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

1) Product summary

CodeConnect serves as a superior platform for coders to enhance their technical skills and to build strong connections within the tech community. 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)

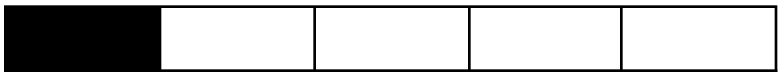
Efficiency: compute approximate average time it took the user to completed task

# . 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 page	It take user 15 minutes to competed the form and he do not know that there is a email sending
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

	Strongly disagree				Strongly agree
1. I think I would practice more coding challenges on this application .			■		
	1	**2**	**3**	**4**	**5**
2. I found the challenge too difficult .				■	
	1	**2**	**3**	**4**	**5**
3. The challenge is well designed and easy to interact with.					■
	1	**2**	**3**	**4**	**5**
4. The user interface is clear and easy to understand how to completed the become mentor form					■
	1	**2**	**3**	**4**	**5**
5. I found the become mentor form filed unnecessarily complex				■	
	1	**2**	**3**	**4**	**5**
6. I think I would need contact technical support for submit the become mentor form					■
	1	**2**	**3**	**4**	**5**
7. I was able to quickly find and view the topic I was searching for			■		

	1	2	3	4	5
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					■
12.The uploaded video appeared correctly and was accessible immediately	■				
13.Creating and submitting a post was simple and intuitive.				■	
14.Viewing and interacting with posts was easy and seamless.		■			

15. Navigating the forum and finding specific content was efficient.	 **1 2 3 4 5**

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.

   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.

   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 succes**	**Pass**
3	**User submits existing email and invalid reset password**	**shezpc@gmail.com, invalid**	**Passwor**	

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 and its clicked before expiring**	**johndoe123@gmail.com**	**Email verification sent, link brings you to home page**	**Pass**
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 sent, link brings you to "reset password" page**	**Pass**

4. Test sequences - XXXX

Test objectives:

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:

QA Test Plan:

Number	Description	Test Input	Expected Output	Pass/Fail
1				
2				
3				

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

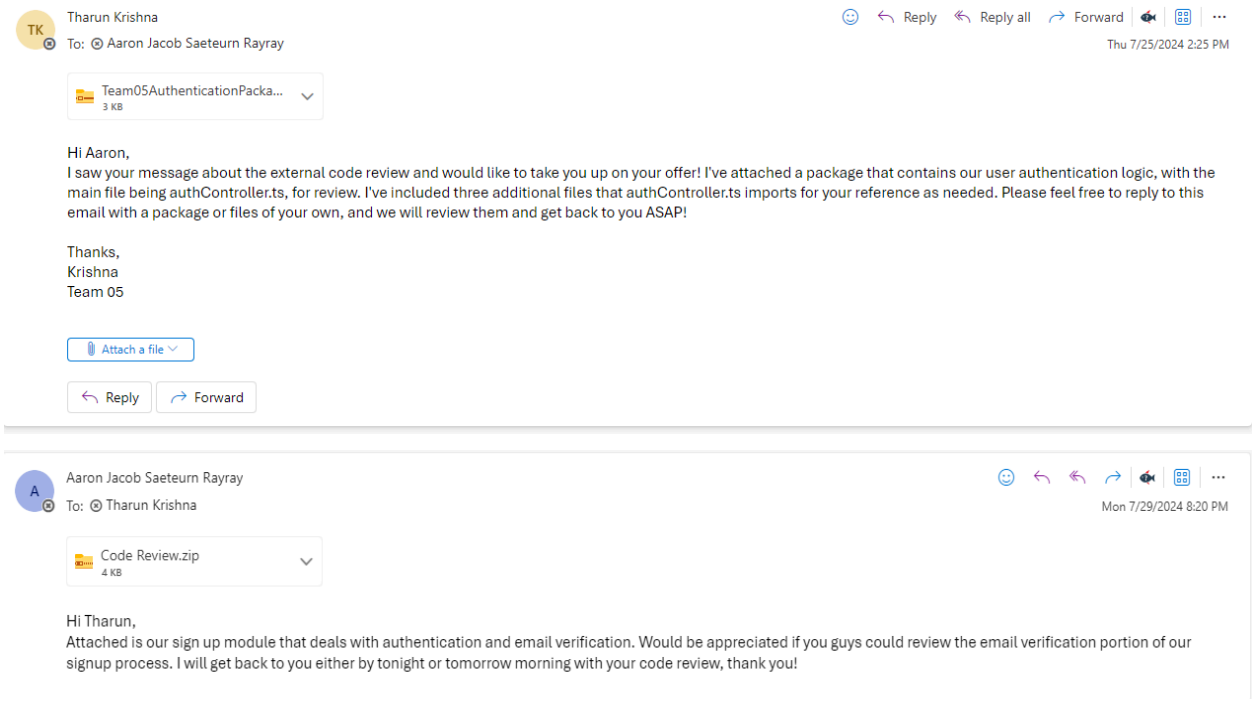
Feature to be tested:

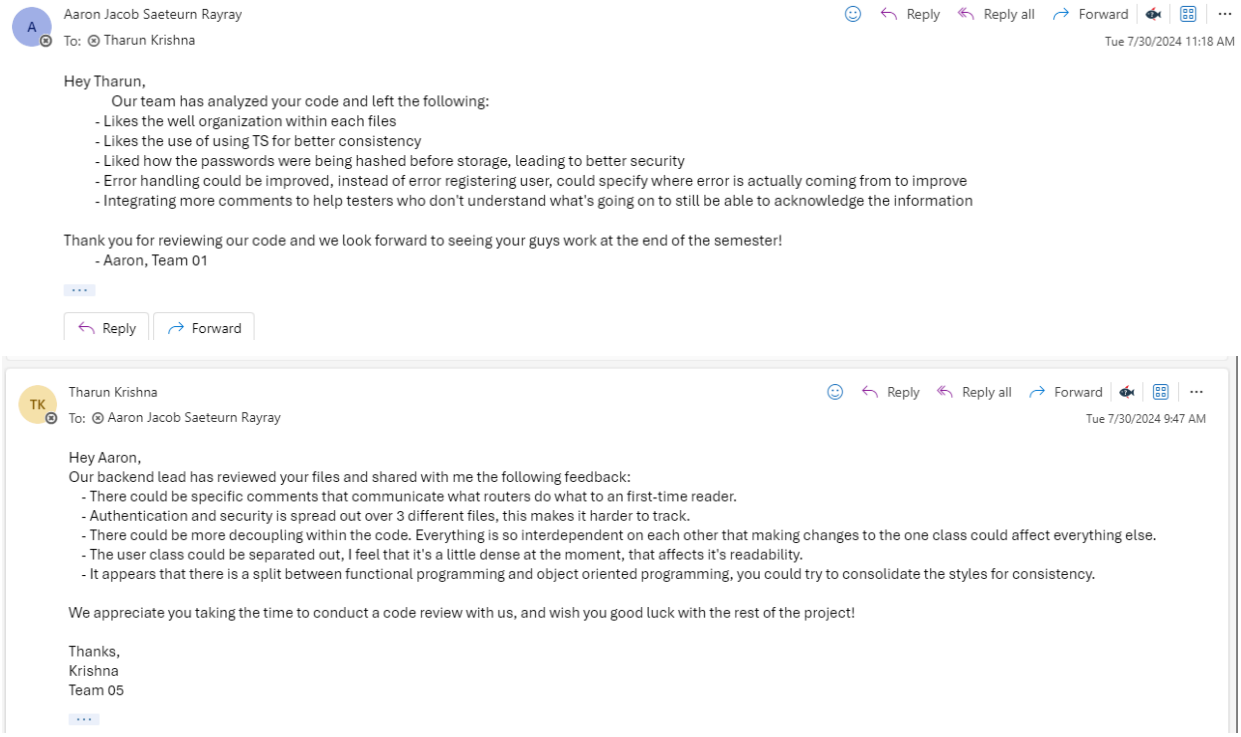
QA Test Plan:

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. Console.log the output**	**The job post should appear in the job posts database with the correct user_ID linkage.**	**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.**	
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.**	

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.

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.	
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.	
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**

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.	
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.	
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.	**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.	
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.	
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.	

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**

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.	
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.	
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.	**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.	
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**

Non-functional	**Status**
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.	**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.	
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.	
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.	

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