<Tutify> Release 1

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Claudia Feochari (40000060)	**compgirl123**	42 points
Cynthia Cherrier (40005808)	Cynthiac3	37 points
Jasmine Latendresse (40011419)	jaslatendresse	37 points
Tanya Multani (40008542)	tanyamultani	26 points
Bilal Nasir (40015010)	bilal101	26 points
Kasthurie Paramasivampillai (40025088)	kasthurie	24 points
Pierre Watine (40027675)	PWatine	42 points

Project summary

Tutify is a web-based application that revolutionizes the way tutors and teachers interact with their students by providing them a more enhanced interactive learning experience. Unfortunately, hiring private tutors can be pricey for some parents and/or hard to find for specific courses. This application offers a solution; centralizing all tutors in a single place and rewarding them for tutoring courses that are the most in demand as well as providing students who could not afford traditional tutoring a more affordable way to have access to course material. The app also supports document sharing between tutors and tutees as well as document publishing for free exercises for the entire Tutify community. 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.

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 application will be deployed and run on a Heroku platform, since it has integrated data services and a powerful ecosystem. This platform uses containers called “dynos” to run apps. Using dynos, Heroku allows to instantly scale apps to

meet demand. This will make it easier to solve problems such as increased traffic, respond to new functionalities or even to meet business scale in future. Heroku also allows us to connect to our custom MongoDb Atlas Database.

Velocity

Burndown Chart Sprint 1
Burndown Chart Sprint 2

Project Total: X stories, X points over X weeks

Iteration 1 **(4 stories, 24 points)**

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 for a course. 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 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 + U1 subtask, 29 points)**

- The mockups of features of sprint 1 and 2 can be found in GitHub

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 dashboard with other pages available to select by the student such as the View Courses Page as well as the Search for Tutors Page and the Payments 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

- Major/Program
- By Year (Engineering 3rd year)

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 Mocha and Chai.

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)

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
- Mockup for tutor public profile page

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
- Tutor has a dashboard (list of students) as well as upload documents, view courses taught
- 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)

It 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 Mocha and Chai.
- Integration of Docker with Mocha and Chai Unit tests with Docker.
- Creating MongoDb Stitch Queries for Deployment on Heroku

Plan For Iteration 4 (Release 1)

Release 1 Total: 17 stories + 3 SubTasks, 121 points over 4 weeks
Release 1 aka Iteration 4, (3 stories, 21 points)

User Story 9 - Tutor sharing files to groups of students enrolled in specific courses (8 points)

User Story 10 - Tutor upload files to student profile and to a specific student (8 points)

User Story 11 - Users can view list of their tutors / students (8 points)

User Story 12 - Users have a welcome / dashboard page (8 points)

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

Iteration 5, (X stories, X points)

Iteration 6, (X stories, X points)

Iteration 7, (X stories, X points)

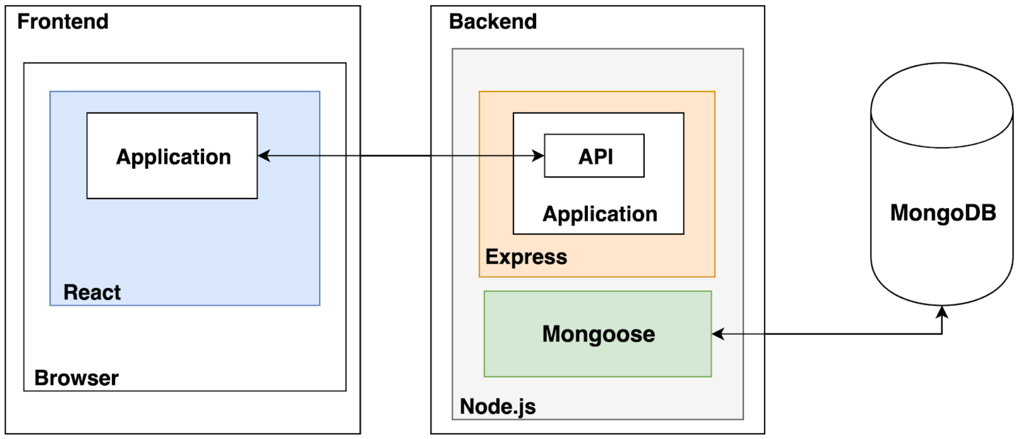
Release 2 Total: X stories, X points over X weeks
Release 2, Iteration 8, (X stories, X points)

...

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

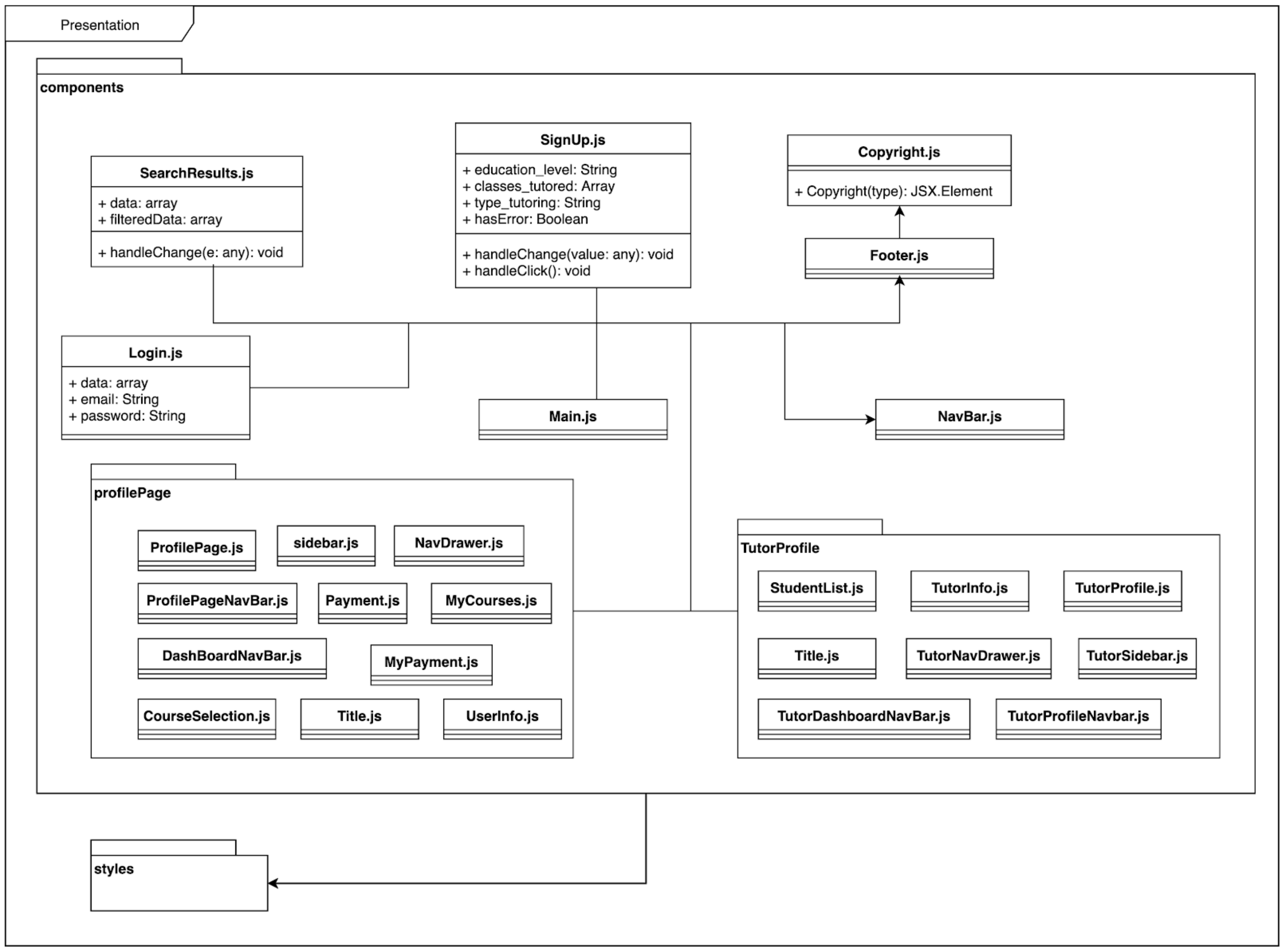
Overall Arch and Class diagram

High-level Architecture Diagram:

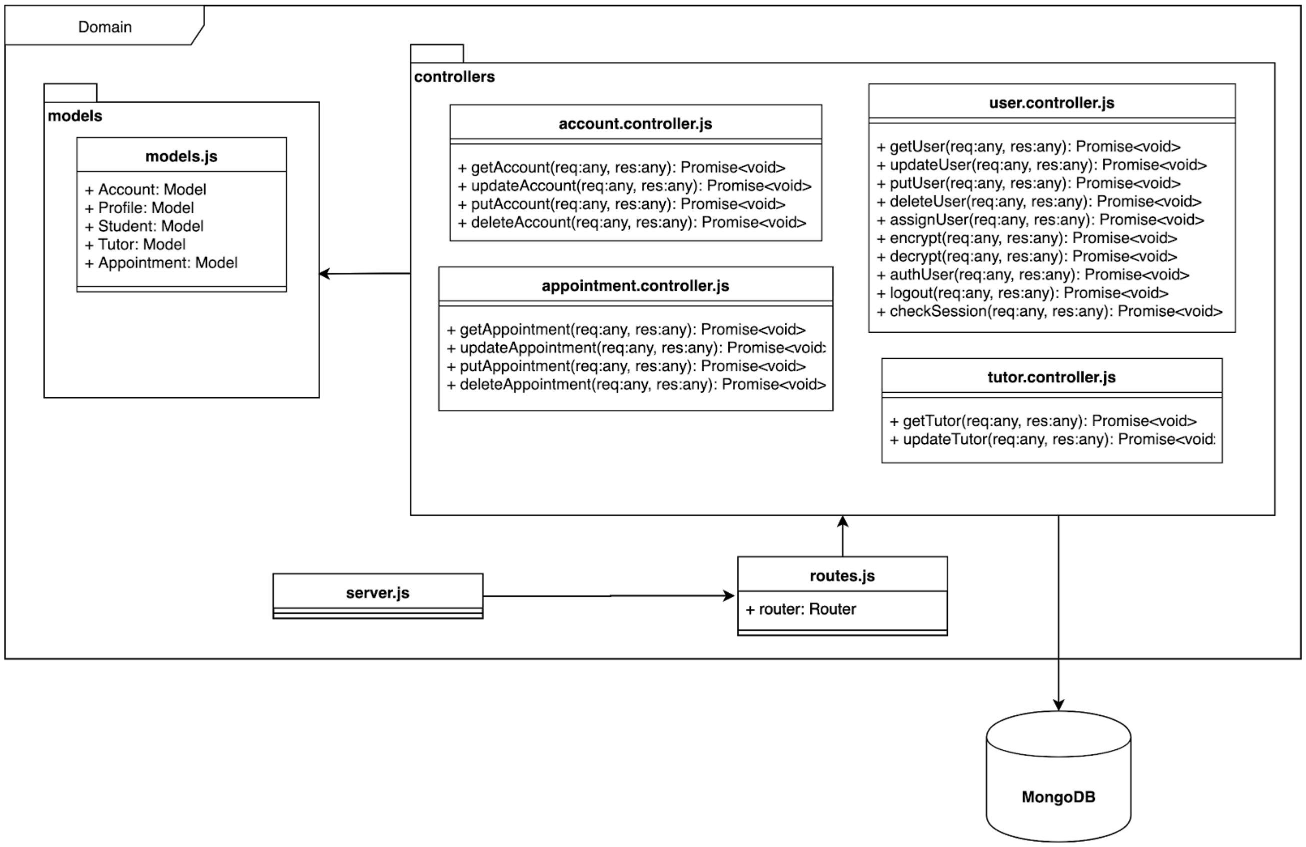


Class Diagram:

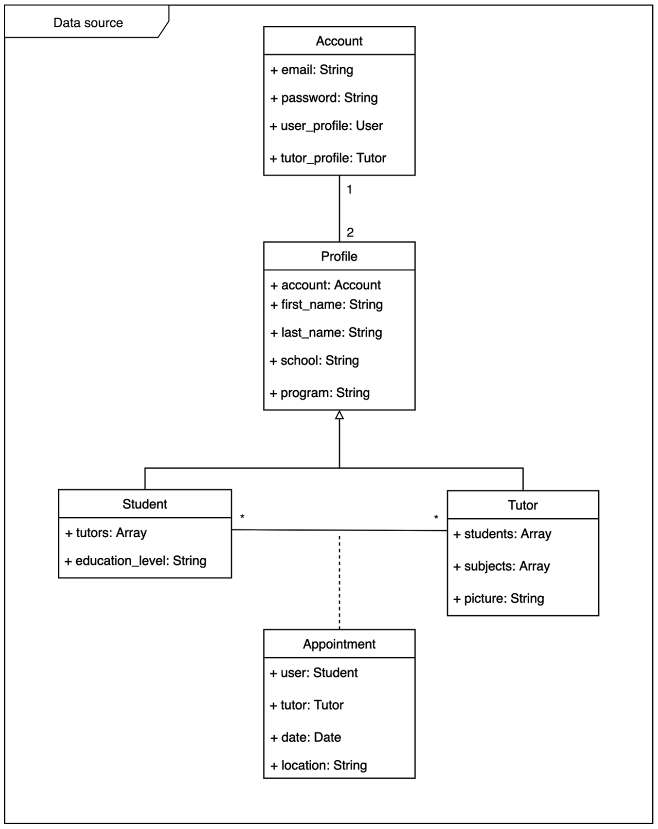
Presentation / Frontend Layer:



Domain / Backend Layer:


Domain
models
models.js
+ Account: Model
+ Profile: Model
+ Student: Model
+ Tutor: Model
+ Appointment: Model
controllers
account.controller.js
+ getAccount(req:any, res:any): Promise<void>
+ updateAccount(req:any, res:any): Promise<void>
+ putAccount(req:any, res:any): Promise<void>
+ deleteAccount(req:any, res:any): Promise<void>
appointment.controller.js
+ getAppointment(req:any, res:any): Promise<void>
+ updateAppointment(req:any, res:any): Promise<void>
+ putAppointment(req:any, res:any): Promise<void>
+ deleteAppointment(req:any, res:any): Promise<void>
user.controller.js
+ getUser(req:any, res:any): Promise<void>
+ updateUser(req:any, res:any): Promise<void>
+ putUser(req:any, res:any): Promise<void>
+ deleteUser(req:any, res:any): Promise<void>
+ assignUser(req:any, res:any): Promise<void>
+ encrypt(req:any, res:any): Promise<void>
+ decrypt(req:any, res:any): Promise<void>
+ authUser(req:any, res:any): Promise<void>
+ logout(req:any, res:any): Promise<void>
+ checkSession(req:any, res:any): Promise<void>
tutor.controller.js
+ getTutor(req:any, res:any): Promise<void>
+ updateTutor(req:any, res:any): Promise<void>
server.js
routes.js
+ router: Router
MongoDB

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.
Jasmine: Testing framework for Javascript that we used for testing purposes.

Name Conventions

List your naming conventions or just provide a link to the standard ones used online.

We will use the Google JavaScript Style Guide.

Code

File path with clickable GitHub link	Purpose (1 line description)
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/SignUp.js	This file corresponds to the Sign Up Page for students who are wishing to receive tutoring from specialized tutors teaching various school subjects and grade levels.
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/Login.js	This file corresponds to the Login Page for users (student + tutors).
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/SearchTutors.js	This file corresponds to the search page for tutors. All the tutors are fetched from our database, and the user can search by name, school, course/subject or program in the search bar which will dynamically filter the list of tutors.
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/App.js	This file contains all the routes paths of the pages of the application.
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/Main.js	This is the first page the user will see when he launches the application. It is

	the main page that contains links to other features.
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/profilePage/ProfilePage.js	This file corresponds to the Profile Page for students where they can see all of their personalized information as well as having the option to select the tutoring type the student wishes to receive as well as the classes they want to be tutored in.
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/profilePage/UserInfo.js	This file corresponds to displaying the user's information on the Profile Page (explained in the box above).
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/profilePage/MyCourses.js	This file corresponds to the Courses Page where the student can see all of the courses they are currently enrolled in with a tutor as well as have access to the course material (this feature will be implemented fully during the next iteration).
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/profilePage/Payment.js	This file corresponds to the Payments Page that will be used when the User purchases courses that require payment. It will also show a list of courses that are free (cost 0$).
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/TutorCard.js	This file contains the tutor card component which consists of the small card in the search page, and the dialog box which opens when the user click on a tutor's image. The dialog box contains the "Connect" button to link the tutor to the current user.
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/TutorProfile/TutorInfo.js	This file contains all of the information required for the tutor page for the tutor when the tutor logs into their profile.
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/TutorProfile/TutorProfileNavbar.js	This file contains information regarding the drawer sidebar for the tutors. This side bar contains options that only tutors can have access too.
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/profilePage/sidebar.js	This file contains information regarding the drawer sidebar for the students. This side bar contains options that only students can have access too.

Testing and Continuous Integration

Testing

Test File path with clickable GitHub link	What is it testing (1 line description)
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/spec/encryption-test.js	It tests the encrypt/decrypt string function.
https://github.com/compgirl123/TutifySoen490/blob/feature/ds2-signup-tests/tutify/spec/SearchPage-test.js	It tests the function that fetches all the accounts from the database. (non-functional so far).
https://github.com/compgirl123/TutifySoen490/blob/ds2-jasminetests/tutify/spec/jasmine-test.js	It tests that the users are currently present in the database as well as checks if the website is indeed up and running on localhost:3000.

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 commit is added to 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 Jasmine.

As for different services, we have set up Docker so that the app could be run through it. This would allow us to add service containers at will and assure scalability and modelability in our app. Because of it, many services can be run in parallel to the app. We followed this documentation: https://docs.docker.com/get-started/ and added this functionality in this commit https://github.com/compgirl123/TutifySoen490/commit/679e8bb16c08ba3a6130310f4458d92e4bfcc5ff.