Final Report

Project title: CSE PhD Qual Practice System (Video Link)

Team: WeCode

Scrum Master: Shruthi Sampathkumar
Product Owner: Abhishek Taur

Team Members:

Abhishek Taur
Pranav Kulkarni
Arif Arman
S M Farabi Mahmud
Raj Vardhan
Shruthi Sampathkumar
Shibin Tazhe Veettil

Overview:

The main customer for this Project is the CSE Department at Texas A&M University. Dr. Duncan Walker is the stakeholder. Our team WeCode was involved in the development of the project. The Project involved enhancing the legacy CSE Ph.D. Qualification Practice Application. The customer requirement included features to improve the application logging, timer for the quiz, Password recovery, bookmarking questions, email verification during registration, statistics for the quiz, ability to quit the quiz, and testing over different computing platforms like Mobile Phone, Desktop. These features were intended to improve the user authentications so that only people with a valid email can register, ability to recover password, ability to log in using Google, FB so that you don't need to register and start using the application as a user, get more insights into your quiz performance and ability to give a timed quiz, improve the UI for Mobile phones.

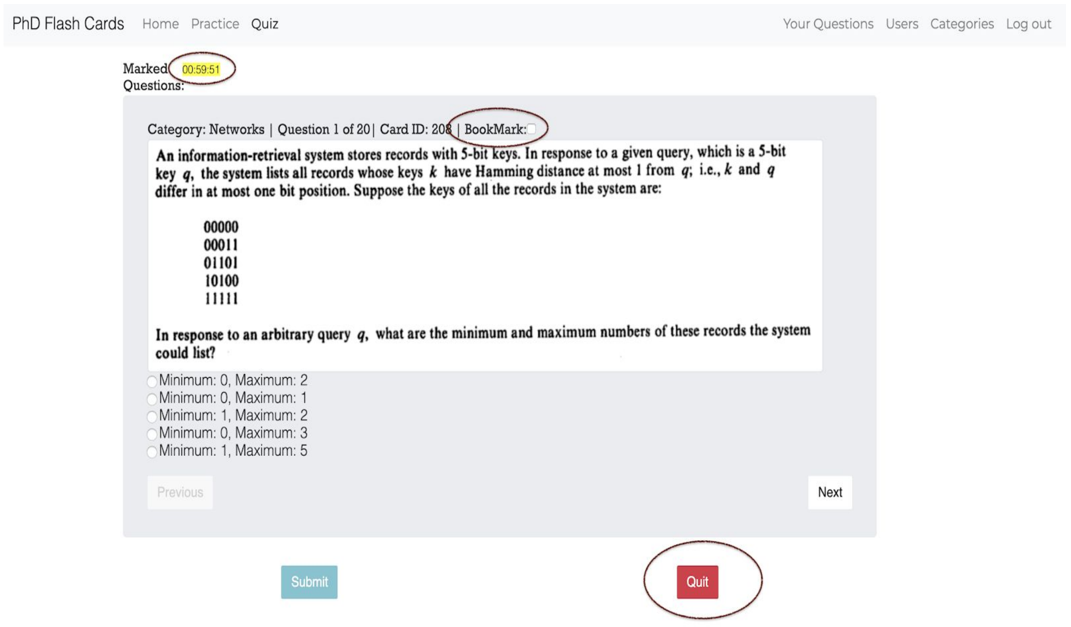
To allow login using Google and FB we introduced Oauth2 authentication. This involved using gems like devise which is a flexible authentication solution for Rails based on Warden, devise, omniauth-google and omniauth-facebook. Devise framework implements the Oauth2 authentication, allows validating the user token with the token received from Google and FB, and manages the user session. The Forgot Password feature allows the user to reset his password. This feature helps to recover an account for any valid registered email address. It sends a reset link via email to the registered email address and once you click the link within a given time frame it resets the user password by updating the user database. Email verification is a feature that allows only valid email addresses to register. To authenticate an email address a confirmation link is sent to the given email address which once clicked validates the email address and if it is not done in a given timeframe the user is prompted for the expired link and is allowed to resend a new link with which the email address can be confirmed. The quit quiz now allows the user to quit the quiz session anytime. It was implemented using simple javascript methods that used URL redirection. The user wasn't able to bookmark the questions in the Practice session before and this can be very troublesome when you want to jump back to a previous question. We added a bookmarking feature that allows us to jump to the bookmarked question. The user was not able to give a timed quiz before which allows the user to solve questions in a given time i.e. the way it would be in the actual Ph.D. Qual exam. We implemented the timer feature with which the user now can give a quiz in the given time mentioned in the timer field. A good insight for the quiz as suggested by our customer was to have quiz stats like the total time it took to do the quiz, the average time per question, and the time for each question. This feature was implemented using the timer values we had for the timer features by running stats and storing time per question. Improvement of the User interface for different platforms was one of the requirements of the user. Extensive testing was done by our team of all the features on Mobile Phone, Desktops using different browsers. We discovered a few bugs which we squashed and improved the overall user interface for the mobile and desktop. This way we were able to complete all the user requirements and were able to make the application more user friendly.

Scrum Master: Shruthi Sampathkumar
Product Owner: Abhishek Taur

User Stories:

- **Add Timer:**
    - Points: 2

- Implementation Status: We implemented the timer functionality such that when the timer completes, the quiz is auto-completed and the user is shown the results. We also did input validation to ensure that the timer is set to a valid value. The implementation involved reading the timer value entered by the user and then reducing the timer until it reaches 0 then triggering an event to show the answer and prompt the user. We used javascript methods to implement the above feature.



Category: Networks | Question 1 of 20| Card ID: 208 | BookMark:

An information-retrieval system stores records with 5-bit keys. In response to a given query, which is a 5-bit key q, the system lists all records whose keys k have Hamming distance at most 1 from q; i.e., k and q differ in at most one bit position. Suppose the keys of all the records in the system are:

```
00000
00011
01101
10100
11111
```

In response to an arbitrary query q, what are the minimum and maximum numbers of these records the system could list?

- Minimum: 0, Maximum: 2
- Minimum: 0, Maximum: 1
- Minimum: 1, Maximum: 2
- Minimum: 0, Maximum: 3
- Minimum: 1, Maximum: 5

Fig 1. Timer Feature for Quiz

- **Tag Questions:**
  - Points: 2
  - Implementation Status: We implemented the functionality using a bookmark checkbox which when checked shows the tagged questions in the Marked Question Section. The user can uncheck the bookmarked question which removes the question from the Marked Question Section. The feature is named 'Bookmark' with a checkbox as shown in Fig 1. The implementation involved using CSS and javascript methods to redirect to the Question.

- **Ability to Quit the session:**
  - Points: 1
  - Implementation Status: This feature allows the user to quit the practice session or quiz in between. To develop this functionality we have added a Quit button for both the Quiz and Practice session which when clicked redirects the user to the

Select Quiz Page in case of Quiz session and to Select Category in case of Practice Session. The feature is displayed in Red color as shown in Fig 1. This feature was implemented using a simple redirection method in javascript which changes the href to the Select Quiz or Practice Page.

- **Omni-login using Google and Facebook:**
  - Points: 3
  - Implementation Status: For signing/logging in, we have used the devise, omniauth-google, and omniauth-facebook. We modified the navbar and body styling a little bit, apart from the existing styles. For multiple authentications, we had to add the Secret key and Client ID of our API using Google API console and Facebook authentication. For example, once the user clicks on the button 'Login with Google' he is redirected to the Google authentication page. This feature is shown in Fig 2. The implementation required setting the Google Secret key and ID in the case of Google as Environment variables. Which were used by Google Oauth2 callback functions to verify that it is getting the user login request from a valid user when the callback happens. The user was redirected to a Google Login page as shown in Fig 3. The implementation involved checking the method of login, the user privileges, and is it already persisted in the database or not. If it's not then the current user is set and the user is persisted on a callback from the Google API. To set up the Google Console API follow the guide https://developers.google.com/identity/protocols/oauth2. This involves setting up the URI and the redirect URI in the Google Console API. An example is shown in Fig 4. Similarly, you need to setup Facebook Developer app and to do so follow this guide: https://developers.facebook.com/docs/facebook-login/access-tokens/. Set up below Environments variables in your production or development environment.
    For Google:
    1) GOOGLE_APP_ID - set Client ID as shown in Fig 4.
    2) GOOGLE_APP_SECRET - set Client Secret as shown in Fig 4.

    For Facebook:
    1) FB_APP_ID - similar to GOOGLE_APP_ID check-in FB APP
    2) FB_APP_SECRET - similar to GOOGLE_APP_SECRET

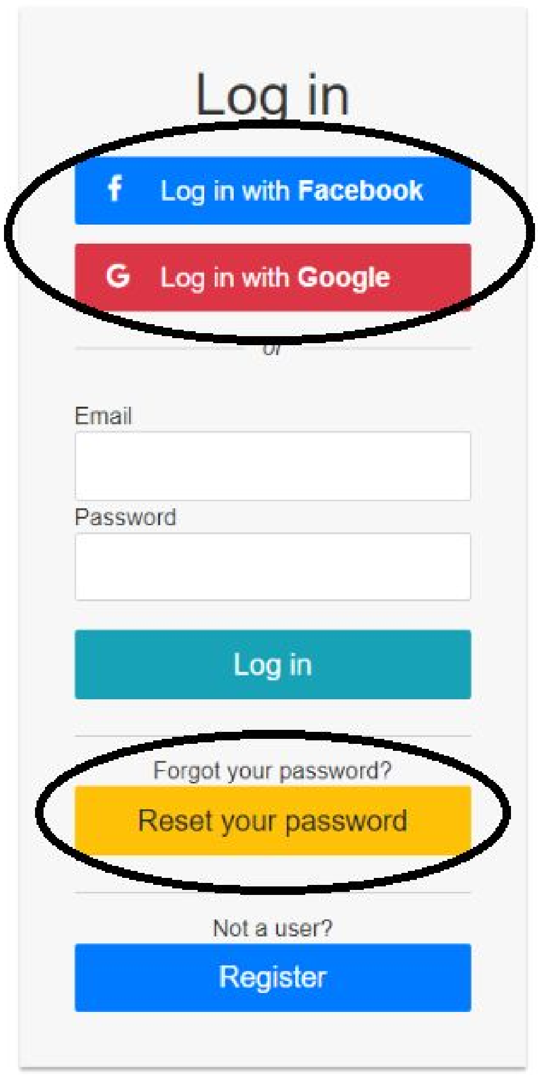
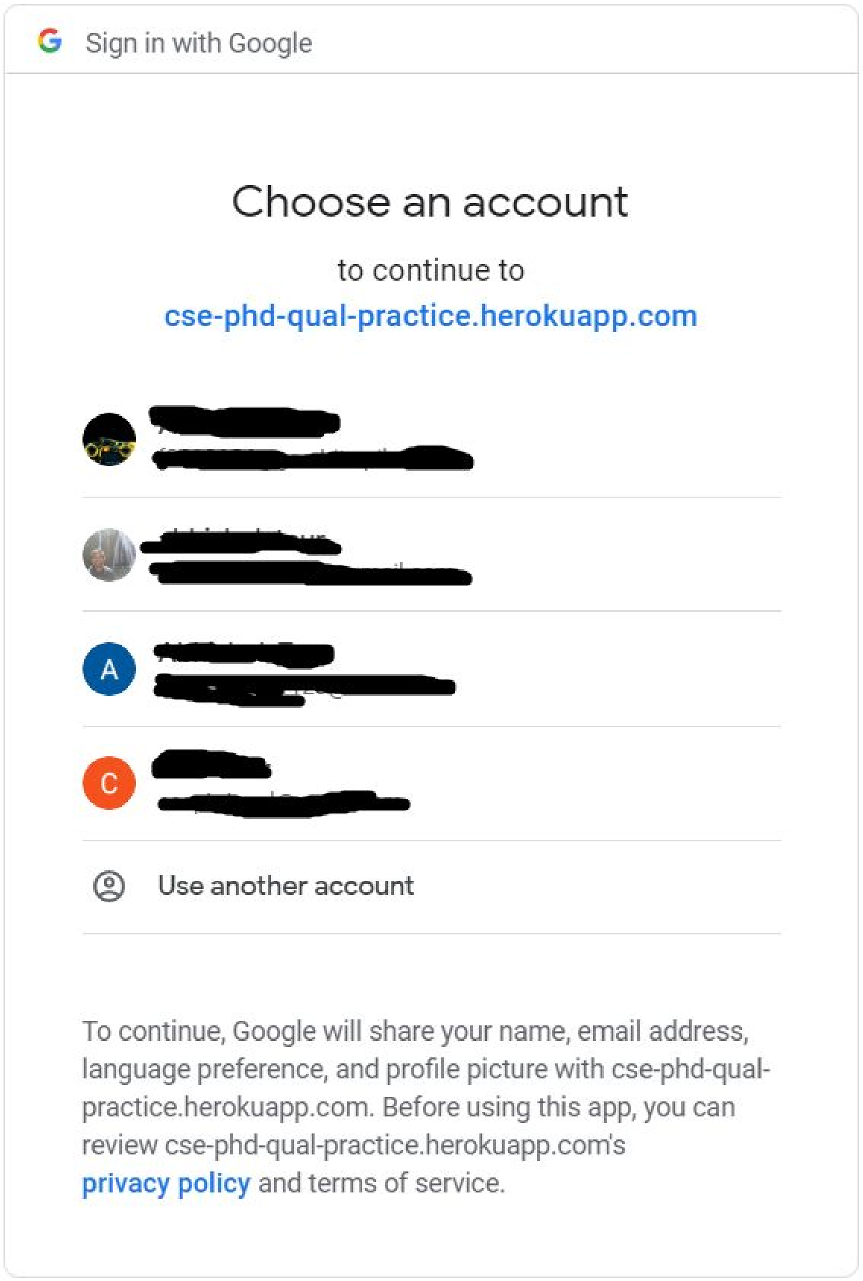


Fig 2. Login with Google, FB and Reset Password


Sign in with Google

Choose an account

to continue to

cse-phd-qual-practice.herokuapp.com

[redacted]

[redacted]

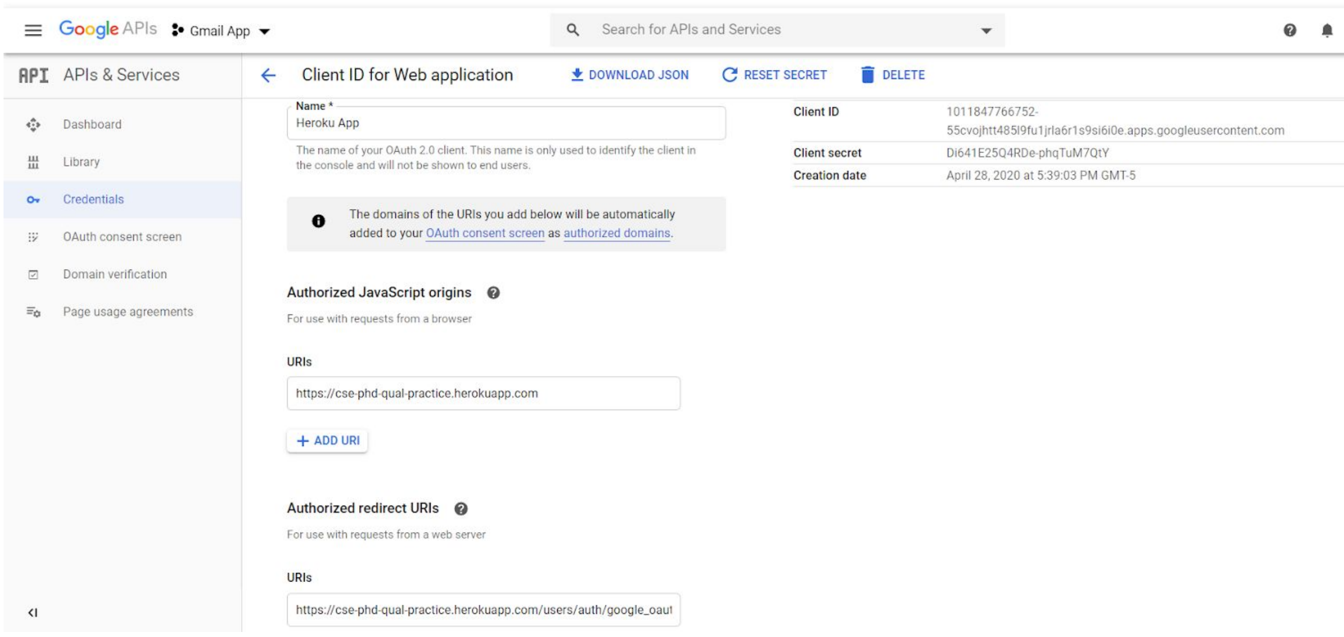
[redacted]

[redacted]

Use another account

To continue, Google will share your name, email address, language preference, and profile picture with cse-phd-qual-practice.herokuapp.com. Before using this app, you can review cse-phd-qual-practice.herokuapp.com's privacy policy and terms of service.

Fig 3. Google Oauth2 Redirection page



Client ID for Web application

Name *
Heroku App

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins

For use with requests from a browser

URIs

https://cse-phd-qual-practice.herokuapp.com

+ ADD URI

Authorized redirect URIs

For use with requests from a web server

URIs

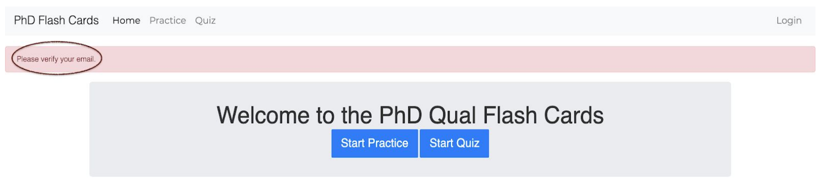
https://cse-phd-qual-practice.herokuapp.com/users/auth/google_oaut

Client ID	1011847766752-55cvojhtt485l9fu1jrla6r1s9si6i0e.apps.googleusercontent.com
Client secret	Di641E25Q4RDe-phqTuM7QtY
Creation date	April 28, 2020 at 5:39:03 PM GMT-5

Fig 4. Example Google Console API setting

- **Admin Password Recovery:**
  - Points: 3
  - Implementation Status: In the login page we have added an additional option 'Reset your Password' as shown in Fig 2. This feature uses the dotenv gem and the changes made to devise.rb and production.rb to enable the SMTP server and also uses the email address, entered by the user in forgot password page, to send an email. Once a user clicks on the link which is received via email, he is redirected to the Reset Password page where he can enter the new password.

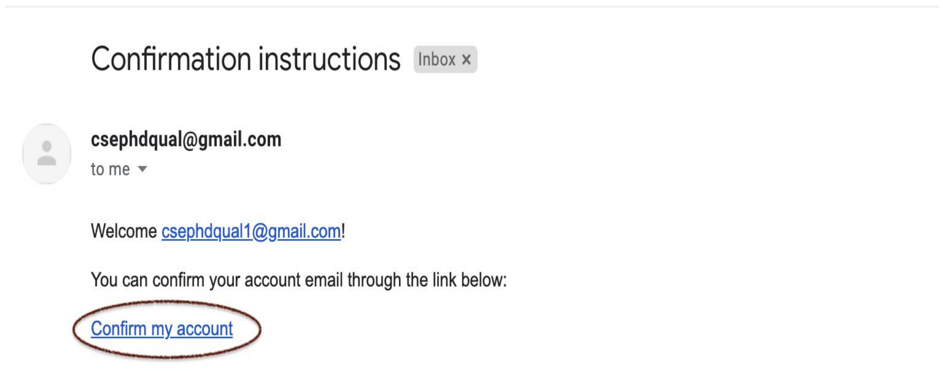
- **Email Verification at Registration:**
  - Points: 3
  - Implementation Status: If a new user registers, we send an email to the respective email address with a verification link as shown in Fig 6. When the user logs in without confirming the email, we show a notification on the Home page as shown in Fig 5. In case the verification email link expires, on clicking the link, we redirect to the page to request it again. Apart from the controller modification, we had to make changes in the model to accommodate the feature. We created a new migration with confirmation_token, confirmed_at, confirmation_send time. Once the user clicks on the verification link before it's expiry, the email address is validated and the user is shown the result as shown in Fig 7.



Please verify your email.

Welcome to the PhD Qual Flash Cards

Fig 5. Verification prompt



Confirmation instructions

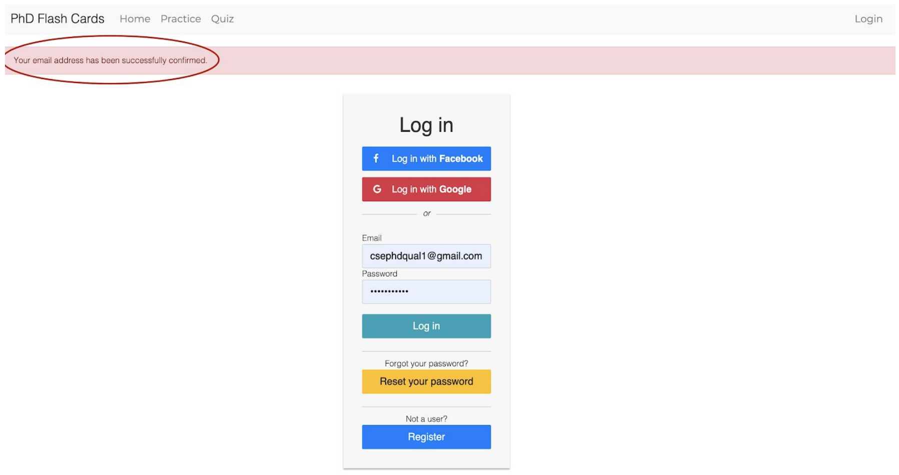
csephdqual@gmail.com
to me

Welcome csephdqual1@gmail.com!

You can confirm your account email through the link below:

Confirm my account

Fig 6. Confirmation email



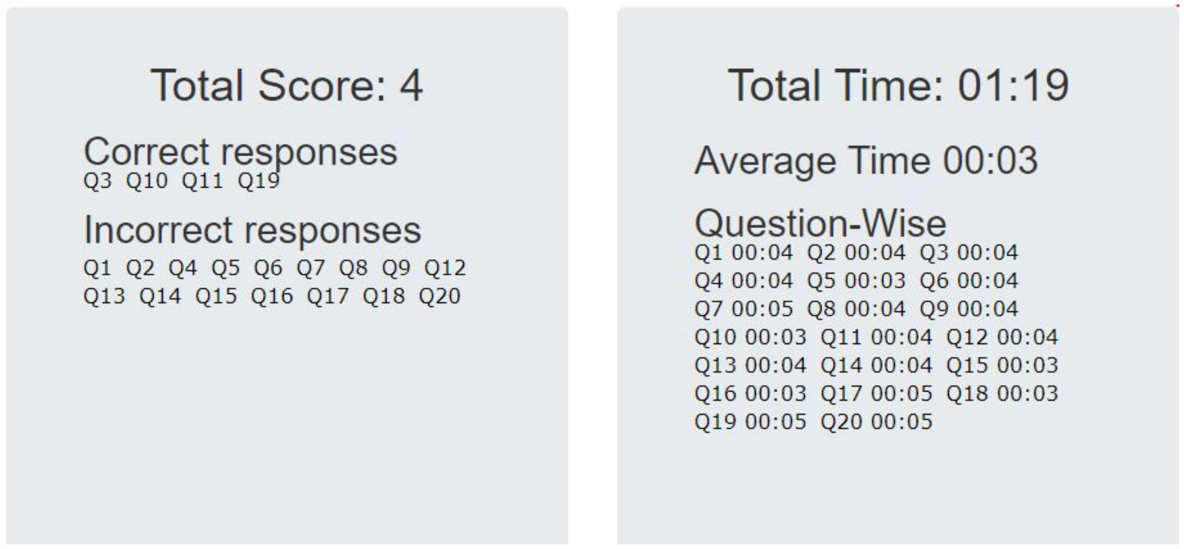
Your email address has been successfully confirmed.

Email: csephdqual1@gmail.com

Password: ••••••••••••

Fig 7. User login after email verification

- **Quiz Statistics:**
    - Points: 2
    - Implementation status: This feature shows the usage statistics regarding the quiz like the average time to solve all the questions, time for solving each question, Total time to solve all questions. To implement this feature we wrote a Javascript method that uses the timer values to calculate the time for each question and sums it up to get the total time. And there is a method that converts the time to string and then plugs it in the HTML when the submit button is clicked. The stats are as shown in Fig 8.



Total Score: 4

Correct responses
Q3 Q10 Q11 Q19

Incorrect responses
Q1 Q2 Q4 Q5 Q6 Q7 Q8 Q9 Q12
Q13 Q14 Q15 Q16 Q17 Q18 Q20

Total Time: 01:19

Average Time 00:03

Question-Wise
Q1 00:04 Q2 00:04 Q3 00:04
Q4 00:04 Q5 00:03 Q6 00:04
Q7 00:05 Q8 00:04 Q9 00:04
Q10 00:03 Q11 00:04 Q12 00:04
Q13 00:04 Q14 00:04 Q15 00:03
Q16 00:03 Q17 00:05 Q18 00:03
Q19 00:05 Q20 00:05

Fig 8. Quiz Stats

Scrum Iteration Details:

- **Iteration 0**:
    - 0 points completed
    - We received the legacy code in the form of a GitHub repository. We deployed the code in the Heroku test server to see in detail how the codes were implemented with relevance to the different existing features.
    - We came up with user stories after meeting with the customer and understanding the requirements.
    - We had two sets of user stories: ones that were completely new and the existing ones in legacy code that required modifications/improvements.

- **Iteration 1**:
  - 5 points completed.
  - Implemented user stories: Timer for the quiz, Omni Login with Google, and Facebook.
  - We added RSpec and Cucumber tests for all numbers of questions and with valid and invalid timer value, with and without logging in, to check if the timer is displaying an appropriate error message.
  - We added RSpec and Cucumber tests for Omni login by creating a new user, creating a user who is already registered, by logging in after registering for the first time.
- **Iteration 2**:
  - 5 Points Completed.
  - Implemented user stories: Tag Questions in Quiz and Practice, Forgot Password.
  - We tested tag questions by marking questions and checking if they are visible as marked ones. Also, we unmarked the questions to check if they are being removed from the list. The marked questions were clicked upon to check if the webpage can be rerouted.
  - We tested the forgot password feature by clicking on the reset password button to see if the redirection is as expected, by sending a reset password email, by clicking on the link in reset password email to redirecting to the password reset page where the user is prompted to enter the new password, by not entering the same password in 'new password' and 'confirm password' text fields, by correctly changing to the new password and logging in with it.
- **Iteration 3**:
  - 4 Points completed.
  - Implemented user stories: Quit Session, Email verification at Registration.
  - We tested by logging in as a user and starting a quiz followed by quitting it in between the quiz. This redirected the user to Select Quiz Page as expected after prompting to confirm the redirection.
  - We repeated the above tests for the practice session as well and found that the user was redirected to Select Category Page as expected.
  - We tested the registration of a user with the help of an email sent to the user for confirmation. Also, after doing this er tried to log in with the newly created user credentials to check if the database is updated.
- **Iteration 4**:
  - 5 Points completed.
  - Implemented user stories: Quiz statistics, Testing on different computing devices like mobile and Desktop using different browsers.

    - RSpec and Cucumber tests were added to see if the total number of questions solved, the total time is taken, the time is taken per question, and the displayed average time taken is correct.
    - Did manual testing of the features of the application. All the features that were tested are mentioned below and related bugs were fixed as well the bugs mentioned in the previous bug report were fixed. You can find the full fixed bug details here. There was on particular bug related to Mathjax library. We made code changes, but this feature is yet to be functional. It'll automatically work once they update. More details from the official page can be found here.
        - User Registration as an admin as well as a normal user(Includes the email verification too).
        - Admin makes other users as admin.
        - Login using the normal username and password.
        - Login using Google and Facebook.
        - Reset Password i.e. Forget Password.
        - Quit Quiz.
        - Solve the quiz with different questions and Timer values and test features like Next, Previous and Show Answer, Submit and Quit Quiz, and Show Statistics.
        - Practice session with Different categories(As many combinations as possible) and Quit Practice Session and Show answers.
        - Added and deleted categories(And only admin should be able to approve it).
        - Add, delete, and edit questions(And only admin should be able to approve it).
        - Sort Ascending and descending to the questions based on the column used(sort by ID).

Customer Meetings:

Meeting 1:
We met with Dr. Walker (customer) on Thursday 27th February 2020 at 2 pm. We discussed the customer requirements for the Project.

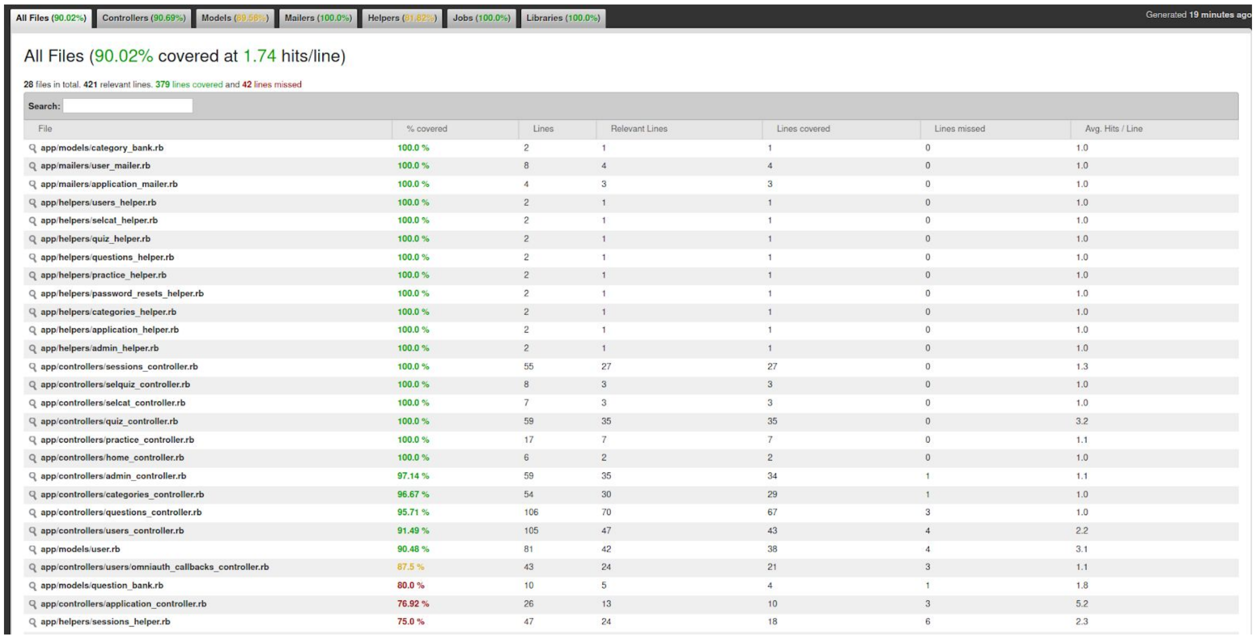
Meeting 2:
The meeting for iterations 1 and 2 with the customer Dr. Duncan Walker occurred on 04/03/2020 at 11:30 am CST. The team demonstrated the Tag questions in the Practice session feature, the Forget Password, Omni login using Google and FB to the customer.

The customer was satisfied with feature implementation and was notified with the issues which we are still facing and we will be demonstrating the fix for the issues in the next meeting.

Meeting 3:
The team met with Dr. Duncan Walker on April 30th at 11 am and discussed the user stories in detail for iterations 3 and 4. The stories that were discussed during the meeting were Quit Session, Quiz Statistics, Forget Password, Register with email verification, and the testing all the features and what all we tested. The customer was satisfied with the implementation.

BDD and TDD approach used:

We used a BDD approach before implementing the story by adding cucumber tests which were based on the user stories framed from the customer requirements. Every possible scenario for the user story was noted down and corresponding scenarios were in cucumber. The implementation was done in a way to first pass a happy path and then a sad path. Then subsequently more unit-level tests were written using RSpec to implement the TDD approach. This allowed us to check on unit-level each method with different data to get more insights into were the bug can might be. We were able to verify implementation bugs with this approach and also validate that the implementation is as per the customer requirement.



All Files (90.02%) | Controllers (90.69%) | Models (89.58%) | Mailers (100.0%) | Helpers (81.82%) | Jobs (100.0%) | Libraries (100.0%)

Generated **19 minutes ago**

All Files (90.02% covered at 1.74 hits/line)

28 files in total. **421** relevant lines. **379** lines covered and **42** lines missed

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/models/category_bank.rb	100.0 %	2	1	1	0	1.0
app/mailers/user_mailer.rb	100.0 %	8	4	4	0	1.0
app/mailers/application_mailer.rb	100.0 %	4	3	3	0	1.0
app/helpers/users_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/selcat_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/quiz_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/questions_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/practice_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/password_resets_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/categories_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/application_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/admin_helper.rb	100.0 %	2	1	1	0	1.0
app/controllers/sessions_controller.rb	100.0 %	55	27	27	0	1.3
app/controllers/selquiz_controller.rb	100.0 %	8	3	3	0	1.0
app/controllers/selcat_controller.rb	100.0 %	7	3	3	0	1.0
app/controllers/quiz_controller.rb	100.0 %	59	35	35	0	3.2
app/controllers/practice_controller.rb	100.0 %	17	7	7	0	1.1
app/controllers/home_controller.rb	100.0 %	6	2	2	0	1.0
app/controllers/admin_controller.rb	97.14 %	59	35	34	1	1.1
app/controllers/categories_controller.rb	96.67 %	54	30	29	1	1.0
app/controllers/questions_controller.rb	95.71 %	106	70	67	3	1.0
app/controllers/users_controller.rb	91.49 %	105	47	43	4	2.2
app/models/user.rb	90.48 %	81	42	38	4	3.1
app/controllers/users/omniauth_callbacks_controller.rb	87.5 %	43	24	21	3	1.1
app/models/question_bank.rb	80.0 %	10	5	4	1	1.8
app/controllers/application_controller.rb	76.92 %	26	13	10	3	5.2
app/helpers/sessions_helper.rb	75.0 %	47	24	18	6	2.3

We put a great effort into increasing the code coverage. Initially, it was 40% and most of the controllers were not tested or dummy. We had to take this as a parallel to improve the quality of the product. Please find the final coverage report screenshot, same you can generate by running rspec in the project folder. As you can see the last few components have very low cc, this is due to the fact that we don't use those any more. One more point to mention, we have used factorybot to create mock object and reuse. You can find more quick intro from https://devhints.io/factory_bot

Configuration Management:

The system information like the Categories for the quiz, Questions for different categories, and the baseline infrastructure was management by rake and we had a seed.rb where the categories and questions were stored. Apart from the rake, we used Github as the Version Control System. The developers created their own branches to implement features or forked the Scrum Master Github repo. The branches whose code was finally merged in the master merge were: Code-coverage, feat-auth. Some gems that we used were omniauth-facebook, omniauth-google-oauth2, devise, simplecov, dotenv-rails, rubocop-rspec. The benefits of these gems included Oauth2 implementation for Google and FB, a framework for user registration and session management using Oauth2, Password recovery, coverage report generation, factory bot for RSpec.

User Account Details:

For the Google Oauth2 login, Forget Password, and Reset Password we created a new Google account, and similarly, for Facebook Oauth2 login we needed a Facebook Developer account. This account has been used to generate the Oauth 2.0 Client ID in Google Console API as well as Facebook Developer Application.
The details for the accounts are as below:

Google, FB email: csephdqual@gmail.com
Google, FB Password: CsePhdQual123

Links:
Heroku : cse-phd-qual-heroku
Github: cse-phd-qual-github
Pivotal Tracker: pivotal-tracker

Facebook oauth account creation.
The facebook authentication depends on the facebook oauth service. Please follow this link https://developers.facebook.com/docs/facebook-login/access-tokens/ to make it working. In nutshell, you have to follow the following steps.

1. Create an account from https://developers.facebook.com.
2. Create a new application.
3. Add oauth service to your application.

These steps are generic ones. But you have to whitelist the callback URL for omni-authenication to work. https://<hostname>/users/auth/facebook/callback is the format of your callback url. For local development use localhost:portnumber. This configuration can be found in the setting of facebook login product in your product.
There is a restriction on the development app. This only works with an invited account(by default only with creator account.). You have to publish the app to work across the generic account. Then in the application, you have to set the FB_APP_ID=*** export FB_APP_SECRET=***. This can be done using the export command in local Linux development, set in windows machine. Heroku requires you to use config command. More information can be found in https://devcenter.heroku.com/articles/config-vars link.

Google oauth account creation
Google oauth2 has a very straightforward approach. All the information can be found at the following link https://developers.google.com/identity/protocols/oauth2. To begin, obtain OAuth 2.0 client credentials from the Google API Console. Then in the application, you have to set the GOOGLE_APP_ID=*** export GOOGLE_APP_SECRET=***. This can be done using the export command in local Linux development, set in windows machine. Heroku requires you to use config command. More information in https://devcenter.heroku.com/articles/config-vars link. More information can be found on the top of the page.

Mail Server Setup
Application has a dependency on the smtp mail server for delivering forgot password and resets password features. Currently, we are using the Gmail account for this purpose. Firstly, you have to enable https://myaccount.google.com/lesssecureapps feature. After you have to set the GMAIL_USERNAME, GMAIL_PASSWORD variable. You can follow the steps described in the google OAuth last part to understand the variable value setting.