ASE Pair Programming Assignment One
04/10/16

Akanksha Gupta (ag3749@columbia.edu)
Ishan Guru (ig2333@columbia.edu)
(ASE Avengers - JP Morgan Team)

GitHub Repository URL:

https://github.com/guptakanksha1992/ASEPairProgramming-HW

Description:

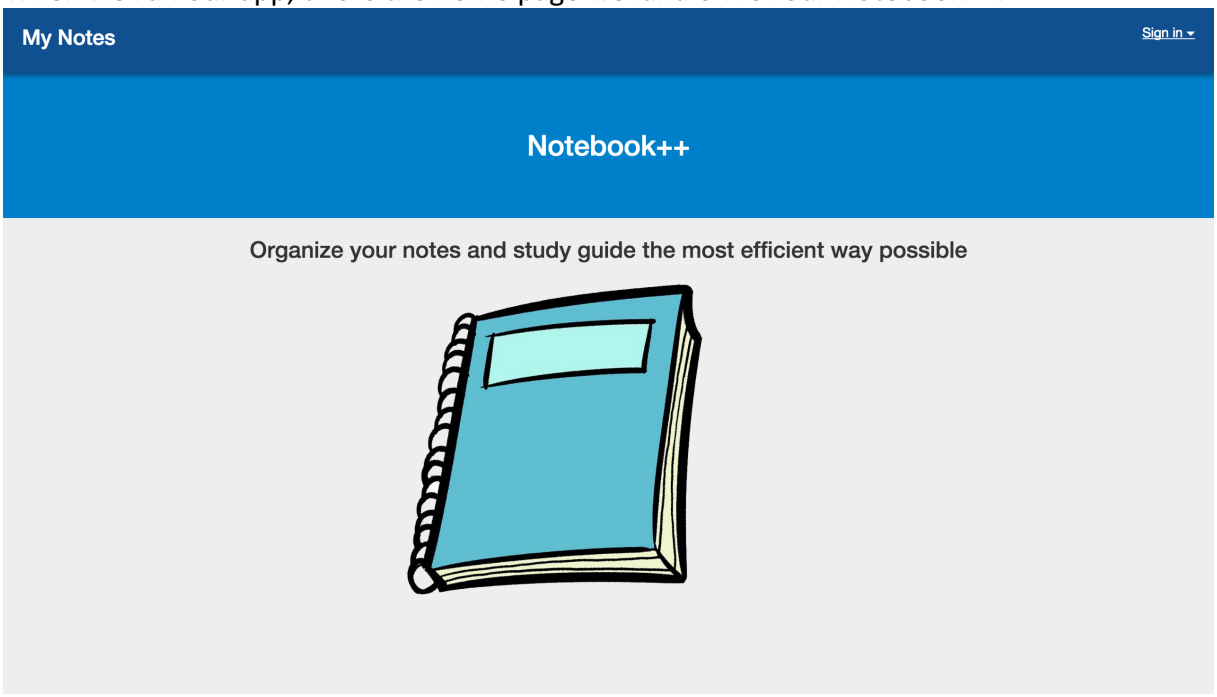
Our team chose the Meteor framework for full stack development for our Advanced software engineering project. The database technology we used is MongoDB. For this pair programming assignment, we developed an app "Notebook++" that is a dynamic app that allows users to "take" notes pertaining to some subject, along with logging in and out of the system to make sure notes are personalized.

Our app "Notebook++" is a one stop solution for students to help manage their day to day notes. The app has an option for a student to register an account if he/she is using it for the first time or sign in if an account already exists. Next the dashboard opens where the student can see their previous notes and also add new notes based on date and subject. In addition to this, our app provides a functionality for a student to add specific notes for review later. The student can add the notes on the review list and after review can remove them from the review list.

After adding/deleting/reviewing notes, the student can safely logout from our app. Also, used JSLint and the Mocha framework that Meteor supports for static analysis and testing, which can be seen later on in the document. We ran few test cases to get the hold on technology stack since both of us are new to Meteor, and ensured that all the methods written for the application passed certain tests. Further we used Travis CI for continuous integration and build. Version control is being done using GitHub, and automatic build control is being managed by Travis CI.

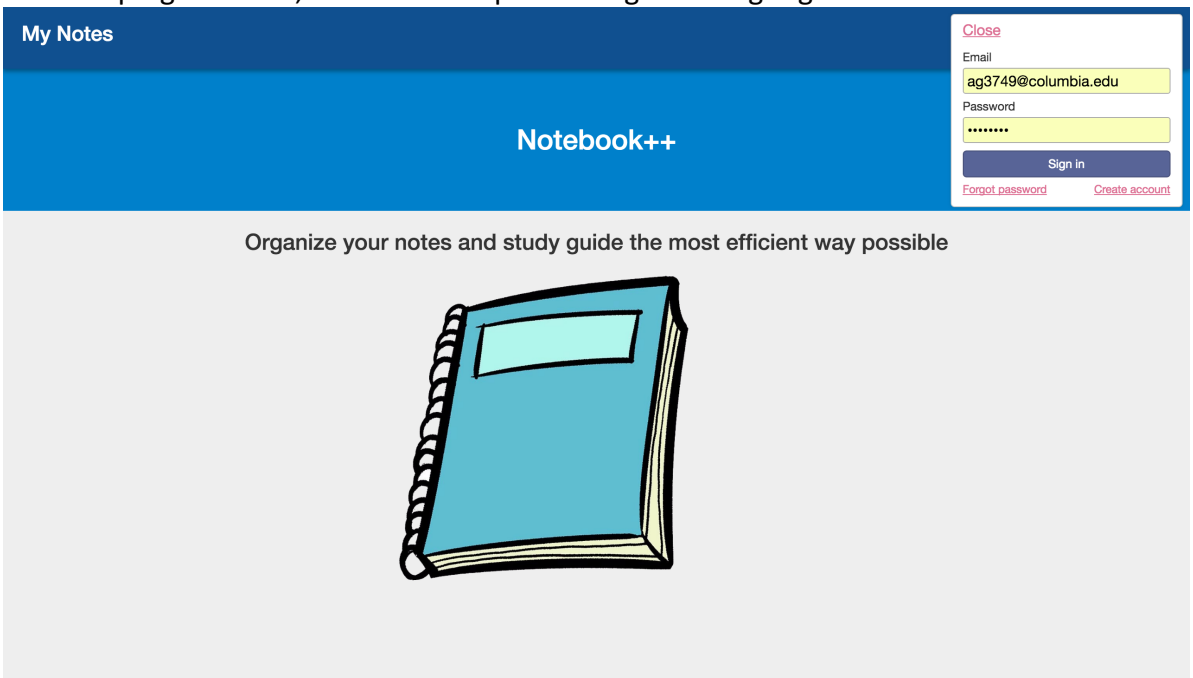
Home Page

When we run our app, this is the home page we land on for our Notebook++.



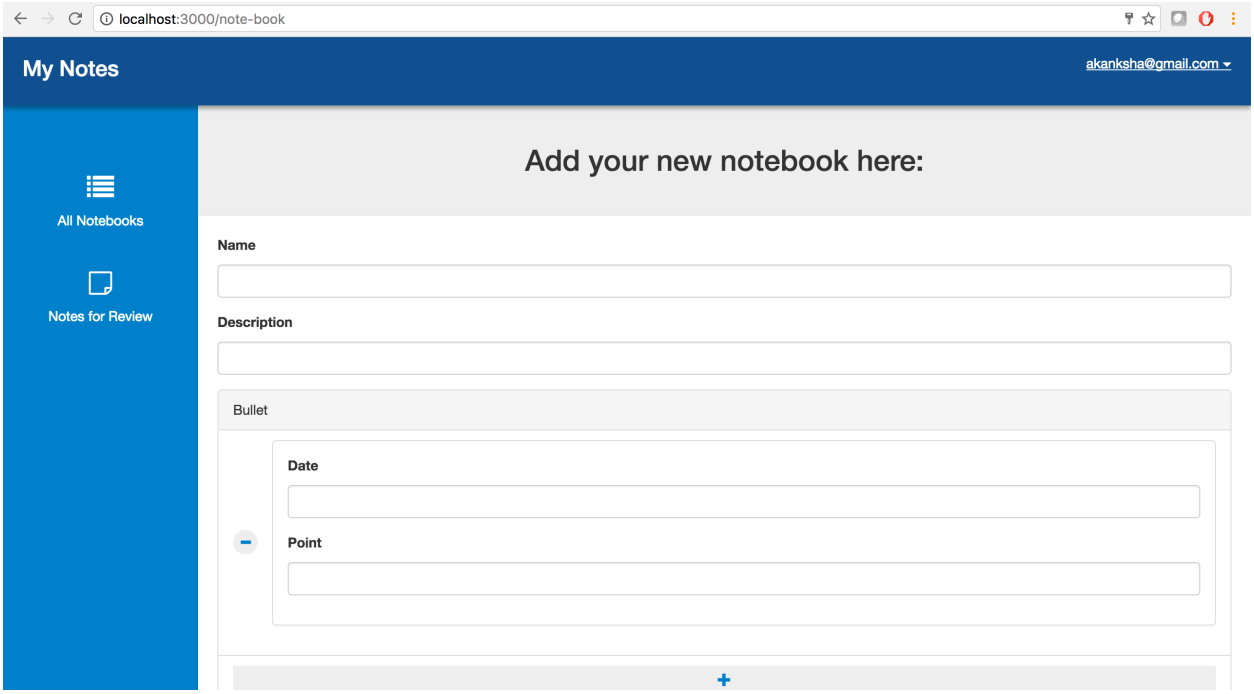
Sign In

At the top right corner, we have the option to sign in using registered account or new one.

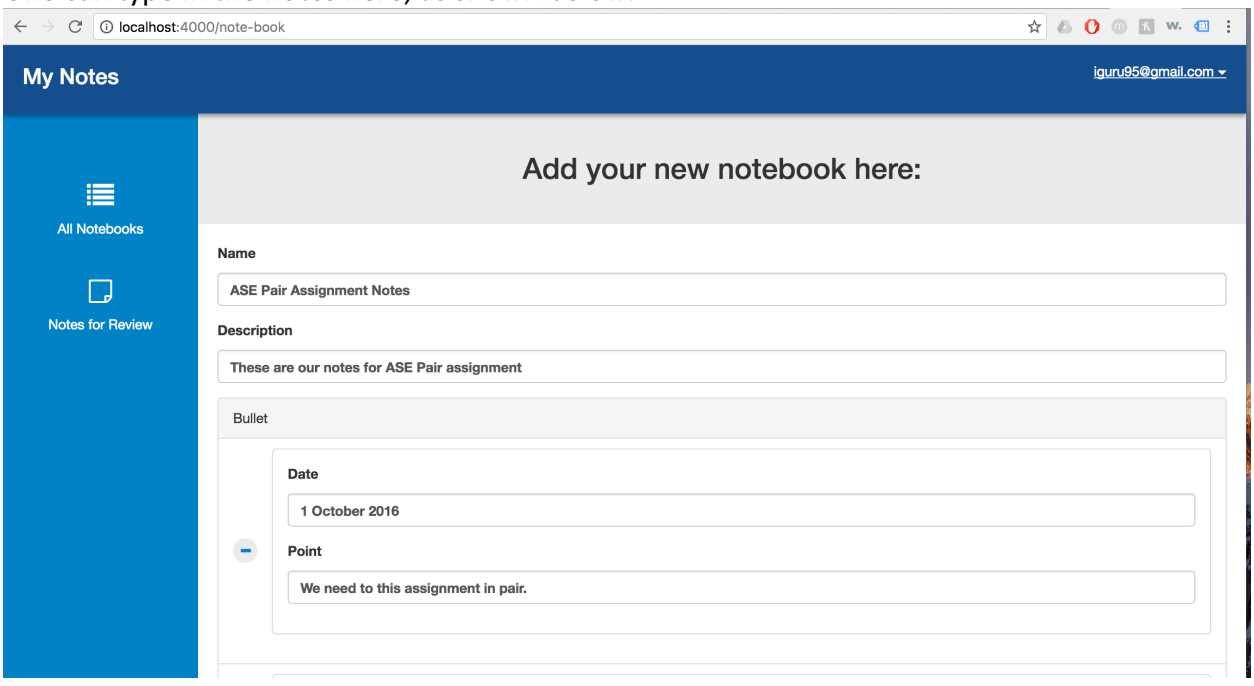


Adding Notes to Notebook

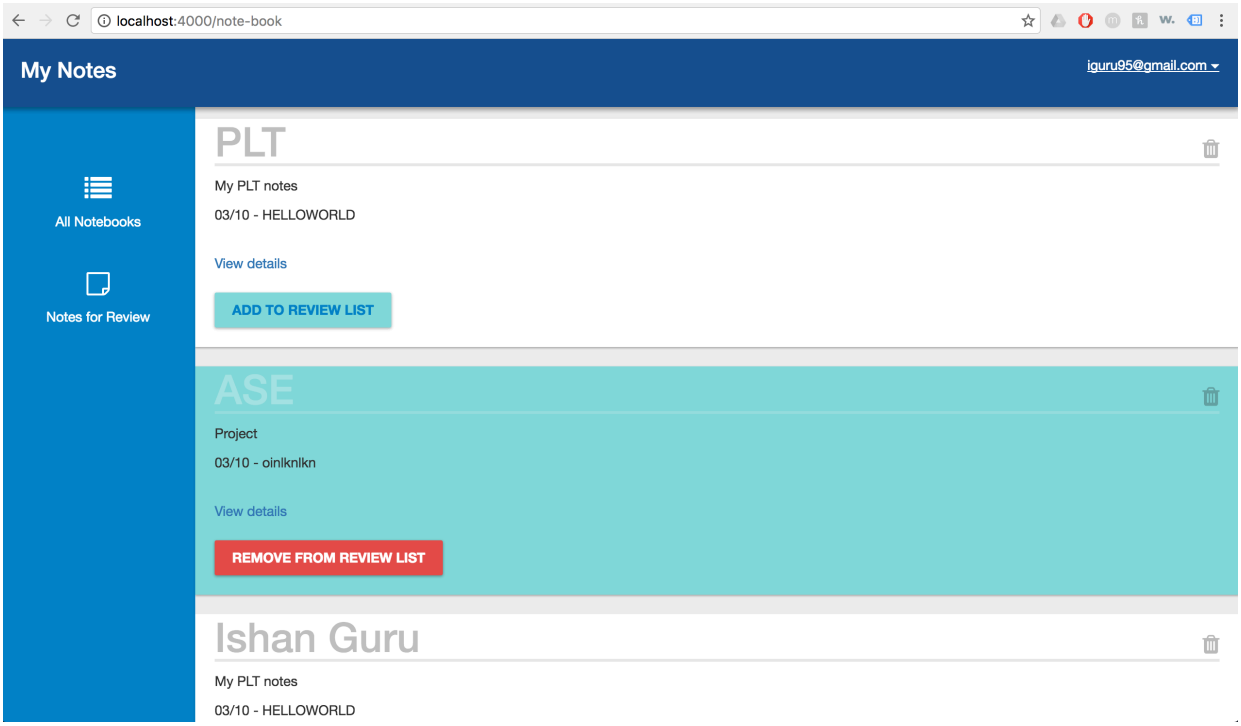
After sign in, personalized dashboard pops up that gives the option to add new notes.



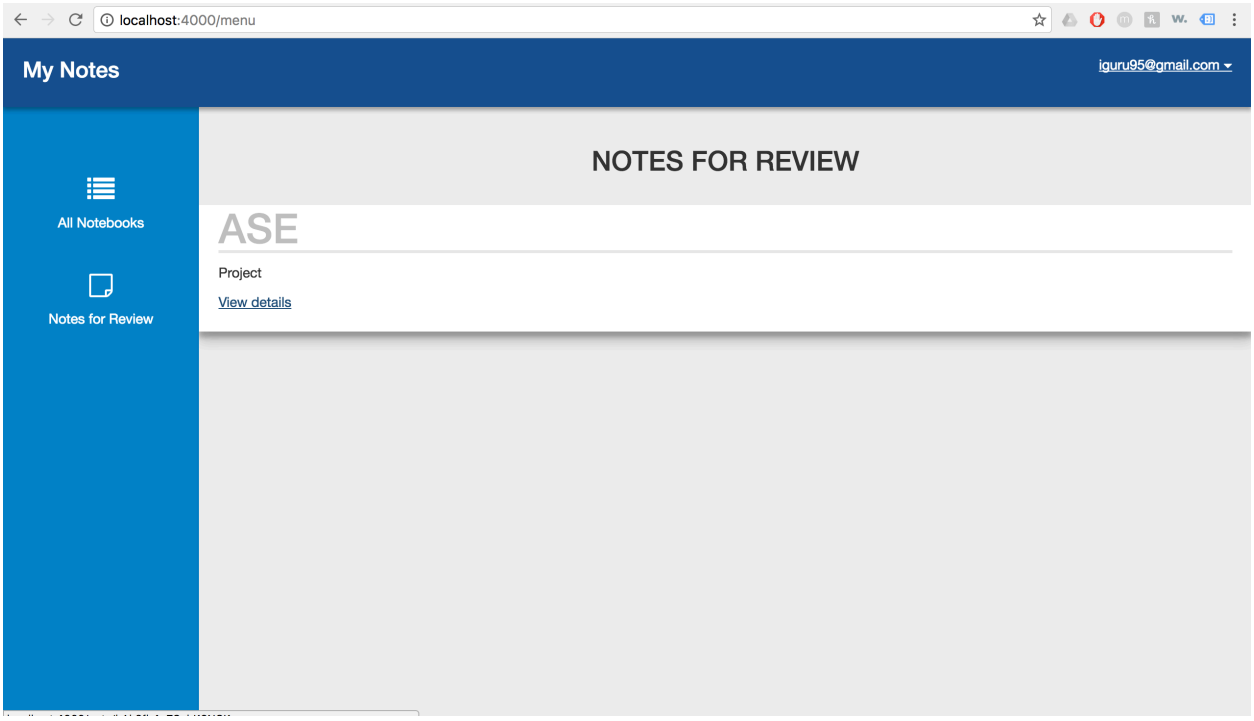
One can type in the notes here, as shown below.



After adding the notes, one can view the notes added for various subjects with dates and key information as shown below.

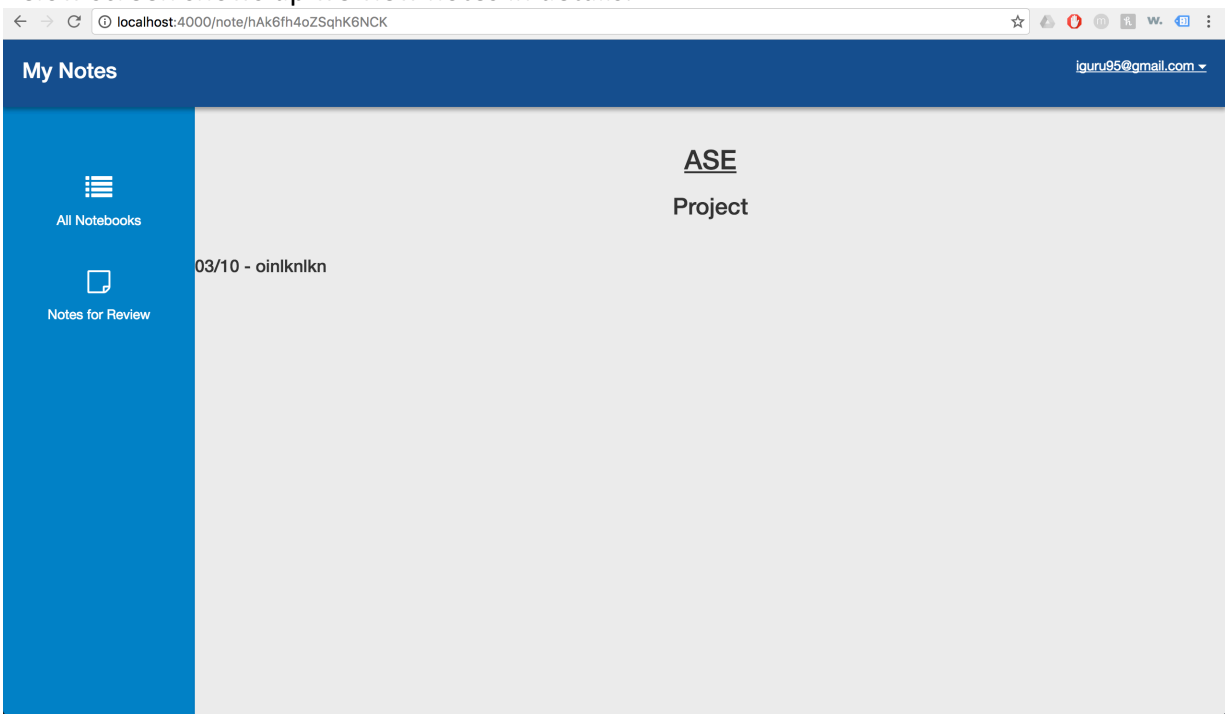


Further, one can mark notes that one wants to review later to the Review List by clicking on Add to Review List option (Green Button). Later, one can delete notes from Review List by clicking Remove from List. (Red button).



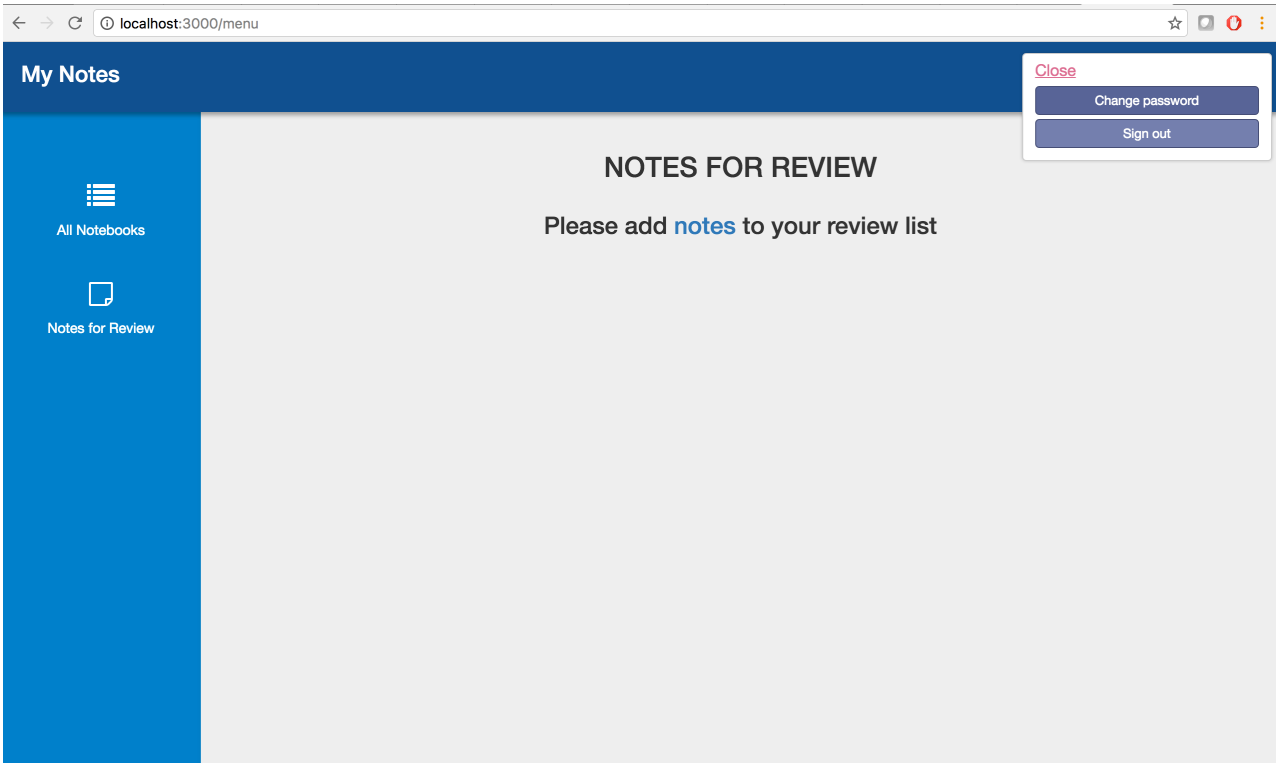
View Details

Below screen shows up we view notes in details.



Logout

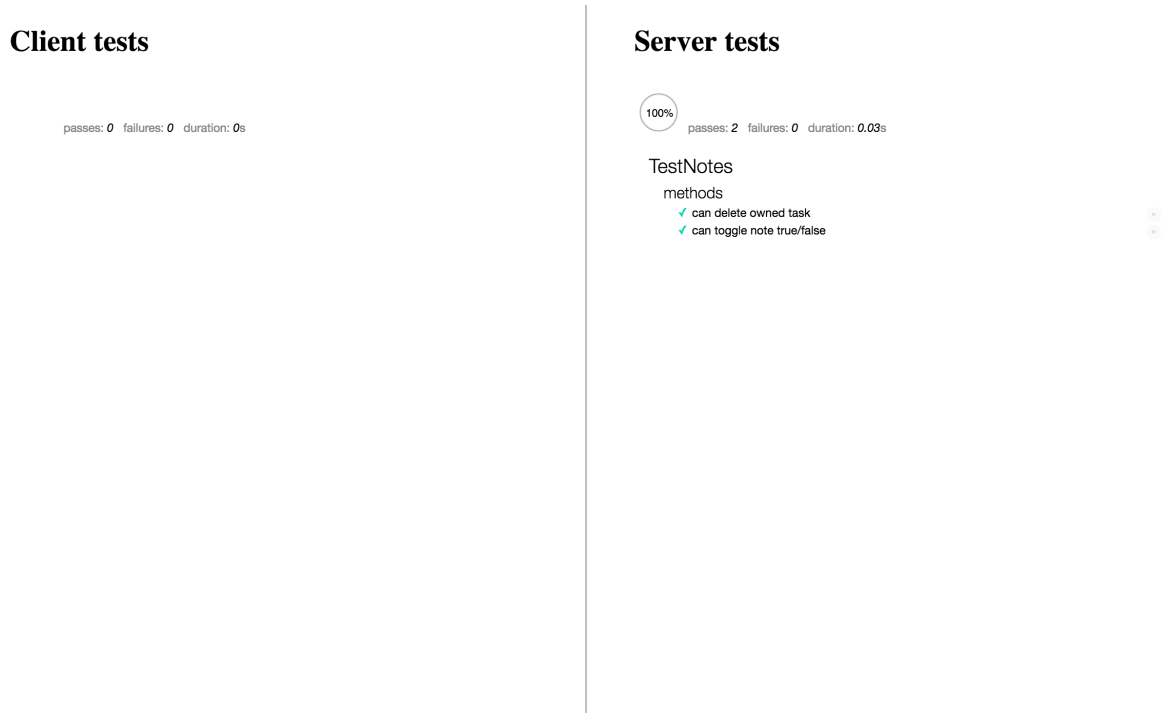
One can click on Logout to safely logout of our app.



Testing

We experimented with unit testing for this pair assignment. We used the JSLint framework with Mocha package as it provides excellent testing options with Meteor. We wrote unit test cases to check basic functionalities like deletion of notes, and the toggle to add notes to review list or remove from the review list, etc. Since Mocha generated a UI side display, it was simple to see which tests were failing and passing.

Below is the screenshot of our testing. Both the tests were written for server side testing.



Static Analysis:

Further we ensured we run a static analyzer on our code base, using the learning from our first static analyzer and code smell assignment. We used jshint as our static analyzer, as there is a simple package addition to meteor that allows for analysis each time the project is run.

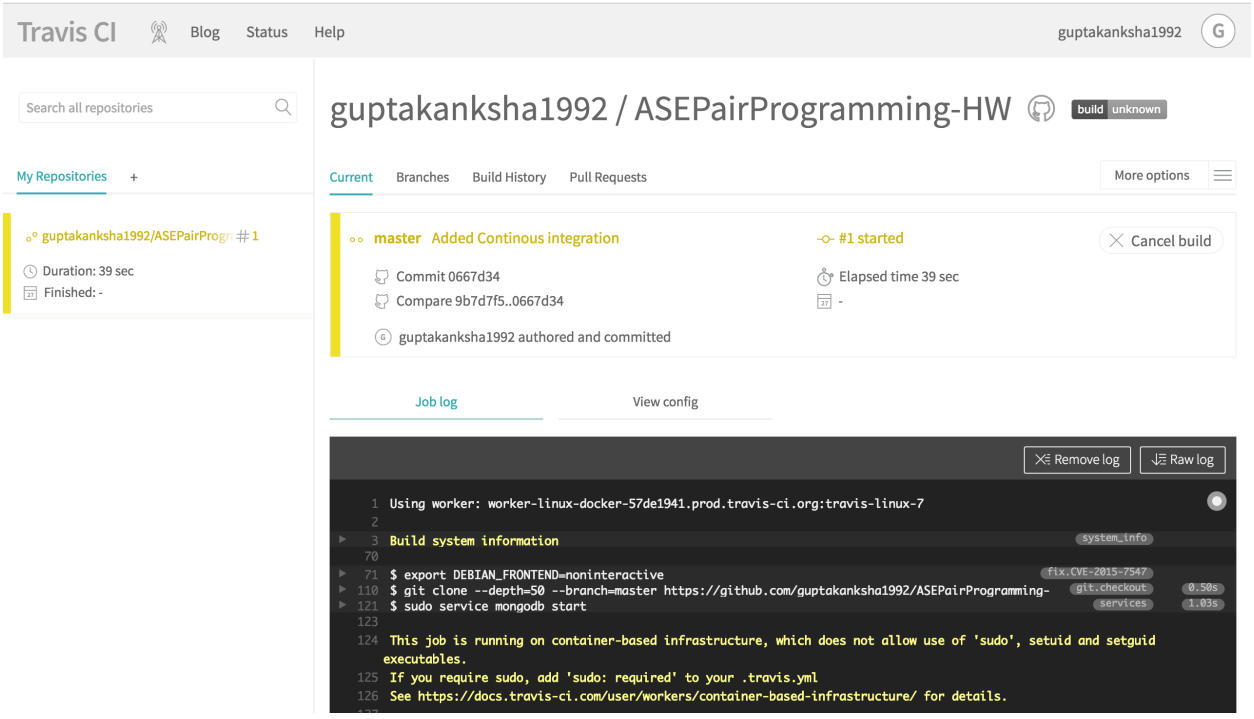
We found some suggestions and code smells from the static analyzer. The results are attached in the Appendix.

Version Control

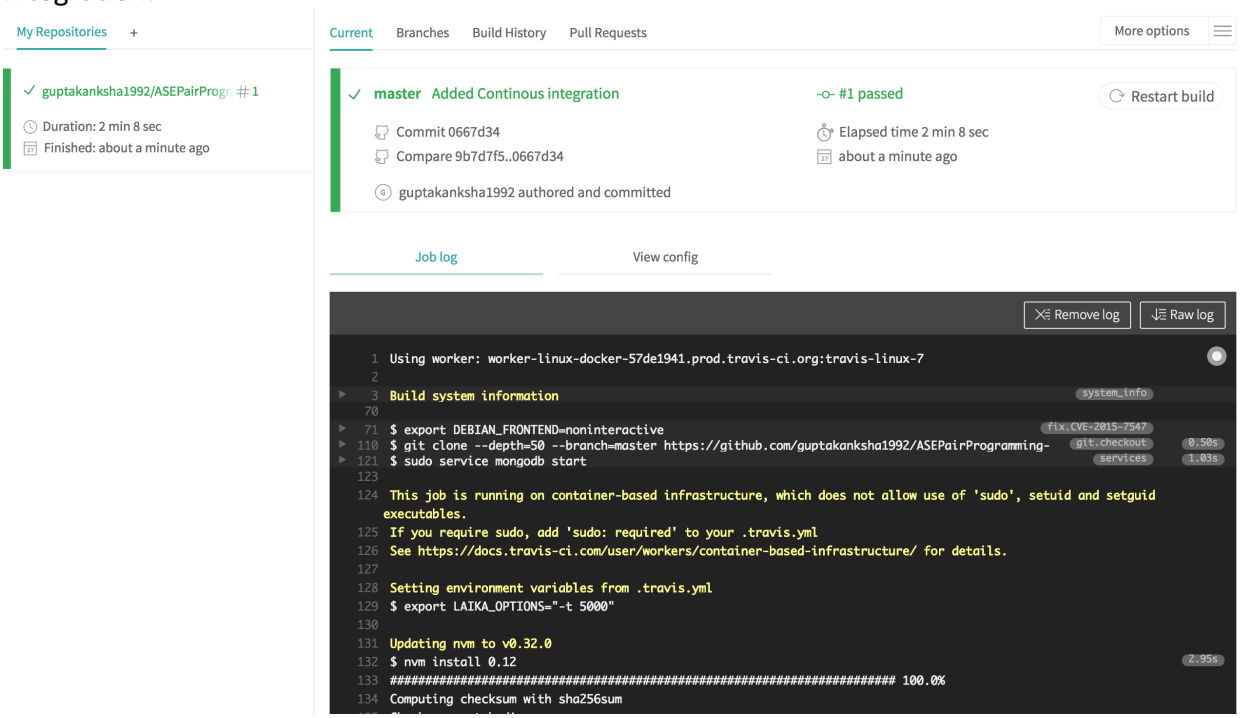
We used GitHub to version control our app. Since we did pair programming, we were always together while coding and so pushed our changes time to time to GitHub.
The link to our GitHub repository:
https://github.com/guptakanksha1992/ASEPairProgramming-HW

Continuous Integration

We used Travis CI to build our project for Continuous Integration. The .yml file was also pushed to GitHub so that we can use the features of Continuous Integration.



Also it displayed detailed messages to prompt about the status of the build and continuous integration.



Experience using each part of your team's chosen technology:

We had a great experience on working on this new powerful web framework, Meteor. Firstly, since both of us are new to the technology, we had to prepare ourselves for the technology. We watched video tutorials to prep ourselves for Meteor. Meteor is based on Node and JavaScript and so it was a great learning experience.

As far as the database is considered, we used MongoDB and we were pretty comfortable using it. Ishan had previously worked with MongoDB and that was an added advantage to us. Further for version control we used GitHub which we both were comfortable using. Travis CI for continuous integration was new to both of us but since we just had to make the configuration file and upload to GitHub, we found it easy and didn't run into any challenges.
Also our previous assignment on static analyzers helped us to easily run a static analyzer and find code smells in our code.

Did you run into any challenges?

Since both of us were new to Meteor, we faced initial set of challenges trying to understand how Meteor works and the concept of Templates in Meteor. We then referred the official site of Meteor and watched some video tutorials to prepare ourselves. The step by step guide and tutorials helped us understand how Meteor works and we learnt the best practices to code in Meteor. We did face some challenge while testing the unit tests. Since we used the JSLint to unit test functions, we were having challenges to think of unit tests and actually make them run on Meteor. Reading more about testing specific frameworks of Meteor and about Mocha package helped us in testing. All in all, we really felt that pair programming helped us do the tasks better and quicker.

Was there anything you wanted to do but couldn't get working?

We faced challenges in testing initially but with more reading and tutorials we understood the testing framework for Meteor and could get the things working and test our app. So in all, what all we wanted to achieve was achievable.

Appendix

Results from JSLint Static analyzer (Readme.md on GitHub)

dyn-160-39-205-47:ASEPairProg akankshagupta$ meteor lint

While linting files with jshint for the app (Server): lib/route.js:19:5: 'concise methods' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz). lib/route.js:29:9: 'concise methods' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz). lib/route.js:36:5: 'concise methods' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz). lib/route.js:43:5: 'concise methods' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz). lib/route.js:11:47: 'redirect' is defined but never used. lib/route.js:11:38: 'context' is defined but never used. collections/Notes.js:51:35: Missing semicolon. collections/Notes.js:63:35: Missing semicolon. collections/Notes.js:1:1: 'Notes' is not defined. collections/Notes.js:3:1: 'Notes' is not defined. collections/Notes.js:12:1: 'bullet' is not defined. collections/Notes.js:21:1: 'NotesSchema' is not defined. collections/Notes.js:34:16: 'bullet' is not defined. collections/Notes.js:73:8: 'Notes' is not defined. collections/Notes.js:80:8: 'Notes' is not defined. collections/Notes.js:84:1: 'Notes' is not defined. collections/Notes.js:84:20: 'NotesSchema' is not defined. collections/Notes.js:4:31: 'doc' is defined but never used.meteor test --full-app --driver-package=practicalmeteor:mocha collections/Notes.js:7:31: 'doc' is defined but never used. server/publish.js:6:11: 'Notes' is not defined. server/main.js:1:1: 'import' is only available in ES6 (use esnext option). server/main.js:3:17: 'arrow function syntax (=>)' is only available in ES6 (use esnext option).

While linting files with jshint for the app (Client): lib/route.js:19:5: 'concise methods' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz). lib/route.js:29:9: 'concise methods' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz). lib/route.js:36:5: 'concise methods' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz). lib/route.js:43:5: 'concise methods' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz). lib/route.js:11:47: 'redirect' is defined but never used. lib/route.js:11:38: 'context' is defined but never used. client/recipes/NoteSingle.js:9:12: 'arrow function syntax (=>)' is only available in ES6 (use esnext option). client/recipes/NoteSingle.js:11:16: 'Notes' is not defined. client/recipes/Notes.js:10:13: 'arrow function syntax (=>)' is only available in ES6 (use esnext option). client/recipes/Notes.js:11:16: 'Notes' is not defined. client/Menu.js:10:13: 'arrow function syntax (=>)' is only available in ES6 (use esnext option). client/Menu.js:11:16: 'Notes' is not defined. collections/Notes.js:51:35: Missing semicolon. collections/Notes.js:63:35: Missing semicolon. collections/Notes.js:1:1: 'Notes' is not defined. collections/Notes.js:3:1: 'Notes' is not defined. collections/Notes.js:12:1: 'bullet' is not defined. collections/Notes.js:21:1: 'NotesSchema' is not defined. collections/Notes.js:34:16: 'bullet' is not defined. collections/Notes.js:73:8: 'Notes' is not defined. collections/Notes.js:80:8: 'Notes' is not defined. collections/Notes.js:84:1: 'Notes' is not defined. collections/Notes.js:84:20: 'NotesSchema' is not defined. collections/Notes.js:4:31: 'doc' is defined but never used. collections/Notes.js:7:31: 'doc' is defined but never used.