylineParser package. This will load the storyline.json in the resources package.
- Note that the app only supports the exact JSON format of the included storyline.json.

Known Issues

The calories burned per year bar graph will not load unless there are at least five years of data. Uploading the json by navigating to upload within the application UI or running the storylineParser Main should suffice.

Authors

•Zachary Sylvane

•Donghyeon Kim (Sam)

•Pradeepti Reddy Tandra