List your team members, providing name, email, and role for this project. Also provide the URL where your live Heroku-hosted web front-end can be found.

Soniet Berrios(sfb219@lehigh.edu)- Project Manager
Lizzie Shaffran(ehs219@lehigh.edu)- Admin
Tamara Johnson(tjj218@lehigh.edu)- Web
Emily Weston(ebw219@lehigh.edu)- backend
Kelli Frank(kdf219@lehigh.edu)- Android

https://sleepy-dusk-34987.herokuapp.com/

Back-End Server

Describe the changes to the set of REST API endpoints that your team agreed to implement.

-Added a lot of them because each table needed its own post route and at least 2 get routes
-Changed what the routes took in, like some get routes need 2 inputs, etc
-Some of the routes needed to return something other than a json object, like an int

Did your data model deviate significantly from the model discussed in class? If so, why?

- No it did not deviate

Were there any issues that kept the back end from reaching 100% functionality? If so, what were they?

- Biggest issue, didn’t realize that she needed to redo the whole backend basically for this face
- Time
- Couldn’t figure out how to check if the correct password was entered for the corresponding username, so currently the backend does not have password authentication. It is set up so that a user cannot register with a username that is already being used, so that is some type of control, but not anywhere near the security provided by a password.

Is the back-end code appropriately organized into files / classes / packages?

- Yes

- Functions, prepared statements, etc. are organized very well into separate files and classes. All the routes are in one file, and it's a little difficult to sort through and find what you want at the moment.

Are variables, functions, classes and packages named well?

- Yes, they're all named well

Are the dependencies in the pom.xml file appropriate? Were there any unexpected dependencies added to the program?

- Yes they're appropriate, only added one and it wasn't unexpected since we were told to add it in class
- Added a mockrunner dependency to help set up a mock database to write junit tests. However, still couldn't get the tests to work.

What was the biggest issue that came up in code review of the back-end server?

- The biggest issue that came up in code review would be that not every single thing works yet.

What technical debt do you see in the current back-end server implementation?

- Every table has its class and its own set of methods and they're all very similar but we're not sure how to make this better.
- To make the routes, ended up having to make a new connection for each table and we know that there has to be a better way to do this than connecting to the database up to 5 times, but we're not sure how to fix it.
- All the routes are in one file and it's really long
- Need to change it so we use query parameters for the routes
- Used the query parameters method Spear mentioned to authenticate the user on the /messages (get all messages from table) route, and it works. Kept the code for that in App.java, however did not add it to the rest of the routes because android and web did not have a chance to change their code to adapt to it.

Describe any refactoring that was performed to reduce technical debt from the last phase

- Basically, redid backend in phase 2, we only needed 1 tables and in this phase we needed 5, also made votes updates that wasn't working it
- Put all methods, prepared statements, etc. for each table into a separate class to keep everything organized

Are the unit tests for the back-end server complete/satisfactory? If yes, why do you have confidence in your answer. If not, what's missing?

- The unit tests are not satisfactory.
- Chose to spend more time on the actual code itself than writing the tests for it

Web Front-End

Did you use MVC to organize your web front end? If so, did it help? If not, why not?

- No, they chose not to because they were unfamiliar with it and since there was a lot of catching up to do with web, we focused on that

Does the user interface match the interface for the Android app? Why or why not?

- Not really, we were very focused on getting things to work that design sort of fell through.

Is the web front-end appropriately organized into files / classes / packages?

- Yes, for the most part

Are variables, functions, classes and packages named well?

- They are named well and have comments that explain things if necessary

Are the dependencies in the package.json file appropriate? Were there any unexpected dependencies added to the program?

- I believe so

What was the biggest issue that came up in code review of the web front-end?

- Could not have a complete code review because of unforeseen personal circumstances.

What technical debt do you see in the current web front-end implementation?

- The biggest debt would be that it is incomplete, so the next person will have a lot of catching up

Describe any refactoring that was performed to reduce technical debt from the last phase

- Finish the tutorials from last phase to get the web up to date

Are the unit tests for the web front-end complete/satisfactory? If yes, why do you have confidence in your answer. If not, what's missing?

- Not complete

Android App

Describe any significant deviations from the UI design discussed in class.

- No, at least not any that we are aware of

Is the android app appropriately organized into files / classes / packages?

- Yes

Are variables, functions, classes and packages named well?

- Yes, for the most part

Are the dependencies in the build.gradle file appropriate? Were there any unexpected dependencies added to the program?

- Yes, the dependencies are appropriate and no unexpected dependencies were added.

What was the biggest issue that came up in code review of the Android app?

- Can’t figure out how to get the votes to update instantaneously

What technical debt do you see in the current Android app implementation?

- A lot of the rest requests are written out over again throughout the code and this could have been reduced by doing a method

Describe any refactoring that was performed to reduce technical debt from the last phase

- Basic renaming of variables and re-organizing the code

Are the unit tests for the Android app complete/satisfactory? If yes, why do you have confidence in your answer. If not, what's missing?

- Unsure if they were finished, but when the code review was done they were incomplete

The Admin App

Describe the process that your team decided upon for managing the process of registering and activating users.

- Decided that it would make sense for someone to suggest their own password and we check the email manually and then activate it
- Now realizing that this idea may not have been when it comes to security because there's no way of making sure the user gave a valid email account since we don't send them an email

Is the Admin app appropriately organized into files / classes / packages?

- Yes

Are variables, functions, classes and packages named well?

- Yes

Are the dependencies in the pom.xml file appropriate? Were there any unexpected dependencies added to the program?

- Yes and added one for sendgrid but it wasn't unexpected

What was the biggest issue that came up in code review of the Admin app?

- The biggest issue was that there were no unit tests made

What technical debt do you see in the current Admin app implementation?

- The organization of the app isn't very good, for example she used a bunch of else ifs and we could have done a switch.
- Wrote different prepared statements and methods to delete each table, when there probably is a way to do just one

Describe any refactoring that was performed to reduce technical debt from the last phase

- Added 5 new tables and deleted tbldata
- Also made things that the app can do more general -- like not made for just one table

Are the unit tests for the Admin app complete/satisfactory? If yes, why do you have confidence in your answer. If not, what's missing?

- Not completed since her laptop broke and she submitted it to be fixed.

Project-Wide

How well did your team estimate effort?

- Everyone started earlier from last time, because we learned from phase 2.
- We knew it would be a lot of work, but we thought it would be manageable but it ended up being harder.

What techniques (team programming, timelines, group design exercises) did you use to mitigate risk?

- Set up times to meet at the beginning of the phase and set up times that people could go to office hours together
- Set dates to have trackable progress (meaning functioning additions to code pushed to bitbucket)

Were there any team issues that arose?

- Waiting for others to do their part because we thought that so many things relied on one another

Describe the most significant obstacle or difficulty your team faced.

- Figuring out how to work log-in

What is your biggest concern as you think ahead to the next phase of the project?

- Since many unforeseen circumstances happened during this phase, I believe the biggest concern for the next one is getting everyone caught up so that this doesn't affect future phases.