Diamond Hands Investment Fantasy League

Individual Contributions

Software Engineering

14:332:452

Group 3:

Aarushi Pandey

Aarushi Satish

Apoorva Goel

Christine Mathews

David Lau

Jacques Scheire

Jawad Jamal

Krishna Prajapati

Riya Tayal

Sahitya Gande

Yati Patel

Yatri Patel

1 Project Management

All group members contributed equally to this project. We will breakdown our reasoning for that in the following sections, but wanted to state it clearly in the beginning.

1.1 Accounts for the Project

For this project we needed multiple accounts. To start, we all needed to have GroupMe and Discord accounts for easy communication. This allows us to easily set-up meetings and also streamlines group communication. For code organization and coordination, everyone had to create a separate Github account or use an existing Github account to be able to collaborate on the project. We also all created accounts for Figma and Creately to create mockups for the frontend and diagrams for services and features, respectively. This helped all of us have a say in what the project would look like as well as allowed everyone to help anyone that needed it.

In addition to all of the accounts required for starting our project, we all needed MongoDB and IEX Cloud accounts. A MongoDB account was required in order for each of us to have our own database for testing. An IEX Cloud account was required in order for all of us to be able to request stock information.

1.2 Organizing Meetings

We have a standing meeting every Friday to meet and discuss what everyone has accomplished in the past week and what our plans for the next week will look like. During this time, we also try and plan for additional meetings that we may need to have to finish up any pending submissions. In such cases, we normally meet the day before and the day of the submission to assess the amount of work completed and the amount still pending. We all are also a part of group messaging platforms that allow anyone to suggest a meeting when required. If someone is unable to attend, then we have a chat that summarizes what was discussed during the meeting, as well as responsibility breakdowns for the week. There is no designated person that summarizes the meetings, so anyone can summarize the meeting if they notice that no summary has been posted already. Anyone can additionally add to the summary if something was missing.

1.3 Coordinating Activities

During our meetings, we all take time to break up work assignments for both the reports as well as the project. Everyone takes on a different amount of issues and a different amount of work on a weekly basis, so the breakdown of responsibilities ends up equalizing. Additionally, if any person or group is having any trouble, then they can ask the rest of the group and someone will help them. Therefore, to a certain extent, every issue is a group effort.

1.4 Installing & Maintaining Developer Resources

Since everyone is working locally and pushing to the repository, everyone has to make sure that they are maintaining all of the different versions required for the different resources being used within the project. If a new resource is added then it is the responsibility of the person adding it to add that change to the package-lock.json file, as well as inform the group of the new resources being added via a Github pull request or via a message in our group chat. All other group members can run a simple install command to receive the package and its dependencies.

1.5 Formatting & Combining Contributions

We all created a formatting system for our database in the early stages of our project. So whenever anyone addresses an open issue, all they have to do is follow the existing system. If anyone adds anything that does not currently have a formatting system, then we discuss how it should be formatted during one of our group meetings or via chat.

A method that we have used to maintain proper formatting and cohesion when combining contributions, is through code reviews. In our Github repository, group members are only allowed to push to a separate branch, which is a Pro setting that the repository owner can enable. This prevents anyone from pushing to the master branch, which prevents merge conflicts and unresolved bugs from affecting the entire project. After making all of their edits and commits on a separate branch, the branch owner can make a pull request, which requires a minimum of 3 reviews before it can be merged. The reviewers can make comments on parts of the code within the pull request and request changes to any part of the code. After addressing these changes and receiving the approvals, the code can be merged.