<Tutify> Iteration 9

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Claudia Feochari (40000060)	**compgirl123**	153 points
Cynthia Cherrier (40005808)	cynthiac3	147 points
Jasmine Latendresse (40011419)	jaslatendresse	119 points
Tanya Multani (40008542)	tanyamultani	100 points
Bilal Nasir (40015010)	bilal101	100 points
Kasthurie Paramasivampillai (40025088)	kasthurie	117 points
Pierre Watine (40027675)	PWatine	126 points

Proof of Concept

The creation of a web-based application that bridges the gap between connecting students of various grade levels with various tutors from different fields of study is an absolute necessity. Hiring private tutors can also be pricey and hard to find for specific courses. Our application centralizes all tutors in a single place and provides students who could not afford traditional tutoring a more affordable way to have access to course material. Currently, we have a viable product, Tutify, that does just that. It allows students to search for tutors and courses, to manage their work through to-do lists and to have access to a resource page. Tutors can also send announcements, schedule meetings, send documents.

Project summary

Tutify is a web-based application that improves the way tutors and teachers interact with their students by providing them a more enhanced interactive learning experience. It allows students to search for tutors and courses and to connect with them. The app supports messaging and document sharing between tutors and tutees as well as document publishing for free exercises for the entire Tutify community. It has an integrated calendar to schedule appointments. That's right! Just having an account gives free learning resources.

Risk

1. Data provided by students and tutors on the application (their names, their email addresses). This might be risky in the rare case that a data leak occurred but this would be mitigated by using an appropriate database encryption system that will ensure that the data is kept safe and only permitted users are able to access it.

2. Copyright Fraud is also an issue as we need to ensure that services that are implemented in our application do not conflict with other services that have been implemented by other tutoring companies.

3. As this is the first sprint, time and effort can be underestimated at the beginning. The estimation will be adjusted for future sprints depending on how the team is performing.

4. The document sharing feature involves storing these documents in our database, which can require a lot of storage space. Considering this, we chose to use a MongoDB managed cluster hosted in the cloud. MongoDB atlas allows us to augment the capacity of our database by paying for a higher tier if required.

Legal and Ethical issues

1. If tutors want to sign up for our service, we will need to do a background check to verify their qualifications and history.

2. User data should be protected. The users of the web application should know who has access to their information and how the data is used. The users have rights to know if their information has been transferred to a third party.

3. Offensive content and inappropriate language is not permitted and will be considered as illegal, especially because the majority of the web app users will be children.

4. As mentioned in the list of risks, copyright is another legal issue. Copying content from other tutoring companies that already exist is prohibited.

Storage and Scalability of Data

1. All data (information about users, tutors, and courses) and links of images used for our app are stored in the database (MongoDb Atlas) in the form of links to the image hosting service imgur for now. Imgur (https://imgur.com/) has the ability to store images privately and only accessible via a private link. However, when we are going forward throughout this project, we may want to store on an external web services like Amazon S3, since databases are typically designed to store smaller data objects and size can become a problem.

2. Our app is deployed on Google Cloud using the Google App Engine. The App Engine can automatically scale depending on the application traffic and consumes resources only when the code is running. This will ensure the scalability and modelability of our deployed app.

Velocity

***Release 1 Total*:** 19 stories + 4 SubTasks, 145 points over 8 weeks
Release 2 Total: 24 stories + 4 SubTasks, 223 points over 11 week (Release 5 was over 4 weeks due to holidays)

Velocity: Approx. 46 points/ sprint (2 weeks)

Burndown Chart Sprint 1
Burndown Chart Sprint 2
Burndown Chart Sprint 3
Burndown Chart Sprint 4
Burndown Chart Sprint 5
Burndown Chart Sprint 6
Burndown Chart Sprint 7
Burndown Chart Sprint 8
Burndown Chart Sprint 9

Mockups

The mockups for sprint 1 and 2 can be found here. A separate issue was created to put the mockups. Later, we thought that adding the mockups as a comment to the feature's issues

would be more efficient. Therefore, mockups for sprint 3 and 4 can be found as comments in the respective feature's issue.

Iteration 1 (4 stories, 24 points)

Summary: For Sprint 1, we wanted to setup the development and testing environment to start coding. We also implemented the sign up/login for users and a basic search page that students can use to look up for tutors by name.

User story 1 - Account creation (8 points)
As a user, I want to be able to create an account and log in. 1- Account creation: - Username - Password - Email - Level of education - Program 2-Login - Email - Password

User story 2 - Search for Tutor (8 points)
As a user, I want to be able to search for an available tutor. Tasks: - Create a welcome page with a search button to redirect to search page - Create search page with space to display search results - Implement search function that changes display dynamically with content queried from the database

Developer story 1 - Environment setup (5)
As a developer, I want to have a functional environment to run the application. Tasks: - Install react locally - Set up the MongoDB database

Developer story 2 - Continuous integration (3)

As a developer, I want to be able to automate the process for integration and deployment of the application.
Tasks:

- Integrate travis in our GitHub repo
- Setting up jasmine testing

Iteration 2 (4 stories + 1 subtask, 29 points)

Summary: For Sprint 2, we wanted to setup docker, and Heroku and fix the unit tests. We also implemented a basic user profile page and an enhanced search feature. Additionally, we refactored sign up/login code (added sessions, encryption, fields validation) and organized better the files/folders of our project by creating components, deleting duplicate styling code, etc.

User Story 3 - Profile page for users (8 points)
As a user, I want to be able to view my Profile page as well as having a menu with other pages available to select by the student such as the View Courses Page as well as the Search for Tutors Page. These pages are contained in a side drawer menu that the student can open if they wish to navigate to another page. Tasks: - Create a profile main page where the user info is stored - Create other pages to view info on assignments, grades, payments, courses, etc... - Create a navigation side drawer that can be opened and closed as needed in order to allow the student to view a page in fullscreen or allow the user to navigate to another page.

User Story 4 - Enhanced Search for tutors (8 points)

As a user, I want to be able to search based on course name, tutor, courses, etc
 	
- Create a dropdown menu that will select the criterion of search
- Display the results based on the search criterion
- Allow users to search by:
  - Name (Already implemented in Iteration 1 - Issue #3)
  - School
  - Course/Subject
  - Program

Developer Story 3 - Setting Up Hosting with Docker and Heroku (5 points)
As a developer, I want to be able to automate the process for integration and deployment of the application. I would also like a way to be able to test my application via a containerized application. - Setting Up Hosting with Docker - Setting up Heroku - Setting up Deployment with Heroku - Setting Up Heroku website as well as server side database. - Creation of Unit tests using a customized Tutify Docker container with Jest.

Developer Story 4 - Refactoring the code (3 points)
As a developer, I would like to refactor the code in order to facilitate eventual maintenance and development. - Make an AppBar Component - Place duplicate/common styles into one file

U1 Subtask Refactoring login page and Sign Up Page (5 points)

As a developer, I would like to refactor the login and sign up page
 	
- For Login Page
  • Being able to save sessions for each user
  • Creating a smooth login experience with concurrency amongst the navigation bars
  • Fixing Encryption and decryption
- For Sign Up Page
  • Fixing Validation with empty fields
  • Redirecting to appropriate pages upon Sign Up (if all the fields are not filled, redirect back to Sign Up else redirect to Search Page)

Iteration 3 (6 stories + 2 subtasks , 47 points)

Summary: For Sprint 3, we wanted to finish setting up docker and fix the unit testing. We also wanted our website to display different user profile page with the appropriate drawer menu, depending on whether the user is logging in as a student or a tutor. Finally, we also fixed bugs in sign up/login and implemented the ability for students to choose a tutor (student assignment to a tutor).

User story 5 - Student Assignment to a Tutor (13 points)
As a user, I want to be able to register to a specific tutor. - Change db model - user inheritance (student & tutors) - Change tutor model to have list of students - Change student model - have a list of tutors - Student can access public tutor profile page - Add tutor subscription button & implement logic in db

User Story 6 - Tutor Profile Page/ Login (5 points)

As a tutor, I would like to have a personalized profile page where there are specific options curated for uploading files for students and manage assignments.
 	
- Change drawer menu options for a tutor/user based on login.
- Tutor has a profile page dashboard:
  - List of students
  - Courses taught (as well as upload documents button)
- Restricting access to for logged in tutors [Session Checking (Application only displays certain types of information to specific users)]

User Story 7 - Session Fixing/ Schema Fixing (8 points)
As a User, I would primarily like to automatically view my Profile Page Information Right after Login without needing to reload my page to load the information. I would also like to fix any issues that might have occurred when the database schema was changed. - Restricting access (logged in or out), check if user is a student or tutor - Tutor can login (changing db model, access restrictions)

User Story 7 SubTask - Fixing Bug with New Database Schema Task (5 points)

As a User and a tutor, I would like to make sure that my information is saved in the appropriate place when using my application.

- Fixing issues with linking to the appropriate database.
- Adding connections to specific databases.

US 8 - Password Encryption System For Students and Tutors (3 points)
As a User, I would like to safely enter my password when registering and ensure that my password will be stored in a safe manner that prevents hackers from getting access to the password. - Save passwords that users sign up with as hashes in the database - Save passwords for tutor login information as hashes in the database - Allow users to sign login with their original (non-hashed) passwords - Allow tutors to sign login with their original (non-hashed) passwords

SubTask - Bug: Sign up page does not verify email availabilities (5 points)
Current sign up checks the syntax of the email, but not if there is already an account associated with it. That is a bug. This bug is mitigated in this subtask. - Setting up MongoDb Stitch Environment - Using MongoDB querying with Stitch

DS5 - Fixing Testing Environment incorporated with Docker and Heroku (8 points)
As a Developer, I would like to have a containerized application setup with Docker in order to be able to create tests for the features of our application. - Docker Containerization - Creation of Unit Tests using Jest. - Integration of Docker with Jest Unit tests with Docker. - Creating MongoDb Stitch Queries for Deployment on Heroku

---

Iteration 4 (Release 1)

Release 1 aka Iteration 4, (**5 stories + 1 subtask, 45 points**)

Summary: For Sprint 4, we wanted to have a working product with basic core features. We wanted to make sure that the UI is consistent throughout the website and the features'

functionality were tested. We implemented a basic welcome/dashboard page with the list of their students/tutors. Students and Tutors are able to view the courses they are enrolled and the courses that they teach. Additionally, we implemented the ability to edit student/tutor info in their profile page and to enroll in a specific course after selecting a tutor to connect with.

US 9 - Users can view a list of their tutors / students (8 points)
As a user, I want to be able to see a list of my tutors. Tasks: - For students, add a list of their tutors in the nav drawer - For tutors, add a list of their students on their profile page - Make sure to check if logged in or not - Make sure to display content according to who is logged in.

US 10 - Users have a welcome / dashboard page (8 points)
As a user, I want to be able to see a welcome page. Tasks: - Display customized dashboard page for students - Refactor the nav drawer according to content of dashboard page Possible content of dashboard page: - To-do list - Tutor announcements - Open dashboard

US 11 - Tutor Profile Updates (8 points)
As a tutor, I would like to be able to update my personal profile that would be visible to my students. I would like to update the courses that I teach as well as my personal profile and overall description as a tutor. Tasks: - Allow tutor to Select new subjects for their profile - Allow tutors to change their overall information

Subtask - Front-End Refactoring of Profile Dashboard (8 points)

As a user, I would like to have a good UI design to have an appropriate view of everything.

Tasks:
- Making the UI look prettier for the Entire Profile Page in preparation for Release

US 13 - Course List Viewer (8 points)

As a tutor, I would like to view the list of Courses that I am offering to teach.
As a student, I would like to view a list of Courses that I am taking.

Tasks:
- Update course list page for the student
- Update course list page in profile page for tutor

DS5 - Fixing Testing Environment incorporated with Docker (5 points)

As a Developer, I would like to be able to run tests automatically for the application.

- Continuation of Unit Testing
- Making sure core features are tested

Iteration 5, **(6 stories, 50 points)**

Summary: For Sprint 5, we wanted to implement 4 main features, which are the following: enhanced to-do list, an integrated calendar, ability to view files and documents shared by the tutor, and to receive documents/files uploaded by tutors upload files to student's profile. We will be fixing these features and complete them for next iteration. We also decided to fix the deployment with containerization using Docker and finally restructure the models in our database.

US 14 - Tutor upload files to student profile and to a specific student (8 points)

As a user, I want to be able to receive documents/files uploaded by tutors upload files to my profile.

US 16 - Specific Course Page Creation / Fixing Connect Tutor Button (8 points)

As a Student user, I would like to have a specific courses view page that I am redirected to after clicking on the course I am registered with that I would like to view.

This page contains information such as :

- course documents (view only)

- Tutor Name and contact information (email address)

As a Tutor user, I would like to have a specific courses view page that I am redirected to after clicking on the course I am registered with that I would like to view.

This page contains information such as :

- course documents

- upload documents

- Student list view for each course

US-17 - Enhanced To-Do List (13 points)

As a User, I would like to have a to do list and save this list so the to do list would appear ever after I logout from my profile.

I would also like to view the sent notifications from the tutor's to do list for each of the courses I am signed up for.

As a Tutor, I would like to have a to do list for my students to remind them to do certain exercises and worksheets for specific classes.

I would also like to send notifications to students as well about these reminders.

US 18 - Calendar Integration Setup (8 points)

As a Student User, I would like to view my appointments with tutors.

As a Tutor, I would like to plan my events and add my availabilities to the calendar as well as appointments with my students.

DS 6 - Fixing deployment method with containerization with Docker

(5 points)
As a developer, I want to fix the deployment with containerization using Docker.

DS 7 - Restructuring models / connecting to Tutor First before enrolment, fixing bugs on tutor list (8 points)
As a developer, I would like to easily be able to access and view my models in the database. I have also taken one of the stakeholder's suggestions into consideration and will now only allow the students to enrol in courses taught by tutors only after they are connected to the tutors. I also want to fix an issue with sessions on the tutor list for students.

Iteration 6, **(6 stories, 58 points)**

Summary: For Sprint 6, we wanted to complete the features that we started at sprint 5: enhanced to-do list, an integrated calendar, ability to view files and documents shared by the tutor, and to receive documents/files uploaded by tutors upload files to student's profile. We also decided to write unit tests for our features.

US 18 - Calendar Integration Setup (8 points)
As a Tutor, I would like to view, add and delete events on my personal calendar.

DS8 - Unit Testing Features (13 points)
As a developer, I would like to make sure that the app is functional, that all our features are tested. Unit tests for the following features: - Announcement Page for Tutors - Student Dashboard Page Containing To Do List and Notifications - Encryption Test that tests the encryption system used for password storing - List of Students Page for Tutors - Login Page Test - My Courses Page Test for Students

- My Courses Page Test for Tutors
- Profile Page Tests for Tutors, Add Courses Feature
- Profile Page Tests for Tutors, Update Profile Feature
- Profile Page Tests for Students, Update Profile Feature
- Search Page Tests for Tutors
- Sign Up Page Tests

US 14 - Tutor upload files to student profile and to a specific student (8 points)
As a user, I want to be able to receive documents/files uploaded by tutors to my profile.

US 15- Tutor sharing files to groups of students enrolled in specific courses. (8 points)
As a user, I want to be able to view files and documents shared by the tutor for all students enrolled in specific courses.

US-17 - Enhanced To-Do List and Announcements (13 points)
Student User: ● As a user, I want my to be able to save my todo list so it can be displayed and updated in my dashboard page. Tutor User: As a user, I want to be able to send notifications to my students about announcements. ● to all students currently enrolled in a course about announcements for the course. ● to specific students about an announcement specific to them.

Iteration 7, **(5 stories + 2 Subtasks, 50 points)**

Summary: For Sprint 7, we wanted to make some adjustments to the calendar and upload documents feature. We also wanted to start working on a new feature which is adding a resource page to learn about different subjects. For the deployment, we want to complete creating a containerization for our entire application using Docker and deploy our app on Google Cloud Platform (GCP). Finally, we also decided to continue writing unit tests for our features and start planning for a WOW feature that we will start implementing in sprint 8.

US 14 - Tutor upload files to general public (8 points)
As a tutor, I want to be able to upload documents/files to everyone

Risk: Medium
Priority: High
Points: 8

US 15- Tutor sharing files to groups of students enrolled in specific courses and / or to a specific student (8 points)

As a user, I want to be able to view files and documents shared by the tutor for all students enrolled in specific courses as well as student profile and/ or to a specific student.

Ui Additions:

- Add a checkbox and button in order for tutors to filter who can see the document they shared.
- Adding a Page for students for them to view their course documents uploaded by tutors.

Risk: Medium
Priority: High
Points: 8

US 16 - Student Resource Page(8 points)

As a user, I would like to have access to a resource page.

Tasks:

- Make mockups
- Create link to main resource page accessible from navbar
- Create a page where students can select what level of education they want, what subject, etc
- Create routes for every resource page based on education level
- Separate resources by education level
- Create reusable component for resources
- Create navigation buttons that dynamically changes the page content based on what user selects

  (All, learning, studying, etc... see mockup)

- Add content for resources to show functionalities

Some ideas to be implemented later:

- Search resources by keyword
- Add web scraper that would scrape the web for links related to keyword (possibly, only links that are academic resources, peer reviewed articles, etc)
- Tutors could add resources they find interesting such as links, videos, etc.

8 points
Risk: Medium
Priority: High

DS 6 - Fixing deployment method with containerization with Docker (8 points)
As a developer, I want to containerize our application using Docker and deploy it on google cloud platform (GCP). ● Creating a containerization for our entire application using Docker ● Deploying our application on Google Cloud Platform (GCP)

DS9 - Unit Testing of new Features (5)
As a developer, I would like to make sure that the app is functional, that all our features are tested. Continuous unit testing throughout all of the new created features. Unit tests for the following features: ● Calendar page for Tutors ● Upload Documents page for Tutors

SubTask: Bug Fixes and Ui Fixes for Upload Document(8 points)
SubTask of US 14 - Tutor upload files to general public ● Fixing the error with US-14 where the uploaded file can only be viewed in an encrypted format. ● Adding a better UI for the view documents page that does not get cut off and where you need to scroll. ● Adding the last uploaded on the upload documents page (use date of uploaded file to show it). 8 Points Priority : High Risk: Low

Subtask: Calendar Students View Events for Them (8 points)
Subtask of #102 As a student, I would like to view my future event bookings with my tutors. ● Students can only view events, not add. ● Database restricting audiences to who can view events for tutors/ private events. 8 Points Priority : Medium Risk: High

---

Release 2, Iteration 8, **(7 stories + 2 subtasks, 65 points)**

Summary: For Sprint 8, we added deletion to notifications feature. We wanted to fix, clean and refactor code for upload documents feature. We also refactored the code for profile feature, by cleaning the files and combining components to include both tutor and student. We made some adjustments for tutor profile to add the ability to change the bio on profile page and choose/select the courses to be taught. We also added new resource pages to learn about different subjects. For the deployment, we were able to complete the deployment of our application on Google Cloud Platform (GCP). Finally, we wrote new unit tests for our features and continued planning for WOW feature that we will start implementing in the next sprints.

US 19- Delete Notification (5 points)
As a user, I want to be able to delete notifications ● Create delete button ● Function to delete notification ● Implement send notification to "all" option Points: 5 risk: Medium priority: High

US 20- Tutor adding and selecting courses (8 points)
As a tutor, I want to have a page where I can see all the existing courses and select the ones that I teach, and have the option to create a new one if it doesn't exist already.

Points: 8
Risk: Medium
Priority: High

US 21- Tutor can update their bio/description (5 points)

As a tutor, I want to be able to modify my bio in my profile page.

Points: 5
Risk: Low
Priority: Medium

Subtask: Share Document Feature Fixes (13 points)

Subtask of #36

Tasks:

- Making checkbox work
- Allowing Many students to be selected in order to send documents individually to many students
- Fixing issue with routing, don't show /:file when selecting My Students
- Adding a column in /doclist that displays the class the document is shared to, if it has not been shared to a class
- Fixing the date uploaded in /doclist that displays date of uploaded document. Pleas format the date correctly so we could see it in a clean format with date and time
- Adding the actual tutor name instead of tutor id in the Column
- Query by email not by first and last name for the share to classes and individual students

Priority: High
Risk: Medium
Points: 13

Subtask: Delete uploaded document, tutor view individual files to students, view latest file (13 points)

SubTask of #37

- As a tutor, I would like to be able to delete the documents uploaded for one student
- As a tutor, I would like to be able to delete the documents uploaded for many students
- Add reload function

- As a tutor, I would like to see the uploaded documents for my individual students
- As both a tutor and a student, I would want to have a cleaner view with button padding for documents view
- As a tutor, I would like to see my latest doc on upload document page
- Clean code, ex: delete unused code
- Fixing issues in user_controller with Object Keys function. Use req.session instead.
- UI fixes (Redirect the upload doc button on profile page to select course page, change it to one button

Priority: High
Risk: Medium
Points: 13

DS 10: Unit testing of remaining feature (5 points)
As a developer, I would like to make sure that the app is functional, that all our features are tested. Continuous unit testing throughout all of the new created features. Unit tests for the following features: • Calendar page for Tutors • Upload Doc • Delete Notifications • Tutor selecting/adding courses to teach Points: 5 Priority: high Risk: medium

DS 11: Course Pages/ Uploaded Document Refactoring Clean-up (3 points)
As a developer, I would like to refactor the code in order to facilitate eventual maintenance and development. Tasks: • Make Components • Place duplicate/common styles into one file • Delete unused piece of codes Points: 3 Risk : Low

Priority : High

DS 12: Profile Page Refactoring (5 points)
As a developer, I would like to refactor the profile page code in order to facilitate eventual maintenance and development. Tasks: ● Combine userInfo component to include both tutor and student ● Combined tutor and student profile page to make one ● Place duplicate/common styles into one file ● Delete unused piece of codes Risk : Low Priority : High Points: 5

DS 6 - Fixing deployment method with containerization (8 points)
As a developer, I want to containerize our application and deploy it on google cloud platform (GCP). ● Creating a containerization for our entire application using Dockerfiles ● Deploying our application on Google Cloud Platform (GCP) Priority: Medium Risk: Medium Points: 8

Iteration 9, **(5 stories + 1 subtasks, 40 points)**

Summary: For Sprint 9, we added deletion to courses added by the tutor feature, so a tutor can decide to not teach a course anymore. We added a student submission feature so that a student is able to send documents to tutors. We added the ability for tutors to add new resources to the public resource page when they wants. Finally, we implemented an error page to which the user will be redirected when accessing pages that do not exist, and the user is now redirected to the login page if they try to access a page without being logged in. There was also cleanup and refactoring of our existing frontend components to improve the reusability and readability of our application code.

DS 13 - Redirect to login or error page (5 points)

As a developer, I want to add an error page to which the user will be redirected whenever they access pages that do not exist, and the login page if the page exists but the user is not logged in.

Tasks:

- Add PrivateRoute component to redirect to login page if user is not logged in
- Add MainRoute component to redirect to /dashboard or /profile depending on the type of user logged in
- Fix redirect after logging in
- Add error page for page that do not exist
- Cleanup unnecessary checkSessions in all components
- Fix login session error when authenticating user

Points: 5
Risk: Low
Priority: High

DS 14 - Unit Testing Refactorization (3 points)

As a developer, I would like to make sure that the app is functional, that all our features are tested and ensure that each Issue is linked with the appropriate unit test.

Points: 3
Risk: Medium
Priority: Medium

US 23 - Adding new Resources (8 points)

As a student, I would like to be able to add new resources by myself to my account resource page.

Tasks:

- Add resource model, controller and routes to backend
- Add resource link to tutor sidebar
- Check if tutor is logged in in the resource levels page
- Conditional rendering of the "add a resource" component
- Create modal window opening upon clicking the add a resource component
- Create form
- Save form data to backend
- Display resources by fetching them from db instead of hardcoded
- Create drop down components for education level and category user input

- Add function to "get" the selected index from the drop down menu and show the user which option is selected
- Add default image to handle case where an image link is not provided or is unavailable
- Make it aesthetic
- Add form validation to make sure no field is empty
- clean backend from test inputs so we don't have random resources
- Reset input values upon adding resource successfully and close modal window
- Make image field not required, but let user know that it should be a valid URL, otherwise, a default image will be provided
- Acceptance tests

Points: 8
Priority: Medium
Risk: Low

US 24 - Student submitting documents (8 points)

As a student, I want to be able to upload documents and submit it to selected tutors (for assignment/homework submission purposes)

Front-End Pages Design:
- Page for Students to Submit Documents to Tutor (kept the same page)
- Add a Tab to switch view
- Added if-else statement to view tab depending if its a tutor or a student
- Add links to the tab to view pages

Back-End Process:
- Add a "sharedToTutors" Field for profile "student" collection and in the uploaded_files collection.
- Connecting Process: Add routing to the new pages added.

Risk: Low
Priority: High
Points: 8

SubTasks: #173 #172

Subtask US 20: Tutor selecting and deleting courses (8 points)

Subtask #158

As a tutor, I want to have a page where I can see all the existing courses and select the ones that I want to delete .

Task breakdown:

- frontend- Creating user interface by adding delete button inside course page
- backend- Creating a function that handles the event to delete the course from front end to backend

Points: 8
Risk: Medium
Priority: High

Plan for Iteration 10

US 22 - Upload profile Pic (8 points)
As a user, I want to be able to upload my profile pic on the profile page Tasks: • Tutors should be able to upload their profile pic • Students should be able to select an avatar as their profile picture Priority: High Risk: Low Points: 8

US 25 - Forgot Password Reset (8 points)
As a user and a tutor, I want to be able to change my password in the case where I forget it. Priority: High Risk: High Points: 8

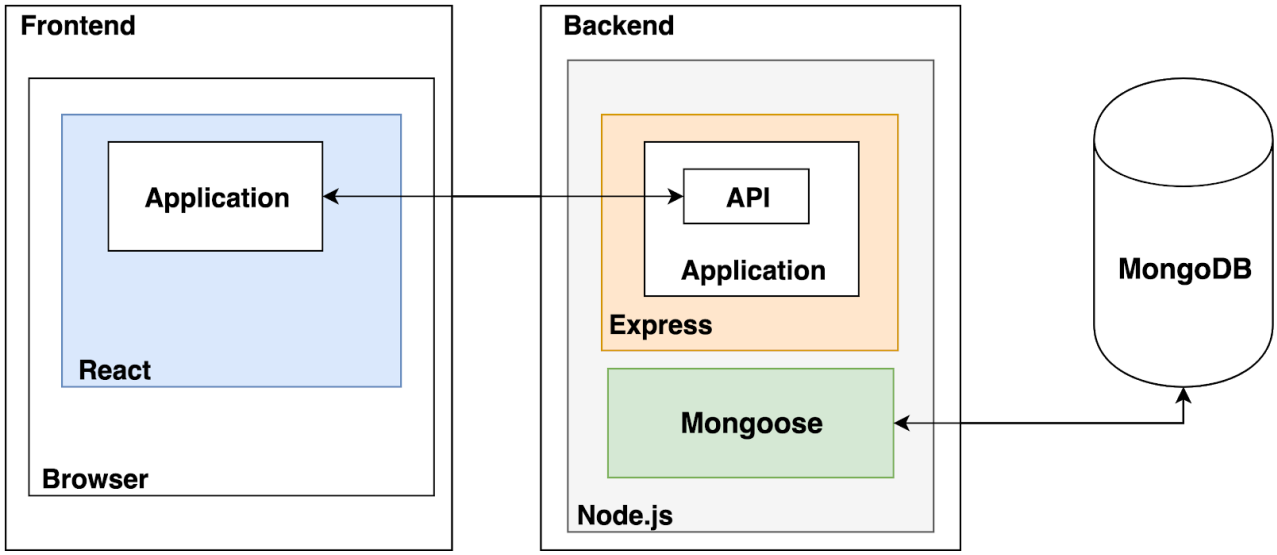
Release 3 Total: X stories, X points over X weeks
Release 3, Iteration 13, (X stories, X points)

Planning for Release 3:

For the next release, we were thinking about certain "wow" features that would be very interesting to integrate into our application. The first feature would include data analysis techniques related to AI and Machine Learning. We are thinking of **using data analytics to read a text file the tutor uploads and extract questions for students from it** In order to achieve this, we are thinking of using the Spacy Library adapted for Node.Js as well as using Natural Language Programming (NLP) techniques for text classification and analysis.
The second would be **video calling Feature for tutoring sessions**. The tutors would have the ability to save these video calling sessions and share with different classes and students. There could also be a video display page with pre recorded content from tutors. To add to this, we were also thinking of a **white board feature** that will allow tutors to write down their notes for their lessons and ability to share this directly in the Documents Page.

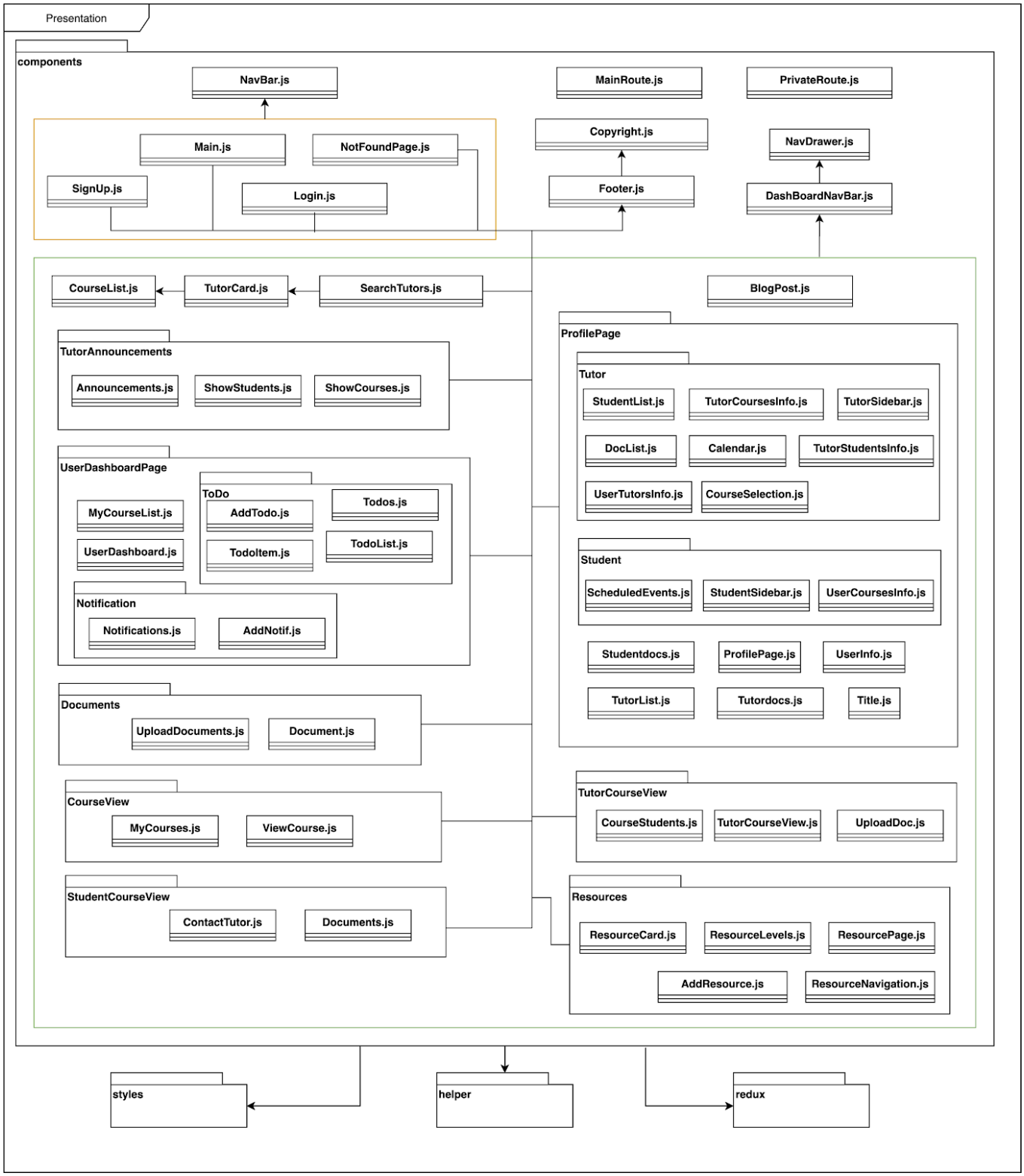
Overall Arch and Class diagram

High-level Architecture Diagram:

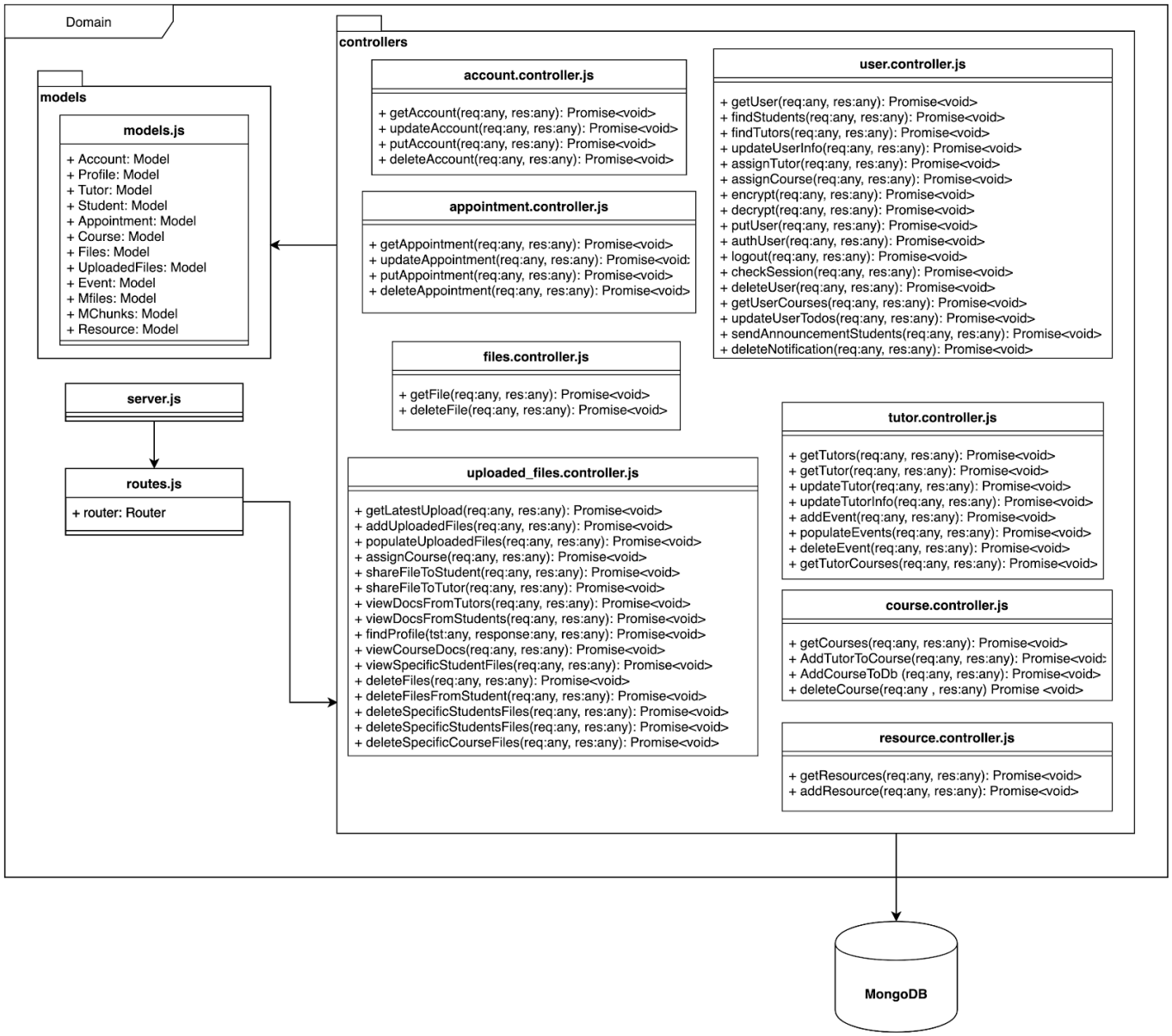


Class Diagram:

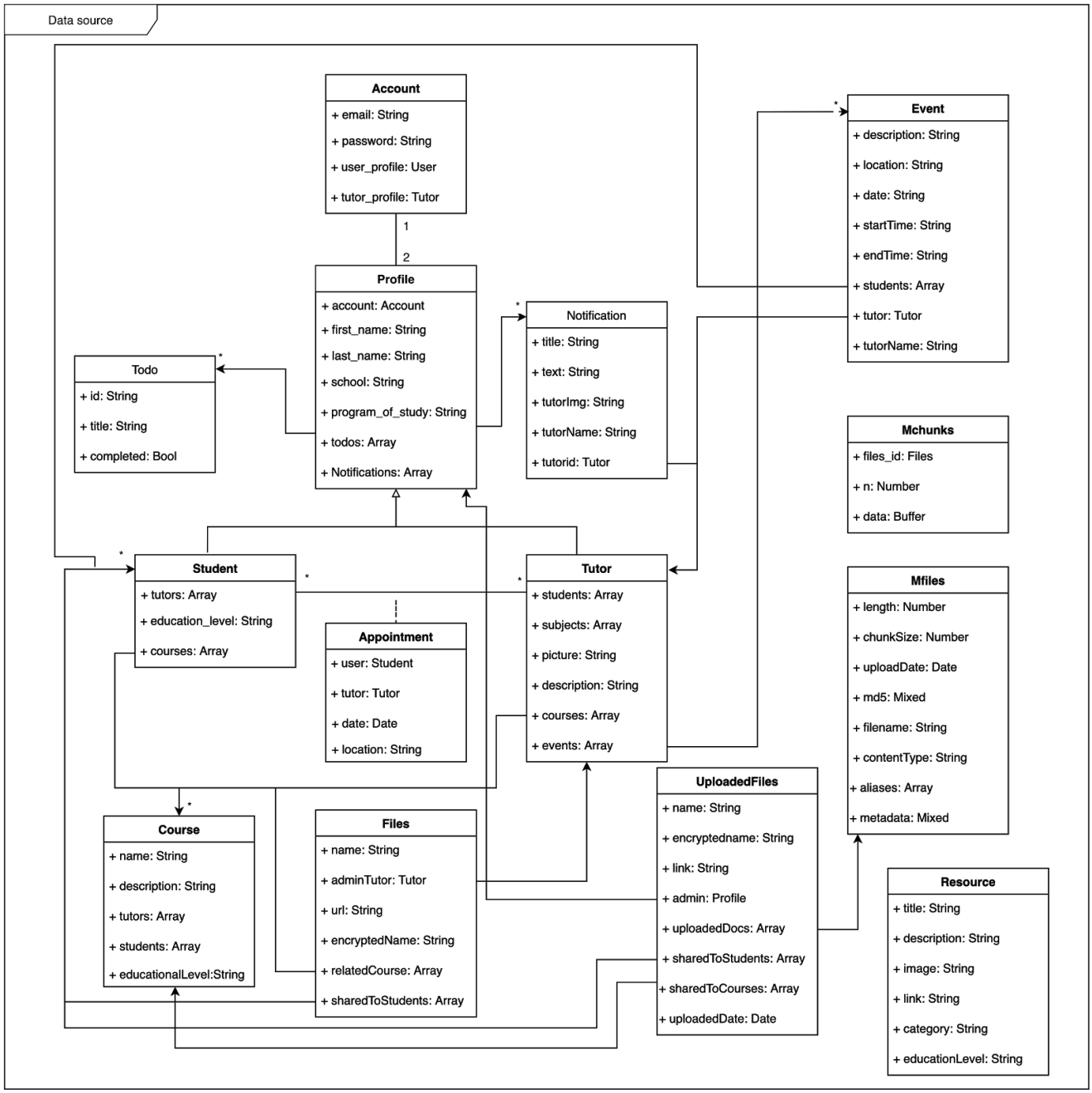
Presentation / Frontend Layer:



Domain / Backend Layer:



Data source Models:



Infrastructure

React: the JavaScript library used for building user interfaces.
Node.js: the JavaScript run-time environment that we use for our backend server.
Express.s: a minimalist web framework for Node.js that we use for creating and running our web server with Node.
Mongoose: the MongoDB object modeling tool designed to work in an asynchronous environment.
MongoDB Atlas: for our fully-managed cloud database.
Jest: Testing framework for Javascript that we used for testing purposes.

Name Conventions

We use the Google JavaScript Style Guide.

Code

File path with clickable GitHub link	Purpose
tutify/src/components/TutorProfile/DocList.js	This file corresponds to the List of Documents Page for tutors. Here, the tutor can view all of the files currently uploaded to the system. The tutors also have the ability to share these documents to courses and students as well as delete them from the system if they desire.
tutify/src/components/TutorProfile/Calendar.js	This file corresponds to the Calendar Widget Page for both students and tutors. Here, the tutor can create events as well as share them with students. They have the ability to delete events. However, students cannot create nor delete events, they can only view them.
tutify/src/components/Resources/ResourcePage.js	This file corresponds to the Resource Page where students can access different resources based on their education level. Once an education level is selected, the student is redirected to a page where resources of different categories are displayed. Once a

	resource card is clicked, a new tab opens in the browser for the student to visit.
tutify/src/components/ProfilePage/Studentdocs.js	This file corresponds to the Student Documents Page for students where they can see all of their individually shared documents from their tutors. They have the ability to download and view the documents.
tutify/src/components/ProfilePage/ProfilePage.js	This file corresponds to the profile page for both students and tutors. For tutors, the page displays their personal information, their students, the subjects they're teaching and a calendar. For students, the page displays their personal information, their tutors, the courses they're taking and a list of scheduled events.

Testing and Continuous Integration

Testing

The way that the unit tests were run was through using the Jest Testing Framework working with our JavaScript React App: https://jestjs.io/. Jest is a Framework that is used to run unit tests to verify if certain functions work appropriately with the appropriate input. Our tests use mocked data to verify if the expected value matches the actual value when a certain mock is passed to the function. We had initially used the Mocha and Chai framework to test our code but we found that jest worked best with React as it is a newer framework that requires less set up than Mocha and Chai. It also comes with an easy to use assertion library that is ready to be used out of the box. It also can be combined with Enzyme react, a framework that eases the testing process for React components.

A summary of how each unit test was written is described in the table below.

Test File path with clickable GitHub link	What is it testing?
tutify/test/announcement-tests-tutors.spec.js	The Announcements page test verifies what the content of the announcement is as well as to which audiences this announcement notification is delievered to (to specific courses or to specific students or to everyone).
tutify/test/view-course-tests-students.spec.js	The View Courses tests for students verifies if the information on the particular document(s) shared by the tutor to the classes the students are enrolled in are displayed correctly on this page.
tutify/test/listofstudents-tests-tutors-sharedoc.spec.js	The list of student page share document test verifies if all the student information is present on this page for a specific tutor's students. It basically shows a list of students to whom the document can be shared to.
tutify/test/profilepage-updateprofile-tests-tutors.spec.js	The update Profile Page test for tutor profile page verifies the update information feature of the tutor profile. It checks if all of the fields present of the tutor information are updated according to the new parameters entered.
tutify/test/calendar-tests-tutors.spec.js	The calendar test for tutors tests for the addition of events on the calendar as well as viewing the added event(s). It makes sure that all the event information (date, time, name and description of event) as well as delete button are present.

For some stories, such as connecting a tutor and course to a user, the function could not be tested in a unit test as it involves storing the data in the database, and nothing is explicitly returned to our class component. In a case like this, we ensured the quality of the feature by performing acceptance tests. The result of storing new data in the database is often displayed visually in other component(s), so we can verify the validity of our operations by looking at the results displayed. A list of our acceptance tests can be found in our GitHub wiki: https://github.com/compgirl123/TutifySoen490/wiki/Acceptance-Test

Continuous Integration

Travis Continuous Integration: https://travis-ci.com/compgirl123/TutifySoen490

The continuous integration environment that was used was Travis-Ci. Travis Ci is a continuous integration service that can analyze projects directly linked on GitHub. It can analyze different branches present on the GitHub repository and is not limited to only analyzing the master branch. Travis detects code smells such as unused variables and other variables that might break the code. Travis automatically builds and tests changes every time a new pull request is made for a particular branch. Testing and development in Travis is done in small incremental quantities of code.

We currently only have one stage to our build, which is a job that runs the build and also execute our tests using Jest.

Deployment

As for deployment, we have deployed our app manually on Google Cloud using the Google App Engine. The App Engine can automatically scale depending on the application traffic and consumes resources only when the code is running. The App Engine flexible environment also allows us to customize the runtime and operating system of the virtual machine of our application using Dockerfiles. This will ensure the scalability and modelability of our deployed app. The configuration for our application is contained in our client.yaml file.

Our deployed application can be accessed here: http://tutify-259321.appspot.com/