Backend summary report:

There are two python files that compose the backend of our project currently. api.py and database.py, database.py handles all of the database calls while api.py interprets requests and then calls database.py as appropriate.

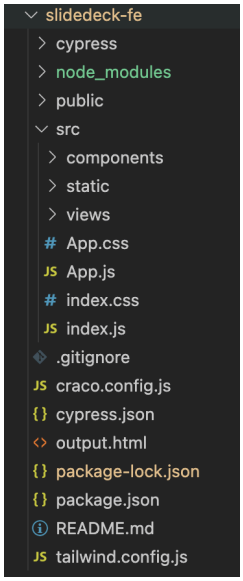
Database.py has 85% coverage. The only parts that are not covered are the error handling for cases where calls to mongodb fail, a brief bit of setup for the mongodb connection, and an unfinished method at the end. The calls that use the mongodb library will only fail if something outside of the scope of my code happens. It is more work then it is worth to simulate the failures necessary for testing, and since they are wrapped in try-except blocks a failure in them will not break the code.

Api.py has 85% coverage the only things not covered are some setup code as well as some returns for errors. All of the logic is tested or so simplistic that it is not worth testing. Particular many untested lines are the same authentication code copy paste between methods.

In my opinion all of the logic in this code is effectively tested and covered. The only things not covered are error returns and logic so simplistic that it is nearly fail proof.

Frontend summary report:

Our frontend consist of a react application whose folder structure looks as follows:



Where the main pages in our web application consist of views that are populated by components for instance the Dashboard view has an enter data component.
In the current state of our application we have most of the backend API figured out and we are integrating all of this to our newly designed user interface. Couple of weeks ago we re-designed our user interface, therefore we are reconnecting the APIS to the application. Hence, our front end testing right now is testing the login page as well as the login functionality, the dashboard page as well as each of the dashboard subpages. At all times during our testing, while our test

suite tool is navigating the app is also making sure that the router displays the correct url meaning that the user is being taken to the page it's supposed to navigate. In addition our test cases also make sure that each page contains all the components following the guidelines from our clickable prototype. For instance our login pages must contain 2 input boxes, one of type text for the username and another of type password for the user password, as well as a LOGIN button and a forgot password prompt.

For the 2 main pages login and dashboard the Frontend tested as expected with all the implemented functionality working perfectly and all the components were displayed correctly. Meaning that our application passed all the assertions of our frontend test suite.

Right now the front end testing does not cover all the frontend of the application since there are still lots of features to implement.