University of Colorado Boulder

CSCI 5448 OBJECT ORIENTED ANALYSIS AND DESIGN

PROJECT REPORT

Profile Aggregator

by

Vishwanath Kulkarni
Sitesh Ranjan

OVERVIEW

Profile – Aggregator is a website built over the need for accumulating profiles of all platforms and ease of access while applying for the job. It will link the profiles of a user from various platforms like (LinkedIn, GitHub, LeetCode) and provide a one-page view of all data which can be handy while applying for jobs. This will help in providing all data in one place and also come in handy with an easy copy of each data with just one click of a button.

We are trying to achieve ease of applying for any job, by providing all the information in one place. This will not only save your time but also

Our system will accumulate all details from all platform (LinkedIn, GitHub, LeetCode) and provide a one-page view of all data which can be handy while applying for jobs. System Functionalities:

1. Personal Dashboard for each user.
2. Fetch details from each platform and allow user to get them in real time.
3. Maintain User details and allow user to modify those details.

GitHub : https://github.com/vishwakulkarni/profile-aggregator
Website : https://aggregator007.herokuapp.com

Final State of System

We have completed our project with all functionality which we proposed in earlier reports the OOAD project submission. Below are the task details completed and not completed.

Completed Tasks	Incomplete Tasks
• Login and Registration • Association of LinkedIn profile • Association of GitHub profile • Association of Leetcode profile • Easy Access to self-profile • Upload and download Resume • Insert and Update Education • Insert and Update Projects • Insert and Update Education • Delete user details • Auto fetch users GitHub repositories with username • Copy buttons on all available user data • Upload and Display of user profile • Easy sharing of the User profile with recruiters • Handling users' images and resumes • Using online free resource management Cloudinary • Database management using MongoDB • Free Hosting on Heroku • Long persistent session • Easy scaling of Application	• Login with LinkedIn (Issues was with LinkedIn) • Fetch user details from LinkedIn

Completed tasks:

We have proposed at the beginning of our coursework that we will develop a website that will help students or professionals to accumulate profiles of all platforms and help in access to data while applying for the job.

We have completed this project with a completely satisfactory result. In the process of completion, we realized that UML diagrams are a boon to project development and with knowledge of OOAD design patterns one can complete projects with ease.

While working on project, we had few issues with database and flask server, so we need to switch to NodeJS server and mongoDB database.

Previous tools	**Changed Tools**
MYSQL Database	MongoDB Database
Flask Server	NodeJS server
GCP hosting	Heroku hosting

Changes:

1. We changed from MYSQL database to mongoDB as mongoDB was easy to manage our user's data and had easy integration to the system.
2. Flask server was way slow and was giving issues with user management. Thus, changed to NodeJS server.
3. GCP hosting was providing us IP address instead of domain name, thus we went to Heroku for hosting our app for FREE!

In-Completed tasks:

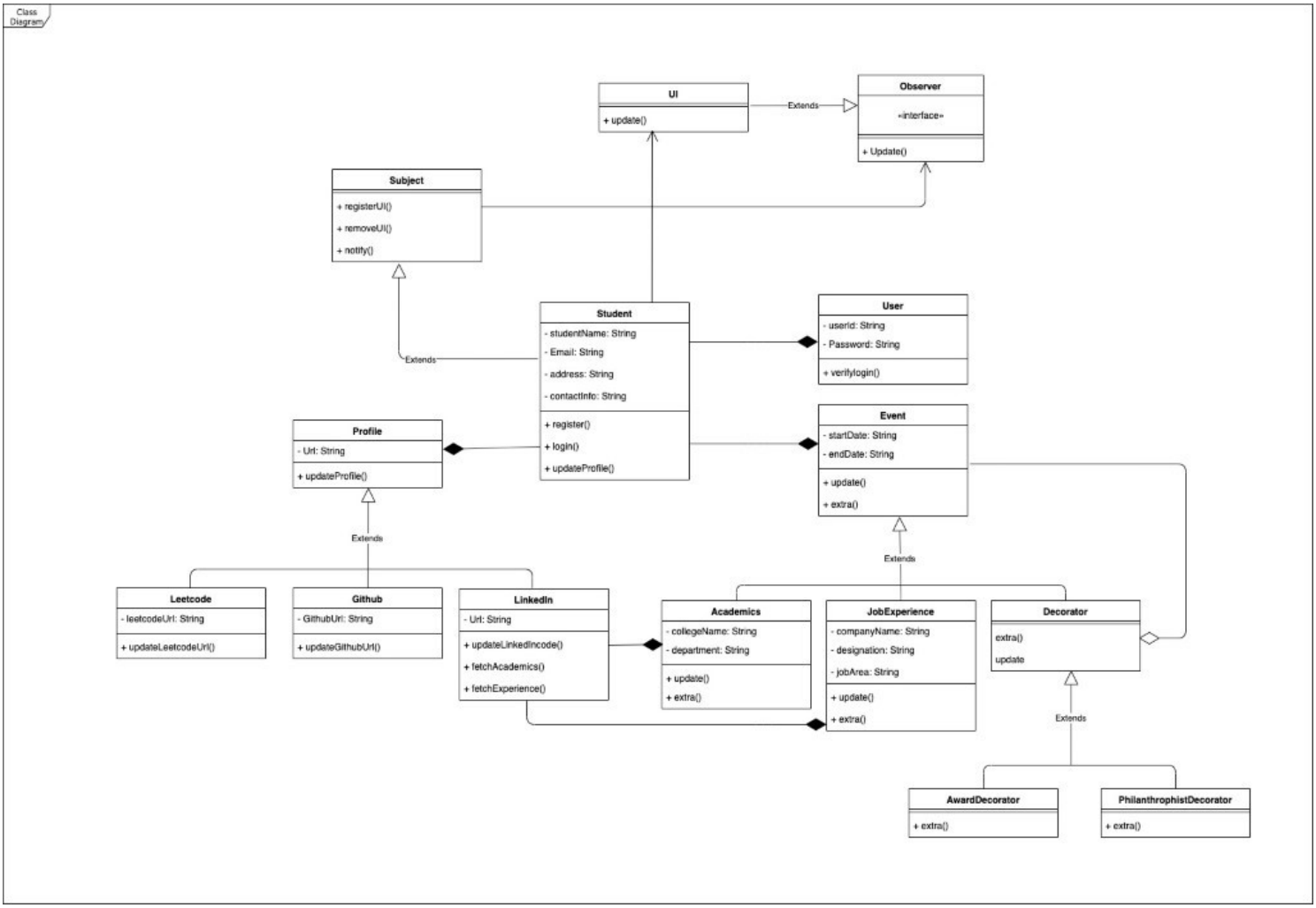
1. Login with LinkedIn (Issues was with LinkedIn)
   Allowing user to login with LinkedIn was creating issue as dual registration was happening in backend and handling both EMAIL registration and LinkedIn was creating problem with user management

2. Fetching details from LinkedIn

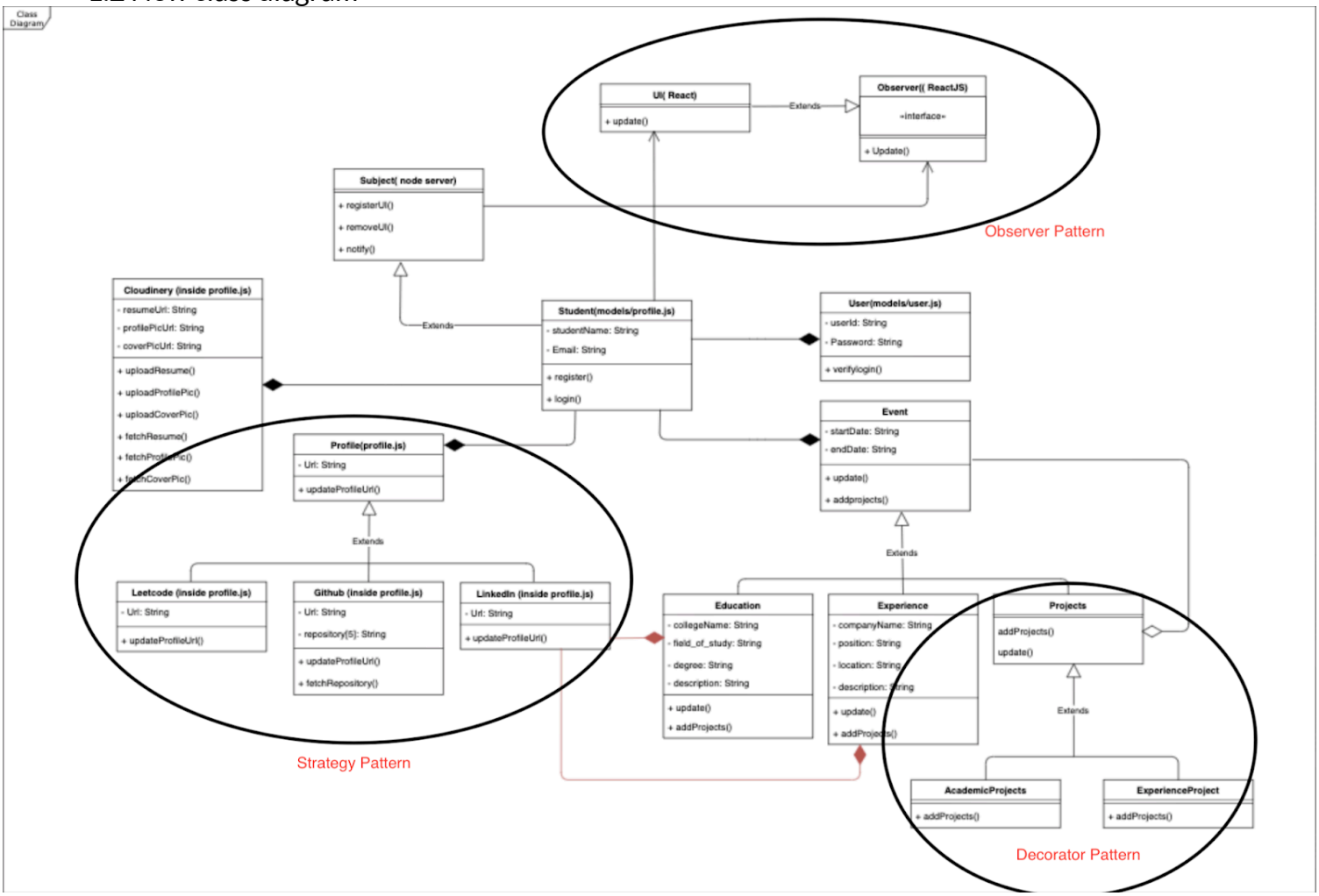
LinkedIn doesn't allow user data to be fetched (LinkedIn issue) thus could not complete it.

Final Class Diagram and Comparison statement

1.1 Previous class diagram



1.2 New class diagram



Comparison of Class diagram and pattern usage: -

As mentioned in project 4 submission, we have used three patterns in our design.

1. **Observer Pattern**
2. **Decorator Pattern**
3. **Strategy Pattern**

- We used all these 3 patterns for designing our application. However, there are few changes that got introduced in our new class diagram.

- We planned to update education and experience by fetching user data from LinkedIn. But as LinkedIn doesn't allow user data to be fetched. Therefore, we are updating education and experience through user input. This change got reflected in our new class diagram as delegation relationship between Education class and LinkedIn class breaks. Similarly, the relationship between Experience class and LinkedIn class also breaks away. This breaking of relation is depicted in red in new class diagram.

- For adding Academic Projects and Experience projects, we are using Decorator pattern. We are displaying academic projects and experience projects together in Project component. So, we decided to use decorator pattern to add projects on top of Education class and Experience class. We earlier decided to use decorator pattern for extra awards and social works. But, we found a better approach to use decorator pattern, thus removed this functionality.

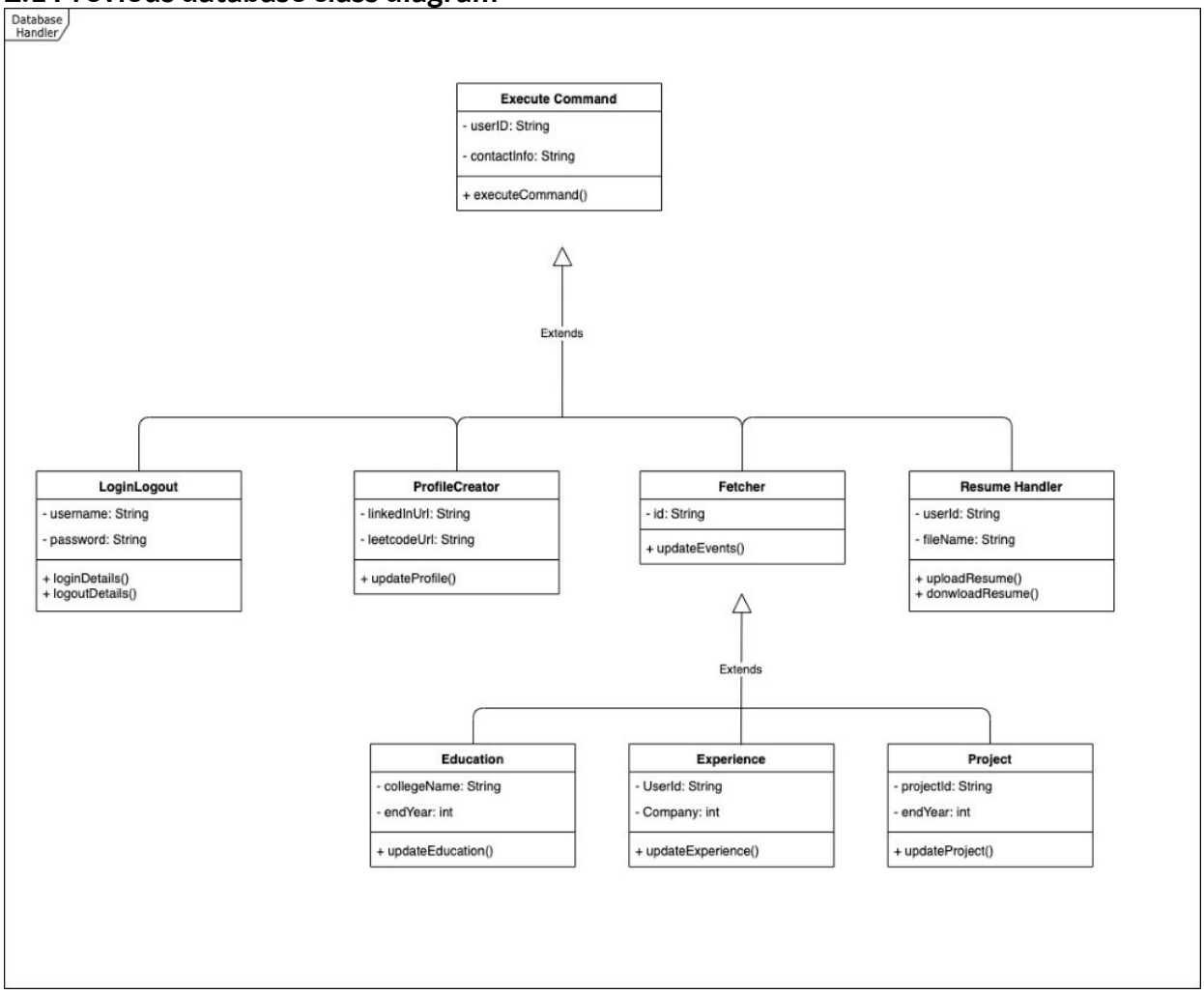
- We added a new Cloudinary class which will store user resume, profile pic and cover pic into the cloud. This class helps in uploading and fetching user resume, profile pic and cover pic.

- We have updated our GitHub class as we have added a new functionality. We are fetching GitHub repository of the user and displaying it in user profile. We added this functionality in our GitHub class and thus it gets updated. The updated GitHub class diagram in shown in new class diagram.

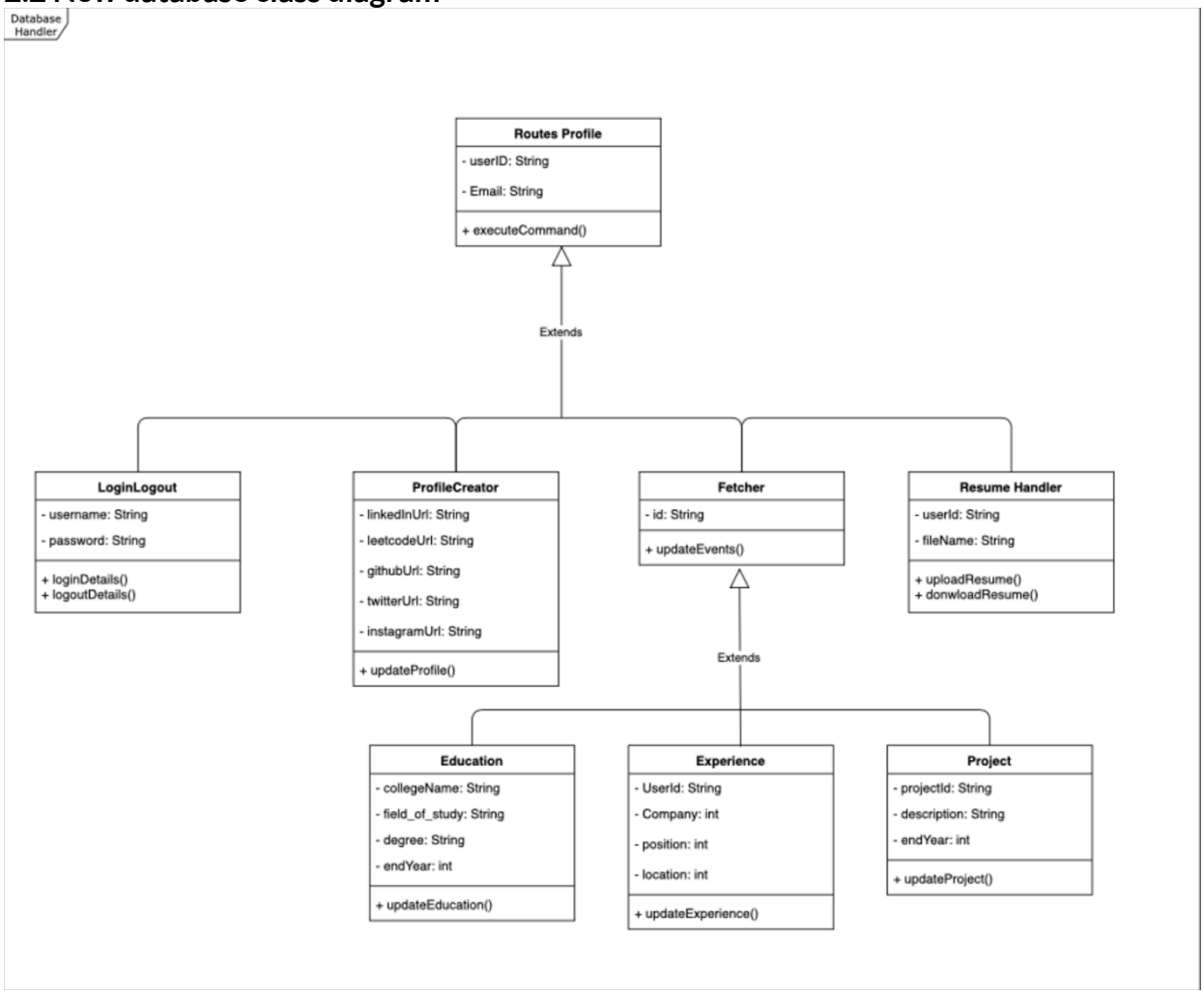
- We are using a strategy pattern in the NodeJS server. For most use-cases, we need to fetch or write different data into the database. For accessing the database, we created router API which handles executing database calls. All the objects in the database diagram will implement Execute() in their own way. So we are using strategy pattern to separate these method and using delegation to implement these methods in their own way.

- We are using an observer pattern to display all the user information on UI. We have made a class named UI which will be responsible for displaying all the necessary information on the UI. The student class will notify the UI about the changes and then class UI will update those changes.

2.1 Previous database class diagram



2.2 New database class diagram



Database changes:

In our database system we had to change the way we interact with database. We created a Routes API which now handles all backend database interaction. It handles all update and delete calls.

Third-Party code vs. Original code Statement

Below is the tutorial we followed to write our code.

1. We used below tutorial for understand NodeJS and mongoDB database.
   https://www.udemy.com/course/mern-stack-front-to-back/
   In this tutorial it was designed for students to understand NodeJS server and MongoDB backend and React for frontend.

2. React JS front end was used for basic user management.
   We used https://cloudinary.com/documentation/node_integration for storing images and resume

3. This Developer profile helped us debug and write code which was taught to us in the tutorial DevFinder

Below is functionality which tutorial taught us and we wrote code according to this tutorial with our OOAD design pattern.

1. MongoDB database management
2. User registration and profiles

Below is the functionality which we added according to our project (profile aggregator)

1. Social media profiles
2. Projects and copy links
3. Student profiles sharing buttons
4. Cloudinary management for images and resume
5. Resume display management

JS files taught in the tutorial. (coded by us, taught in tutorial)

Server.js
Auth.js
dashboard.js
Login.js
Register.js
Navbar.js
ProfileItem.js
Profiles.js
routing
PrivateRoute.js
Routes.js
client/src/components/profile-forms/AddCourse.js
client/src/components/profile-forms/AddEducation.js
client/src/components/profile-forms/AddExperience.js
client/src/components/profile-forms/CreateProfile.js

client/src/components/profile-forms/EditCourse.js
client/src/components/profile-forms/EditEducation.js
client/src/components/profile-forms/EditExperience.js
client/src/components/profile-forms/EditProfile.js
client/src/components/profile/Profile.js
client/src/components/profile/ProfileAbout.js
client/src/components/profile/ProfileExperience.js
client/src/components/profile/ProfileEducation.js
client/src/components/profile/ProfileCourse.js
client/src/components/profile/MyProfile.js
client/src/components/profile/ProfileTop.js

JS files we wrote/updated.

/client/src/components/profile-forms/UploadResume.js
client/src/components/profile/ProfileResume.js
client/src/components/profile/ProfileGithub.js
client/src/components/profile-forms/UploadFile.js
client/src/components/profile-forms/UploadImages.js
client/src/components/profile-forms/UploadResume.js
server.js
routes/api/auth.js
routes/api/profile.js
routes/api/users.js
middleware/auth.js
models/Profile.js
models/User.js

Statement on the OOAD process for your overall Semester Project

We experienced a lot Up's and Down's in our project and would like to share with the course management.

Below are the learnings from this project

1. The **class diagra**m and activity diagram helped us keep our code clean and not include any redundant code.
2. **Sequence diagram** has helped us complete all functionality with fewer bugs
3. **Mock-up's** helped us to keep UI clean and easy.
4. Coding was lot easier with **OOAD concepts** and the process followed from Project 3,4,5 to project 6 was structured properly.

One Down point:

1. We had problems in choosing right technology up-front like **database** and **backend** server, had to change in fall break (from flask to NodeJS and MySQL to mongo dB). Needed some method to analyse our system and suggest correct database.