<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**	63 points
Cynthia Cherrier (40005808)	cynthiac3	69 points
Jasmine Latendresse (40011419)	jaslatendresse	53 points
Tanya Multani (40008542)	tanyamultani	42 points
Bilal Nasir (40015010)	bilal101	42 points
Kasthurie Paramasivampillai (40025088)	kasthurie	40 points
Pierre Watine (40027675)	PWatine	63 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 and to connect with tutors that best align with their needs.

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 will eventually support 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.

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 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

***Release 1 Total*:** 19 stories + 4 SubTasks, 145 points over 8 weeks

Velocity: Approx. 36 points/ sprint

Burndown Chart Sprint 1
Burndown Chart Sprint 2
Burndown Chart Sprint 3
Burndown Chart Sprint 4

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

Plan For Iteration 5 Iteration 5, (X stories, X 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)

Developer Story 6 - Fixing deployment method with containerization with Docker (5 points)

Iteration 6, (X stories, X points)

Iteration 7, (X stories, X points)

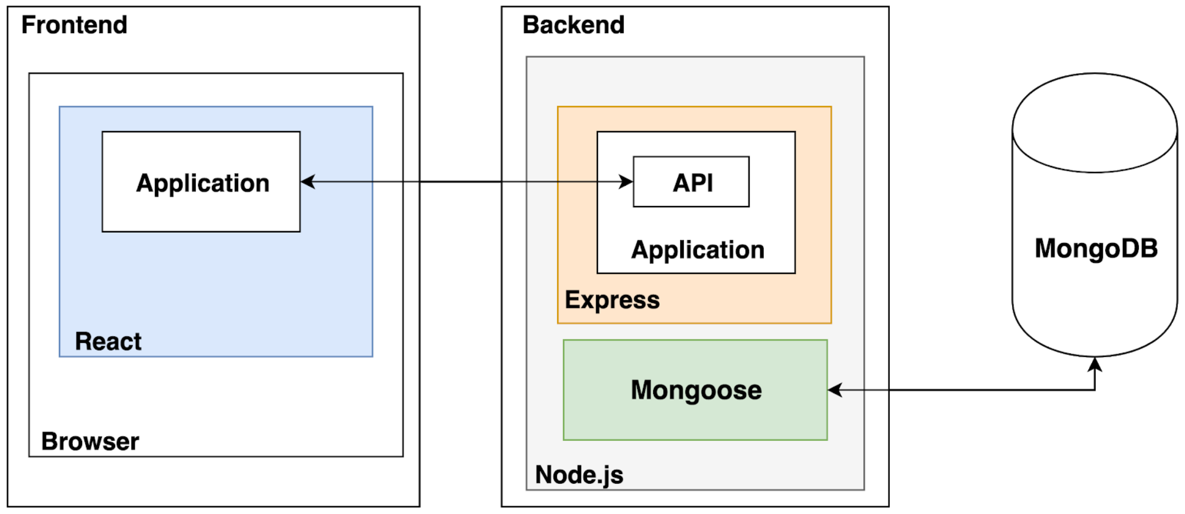
Release 2 Total: X stories, X points over X weeks
Release 2, Iteration 8, (X stories, X points)

...

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

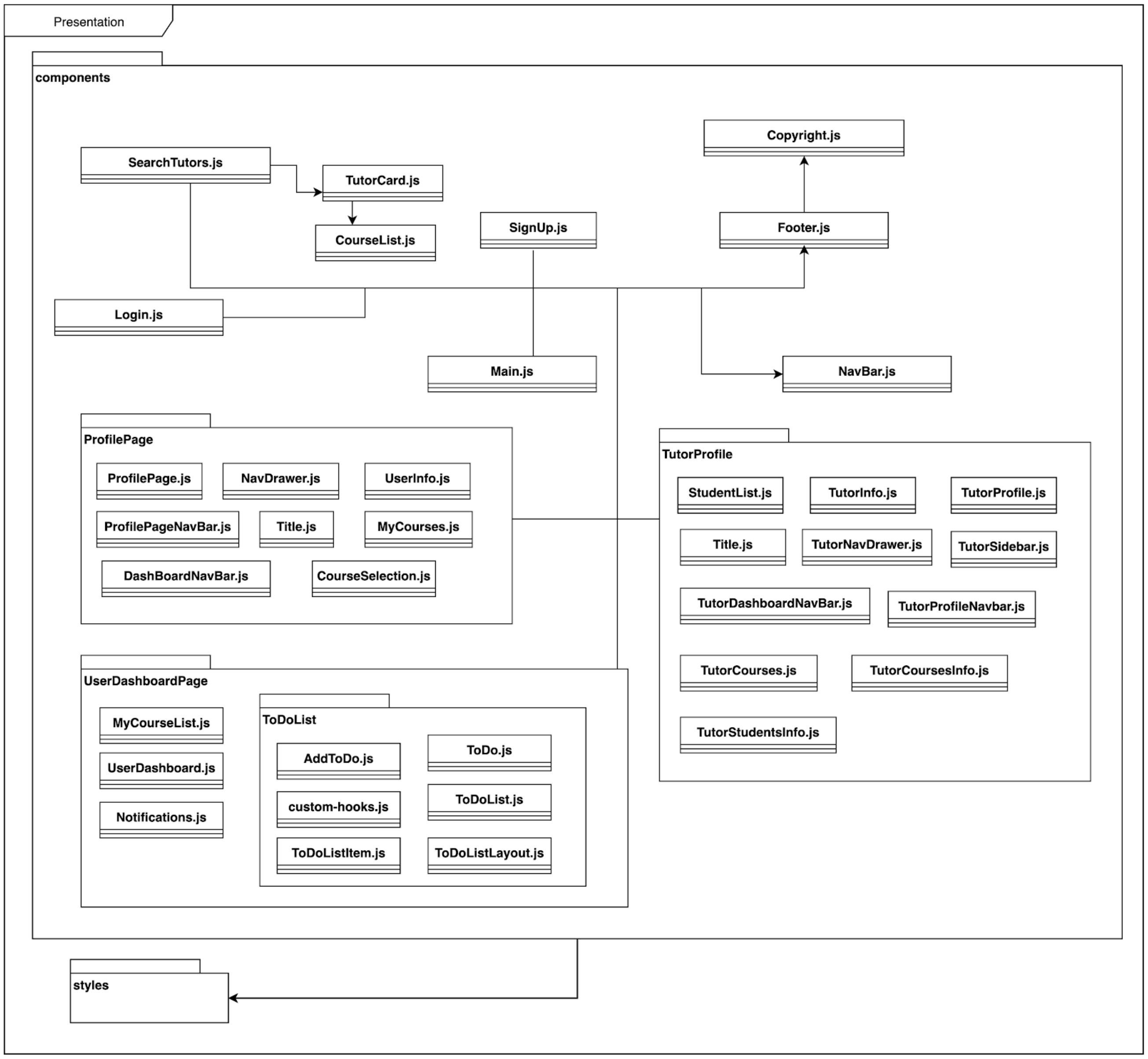
Overall Arch and Class diagram

High-level Architecture Diagram:

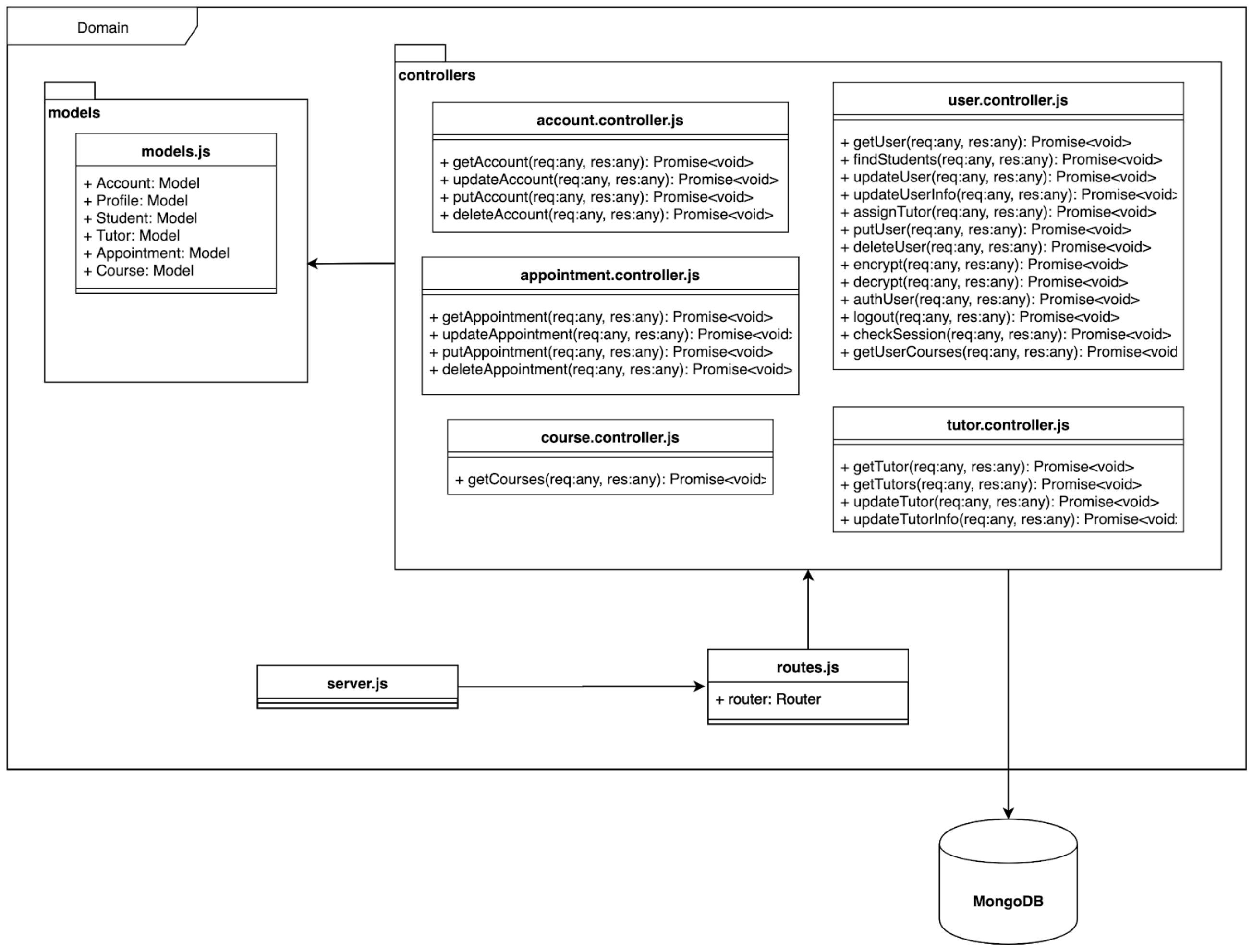


Class Diagram:

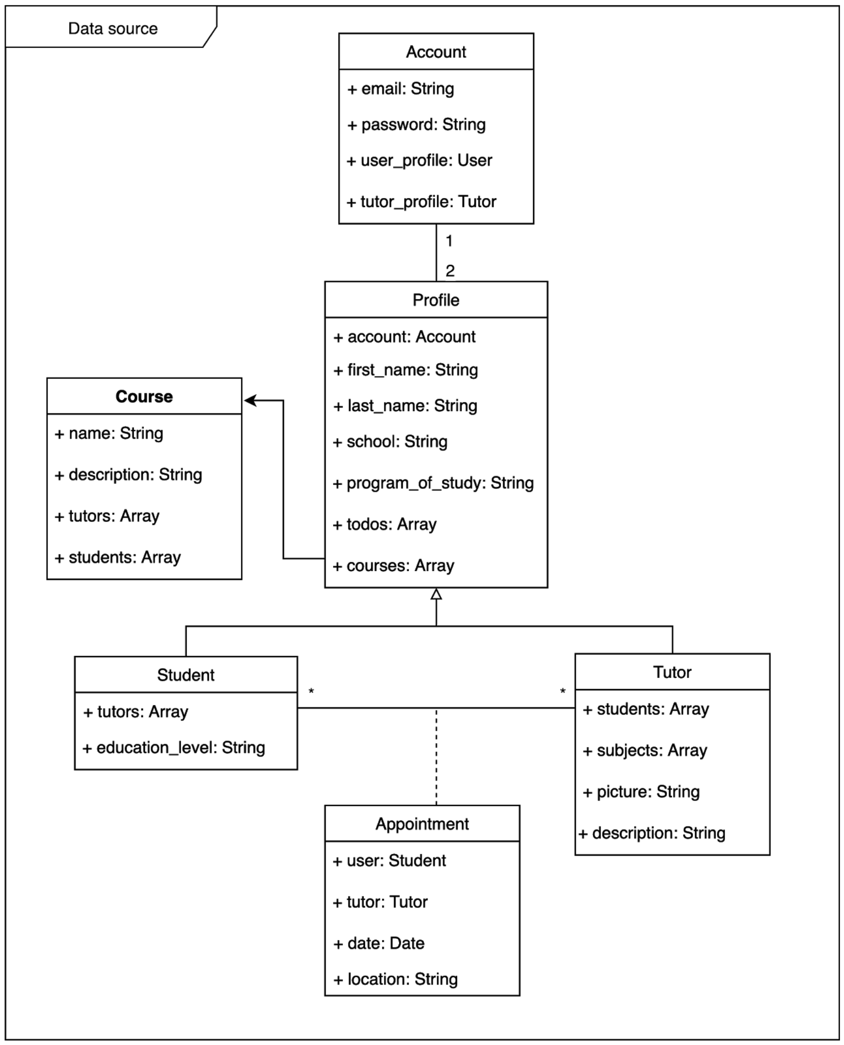
Presentation / Frontend Layer:



Domain / Backend Layer:


Domain
models
models.js
+ Account: Model
+ Profile: Model
+ Student: Model
+ Tutor: Model
+ Appointment: Model
+ Course: 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>
course.controller.js
+ getCourses(req:any, res:any): Promise<void:
user.controller.js
+ getUser(req:any, res:any): Promise<void>
+ findStudents(req:any, res:any): Promise<void>
+ updateUser(req:any, res:any): Promise<void>
+ updateUserInfo(req:any, res:any): Promise<void:
+ assignTutor(req:any, res:any): Promise<void>
+ putUser(req:any, res:any): Promise<void>
+ deleteUser(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>
+ getUserCourses(req:any, res:any): Promise<void
tutor.controller.js
+ getTutor(req:any, res:any): Promise<void>
+ getTutors(req:any, res:any): Promise<void>
+ updateTutor(req:any, res:any): Promise<void>
+ updateTutorInfo(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.
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
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/components/TutorProfile/TutorProfile.js	This file corresponds to the Profile Page for tutors where they can see all of their personalized information. It contains a drawer sidebar that has options only available to tutors (with links to other pages)
https://github.com/compgirl123/TutifySoen490/blob/master/tutify/src/components/UserDashboardPage/UserDashboard.js	This file contains the code for user dashboard page, which includes notifications, a to-do list and a list of courses that the student is currently enrolled in.

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 mockedData to verify if the expected value matches the actual value when a certain mock is run on 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/encryption-test.spec.js	It tests the encrypt/decrypt string function.
tutify/test/search-tutors-test.spec.js	It tests the filtering of tutors by the name, school, courses or programs they teach. This feature is present on the tutor search page.
tutify/test/login-test.spec.js	It tests the login function from the login page. It uses the email and password of the mocked and inputs them into the text fields. It also simulates the clicking of the Login button.

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.

As for deployment, 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/. The dockerfile for the frontend of our app can be found here: https://github.com/compgirl123/TutifySoen490/blob/master/tutify/Dockerfile.