Worksy Release Summary

# Team members

|  |  |  |
| --- | --- | --- |
| Name and email | GitHub id | Role of each member and tasks done by each member (brief description) |
| Noel Omeiza  omeizan@myumanitoba.ca | omeizan | Full-Stack, Architecture Design |
| Evan Murray  murraye2@myumanitoba.ca | evanmurray99 and howlndog | Backend, Testing, Github organization, code reviewer |
| Casandra Hayward  haywardc@myumanitoba.ca | chayward3113 | Frontend, Testing,  UI - Design |
| Diljot Singh  singhd18@myumanitoba.ca | Dilpunjab | Frontend, Testing, documentation. |
| Tung Nguyen  nguye49@myumanitoba.ca | xuaantung | Developer – CI/CD |

# Project summary

Worksy is a web-based client-server application designed for students to advertise their skills to help them make money for university. It also gives student entrepreneurs a platform to launch their business ideas and in doing so it will help them to build and grow a customer base. Additionally, they can receive feedback on their products to help them further refine their business model. However, what good is advertising if it isn’t reaching your target audience. So, the other side of our web-based application is available to everyone, not just students. It provides users with the ability to search for a particular business or skillset that they desire. Then upon finding a service that they would like to use they can reach out to the services author through our chat feature and negotiate prices, discuss the overall project and place an order. Subsequently the user can post a review to provide the service creator with insights into what went well, what didn’t go so well and what they can do better. On the student service creator side, it provides them the opportunity to make money while refining their skills and getting real world experience that they can put on a resume. On the other hand, from the side of users who are seeking a service it provides them the opportunity to support local students instead of big corporations.

Differences between final version and proposal:

**Account Management:**

* Accounts do not have usernames instead we use their email.
* User accounts do not have privacy settings as the only information publicly available are the posts, chats and reviews that they create along with their first and last name which is used as a username in certain features such as review.
* We restricted the create post feature to student accounts only because this application is designed to help students advertise their skills.

**In-App Messaging:**

* The chat feature was initially planned to be integrated into the post feature however instead we made the chat feature distinct. Therefore, to chat with another user about a post you have to navigate to the post in the chat feature instead of being able to chat from within the post itself.
* Users cannot delete their chat rooms.

**Post:**

* We did not implement a bookmark feature.
* Posts are not set up to contain images as was initially planned.

**Reviews:**

* There is no order feature as it is expected that orders will be formed in the chat due to the likeliness of negotiations in price, timeline and project details. Therefore, anyone can rate any service regardless of whether they have "ordered it or not".
* For review we changed the lowest possible rating to 0 stars instead of 1 star as was initially planned.
* We removed the review breakdown of all 1, 2, 3, 4 and 5 star reviews.
* We also moved the individual reviews into a popup which contains all reviews for that post.

**Search:**

* Filters differ slightly from the description. There is a price filter and there is a set of predefined category filters which don’t include conditions and locations.

## GitHub repository Link

<https://github.com/evanmurray99/Worksy>

**DockerHub repository link**

DockerHub link: <https://hub.docker.com/u/worksy4350>

**How to run docker images:**

- Clone the GitHub Repository

- Change the directory to the project directory

- Run “docker-compose pull”

- Run “docker-compose up --build –d"

- Go to the following URL: <http://localhost:3030>

## List of user stories for each sprint

[Sprint1](https://github.com/evanmurray99/Worksy/milestone/4)

None

[Sprint 2](https://github.com/evanmurray99/Worksy/milestone/1?closed=1)

US #1: [As an existing user, I want to be able to view and update my account information](https://github.com/evanmurray99/Worksy/issues/21) [Status: Done]

US #2: [As a logged-in user, I want to be able to log out when I am finished on the application to ensure my account is secure](https://github.com/evanmurray99/Worksy/issues/23) [Status: Done]

US #3: [As an existing user, I want to be able to easily find the services I have created](https://github.com/evanmurray99/Worksy/issues/22) [Status: Done]

US #4: [As a logged-in user, I want to be able to create my own service offering to attract potential customers](https://github.com/evanmurray99/Worksy/issues/4) [Status: Done]

US #5: [As a logged-in user, I want to be able to delete my listing so that if I decide to stop offering services I don't have to worry about people trying to contact me](https://github.com/evanmurray99/Worksy/issues/6) [Status: Done]

US #6: [As an existing user I want to be able to login to my account so that I only see the content relevant to me](https://github.com/evanmurray99/Worksy/issues/8) [Status: Done]

US #7: [As a new user I want to be able to create an account to ensure that my content and interactions are kept private](https://github.com/evanmurray99/Worksy/issues/7) [Status: Done]

[Sprint 3](https://github.com/evanmurray99/Worksy/milestone/2?closed=1)

US #8: [As a potential customer, I want to be able to search for a service by preset categories so I can find relevant freelancers](https://github.com/evanmurray99/Worksy/issues/3) [Status: Done]

US #9: [As a potential customer, I want to be able to search for a service by free-form keywords so I can find relevant freelancers](https://github.com/evanmurray99/Worksy/issues/2) [Status: Done]

US #10 [As a logged-in user, I want to be able to message potential buyers or potential sellers to inquire or answer questions about a service](https://github.com/evanmurray99/Worksy/issues/9) [Status: Done]

US #11: [As a logged-in user, I want to be able to view my chat history so I have a record of my conversations](https://github.com/evanmurray99/Worksy/issues/10) [Status: Done]

US #12: [As a logged-in user, I want to be able to delete conversations I no longer need so that I only see the relevant conversations](https://github.com/evanmurray99/Worksy/issues/12) [Status: Done]

US #13: [As a logged-in user, I want to be able to make changes to my service offering so that my services offered are up to date](https://github.com/evanmurray99/Worksy/issues/5) [Status: Done]

[Sprint4](https://github.com/evanmurray99/Worksy/milestone/3?closed=1)

US #14: [As a logged-in user, I want to be able to receive notifications when someone sends me a message so I can be up to date with my communication](https://github.com/evanmurray99/Worksy/issues/11) [Status: Done]

US #15: [As a potential buyer, I want to be able to see how others felt about a service before I decide to buy that service](https://github.com/evanmurray99/Worksy/issues/24) [Status: Done]

US #16: [As a buyer, I want to be able to post reviews of the services I have used so that I can provide feedback to the seller](https://github.com/evanmurray99/Worksy/issues/13) [Status: Done]

US #17: [As a seller, I want to be able to see how others responded to my service](https://github.com/evanmurray99/Worksy/issues/14) [Status: Done]

# User manual

To run the application effectively and make the most of its core features, here are step-by-step instructions for each feature:

**1. Account**

There are two types of accounts Students and Clients where the primary difference between account types is that only students can create posts.

1. **For Students:**

* **Registration:** Sign up using your student email to verify your student status.
* **Profile Setup:** Complete your profile by providing the necessary details like First name, last name and password.

1. **For Clients:**

* **Registration:** Sign up with your email and necessary details.
* **Profile Setup:** Complete your profile by providing the necessary details like First name, last name and password.

After setting up the Account you will be directed to the login page because you still need to login. To login provide your email and password and click login then you will be directed to the Homepage.

**Manage Accounts**

* On the dashboard click on ‘My content’
* Click on ‘Account Information’
* In that form, the User can change Email, First Name, Last Name Biography and password.
* After changing any or all of the fields click on the ‘Update’ button.

Users can log out simply by clicking on the ‘Logout’ button.

**2. Post**

* On the dashboard click on ‘My content’
* Click on ‘Create Post’ **- This option is only available for students**
* Enter the Title, Cost, Categories and Description of the service.
* Click on ‘Create New’

**Managing Posts:**

* On the dashboard click on ‘My Content’
* Click on ‘Your Posts’
* The user will see all the posts that they have created.
* To edit a post the user can click on the edit button which is a blue-green button that contains a pencil icon.
* This will produce a popup where the user can change the post's title, cost, description, and categories.
* Then once the user is satisfied with the changes they can click on ‘Update’ to save the changes.
* Users can also delete their posts by clicking on the Delete button which is a red button that contains a trashcan icon.
* When the delete button is clicked a popup will appear warning the user that the delete action cannot be undone.
* From here the user can either proceed by clicking “Delete” or they can cancel the delete by clicking cancel, the X button in the top left corner, or by clicking outside of the popup.

**3. Search**

* The search bar is present on the ‘Dashboard’ and ‘Services’ pages.
* You can search by entering keywords in the search bar and clicking on the search button.
* Then all the posts related to the keywords that you entered into the search bar will be displayed. There will also be a set of filters on the left side of the screen.
* Users can change the category filters by simply clicking on the required filters. If you navigate to the search page manually or by using the search bar, then all categories will default to selected.
* Users can also filter by prices by moving the price slider. By doing so it will remove all prices that are greater than the selected value.
* Users can change the order of the search results by relevance, date, and price.
* Search results will be displayed according to keywords and filters.
* From the dashboard page you can click on one of the tile buttons and it will direct you to the services page with the category represented by the tile selected and all others unselected.
* Finally, as there may be hundreds of posts available, we used pagination to only display 10 posts at a time.
* To view the entire contents of a page you need to scroll.
* Then to view the contents of another page you need to click on either the > button to go up one page or the < button to go down one page.

**4. Review**

Assuming the User has already found the desired post. To write the Review

* Navigate to the post you want to review.
* Click on the ‘Review’ button.
* Users will see all the reviews and ratings of the post.
* If the User just wants to check the reviews, then the user can scroll through the reviews and then click the ‘Close’ button to exit
* Otherwise, the user can rate the product and write a comment in the supplied form.
* Then to submit the review they click on the ‘Submit’ button.

**Managing Reviews**

* On the dashboard click on ‘My Content’
* Click on ‘Your Reviews’
* The user will see all the Reviews that they have created.
* The user can change the comment and rating.
* The user can also delete the review by clicking on the Delete button which is the red button containing the trashcan icon.

**5. Chat**

Assuming the User has already found the desired post. If the user wants to chat with the service provider. Then to start a chat

* Navigate to the post.
* Click on the ‘Chat button, the button is next to the Service provider’s name on the post.
* There will be a pop-up window with a default message, the user can send the default message or replace it with one of their own then they will send the message by clicking the ‘SEND’ button.
* Users can access all of their ongoing chats by clicking on the chat icon on the bottom right of the display.
* After clicking on the Chat icon, there will be pop-up windows with all of their conversations.
* Users can read new messages and reply to the messages.
* To close the window simply click on the chat icon again.

# Overall Arch and Design

Provide the link to the overall arch, class diagram on GitHub.

[Architecture Diagram](https://github.com/evanmurray99/Worksy/blob/main/docs/Architecture-Diagram.png)

[Class Diagram](https://github.com/evanmurray99/Worksy/blob/main/docs/Class-Architecture.jpeg)

## Infrastructure

**MongoDB:** Enabled us to store data in a document model through Mongoose. MongoDB uniquely generates object IDs and manages deletions by invalidating object IDs used as references for deleted items, this means we did not have to manually handle updating records with references to deleted records. It also automatically indexes IDs by default for all tables. This indexing enhances search speed, especially for large tables such as the “messages” table. Finally, MongoDB stores data in object form, eliminating the need for additional result parsing, so we only needed to handle rendering. These functionalities would have required manual implementation with traditional relational databases. Overall MongoDB was a good fit for our project since we did not require complex queries but needed speed and flexibility.

**React:** Our project requires a lot of component reuse as well as state maintaining, using react hooks and state management, we were able to effectively handle all the API calls and update pages and components neatly. This would have been much more complex to implement with traditional HTML or AngularJS event broadcasting.

**Chai/Istanbul:** Since our unit tests interact directly with an instance of our MongoDB database, chai was good for asserting the type and contents of the results that were returned from the database, the test logs also included durations for each test which helped give a sense of the speed of our APIs. It also allowed us to define test dependencies which we used to maintain the order of the tests we ran. Chai worked well with an additional library (‘Istanbul’) which we used to report code coverage on our tests  
  
**Mongoose:** Mongoose stands out as the preferred ORM (ODM in our context) for MongoDB, facilitating the creation of objects from the results of API calls. It also enforced type checks and ensured input validity when transmitting data to the database.

**JsonWebToken:** URL-safe, helped in encrypting payload that holds JSON – also allowed us to build expiration flags into our token which we used in session management.

**Node + Express**: Node.js, paired with Express, proved to be the ideal combination for our backend, especially considering that our frontend is written in JavaScript and heavily involves JSON handling. Node.js was well-suited for our backend as it seamlessly aligns with JavaScript, while Express, being the de facto standard server framework for Node.js, provided us with a straightforward platform to build our APIs.

**Socket.io**: Socket.io was easy to understand and instrumental in implementing real-time chat functionality for our project. Its seamless integration with Node.js, along with its support for the same Express app used for database connections, made the integration process straightforward and efficient.

## Name Conventions

Convention used: [Google JavaScript Style Guide](https://google.github.io/styleguide/jsguide.html)

Exception: Files named (and hence Component function names) using UpperCamelCase

## Code

|  |  |
| --- | --- |
| File path with a clickable GitHub link | Purpose (1 line description) |
| [Server.js](https://github.com/evanmurray99/Worksy/blob/main/Server/Server.js) | Initializes server connection to database and handles socket events for chat |
| [UserController.js](https://github.com/evanmurray99/Worksy/blob/main/Server/Controllers/UserController.js) | Handles all backend user/authentication processes |
| [ServiceController.js](https://github.com/evanmurray99/Worksy/blob/main/Server/Controllers/ServiceController.js) | Handles all backend functionality related to services/posts |
| [Services.jsx](https://github.com/evanmurray99/Worksy/blob/main/Client/src/Pages/Services.jsx) | Frontend component for searching, filtering and displaying services |
| [ChatComponent.jsx](https://github.com/evanmurray99/Worksy/blob/main/Client/src/Components/ChatComponent.jsx) | Frontend component for real-time chat functionality |

# Continuous Integration and deployment (CI/CD)

1) Describe your CI/CD environment and the clickable link to your CI/CD pipeline. For instance, if you use GitHub Action, provide the link to the workflow

<https://github.com/evanmurray99/Worksy/tree/develop/.github/workflows>

2) Snapshots of the CI/CD execution. Provide one for CI and one for CD to demo you have successfully set up the environment.

- CI Client

A screenshot of a computer

Description automatically generated

- CI Server

A screenshot of a computer

Description automatically generated

- CD pipeline

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Testing**

### Link to testing plan

<https://github.com/evanmurray99/Worksy/blob/main/docs/testing-plan.md>

### Unit/integration/acceptance test

Each story needs a test before it is complete. In other words, the code coverage (in terms of statements) should be 100%. If some classes/methods are missing unit tests, please describe why and how you are checking their quality. Please describe any unusual/unique aspects of your testing approach.

* We have 100% coverage of every file except the app.js and db.js config file. There are only 4 lines of code not being tested between these files. They are setting up the connection to our database. If there was a bug in any of these lines none of the tests would pass, making it easy to check its quality.

List the **10** most important unit tests with links below.

|  |  |
| --- | --- |
| Test File path with clickable GitHub link | What is it testing (1 line description) |
| [Category.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/Category.test.js#L201) | Getting all categories and services |
| [Category.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/Category.test.js#L147) | Add a service to a category |
| [Chat.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/Chat.test.js#L116) | Create a chat |
| [Chat.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/Chat.test.js#L195) | Add message to chat |
| [Review.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/Review.test.js#L131) | Getting reviews by service ID |
| [Review.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/Review.test.js#L81) | Create a review |
| [Service.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/Service.test.js#L97) | Get a service |
| [Service.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/Service.test.js#L144) | Get all services |
| [User.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/User.test.js#L23) | Create a user |
| [User.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/User.test.js#L148) | Login |

List the **5** most important integration tests with links below.

|  |  |
| --- | --- |
| Test File path with clickable GitHub link | What is it testing (1 line description) |
| [ChatsIntegration.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/ChatsIntegration.test.js#L69) | User should not be able to join multiple chats |
| [ReviewsIntegration.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/ReviewsIntegration.test.js#L64) | User can create multiple reviews on of the same service |
| [SearchIntegration.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/SearchIntegration.test.js#L121) | Search by categories and keyword |
| [ServiceIntegration.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/ServiceIntegration.test.js#L80) | Finding all services for a category |
| [UserIntegration.test.js](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Server/test/UserIntegration.test.js#L39) | Getting all services a user has created |

List the **5** most important acceptance tests with links below.

|  |  |
| --- | --- |
| Test File path with clickable GitHub link | Which user story is it testing |
| [Login.test.jsx](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Client/src/__test__/Login.test.jsx#L38) | A user wishes to login to their account with their correct email and password (user is returned a token) |
| [Review.test.jsx](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Client/src/__test__/Review.test.jsx#L53) | User wishes to see the reviews from a service to see if it is a quality product |
| [Review.test.jsx](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Client/src/__test__/Review.test.jsx#L77) | User wishes to create their own review after they have purchased a service |
| [Signup.test.jsx](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Client/src/__test__/Signup.test.jsx#L10) | User wishes to view what information is needed to create an account |
| [Signup.test.jsx](https://github.com/evanmurray99/Worksy/blob/1f14cc0102327285a53dbc09d02d18b752d8e930/Client/src/__test__/Signup.test.jsx#L31) | User wishes to create a new account to use Worksy |

Tests are being run with GitHub actions within the CI pipeline. As it runs all front-end tests.

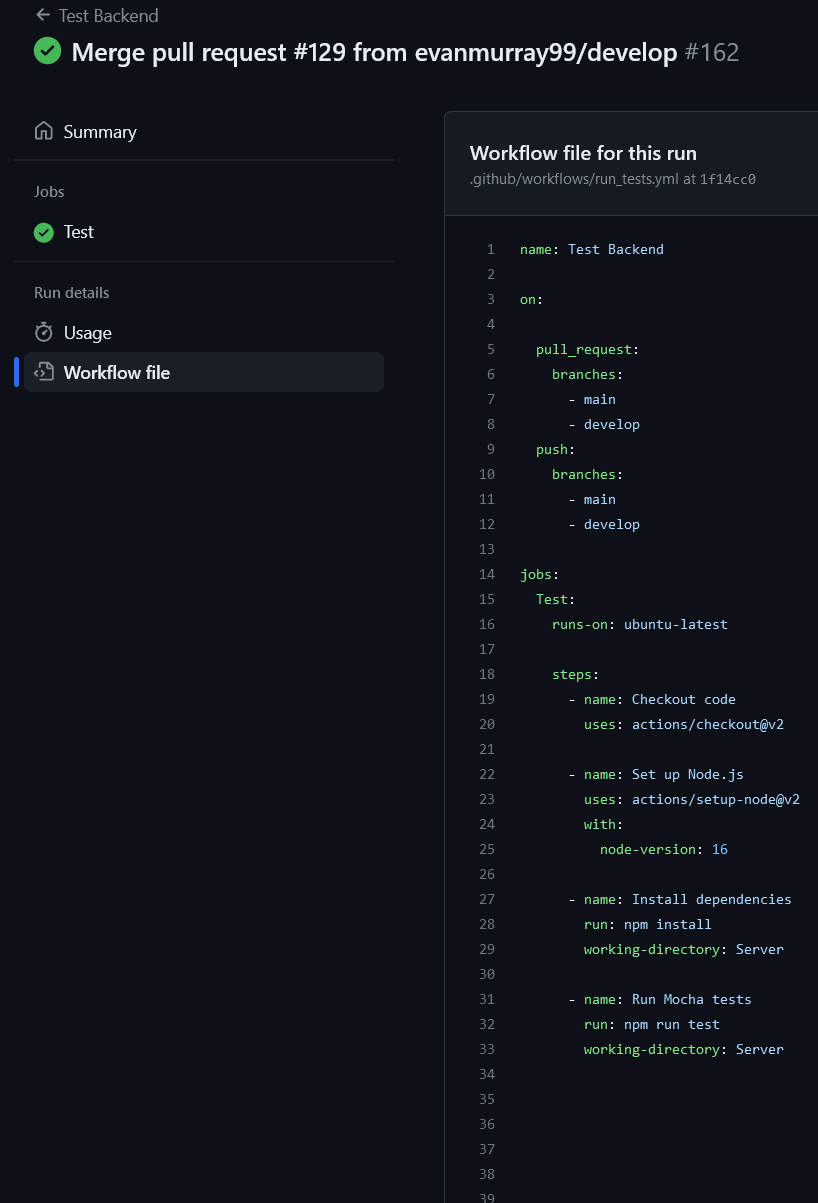
### Regression testing

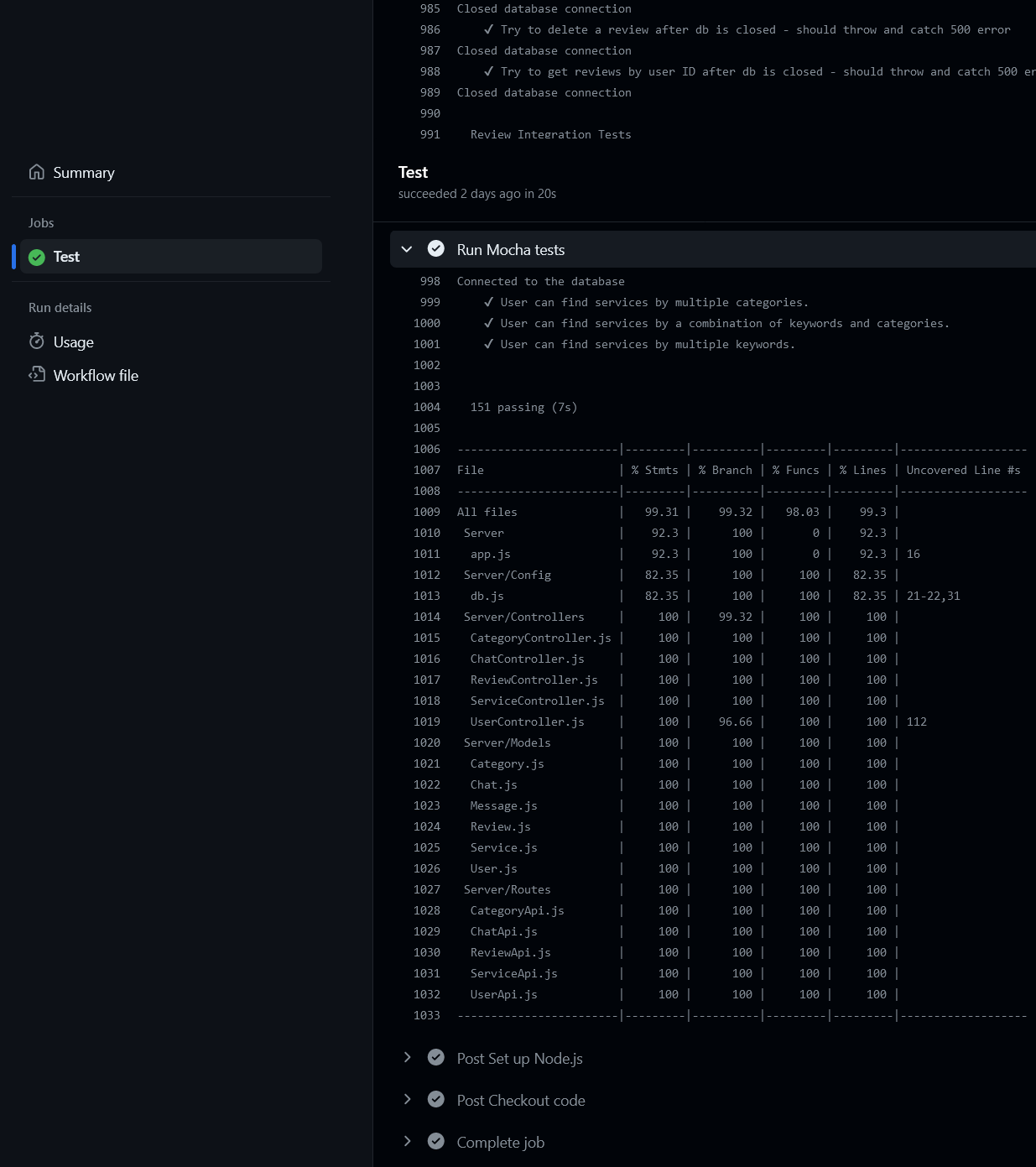
1) Describe how you run the regression testing (e.g., which tests are executed for regression testing and which tool is used?).

2) Provide the link to regression testing script and provide last snapshot of the execution and results of regression testing.

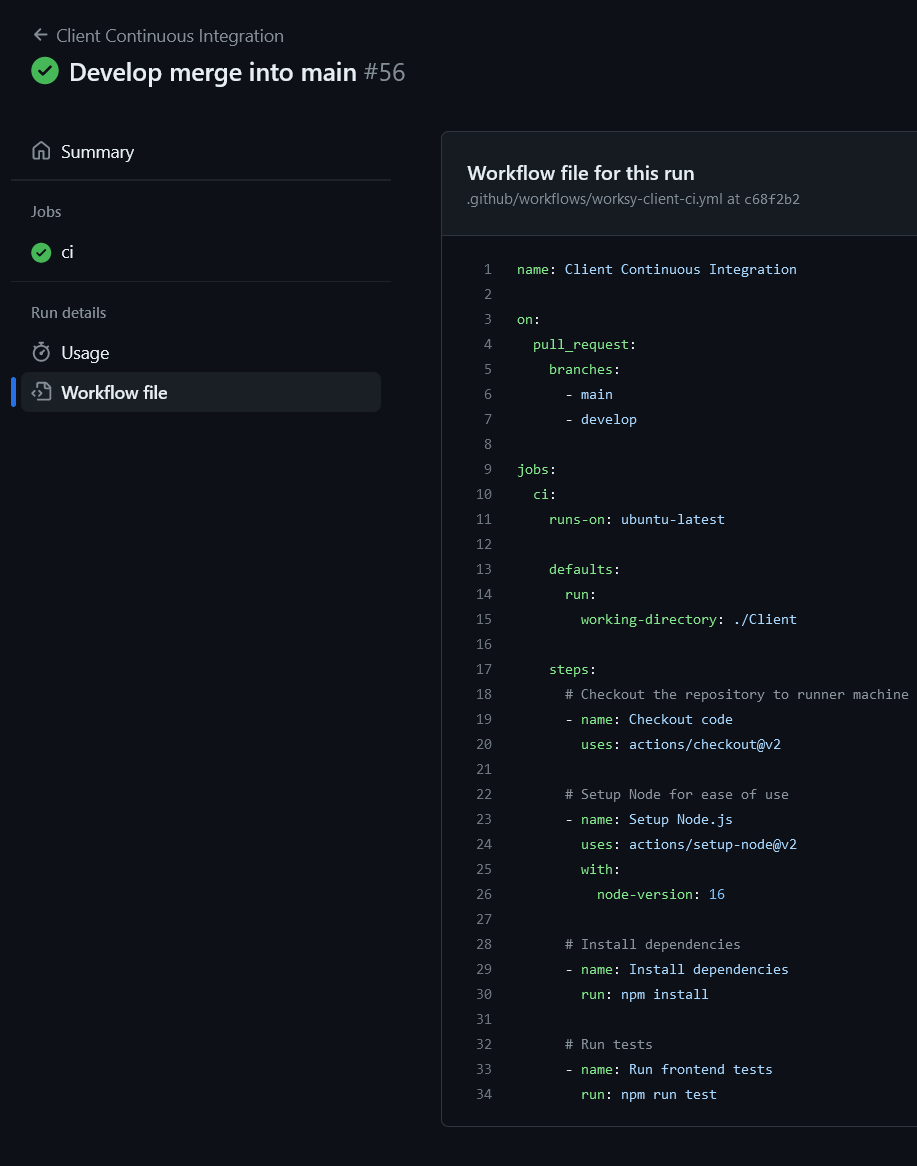
We are using mocha to run the tests and nyc to display the coverage at the of the test run. All unit and integration and acceptance tests are being executed for regression tests. We are using github actions to run the tests.

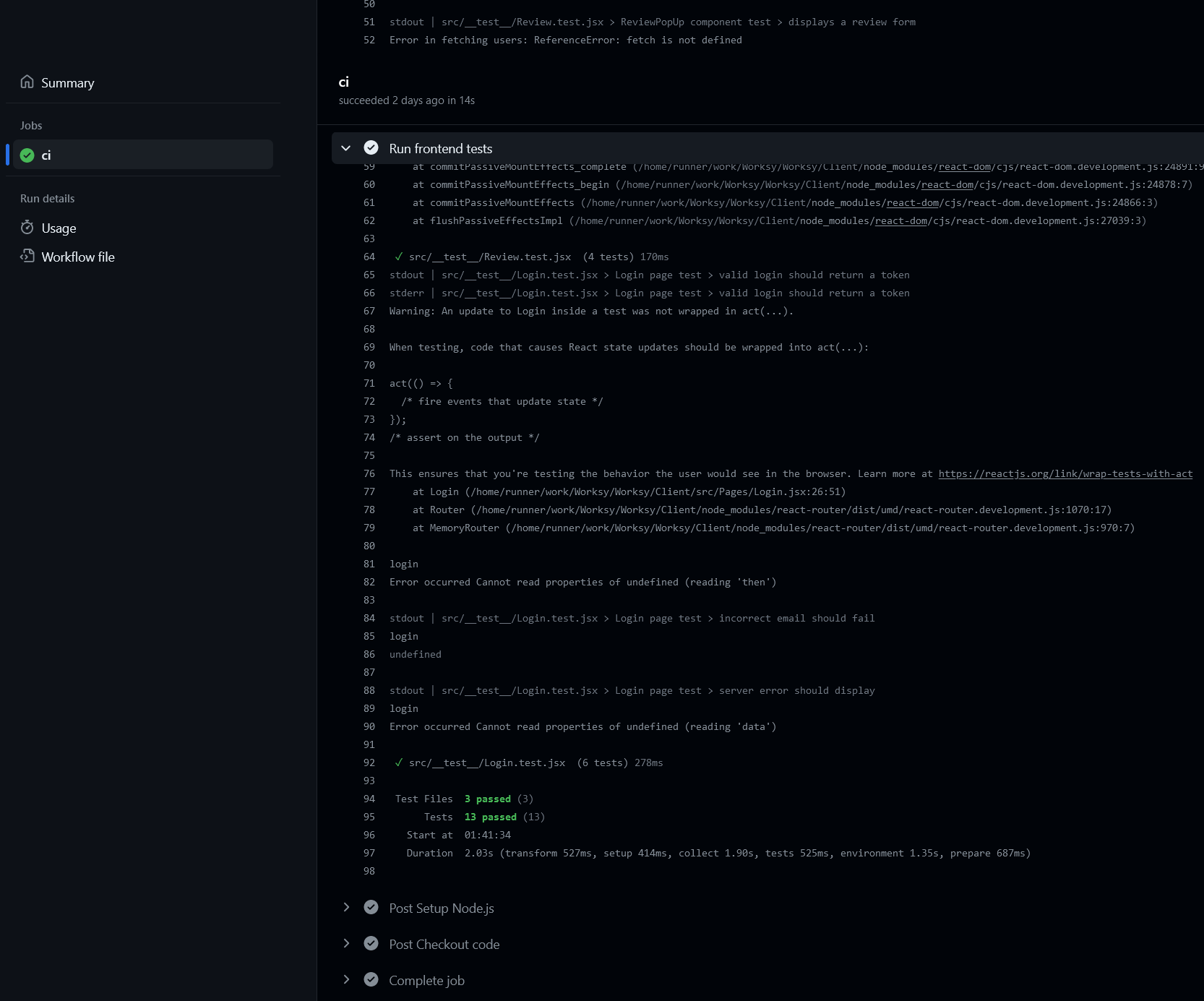
Backend (integration and unit)





Frontend acceptance tests





### Load testing

1) Describe the environment for load testing, such as tools and load test cases. 2) provide the test report for load testing.

3) discuss one bottleneck found in the load testing.

[LINK TO jmx file](https://github.com/evanmurray99/Worksy/blob/main/Server/stress-test/stressTest.jmx)

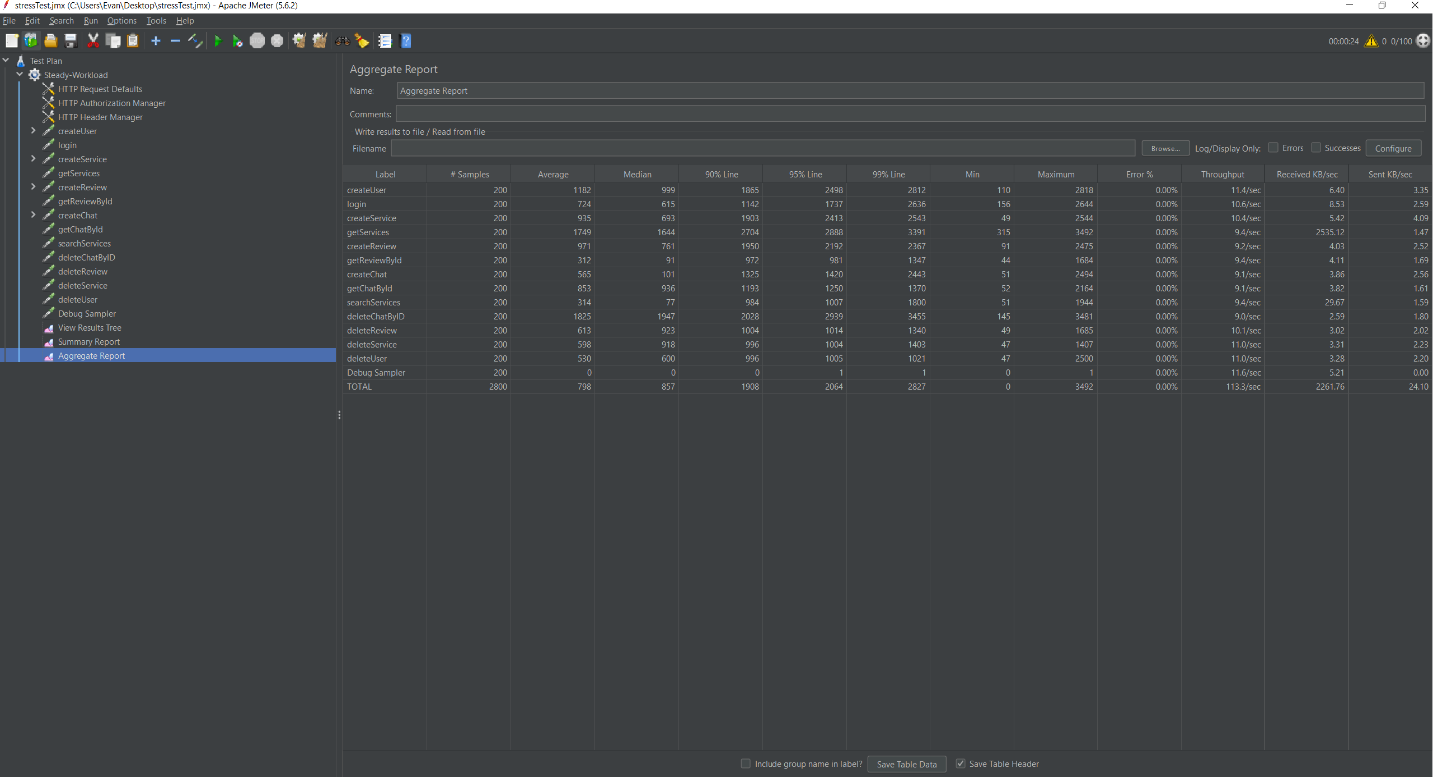
We used JMeter for load testing, API calls that were tested are

* createUser
* Login
* createService
* getServices
* createReview
* getReviewById
* createChat
* getChatById
* searchServices
* deleteChatById
* deleteReview
* deleteService
* deleteUser

With the JMeter test 200 requests were sent per method, with a total of 2600 calls occurring within a couple of seconds. We achieved no errors and every one of the 2600 calls responded correctly and passed the tests.

Since we used a cloud database (MongoDB Atlas) it was very good at responding to every call, as we found no bottlenecks. The throughput for each method was very similar at around 10 per second at once, with the total being 113 per second.

[Report Link](https://github.com/evanmurray99/Worksy/blob/main/Server/stress-test/testReport.png)



**Security analysis**

1) Describe the choice of the security analysis tool and how do you run it. The security analysis tool should analyze the language that is used in the majority of your source code.

- SonarCloud is easily integrated with GitHub Actions.

- It offers various ways to integrate, the two most popular ones being CI-based and automatic. With automatic integration, SonarCloud provides a bot message whenever there is a new code change and details whether the new code passes the Quality Gate. On the other hand, the CI-based integration offers a more controlled way of how and when SonarCloud will be run

2) Attach a report as an appendix below from static analysis tools by running the security analysis tool on your source code. Randomly select 5 detected problems and discuss what you see. Note that you are not required to fix the alarms (bugs and vulnerabilities) in the course.

A screenshot of a computer

Description automatically generated

- Problem 1:

A screenshot of a computer program

Description automatically generated

- The use case for var is very limited. It's often recommended to use let or const instead.

- According to this [article](https://evertpot.com/javascript-let-const/) var can lead to unexpected results due to 'hoisting'

- Problem 2:

A screenshot of a computer

Description automatically generated

- It suggests not to implement methods that have more than 7 parameters

- Problem 3:

A screenshot of a computer program

Description automatically generated

- Props is often passed as an object and without type safety, it can cause errors and confusion when other read the code

- Problem 4:

A screenshot of a computer program

Description automatically generated

- It's conventional to name the variable and its update function to have similar names to follow best practices. In this case, it should be 'setIsOpen'

- Problem 5:

A screenshot of a computer program

Description automatically generated

- 'for-of' is more simplified than regular 'for' loop