SW Engineering CSC648-01 Summer 2024

CodeConnect

Team 1 - PikaDevs

Max Shigeyoshi - mshigeyoshi@sfsu.edu - Team Lead
Aaron Rayray - arayray@sfsu.edu - Github Master
Noah Hai - nhai@sfsu.edu - Doc Editor
Shez Rahman - srahman2@sfsu.edu - Backend Lead
Ghadeer Al-Badani - galbadani@sfsu.edu - Backend
Majd Alnajjar - malshemari@sfsu.edu - Frontend
William Pan - wpan1@sfsu.edu - Database Admin
Phillip Ma - pma1@mail.sfsu.edu - Frontend Lead

“Milestone 5“

7/31/2024

History Table:

Date	Changes

1) Product summary

http://54.153.3.191:3000/index

CodeConnect serves as a superior platform for coders to enhance their technical skills and to build strong connections within the tech community. CodeConnect user profiles celebrate their journeys and progress by highlighting skills and interests with projects they've worked on, courses or challenges they've completed, communities they are a part of, and more. Through our website, coders can join groups and participate in forums targeted towards niche technical interests, and become deeper rooted in their communities through events and social gatherings promoted by other users through the website. The website encourages a diversity of different skill levels by enabling mentorship programs with different tiers of subscriptions at the discretion of the mentors, who are also encouraged to take on mentees through their own points and ranking system as mentors. This encourages people of all skill levels to connect with each other, enabling the upward mobility of all users. Mentorships can work through a variety of mediums including group classes, 1 on 1 sessions, prerecorded tutorials, resume advice, or mock interviews.Our main audience consists of rising coders, whether it be a hobbyist or a student in university. Our services go beyond just beginners, our target audience extends much further going to experienced developers working in a tech related field or even instructors that teach coding. By integrating both educational and social interaction, CodeConnect offers a unique oppurtunity for our audience and sets it apart from other platforms. Our marketing strategy consists of digital campaigns, partnerships and engaging within the community. We plan to enhance our website with social media marketing and create captivating content to attract traffic to our platform. By implementing partnerships with institutions such as schools and tech companies, we will further expand our reach while at the same engaging with events in the tech industry and conferences which will further increase our visibility to the public. By collaborating with online communities and hosting events online we will build a strong user foundation and offer a referral program that will give incentive to current users to join. CodeConnect will operate on a free model, offering a free tier with basic features, and a premium subscription option with more advanced features. We will also be providing custom mentorship and training programs and will get a dedicated team to reach out to schools, tech companies, coding bootcamps, etc to promote our platform. By using this approach of marketing and sales, CodeConnect aims to become the number one platform for coders of all skill levels. Our vast accumulation of resources and opportunities to grow make us the number one option for users looking to thrive in the tech industry, which is why users should join CodeConnect today and take the first step in becoming a better coder.

Final P1 Functions:

1) **All Users:**
   *1.1 Users shall be able to explore some portions of the product without a profile
   *1.2 Users shall be able to solve an example problem.
   *1.3 Users shall create a profile
   *1.4 Users shall be able to Log in/Log out (only with created profile)
   *1.20 Users shall be able to check their coding ranking

*1.21 Users shall be able to check leaderboards
*1.22 Users shall be able to subscribe to Premium
*1.23 Users shall be able to unsubscribe to Premium
*1.32 Users shall be able to display other socials on their profiles
*1.34 Users shall be able to request additional features from the dev team
*1.35 Users shall be able to utilize live chat w/ other users
*1.36 Users shall be able to direct message other users
*1.45 Users shall be able to submit support forms to submit feedback regarding the app
*1.54 Users shall be able to join group session workshops led by mentors
*1.60 Users shall be able to get assessed to become a mentor
*1.61 Users shall be able to use a compiling development environment without having to create an account
*1.66 Users shall be able to see the difference between premium and free membership perks
*1.67 Users shall be able to gain points by starting forum threads
*1.68 Users shall be able to gain points by commenting in threads
*1.69 Users shall be able to gain points by gaining friends
*1.70 Users shall be able to gain points by gaining mentees
*1.71 Users shall be able to gain points by gaining mentors
*1.72 Users shall be able to start new forum threads
*1.95 Users shall be able to reply to existing forum threads
*1.96 Users shall be able to have private forums, also known as groups
*1.97 Users shall be able to join mentor forums with mentees sharing common mentor
*1.76 Users shall be able to search for other users
*1.77 Users shall be able to search for forums also known as groups
*1.78 Users shall be able to receive notifications for messages.
*1.99 Users shall be able to receive notifications for replies to their threads
*1.100 Users shall be able to view information about mentors before selecting them
*1.93 Users shall be able to view a list of job listings from home page
*1.94 Users shall be able to view other users' activity from their homepages
*1.101 Users shall be able to raise their rank by crossing point thresholds
*1.102 Users shall be able to click on other users' profiles from leaderboard
*1.103 Users shall be able to click on other users' profiles from their forum posts
*1.104 Users shall be able to have profile icon pictures
*1.105 Users shall be able to meet other users by joining a workspace
*1.107 Users shall be able to add a password to a new workspace
*1.108 Users shall be able to filter mentors by category

2) **<u>Free Users</u>**
*2.2 Free Users shall be able to check code challenge repo

*2.3 Free users shall be able to view three pseudocode hints per month
*2.4 Free users shall be able to view public forums
*2.5 Free users shall be able to view other profiles
*2.6 Free users shall be able to create an account
*2.7 Free users shall be able to view a splash page to inform them about the website

2)Milestone Documents

MILESTONE 1v1:

SW Engineering CSC648-01 Summer 2024

Milestone 1

CodeConnect

Team 1 - PikaDevs

Max Shigeyoshi - mshigeyoshi@sfsu.edu - Team Lead
Aaron Rayray - arayray@sfsu.edu - Github Master
Noah Hai - nhai@sfsu.edu - Doc Editor
Shez Rahman - srahman2@sfsu.edu - Backend Lead
Ghadeer Al-Badani - galbadani@sfsu.edu - Backend
Majd Alnajjar - malshemari@sfsu.edu - Frontend
William Pan - wpan1@sfsu.edu - Database Admin
Phillip Ma - pma1@mail.sfsu.edu - Frontend Lead

6/13/2024

History Table:

Date	Changes

Executive Summary:

The rise of software development has opened pathways for people all over the world to innovate, express, and even attain economic mobility for themselves and their families. Learning how to code is learning how to learn – while anyone can do it, there exist barriers of entry that can discourage or slow down the journeys of aspiring programmers; these barriers include the intimidation factor of learning new tools, as well as a lack of social support. While other companies have contributed to this pool with collaborative environments and guided tutorials, there exists a need in the marketplace for a website that not only cultivates skill growth and career mobility but also actively connects coders with each other. A hub is needed for coders to connect with their peers, find mentors, or create new friendships over a common interest in coding.

This is where CodeConnect takes the wheel. CodeConnect user profiles celebrate their journeys and progress by highlighting skills and interests with projects they've worked on, courses or challenges they've completed, communities they are a part of, and more. Through our website, coders can join groups and participate in forums targeted towards niche technical interests, and become deeper rooted in their communities through events and social gatherings promoted by other users through the website. The website encourages a diversity of different skill levels by enabling mentorship programs with different tiers of subscriptions at the discretion of the mentors, who are also encouraged to take on mentees through their own points and ranking system as mentors. This encourages people of all skill levels to connect with each other, enabling the upward mobility of all users. Mentorships can work through a variety of mediums including group classes, 1 on 1 sessions, prerecorded tutorials, resume advice, or mock interviews.

Along with sharing projects or certifications done through external sources, CodeConnect also provides tutorial packages and challenges on a range of different software sectors. By participating in these exercises, user profiles are further enriched, and recommendations for connections based on these interests are fostered for mentorships and connections with new peers.

Personas:

- (Aspiring Developer)
- (Student)
- (Mentor)
- (Coder)
- (Employment Prepper)
- (Interview Facilitator)
- (Verified Software Engineer)
- (Resume Assistant)
- (Startup Founder/entrepreneurs)
- (Recruiter)
- (Experienced Programmer)
- (Coding Enthusiast)
- (Bootcamp Graduate)
- (CS Graduate)
- (CE Graduate)
- (Internship Seeker)
- (Freelancer)
- (No-experience coder)
- (Full-time Job seeker)
- (coding instructor)
- (hackathon organizer)
- (Project Manager)
- (Game Developer)
- (Cybersecurity Specialist)
- (Programming Specialist)
- (Data Analysts)
- (Transitioning to a Tech Career)
- (College Instructor)
- (Experience with Frontend but not Backend)
- (Investor)
- (Aspiring Hackathon Participant)

Use-Cases:

1)

- **Actors: Toby(User), CodeConnect(Company)**
- **Assumptions:**

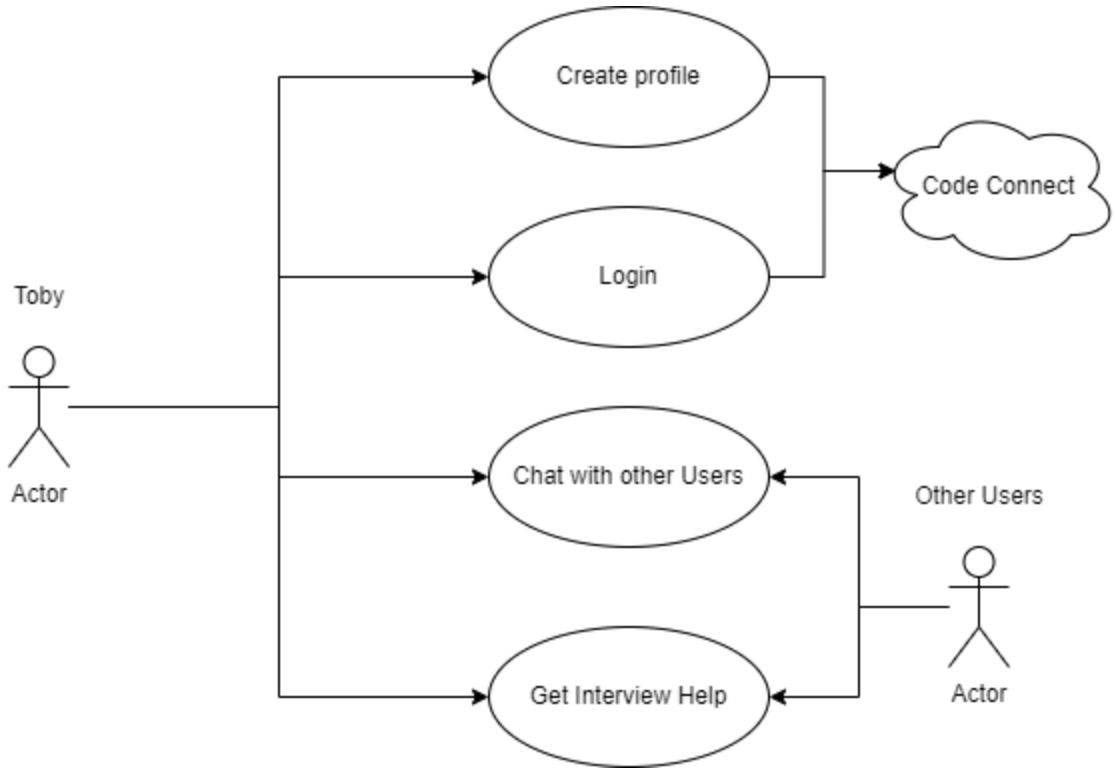
- Toby is a Computer Science Student.
- Toby is looking for a job within the tech industry.

- **Use Case:**

Toby is a senior at San Francisco State University and is coming close to finishing up his years in college. He is overwhelmed by the lack of internship opportunities that he has had throughout his college years. He is concerned that he will not have any experience when it comes to finding a job after he graduates. He started his undergraduate degree in Computer Science during the peak of COVID-19, so it was hard to network with other classmates during remote and Zoom meetings. He searched online for a networking website strictly for coders and computer science students, leading to his finding of CodeConnect. The website had much to offer, beginning from learning the basics of each language and to finding new friends to talk about their internship experiences. He found a section that even allowed him to learn about exact questions used for specific companies and their roles. Although these features were useful towards Toby, that wasn't his main point of going on this website. He wanted to get an internship/job after his college years. This led him to dive into the website more and find a resume helper/builder and get feedback on his current one. After fixing up his resume, now he wanted to learn more about technical interviews and how to ace them. He finally found the interview helper and that guided him to learn more about problem sets and how to respond to interviewers with more intent of learning on the job, due to no experience. With all these tools available from this website, he finally landed a job 3 months later.

- **Benefits for Toby:**

- Met new people in his field to network and gain more information
- Found a new website that allowed him to learn and get accessible help from experienced coders
- Relieved the stress of post-graduate job searching



2)

- **Actors: Harold(User), CodeConnect(Company)**
- **Assumptions:**

- Harold has no experience programming.
- Harold wants to learn coding basics and see if he likes it.

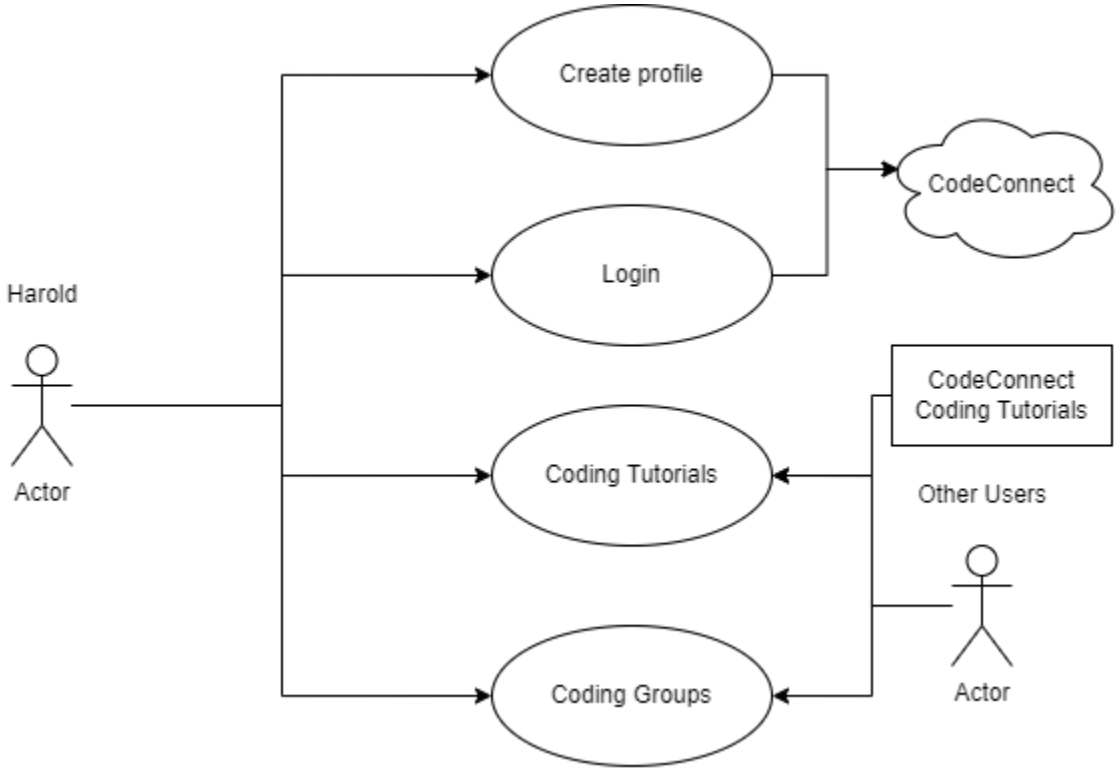
Use Case:

Harold has been working as a barista for 6 years and has decided that he wants to venture into something that generates more income and isn't as hard on his body. He's creative and likes playing video games, so he's always been curious about how he'd do making software. Harold's also an extrovert and learns better in group settings, but he doesn't want to make the commitment to going to school to learn how to code. At the same time, he wants something credible and serial – not just a simple YouTube tutorial for a single program. Harold types in "coding lessons" in Google and finds CodeConnect. At a glance, he sees a tutorial for a basic "Hello World!" java program, completes it, and gains points. Harold is invited to create an account to log the points he just earned, and eager to do so with the dopamine from achievement, he creates his profile. He's recommended a couple of connections who have completed the same tutorial as well as a mentor who does group sessions every Tuesday at 6 pm. Harold attends his first intro to Java group session and meets other like-minded beginner coders as well as a mentor who he continues building on his coding journey with.

- **Benefits for Harold:**

- Learned coding basics in Java and has access to serial tutorials
- Connected with other beginner coders

- Connected with a mentor



3)

- **Actors: Josh(User), CodeConnect(Company)**
- **Assumptions:**

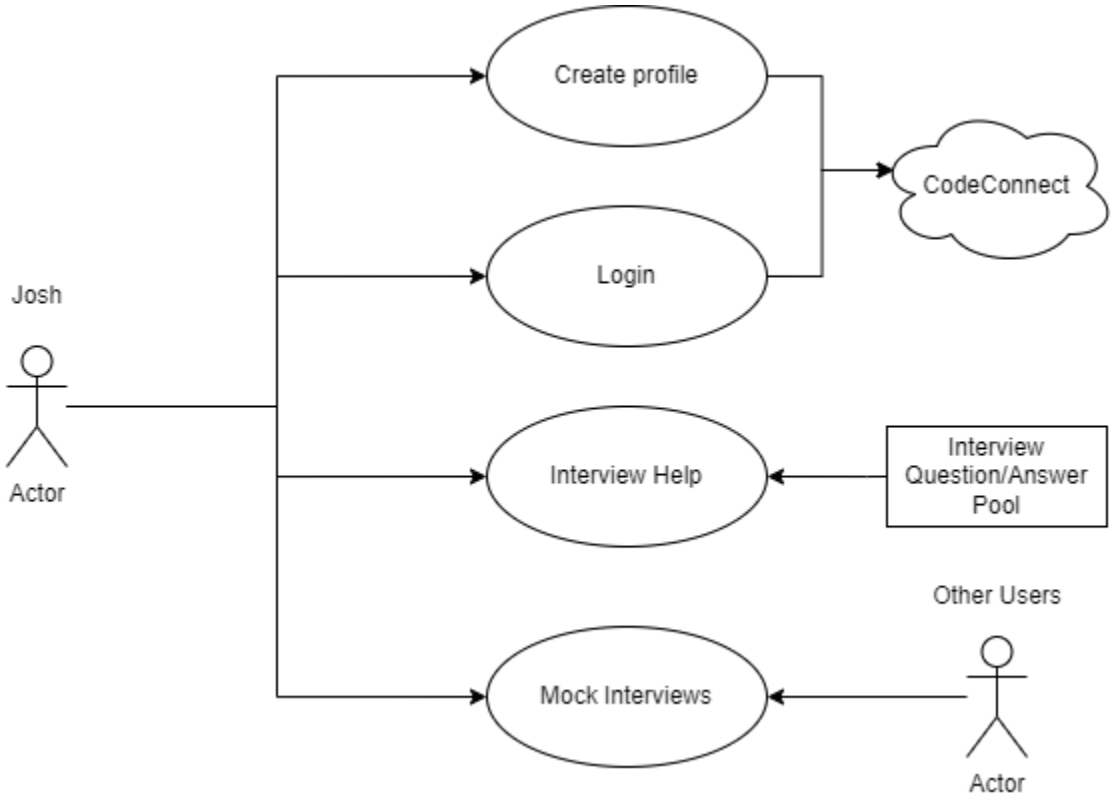
- Josh is an experienced coder.
- Josh is looking for interview help.

Use Case:

Josh is an experienced coder who wants to improve his interview skills as he looks for a new job. He has utilized a wide variety of programming languages and development methodologies during his career. He feels unprepared for coding jobs at the moment because they need an entirely new set of abilities and perspectives. Josh comes upon CodeConnect while browsing the web completely by accident; it's a platform where programmers can interact, share knowledge, learn from one another, and prepare for interviews. Josh comes upon CodeConnect while browsing the web completely by accident; it's a platform where programmers can interact, share knowledge, learn from one another, and prepare for interviews. Curious, he decides to sign up and check out the service, which he finds enjoyable. Josh finds a wealth of useful materials, including an exhaustive list of frequently asked interview questions, which are sorted by firm and job. During mock interviews, his teachers and friends provide him with immediate feedback, which he eagerly incorporates to enhance his performance and boost his confidence for upcoming interviews. As a result of these sessions, Josh is able to recognize phony job interviews and improve his replies and coding skills with the help of recorded comments he finds

online. His impressive use of these methods during practice interviews shows how far he has come and boosts his confidence. Josh is getting closer and closer to landing his ideal job thanks to the help of CodeConnect. He feels prepared for his future interviews.

- **Benefits for Josh:**
- Sharpened interview abilities via practice and feedback sessions
- Access to an extensive database of interview materials, including FAQs, which will help with thorough preparation
- Gained a better grasp of computer programming after using CodeConnect feedback and resources



4)

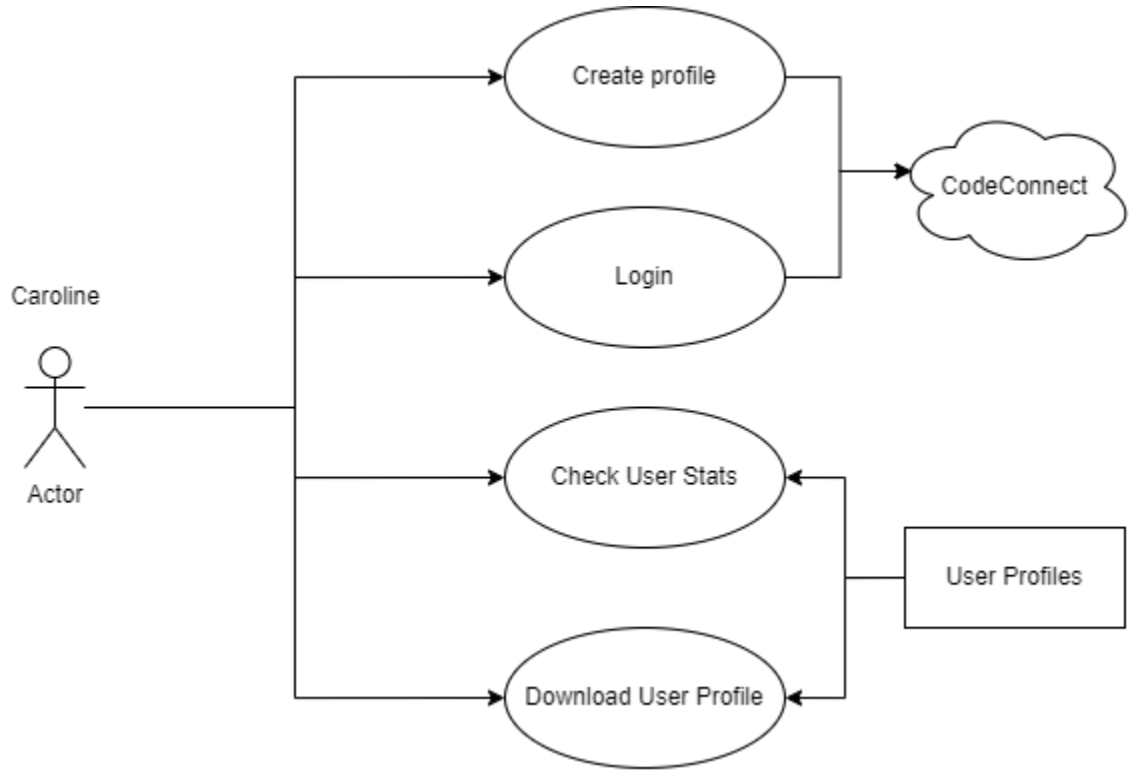
- **Actors: Caroline(User), CodeConnect(Company)**
- **Assumptions:**
- Caroline is a recruiter.
- Caroline is looking to hire a new employee.
- **Use Case:**

Caroline is a recruiter for a company, and looking to hire a new employee. Caroline is going through the hiring process at her company and doesn't really know how to differentiate between many of the qualified applicants. Since Caroline isn't a programmer, she may need to consult with other members of her hiring team who are and will need to find more information as to the abilities of the applicants. Her company has their own testing, however, Caroline has doubts as to

how rigorous those tests are and is looking for more information that could help her determine the applicant's abilities. Some of the applicants have added that they are members of our website, and Caroline is going to sign up for our Recruiter member profile. After verifying her employment, Caroline can see the solutions of the potential employees and was able to send these solutions to other team members to verify that these solutions are signs of a capable Software Engineer/Programmer. Caroline was able to search through our rankings and see the relative rankings of all the applicants that she was searching for during her hiring process. Since she was able to see the potential of some of the qualified applicants, Caroline was able to make a more informed decision about who she should hire.

- **Benefits for Caroline:**

- The capability to verify the coding abilities of a possible employee.
- The ability to make a better hiring decision based on the information we provide about our member's abilities.
- The ability to connect with other potential employees.
    - The ability to share the information from our website with other team members.



5)

- **Actors: Heide(User), CodeConnect(Company)**
- **Assumptions:**

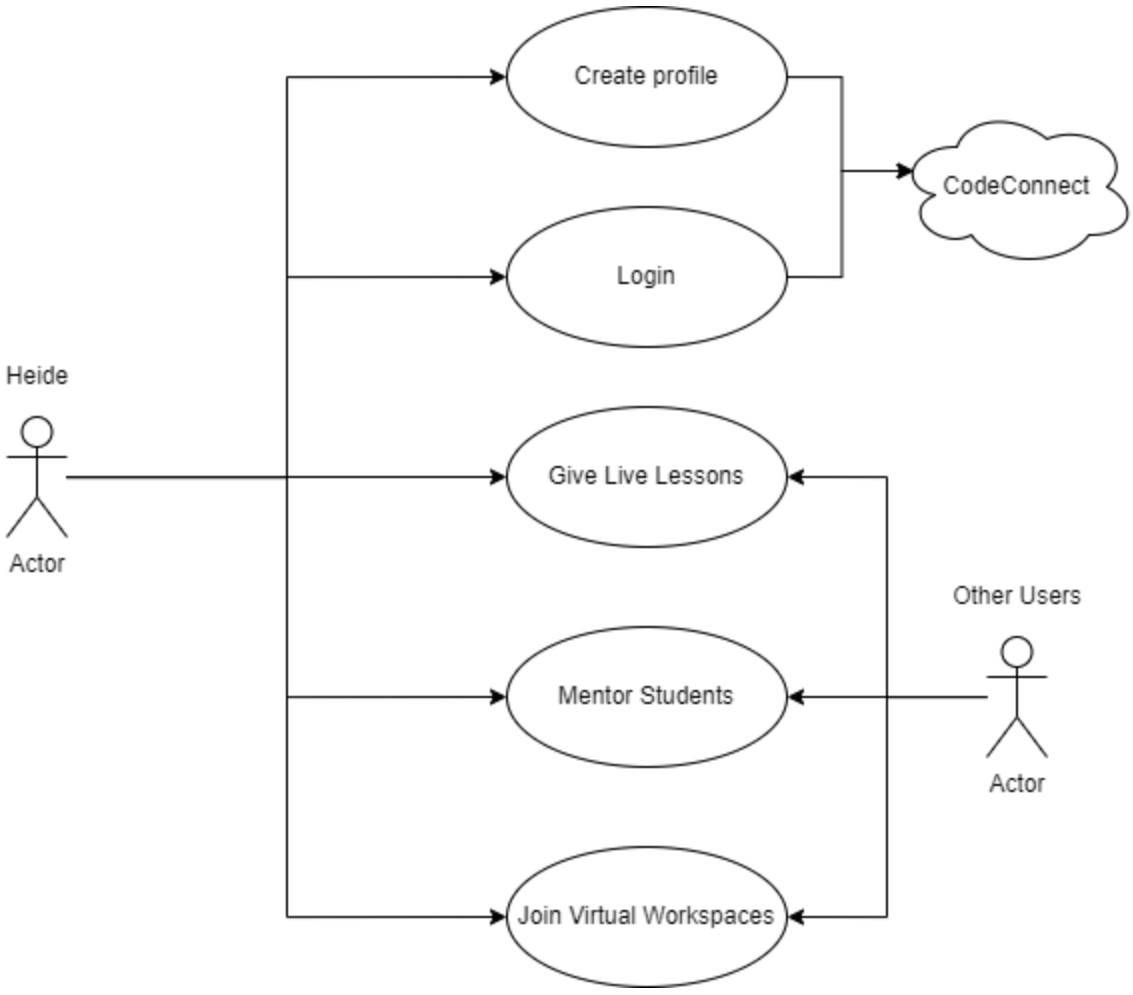
- Heide is an experienced coder, working in the tech industry.
- Heide wants to mentor the next generation of coders.

- **Use Case:**

Heide is a front-end Software Engineer for BigCommerce and is looking for ways to help others learn. Heide is from a small town and there aren't many big tech companies around as she works remotely for her job. She graduated with a B.A. in Computer Science in 2017 and has been working at BigCommerce since then. She loves her job but hopes to teach the youth and others what she has been taught and doesn't want to gatekeeper her knowledge. Heide then scrolls the internet for places to volunteer and can't seem to find anything locally as she lives in a small town. She finally comes across a website called CodeConnect and she explores the wonders it has to offer. Judging from the website, she thought it was initially just another website with information on learning how to code until she stumbled upon a feature stating, "Have experience? Mentor now!" This is exactly what she was looking for, she then signs up for an account and goes through the mentorship approval stages. Also with doing this, she updates her portfolio and resume so the verifiers can go through her approval quicker. The next stage was the simple interview process to make sure Heide was the real deal and in less than a week, she was approved. She now owns the Mentor role for this website and could teach/give lessons to others within the virtual workspace. She also has the authority to leave feedback on code review and interview help.

- **Benefits for Heide:**

- Gaining mentorship experience
- Giving others her knowledge within the tech industry
- Giving back to the community as that is what she wanted to do initially



6)

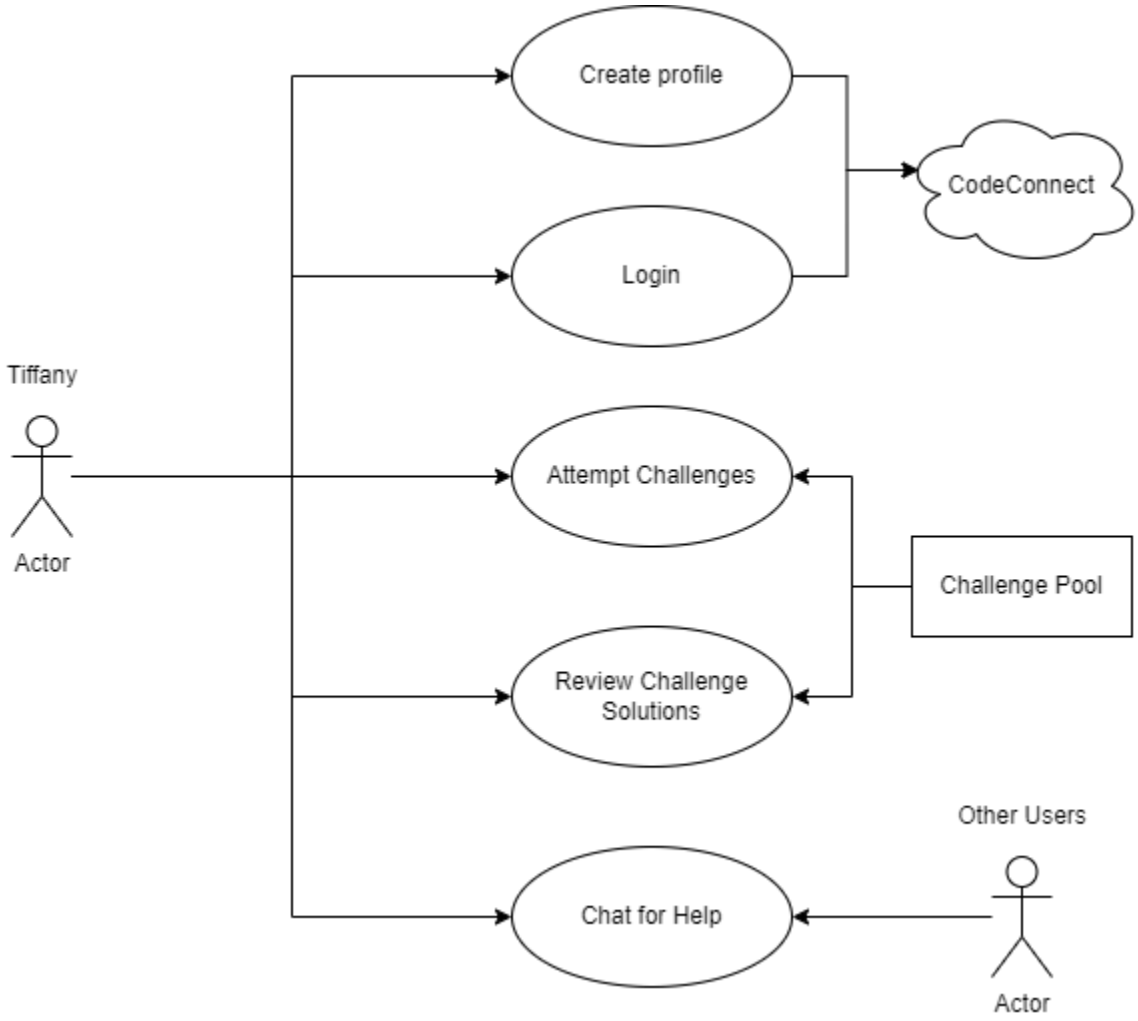
- **Actors: Jeffrey(User),CodeConnect(Company)**
- **Assumptions:**
- Jeffrey is a new graduate, with a few projects in his portfolio
- Jeffrey is looking to learn some new languages, in order to expand his coding experience.
- **Use Case:**

Jeffrey is a new graduate just starting to look for work in the tech industry as a software developer. He has a few projects from school and has a few languages under his belt, however, he has been looking to expand his experience a bit. He finds Leetcode challenges enjoyable, but once he is stuck on a problem he finds it difficult to figure out resources or mentorship to help him. After a quick Google search, Jeffrey found our website and noticed that we offer mentorship with other more senior members on our website. He also noticed that many of the languages that he wanted to learn had challenge problems that he could work through to learn the language. Jeffrey made an account on our site, and starting working on the challenges until he found one that he wasn't familiar with. He then utilized our chat feature to talk to other users in the attempt to gain more insight on the problem and was able to figure out the problem with their help. After a few more problems, he found a problem that he was having problems with, and

after consulting the chat, with little success, he decided to attempt to reach out to a mentor. After connecting with a mentor, Jeffrey was able to work through a few of the more challenging problems successfully.

- **Benefits for Jeffrey:**

- Can experience new languages in the challenges we provide.
- Able to gain help through the chat feature, as well as the mentor program.
- Now has experience that he can speak to in interviews for jobs he is applying for.



7)

- **Actors: Tiffany(User),CodeConnect(Company)**
- **Assumptions:**

- Tiffany is an experienced designer
- Tiffany needs backend programming for a freelance project

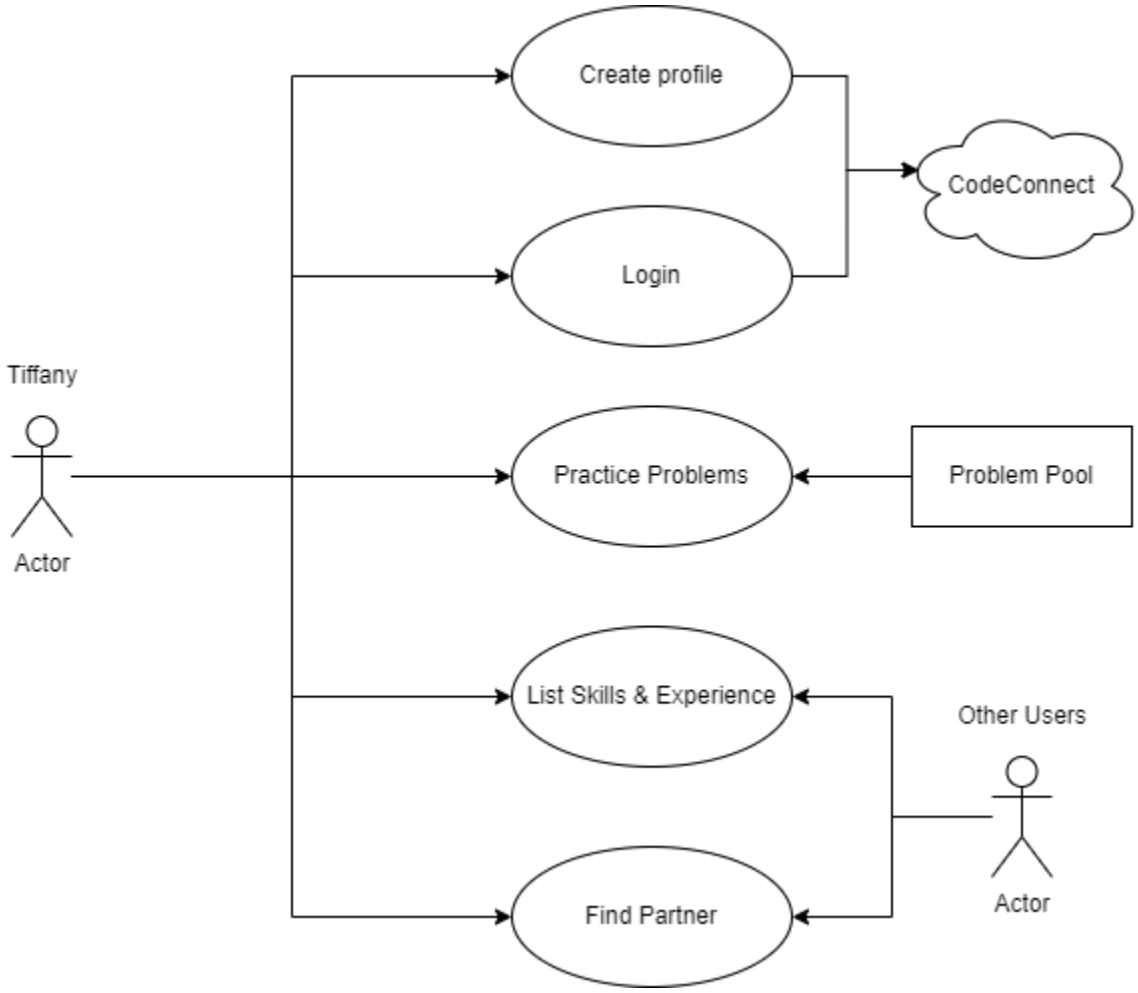
- **Use Case:**

Tiffany is an experienced designer and has been working on the front end of projects at the same company for 3 years. While she's happy where she's at, she wants to take on freelance projects for more creative freedom and to make extra income on the side. Luckily, a potential

client approached her with an idea for a smartphone app where you count sheep flying through the sky to help you fall asleep. However, one issue is that she doesn't have much experience connecting the back end with the design work she does. In order to remedy this, she has two options – either learn how to code on the backend or find someone who would be a partner for her on the project. She remembers a story from a friend who said they basically lived on CodeConnect to build a passion project in the past and visited the website by word of mouth. In the process of creating a profile, she marks that she has front-end experience and an interest in learning backend development. She is then recommended connections with other developers with backend experience and realizes she can use these recommendations to find a partner instead of expending time and resources she didn't have to learn how to take on this project by herself. She also has access to more tutorials to learn backend programming at her own pace.

- **Benefits for Tiffany:**

- Tiffany was able to find a partner to accept a freelance project.
- Tiffany can learn backend programming at her own pace.



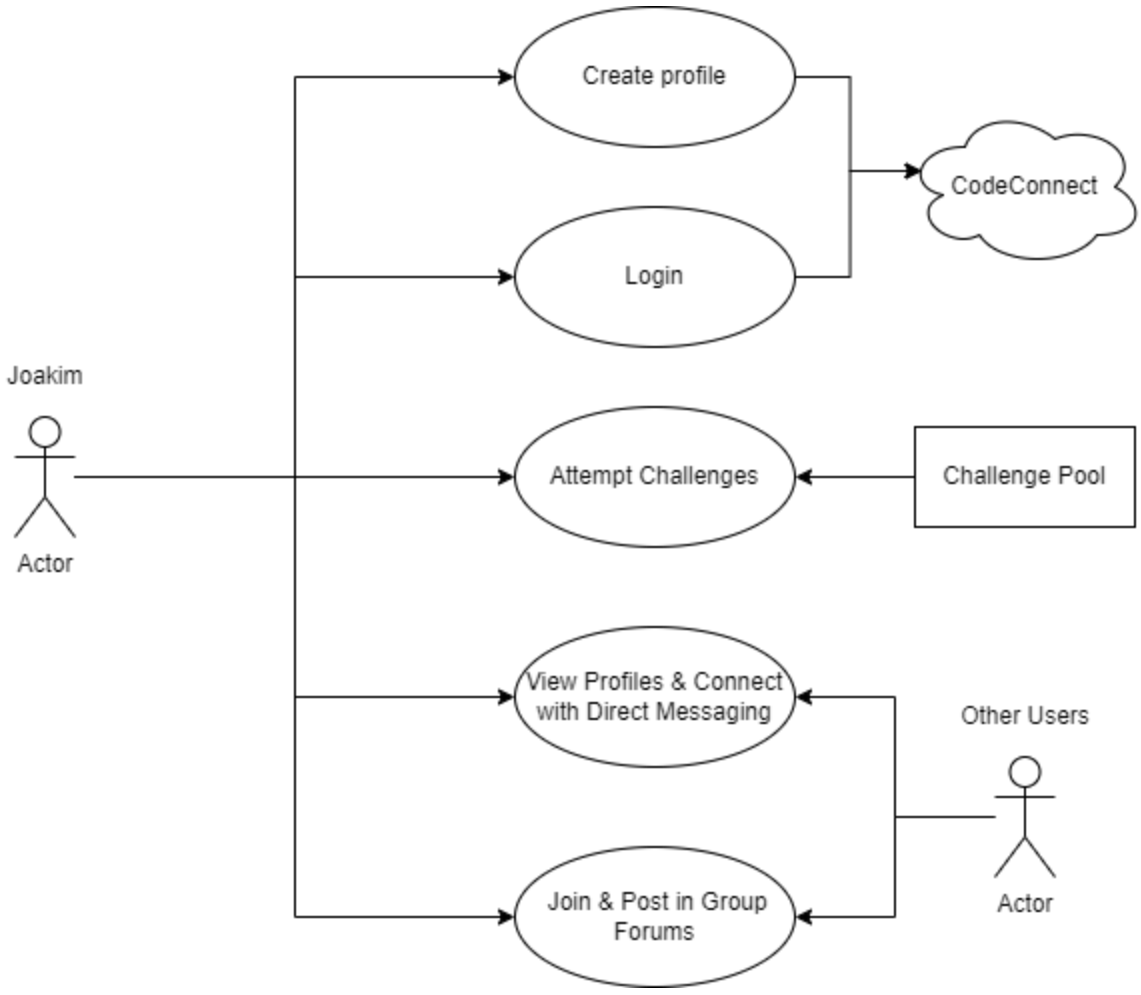
8)

- **Actors: Joakim(User),CodeConnect(Company)**

- **Assumptions:**
- Joakim has had a CodeConnect profile for 3 months
- Joakim needs a team for a hackathon
- **Use Case:**

Joakim wants to participate in his first hackathon but has no experience doing so. As an introvert, he hasn't made many friends throughout school but needs a team to participate. Since Joakim has already graduated from SFSU, there aren't many ways for him to meet other programmers without scouring websites like StackOverFlow and GitHub. Fortunately, since he has been using CodeConnect for 3 months and has built a profile that lists a sizable amount of challenges and projects he has completed, he knows he can rely on CodeConnect to link him with people he might work well with. He goes into a group he's already a member of on CodeConnect titled "SFSU Alumni", and posts in a forum that he's looking for other teammates for an upcoming hackathon. Five other members of the group reply indicating interest. Joakim looks at their profiles to see challenges they have completed, groups they're a part of, and their about sections to get a feel for who may be the best matches for his team. He contacts his two favorite users expressing that he would love to group up with them. They form a team and Joakim participates in his first hackathon.

- **Benefits for Joakim:**
- Joakim was able to find teammates with compatible skills and personalities.
- Joakim was able to enter his first hackathon.



9)

- **Actors: Tory(User),CodeConnect(Company)**
- **Assumptions:**

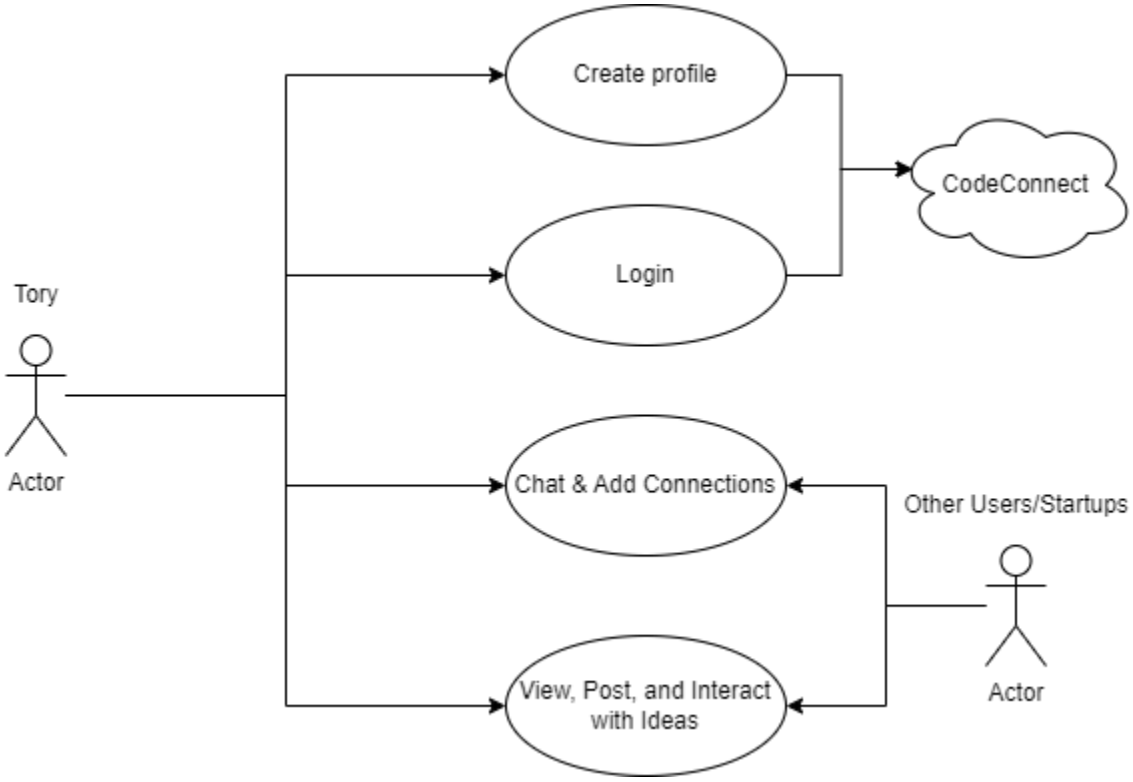
● Tory is a startup CEO trying to find investors for her tech company
● Tory has little connection in the tech industry

- **Use Case:**

Tory is an entrepreneur with a background in biochemistry but lacks connections in the tech industry. She got the brilliant idea of using AI to detect cancer through blood samples. By discovering CodeConnect, she creates a detailed profile and uploads her innovative ideas. Her post quickly gained lots of interaction scores in the community and became one of the most talked-about topics in the forum attracting numerous messages of support and interest. Through CodeConnect, Tory connects with AI experts eager to collaborate, attends workshops that refine her strategy, and engages with angel investors interested in her project. After successful pitches and negotiations, she secures funding from an investor passionate about her startup. With technical support from her new connections and financial backing, Tory's startup makes significant strides.

Benefits for Tory:

- Tory shall able to connect with industry experts
- Tory shall able to find an angel investor that expands her company



10)

- **Actors: Dan(User),CodeConnect(Company)**
- **Assumptions:**

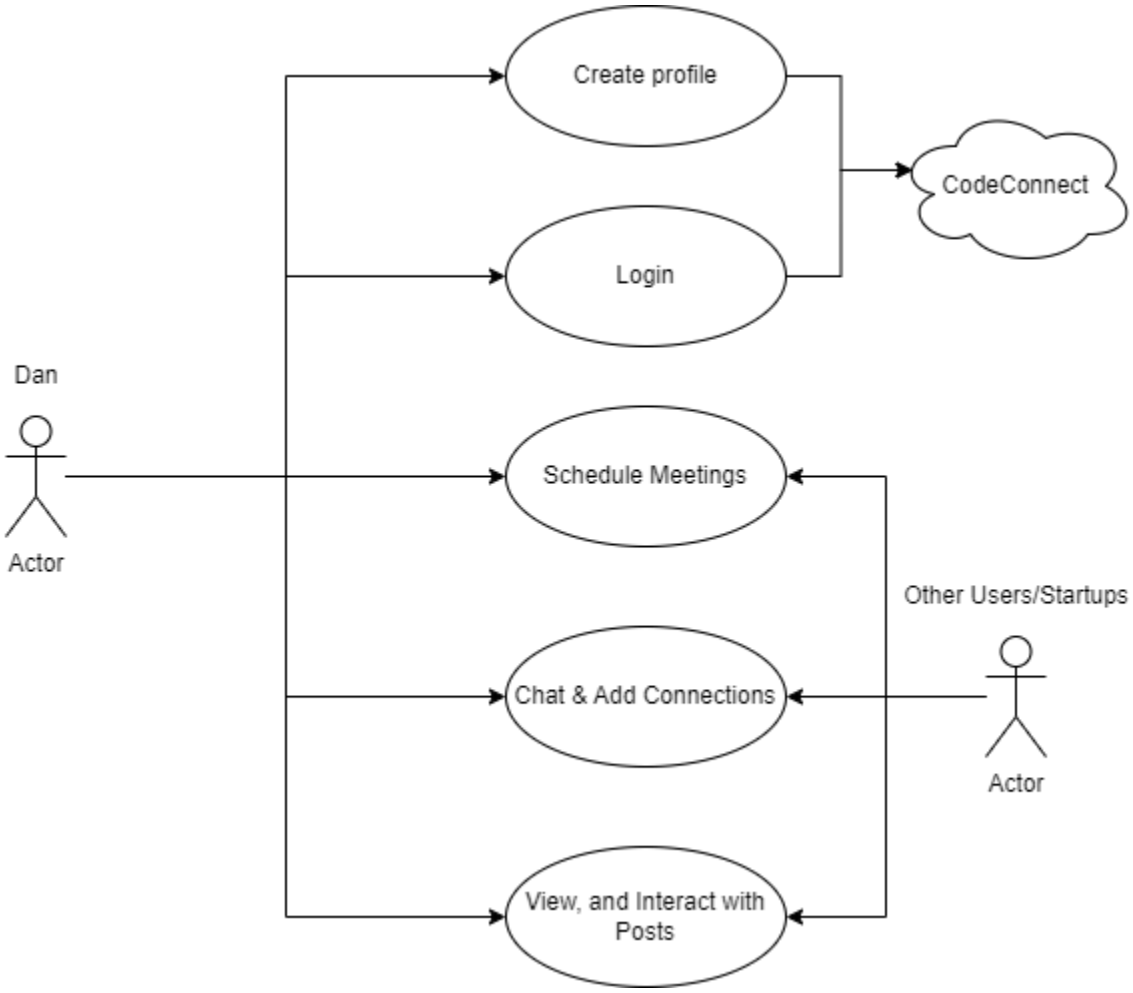
- Dan is an angel investor interested in technologies and healthcare innovations.
- Dan is busy and wants to have a virtual discussion with startup CEO

- **Use Case:**

Dan is an angel investor interested in technologies and healthcare innovations. He heard that codeConnect has a lot of interesting startups, so he created his profile by uploading a picture and setting up his account to the public. While exploring the platform, he comes across Tory's startup, which uses AI to detect cancer from blood samples. Her project is trending with a high interaction score, so Dan checks out her profile, follows her, and sends a direct message to learn more. They schedule a meeting through the codeConnect calendar, and after a thorough discussion, Dan is impressed by Tory's vision and the progress she's made with AI experts. Thus, he decides to invest in

Tory's company. To further support Tory, Dan subscribes to premium features, participates in group workshops, and uses the virtual workspace for collaboration. He also uses the internship/job portal to help Tory find top talent, ensuring her startup's growth and success.

- **Benefits for Dan:**
- Dan shall be able to expand his investment opportunities in a more efficient way
- Dan shall be able to gain deeper insights by connecting with startup CEOs directly



<u>Main Data Items and Entities</u>

Data Structure:

- User : The user that creates the account.
    - User ID
    - First Name
    - Last Name

    - User Name
    - Membership Status (Free user, paid user, mentor)
    - Email
    - Password
    - Points
    - Rank
    - Array of challenge IDs of completed challenges
    - Number of challenges completed
    - TrophyCollection
    - Streak of challenges completed
    - **Social Links**
    - **Coins**
    - **Mentees? Or in profile?**
    - Notification list
    - **Bookmarks**
    - **Saved Posts**
    - **History**
    - **Analytics**

- Notification
    - Title string with template
    - Redirect link upon clicking the notification
    - Date
    - Time

- Premium User : A user that has paid the subscription fee for premium features
    - All data from normal user
    - Payment information

- Mentor User : A user that has become a mentor to other users. Must be approved by interview.
    - Inherits User
    - Payment information
    - Number of solution feedback forms completed
    - List of mentor groups mentored

- **Hiring/Manager User:**
    - **Company**
    - **Position**

- Profile : The user's profile which has information about the user.
    - User ID (From user who posted)
    - Profile ID
    - Profile type (student, mentor, recruiter, etc)
    - Profile info (current role, education)
    - Resume
    - Project
    - **External links - dropdown menu of linkedin, x, instagram, tiktok, facebook? Empty text box? FR 1.32**
    - Rank
    - **Mentees? Or in user?**

- Forum post : Posts that users can create in order to interact with the larger community.
    - User ID (from user who posted)
    - User rank
    - Post ID
    - Post title
    - Post content
    - List of post comments
    - Post code block
    - Post time
    - Post likes

- Forum : A collection of forum posts and data about the forum
    - Forum name
    - Data created
    - List of members who have access
    - Collection of forum posts

- **Mentor forum post:**
    - **Forum post**
    - **Mentor tag for mentors**

- Coding challenge : The coding challenges that each user has access to and can attempt to solve.
    - User ID (from user who posted)
    - Coding title
    - Coding ID
    - Coding desc
    - Coding language

        - Coding difficulty
        - Coding code block
        - Coding submission deadline
        - Coding leaderboard
        - Coding rank points
        - Coding participants
        - Coding solutions
        - Coding tests
        - Pseudocode hint

- Coding challenge submission
    - Coding challenge data
    - User data who is submitting
    - Date submitted
    - Time submitted
    - verifiedSolution: Boolean

- Challenge post : (FR 1.18)
    - Coding challenge object
    - Likes
    - Dislikes
    - Comments

- Coding challenge feedback : The review form that mentors use to give feedback to mentees on their coding challenge solutions.
    - Coding challenge solution
    - Evaluation of formatting of solution code
        - Written comment by mentor
        - Scale of 1-10 rating
    - Evaluation of efficiency of code logic
        - Written comment by mentor
        - Scale of 1-10 rating
    - Evaluation of utility of comments
        - Written comment by mentor
        - Scale of 1-10 rating

- Leaderboard : list of all users organized by ranking points to be displayed as a table
    - List of all registered Users
        - Each user’s number of points held
        - Each user’s rank title

        - Each user's rank icon

- Rank class : the different icons and titles that users can acquire as they gain more points
    - Rank icon
    - Rank title
    - Rank ID
    - Points range for this rank

- Ranks : list of different rank objects that users can acquire as they gain more points
    - Collection of rank objects
    - checkRequirements: function to check what rank User has and return the correct rank

- Base Trophy class : inherited by SpecificTrophy class
    - Trophy name
    - Trophy ID
    - String describing requirements to earn trophy
    - Trophy flag : true if user possesses trophy
    - checkRequirements : abstract method, implemented by SpecificTrophy

- SpecificTrophy class : users can earn trophies for different achievements
    - Inherits base Trophy class
    - Implements specific checkRequirements function

- Group : users can join these to bond over commonalities such as being alumni from the same school, or having interest in a certain technology
    - Member users
    - Private forum for member users
        - Forum posts

- Mentor group : a group for mentors to communicate with their mentees
    - Mentor users
    - Mentee users
    - Private forum for members
    - Resource page

- Groups : a list of all groups for access
    - List of mentor groups
    - List of regular groups

- Mentor program : Mentorship program that users (free and premium) can attempt to sign up for.
    - User ID (from user who posted)
    - Program ID
    - Program title
    - Program desc
    - Program length
    - Program schedule
    - Mentor group

- Job listing : Job listings that are available for any user to apply for.
    - User ID (from user who posted)
    - Job ID
    - Job title
    - Job company
    - Job location
    - Job desc
    - Job requirements
    - Job date posted
    - Job app link

- Project : these are data items to be stored in portfolio
    - Link to project
    - Text description of project
    - Title of project
    - Screenshots of project

- Portfolio: Users will have this to display their projects
    - List of projects
    - Visibility setting (Public/private)

- Payment information: Users will be able to store their payment information for purchases
    - Card number
    - Name on card
    - Zipcode
    - CVV/CVC code

- Message : Sent between users as direct messaging
    - Name
    - Time

    - Date
    - Message Content

- Inbox : Every user contains an inbox of messages that other users may send them
    - List of messages

- Meeting : Meeting rooms which users can use to meet with other users (free or premium).
    - User ID (from user who posted)
    - Meeting date
    - Meeting title
    - Meeting invitees (list of user_ID's)

- Chatbot : alternative to support form for users to communicate with an AI rep for the company

- Support Form : All users can use these to communicate with the company
    - Name
    - Date
    - Time
    - Message

Functional Requirements: (50+)

1. **All users:**

*1.1 Users shall be able to explore some portions of the product without a profile
*1.2 Users shall be able to solve an example problem.
*1.3 Users shall create a profile
*1.4 Users shall be able to Log in/Log out (only with created profile)
*1.5 Users shall be able to use a SSO login for companies/schools
*1.6 Users shall be able to upload profile picture
*1.7 Users shall be able to Delete profile
*1.8 User shall be able to update payment information
*1.9 Users shall be able to make their profile private or public

**1.11 Users shall be able to follow other users/mentors
*1.12 Users shall be able to check other users profiles/stats
*1.13 Users shall be able to do coding challenges
***1.14 Users shall be able to see popular submissions (how is popular measured?)**
*1.15 Users shall be able to award different profile trophies for achievements

****1.16 Users shall be able to allow switching coding languages for challenges (cross compatibility) (different compilers - maybe 2 or 3 priority here)(would prefer just different sets of challenges based on language)**
*1.17 Users shall be able to earn points for coding challenges
*1.18 Users shall be able to like/comment on challenge posts

*1.20 Users shall be able to check their coding ranking
*1.21 Users shall be able to check leaderboards
*1.22 Users shall be able to subscribe to Premium
*1.23 Users shall be able to unsubscribe to Premium
**1.24 Users shall be able to see pop-up ads for premium/paid utilities
*****1.25 Users shall be able to application survey for mentorship (?)**

*****1.28 Users shall be able to take hiring questionnaire challenges (which are siblings to coding challenges**
*1.32 Users shall be able to connect other socials
*1.34 Users shall be able to request additional features from the dev team
*1.35 Users shall be able to utilize live chat w/ other users
*1.36 Users shall be able to direct message other users
*****1.37 Users shall be able to access free/paid resources (videos/textbooks)**
***1.38 Users shall be able to check coding streak counter of daily challenges**
*1.39 Users shall be able to create a portfolio
*1.40 Users shall be able to check/update portfolio
*1.41 Users shall be able to change portfolio visibility (public/private)
*1.42 Users shall be able to access portfolio review
*****1.43 Users shall be able to utilize chatbot (this function would be cool but bold because need to know the data items) (Use support form instead)**
***1.44 **Users shall be able to access virtual workspace (define virtual workspace?)**
*1.45 Users shall be able to submit support forms to submit feedback regarding the app
***1.46 **Users shall be able to open source/collaborative coding projects (users can participate in other people's coding projects) (Similar to 1.44?)**
***1.47 Users shall be able to read weekly digests (recommendations created for users based on interests/what they worked on)
**1.48 Users shall be able to take mock interviews (practice interviews with real-time communication and feedback from peers and other users)
**1.49 Users shall be able to view featured users (spotlight user that's completed most challenges/stayed the most active for the past month. Featured user gets changed monthly)
***1.50 **Users shall be able to utilize pair programming sessions (allows users to work on projects/code collaboratively) (similar to 1.44 and 1.46)**

**1.51 Users shall be able to utilize discounts/offers on apps premium features
***1.52 **Users shall be able to read technical news (similar to 1.47)**
**1.54 Users shall be able to join group session workshops led by mentors
*1.55 **Users shall be able to link account to other socials(github,linkedin,etc)**
1.56 **Users shall be able to view/update their own calendar(scheduled hackathons, virtual meetings..) (API or creating our own calendar?)
***1.57 **Users shall be able to create Recruiter/Hiring Manager specific profile**
***1.58 Users shall be able to pass a Recruiter/Hiring manager verification/background check
****1.59 Users shall be able to utilize a button to change the screen to dark mode.**
****1.60 Users shall be able to get assessed to become a mentor**
*1.61 Users shall be able to use an IDE without having to create an account
****1.63 Users shall be able to choose their preferred speaking language(How does that look in implementation?)**
1.64 Users shall be able to choose their own country/region (Nonfunctional req? Why is this needed?)
*1.65 Users without premium shall be able to view three solutions per month
*1.66 Users shall be able to see the difference between premium and free membership perks
*1.67 Users shall be able to gain points by starting forum threads
*1.68 Users shall be able to gain points by commenting in threads
*1.69 Users shall be able to gain points by gaining friends
*1.70 Users shall be able to gain points by gaining mentees
*1.71 Users shall be able to gain points by gaining mentors
*1.72 Users shall be able to post text and images to their profiles
*1.73 Users shall be able to view a general feed of other users' updates and activities
1.74 **Users shall be able to earn coins
1.75 **Users shall be able to trade coins for premium subscriptions
-Or top leaderboard users get a premium reward
*1.76 Users shall be able to search for other users
*1.77 Users shall be able to search for groups
*1.78 Users shall be able to receive notifications for new coding challenges.
*1.78 Users shall be able to receive notifications for messages and comments.
*1.79 Users shall be able to customize notification preferences.
1.80 **Users shall be able to bookmark coding challenges.
*1.81 **Users shall be able to save forum posts for later reading.**
*1.82 Users shall be able to report inappropriate content.
*1.83 Users shall be able to block or mute other users.
*1.84 Users shall be able to set privacy settings for profile visibility.
*1.85 **Users shall be able to view the history of coding challenges attempted.**

*1.86 Users shall be able to request a mentor recommendation.
**1.87 Users shall be able to participate in live coding sessions.
***1.88 Users shall be able to join virtual coding bootcamps.
***1.89 **Users shall be able to create and manage a personal blog.**
*1.90 Users shall be able to subscribe to notifications for specific forums or groups
1.91 **Users shall be able to access analytics on their coding performance.
****1.92 Users shall be able to receive notifications for new coding challenges.**
*1.93 Users shall be able to view a scrolling list of job listings from home page
*1.94 Users shall be able to view other users' activity from their profiles

2. **Free Users:**
   ****2.1 Free Users shall be able to check the most average solutions after completing a challenge.**
   *2.2 Free Users shall be able to check code challenge repo
   *2.3 Free users shall be able to view three pseudocode hints per month

3. **Premium Users:**
   ***3.0 Premium Users shall be able to check a single system chosen solution after completing a challenge**
   ****3.1 Premium Users shall be able to check a system chosen set of 2-3 solutions after completing a challenge**
   *3.2 Premium Users shall be able to request mentorship (premium)
   ***3.3 Premium Users shall be able to Match mentors with similar coding language experience (search? Automatic matching?)**
   *3.4 Premium users shall be able to view one pseudocode hint per challenge
   *****3.5 Premium Users shall be able to create code challenge repo, which are reviewed by the dev team prior to publication. (with a full repo it could be harder to check solutions vs a simple IDE plug in/compiler. Resources may be difficult on this topic…maybe talk to prof)**
   *****3.6 Premium Users shall be able to check/update code challenge repo(^see note for 3.5)**
   *3.7 Premium Users shall be able to request code review from a mentor

4. **Mentor Users:**
   *4.1 Mentor Users shall be able to review code solutions of other users they are mentoring.
   *****4.2 Mentor Users shall be able to set up group work stations. (what is a group work station? Also potentially lower priority)**
   **4.3 Mentor Users shall be able to create coding challenges.
   *4.4 Mentor Users shall be able to review Free/Premium users resumes/portfolios

*4.5 Mentor users shall be able to complete challenge feedback on coding challenges completed by mentees
*4.6 Mentor users shall be able to gain points by completing challenge feedback on mentee solutions
*4.7 Mentor Users shall be able to upload videos
*4.8 Mentor users shall have their number of mentees displayed on their profiles
*4.9 Mentor users shall have their number of solutions reviewed displayed on their profiles
*4.10 Mentor users shall have the groups they lead displayed on their profiles

Non-Functional Requirements:

1. **Database Specs**
   1.1) Databases should use transparent data encryption (TDP) to encrypt data at reset with AES.
   1.2) Databases should dynamically mask and de-identify users' PII information for database users and accessed only for privileged ones.
   1.3) Critical databases should automatically save backups preferably once a day.
   1.4) Database's Backups should be kept for at least 7 days in a cost effective storage solution like AWS S3 bucket.
   1.5) Databases should store audit logs according compliance standards and have low overhead on the performance.
   1.6) Any database audit logs should be filtered for sensitive data, encrypted, and compressed.
   1.7) Website databases should be able to scale vertically and horizontally with less than 2 hours downtime.
   1.8) App databases should be able to run when needed, in a high availability cluster (HA) with automated failover and fault tolerance.
   1.9) Any database's super user (SA) and backend user should have at least 32 random hex characters password.
2. **Storage**
   2.1) Premium users shall have all the same information from a free user, but will also have payment information included.
   2.2) Job posts shall be in their own database, linked to user_ID.
   2.3) We shall only have one user database, which indicates what type of user (free, premium, mentor) that user is.
   2.4) Payment information will be in its own database, linked to a user_ID.
   2.5) Forum posts shall be housed in their own database, linked to user_ID.

2.6) user_ID is used to tie all posts to a single user on the backend, but the username will be used for searching purposes.

3. **Security**

3.1) Passwords should be saved using at least a 256 bit hashing algorithm like SHA-256.
3.2) Any database that stores payment information must be PCI compliant. If standards could not be met, they can be stored with a PCI certified external provider.
3.3) Passwords should be hashed with at least 128 bytes of random generated salt to prevent rainbow tables attack.
3.4) Website backend should connect to the database with SSL in case they are in different instances.
3.5) The website should use generated SSL certificates to secure data in transit.
3.6) Website backend or load balancer if used should prevent HTTP connections and forward them to HTTPS.
3.7) Website backend should verify requests source domain with CORS.
3.8) Website backend should set required headers to prevent cross site scripting.
3.9) Website backend should be able to run on multiple servers using a load balancer.
3.10) Website backend should be able to generate a json web token (JWT) that expires in 8 hours or 60 days to keep me signed in logins.
3.11) The website should provide a forget me functionality to allow password resets.
3.12) The generated token must be encrypted using an at least 32 characters random secret.
3.13) The generated token should include the unique user id to facilitate authorization.
3.14) Required authenticated requests should use browser cookies to store the token.
3.15) After a sign up, an email verification should be sent and prevent the user from signing in until verified.
3.16) Need an SSL in order to encrypt the network traffic from our website.
3.17) The backend should be able to identify users from the previously mentioned tokens and reject expired ones.
3.18) The backend should be able to run at least 10 solutions parallelly in separate environments.
3.19) The backend should be able to accept run requests even at max capacity and manage executions on a first come first serve basis.

4. **Performance**

4.1) A continuous integration and continuous delivery pipeline (CI/CD) should be developed after first release to facilitate fast, reliable updates.
4.2) The email verification and password reset urls should use tokens that expire after 30 minutes.
4.3) Chat features should be in real-time and have minimal latency.
4.4) The static front end files should be served from a content delivery network (CDN) to enable a fast global delivery with lower backend traffic overhead.

The website shouldn't take more than a few minutes to verify a solution to a challenge problem.

4.5) All rest APIs should respond in under 5 seconds in all situations.

4.6) Logging into the website shouldn't take more than a few seconds.

4.7) Profile Rankings should be updated live, as a person may find some challenges easier than others and might complete a few of them quickly.

4.8) If a profile does not have a picture associated with it, there will be a default picture placed.

4.9) The user interface should be intuitive and user-friendly, requiring no more than 3-5 clicks to access primary functions

5. **Expected Load**

   5.1) We would expect at least 1000 users at any given time.

   5.2) Leaderboards should be updated every 24 hours, as people's rankings may increase rapidly during that time.

6. **Compatibility and UI**

   6.1) The UI should scale and resize based on window size.

   6.2) Logos should be in the header and footer of every page visited

   6.3)All aspects of any web page concerning lists (such as lists of challenges) should be the largest portion of the webpage, except the challenges which will take up all of the page as it is the most important portion of the product.

   6.4) Username is used mainly in the header to display to users they are logged in, unless .

   6.5) Solutions should be graded within 3-5 minutes of submission.

   6.6) Solutions should be accurate to all tests provided.

   6.7) Solution review should only show the best solutions if the user is a premium member.

   6.8) User's machines shall run Windows and should work on any browser.

   6.9) Links to other social media shall only be viewable on the web page of a user if the user provided those links to us.

Competitive Analysis:

Feature/Company	LeetCode	LinkedIn	Stack Exchange	Instagram	Udemy
Strengths	Helpful splash page, forum posts, clean UI, code challenges,	Networking, Job search features, Company pages, LinkedIn Learning	Group Q&A, variety of topics, Upvote/downvotes	Large user base, broad advertising, engaging content	Variety of topics, affordable, discounts, permanent access to purchased courses, good user testimonials

Weaknesses	Mentorship, company presence, profile presence, private messaging	Limited interactions, spam notifications	Difficult to find specific info, limited interaction, not beginner friendly	Time consuming, concerns for privacy, algorithm can affect content visibility	Little to no interactions w/ instructors, no credentials given once course is completed, quality of courses may vary
Pricing	$35/mo Or $159/yr	Free or 29.99/mo for Premium	Free	Free, ad costs vary	Prices vary based on course, frequent discounts/sales available
Social media	Blog Posts	Blog Posts, articles, Job Updates, Position Status, Looking for Employment, Celebrating achievements, sharing career milestones	Meta forum discussions, community posts	Posts, stories, reels, threads, events, groups, polls, IGTV,	Blog posts, articles, community forums, Meta, Twitter, LinkedIn
Onboarding experience	Personalized content, guided setup, tutorials for features	Initial tutorial, guidance on asking/answering questions from posts	Account setup, tour of features, recommendations to follow people/like pages	Account setup, customizable profile, follow recommendations, content discovery	Easy sign up, personalized recommendations for courses, intro videos/previews for courses, easy UI

Competitive Analysis Summary:

Comparing CodeConnect with our competitors, we have a lot of functional features that will stand out from the rest. One function that stands out from the rest is that we will have a live chat feed from people all across the world to be able to help others with their problems and/or communicate about the technological scene. Another function that stands out from our competitors is that we have a live feed of our leaderboards and users can have their scores displayed for an incentive. This leaderboard consists of a point system that relates to the challenges. The more challenges you complete, the higher you go up in the leaderboards. We offer different amounts of points for each level of challenge. Although we have some features as our competitors such as coding challenges and mentorships, we offer more quality functions that adds to it. For example, our coding challenges offer live support in virtual workspaces that can have mentors individually help you one on one, making it more interactive unlike our

competitors, CodeConnect will have a live chat and mentorship programs that will allow for great user support and keep them engaged. To add to this feature, we offer videos of live demonstrations of completing the challenge or to subjects it's related to. CodeConnect will also include a more customized and personalized onboarding process once users first join the app, taking users through a setup of personalizing their content, tutorials for features and an easy UI which will allow new users to easily be able to learn our platform functionalities. CodeConnect will integrate the strongest features of competitors platforms while improving their downsides which will over all make it the best choice for users looking to break through in tech.

Competitive Features:

+ Feature exists; ++ Superior; - Does not exist

Feature/Company	LeetCode	LinkedIn	Stack Exchange	Instagram	Udemy	CodeConnect
Direct Messaging	-	+	-	++	-	+
Mentorship	-	-	-	-	+	++
Code Review	+	-	+	-	-	++
Technical News	+	++	+	-	-	+
Forums/Groups	+	+	+	+	-	++
Coding challenges/tutorials	++	-	-	-	+	+

Checklist:

- Team found a time slot to meet outside of class (DONE)
- Github Master chosen (DONE)
- Team decided and agreed together on using the listed SW tools and
- deployment server (DONE)
- Team ready and able to use the chosen back and front end frameworks and
- those who need to learn are working on learning and practicing (ON TRACK)
- Team lead ensured that all team members read the final M1 and agree/
- understand it before submission (DONE)
- Github organized as discussed in class (e.g. master branch, development
- branch, folder for milestone documents etc.) (ON TRACK)

High-Level System Architecture:

- Windows (OS)
- AWS (Servers)
- MYSQL (Database)
- Backend Framework
    - ExpressJS (NodeJS)
- Frontend Framework
    - React (Javascript)
    - HTML5
    - CSS
    - Bootstrap (Javascript)

MILESTONE 2v1:

SW Engineering CSC648-01 Summer 2024

CodeConnect

Team 1 - PikaDevs

Max Shigeyoshi - mshigeyoshi@sfsu.edu - Team Lead
Aaron Rayray - arayray@sfsu.edu - Github Master
Noah Hai - nhai@sfsu.edu - Doc Editor
Shez Rahman - srahman2@sfsu.edu - Backend Lead
Ghadeer Al-Badani - galbadani@sfsu.edu - Backend
Majd Alnajjar - malshemari@sfsu.edu - Frontend
William Pan - wpan1@sfsu.edu - Database Admin
Phillip Ma - pma1@mail.sfsu.edu - Frontend Lead

“Milestone 2“

6/26/2024

History Table:

Date	Changes
7-22-24	Made corrections to Data definitions, database requirements

1. Data Definitions

1. User : Class - The user that creates the account.
    - 1.1. userID : Number - Primary key
    - 1.2. firstName : String
    - 1.3. lastName : String
    - 1.4. userName : String
    - 1.5. membershipType : String-Symbol (FREE, PREMIUM, MENTOR)
    - 1.6. email : String
    - 1.7. password : Number
    - 1.8. salt : Number - random data for hashing
    - 1.9. emailVerified : Boolean
    - 1.10. resetPasswordToken: String
    - 1.11. resetPasswordExpires : Date
    - 1.12. points : Number
    - 1.13. rankID : Number - Rank object
    - 1.14. challengesCompleted : Number - Array of Numbers(completed challenge IDs)
    - 1.15. numChallengesCompleted : Number
    - 1.16. allTrophies : Number- Array of SpecificTrophy objects
    - 1.17. Streak of challenges completed : Number
    - 1.18. coins : Number
    - 1.19. mentees : Json - Array of userIDs
    - 1.20. notificationList :Json - Array of Notification objects
    - 1.21. bookmarks : String - Array of postIDs
    - 1.22. numPosts : Number - number of posts + number of comments
    - 1.23. groups : Json - Array of Group objects
    - 1.24. groupsMentored : Json - Array of MentorGroup objects
    - 1.25. isPremium : boolean
    - 1.26. isMentor : boolean

2. PremiumUser : Class, extends User : A user that has paid the subscription fee for premium features
    - 2.1. userID : Number - Primary and foreign key

3. MentorUser : Class, extends User - A user that has become a mentor to other users. Must be approved by interview.
    - 3.1. userID : Number - Primary and foreign key
    - 3.2. feedbackCompleted: String- Array of FeedbackForm objects

4. Profile : The user's profile which has information about the user.
    4.1. profileID: Number - Primary key
    4.2. userID: Number - foreign key (the user who owns the profile)
    4.3. isMentor : boolean
    4.4. biography : String
    4.5. resume : String
    4.6. portfolioID : Number

5. ExternalLinks : Class - contains external links to be used by profile
    5.1. externalLinksID: Number - Primary key
    5.2. profileID : Number - foreign key
    5.3. xLogo: String - filepath to image
    5.4. linkedinLogo: String - filepath to image
    5.5. instagramLogo: String - filepath to image
    5.6. tiktokLogo: String - filepath to image
    5.7. facebookLogo: String - filepath to image
    5.8. xLink: String
    5.9. linkedinLink: String
    5.10. instagramLink: String
    5.11. tiktokLink: String
    5.12. facebookLink: String
    5.13. hasX: Boolean
    5.14. hasLinkedin: Boolean
    5.15. hasInstagram: Boolean
    5.16. hasTiktok: Boolean
    5.17. hasFacebook: Boolean

6. Portfolio: Class - Users will have this to display their projects
    6.1. portfolioID : Number - Primary key
    6.2. userID : Number - foreign key (user that create portfolio)
    6.3. visibility: Symbol (public or private)

7. Project : Class - these are data items to be stored in portfolio
    7.1. projectID : Number - Primary key
    7.2. Portfolio : Number - Foreign key
    7.3. link: String - link to project
    7.4. desc: String - text description of project
    7.5. title: String - title of project
    7.6. pictures: String - Array of file paths to images

8. Notification: Class
    8.1. notificationID : Number - primary key
    8.2. title: String - title of notification with template
    8.3. redirectLink: String - clicking notification takes you to link
    8.4. date: Date
    8.5. time: Time

9. UserNotification: A junction table(Associative entity) for notification and user Since a user has many notification and a notification belong to many user
    9.1. userNotificationID: Number - primary key
    9.2. userID: Number - foreign key
    9.3. notificationID: Number - foreign key

10. Group : Class - users can join these to bond over commonalities such as being alumni from the same school, or having interest in a certain technology
    10.1. GroupsID: Number - primary key
    10.2. allMembers: String - Array of User objects who have access
    10.3. forum: String - Forum object
    10.4. groupsID : Number - foreign key

11. UserGroup: A junction table(Associative entity) for group and user
    Since a mentorUser can join many MentorGroup and A MentorGroup can have many mentorUser
    11.1. UserGroupID: Number - primary key
    11.2. groupID: Number - foreign key reference group
    11.3. userID: Number - foreign key reference User

12. MentorGroup : Class extends Group - a group for mentors to communicate with their mentees
    12.1. groupID: Number - primary key
    12.2. mentorMembers: String - Array of User objects
    12.3. groupsID : Integer - foreign key

13. UserMentorGroup: A junction table(Associative entity) for group and user
    Since a mentorUser can join many MentorGroup and A MentorGroup can have many mentorUser
    13.1. UserMentorGroupID: Number - primary key
    13.2. groupID: Number - foreign key reference Mentorgroup
    13.3. userID: Number - foreign key reference mentorUser

14. Groups : Class - a list of all groups for access
    14.1. GrouopsID: Number - primary key
    14.2. mentorGroups : String - Array of MentorGroup objects
    14.3. groups:String - Array of Group objects

15. Post : Class - Posts that users can create in order to interact with the larger community.
    15.1. postID: Number - primary key
    15.2. userID: Number - foreign key
    15.3. content : String - the text content
    15.4. comments: String - Array of comments/replies to this post
    15.5. codeBlock: String
    15.6. date: Date
    15.7. time: Time
    15.8. likes: Number - number of likes on the post
    15.9. threadID : foreign key

16. Forum : Class - A collection of posts and data about the forum
    16.1. forumID: Number - primary key
    16.2. threadID : Number - foreign key
    16.3. threadTitle: String - Title of the forum thread
    16.4. date: Date - object
    16.5. time: Time - object
    16.6. access : String - List of members who have access
    16.7. threads: String - Array of ForumThread objects

17. ForumThread: Class - used to contain all posts under a thread topic
    17.1. threadID: Number - label each thread in unique identifier
    17.2. originalPoster: String - User object who started the thread
    17.3. threadTitle: String - title of the forum thread
    17.4. posts: Array of Post objects which make up the thread
    17.5. date: Date - object
    17.6. time: Time - object
    17.7. forumID: Number - foreign key

18. CodeChallenge : Class - The coding challenges that each user has access to and can attempt to solve.
    18.1. challengeID : Number - primary key
    18.2. title: String

18.3. description: String
18.4. language: String
18.5. difficulty: String
18.6. codingBlock: String
18.7. deadline: Date - object
18.8. completionPoints: Number - points gained for completing this challenge
18.9. solutions: String - array of ChallengeSubmission objects - record of X successful solutions
18.10. codingTests: String
18.11. pseudocodeHint: String

19. UserChallenge: A junction table(Associative entity) for user and codeChallenge. Since a user has many codeChallenge and A codeChallenge belong to many user
19.1. userChallengeID : Number - primary key
19.2. userID: Number - foreign key
19.3. challengeID: Number - foreign key

20. ChallengeSubmission: Class - contains submission data
20.1. challengeSubID:Number -primary key
20.2. userID: Number - foreign key
20.3. challengeID: Number - foreign key
20.4. codSub: string- the code is submitted
20.5. date: Date - object
20.6. time: Time - object
20.7. verifiedSolution: Boolean - true if the submission was successful

21. SubmissionShare : Class (FR 1.18) - unique class to share with peers when you complete a challenge
21.1. shareID : Number - primary key
21.2. userID : Number - foreign key
21.3. likes: Number
21.4. Comments : String

22. Feedback : Class - The review form that mentors use to give feedback to mentees on their coding challenge solutions.
22.1. feedbackID : Number - primary key
22.2. userID : Number - foreign key reference mentorUser
22.3. challengeSubID : Number - foreign key reference challengeSubmission
22.4. solutionSubmission: String - ChallengeSubmission object the feedback is in response to

22.5. organizationFeedback: String - mentor's written feedback on formatting/organization of code
22.6. organizationScore: Number - mentor's scoring on scale of 1-5
22.7. logicFeedback: String - mentor's written feedback on code logic and efficiency
22.8. logicScore: Number - mentor's scoring on scale of 1-5
22.9. commentFeedback: String - mentor's written feedback on code comments
22.10. commentScore: String - mentor's scoring on scale of 1-5

23. Leaderboard : Class - list of all users organized by ranking points to be displayed as a table
    23.1. leaderboardID: Number - primary key
    23.2. userID: Number - foreign key
    23.3. numPoints : Number - Each user's number of points held
    23.4. rankTitle: String - Each user's rank title
    23.5. rankIcon: String - corresponding icon for user ranking

24. Rank : Class - the different icons and titles that users can acquire as they gain more points
    24.1. icon : String - path to image file
    24.2. title : Symbol
    24.3. rankID: Number - primary key
    24.4. ranksID: Number -foreign key
    24.5. pointsRange: Number - function returning true if User collected points fall in the range
25. Ranks : Class - list of different rank objects that users can acquire as they gain more points, and functions involving the ranks
    25.1. ranksID : Number - primary key
    25.2. rankID : Number - foreign key
    25.3. checkRequirements: String - function to check what rank User has and return the correct rank

26. Trophy : Class - inherited by SpecificTrophy class. Users earn these for specific achievements
    26.1. name: String
    26.2. trophyID: Number - primary key
    26.3. description: String - describes requirements to earn trophy
    26.4. Trophy flag : Boolean - true if user possesses trophy
    26.5. userID: Number - foreign key

26.6. checkRequirements : String- abstract method, implemented by SpecificTrophy

27. SpecificTrophy: Class extends Trophy - users can earn trophies for different achievements
    27.1. trophyID: Number - primary key
    27.2. hasTrophy : Boolean - used by User, is 1 if checkRequirements returns
    checkRequirements : function to be implemented

28. JobListing: Class - Job listings that are available for any user to apply for.
    28.1. userID: Number - foreign key
    28.2. jobID: Number - primary key
    28.3. title: String
    28.4. company: String
    28.5. location: String
    28.6. description: String
    28.7. requirements: String
    28.8. date: Date - object
    28.9. applicationLink: String

29. PaymentInfo: Class - Users will be able to store their payment information for purchases
    29.1. paymentID: Number- primary key
    29.2. cardNumber: Number
    29.3. cardName: String
    29.4. zipCode: Number
    29.5. backCode: Number - CVV/CVC code
30. UserPayment: A junction table(Associative entity) for payment and user
    Since a user has many paymentInfo and a paymentInfo belong to many user
    30.1. userPaymentID: Number - primary key
    30.2. userID: Number - foreign key
    30.3. paymentID: Number - foreign key

31. Message : Class - Sent between users as direct messaging
    31.1. MessageID: Number -primary key
    31.2. sendingUser: Number - who's sending the message (foreign key)
    31.3. receivingUsers: Number - who's receiving the message(foreign key)
    31.4. time: Time- object
    31.5. date: Date - object
    31.6. content: String - content of the message

32. MessageThread: Class - includes all past replies to a single message thread
    32.1. messageThreadID : Number - primary key
    32.2. messageID: Number -foreign key
    32.3. participatingUsers: String - Array of User objects who are in the message thread
33. Inbox : Class - Every user contains an inbox of messages that other users have sent them
    33.1. inboxID: Number - primary key
    33.2. userID: Number - foreign key
    33.3. messageThreads:String - array of MessageThread objects
34. SupportForm : Class - All users can use these to communicate with the company
    34.1. supportFormID : Number - primary key
    34.2. from_userID : Number - foreign key reference user
    34.3. to_userID : Number - foreign key reference userHiring
    34.4. date: Date - object
    34.5. time: Date - object
    34.6. message: String - populated by user from UI
35. userHiring: Class extends User - A company user that hiring user
    35.1. userID : Number : primary key
    35.2. company : String - the company that user work for
    35.3. [1]position : String - The position of user is in company
36. chatSession : Meeting rooms which users can use to meet with other users (free or premium). (Assuming this relates to workspace idea - what separates this from an inbox thread with multiple recipients?)
    36.1. chatSessionID: Number - primary key
    36.2. User ID : Number - foreign key
    36.3. date : Date - object
    36.4. title : String
    36.5. invitees : Json - list of user_ID's
37. UserChatSession: A junction table(Associative entity) for user and chatSession Since a user has many codeChallenge and A codeChallenge belong to many user
    37.1. userChatSessionID: Number - primary key
    37.2. userID: Number - foreign key
    37.3. chatSessionID: Number - foreign key
38. Chatbot : alternative to support form for users to communicate with an AI rep for the company
    38.1. Chatbot ID: Number - primary key
    38.2. User ID : Number - foreign key reference userHiring

---

1

2. Prioritized Functional Requirements

Priority 1:

1. **All Users:**

*1.1 Users shall be able to explore some portions of the product without a profile
*1.2 Users shall be able to solve an example problem.
*1.3 Users shall create a profile
*1.4 Users shall be able to Log in/Log out (only with created profile)
*1.5 Users shall be able to use a SSO login for companies/schools
*1.6 Users shall be able to upload profile picture
*1.7 Users shall be able to Delete profile
*1.8 User shall be able to update payment information
*1.9 Users shall be able to make their profile private or public
*1.12 Users shall be able to check other users profiles/stats
*1.13 Users shall be able to do coding challenges
*1.15 Users shall be able to award different profile trophies for achievements
*1.17 Users shall be able to earn points for coding challenges
*1.18 Users shall be able to like/comment on challenge posts

*1.20 Users shall be able to check their coding ranking
*1.21 Users shall be able to check leaderboards
*1.22 Users shall be able to subscribe to Premium
*1.23 Users shall be able to unsubscribe to Premium
*1.32 Users shall be able to connect other socials
*1.34 Users shall be able to request additional features from the dev team
*1.35 Users shall be able to utilize live chat w/ other users
*1.36 Users shall be able to direct message other users
***1.38 Users shall be able to check coding streak counter of daily challenges**
*1.39 Users shall be able to create a portfolio
*1.40 Users shall be able to check/update portfolio
*1.41 Users shall be able to change portfolio visibility (public/private)
*1.42 Users shall be able to access portfolio review
*1.45 Users shall be able to submit support forms to submit feedback regarding the app
*1.55 Users shall be able to link account to other socials(github,linkedin,etc)
*1.61 Users shall be able to use an IDE without having to create an account
*1.64 Users shall be able to choose their own country/region
*1.65 Users without premium shall be able to view three solutions per month
*1.66 Users shall be able to see the difference between premium and free membership perks
*1.67 Users shall be able to gain points by starting forum threads
*1.68 Users shall be able to gain points by commenting in threads

*1.69 Users shall be able to gain points by gaining friends
*1.70 Users shall be able to gain points by gaining mentees
*1.71 Users shall be able to gain points by gaining mentors
*1.72 Users shall be able to post text and images to their profiles
*1.73 Users shall be able to view a general feed of other users' updates and activities
*1.76 Users shall be able to search for other users
*1.77 Users shall be able to search for groups
*1.78 Users shall be able to receive notifications for new coding challenges.
*1.78 Users shall be able to receive notifications for messages and comments.
*1.79 Users shall be able to customize notification preferences.
*1.81 **Users shall be able to save forum posts for later reading.**
*1.82 Users shall be able to report inappropriate content.
*1.83 Users shall be able to block or mute other users.
*1.84 Users shall be able to set privacy settings for profile visibility.
*1.85 **Users shall be able to view the history of coding challenges attempted.**
*1.86 Users shall be able to request a mentor recommendation.
*1.90 Users shall be able to subscribe to notifications for specific forums or groups
*1.93 Users shall be able to view a scrolling list of job listings from home page
*1.94 Users shall be able to view other users' activity from their profiles

2. **Free Users**
   *2.2 Free Users shall be able to check code challenge repo
   *2.3 Free users shall be able to view three pseudocode hints per month

3. **Premium Users**
   *3.0 Premium Users shall be able to check a single system chosen solution after completing a challenge
   *3.2 Premium Users shall be able to request mentorship (premium)
   *3.3 Premium Users shall be able to Match mentors with similar coding language experience
   *3.4 Premium users shall be able to view one pseudocode hint per challenge
   *3.7 Premium Users shall be able to request code review from a mentor
4. **Mentor Users**
   *4.1 Mentor Users shall be able to review code solutions of other users they are mentoring.
   *4.4 Mentor Users shall be able to review Free/Premium users resumes/portfolios
   *4.5 Mentor users shall be able to complete challenge feedback on coding challenges completed by mentees
   *4.6 Mentor users shall be able to gain points by completing challenge feedback on mentee solutions
   *4.7 Mentor Users shall be able to upload videos

*4.8 Mentor users shall have their number of mentees displayed on their profiles
*4.9 Mentor users shall have their number of solutions reviewed displayed on their profiles
*4.10 Mentor users shall have the groups they lead displayed on their profiles

Priority 2:

1. **All Users:**
   **1.11 Users shall be able to follow other users/mentors
   **1.16 Users shall be able to allow switching coding languages for challenges (cross compatibility) (different compilers - maybe 2 or 3 priority here)(would prefer just different sets of challenges based on language)
   **1.24 Users shall be able to see pop-up ads for premium/paid utilities
   **1.48 Users shall be able to take mock interviews (practice interviews with real-time communication and feedback from peers and other users)
   **1.49 Users shall be able to view featured users (spotlight user that's completed most challenges/stayed the most active for the past month. Featured user gets changed monthly)
   **1.51 Users shall be able to utilize discounts/offers on apps premium features
   **1.54 Users shall be able to join group session workshops led by mentors
   **1.56 Users shall be able to view/update their own calendar(scheduled hackathons, virtual meetings..) (API or creating our own calendar?)
   **1.59 Users shall be able to utilize a button to change the screen to dark mode.
   **1.60 Users shall be able to get assessed to become a mentor
   **1.63 Users shall be able to choose their preferred speaking language(How does that look in implementation?)
   1.74 **Users shall be able to earn coins
   1.75 **Users shall be able to trade coins for premium subscriptions
   -Or top leaderboard users get a premium reward
   1.80 **Users shall be able to bookmark coding challenges.
   **1.87 Users shall be able to participate in live coding sessions.
   1.91 **Users shall be able to access analytics on their coding performance.
   ****1.92 Users shall be able to receive notifications for new coding challenges.**

2. **Free Users:**
   ****2.1 Free Users shall be able to check the most average solutions after completing a challenge.**

3. **Premium Users:**

****3.1 Premium Users shall be able to check a system chosen set of 2-3 solutions after completing a challenge**

4. **Mentor Users:**
   **4.3 Mentor Users shall be able to create coding challenges.

Priority 3:

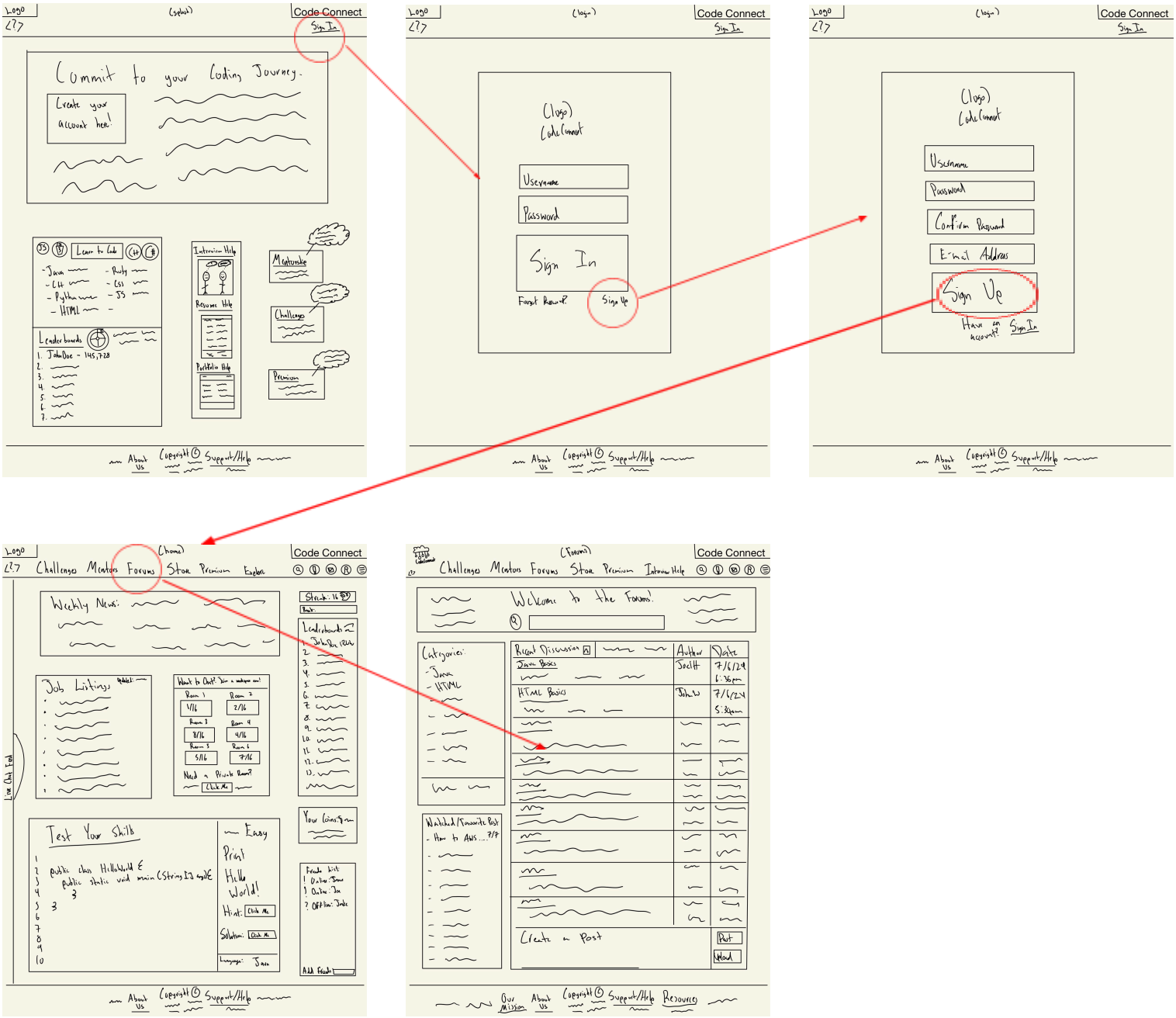
1. **All Users:**
   ***1.14 Users shall be able to see popular submissions (how is popular measured?)
   ***1.25 Users shall be able to application survey for mentorship (?)
   ***1.28 Users shall be able to take hiring questionnaire challenges (which are siblings to coding challenges
   ***1.37 Users shall be able to access free/paid resources (videos/textbooks)
   ***1.43 Users shall be able to utilize chatbot (this function would be cool but bold because need to know the data items) (Use support form instead)
   ***1.44 Users shall be able to access virtual workspace (define virtual workspace?)
   ***1.46 Users shall be able to open source/collaborative coding projects (users can participate in other people's coding projects) (Similar to 1.44?)
   ***1.47 Users shall be able to read weekly digests (recommendations created for users based on interests/what they worked on)
   ***1.50 Users shall be able to utilize pair programming sessions (allows users to work on projects/code collaboratively) (similar to 1.44 and 1.46)
   ***1.52 Users shall be able to read technical news (similar to 1.47)
   ***1.57 Users shall be able to create Recruiter/Hiring Manager specific profile
   ***1.58 Users shall be able to pass a Recruiter/Hiring manager verification/background check
   ***1.88 Users shall be able to join virtual coding bootcamps.
   ***1.89 Users shall be able to create and manage a personal blog.

2. **Free Users: N/A**
3. **Premium Users:**
   ***3.5 Premium Users shall be able to create code challenge repo, which are reviewed by the dev team prior to publication. (with a full repo it could be harder to check solutions vs a simple IDE plug in/compiler. Resources may be difficult on this topic…maybe talk to prof)
   ***3.6 Premium Users shall be able to check/update code challenge repo(^see note for 3.5)
4. **Mentor Users:**

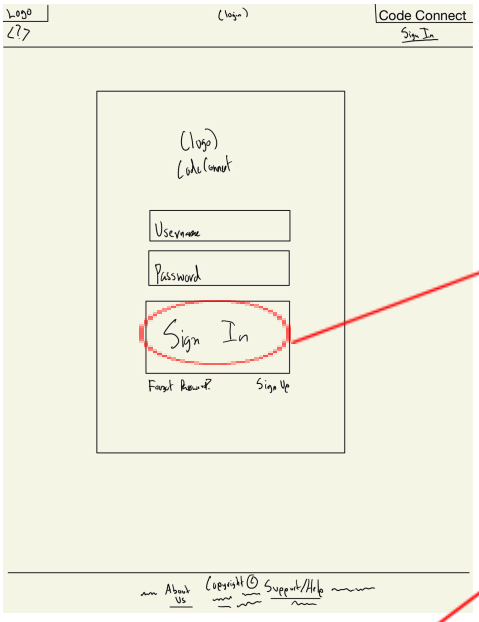
***4.2 Mentor Users shall be able to set up group work stations. (what is a group work station? Also potentially lower priority)

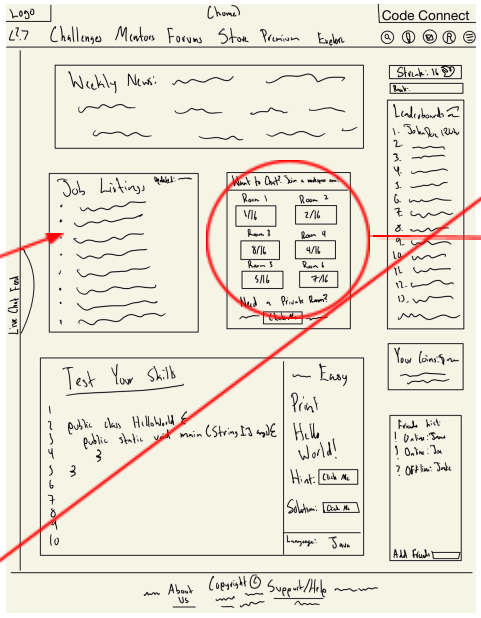
3. UI Mockups and Storyboards (high level only)

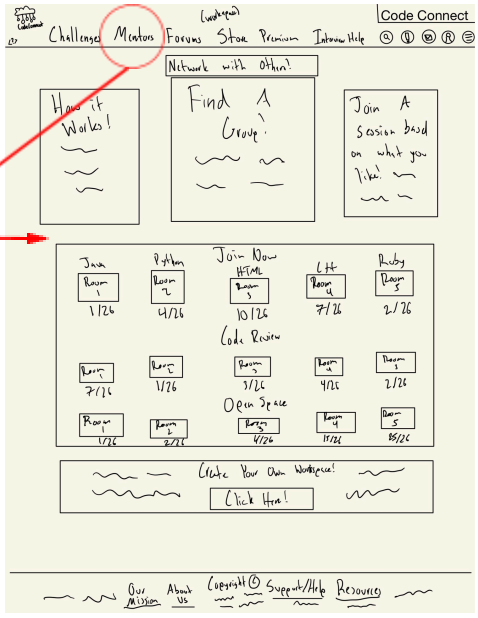
Use Case 1: Toby (Unregistered User) enters the forums

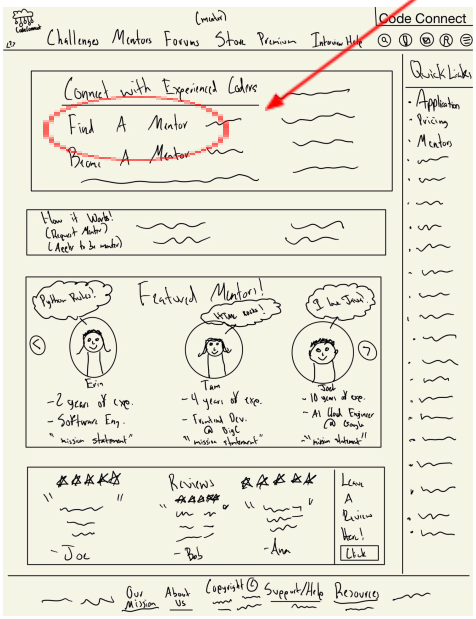


Use Case 2: Harold (Registered User) Attends a group session, meets with a mentor

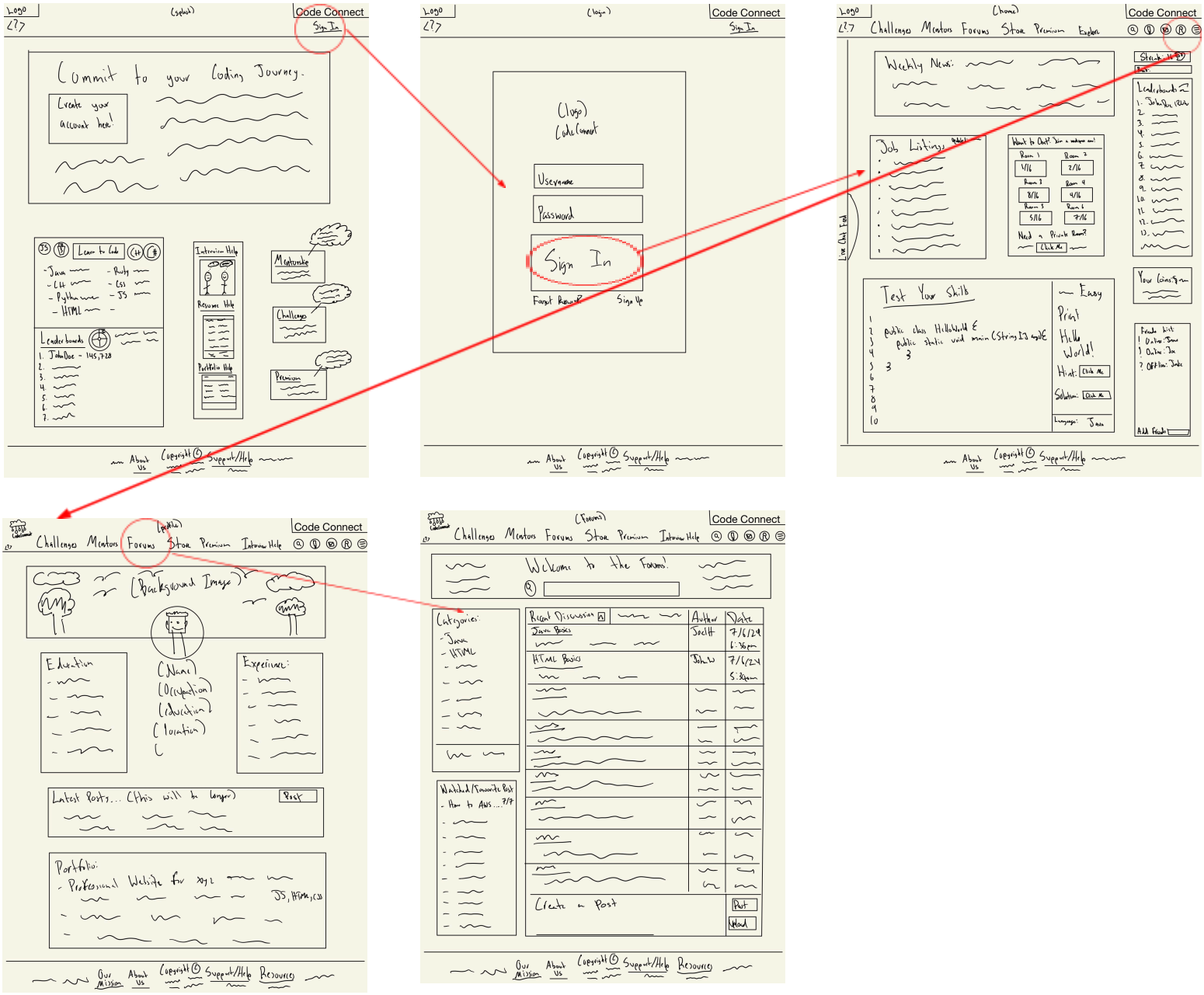




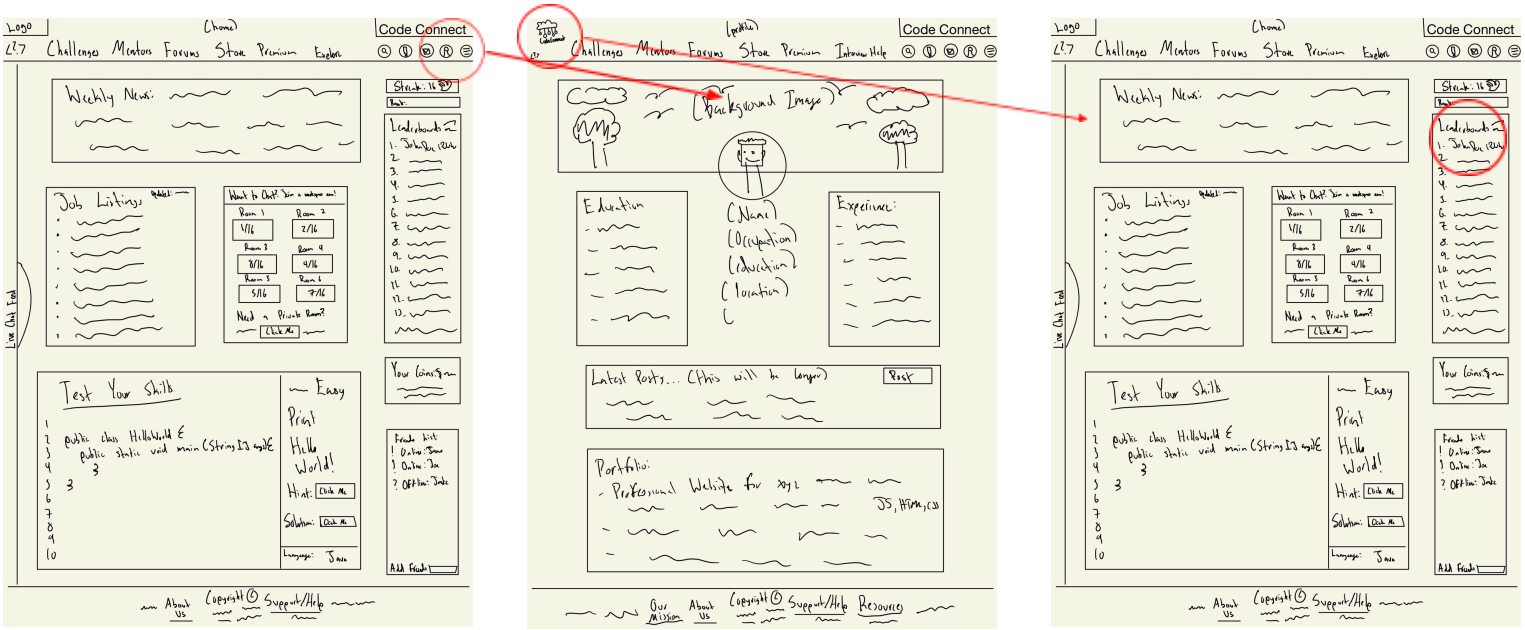




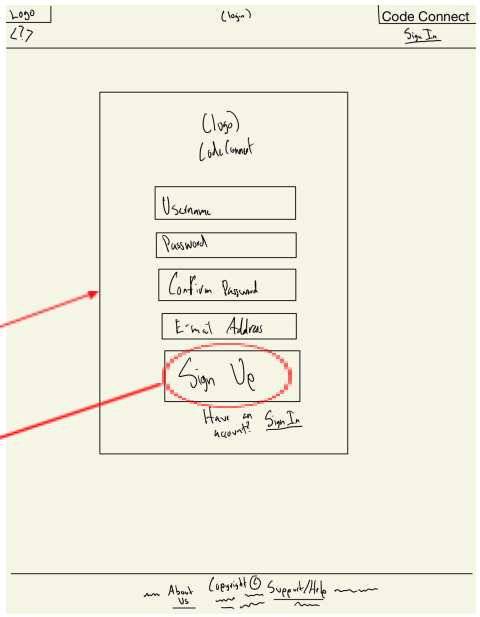
Use Case 3: Josh (Registered User) Changes profile and goes to forums

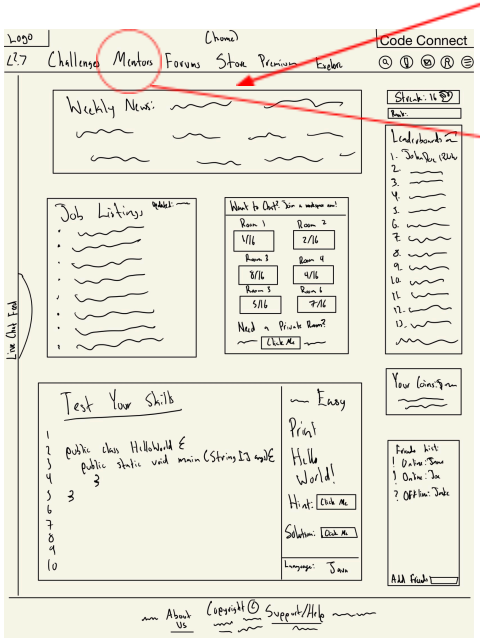


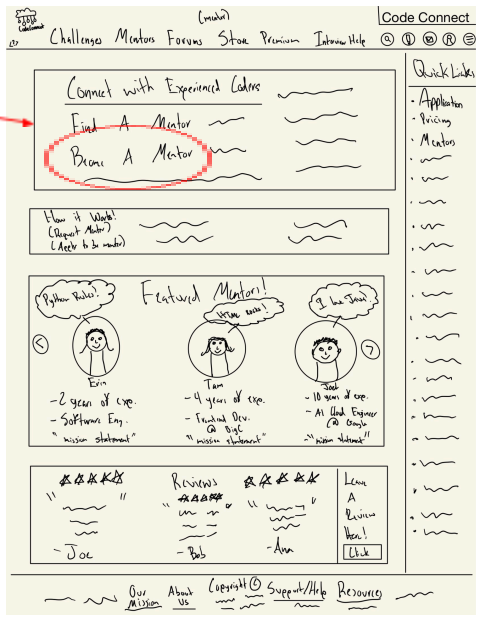
Use Case 4: Caroline (Registered User) Changes Profile and looks at rankings



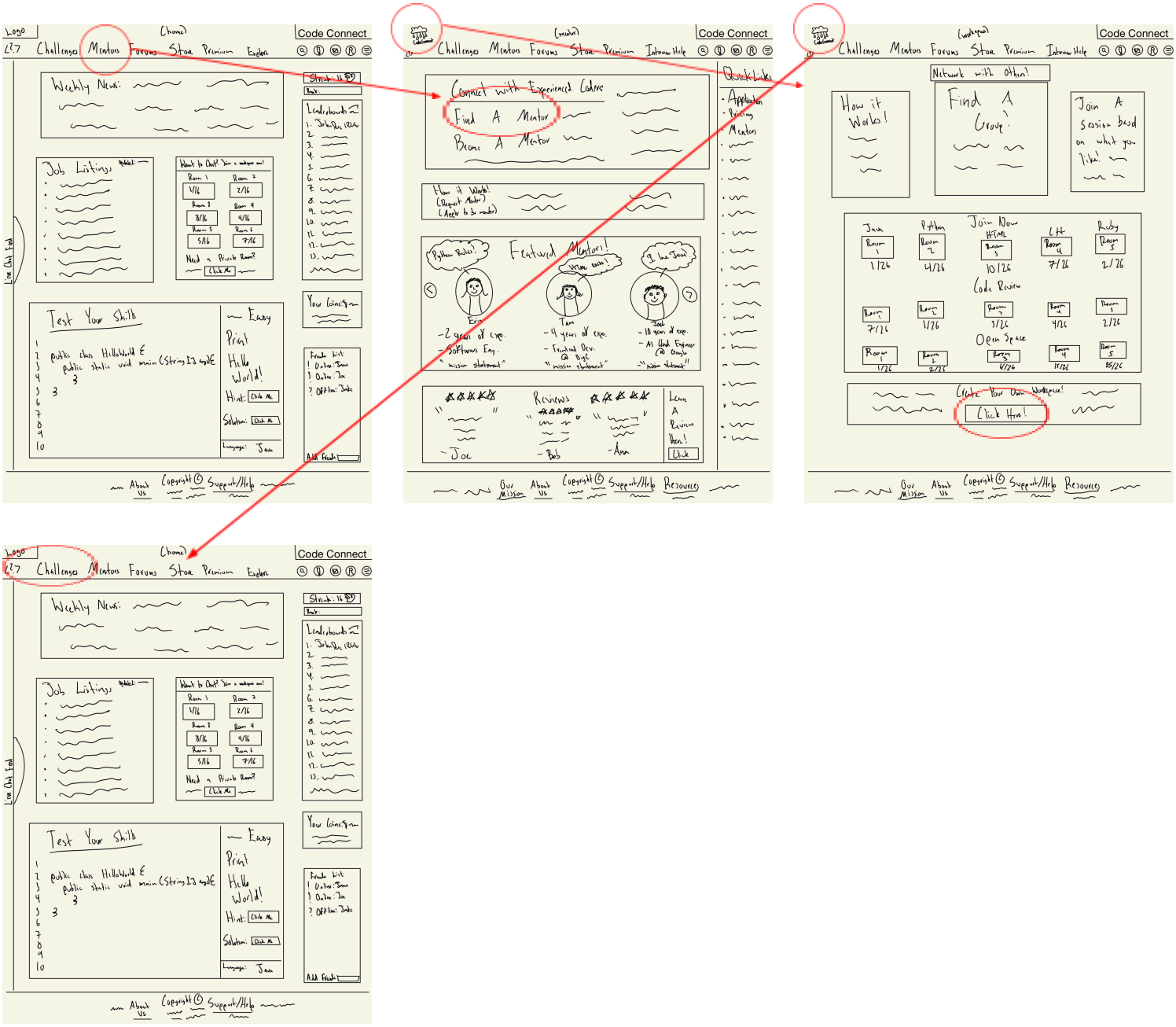
Use Case 5: Heide (Unregistered User) Signs up to be a Mentor



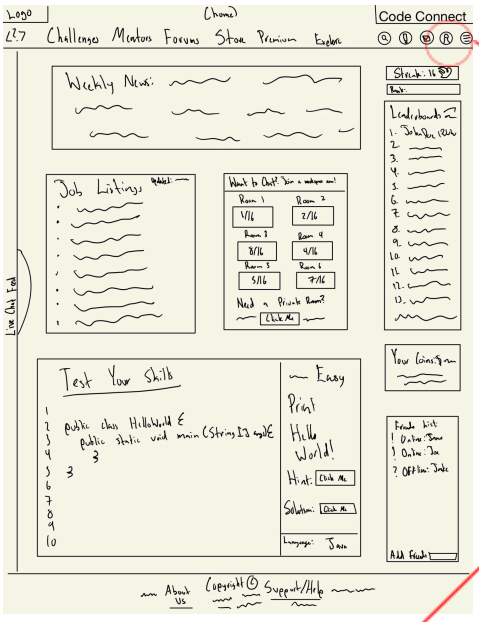


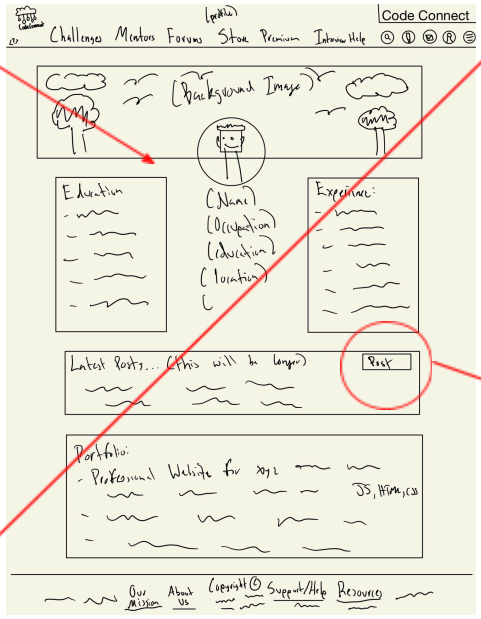


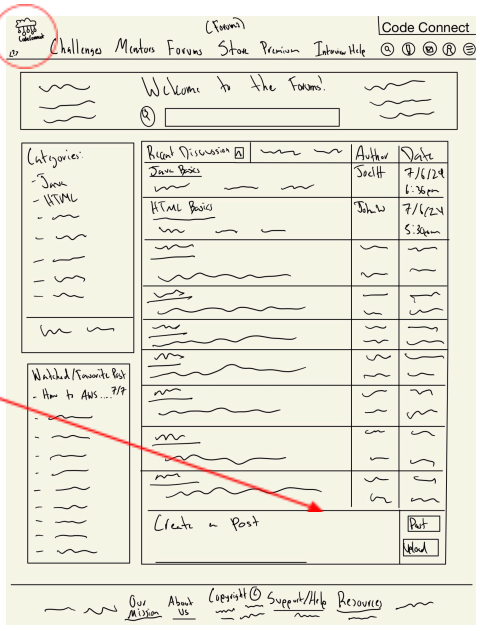
Use Case 6: Jeffrey (Registered User) Finds a mentor, schedules a meeting, and completes the challenges

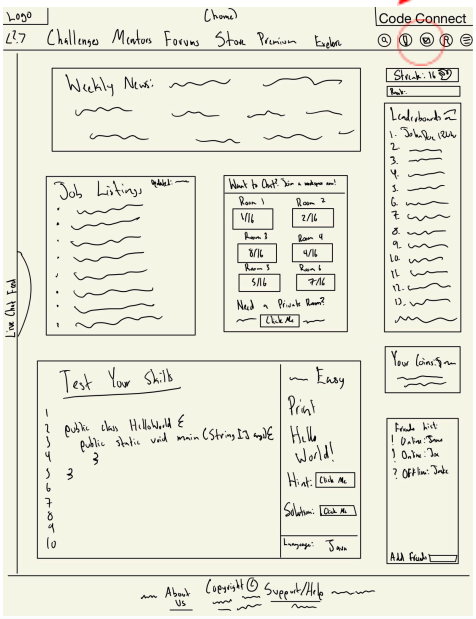


Use Case 7: Tiffany (Registered) Updates profile, Goes into forums, contacts people directly

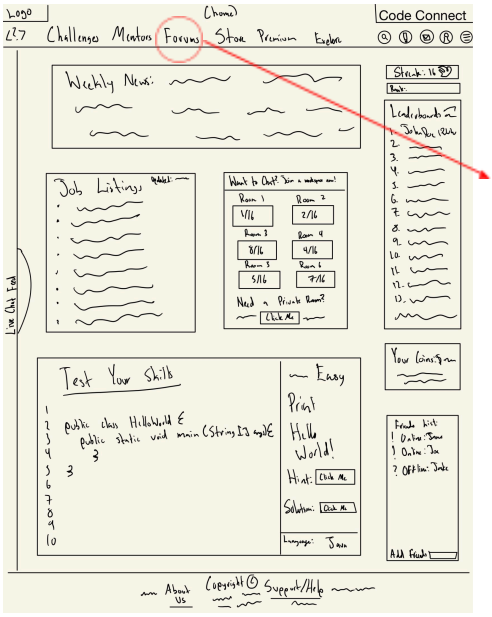


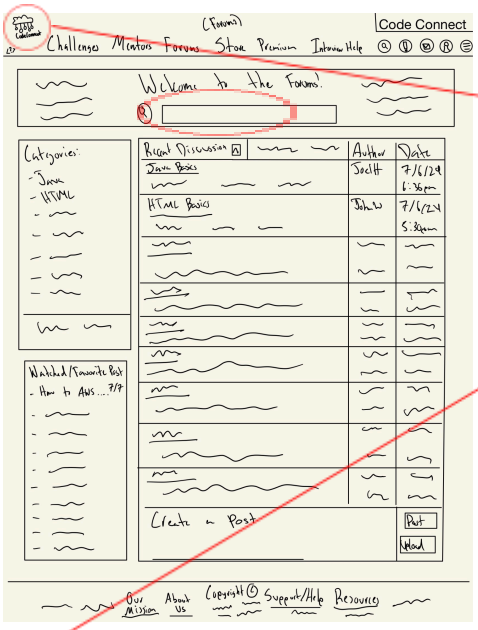


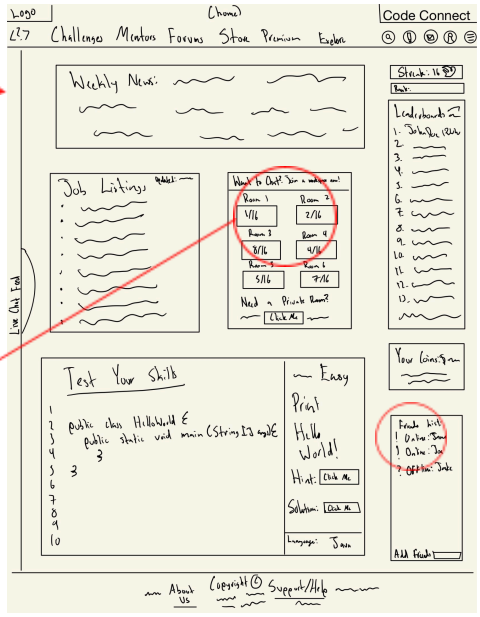


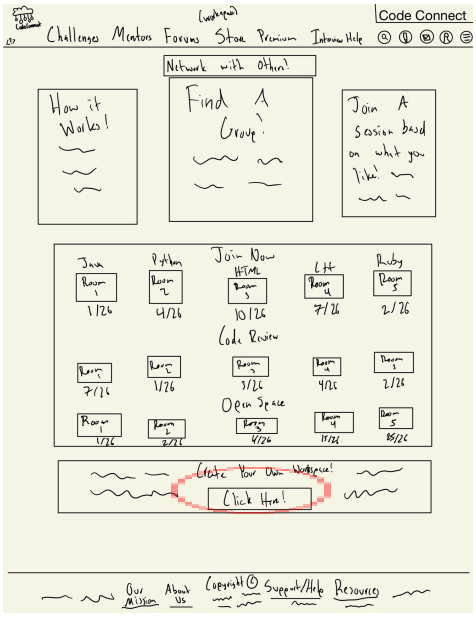


Use Case 8. Joakim (Registered User) Joins forums, connects with users in a group









4. High level database architecture and organization

1. **Database Requirements**:
    1.1. **First Iteration**: Outline the database requirements derived from functional and non-functional requirements.
        1) User
            a) A user shall receives many notice
            b) A user shall see one leaderboard
            c) A user shall create one profile
            d) A user shall create one portfolio
            e) A user shall able to view many job
            f) A user shall able to join many meeting room
            g) A user shall able to create many post
            h) A user shall able to have many trophy
            i) A user shall able to send message to many user
            j) A user homepage shall show one or zero inbox
            k) A user shall able to solve many coding Challenge
            l) A user shall able to submit many coding challenge submission
            m) A user shall be able to share completed challenge on many posts
            n) A user could be one Premium user
            o) A user shall able to join many group
            p) A user shall have one rank
            q) A user shall have one payment
            r) A user shall become one userHiring
            s) A user shall able to write many supportForm
            t) A user shall able to use one chatbot
        2) Premium User
            a) A premium user are one and only one user
            b) A premium user can become one mentor user
        3) mentor
            a) A mentor user shall have one and only one premium user
            b) A mentor shall be able to give many feedback
            c) A mentor shall join many mentor group to communicate with their mentee
            d) A mentor shall join many mentorGroup
        4) profile
            a) A profile have one and only one user
            b) A profile shall have one externalLinks
            c) A profile shall have one portfolio
        5) External link
            a) A externalLinks shall have one and only one profile
        6) portfolio
            a) A portfolio shall have one and only one user
            b) A portfolio shall have many project

c) A portfolio have one and only one profile

7) project
    a) A project have one and only one portfolio

8) notification
    a) A notice have one or many user

9) group
    a) A group has many user
    b) A group shall have many mentorGroup
    c) A group has one and only one groups
    d) A mentorGroup has one and only one group

10) mentor group
    a) A mentorGroup has many mentor
    b) A mentorGroup has one and only one group
    c) A mentorGroup has one and only one groups
    d) A mentorGroup has one or many mentor

11) groups
    a) A groups has one or many group
    b) A groups has one or many mentorGroup

12) post
    a) A post have one and only one user
    b) A post has one and only one forum Thread

13) forumThread
    a) A forum Thread shall contain many posts

14) forum
    a) A forum shall contain many forumThread
    b) A forum Thread has one and only one forum

15) coding Challenge
    a) A coding challenge have many users
    b) A coding challenges shall contain many challenge submission

16) challenge submission (challengeSub)
    a) A challenge submission have one and only one user
    b) A challenge submission should receive many feedback
    c) A challenge submission has one and only one coding challenge
    d)

17) submissionShare
    a) A share completed challenge should have one and only one use

18) feedback
    a) A feedback should has one and only one mentor
    b) A feedback shall belong to one and only one challenge submission

19) leaderBoard
    a) A leaderboard have many users

20) rank
    a) A rank have one and only one user
    b) A rank has one ranks

21) ranks
    a) A ranks shall acquire many ranks

22) trophy
    a) A trophy have one and only one user
    b) A trophy shall have many specific trophy
23) SpecificTrophy
    a) A SpecificTrophy shall have one and only one trophy
24) jobList
    a) A job have one or many user
25) paymentInfo
    a) A payment has one or many users
26) message
    a) A message have one and only one user (Not support one message to multiple users)
    b) A message have many inbox
    c) A message has one thread.
27) MessageThread
    a) A message thread shall contain message
28) inbox
    a) A inbox have one and only one user
    b) A inbox shall have contain many message
29) supportForm
    a) A supportForm has one and only one user
    b) A supportForm has one and only one chatbot
30) userHiring (company)
    a) A userHiring is one or many user
    b) A userHiring are able to revives many support form
    c) A userHiring has one chatbot
31) mentor
    a) A mentor shall join many mentor group to communicate with their mentee
    b) A mentor shall join many mentorGroup
32) meeting
    a) A meeting room have many users
33) chatbot
    a) A chatbot shall able to support one and only one userHiring(company)
    b) A chatbot shall reply to one and only one user.

1.2. **DBMS Selection**: Briefly (in one or two sentences) explain the chosen DBMS (Database Management System) or associated SQL frameworks and justify why they are the best fit for this project..

Since we use Amazon Ec2 instance for our server, we also use Amazon RDS for mysql database by installing mariadb which will be easy to connect and maintain either by mysql workbench or command line from Ee2 instance.

2. **Database Organization**:

2.1. **Entities and Relationships**: Describe your entities, their attributes, relationships, and domains at a high level.

1) User : Class - The user that creates the account.
    a) userID : Number - Primary key
    b) firstName : String
    c) lastName : String
    d) userName : String
    e) membershipType : String-Symbol (FREE, PREMIUM, MENTOR)
    f) email : String
    g) password : Number
    h) salt : Number - random data for hashing
    i) emailVerified : Boolean
    j) resetPasswordToken: String
    k) resetPasswordExpires : Date
    l) points : Number
    m) rankID : Number - Rank object
    n) challengesCompleted : Number - Array of Numbers(completed challenge IDs)
    o) numChallengesCompleted : Number
    p) allTrophies : Number- Array of SpecificTrophy objects
    q) Streak of challenges completed : Number
    r) coins : Number
    s) mentees : Json - Array of userIDs
    t) notificationList :Json - Array of Notification objects
    u) bookmarks : String - Array of postIDs
    v) numPosts : Number - number of posts + number of comments
    w) groups : Json - Array of Group objects
    x) groupsMentored : Json - Array of MentorGroup objects
    y) isPremium : boolean

z) isMentor : boolean

2) PremiumUser : Class, extends User : A user that has paid the subscription fee for premium features
    a) userID : Number - Primary and foreign key

3) MentorUser : Class, extends User - A user that has become a mentor to other users. Must be approved by interview.
    a) userID : Number - Primary and foreign key
    b) feedbackCompleted: String- Array of FeedbackForm objects

4) Profile : The user's profile which has information about the user.
    a) profileID: Number - Primary key
    b) userID: Number - foreign key (the user who owns the profile)
    c) isMentor : boolean
    d) biography : String
    e) resume : String
    f) portfolioID : Number

5) ExternalLinks : Class - contains external links to be used by profile
    a) externalLinksID: Number - Primary key
    b) profileID : Number - foreign key
    c) xLogo: String - filepath to image
    d) linkedinLogo: String - filepath to image
    e) instagramLogo: String - filepath to image
    f) tiktokLogo: String - filepath to image
    g) facebookLogo: String - filepath to image
    h) xLink: String
    i) linkedinLink: String
    j) instagramLink: String
    k) tiktokLink: String
    l) facebookLink: String
    m) hasX: Boolean
    n) hasLinkedin: Boolean
    o) hasInstagram: Boolean
    p) hasTiktok: Boolean
    q) hasFacebook: Boolean

6) Portfolio: Class - Users will have this to display their projects
    a) portfolioID : Number - Primary key

b) userID : Number - foreign key (user that create portfolio)
c) visibility: Symbol (public or private)

7) Project : Class - these are data items to be stored in portfolio
    a) projectID : Number - Primary key
    b) Portfolio : Number - Foreign key
    c) link: String - link to project
    d) desc: String - text description of project
    e) title: String - title of project
    f) pictures: String - Array of file paths to images

8) Notification: Class
    a) notificationID : Number - primary key
    b) title: String - title of notification with template
    c) redirectLink: String - clicking notification takes you to link
    d) date: Date
    e) time: Time

9) UserNotification: A junction table(Associative entity) for notification and user

Since a user has many notification and a notification belong to many user

    a) userNotificationID: Number - primary key
    b) userID: Number - foreign key
    c) notificationID: Number - foreign key

10) Group : Class - users can join these to bond over commonalities such as being alumni from the same school, or having interest in a certain technology
    a) GroupsID: Number - primary key
    b) allMembers: String - Array of User objects who have access
    c) forum: String - Forum object
    d) groupsID : Number - foreign key

11) UserGroup: A junction table(Associative entity) for group and user

Since a mentorUser can join many MentorGroup and A MentorGroup can have many mentorUser

    a) UserGroupID: Number - primary key
    b) groupID: Number - foreign key reference group
    c) userID: Number - foreign key reference User

12) MentorGroup : Class extends Group - a group for mentors to communicate with their mentees
    a) groupID: Number - primary key
    b) mentorMembers: String - Array of User objects
    c) groupsID : Integer - foreign key

13) UserMentorGroup: A junction table(Associative entity) for group and user Since a mentorUser can join many MentorGroup and A MentorGroup can have many mentorUser
    a) UserMentorGroupID: Number - primary key
    b) groupID: Number - foreign key reference Mentorgroup
    c) userID: Number - foreign key reference mentorUser

14) Groups : Class - a list of all groups for access
    a) GrouopsID: Number - primary key
    b) mentorGroups : String - Array of MentorGroup objects
    c) groups:String - Array of Group objects

15) Post : Class - Posts that users can create in order to interact with the larger community.
    a) postID: Number - primary key
    b) userID: Number - foreign key
    c) content : String - the text content
    d) comments: String - Array of comments/replies to this post
    e) codeBlock: String
    f) date: Date
    g) time: Time
    h) likes: Number - number of likes on the post
    i) threadID : foreign key

16) Forum : Class - A collection of posts and data about the forum
    a) forumID: Number - primary key
    b) threadID : Number - foreign key
    c) threadTitle: String - Title of the forum thread
    d) date: Date - object
    e) time: Time - object
    f) access : String - List of members who have access
    g) threads: String - Array of ForumThread objects

17) ForumThread: Class - used to contain all posts under a thread topic

a) threadID: Number - label each thread in unique identifier
b) originalPoster: String - User object who started the thread
c) threadTitle: String - title of the forum thread
d) posts: Array of Post objects which make up the thread
e) date: Date - object
f) time: Time - object
g) forumID: Number - foreign key

18) CodeChallenge : Class - The coding challenges that each user has access to and can attempt to solve.
    a) challengeID : Number - primary key
    b) title: String
    c) description: String
    d) language: String
    e) difficulty: String
    f) codingBlock: String
    g) deadline: Date - object
    h) completionPoints: Number - points gained for completing this challenge
    i) solutions: String - array of ChallengeSubmission objects - record of X successful solutions
    j) codingTests: String
    k) pseudocodeHint: String

19) UserChallenge: A junction table(Associative entity) for user and codeChallenge. Since a user has many codeChallenge and A codeChallenge belong to many user
    a) userChallengeID : Number - primary key
    b) userID: Number - foreign key
    c) challengeID: Number - foreign key

20) ChallengeSubmission: Class - contains submission data
    a) challengeSubID:Number -primary key
    b) userID: Number - foreign key
    c) challengeID: Number - foreign key
    d) codSub: string- the code is submitted
    e) date: Date - object
    f) time: Time - object
    g) verifiedSolution: Boolean - true if the submission was successful

21) SubmissionShare : Class (FR 1.18) - unique class to share with peers when you complete a challenge
    a) shareID : Number - primary key
    b) userID : Number - foreign key
    c) likes: Number
    d) Comments : String

22) Feedback : Class - The review form that mentors use to give feedback to mentees on their coding challenge solutions.
    a) feedbackID : Number - primary key
    b) userID : Number - foreign key reference mentorUser
    c) challengeSubID : Number - foreign key reference challengeSubmission
    d) solutionSubmission: String - ChallengeSubmission object the feedback is in response to
    e) organizationFeedback: String - mentor's written feedback on formatting/organization of code
    f) organizationScore: Number - mentor's scoring on scale of 1-5
    g) logicFeedback: String - mentor's written feedback on code logic and efficiency
    h) logicScore: Number - mentor's scoring on scale of 1-5
    i) commentFeedback: String - mentor's written feedback on code comments
    j) commentScore: String - mentor's scoring on scale of 1-5

23) Leaderboard : Class - list of all users organized by ranking points to be displayed as a table
    a) leaderboardID: Number - primary key
    b) userID: Number - foreign key
    c) numPoints : Number - Each user's number of points held
    d) rankTitle: String - Each user's rank title
    e) rankIcon: String - corresponding icon for user ranking

24) Rank : Class - the different icons and titles that users can acquire as they gain more points
    a) icon : String - path to image file
    b) title : Symbol
    c) rankID: Number - primary key
    d) ranksID: Number -foreign key

    e) pointsRange: Number - function returning true if User collected points fall in the range

25) Ranks : Class - list of different rank objects that users can acquire as they gain more points, and functions involving the ranks
    a) ranksID : Number - primary key
    b) rankID : Number - foreign key
    c) checkRequirements: String - function to check what rank User has and return the correct rank

26) Trophy : Class - inherited by SpecificTrophy class. Users earn these for specific achievements
    a) name: String
    b) trophyID: Number - primary key
    c) description: String - describes requirements to earn trophy
    d) Trophy flag : Boolean - true if user possesses trophy
    e) userID: Number - foreign key
    f) checkRequirements : String- abstract method, implemented by SpecificTrophy

27) SpecificTrophy: Class extends Trophy - users can earn trophies for different achievements
    a) trophyID: Number - primary key
    b) hasTrophy : Boolean - used by User, is 1 if checkRequirements returns checkRequirements : function to be implemented

28) JobListing: Class - Job listings that are available for any user to apply for.
    a) userID: Number - foreign key
    b) jobID: Number - primary key
    c) title: String
    d) company: String
    e) location: String
    f) description: String
    g) requirements: String
    h) date: Date - object
    i) applicationLink: String

29) PaymentInfo: Class - Users will be able to store their payment information for purchases
    a) paymentID: Number- primary key
    b) cardNumber: Number

c) cardName: String
d) zipCode: Number
e) backCode: Number - CVV/CVC code

30) UserPayment: A junction table(Associative entity) for payment and user Since a user has many paymentInfo and a paymentInfo belong to many user
a) userPaymentID: Number - primary key
b) userID: Number - foreign key
c) paymentID: Number - foreign key

31) Message : Class - Sent between users as direct messaging
a) MessageID: Number -primary key
b) sendingUser: Number - who's sending the message (foreign key)
c) receivingUsers: Number - who's receiving the message(foreign key)
d) time: Time- object
e) date: Date - object
f) content: String - content of the message

32) MessageThread: Class - includes all past replies to a single message thread
a) messageThreadID : Number - primary key
b) messageID: Number -foreign key
c) participatingUsers: String - Array of User objects who are in the message thread

33) Inbox : Class - Every user contains an inbox of messages that other users have sent them
a) inboxID: Number - primary key
b) userID: Number - foreign key
c) messageThreads:String - array of MessageThread objects

34) SupportForm : Class - All users can use these to communicate with the company
a) supportFormID : Number - primary key
b) from_userID : Number - foreign key reference user
c) to_userID : Number - foreign key reference userHiring
d) date: Date - object
e) time: Date - object
f) message: String - populated by user from UI

35) userHiring: Class extends User - A company user that hiring user

a) userID : Number : primary key
b) company : String - the company that user work for
c) [2]position : String - The position of user is in company

36) chatSession : Meeting rooms which users can use to meet with other users (free or premium). (Assuming this relates to workspace idea - what separates this from an inbox thread with multiple recipients?)
    a) chatSessionID: Number - primary key
    b) User ID : Number - foreign key
    c) date : Date - object
    d) title : String
    e) invitees : Json - list of user_ID's
37) UserChatSession: A junction table(Associative entity) for user and chatSession

Since a user has many codeChallenge and A codeChallenge belong to many user

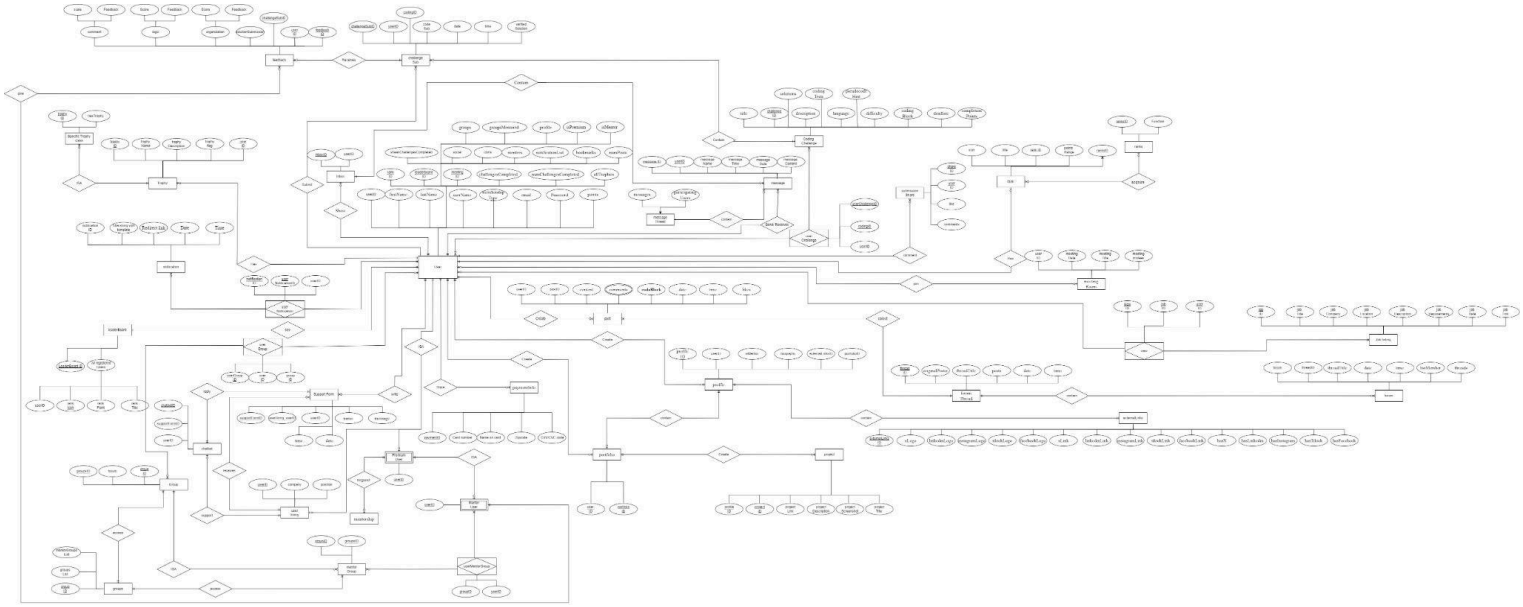
a) userChatSessionID: Number - primary key
b) userID: Number - foreign key
c) chatSessionID: Number - foreign key

38) Chatbot : alternative to support form for users to communicate with an AI rep for the company
    a) Chatbot ID: Number - primary key
    b) User ID : Number - foreign key reference userHiring

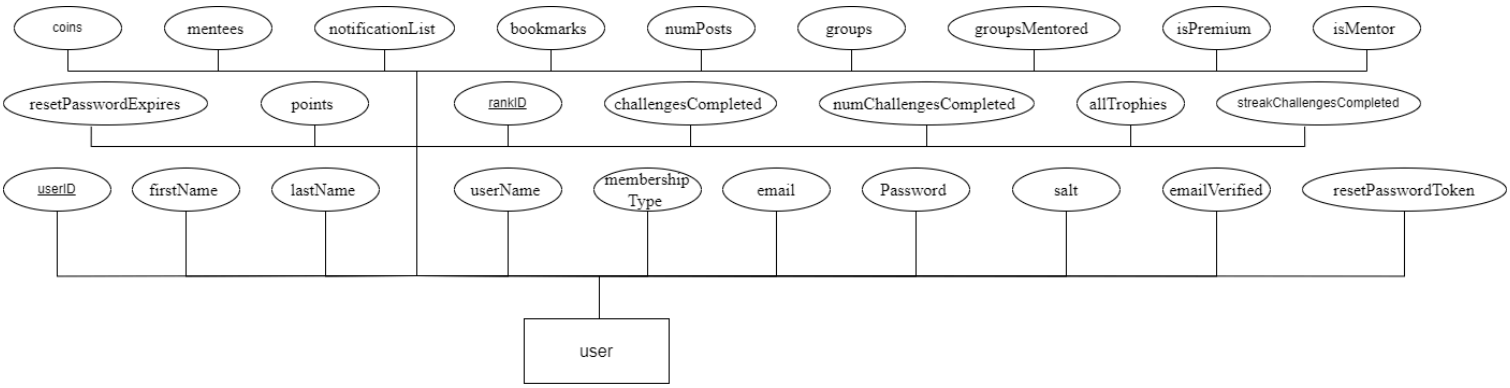
2

2.2. **ERD Creation**: Develop an Entity Relationship Diagram (ERD) using a software tool like draw.io, focusing on the main functionalities of your system.**(Refer to conceptual design)**

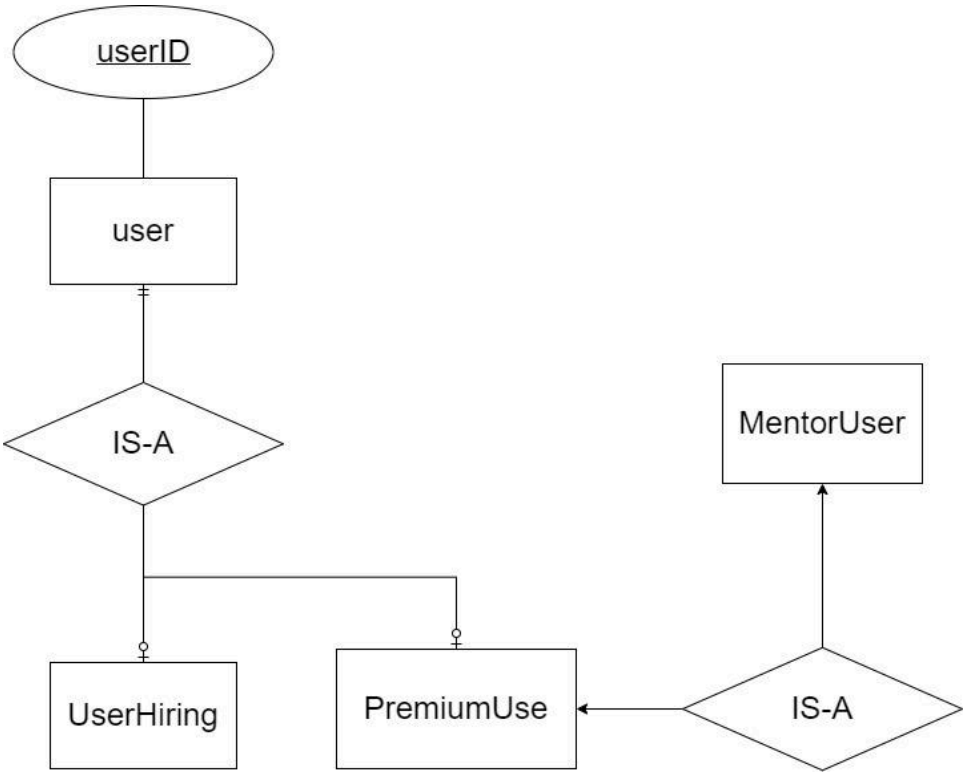
1) overView:



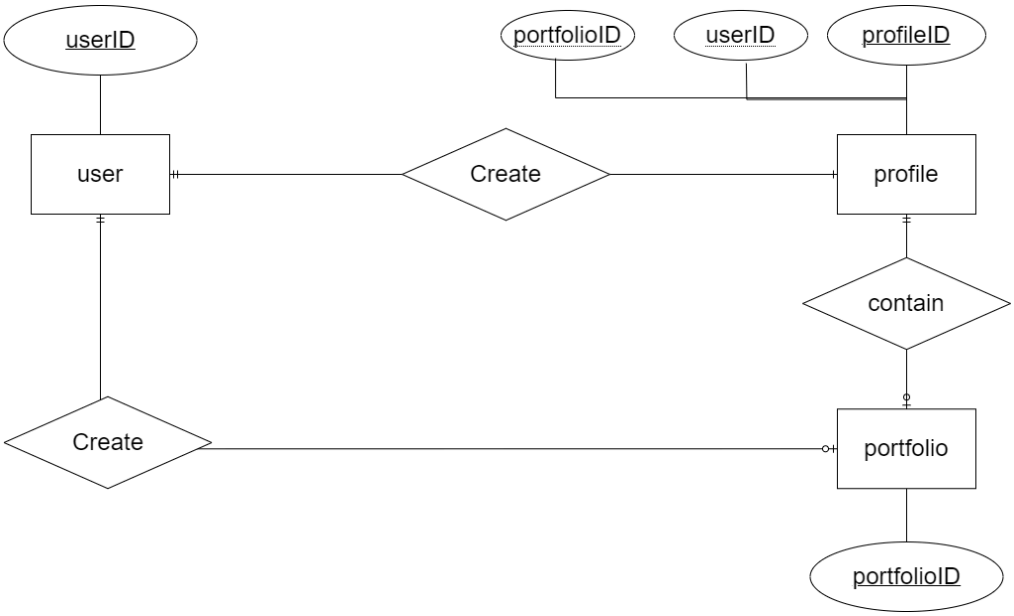
2) User table



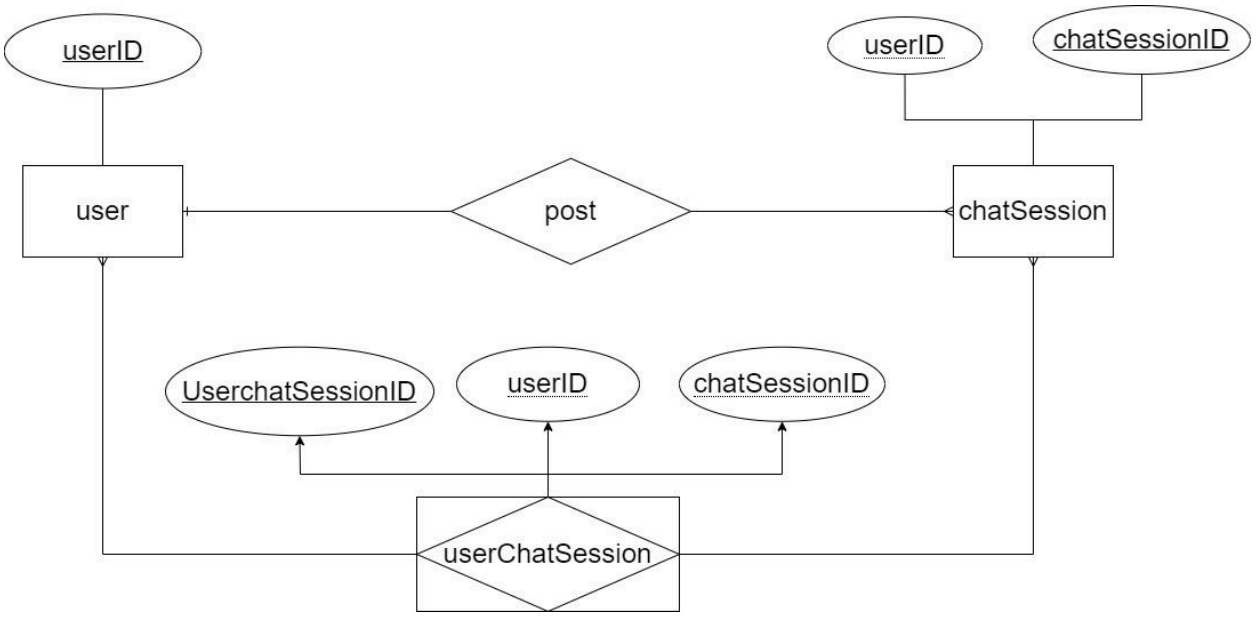
3) Inheritance



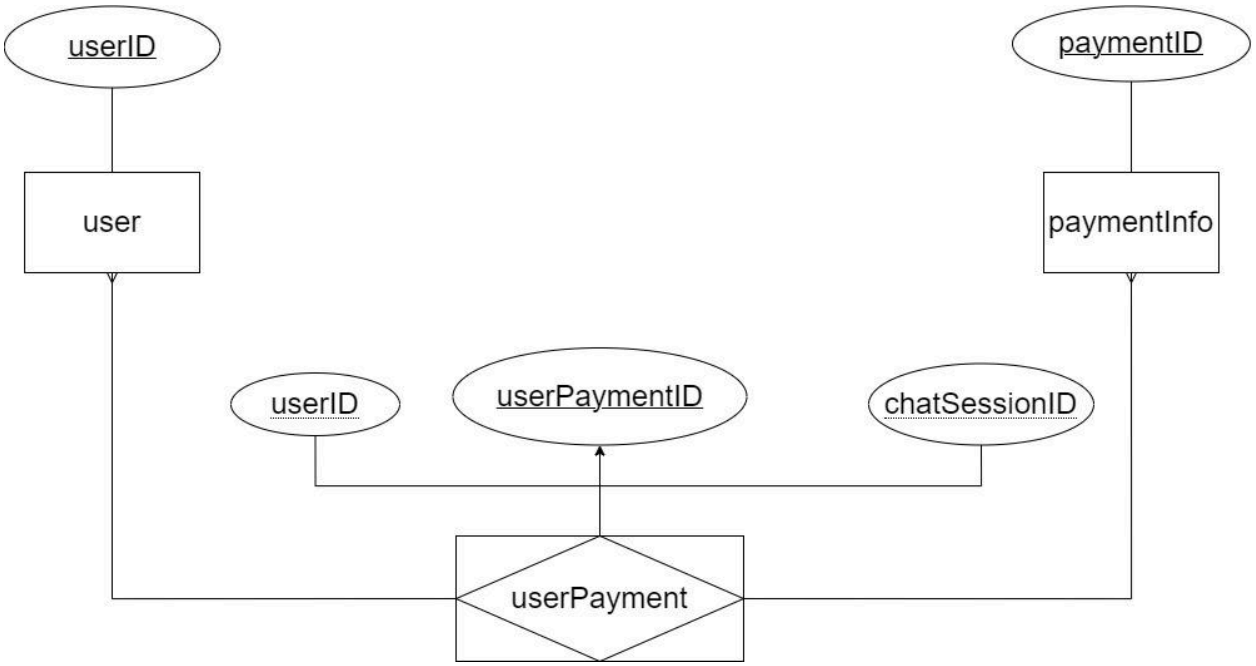
4) user, Profile and portfolio relationship



5) ChatSession and userChatSession

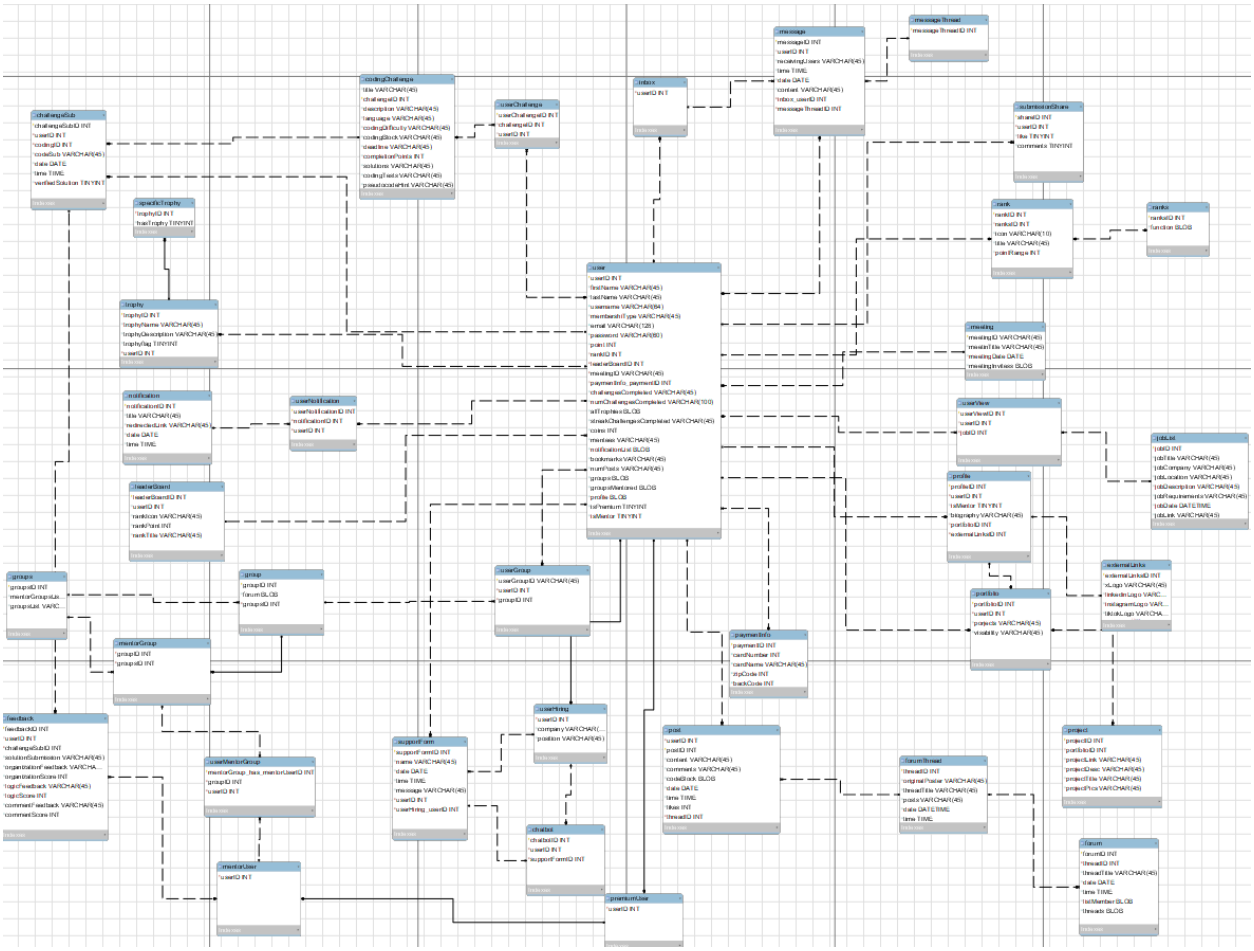


6) PaymentInfo and userPayment



ERDV2 Draw Io link:
https://drive.google.com/file/d/1zurpNjxRKnkSV0DKK_vZ_35LPtuBBGZS/view?usp=drive_link

2.3. **EER Diagram**: Create an Entity Establishment Relationship (EER) diagram using MySQLWorkbench or DBeaver.



github link:
https://github.com/sfsu-joseo/csc648-848-05-sw-engineering-su24-T1/blob/backend-dev/application/server/db/High_level_database_architecture/EER.mwb

branch:backend-dev

folder: application/server/db/High_level_database_architecture/ EER.mwb

2.4. **Forward Engineering**: Forward engineer your database model and include it in this section.

Githlink:https://github.com/sfsu-joseo/csc648-848-05-sw-engineering-su24-T1/blob/backend-dev/application/server/db/High_level_database_architecture/forward_engineer_process.sq

branch:backend-dev

folder: application/server/db/High_level_database_architecture/
forward_engineer_process.sql

3. **Media Storage Decision**:

3.1. Decide if images and video/audio will be stored in file systems or in DB BLOBs (Binary Large Objects).

After discussion we will keep the image in the instance and the metadata for image and video link will be stored in mysql, so the user shall search for the component they need.

3.2. Describe any other special data format requirements like for video/audio/GPS, etc.

The user profile entity contains another externalLinks which link to another entity that has components for the user to fill out. For example xLink,linkedinLink, ,instagramLink,tiktokLink and facebookLink. For coding challenges because it is plain text it will also be stored in mysql as an object.

5. High Level APIs and Main Algorithms

APIs List

- api/auth
  - api/auth/login
    Authenticates users and sets authentication cookies (token).
  - api/auth/signup
    Verifies if a user already exists, if not create a new user and send verification email
  - api/auth/verify/:token
    Decode JWT token and verify related email
  - api/auth/reset/:token
    Decode JWT Token and allow user to reset password

- api/user
    - api/user/:user_id
    - api/user/:user_id/rank
    - api/user/:user_id/trophy
    - api/user/:user_id/portfolio
    - api/user/:user_id/mentorship
    - api/user/:user_id/mentorship?mentor_id=
    - api/user/:user_id/mentorship/:mentorship_id
    - api/user/:user_id/payment
    - api/user/:user_id/payment/:payment_id
    - api/user/:user_id/feed
    - api/user/:user_id/thread
    - api/user/:user_id/thread/:thread_id
    - api/user/:user_id/thread/:thread_id/message
    - api/user/:user_id/thread/:thread_id/message/:message_id

- api/portfolio
    - api/portfolio/:portfolio_id
    - api/portfolio/:portfolio_id/project

- api/leaderboard
  Get leaderboard details

- api/project
    - api/project/:project_id

- api/meeting
    - api/meeting/:meeting_id

- api/group

    - api/group/:group_id
    - api/group/:group_id/member
    - api/group/:group_id/join

- api/challenge
    - api/challenge/:challenge_id
    - api/challenge/:challenge_id/submission
    - api/challenge/:challenge_id/submission/:submission_id
    - api/challenge/:challenge_id/submission/:submission_id/run

        - api/challenge/:challenge_id/submission/:submission_id/feedback
        - api/challenge/:challenge_id/submission/:submission_id/feedback/:feedback_id
        - api/challenge/:challenge_id/feedback
        - api/challenge/:challenge_id/feedback/:feedback_id

- api/forum
    - api/forum/:forum_id

- api/post
    - api/post/:post_id
    - api/post/:post_id/comment
    - api/post/:post_id/comment/:comment_id

- api/job
    - api/job/:job_id
    - api/job/:job_id/apply
    - api/job/:job_id/application
    - api/job/:job_id/application/:application_id

Significant Algorithms or Processes

1. Verification Emails
   It will send emails to verify users and reset passwords. It will use a URL and short lifetime json web token (JWT). If the user opens the URL in browser (Get Request) within the timeframe, it will perform the required operation like verify email or reset password. This functionality utilizes SMTP to send the emails to users.

2. Login
   Authenticate users. It retrieves the password salt, hashes the user input with HMAC SHA-256, and compares it with the user hashed password in the database. It also generates a json web token and responds with the HTTP header Set-Cookies. It will support resetting passwords by email.

3. Signup
   It will make sure no other user exists with the same email. It will verify password strength, generate random salt, and hash the password with HMAC SHA-256. It stores the values in the database. It will also verify user email with a verification link.

4. Resource Authorization

It will ensure the request user is privileged to perform the resource operation. It will check if the user is the owner of said record or it is public.

5. Resource Data Operations
   CRUD: create, read, update, delete with sequelize ORM. It will also sync required search fields to the Opensearch instance.
   List: It will rank, sort data according to resource type and user properties. For example, if the user is browsing challenges, it will make a request to the search instance with the user fields and sort according to relevance. If the user is listing solutions, it will rank them according to run metrics like resource utilization and time.
   Filter: It will handle users requests to filter records with any combination values of the resource properties.
   Search: It will make a request to the search engine with a user query. It will also pass additional fields to boost and ranking results according to user preference/filters. If a user requests a search on a previous search result, this API will just include the previous query with an and parameter for the new one. If no results are found, it will omit the user query and pass the additional fields to find relevant ones.

6. Challenge Creation
   This API will accept required data to create a challenge and ensure the challenge can be solved. It will require a base code snippet, expected stdout, test snippet, programming language, and required environment constraints (memory, CPU, time, network).

7. Challenge Submission
   It will run a submission with the appropriate programming language and get run metrics like resource usage and time. It will combine the user, test, base codes, make a request to Judge0 endpoint with a generated callback URI for this specific submission. Judge0 will make a PUT request to the callback URL and the API will update the submission result accordingly.

8. Submissions Ranking
   The API will sort submissions according to their run metrics like resource utilization and execution time. Judge0 will execute the successful submissions multiple times to gather accurate metrics.

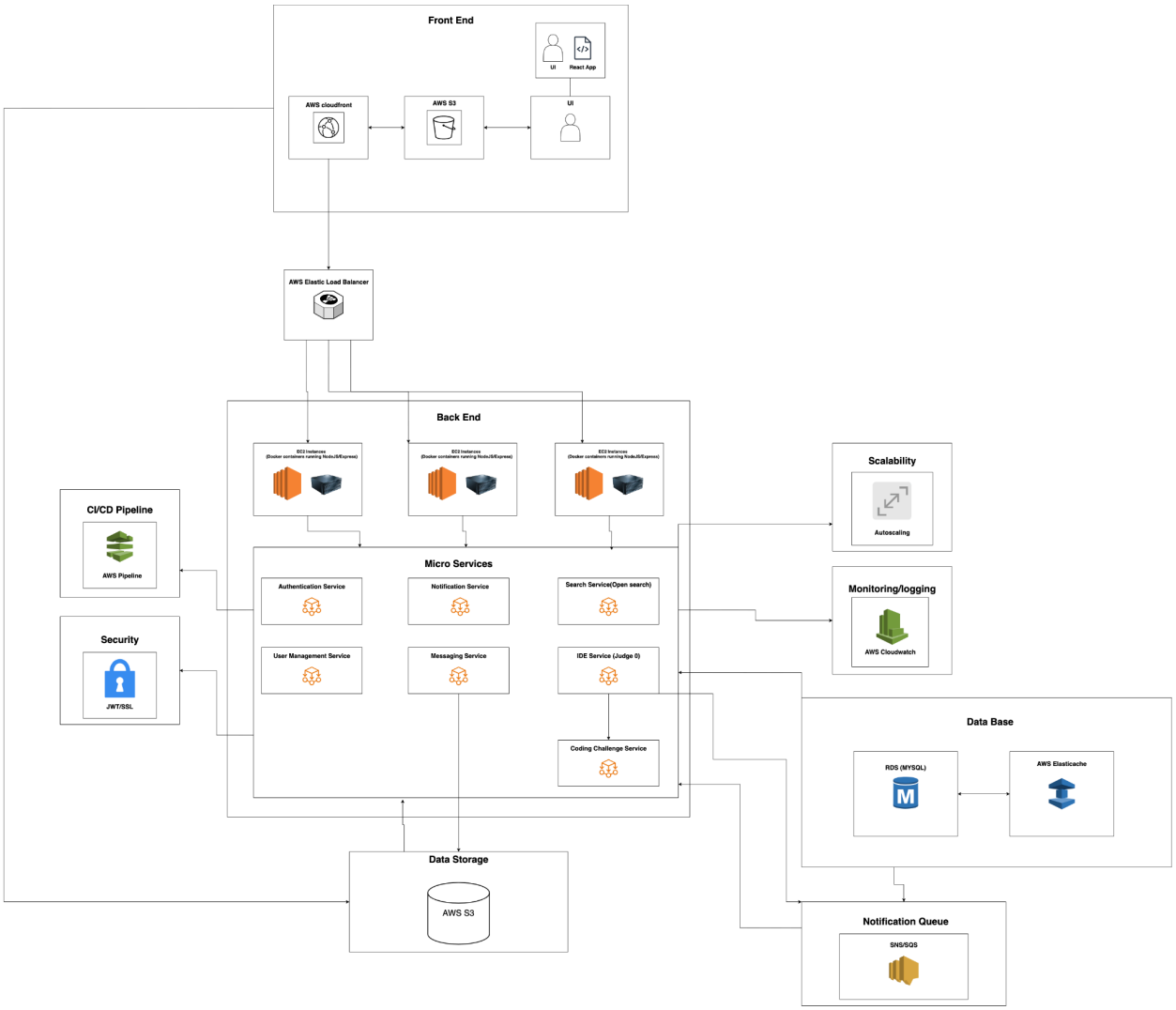
Microservices and tools

1. Judge0 (Still in discussion)
   It is an open source online IDE to run different code snippets, manage sandbox environments, resources, and queues. It comes with the following features:

- It has scalable architecture in which it can scale vertically in increased threads and horizontally in multiple instances. It has a queue and creates multiple workers. The application API can decide the submission instance according to availability.
- Sandboxed compilation and execution where user untrusted codes are run in a managed sandbox with support of setting multiple options like network access, memory limit, CPU limit, timeout, and isolate concurrent submissions.
- Support for custom user-defined compiler options, command-line arguments, and time and memory limits.
- Detailed execution results like stdout of execution, memory/cpu utilizations, and time. It allows even running the code multiple times to get accurate results and enable better ranking results.
- Webhooks (HTTP callbacks) in which the application API will not have to wait for the submission to end or pool for results and just receive the call back request once the submission is done.
- Open Source and can be hosted in an EC2 instance but we will use a shared cloud in rapid apis and migrate once scaling is required.

2. Opensearch Instance
   It is an open-source search and indexing engine. It enables advanced search and indexing functionality.
   - Fuzzy matching to handle misspellings
   - Result ranking and filtering
   - Multi field queries and field boosting
   - K-NN relevance matches
   - Realtime results
   - Scale out to accommodate large data

6. System Design



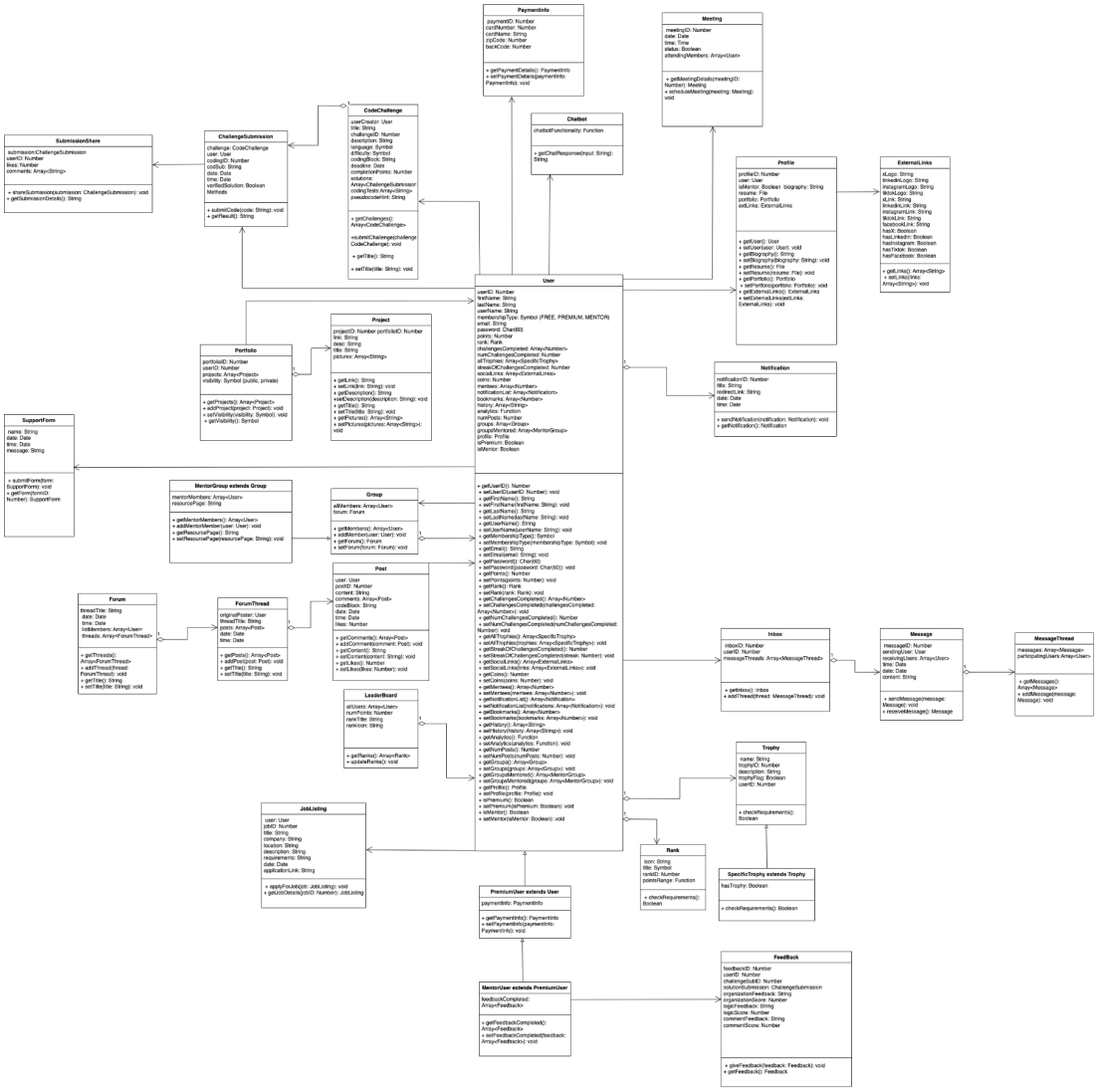
https://drive.google.com/file/d/1doonMbyxpfyKRlWUUELP12gjqkFxR1u_/view?usp=sharing

For our system design we use a microservices approach, where independent services like authentication, user management, coding challenges, notifications, messaging, search, and an integrated development environment communicate with one another to form the product. This improves the manageability, scalability, and flexibility of our product. For high availability and dependability, a lot of EC2 instances performing backend services are spread out with traffic using Amazon Elastic Load Balancer. Data that is frequently needed is cached using ElastiCache, which reduces the load on the main database and speeds up finding data.

Scalability is achieved through auto scaling groups, which change the number of EC2 instances according to traffic load for fault tolerance and reliability. AWS S3 stores huge files and backups with built in reliability, while the database is hosted in a multiple availability zone architecture for high availability and data security. Docker containers deployed on EC2 instances contain backend services, which assure consistency and make deployment and scaling easier. ElastiCache improves performance by caching data, while RDS (MySQL) guarantees data replication across different availability zones.

JWT is used for safe authentication of users, data encryption for protecting sensitive information, and SSL/TLS encryption for data in transit. Network traffic to EC2 instances is managed by security groups. The user interface (React App), AWS CloudFront for distributing static components, and AWS S3 for storing data make up the frontend layer. EC2 instances running Docker containers with NodeJS/Express services and other microservices form the backend layer. RDS (MySQL) and ElastiCache are features of the database layer. SNS/SQS manages notifications, AWS CloudWatch handles logging and monitoring, and AWS CodePipeline handles CI/CD. Overall our architecture ensures a safe, scalable, and adaptable system that can manage changing traffic and offer a good user experience

UML Diagram

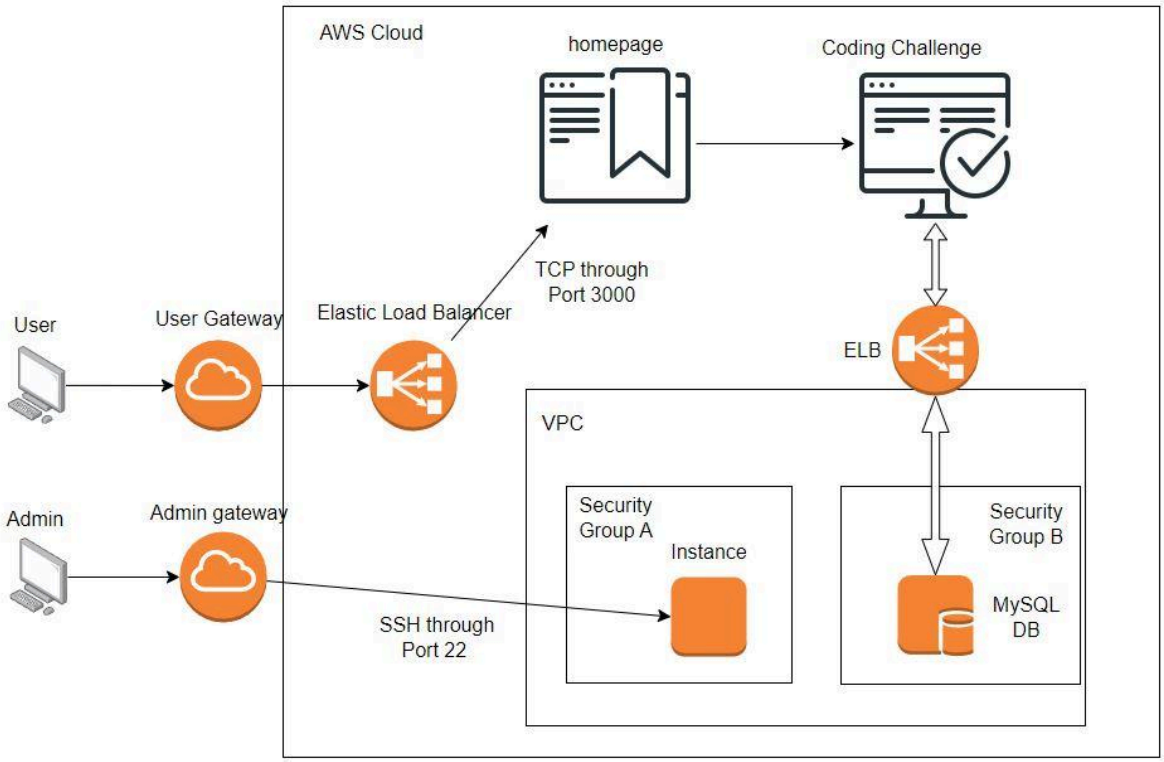


https://drive.google.com/file/d/1hG_XbO-yDG5hdWcWTPN0i2EP4WwMEPFY/view?usp=sharing

A number of patterns are used in the UML class diagram for our app to improve maintainability, scalability, adaptability, and reusability. For data consistency and to reduce cost, the

Leaderboard class follows a pattern that guarantees a single instance. The creation process is encapsulated for flexibility and manageability in the pattern used to build ChallengeSubmission objects. Subscribed users can get updates based on events from the NotificationService thanks to our implementation, which separates the sender and recipient. We provide an ongoing combination of behaviors by adding premium functionalities to User profiles without changing the base class. Our design for the ranking system included the possibility to use various ranking algorithms, which improved scalability and flexibility. Finally, by creating a single interface, we simplify user related tasks and increase the readability and maintenance of the code.

7. High Level Application Network and Deployment Design



8. Identify actual key risks for your project at this time

1) **Skills Risk (do you have the right skills):**
    a) Most of our group has not taken a database course yet, so we rely on only one team member to keep the backend team up to speed.
    b) Previous networking experience is limited, so our network diagram is very simple.
2) **Schedule Risk (can you make it given what you committed and the resources):**

a) People have jobs and scheduling meetings with the whole team doesn't always line up with everyone's schedule.
b) People sometimes have things come up(emergency, etc) and they're not in the loop of our plan
c) We didn't plan for the potential for a bottle neck, which delayed some of our internal deadlines longer than expected.
d) Recent heat advisories may affect internet access with blackouts or brownouts from overloaded electric grids

3) **Technical Risks (any technical unknowns to solve):**
    a) Not everyone is familiar with the backend technologies/micro services, thus we needed to take extra time to learn them and implement them.
4) **Teamwork Risks (any issues related to teamwork):**
    a) Notion is a new tool, thus our team isn't used to updating their tasks through it yet.
    b) With people having work, and thus unable to make every meeting, we rely on recordings to keep everyone on the same page.
    c) Potential communication outside of team channel may cause disarray unless everything is relayed back to the team channel
5) **Legal/Content Risks (can you obtain content/SW you need legally with proper licensing, copyright):**
    a) We do not have a collection of challenges to populate the coding challenges section of the product.
    b) We would have to seek some kind of legal advice for the use of social iconography.
    c) Would have to get permission from legal teams to use their problems and solutions
    d) Would have to get permission from videographers to use their solutions
    e) Company permission to be represented within our application

9. Project management

After reflecting on Milestone 1, it was clear that we needed to be more organized in our approach to Milestone 2. First, we created our Milestone 2 template, with each section of the document being pulled directly from the Milestone 2 document. Our team got together in a meeting and we reviewed the document in full going through each goal listed on the document and, using Notion, created a rough idea of who might be the owner of each task. Through our meetings, we were able to refine some of the tasks down and organize the larger goals into more manageable chunks. Notion was also a huge help when we needed to organize tasks, and

I also wished we had used it for Milestone 1 because we were able to assign tasks and update the group as to the status easier than simply using Discord. We also were able to silo some of the work between team members that were more focused on front-end or back-end development. We still looked over the work and were able to critique each deliverable as it was added to the M2 document. Moving forward we will use this approach, as it made the M2 sprints easier to manage.

MILESTONE 3v2:

SW Engineering CSC648-01 Summer 2024

CodeConnect

Team 1 - PikaDevs

Max Shigeyoshi - mshigeyoshi@sfsu.edu - Team Lead
Aaron Rayray - arayray@sfsu.edu - Github Master
Noah Hai - nhai@sfsu.edu - Doc Editor
Shez Rahman - srahman2@sfsu.edu - Backend Lead
Ghadeer Al-Badani - galbadani@sfsu.edu - Backend
Majd Alnajjar - malshemari@sfsu.edu - Frontend
William Pan - wpan1@sfsu.edu - Database Admin
Phillip Ma - pma1@mail.sfsu.edu - Frontend Lead

“Milestone 3“

6/26/2024

History Table:

Date	Changes
7-28-24	Missing wireframes added

1. Data Definitions

1. User : Class - The user that creates the account.
    - 1.1. userID : Number - Primary key
    - 1.2. firstName : String
    - 1.3. lastName : String
    - 1.4. userName : String
    - 1.5. membershipType : String-Symbol (FREE, PREMIUM, MENTOR)
    - 1.6. email : String
    - 1.7. password : Number
    - 1.8. salt : Number - random data for hashing
    - 1.9. emailVerified : Boolean
    - 1.10. resetPasswordToken: String
    - 1.11. resetPasswordExpires : Date
    - 1.12. points : Number
    - 1.13. rankID : Number - Rank object
    - 1.14. challengesCompleted : Number - Array of Numbers(completed challenge IDs)
    - 1.15. numChallengesCompleted : Number
    - 1.16. allTrophies : JSON- Array of SpecificTrophy objects
    - 1.17. streakChallenge: Number - Streak of challenges completed
    - 1.18. coins : Number
    - 1.19. mentees : Json - Array of userIDs
    - 1.20. notificationList :Json - Array of Notification objects
    - 1.21. bookmarks : String - Array of postIDs
    - 1.22. numPosts : Number - number of posts + number of comments
    - 1.23. groups : Json - Array of Group objects
    - 1.24. groupsMentored : Json - Array of MentorGroup objects
    - 1.25. isPremium : boolean
    - 1.26. isMentor : boolean

2. PremiumUser : Class, extends User : A user that has paid the subscription fee for premium features
    - 2.1. userID : Number - Primary and foreign key

3. MentorUser : Class, extends User - A user that has become a mentor to other users. Must be approved by interview.
    - 3.1. userID : Number - Primary and foreign key

3.2. feedbackCompleted: String- Array of FeedbackForm objects

4. Profile : The user's profile which has information about the user.
   4.1. profileID: Number - Primary key
   4.2. userID: Number - foreign key (the user who owns the profile)
   4.3. isMentor : boolean
   4.4. biography : String
   4.5. resume : String
   4.6. portfolioID : Number

5. ExternalLinks : Class - contains external links to be used by profile
   5.1. externalLinksID: Number - Primary key
   5.2. profileID : Number - foreign key
   5.3. xLogo: String - filepath to image
   5.4. linkedinLogo: String - filepath to image
   5.5. instagramLogo: String - filepath to image
   5.6. tiktokLogo: String - filepath to image
   5.7. facebookLogo: String - filepath to image
   5.8. xLink: String
   5.9. linkedinLink: String
   5.10. instagramLink: String
   5.11. tiktokLink: String
   5.12. facebookLink: String
   5.13. hasX: Boolean
   5.14. hasLinkedin: Boolean
   5.15. hasInstagram: Boolean
   5.16. hasTiktok: Boolean
   5.17. hasFacebook: Boolean

6. Portfolio: Class - Users will have this to display their projects
   6.1. portfolioID : Number - Primary key
   6.2. userID : Number - foreign key (user that create portfolio)
   6.3. visibility: Symbol (public or private)

7. Project : Class - these are data items to be stored in portfolio
   7.1. projectID : Number - Primary key
   7.2. Portfolio : Number - Foreign key
   7.3. link: String - link to project
   7.4. desc: String - text description of project
   7.5. title: String - title of project

7.6. pictures: String - Array of file paths to images

8. Notification: Class
   8.1. notificationID : Number - primary key
   8.2. title: String - title of notification with template
   8.3. redirectLink: String - clicking notification takes you to link
   8.4. date: Date
   8.5. time: Time

9. UserNotification: A junction table(Associative entity) for notification and user Since a user has many notification and a notification belong to many user
   9.1. userNotificationID: Number - primary key
   9.2. userID: Number - foreign key
   9.3. notificationID: Number - foreign key

10. Group : Class - users can join these to bond over commonalities such as being alumni from the same school, or having interest in a certain technology
   10.1. GroupsID: Number - primary key
   10.2. allMembers: String - Array of User objects who have access
   10.3. forum: String - Forum object
   10.4. groupsID : Number - foreign key

11. UserGroup: A junction table(Associative entity) for group and user
   Since a mentorUser can join many MentorGroup and A MentorGroup can have many mentorUser
   11.1. UserGroupID: Number - primary key
   11.2. groupID: Number - foreign key reference group
   11.3. userID: Number - foreign key reference User

12. MentorGroup : Class extends Group - a group for mentors to communicate with their mentees
   12.1. groupID: Number - primary key
   12.2. mentorMembers: String - Array of User objects
   12.3. groupsID : Integer - foreign key

13. UserMentorGroup: A junction table(Associative entity) for group and user
   Since a mentorUser can join many MentorGroup and A MentorGroup can have many mentorUser
   13.1. UserMentorGroupID: Number - primary key
   13.2. groupID: Number - foreign key reference Mentorgroup

13.3. userID: Number - foreign key reference mentorUser

14. Groups : Class - a list of all groups for access
    14.1. GrouopsID: Number - primary key
    14.2. mentorGroups : String - Array of MentorGroup objects
    14.3. groups:String - Array of Group objects

15. Post : Class - Posts that users can create in order to interact with the larger community.
    15.1. postID: Number - primary key
    15.2. userID: Number - foreign key
    15.3. content : String - the text content
    15.4. comments: String - Array of comments/replies to this post
    15.5. codeBlock: String
    15.6. date: Date
    15.7. time: Time
    15.8. likes: Number - number of likes on the post
    15.9. threadID : foreign key

16. Forum : Class - A collection of posts and data about the forum
    16.1. forumID: Number - primary key
    16.2. threadID : Number - foreign key
    16.3. threadTitle: String - Title of the forum thread
    16.4. date: Date - object
    16.5. time: Time - object
    16.6. access : String - List of members who have access
    16.7. threads: String - Array of ForumThread objects

17. ForumThread: Class - used to contain all posts under a thread topic
    17.1. threadID: Number - label each thread in unique identifier
    17.2. originalPoster: String - User object who started the thread
    17.3. threadTitle: String - title of the forum thread
    17.4. posts: Array of Post objects which make up the thread
    17.5. date: Date - object
    17.6. time: Time - object
    17.7. forumID: Number - foreign key

18. CodeChallenge : Class - The coding challenges that each user has access to and can attempt to solve.
    18.1. challengeID : Number - primary key

18.2. title: String
18.3. description: String
18.4. language: String
18.5. difficulty: String
18.6. codingBlock: String
18.7. deadline: Date - object
18.8. completionPoints: Number - points gained for completing this challenge
18.9. solutions: String - array of ChallengeSubmission objects - record of X successful solutions
18.10. codingTests: String
18.11. pseudocodeHint: String

19. UserChallenge: A junction table(Associative entity) for user and codeChallenge. Since a user has many codeChallenge and A codeChallenge belong to many user
19.1. userChallengeID : Number - primary key
19.2. userID: Number - foreign key
19.3. challengeID: Number - foreign key

20. ChallengeSubmission: Class - contains submission data
20.1. challengeSubID:Number -primary key
20.2. userID: Number - foreign key
20.3. challengeID: Number - foreign key
20.4. codSub: string- the code is submitted
20.5. date: Date - object
20.6. time: Time - object
20.7. verifiedSolution: Boolean - true if the submission was successful

21. SubmissionShare : Class (FR 1.18) - unique class to share with peers when you complete a challenge
21.1. shareID : Number - primary key
21.2. userID : Number - foreign key
21.3. likes: Number
21.4. Comments : String

22. Feedback : Class - The review form that mentors use to give feedback to mentees on their coding challenge solutions.
22.1. feedbackID : Number - primary key
22.2. userID : Number - foreign key reference mentorUser
22.3. challengeSubID : Number - foreign key reference challengeSubmission

22.4. solutionSubmission: String - ChallengeSubmission object the feedback is in response to
22.5. organizationFeedback: String - mentor's written feedback on formatting/organization of code
22.6. organizationScore: Number - mentor's scoring on scale of 1-5
22.7. logicFeedback: String - mentor's written feedback on code logic and efficiency
22.8. logicScore: Number - mentor's scoring on scale of 1-5
22.9. commentFeedback: String - mentor's written feedback on code comments
22.10. commentScore: String - mentor's scoring on scale of 1-5

23. Leaderboard : Class - list of all users organized by ranking points to be displayed as a table
    23.1. leaderboardID: Number - primary key
    23.2. userID: Number - foreign key
    23.3. numPoints : Number - Each user's number of points held
    23.4. rankTitle: String - Each user's rank title
    23.5. rankIcon: String - corresponding icon for user ranking

24. Rank : Class - the different icons and titles that users can acquire as they gain more points
    24.1. icon : String - path to image file
    24.2. title : Symbol
    24.3. rankID: Number - primary key
    24.4. ranksID: Number -foreign key
    24.5. pointsRange: Number - function returning true if User collected points fall in the range
25. Ranks : Class - list of different rank objects that users can acquire as they gain more points, and functions involving the ranks
    25.1. ranksID : Number - primary key
    25.2. rankID : Number - foreign key
    25.3. checkRequirements: String - function to check what rank User has and return the correct rank

26. Trophy : Class - inherited by SpecificTrophy class. Users earn these for specific achievements
    26.1. name: String
    26.2. trophyID: Number - primary key
    26.3. description: String - describes requirements to earn trophy
    26.4. Trophy flag : Boolean - true if user possesses trophy

26.5. userID: Number - foreign key

26.6. checkRequirements : String- abstract method, implemented by SpecificTrophy

27. SpecificTrophy: Class extends Trophy - users can earn trophies for different achievements
    27.1. trophyID: Number - primary key
    27.2. hasTrophy : Boolean - used by User, is 1 if checkRequirements returns checkRequirements : function to be implemented

28. JobList : Class - Job listings that are available for any user to apply for.
    28.1. userID: Number - foreign key
    28.2. jobID: Number - primary key
    28.3. title: String
    28.4. company: String
    28.5. location: String
    28.6. description: String
    28.7. requirements: String
    28.8. date: Date - object
    28.9. applicationLink: String

29. userView: Class - A junction table(Associative entity) for joblist and user Since a user can view many jobList and a jobList belong to many user
    29.1. userViewID:Number - primary key
    29.2. userID: Number -foreign key
    29.3. jobId: Number - foreign key

30. PaymentInfo: Class - Users will be able to store their payment information for purchases
    30.1. paymentID: Number- primary key
    30.2. cardNumber: Number
    30.3. cardName: String
    30.4. zipCode: Number
    30.5. backCode: Number - CVV/CVC code
31. UserPayment: A junction table(Associative entity) for payment and user Since a user has many paymentInfo and a paymentInfo belong to many user
    31.1. userPaymentID: Number - primary key
    31.2. userID: Number - foreign key

31.3. paymentID: Number - foreign key

32. Message : Class - Sent between users as direct messaging
    32.1. MessageID: Number -primary key
    32.2. userID: Number - who's sending the message (foreign key)
    32.3. inboxID: Number - who's receiving the message(foreign key)
    32.4. time: Time- object
    32.5. date: Date - object
    32.6. content: String - content of the message
    32.7. messageID: Number -foreign key

33. MessageThread: Class - includes all past replies to a single message thread
    33.1. messageThreadID : Number - primary key
    33.2. participatingUsers: String - Array of User objects who are in the message thread

34. Inbox : Class - Every user contains an inbox of messages that other users have sent them
    34.1. inboxID: Number - primary key and foreign key
    34.2. messageThreads:String - array of MessageThread objects

35. SupportForm : Class - All users can use these to communicate with the company
    35.1. supportFormID : Number - primary key
    35.2. from_userID : Number - foreign key reference user
    35.3. to_userID : Number - foreign key reference userHiring
    35.4. date: Date - object
    35.5. time: Date - object
    35.6. message: String - populated by user from UI

36. userHiring: Class extends User - A company user that hiring user
    36.1. userID : Number : primary key
    36.2. company : String - the company that user work for
    36.3. [3]position : String - The position of user is in company
37. chatSession : Meeting rooms which users can use to meet with other users (free or premium). (Assuming this relates to workspace idea - what separates this from an inbox thread with multiple recipients?)
    37.1. chatSessionID: Number - primary key
    37.2. User ID : Number - foreign key

[3]

37.3. date : Date - object
37.4. title : String
37.5. invitees : Json - list of user_ID's

38. UserChatSession: A junction table(Associative entity) for user and chatSession Since a user has many codeChallenge and A codeChallenge belong to many user
    38.1. userChatSessionID: Number - primary key
    38.2. userID: Number - foreign key
    38.3. chatSessionID: Number - foreign key

39. Chatbot : alternative to support form for users to communicate with an AI rep for the company
    39.1. Chatbot ID: Number - primary key
    39.2. User ID : Number - foreign key reference userHiring

2. Functional Requirements

Priority 1:

1. **All Users:**

*1.1 Users shall be able to explore some portions of the product without a profile
*1.2 Users shall be able to solve an example problem.
*1.3 Users shall create a profile
*1.4 Users shall be able to Log in/Log out (only with created profile)
*1.6 Users shall be able to upload profile picture
*1.7 Users shall be able to Delete profile
*1.8 User shall be able to update payment information
*1.9 Users shall be able to make their profile private or public
*1.12 Users shall be able to check other users profiles/stats
*1.13 Users shall be able to do coding challenges
*1.15 Users shall be able to award different profile trophies for achievements
*1.17 Users shall be able to earn points for coding challenges
*1.18 Users shall be able to like/comment on challenge posts
*1.20 Users shall be able to check their coding ranking
*1.21 Users shall be able to check leaderboards
*1.22 Users shall be able to subscribe to Premium
*1.23 Users shall be able to unsubscribe to Premium
*1.32 Users shall be able to display other socials on their profiles
*1.34 Users shall be able to request additional features from the dev team
*1.35 Users shall be able to utilize live chat w/ other users
*1.36 Users shall be able to direct message other users
*1.38 Users shall be able to check coding streak counter of daily challenges

*1.39 Users shall be able to create a portfolio
*1.40 Users shall be able to check/update portfolio
*1.41 Users shall be able to change portfolio visibility (public/private)
*1.42 Users shall be able to access portfolio review
*1.45 Users shall be able to submit support forms to submit feedback regarding the app
*1.49 Users shall be able to view featured users (spotlight user that's completed most challenges/stayed the most active for the past month. Featured user gets changed monthly)
*1.54 Users shall be able to join group session workshops led by mentors
*1.60 Users shall be able to get assessed to become a mentor
*1.61 Users shall be able to use an IDE without having to create an account
*1.65 Users without premium shall be able to view three solutions per month
*1.66 Users shall be able to see the difference between premium and free membership perks
*1.67 Users shall be able to gain points by starting forum threads
*1.68 Users shall be able to gain points by commenting in threads
*1.69 Users shall be able to gain points by gaining friends
*1.70 Users shall be able to gain points by gaining mentees
*1.71 Users shall be able to gain points by gaining mentors
*1.72 Users shall be able to post text and images to their profiles
*1.73 Users shall be able to view a general feed of other users' updates and activities
*1.76 Users shall be able to search for other users
*1.77 Users shall be able to search for groups
*1.78 Users shall be able to receive notifications for new coding challenges.
*1.78 Users shall be able to receive notifications for messages and comments.
*1.79 Users shall be able to customize notification preferences.
*1.81 Users shall be able to save forum posts for later reading.
*1.82 Users shall be able to report inappropriate content.
*1.83 Users shall be able to block or mute other users.
*1.84 Users shall be able to set privacy settings for profile visibility.
*1.86 Users shall be able to request a mentor recommendation.
*1.90 Users shall be able to subscribe to notifications for specific forums or groups
*1.93 Users shall be able to view a scrolling list of job listings from home page
*1.94 Users shall be able to view other users' activity from their homepages

2. **Free Users**
   *2.2 Free Users shall be able to check code challenge repo
   *2.3 Free users shall be able to view three pseudocode hints per month

3. **Premium Users**

*3.0 Premium Users shall be able to check a single system chosen solution after completing a challenge
*3.2 Premium Users shall be able to request mentorship (premium)
*3.3 Premium Users shall be able to Match mentors with similar coding language experience
*3.4 Premium users shall be able to view one pseudocode hint per challenge
*3.7 Premium Users shall be able to request code review from a mentor

4. **Mentor Users**

*4.1 Mentor Users shall be able to review code solutions of other users they are mentoring.
*4.4 Mentor Users shall be able to review Free/Premium users resumes/portfolios
*4.5 Mentor users shall be able to complete challenge feedback on coding challenges completed by mentees
*4.6 Mentor users shall be able to gain points by completing challenge feedback on mentee solutions
*4.7 Mentor Users shall be able to upload videos
*4.8 Mentor users shall have their number of mentees displayed on their profiles
*4.9 Mentor users shall have their number of solutions reviewed displayed on their profiles
*4.10 Mentor users shall have the groups they lead displayed on their profiles

Priority 2:

1. **All Users:**

**1.11 Users shall be able to follow other users/mentors
**1.16 Users shall be able to allow switching coding languages for challenges (cross compatibility) (different compilers - maybe 2 or 3 priority here)(would prefer just different sets of challenges based on language)
**1.24 Users shall be able to see pop-up ads for premium/paid utilities
**1.48 Users shall be able to take mock interviews (practice interviews with real-time communication and feedback from peers and other users)
**1.51 Users shall be able to utilize discounts/offers on apps premium features
**1.54 Users shall be able to join group session workshops led by mentors
**1.56 Users shall be able to view/update their own calendar(scheduled hackathons, virtual meetings..) (API or creating our own calendar?)
**1.59 Users shall be able to utilize a button to change the screen to dark mode.
**1.63 Users shall be able to choose their preferred speaking language(How does that look in implementation?)
1.74 **Users shall be able to earn coins

1.75 **Users shall be able to trade coins for premium subscriptions
-Or top leaderboard users get a premium reward
1.80 **Users shall be able to bookmark coding challenges.
**1.87 Users shall be able to participate in live coding sessions.
**1.91 Users shall be able to access analytics on their coding performance.
**1.92 Users shall be able to receive notifications for new coding challenges.

2. **Free Users:**
   **2.1 Free Users shall be able to check the most average solutions after completing a challenge.

3. **Premium Users:**
   **3.1 Premium Users shall be able to check a system chosen set of 2-3 solutions after completing a challenge

4. **Mentor Users:**
   **4.3 Mentor Users shall be able to create coding challenges.

Priority 3:

1. **All Users:**
   ***1.14 Users shall be able to see popular submissions (how is popular measured?)
   ***1.25 Users shall be able to application survey for mentorship (?)
   ***1.28 Users shall be able to take hiring questionnaire challenges (which are siblings to coding challenges
   ***1.37 Users shall be able to access free/paid resources (videos/textbooks)
   ***1.43 Users shall be able to utilize chatbot (this function would be cool but bold because need to know the data items) (Use support form instead)
   ***1.44 Users shall be able to access virtual workspace (define virtual workspace?)
   ***1.46 Users shall be able to open source/collaborative coding projects (users can participate in other people's coding projects) (Similar to 1.44?)
   ***1.47 Users shall be able to read weekly digests (recommendations created for users based on interests/what they worked on)
   ***1.50 Users shall be able to utilize pair programming sessions (allows users to work on projects/code collaboratively) (similar to 1.44 and 1.46)
   ***1.52 Users shall be able to read technical news (similar to 1.47)
   ***1.57 Users shall be able to create Recruiter/Hiring Manager specific profile
   ***1.58 Users shall be able to pass a Recruiter/Hiring manager verification/background check
   ***1.88 Users shall be able to join virtual coding bootcamps.

***1.89 Users shall be able to create and manage a personal blog.

2. **Free Users: N/A**
1. **Premium Users:**

   ***3.5 Premium Users shall be able to create code challenge repo, which are reviewed by the dev team prior to publication. (with a full repo it could be harder to check solutions vs a simple IDE plug in/compiler. Resources may be difficult on this topic…maybe talk to prof)

   ***3.6 Premium Users shall be able to check/update code challenge repo(^see note for 3.5)
2. **Mentor Users:**

   ***4.2 Mentor Users shall be able to set up group work stations. (what is a group work station? Also potentially lower priority)

3. Wireframes Based on your Mockups/Storyboards (detailed)

Toby:
https://www.figma.com/design/oCH2zJ2oaSEeECz0uoyhw6/Untitled?node-id=0-1&t=G7FWKgTzsm1XXQas-1

Harold:
https://www.figma.com/design/zwPPr0m5nGPd8Y34Ptfsjf/Harold-(Wireframe)?t=GjovdPw6PjH2z6Qb-1

Heidi:
https://www.figma.com/design/GBu5SQTvxtjvZek5zAEzIV/Heidi-(Wireframe)?t=GjovdPw6PjH2z6Qb-1

Caroline:
https://www.figma.com/design/Chvvr7uvRx5gYCppPQzeFI/Caroline-(Wireframe)?t=GjovdPw6PjH2z6Qb-1

Joakim:
https://www.figma.com/design/rDx8Rfn5Ij3KwYTGEZ7FUr/Untitled?t=GjovdPw6PjH2z6Qb-1

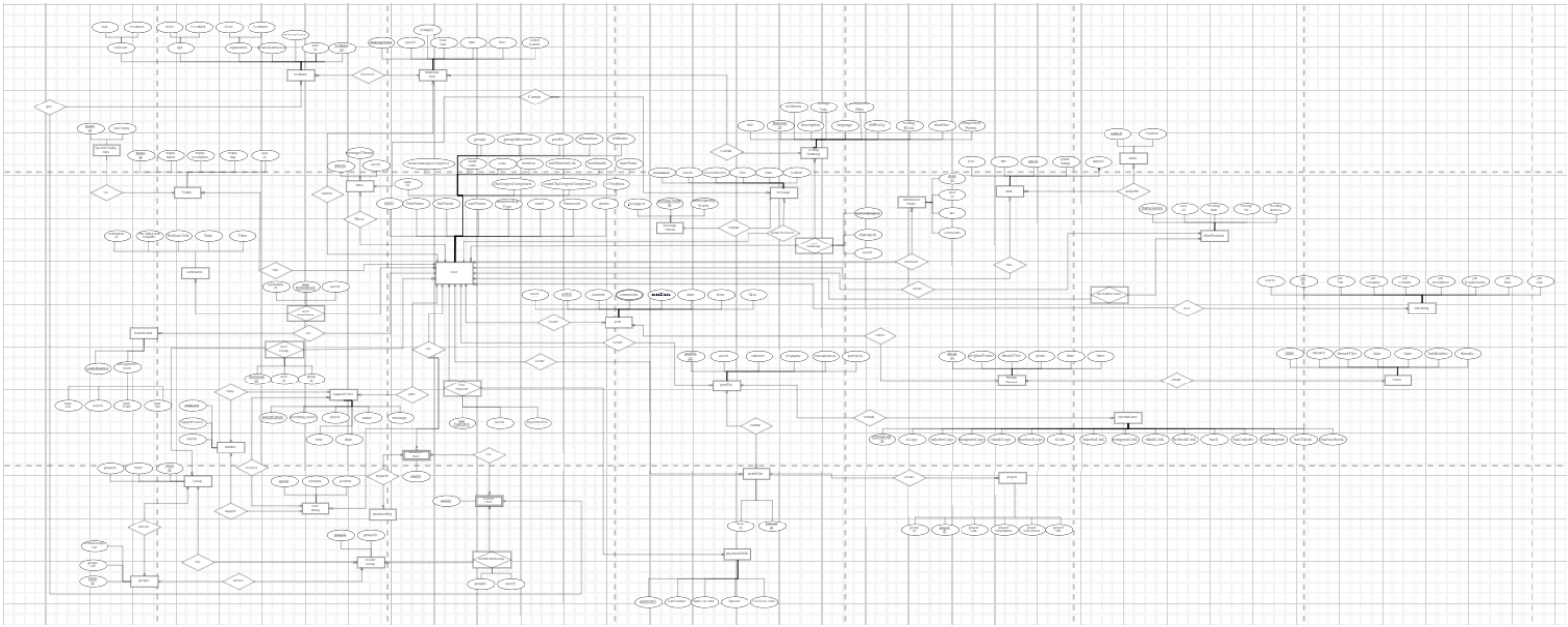
Josh:
https://www.figma.com/design/9dUpEomjSWkZCz15WkQGuv/Josh-(Wireframe)?t=GjovdPw6PjH2z6Qb-1

Jeffrey:
https://www.figma.com/design/zWR0qE8mLOUvptGNGbuI79/Jeffrey-(Wireframe)?t=GjovdPw6PjH2z6Qb-1

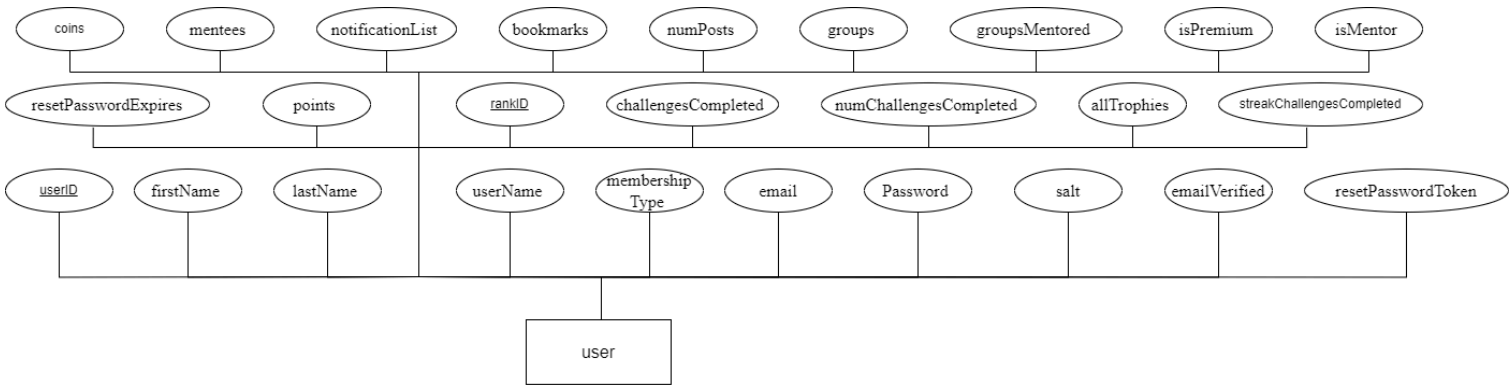
4. High level database architecture and organization (detailed)

ERD:

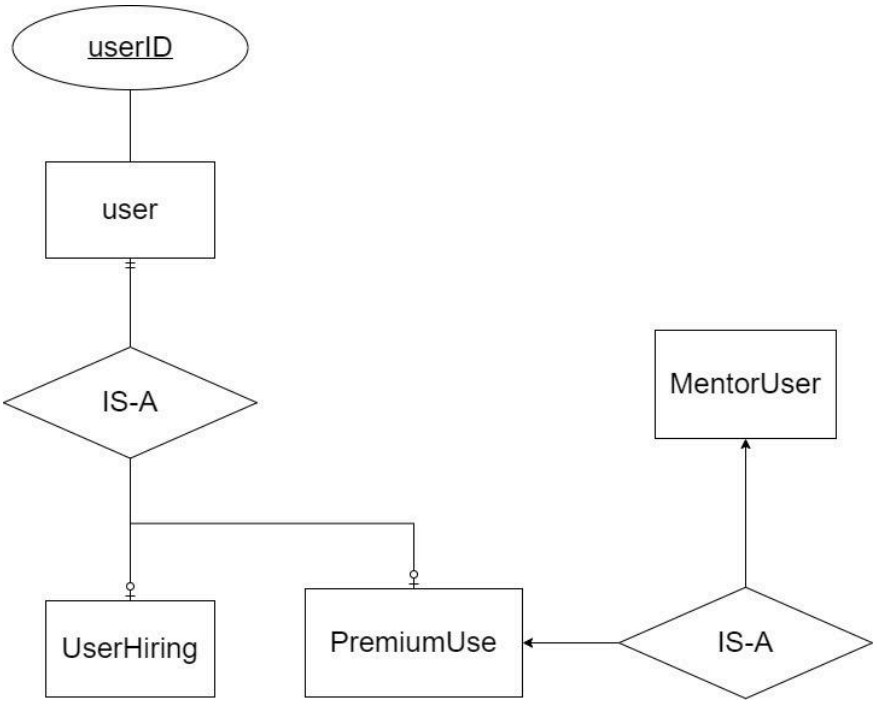
1) **overView** ERDV2 Draw Io link: https://drive.google.com/file/d/1zurpNjxRKnkSV0DKK_vZ_35LPtuBBGZS/view?usp=drive_link



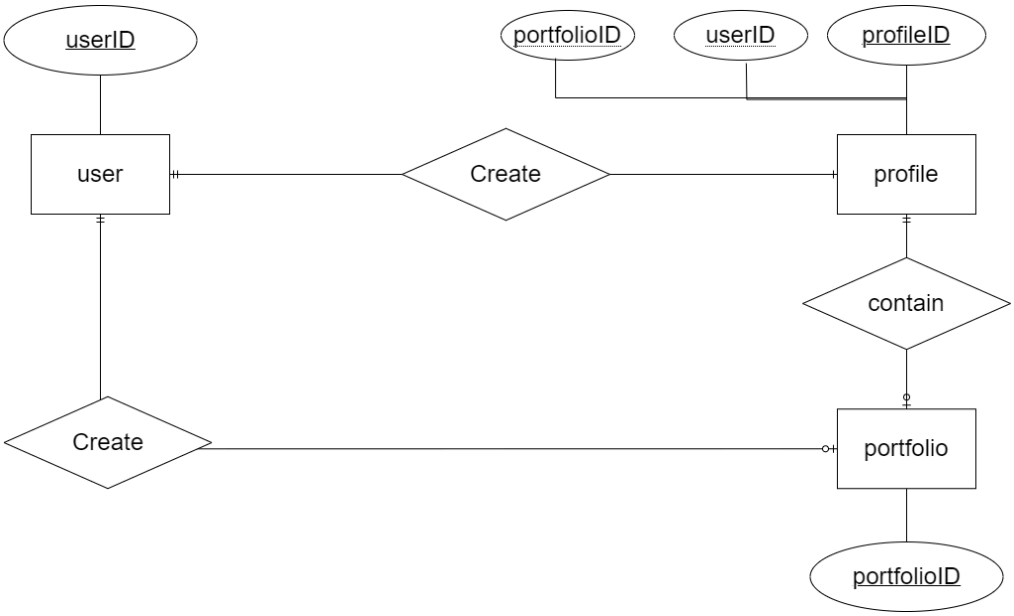
2) User table



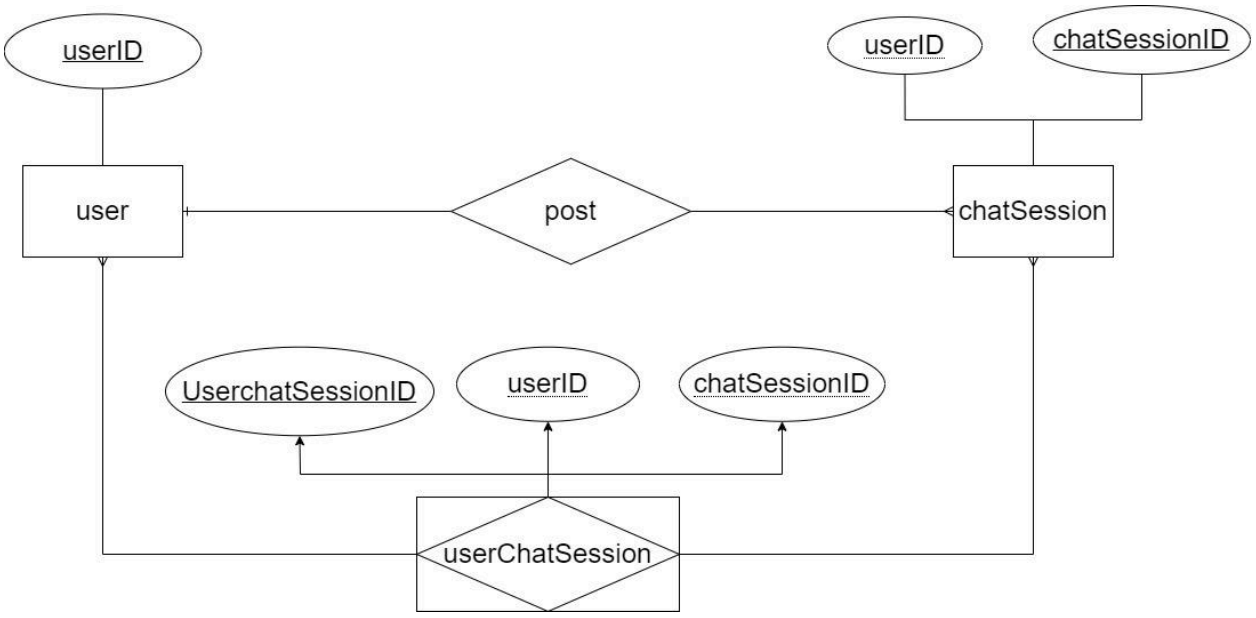
3) Inheritance



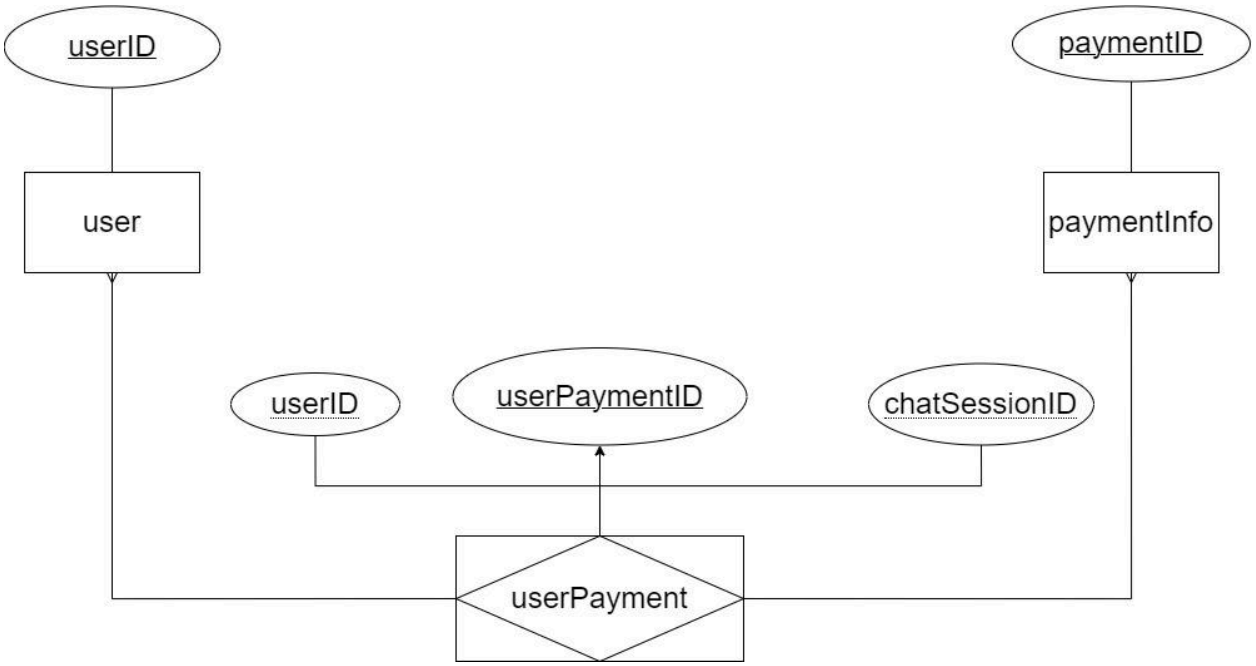
4) user, Profile and portfolio relationship



5) ChatSession and userChatSession

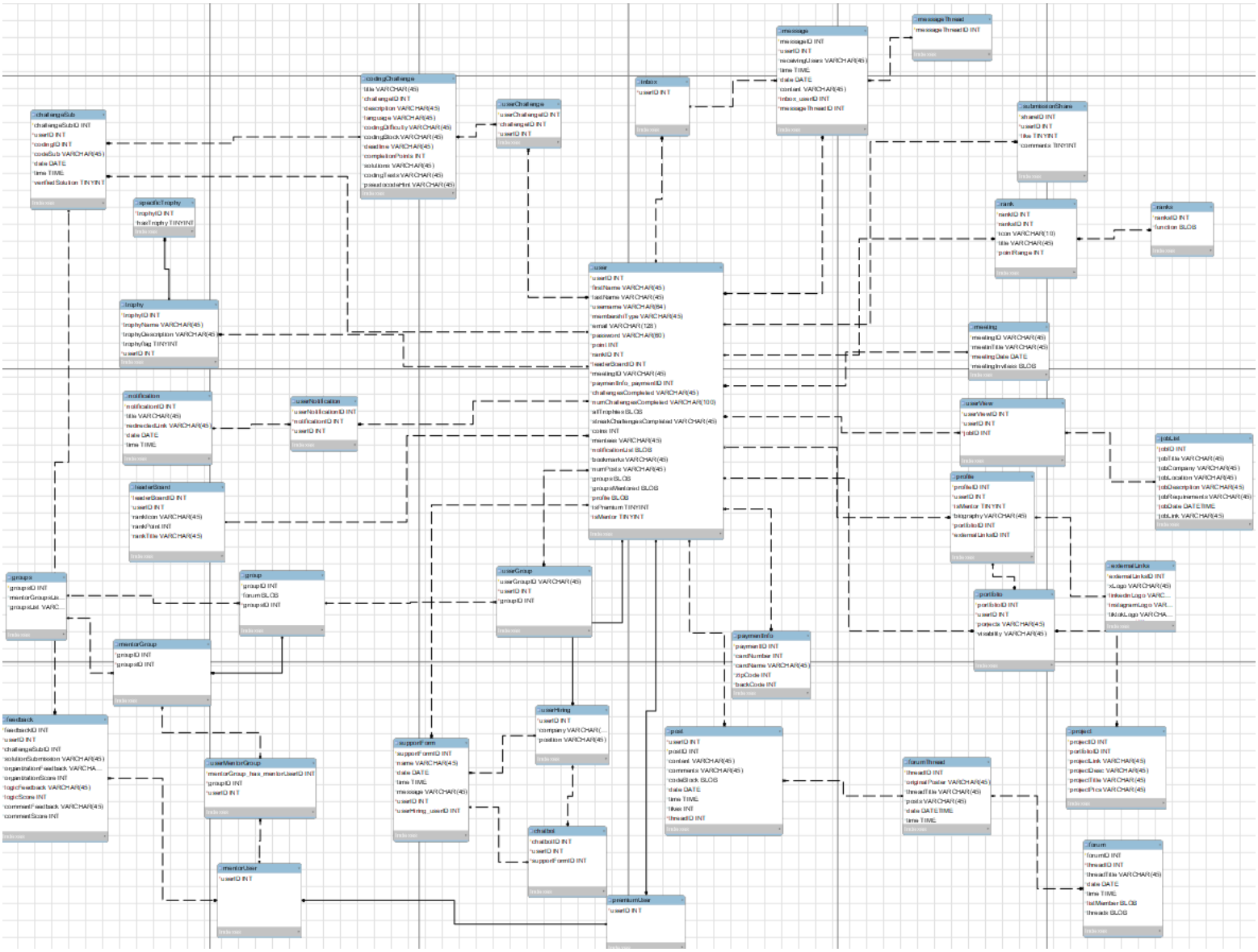


6) PaymentInfo and userPayment



EER:

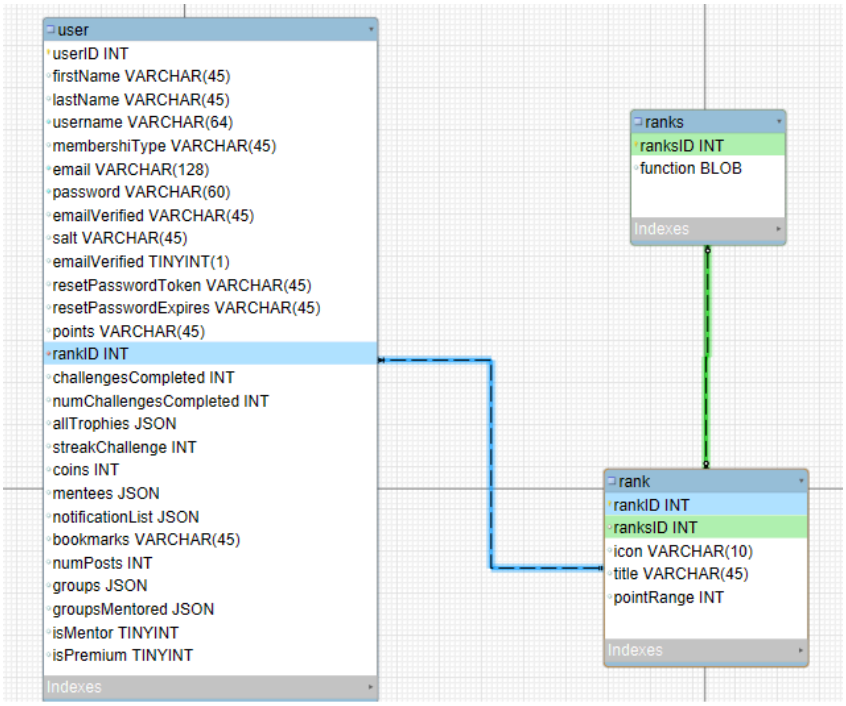
1. Overview



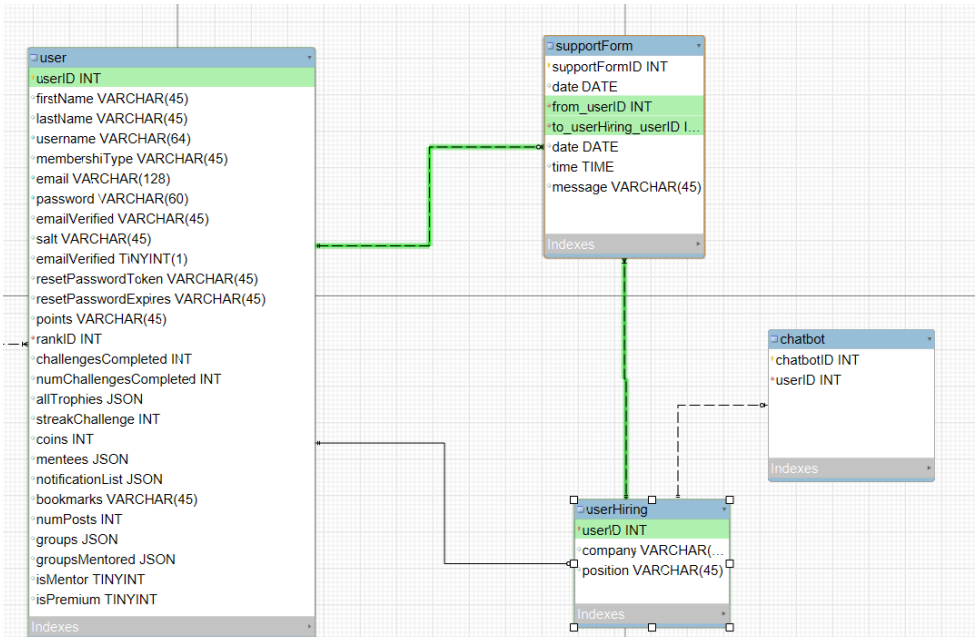
github link:
https://github.com/sfsu-joseo/csc648-848-05-sw-engineering-su24-T1/blob/will-backend-dev/application/server/db/High_level_database_architecture/EERv2.mwbb
ranch:will-backend-dev

folder: application/server/db/High_level_database_architecture/ EERv2.mwb

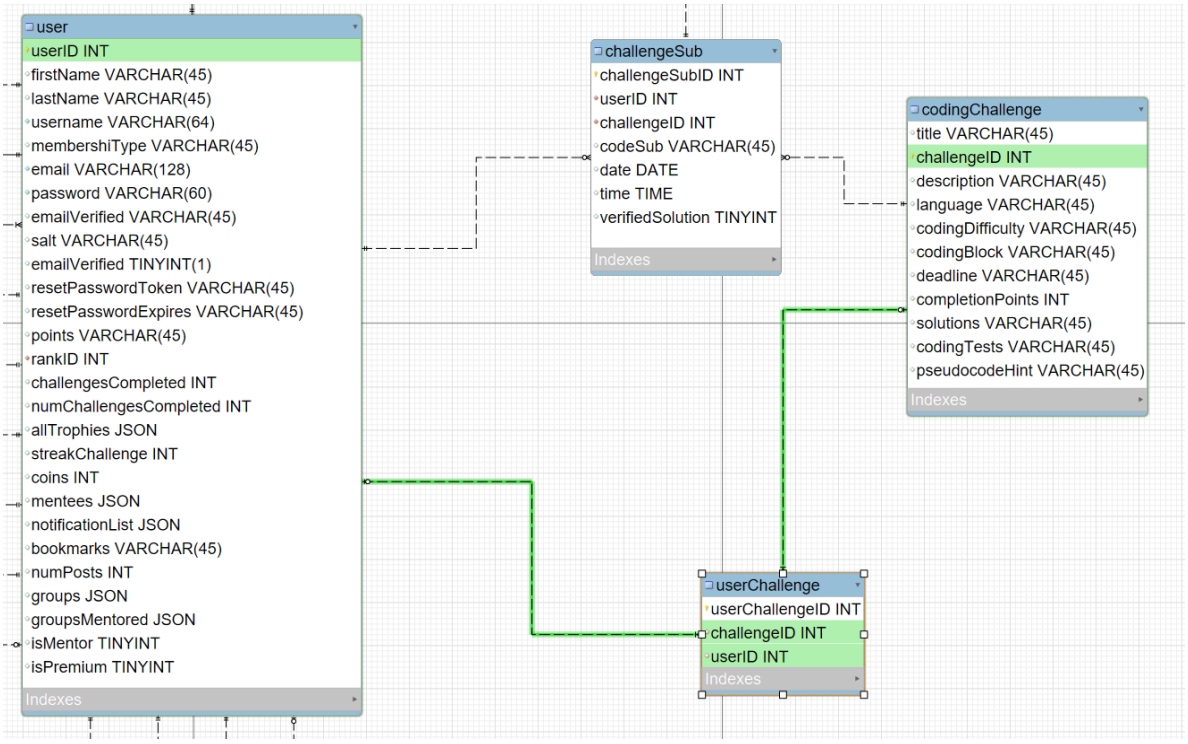
1.1. Ranks and rank



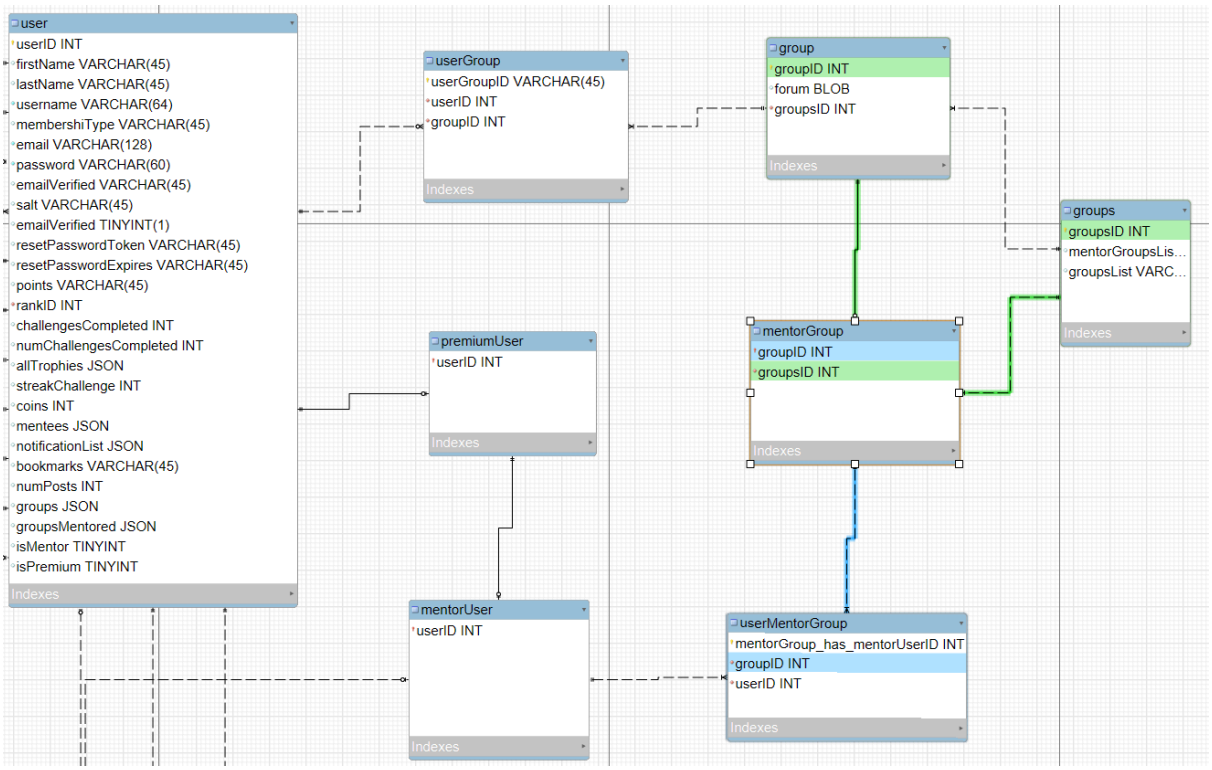
1.2. userHiring and support form



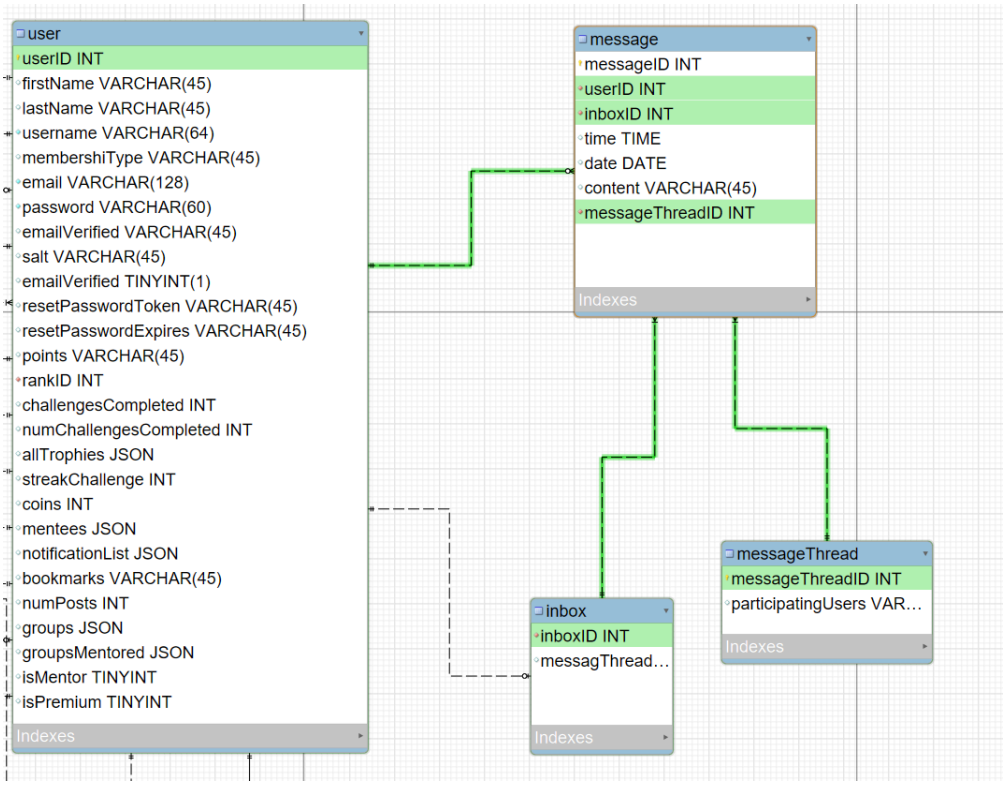
1.3. UserChallenge , codingChallenge and challengeSub



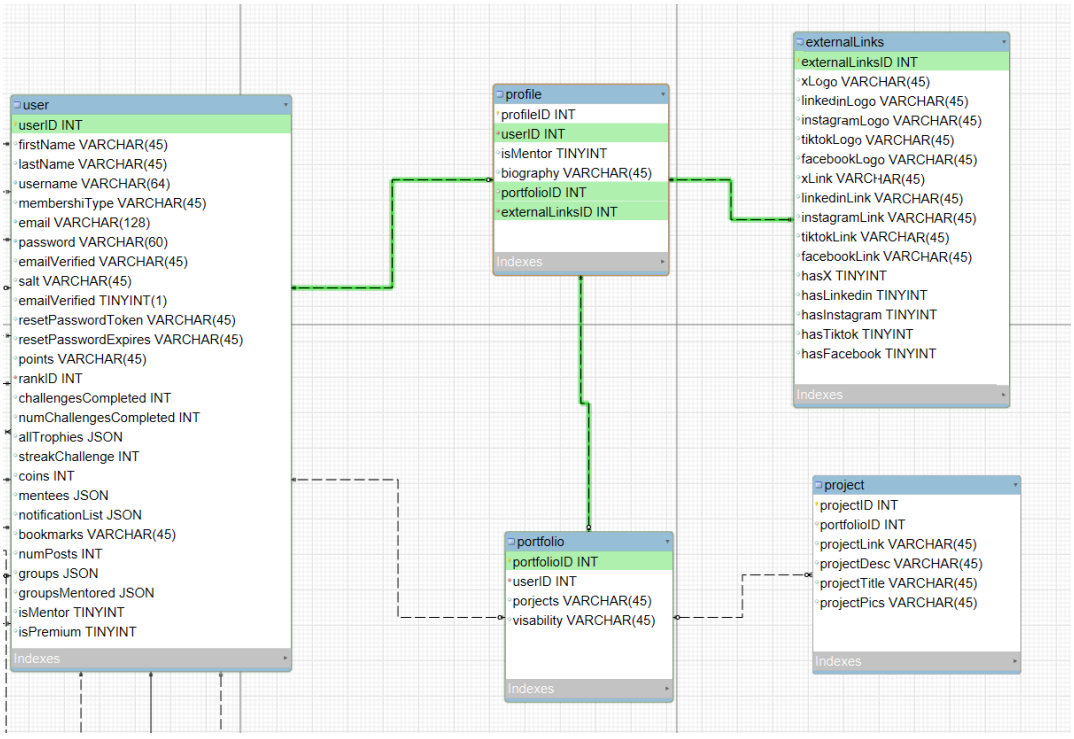
1.4. Group



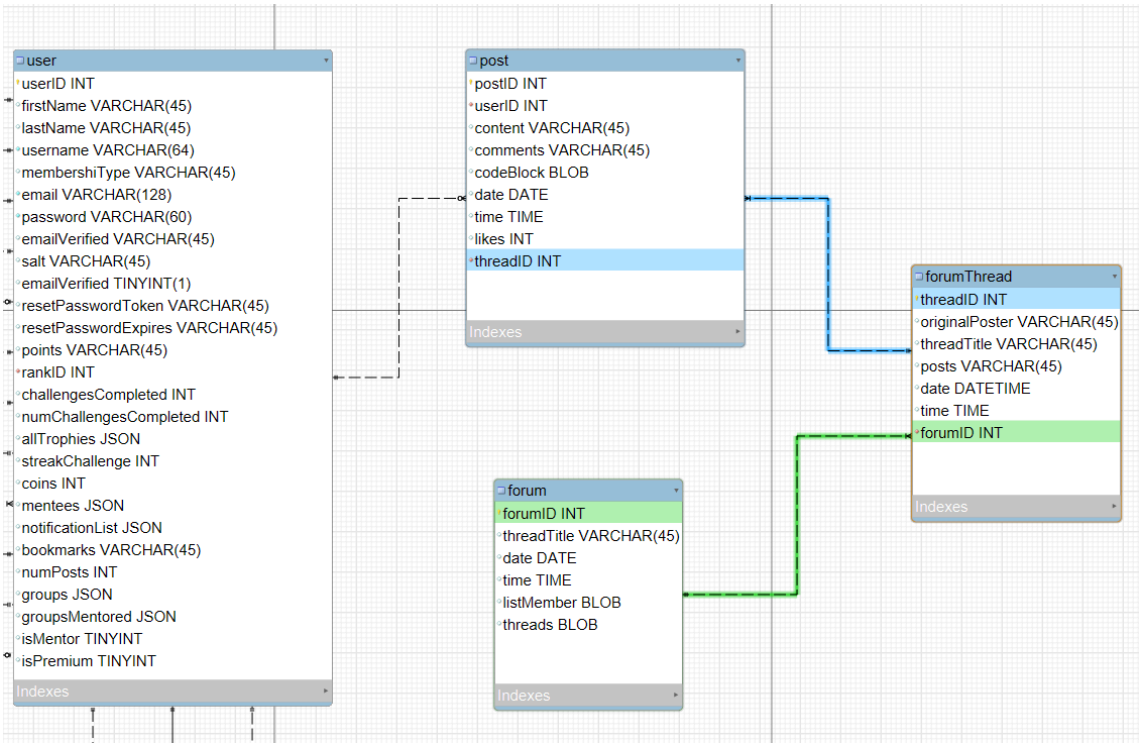
1.5. Message and inbox



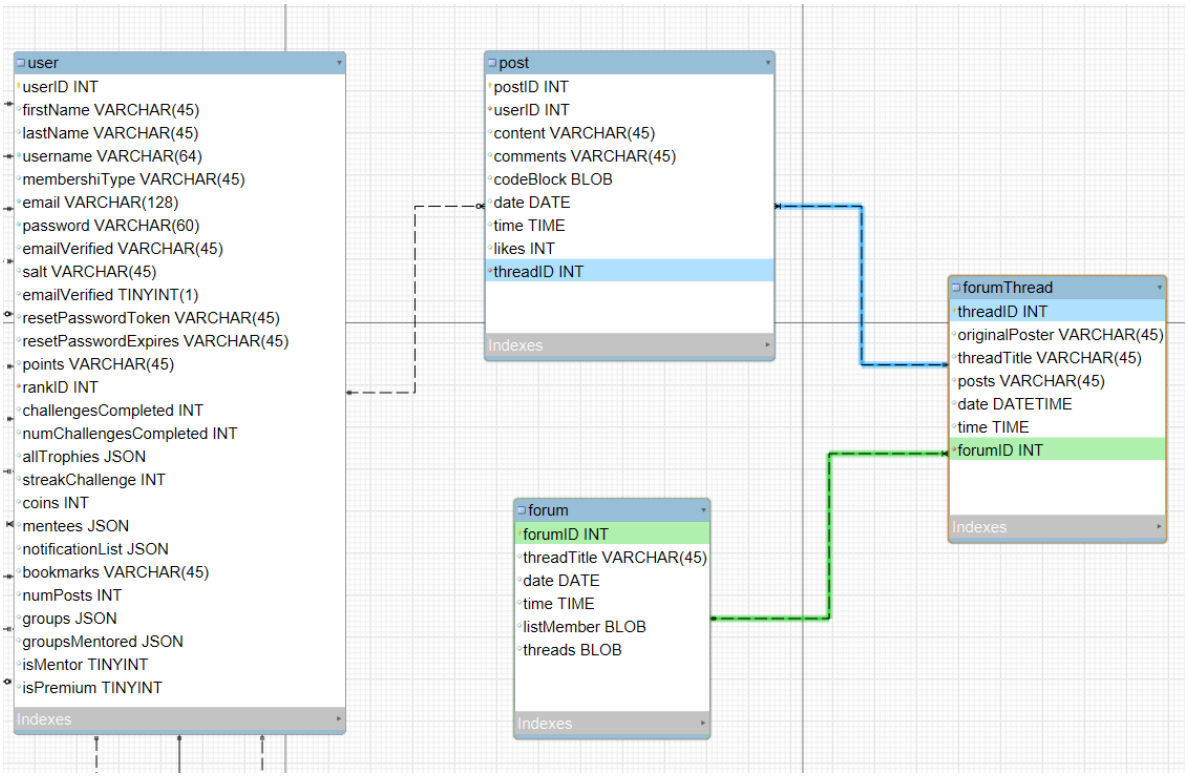
1.6. Profile, portfolio project and external link



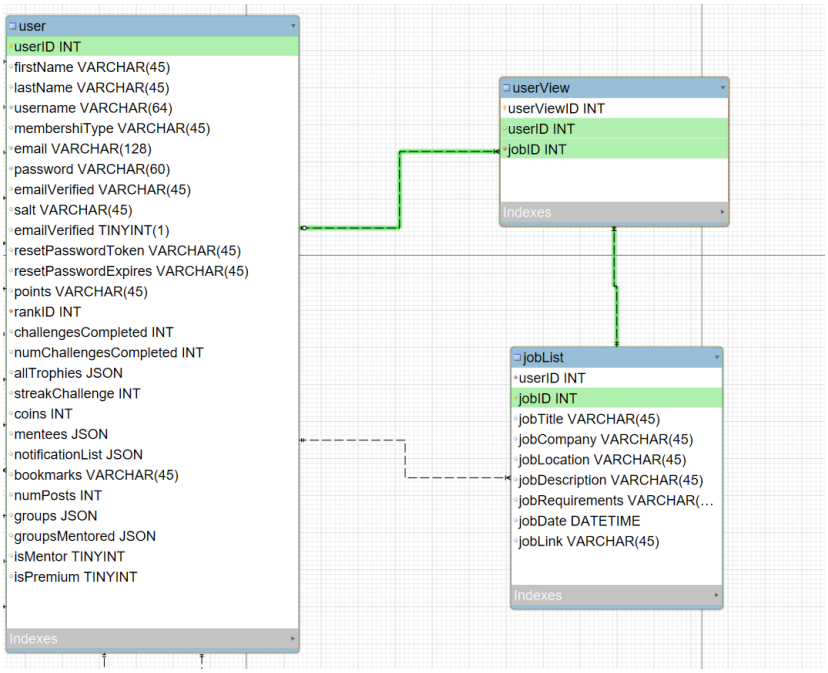
1.7. Post, forumThread,forum



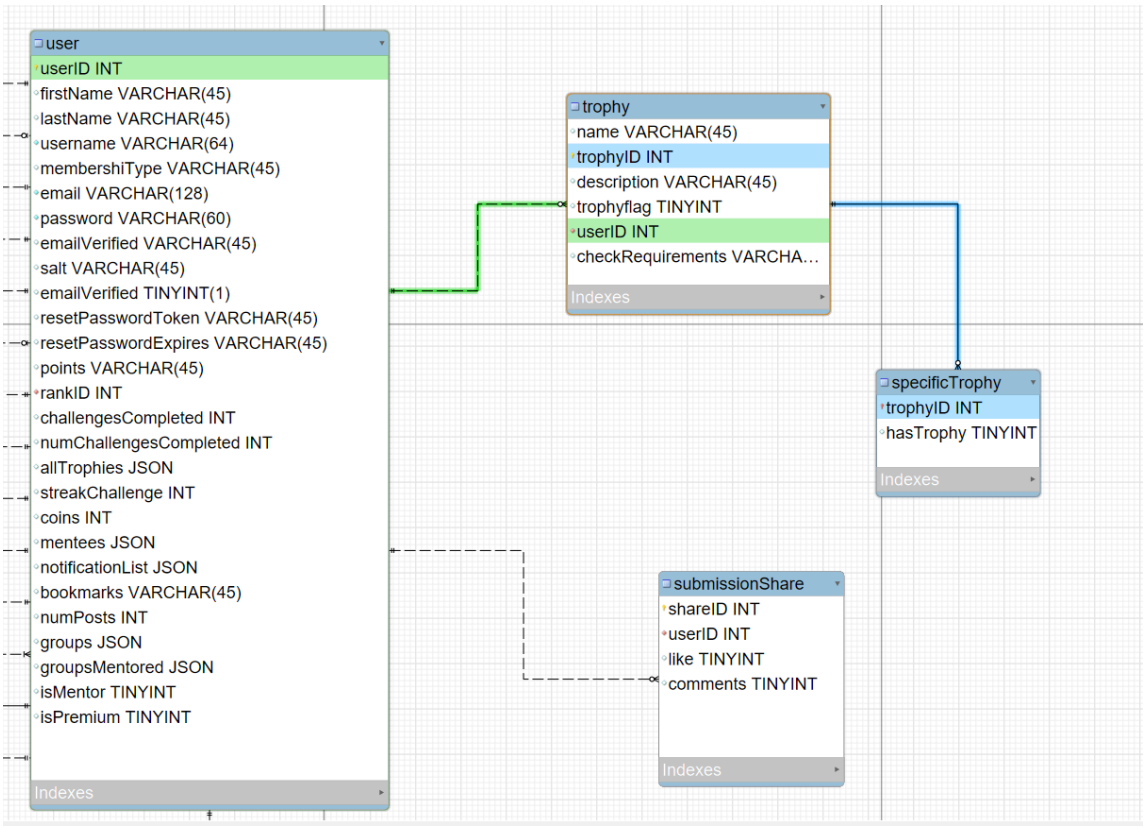
1.8. Notification



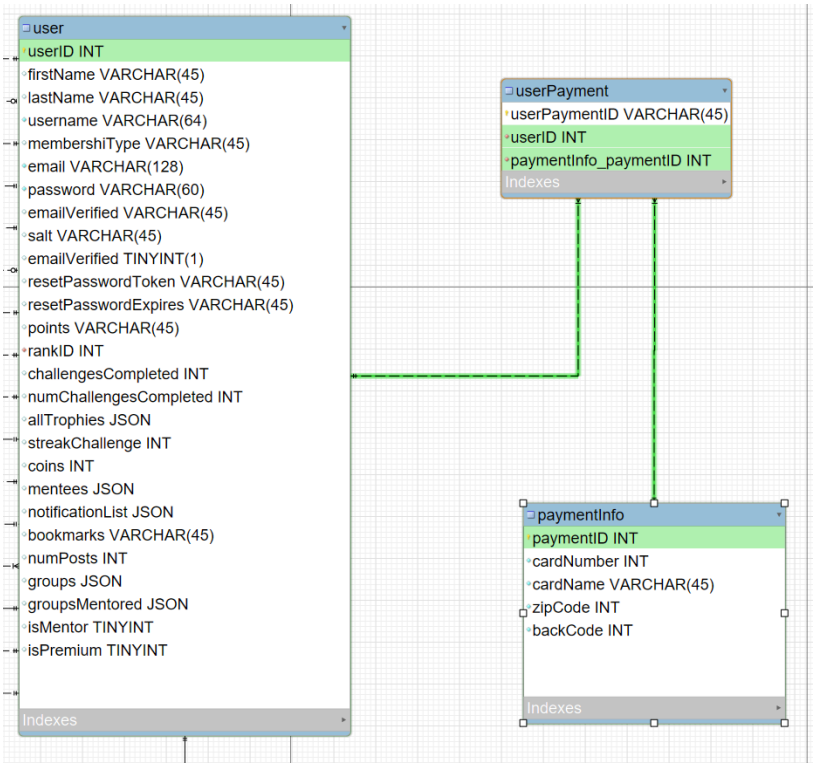
1.9. jobList and userView



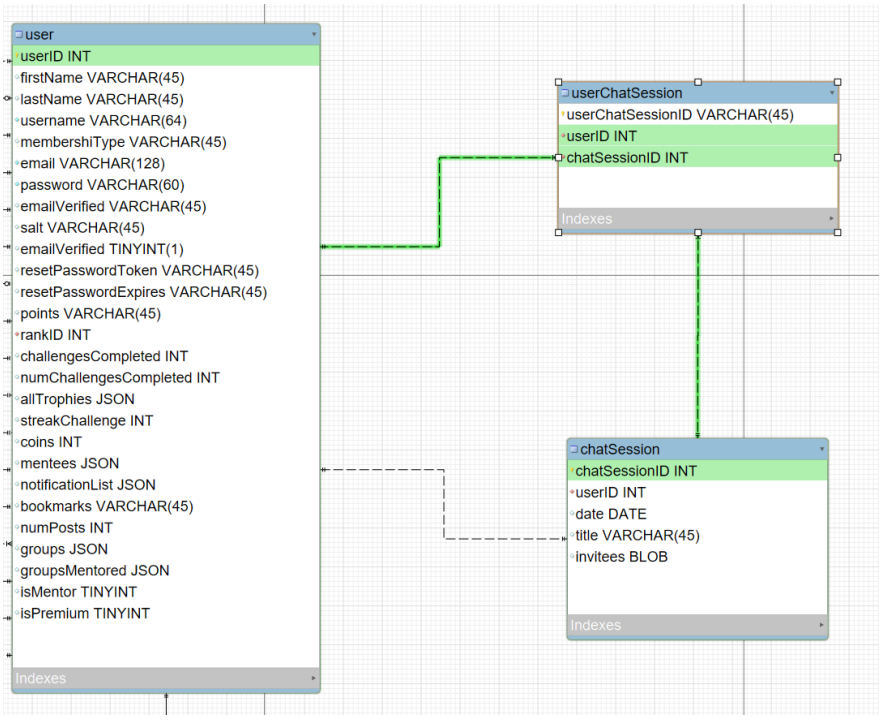
1.10. Trophy and submission share



1.11. Payment and userPayment



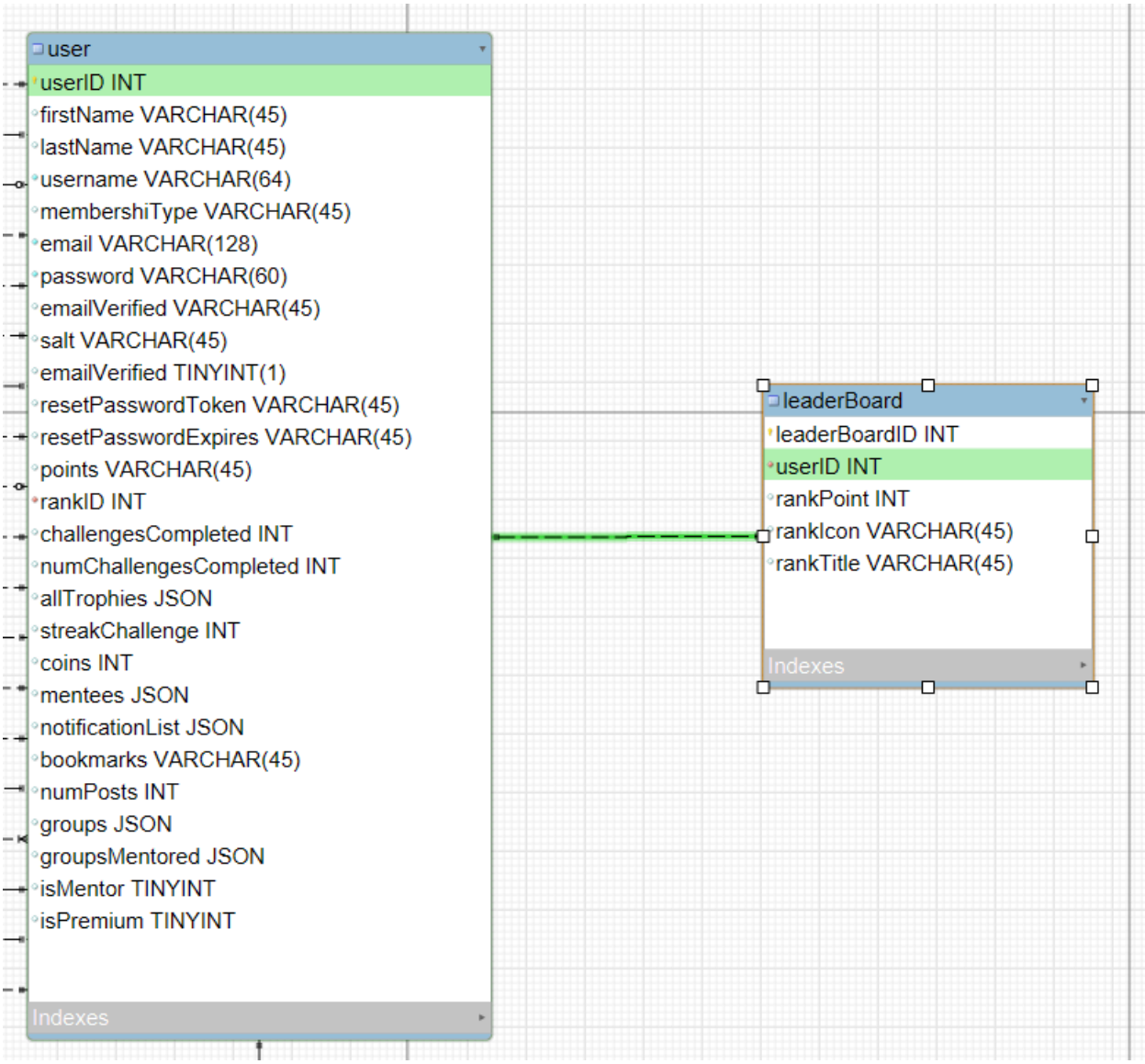
1.12. chatSession



1.13. leaderBoard (it need to updated

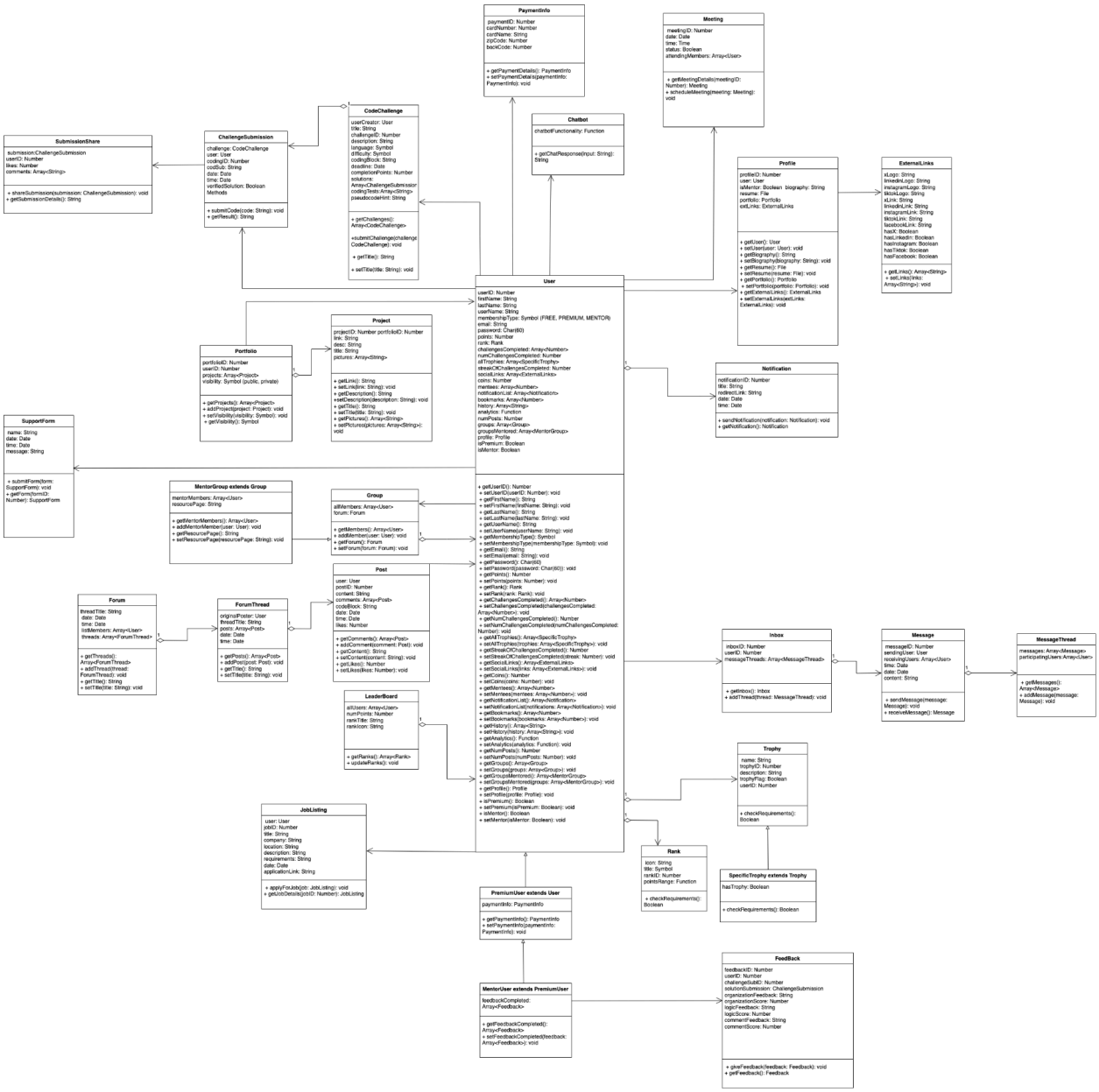
1.13.1. A leaderboard belong to many user

1.13.2. A user has one leaderboard

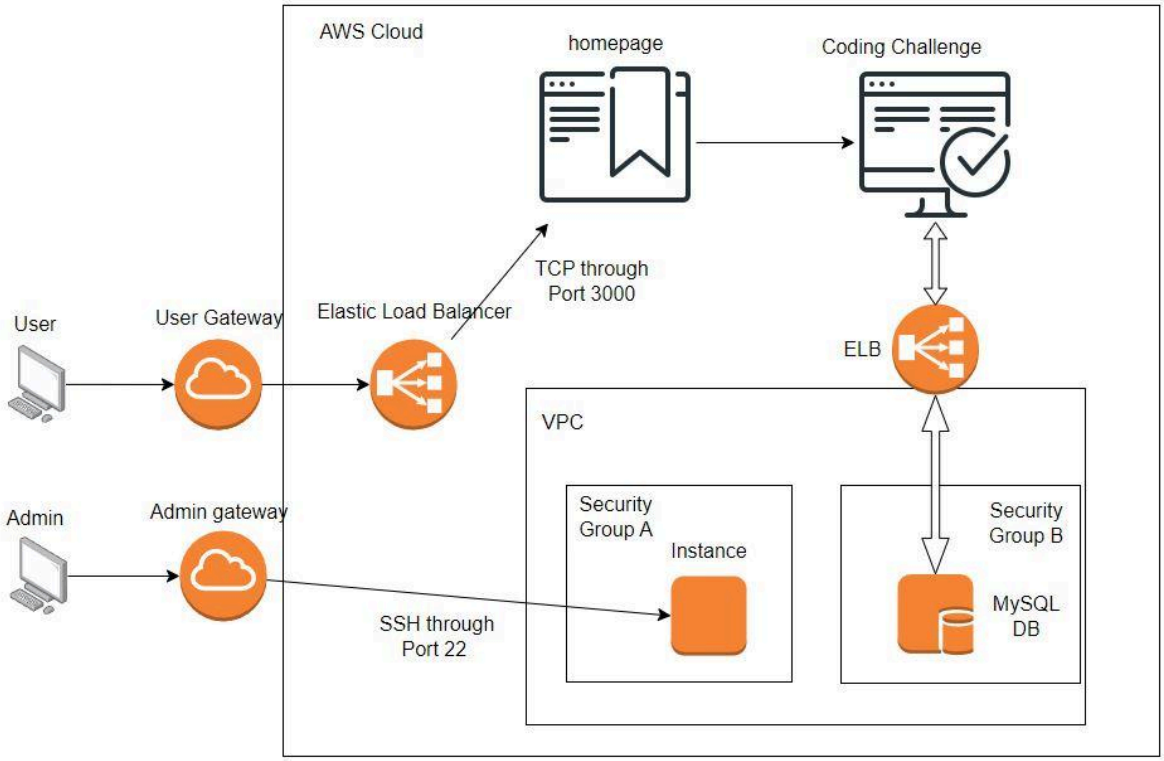


5. High Level Diagrams (detailed)

UML:



Network Diagram:



MILESTONE 3v1 FEEDBACK:

OUR FEEDBACK:

- Splash page needs to be the first page

SOLUTION: We need a consisten link: http:(IPv4):3000/index.html

- No search bar

SOLUTION: Search has a dedicated page, but searched through other pages along with the search function.

- Need to make user to agree with TOS and privacy

SOLUTION: Make a TOS script/page + create a TOS

- Get rid of home page in footer, header would just suffice

SOLUTION: ****

- Premium membership page (input parameters don't need to be so long use the space to explain advantages of subscribing)

SOLUTION: Adjust field sizes.

- Shorten the description in the splash page

SOLUTION: Already changed.

    - Change button
    - Add header to splash page

- Way to let the user to explore without creating a account

SOLUTION: Implement explore page from splash

- Might have to rethink the clicking the logo to get to home

SOLUTION: Logo will now direct to splash, home (header) leads to home page

- Likes our color scheme

MILESTONE 4v2:

SW Engineering CSC648-01 Summer 2024

CodeConnect

Team 1 - PikaDevs

Max Shigeyoshi - mshigeyoshi@sfsu.edu - Team Lead
Aaron Rayray - arayray@sfsu.edu - Github Master
Noah Hai - nhai@sfsu.edu - Doc Editor
Shez Rahman - srahman2@sfsu.edu - Backend Lead
Ghadeer Al-Badani - galbadani@sfsu.edu - Backend
Majd Alnajjar - malshemari@sfsu.edu - Frontend
William Pan - wpan1@sfsu.edu - Database Admin
Phillip Ma - pma1@mail.sfsu.edu - Frontend Lead

“Milestone 4“

7/23/2024

History Table:

Date	Changes
7/31/2024	Added URL to Product summary and updated product summary
7/31/2024	Included screenshot of code review for our own repo

1) Product summary

http://54.153.3.191:3000/index

CodeConnect serves as a superior platform for coders to enhance their technical skills and to build strong connections within the tech community. CodeConnect user profiles celebrate their journeys and progress by highlighting skills and interests with projects they've worked on, courses or challenges they've completed, communities they are a part of, and more. Through our website, coders can join groups and participate in forums targeted towards niche technical interests, and become deeper rooted in their communities through events and social gatherings promoted by other users through the website. The website encourages a diversity of different skill levels by enabling mentorship programs with different tiers of subscriptions at the discretion of the mentors, who are also encouraged to take on mentees through their own points and ranking system as mentors. This encourages people of all skill levels to connect with each other, enabling the upward mobility of all users. Mentorships can work through a variety of mediums including group classes, 1 on 1 sessions, prerecorded tutorials, resume advice, or mock interviews.Our main audience consists of rising coders, whether it be a hobbyist or a student in university. Our services go beyond just beginners, our target audience extends much further going to experienced developers working in a tech related field or even instructors that teach coding. By integrating both educational and social interaction, CodeConnect offers a unique oppurtunity for our audience and sets it apart from other platforms. Our marketing strategy consists of digital campaigns, partnerships and engaging within the community. We plan to enhance our website with social media marketing and create captivating content to attract traffic to our platform. By implementing partnerships with institutions such as schools and tech companies, we will further expand our reach while at the same engaging with events in the tech industry and conferences which will further increase our visibility to the public. By collaborating with online communities and hosting events online we will build a strong user foundation and offer a referral program that will give incentive to current users to join. CodeConnect will operate on a free model, offering a free tier with basic features, and a premium subscription option with more advanced features. We will also be providing custom mentorship and training programs and will get a dedicated team to reach out to schools, tech companies, coding bootcamps, etc to promote our platform. By using this approach of marketing and sales, CodeConnect aims to become the number one platform for coders of all skill levels. Our vast accumulation of resources and opportunities to grow make us the number one option for users looking to thrive in the tech industry, which is why users should join CodeConnect today and take the first step in becoming a better coder.

- **Name of the product**
  CodeConnect
- **Committed Functional Requirements:**

1. **<u>All Users:</u>**

*1.1 Users shall be able to explore some portions of the product without a profile
*1.2 Users shall be able to solve an example problem.
*1.3 Users shall create a profile
*1.4 Users shall be able to Log in/Log out (only with created profile)
*1.6 Users shall be able to upload profile picture
*1.7 Users shall be able to Delete profile
*1.8 User shall be able to update payment information
*1.12 Users shall be able to check other users profiles/stats
*1.13 Users shall be able to do coding challenges
*1.15 Users shall be able to award different profile trophies for achievements
*1.17 Users shall be able to earn points for coding challenges
*1.20 Users shall be able to check their coding ranking
*1.21 Users shall be able to check leaderboards
*1.22 Users shall be able to subscribe to Premium
*1.23 Users shall be able to unsubscribe to Premium
*1.32 Users shall be able to display other socials on their profiles
*1.34 Users shall be able to request additional features from the dev team
*1.35 Users shall be able to utilize live chat w/ other users
*1.36 Users shall be able to direct message other users
*1.38 Users shall be able to check coding streak counter of challenges
*1.39 Users shall be able to create a portfolio
*1.40 Users shall be able to check/update portfolio
*1.41 Users shall be able to change portfolio visibility (public/private)
*1.42 Users shall be able to access portfolio review
*1.45 Users shall be able to submit support forms to submit feedback regarding the app
*1.49 Users shall be able to view featured users (spotlight user that's completed most challenges)
*1.54 Users shall be able to join group session workshops led by mentors
*1.60 Users shall be able to get assessed to become a mentor
*1.61 Users shall be able to use a compiling development environment without having to create an account
*1.65 Users without premium shall be able to view three solutions per month
*1.66 Users shall be able to see the difference between premium and free membership perks
*1.67 Users shall be able to gain points by starting forum threads
*1.68 Users shall be able to gain points by commenting in threads
*1.69 Users shall be able to gain points by gaining friends
*1.70 Users shall be able to gain points by gaining mentees
*1.71 Users shall be able to gain points by gaining mentors
*1.72 Users shall be able to start new forum threads

*1.95 Users shall be able to reply to existing forum threads
*1.96 Users shall be able to have private forums, also known as groups
*1.97 Users shall be able to join mentor forums with mentees sharing common mentor
*1.76 Users shall be able to search for other users
*1.77 Users shall be able to search for forums also known as groups
*1.78 Users shall be able to receive notifications for messages.
*1.99 Users shall be able to receive notifications for replies to their threads
*1.100 Users shall be able to view information about mentors before selecting them
*1.81 Users shall be able to bookmark forum threads
*1.82 Users shall be able to report inappropriate content.
*1.90 Users shall be able to subscribe to notifications for specific forums or groups
*1.93 Users shall be able to view a list of job listings from home page
*1.94 Users shall be able to view other users' activity from their homepages
*1.101 Users shall be able to raise their rank by crossing point thresholds
*1.102 Users shall be able to click on other users' profiles from leaderboard
*1.103 Users shall be able to click on other users' profiles from their forum posts
*1.104 Users shall be able to have profile icon pictures
*1.105 Users shall be able to meet other users by joining a workspace
*1.106 Users shall be able to create a new workspace
*1.107 Users shall be able to add a password to a new workspace
*1.108 Users shall be able to filter mentors by category

2. **Free Users**
   *2.2 Free Users shall be able to check code challenge repo
   *2.3 Free users shall be able to view three pseudocode hints per month
   *2.4 Free users shall be able to view public forums
   *2.5 Free users shall be able to view other profiles
   *2.6 Free users shall be able to create an account
   *2.7 Free users shall be able to view a splash page to inform them about the website

3. **Premium Users**
   *3.0 Premium Users shall be able to check a single system chosen solution after completing a challenge
   *3.2 Premium Users shall be able to request mentorship (premium)
   *3.3 Premium Users shall be able to match with mentors based on coding experience
   *3.4 Premium users shall be able to view one pseudocode hint per challenge
   *3.7 Premium Users shall be able to request code review from a mentor
   *3.8 Premium users shall be added to a mentor's forum group upon mentor acceptance
4. **Mentor Users**

*4.1 Mentor Users shall be able to review code solutions of other users they are mentoring.
*4.4 Mentor Users shall be able to review Free/Premium users resumes/portfolios
*4.5 Mentor users shall be able to complete challenge feedback on coding challenges completed by mentees
*4.6 Mentor users shall be able to gain points by completing challenge feedback on mentee solutions
*4.7 Mentor Users shall be able to upload videos
*4.8 Mentor users shall have their number of mentees displayed on their profiles
*4.9 Mentor users shall have their number of solutions reviewed displayed on their profiles
*4.10 Mentor users shall be able to receive requests for mentorship by other users
*4.11 Mentor users shall have their own forum that only their mentees can view
*4.12 Mentor users shall be able to accept or deny users who request mentorship
*4.13 Mentor users shall have themselves listed on the mentorship page
*4.14 Mentor users shall have a unique title when posting in their mentor forums
*4.15 Mentor users shall be able to have information about them listed on the find a mentor page

2) Usability Test Plan

1. **Purpose of the Usability Test Plan**
    - **1.1. Function 1: ChallengeSub**
        - **The purpose of this test is for user to completed a challenge and submit it without encounter Error**
    - **1.2. Function 2: send_mentor_request_email**
        - **The purpose of this test is for users to register becoming a mentor without error.**
    - **1.3. Function 3: ExplorePage (Search)**
        - **The purpose of this test is for users to locate the forums users want.**
    - **1.4. Function 4: upload video**
        - **Ensure users can easily upload and submit videos.**
    - **1.5. Function 5: Forum Posts**
        - **Verify users can successfully create, view, and interact with forum posts.**

2. **Problem Statement and Objectives**
    - **2.1. Function 1: ChallengeSub**
      Test Objectives:
        - **Successfully submit the challenge**

Problem Statement - Starting point (where the user begins within the system.)

- **Starting it from the home page of our application that run on a local server**

Intended Users:

- **Users looking to hone their skills by doing a challenge**

URL of the System:

- **http://localhost:3000/homepage.html**

2.2. Function 2: send_mentor_request_email

Test Objectives:

- **Register to become a mentor and receive pending email**

Problem Statement - Starting point (where the user begins within the system.)

- **Starting it from the home page of our application that run on a local server**

Intended Users:

- **Users looking to teach/mentor users**

URL of the System:

- **http://localhost:3000/homepage.html**

2.3. Function 3: ExplorePage (Search)

Test Objectives:

- **Comprehensive list of every major page and file on the website, with filter**

Problem Statement:

- **Starting it from the home page of our application that run on a local server**

Intended Users:

- **Users unfamiliar with the tabs and links, want to look up the page**

URL of the System:

- **http://54.153.3.191:3000/explore.html**

2.4. Function 4: upload video

Test Objectives:

- **Users can complete and select then upload a video file..**

Problem Statement - Starting point (where the user begins within the system.)

- **Starting it from the page of upload video section**
- **This is a separated test prototype with the applicated**

URL of the System:
http://127.0.0.1:5500/application/samples/s3_sample/uploadfile.html

2.5. Function 5: Forum Posts

Test Objectives:

- **users can view and access forum posts without issues.**

Problem Statement - Starting point (where the user begins within the system.)

- **Starting it from the home page of our application that run on a AWS**

URL of the System:http://54.153.3.191:3000/index.html

3. **User Profile - Intended Users (Who is the user)**
   - 3.1. **Function 1: ChallengeSub**
     - **User is a student from computer science in his senior year**
     - **Want to practice more coding challenge for interview**
   - 3.2. **Function 2: send_mentor_request_email**
     - **User is a student from computer science in his Junior year**
     - **Want to have more hand on experience**
   - 3.3. **Function 3: ExplorePage (Search)**
     - **User is a graduated computer student,**
     - **who want to gain more knowledge about the latest CS trends.**
   - 3.4. **Function 4: upload video**
     - **Users who want to share or upload video content that related to Software**
     - **Have basic understanding of uploading media**
   - 3.5. **Function 5:Forum Posts**
     - **Users who participate in the forum or community**

4. Method (How the user do this task)

4.1. Function 1: ChallengeSub

- **The user is my classmate doing this at his dorm, who's having lots of knowledge about computer**

4.2. Function 2: send_mentor_request_email

- **The user is my roommate doing this at our dorm, who's bad about computers and have a bad GPA.**

4.3. Function 3: ExplorePage (Search)

- **The user is provided with the link of our appliaction and Connect through discord in different countries.**

4.4. Function 4: upload video

- **Starting the application by us from my laptop in the same room**

4.5. Function 5: Forum Posts

- **logs in to the home section from their laptop in different room**

5. Task List (List all the task user is testing)

5.1. Function 1: ChallengeSub

- **Navigated to difference challenge**
- **Completed some challenges**
- **The code submit from javaScript best practice challenge is compile**

5.2. Function 2: send_mentor_request_email

- **Navigated to mentor request page**
- **Completed all the field of the forum**
- **Receive the mentor pending email from codeconnect**

5.3. Function 3: ExplorePage (Search)

- **Navigated to forum post**
- **Located the desired topic user want to view**
- **Successfully to view the detail of the post**

5.4. Function 4: upload video

- **Find the video upload area.**
- **Choose a file to upload.**
- **Fill in title and description.**
- **Confirm the video is correctly uploaded.**

5.5. Function 5: Forum Posts

- **write and submit a new post.**

- **Read the forum posts from them**
- **Comment existing forum posts..**

6. **Test Environment - System Setup: (what type of machine)**
   - 6.1. **Function 1: ChallengeSub**
     - **Environment Description: The server is running it in background, the User is provided with a URL of the system on a Desktop.**
   - 6.2. **Function 2: send_mentor_request_email**
     - **Environment Description: The server is running it in background, the User is provided with a URL of the system on a laptop.**
   - 6.3. **Function 3: ExplorePage (Search)**
     - **Environment Description: The server is running on aws and user have the URL of the system and tested it on laptop**
   - 6.4. **Function 4: upload video**
     - **Environment Description: The server is running locally on laptop and the page is open for the user**
   - 6.5. **Function 5: point system**
     - **Environment Description: The server is running on aws and user have the URL of the system and tested it on laptop**
7. **Test Monitor Role (in which way you are monitor this test)**
   - 7.1. **Function 1: ChallengeSub**
     - **The user is in the same room using my computer**
   - 7.2. **Function 2: send_mentor_request_email**

- **Backend monitor the user in the difference room**

7.3. Function 3: ExplorePage (Search)

- **Backend monitor and talk during the task via discord**

7.4. Function 4: upload video

- **Backend monitor and observed the user beside**

7.5. Function 5: Forum Posts

- **Backend monitor and observed the user behind**

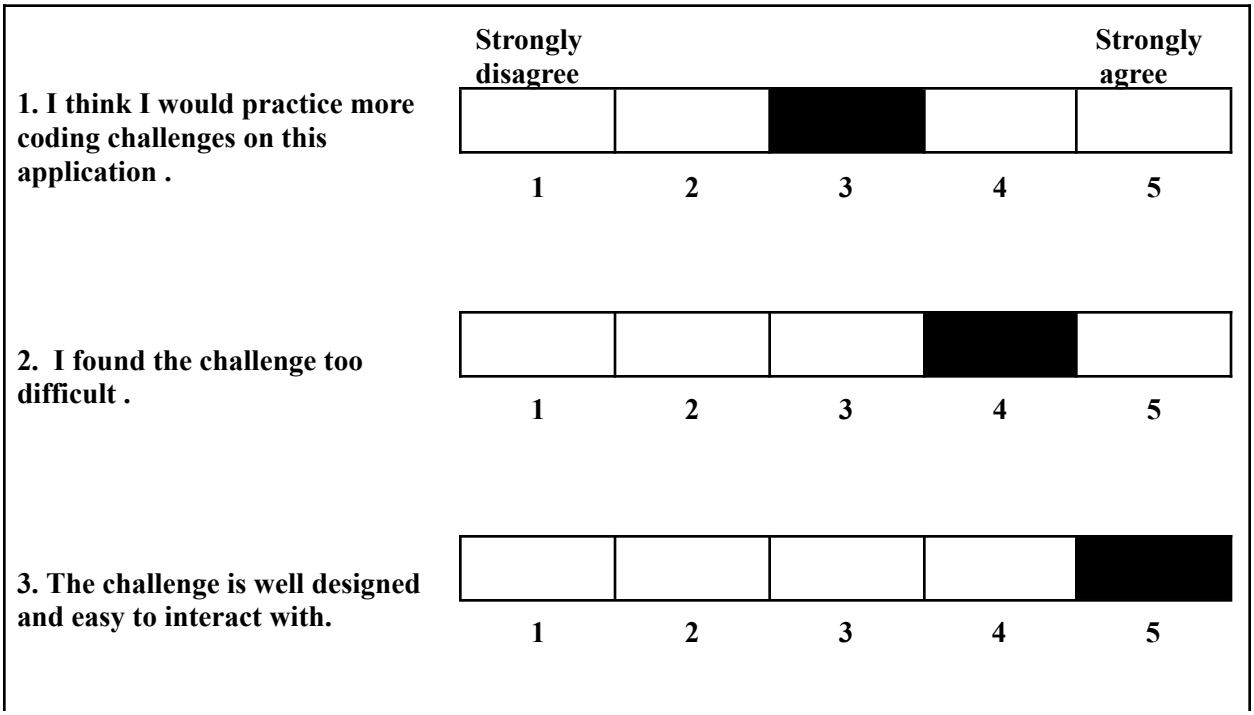
8. Usability test table : Evaluation measures (what is to be measured)

Efficiency: compute approximate average time it took the user to completed task

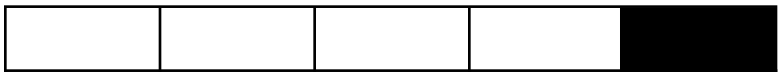
# . Function	Test/use case	% complete	Estimated efficiency	Error from user	comments
1. ChallengeSub	Completed as much challenge as possible	70%	Navigated to the challenge page then finished tasks one by one six times. (~10 minutes)	The user do not know the question is completed and keep submitting	The user spend 25 minutes
2 send_mentor_request_email	Submitted the become challenge form	90%	Navigated to become a mentor page and filled 10 field would properly take less than 10 minutes	The user cannot find the mentor page and try navigated in membership and premium	It take user 15 minutes to competed the form and he do not know that there is a email sending

				page	
3 ExplorePage (Search)	**Located the desired forum via search bar**	**40%**	**Type in the desired topic in search bar then choose the topic want to view(< 3 minutes)**	**The user cannot find the desired topic and sometimes it has no post.**	**When user enter topic that isn't relative would not populated anything (>10 minutes)**
4 upload video	**select and upload a video file.**	**80%**	**Choice the desire video and type in the topic (< 1 minutes)**	**The user could not view what video is post**	**After the user Completed, there is nothing changed on the page.**
5 Forum Posts	**Submitting replies, upvotes, and downvotes**	**60%**	**Filled out the field and navigated to forum post (~ 10 minutes)**	**The user could not find the post in wrong forum**	**The user cannot view his own post via search bar**

9. User satisfaction (3 different statements per function tested)
Ask statement not question

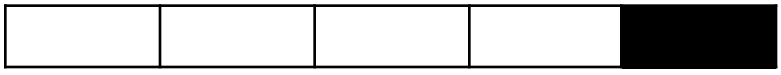


4. The user interface is clear and easy to understand how to completed the become mentor form



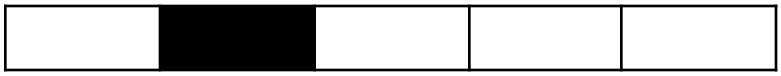
5. I found the become mentor form filed unnecessarily complex

6. I think I would need contact technical support for submit the become mentor form



7. I was able to quickly find and view the topic I was searching for

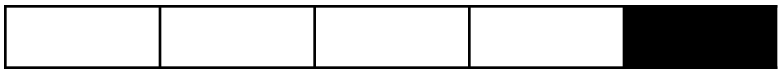
8. The search functionality on the Explore Page provided relevant and accurate results.

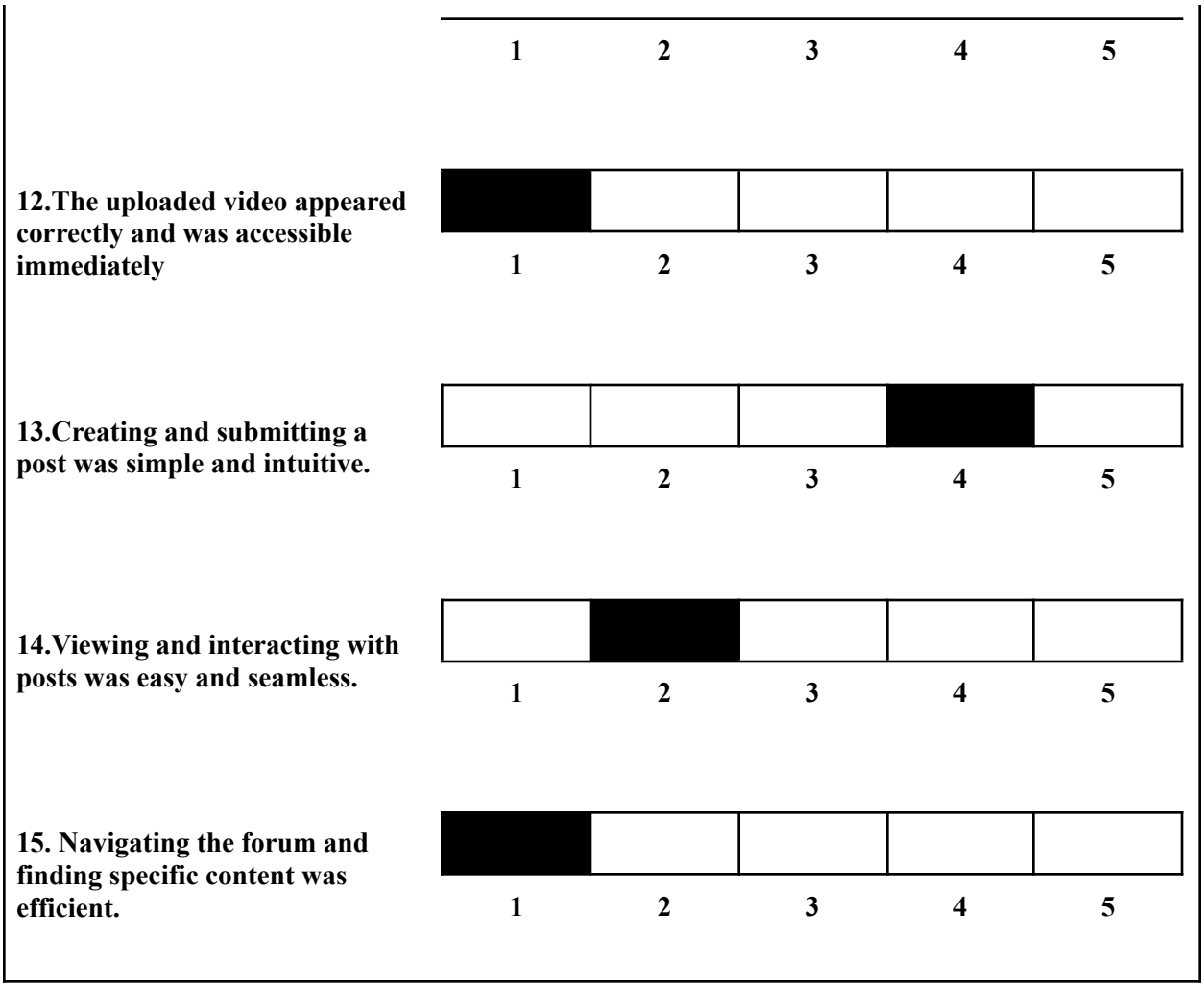


9.Navigating from the search results to the forum post was straightforward and easy to understand.

10. I think uploading a video was quick and straightforward.

11.The metadata fields were easy to fill out





3) QA test plan

1. Test sequences - security

   Test objectives: Passwords should be saved using at least a 256 bit hashing algorithm like SHA-256.

   HW and SW setup:

   Hardware:

   - **AWS**

   Software:

   - Window(OS) , MYSQL (Database) ,ExpressJS (NodeJS) ,React (Javascript)

   Application URL:

   http://54.153.3.191:3000/index

   Feature to be tested: Passwords should be saved using at least a 256 bit hashing algorithm like SHA-256.

QA Test Plan:

Number	Description	Test Input	Expected Output	Pass/Fail
1	**User passes greater than minimum password length during sign up**	**StrongPassword123!**	**At least 256 bit hashing algorithm**	**Pass**
2	**User passes minimum password length during sign up**	**Smalest!**	**At least 256 bit hashing algorithm**	**Pass**
3	**User passes less than password length during sign up**	**Small1!**	**No signup creation**	**Pass**

2. Test sequences - Security

Test objectives: The website should provide a forget me functionality to allow password resets. The generated token must be encrypted using an at least 32 characters random secret.

HW and SW setup (including URL):

Hardware:

- **AWS**

Software:

- Window(OS) , MYSQL (Database) ,ExpressJS (NodeJS) ,React (Javascript)

Application URL:

http://54.153.3.191:3000/index

Feature to be tested: The website should provide a forget me functionality to allow password resets. The generated token must be encrypted using an at least 32 characters random secret.

QA Test Plan:

Number	Description	Test Input	Expected Output	Pass/Fail

1	**User submits nonexistent email**	**fakeemail**	**Invalid email**	**Pass**
2	**User submits existing email and valid reset password**	**shezpc@gmail.com, NewStrongPassword123!**	**Password reset email has been sent, can click the link and reset password successfully**	**Pass**
3	**User submits existing email and invalid reset password**	**shezpc@gmail.com, invalid**	**Password reset email is sent, but password is not reset.**	**Pass**

3. Test sequences - XXXX

Test objectives: Email verification and password reset urls should use tokens that expire

HW and SW setup (including URL):

Hardware:

- **AWS**

Software:

- Window(OS) , MYSQL (Database) ,ExpressJS (NodeJS) ,React (Javascript)

Application URL:

http://54.153.3.191:3000/index

Feature to be tested: Email verification and password reset urls should use tokens that expire

QA Test Plan:

Number	Description	Test Input	Expected Output	Pass/Fail
1	**User inputs email to sign up, token is sent**	**johndoe123@gmail.com**	**Email verification**	**Pass**

	and its clicked before expiring		**sent, link brings you to home page**	
2	**User inputs unregistered email to reset password, token is sent and its clicked before expiring**	**johndoe123@gmail.com**	**Email verification sent, link brings you to "reset password" page**	**Pass**
3	**User inputs unregistered email to reset password, token is sent and its clicked before expiring**	**mrjohndoe123@gmail.com**	**Email verification is not sent**	**Pass**

4. Test sequences - XXXX

Test objectives: After a sign up, an email verification should be sent and prevent the user from signing in until verified.

HW and SW setup (including URL):

Hardware:

- **AWS**

Software:

- Window(OS) , MYSQL (Database) ,ExpressJS (NodeJS) ,React (Javascript)

Application URL:

http://54.153.3.191:3000/index

Feature to be tested: After a sign up, an email verification should be sent and prevent the user from signing in until verified.

QA Test Plan:

Number	Description	Test Input	Expected	Pass/Fail

			Output	
1	**Sign up sends email verification and user tries to signup without verifying**	**Valid email and password**	**User cannot sign in**	**Fail**
2	**Sign up sends email verification and user tries to signup after verifying**	**Valid email and password**	**User can sign in**	**Pass**
3	**Sign up with existing email**	**Already existing email in database**	**User cannot create new account**	**Pass**

5. Test sequences - **Storage**

Test objectives: Job posts shall be in their own database, linked to user_ID.
HW and SW setup (including URL):
Hardware:

- **AWS**

Software:

- Window(OS) , MYSQL (Database) ,ExpressJS (NodeJS) ,React (Javascript)

Application URL:

http://54.153.3.191:3000/index

Feature to be tested:

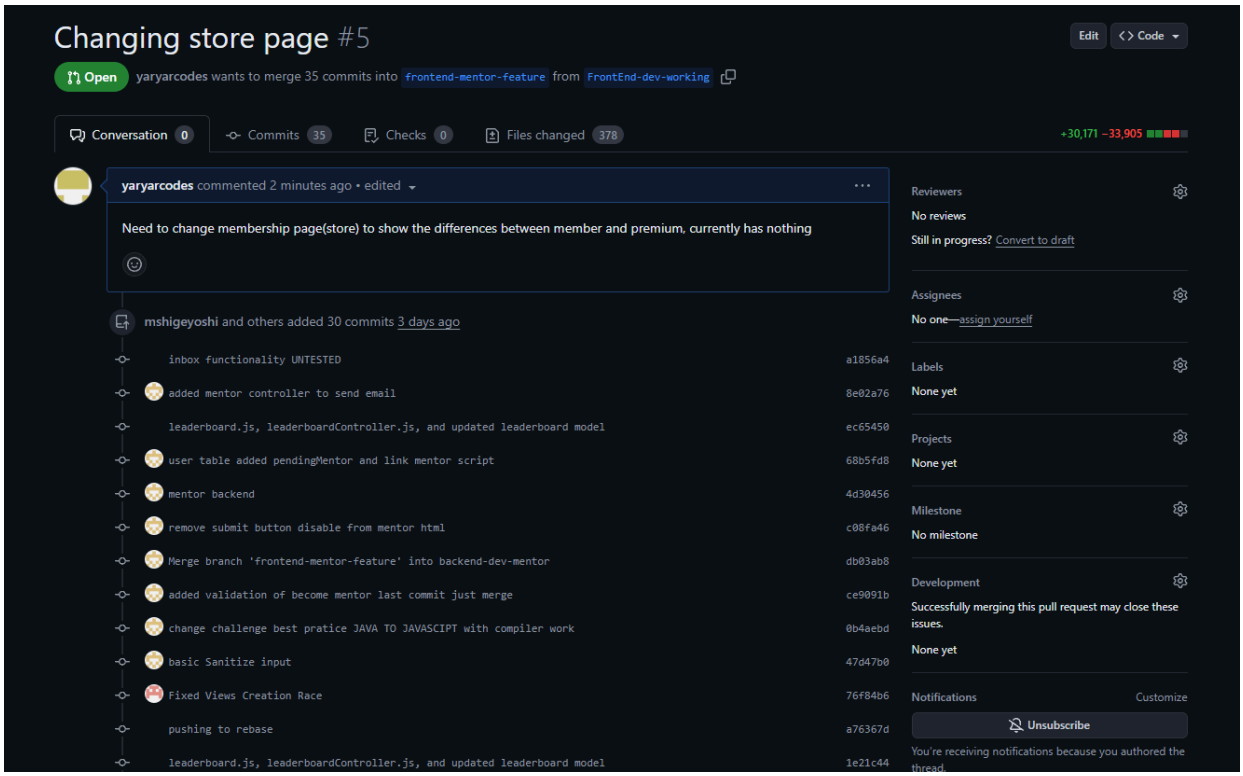
Job posts must be stored in a dedicated database that is distinct from other data. Each job post should be linked to a user_ID, ensuring that each post is associated with the user who created it.

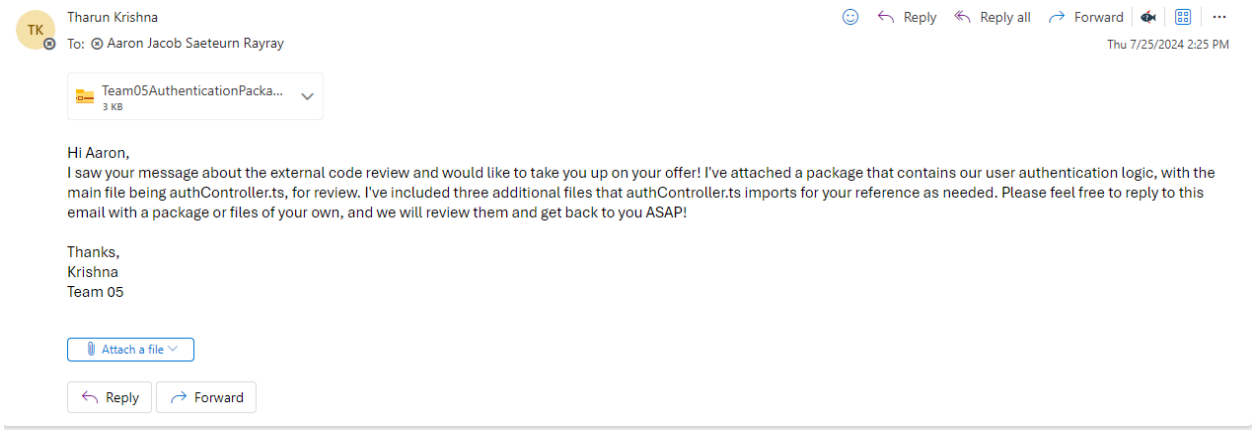
QA Test Plan:

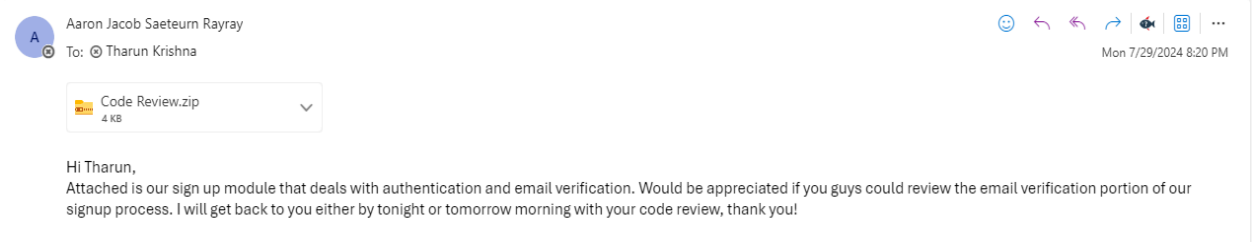
Number	Description	Test Input	Expected Output	Pass/Fail

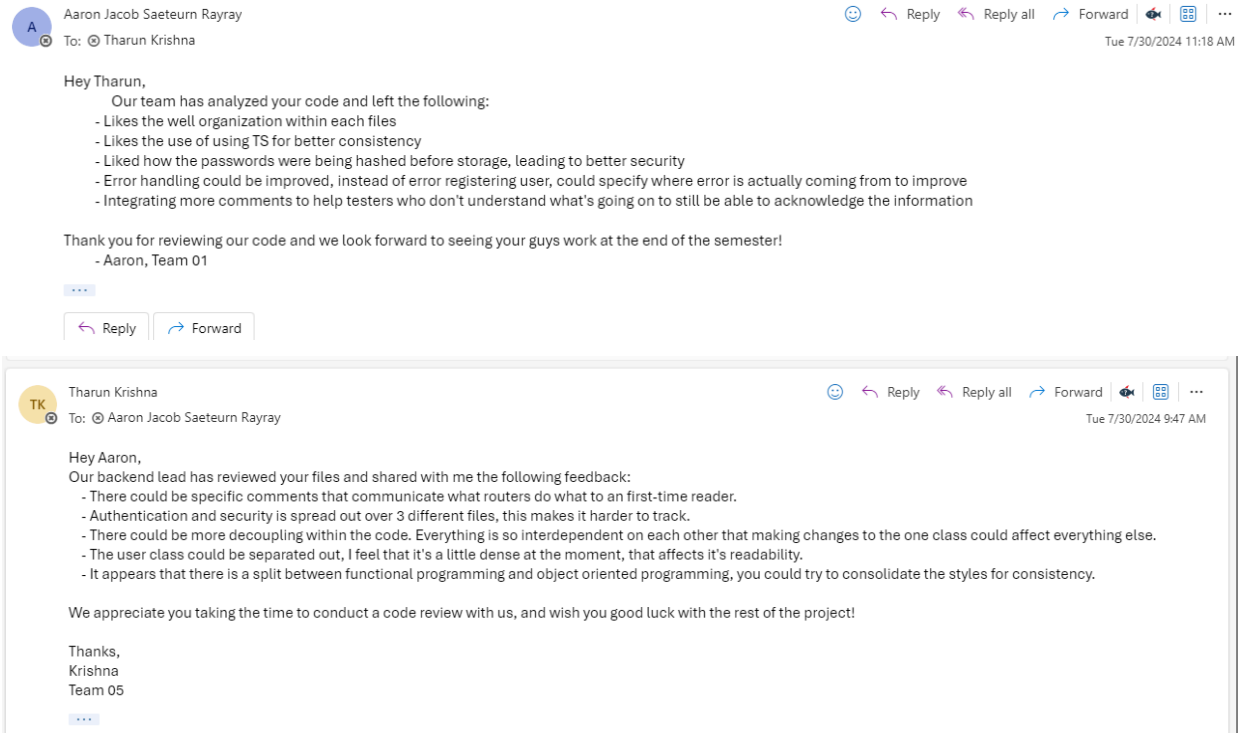
1	**Check if a new job post is correctly added to the job posts database and linked to the user_ID.**	**Create a new job post and submit it. And look at the Console output**	**The job post should appear in the job posts database with the correct user_ID that match with the console**	**Pass**
2	**Ensure that job posts can be retrieved from the database using the user_ID.**	**Query job posts using a specific user_ID.**	**The retrieved job posts should match those created by the specified user_ID.**	**Fail**
3	**Confirm that job posts are stored separately from user information in different databases.**	**Check database schemas and tables.**	**Job posts should be in their own database, separate from user data.**	**Pass**

4) Code Review:









Aaron Jacob Saeteurn Rayray
To: Tharun Krishna
Reply Reply all Forward
Tue 7/30/2024 11:18 AM

Hey Tharun,
Our team has analyzed your code and left the following:
- Likes the well organization within each files
- Likes the use of using TS for better consistency
- Liked how the passwords were being hashed before storage, leading to better security
- Error handling could be improved, instead of error registering user, could specify where error is actually coming from to improve
- Integrating more comments to help testers who don't understand what's going on to still be able to acknowledge the information

Thank you for reviewing our code and we look forward to seeing your guys work at the end of the semester!
- Aaron, Team 01

Reply Forward

Tharun Krishna
To: Aaron Jacob Saeteurn Rayray
Reply Reply all Forward
Tue 7/30/2024 9:47 AM

Hey Aaron,
Our backend lead has reviewed your files and shared with me the following feedback:
- There could be specific comments that communicate what routers do what to an first-time reader.
- Authentication and security is spread out over 3 different files, this makes it harder to track.
- There could be more decoupling within the code. Everything is so interdependent on each other that making changes to the one class could affect everything else.
- The user class could be separated out, I feel that it's a little dense at the moment, that affects it's readability.
- It appears that there is a split between functional programming and object oriented programming, you could try to consolidate the styles for consistency.

We appreciate you taking the time to conduct a code review with us, and wish you good luck with the rest of the project!

Thanks,
Krishna
Team 05

5) Self-check on best practices for security –½page

For our project, we implemented many security features that are considered best practices. The first was to make our EC2 instance only accept traffic from certain ports, this would limit the types of interactions our instance can have, making it difficult for other types of unwanted traffic to reach our product. Next, we focused on keeping our database secure by only allowing privileged users access to the database itself. This keeps our database secure and doesn't allow for multiple users to access and alter the database in any way we might not want. Any user generated information that is created that needs to be stored in the DB is handled by our coding standard which is an ORM, or object-relational mapping. This stops any user from injecting SQL queries into our database, so they can't query information they aren't allowed to have. The DB is also encrypted at reset, so that for any reason, if we have to update the database, everything is encrypted again. We also hash sensitive information, along with salting, in order to protect those pieces of data, using SHA256. Within our code we utilize boom in order to handle our errors, so the errors are not sent back to the user, we do send them a similar error message, but not the logs in total. For storing different kinds of data, such as videos, solutions, or other larger files, we utilized Amazons S3 storage module. This module has a bunch of security features, but is not accessible publicly, and only has one privileged user. With these security features in place, we believe that our product is sufficiently secure, but there are always more possibilities for improvement in the future.

Major Assets:

- User password
- S3 protects user generated content (videos, text, solutions)
- DB Access
- DB Queries
- DB Encryption
- EC2 Access
- Boom for error handling

Validations:

- User email verified check + using boom for error message handling

```
exports.verify_email_get = asyncHandler(async (req, res, next) => {
    try {
        const email = utils.token.get_verify_email(req.params.token);
        const user = await User.findOne({
            where: {
                email: email
            }
        });
        if (user) {
            user.set({
                emailVerified: true
            });
            await user.save();
        // res.send(email);
        // req.url = "/";
            res.status(301).redirect("/");
        } else {
            throw boom.notFound("User not found");
        }
    } catch (err) {
        throw boom.notFound(err.message);
    }
});
```

- User password strength:

Read more about this library https://www.npmjs.com/package/check-password-strength

```
//Read more about this library https://www.npmjs.com/package/check-password-strength
const { passwordStrength } = require('check-password-strength')
```

- Duplicate email per user
- User passwords hashed

```
if (existingEmailUser) {
    throw boom.badData("Email already in use");
} /*else if (existingUsernameUser) {
    throw boom.badData("Username already in use");
}*/ else if (passwordStrength(password).id <= 1) {
    throw boom.badData(`Password is ${passwordStrength(password).value}`);
} else {
    try {
        const salt = await bcrypt.genSalt(10);
        const hashedPassword = await bcrypt.hash(password, salt);
        console.log(salt);
        console.log(hashedPassword);
        const user = User.build({
            email: email,
            //username: username,
            password: hashedPassword,
            emailVerified: false
        });
        await user.save();

    } catch {
        throw boom.internal(err.message);
    }
    try {
        send_verify_email(email);
```

- User sessions

```
 //Middleware to authenticate the user's cookie token
const authCookieTok = (req, res, next) => {
    const token = req.cookies.token;
    if (!token) {
        throw boom.unauthorized("Token not found");
    }

    jwt.verify(token, process.env.TOKEN_SECRET, (err, detokenized) => {
        if (err) {
            throw boom.unauthorized("Invalid token");
        }
        req.userID = detokenized.id;
        next();
    });
};
```

- Sending User and recieving user for messages

```
exports.inbox_message_post = asyncHandler(async (req, res) => {
    const { sendingUserID, receivingUserID, content } = req.body;

    const sendingUser = await User.findOne({
        //parse string into first and last name
        where: {
            userID: sendingUserID
        }
    });

    const receivingUser = await User.findOne({
        where: {
            userID: receivingUserID
        }
    });

    //Verify both sender and receiver are real users in the DB.
    if (!sendingUser || !receivingUser) {
        throw boom.unauthorized("Sending or Receiving User not found");
    }

    console.log('Sending User:', sendingUser);
    console.log('Receiving User:', receivingUser);
```

- S3 Storage has built in features the provided the security such as by default the objects inside the S3 container are private, automatic data encryption, https://aws.amazon.com/s3/security/

6) Self-check: Adherence to original Non-functional specs

1. **Database Specs**
   Status: DONE ,ON TRACK , ISSUE (drop need explain e.g we don't need)

Non-functional	**Status**
1.1 Databases should use transparent data encryption (TDP) to encrypt data at reset with AES.	**DONE**
1.2 Databases should dynamically mask and de-identify users' PII information for database users and accessed only for privileged ones.	**ON TRACK**
1.3 Critical databases should automatically save backups preferably once a day.	**DONE**

1.4 Database's Backups should be kept for at least 7 days in a cost effective storage solution like AWS S3 bucket.	**DONE**
1.5 Databases should store audit logs according compliance standards and have low overhead on the performance.	**ON TRACK**
1.6 Any database audit logs should be filtered for sensitive data, encrypted, and compressed.	**DONE**
1.7 Website databases should be able to scale vertically and horizontally with less than 2 hours downtime.	**ON TRACK**
1.8 App databases should be able to run when needed, in a high availability cluster (HA) with automated failover and fault tolerance.	**ON TRACK**
1.9 Any database's super user (SA) and backend user should have at least 32 random hex characters password.	**ON TRACK**

2. **Storage**

Non-functional	**Status**
2.1 Premium users shall have all the same information from a free user, but will also have payment information included.	**ON TRACK**
2.2 Job posts shall be in their own database, linked to user_ID.	**DONE**
2.3 We shall only have one user database, which indicates what type of user (free, premium, mentor) that user is.	**DONE**
2.4 Payment information will be in its own database, linked to a user_ID.	**DONE**
2.5 Forum posts shall be housed in their own database, linked to user_ID.	**DONE**
2.6 user_ID is used to tie all posts to a single user on the backend, but the username will be used for searching purposes.	**DONE**

3. **Security**

Non-functional	**Status**
3.1 Passwords should be saved using at least a 256 bit hashing algorithm like SHA-256.	**DONE**
3.2 Any database that stores payment information must be PCI compliant. If standards could not be met, they can be stored with a PCI certified external provider.	**ON TRACK**
3.3 Passwords should be hashed with at least 128 bytes of random generated salt to prevent rainbow tables attack.	**DONE**
3.4 Website backend should connect to the database with SSL in case they are in different instances.	**ON TRACK**
3.5 The website should use generated SSL certificates to secure data in transit.	**ON TRACK**
3.6 Website backend or load balancer if used should prevent HTTP connections and forward them to HTTPS.	**ON TRACK**
3.7 Website backend should verify requests source domain with CORS.	**ON TRACK**
3.8 Website backend should set required headers to prevent cross site scripting.	**ON TRACK**
3.9 Website backend should be able to run on multiple servers using a load balancer.	**ON TRACK**
3.10 Website backend should be able to generate a json web token (JWT) that expires in 8 hours or 60 days to keep me signed in logins.	**DONE**
3.11 The website should provide a forget me functionality to allow password resets.	**DONE**

3.12 The generated token must be encrypted using an at least 32 characters random secret.	**DONE**
3.13 The generated token should include the unique user id to facilitate authorization.	**ON TRACK**
3.14 Required authenticated requests should use browser cookies to store the token.	**DONE**
3.15 After a sign up, an email verification should be sent and prevent the user from signing in until verified.	**DONE**
3.16 Need an SSL in order to encrypt the network traffic from our website.	**ON TRACK**
3.17 The backend should be able to identify users from the previously mentioned tokens and reject expired ones.	**DONE**
3.18 The backend should be able to run at least 10 solutions parallelly in separate environments.	**ON TRACK**
3.19 The backend should be able to accept run requests even at max capacity and manage executions on a first come first serve basis.	**ON TRACK**

4. **Performance**

Non-functional	**Status**
4.1 A continuous integration and continuous delivery pipeline (CI/CD) should be developed after first release to facilitate fast, reliable updates.	**ON TRACK**
4.2 The email verification and password reset urls should use tokens that expire after 30 minutes.	**DONE**
4.3 Chat features should be in real-time and have minimal latency.	**ON TRACK**

4.4 The static front end files should be served from a content delivery network (CDN) to enable a fast global delivery with lower backend traffic overhead.The website shouldn't take more than a few minutes to verify a solution to a challenge problem.	**ON TRACK**
4.5 All rest APIs should respond in under 5 seconds in all situations.	**ON TRACK**
4.6 Logging into the website shouldn't take more than a few seconds.	**DONE**
4.7 Profile Rankings should be updated live, as a person may find some challenges easier than others and might complete a few of them quickly.	**ON TRACK**
4.8 If a profile does not have a picture associated with it, there will be a default picture placed.	**ON TRACK**
4.9 The user interface should be intuitive and user-friendly, requiring no more than 3-5 clicks to access primary functions	**ON TRACK**

5. **Expected Load**

Non-functional	**Status**
5.1 We would expect at least 1000 users at any given time.	**ON TRACK**
5.2 Leaderboards should be updated every 24 hours, as people's rankings may increase rapidly during that time.	**DONE**

6. **Compatibility and UI**

Non-functional	**Status**
6.1 The UI should scale and resize based on window size.	**DONE**
6.2 Logos should be in the header and footer of every page visited	**DONE**
6.3 All aspects of any web page concerning lists (such as lists of challenges) should be the largest portion of the webpage, except the challenges which will take up all of the page as it is the most important portion of the product.	**DONE**

6.4 Username is used mainly in the header to display to users they are logged in.	**ON TRACK**
6.5 Solutions should be graded within 3-5 minutes of submission.	**DONE**
6.6 Solutions should be accurate to all tests provided.	**ISSUE**
6.7 Solution review should only show the best solutions if the user is a premium member.	**ON TRACK**
6.8 User's machines shall run Windows and should work on any browser.	**DONE**
6.9 Links to other social media shall only be viewable on the web page of a user if the user provided those links to us.	**ON TRACK**

7) list of contributions

Max Shigeyoshi:(9/10)

- Created the inbox and messaging feature
- Created the leaderboard feature
- Assisted with git merge and resolving conflicts
- Updated the Milestone 3 document
- Assisted with integration frontend-backend
- Created demo for the frontend on how to use JSON objects
- Created Notion board for M4
- Updated the instance to run the master branch

Aaron Rayray:(9/10)

- Handled the Git merges and deletions to keep the git organized
- Handled the Code Review with the other teams
- Created Mentor frontend,i.e mentorvalidation, mentorrequest
- Handled organization within files on it’s respected branch
- Relocated each file to have better organization within the repo
- Redefined mentor page to have a search and filter
- Redefined find a mentor to lead back to forums(private messages)
- Assisted with integration frontend-backend

Noah Hai:(9/10)

- Created Leaderboard frontend
- Wrote Product summary
- Created Milestone 4 document

- Assisted with integration frontend-backend
- Edited premium page as per professors feedback
- Added in TOS and Privacy condition to sign up as per professors feedback
- Resolved css render/linking issues when launching app
- Updated search function in explore page

Shez Rahman:(9/10)

- Delegated backend team tasks
- Created the mentor feature
-
- Assisted with integration frontend-backend

Ghadeer Al-Badani:(9/10)

- S3 storage feature
- Reviewed backend code
- Updated the search and sorting feature
- Assisted with integration frontend-backend

Majd Alnajjar:(9/10)

- Created front end for Forum feature
- Assited in Leaderboard frontend
- Assisted with integration frontend-backend

William Pan:(9/10)

- Updated the DB
- Created the challenges feature
- Usability test plan
- Adherance to non functional requirments
- Assisted in mentor feature
- Usability test case
- Assisted with integration frontend-backend

Phillip Ma:(9/10)

- Delegated frontend team tasks
- Updated the Milestone 3 document
- Updated the Milestone 3 document
- Assisted with integration frontend-backend
- Assisted with Inbox frontend
- Assisted with Explore frontend

3)Post analysis

With the project submitted, there are a few things we would do differently as a team. First, we would not take this class during the summer, this class is an extremely significant course, and the added weeks that we are afforded by the Spring or Fall semesters would have been extremely useful. It also would have made it less stressful when it came time to switch gears as the corrections from one grading would impact our work in the current milestone. Sometimes we would have to wait for the grading to come back to see if we needed to change anything that would have altered our plans. Although the grader and professor would usually return the grades promptly, those few days where we didn't have those recommendations would be costly due to the less amount of time we had over the whole semester. Each day lost would be a significant amount of work and sometimes those overlaps would halt sections of the project.

Next, we would have taken more time at the beginning of the project to onboard each member of the team so that we all had a clearer understanding of the coding standards we were going to be working with. Many of us felt the complexity of the project and this sometimes made it difficult when the work became more siloed. In previous classes, regarding web development basics, we were in classes where the projects might be different, but the tech stack we used was usually going to be the same.

Next, we would have liked to use Notion earlier in the project. Notion had so many features that would have completely changed the way we organized ourselves during the earlier milestones. Even for Milestone 2 where we have to implement a simple about us page, it would have been nice to see the progress of everything as we were working on it. We feel like it also kept the team more engaged, making a central place where all of our individual notes and comments could live in one place. Although it obviously has more use later on in the project when we have more complicated tasks and features to implement. However, it would have been nice to get familiar with the new organizational tool earlier. With how basic Milestone 1 can be, it's a nice way to ease into using Notion, since you only have sections of the document to deliver on.

Finally, we would have focused more on coding on certain milestones, specifically Milestones 1 and 2. Despite our clear improvements on the project between, and after these milestones, we didn't really perform to our potential at these milestones because we focused more on the wireframes and the visual components of the documentation over our coding, and these focus changes, in combination with our strict time tables, meant that we didn't really start delivering the function components well until later Milestones.

4)Team member contributions

- **Max Shigeyoshi (7/10)**
  - (M1)
    - Created Doc
    - Created 5 personas
    - 2 use cases
    - 33 functional requirements
    - 23 non functional requirements
    - 1 entity
    - Reviewed executive summary
    - Reviewed Tech Stack
    - Created AWS instance
    - Set up all dependencies
  - (M2)
    - Created M2 Document
    - Project Management Write-up
    - Network and Distributions Diagram
    - Identifying Key Risks
    - Created Notion Board
    - Reviewed Data Definitions
    - Search Functionality
  - (M3)
    - Corrected search functionality
    - Backend-frontend integration
    - Milestone 3 document creation
    - Merging branches to integration then to master.
  - (M4)
    - Created the inbox and messaging feature
    - Created the leaderboard feature
    - Assisted with git merge and resolving conflicts
    - Updated the Milestone 3 document
    - Assisted with integration frontend-backend
    - Created demo for the frontend on how to use JSON objects
    - Created Notion board for M4
    - Updated the instance to run the master branch
  - (M5)
    - Milestone 5 Document formatting
    - Reviewed Milestone 5 document

- **Aaron Rayray (8/10)**
  - (M1)
    - formatted document
    - 2 use case

        - 11 personas
        - 9 functional requirements
        - 2 non functional requirements
        - Competitive Analysis
        - Reviewed executive summary
        - Reviewed Tech Stack
    - (M2)
        - High level mock ups
        - Splash,Sign In,Sign-Up,Home
        - Mentor,Profile,Workspace,Forums
        - Body portion of the webpages
        - Sign In Page(frontend)
        - Sign Up Page(frontend)
        - Forgot Password Page(frontend)
        - Identified key risks when going through frontend
    - (M3)
        - Updated css for all pages
        - HTML Files : Splash, Workspace, Java best practices, Java, Javascript, Store, Room1-15,
        - React getting started
        - Header configurations for all pages
        - Footer configuration for all pages
        - Creating scripts for pages that needed redirecting
    - (M4)
        - Handled the Git merges and deletions to keep the git organized
        - Handled the Code Review with the other teams
        - Created Mentor frontend,i.e mentorvalidation, mentorrequest
        - Handled organization within files on it's respected branch
        - Relocated each file to have better organization within the repo
        - Redefined mentor page to have a search and filter
        - Redefined find a mentor to lead back to forums(private messages)
        - Assisted with integration frontend-backend
    - (M5)
        - Milestone 5 Product Summary
        - Reviewed Milestone 5 document
        - Troubleshot integration issues

- **Noah Hai (8/10)**
    - (M1)
        - 15 functional requirements
        - 3 personas
        - 5 entities
        - 7 functional requirements
        - Competitive Analysis

        - Reviewed executive summary
        - Reviewed Tech Stack
    - (M2)
        - System Design
        - Created ERD
        - Created UML
    - (M3)
        - Created mentor pages,
        - Created premium page
        - Created book a mentor page
        - Created find a mentor page
    - (M4)
        - Created Leaderboard frontend
        - Wrote Product summary
        - Created Milestone 4 document
        - Assisted with integration frontend-backend
        - Edited premium page as per professors feedback
        - Added in TOS and Privacy condition to sign up as per professors feedback
        - Resolved css render/linking issues when launching app
        - Updated search function in explore page
    - (M5)
        - Milestone 5 Document formatting
        - Reviewed Milestone 5 document
        - Troubleshot integration issues

- **Shez Rahman (10/10)**
    - (M1)
        - 23 functional requirements
        - 3 use case
        - Executive Summary
        - Reviewed Tech Stack
        - Assisted in setting up AWS instance
        - Assisted in troubleshooting tech stack
    - (M2)
        - Data Definitions
        - Prioritized functional reqs.
        - Reviewed Data Definitions
        - Sign in function of the backend
        - Sign up function of the backend
    - (M3)
        - Backend-frontend integration
        - Corrected sign in and sign up features
        - Corrected email verification

        - Corrected forgot password function
        - Troubleshoot all sections of code
    - (M4)
        - Delegated backend team tasks
        - Created the mentor feature
        - Assisted with integration frontend-backend
    - (M5)
        - Reviewed Milestone 5 document
        - Troubleshot integration issues

- **Ghadeer Al-Badani (9/10)**
    - (M1)
        - 6 personas
        - 1 use case
        - 33 non functional requirements
        - reformatted all diagrams
        - Reviewed executive summary
        - Created Tech Stack
        - Assisted in troubleshooting tech stack
        - Assisted in setting up dependencies
    - (M2)
        - High level api and algorithms
        - Created sign in functionality
        - Sign in function of the backend
        - Sign up function of the backend
        - Created README to instruct on how to
    - (M3)
        - Refactored coding style
        - Troubleshoot backend sections of code
        - Assisted in search functionality
        - Frontend backend integration
        - Created commenting standard
    - (M4)
        - S3 storage feature
        - Reviewed backend code
        - Updated the search and sorting feature
        - Assisted with integration frontend-backend
    - (M5)
        - Reviewed Milestone 5 document
        - Troubleshoot integration issues
        - Troubleshoot Instance issues

- **Majd Alnajjar (8/10)**
    - (M1)

        - 15 functional requirements
        - Reviewed executive summary
        - Reviewed Tech Stack
    - (M2)
        - Headers design
        - Footers design
        - CodeConnect icon design
        - Forum page Design
        - About me Design
        - About me page
        - Forum Page
    - (M3)
        - Created a leaderboard where users can view other's accomplishments with the ability to open profiles
        - Created a job posting page where companies post hiring opportunities
        - Created an application page where users can apply for said company
        - Created profile pages for 8 mock users on the leaderboards
        - Added a blank profile picture to profile.html
        - HTML Files: alice.html, bob.html, dave.html, carol,html, frank.html, eve.html, john.html, lenny.html, leaderboards.html, jobs.html, application.html
        - Wireframe for toby
    - (M4)
        - Created front end for Forum feature
        - Assisted in Leaderboard frontend
        - Assisted with integration frontend-backend
    - (M5)
        - Troubleshooting integration issues
        - Redesign splash
        - Fix up forum
        - show/hide replies, leave replies, create groups/posts

- **William Pan (9/10)**
    - (M1)
        - 2 use case
        - 1 persona
        - Reviewed executive summary
        - Reviewed Tech Stack
    - (M2)
        - High Level database design
        - Reviewed Data Definitions
        - Created ERD
        - Created EER
        - Forward Engineering

        - High level mockups
        - Search Functionality
        - Created Test DB
    - (M3)
        - Managed Mysql database
        - Updated models to create database schemas
    - (M4)
        - Updated the DB
        - Created the challenges feature
        - Usability test plan
        - Adherance to non functional requirments
        - Assisted in mentor feature
        - Usability test case
        - Assisted with integration frontend-backend
    - (M5)
        - Reviewed Milestone 5 document
        - Troubleshot integration issues
        - Troubleshoot DB

- **Phillip Ma (9/10)**
    - (M1)
        - 18 functional requirements
        - Competitive Analysis
        - Reviewed executive summary
        - Reviewed Tech Stack
    - (M2)
        - Created all use case diagrams
        - Created body sections for webpages
        - High level mockups
        - Identified key risks
        - Distributed front end tasks
    - (M3)
        - Css style changes for all pages
        - Search function integration
        - Profile, forgot password, forgot password verification, homepage, inbox, explore, challenges
        - Wireframes
        - Frontend backend integration
    - (M4)
        - Delegated frontend team tasks
        - Updated the Milestone 3 document
        - Updated the Milestone 3 document
        - Assisted with integration frontend-backend
        - Assisted with Inbox frontend

    - Assisted with Explore frontend
- (M5)
    - Reviewed Milestone 5 document
    - Troubleshot integration issues