Project Report

Project name: FitHub

Team Canvas group: A13

Team Members, Email, and Github username:

Kaylee N Trevino, kntrevi1@utexas.edu, kntrevi1

Christopher G Saenz, schrodingersfridge@utexas.edu, ChriSaenz

Michelle Wen, michellewen@utexas.edu, michellewen3

Andrew Wu, andy22wu@utexas.edu, wuhoo16

Github repo link: https://github.com/UT-SWLab/TeamA13

Deployed app link: https://imbdproject.wn.r.appspot.com

Motivation and Users: Fitness enthusiasts of all ages and experience levels

Requirements: All project dependencies and required python libraries are listed in the requirements.txt file in the Github buildfiles.

Testing

Since all of the backend infrastructure for this web app project was done in Python, we used the unittest framework to test all of the Python methods in our main.py that communicate

with our remote mongo database. This required setup prior to the TestMain to connect to a new testing remote database with collections that would be setup parallel to our actual 'phase2Database' and avoid corrupting our official database contents. Our unit testing file includes 7 test methods that cover all of the CRUD operations that are used in our backend including creating documents for each of the 3 model collections, loading documents from each of the collections, and cleaning the database by deleting the collections. Note that since our backend and web page implementation never changes the contents of our database after initialization, we do not need a test for updating documents in the database. In addition, our calls to the initialization methods also verifies that our API helper methods are working as expected and that our API keys are not expired. The verification of the database and array lengths, unique ids, and arrayIndices verify that we have enough instances for each type of model and are storing unique instances in the order that we expect them to be in.

For the frontend and GUI, we used Selenium to test multiple components of our website. We tested each of our three model pages separately, our instance pages, and our home and about page. Within each of these we also tested components such as the carousel, the navigation bar, links to other pages in our Bootstrap tables, attributes on the model cards and instance pages, empty attributes, pagination, bootstrap components such as cards, and more. We attempted to test many components, attributes, links, etc. in a way that would give us a better coverage of the website when testing and would ensure that when code is changed, we would have consistent notification if a bug is introduced.

Reflection

Phase 1

In this first phase, we learned how to reference and pull from multiple APIs and easily retrieve data from different sources. It was really interesting to see how Youtube and Google captured complex resources such as a Youtube channel and represented the data in a JSON format. After reading documentation, our whole team gained invaluable experience with how to access data in a RESTFUL manner for developer use cases. Throughout the development of the different parts of our project, we also got more and more comfortable with using Github branches, storyboard, and issue tracker for project management. In addition, we were able to apply the basics of HTML, CSS, and Bootstrap learned in class to a real-world use case and design the different pages on a website with the end user in mind. The majority of our team has never worked with frontend development before, so directly using Bootstrap elements to simplify the web app development and layout was a very helpful skill that we picked up during this phase. Last but not least, we also learned a lot about the details of MongoDB and how to use PyMongo to access and store large datasets to a remote database.

Although we learned a lot, we also had many struggles during this phase. For example, a few of our members were unfamiliar with the Github workflow, causing code integration to be extremely inefficient. In addition, not enough local testing was done before pushing to the public repo. We also had trouble with formatting the model grids nicely, especially with centering alignment. Also, some of the image URLs pulled from the Google Search API were broken, resulting in images not showing for both exercises and equipment models. Additionally, when moving to the next picture on the carousel, the picture sometimes shows up at the bottom of the screen then moves up to its position; this means the carousel still works, but we want to improve on this in the next phase of this project. Finally, calling data from APIs took way too long, around 45 seconds to pull over 200 exercises; therefore, we needed to use MongoDB database to

store data so the APIs are not called during deployment. However, the data and the amount of data we stored in the database still needs to be improved in the future.

Alternatively, we also had many things that went well this phase. Fortunately, the splash page development as a whole went quite smoothly as the navbar was easy to work with. The creation of the channel instance pages also went well, and the data could be easily found in the Youtube API. We were also glad that our images for each team member in the About page worked great after cropping. They very nicely fit on our cards for each member, and we were very happy with how that page was finally completed visually. We were also fortunate that our Google Custom Search API worked well for easily searching images needed for the exercises and equipment page. This allowed us to easily get a large amount of multimedia required for the instances of the three different models. Finally, we also were glad with our idea to use a 'dev' branch and individual branches for each person. These individual branches are updated with the code each person is currently working on, the dev branch has the most updated finalized code, and the main branch holds the code that is due for each phase.

Phase 2

In this second phase, we learned how to further work with the API's we previously mentioned to aggregate even more information. We learned more about how to pull data from the new ebay API, store the data in a remote mongoDB database, and then read the data from the database in order to dynamically populate each of our three instance pages. Our team used MongoDB and thus learned how to work with a NoSQL database, where the structure of the data was extremely flexible and could be designed to match the attributes of our backend objects using key-value pairs. We also gained experience with using a Python driver pymongo to easily capitalize on the powerful operations supported by the mongo database such as easy queries with

filtered keywords to help create our cross-model links. We also learned how to use pagination as well as how to make each of our model pages more attractive through the use of bootstrap and CSS styling. Finally, we learned how to use testing to ensure that our website continues to work the way we have intended, especially as we make small tweaks and our codebase grows larger and larger. We not only learned how to test the backend python methods with unittest, but also how to use Selenium to simulate the role of the user and test that navigation and links throughout the webpage are working as expected. For this particular phase, we did not use Mocha since we currently have no interactive Javascript components in our webpage. All in all, adding in testing showed up the importance of automating the process of testing basic units of our code so that it is easy to find bugs right as they occur.

For the future of this project, we are hoping to add more test cases to better ensure that our website works as intended. We believe that our main.py is getting extremely large and dividing it into modular style would really help our overall project organization and make it easier to make smaller, targeted test files. In addition, from this phase we were met with a lot of extra work since all of our APIs return direct image URLs and therefore we have no control over the reliability of the image URL. This issue was especially obvious for the equipment model page, since ebay sellers often remove their image from the ebay website, which causes new broken images even after we have already manually blacklisted and cleaned broken links. In phase 3, we can look into a long-term solution of this issue by using an image scraper to store the image files statically in our project directory. Third of all, we are hoping to continue to make this site much more user-friendly by adding features such as a search bar and filtering options on all 3 model pages. In addition, we need to improve the image quality and resize the pictures for some of the equipment instances, which Christopher unfortunately did not have time to get to

this phase. Last but not least, we are hoping to continue improving our Github skills and familiarity; we have become more comfortable with Github as a team and are hoping to continue improving in order to reduce integration time and prevent the time-consuming task of resolving merge conflicts.

Although we faced many roadblocks, there are also many tasks that went well for our team during phase 2. For example, working with the API's went smoothly, and we were able to effectively make multiple calls without many issues. In particular, taking initiative on registering for the ebay developer account and providing all of the documentation and resources for Christopher made switching over to using the ebay API pretty streamlined. Secondly, our transition from using static data to dynamic data was completed very easily and we did not have any major issues when integrating this change into our code. Third of all, we were able to implement pagination in a way that nicely complemented the pages we had already built and is dynamic based on the number of instances for each model type. Something that also went much better this phase was that team members were more active in the Slack channel and Kaylee did a great job of being a team leader and taking initiative for all team meetings. Last but not least, we already had clear method definitions for interacting with our database, so testing the backend was very straightforward.