

Brock University

COSC 4P02

Final Project Report

Interactive Timeline System

April 26, 2022

Group members:

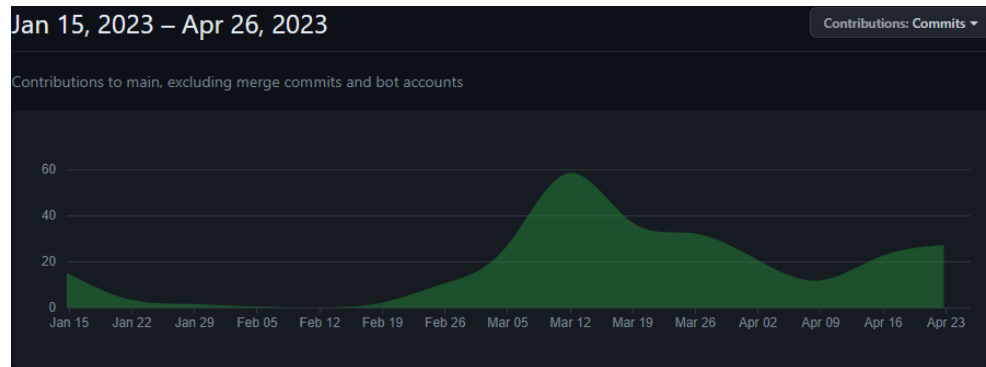
Yathusan Thulasinathan - 6735955
Athavan Jesunesan (AJ) (Leader) - 6705271
Carl Paladino - 6850796
Yousaf Shah - 6704969
Steve John Abraham Jayaseelan - 6856694
Adil Bedri - 6820344
Fahad Khan - 6688998







GitHub URL:



<https://github.com/bit-yottabyte/COSC-4P02-Project/blob/main/4P02%20Final%20Report.pdf>



Github Logs





Disclaimer: the GitHub logs do not fully encapsulate or represent the contribution of all members. This is due to some complications that arose when pushing & pulling code. These complications resulted in the contribution of code being inaccurately represented. Code would also be written collaboratively on some occasions which also skews the contributions. Lastly, Fahad had an issue with his branch and therefore is not represented in the contributions; however, he did still contribute to the development



















commit2 Fahad Khan committed 2 days ago	 533fc5e	
commit1 Fahad Khan committed 2 days ago	 a29c3c8	
Merge branch 'Dev' of https://github.com/bit-yottabyte/COSC-4P02-Project Fahad Khan committed 2 days ago	 b79dad7	

Merge branch 'Dev' of https://github.com/bit-yottabyte/COSC-4P02-Project Fahad Khan authored and Fahad Khan committed 4 days ago	 9755c75	
---	---	---

Commits on Mar 23, 2023		
ryhjhfhg Fahad Khan committed on Mar 23	 217544b	

quiz-update Fahad Khan committed on Mar 12	 c28a1ae	
quiz-update Fahad Khan committed on Mar 12	 cd22b41	

quiz-update Fahad Khan committed on Mar 12	 b235445	
Merge branch 'Dev' of https://github.com/bit-yottabyte/COSC-4P02-Project Palagino committed on Mar 12	 815aa1f	
update js link Palagino committed on Mar 12	 de2a166	
quiz-update Fahad Khan committed on Mar 12	 4a95b4f	
quiz Fahad Khan committed on Mar 12	 5728eee	
quiz Fahad Khan committed on Mar 12	 e54ef09	
Merge branch 'main' of https://github.com/bit-yottabyte/COSC-4P02-Pro... Fahad Khan committed on Mar 12 ✓	 2c1d46d	
quiz Fahad Khan committed on Mar 12	 73e54df	

AJ: bit-yottabyte

Carl: Palagino

Yathusan: yathy1040

Yousaf: YousafShah10

Steve: stevejohn20

Adil: adilkello

Fahad: faeskhan

Contributions:

AJ:

- Project manager
- Ensured deadlines were met
- Created the timeline system page
- Organized team meetings and goals
- Reviewed all pages and edited as required
- Created Technical Solutions whenever problems arose
- Quality assurance
- Created all team docs (reports, architecture, etc.)

Carl:

- Lead full-stack developer
- Designed front-end architecture
- Managed backend development
- Designed and implemented all node.js files and endpoints
- Retrieved and implemented all current data
- Reviewed and adjusted every page to ensure consistency and full functionality
- Helped with creating team docs (reports, architecture, etc.)

Yathusan:

- Helped with cleaning the database and creating mongoose models for the database itself
- Created Questionnaire page
- Made answers on the questionnaire page be added to the database for viewing by admins of the web app
- Helped with cleaning up pages
- created collections like users for login/admin functionality
- Designed and implemented login functionality
- Designed and implemented admin dashboard
- Allowed admins to add/edit/delete artifacts/events from the system using forms to add/edit them and a button on artifacts and events to delete them
- Made admins search for events and artifacts so they could edit/delete a specific artifact/event
- Allowed admins to view questionnaire answers received from users to see feedback on the museum

Yousaf:

- Designed and created artifact details page
- Created artifact query based on ID using this to populate the artifact details page

- Created the tag query and express endpoint
- Assisted with planning of project and documentation
- Integrated the tag query with the index search page
- Made necessary changes to data importer and artifact database to support tags

Steve:

- Team Member
- Design on Figma before actual implementation.
- Created event details page.
- Integrated the backend data to front-end.
- Assisted with planning and documents(database architecture, progress reports,etc)
- Focussed more on front end development
- Floor plan design for the museum(Room1-Room5)
- Occasional Black-Box testing

Adil:

- Created the landing page
- Helped with team javascript writing
- Floor plan design for the museum
- Assisted with documents (i.e progress reports)

Fahad:

- Designed the overall quiz page using Figma
- Implemented the front-end of the Quiz page
- Implemented some of the back-end functionalities of the Quiz page
- Implemented sample questions/answers as data into the MongoDB database
- Assisted with planning and completing the documents (database architecture, progress reports, etc.)
- Designed the map (floorplan of the museum)

User/Technical Manual

Landing Page

This page is the home page, it gives the user a brief description of the museum and allows them to search for an artifact. The search bar works by allowing the user to type out an artifact name, as they type a list of artifacts will appear below. From the list, the user can choose an artifact by clicking it, which takes them to the artifact page of the chosen artifact.

Timeline page

This page lets the user scroll through all of the artifacts in chronological order. Allowing them to choose the artifact they would like to get more information on by clicking the image which takes them to the artifact page to get more detail on the chosen artifact.

Artifact page

This page is used for providing the user with a more detailed page of a single artifact by pulling it based on its artifact ID. It gives a detailed description, image, date, tags, and where to find it in the museum. The tags are used for finding similar artifacts which are closely related to this current artifact and when clicked the user will be directed to the other associated artifacts. The user is then able to open those artifacts to view.

Event page

This page provides the user with a list of current and upcoming events listing them in order from oldest to newest. It allows the user to click an event to get more details by redirecting them to the event detail page.

Event detail page

This page provides the user with more details and information about a particular event which includes a description, venue, and data. This page also leads to registration for the events.

Questionnaire page

The user would enter their answers into the page and then they would submit the answers. The page would ensure that the user fills out the whole form as they cannot submit unless they fill out the whole questionnaire. As well, any invalid inputs like invalid

email addresses cannot be entered into the questionnaire as the form would tell the user to give a valid email address.

Quiz page

This page is used to allow users to test their knowledge regarding various artifacts and events in an interactive way. The page includes the question description, four answer options, a submit button, a scoreboard for the # of correct answers, a map reference that can be used for showing the user where the artifact or event display is located in the museum, and finally a button that takes the user to the next question. The map is referred to for each question by pulling the artifact ID of the artifact that is being quizzed on. The user will be given a series of 15 randomized questions which are pulled directly from the database table. The database table consists of 50 questions in total and it pulls 15 out of the 50 questions. The user selects the answer they believe is correct for each question and clicks submit. Users can only submit one of the four options. These options also have randomized color locations. Once the answer is submitted, it will show a 'Correct' or 'Incorrect' message indicating whether the selected answer is correct or not. Once the message is given, the user may click the 'Next Question' button which will direct the user to the next randomized question. This process will be a continuous cycle until all 15 questions are answered.

Login page

Here any admin is able to login to the web app to access the admin page. The backend here searches for the correct username and password and if both are correct, it lets the user into the admin portal. The backend database has a hashed and salted password that is compared to the password given to the website by the admin and when it is compared to be the same password, the admin is let in. A cookie is sent to the admin with the username and a user session id to ensure that the website knows that the admin is logged in and thus allow it to access the admin pages.

Admin artifact page

This page is where artifacts can be added, edited and deleted. There is a form for adding artifacts and there's a search function for artifacts. After you search for an artifact, a list of artifacts show up and on each artifact, there are buttons to edit or delete the artifact. Pressing delete, deletes the artifact and pressing edit takes the admin to another page where they can edit the artifact. The info of the artifact is automatically put into the form and the admin can edit all the fields of the artifact needed. For this page and the rest of the admin pages, if the user isn't logged in, the page just shows a 403: Forbidden error instead.

Admin event page

This page is very similar to the admin artifact page but it has to do with adding/editing/deleting events instead. This is where events can be added, edited, and deleted. There is a form for adding events and there's a search function for events. After you search for an event, a list of artifacts shows up, and on each artifact, there are buttons to edit or delete the event. Pressing delete deletes the event, and pressing edit takes the admin to another page where they can edit the event. The info of the event is automatically put into the form and the admin can edit all the fields of the event needed.

Admin questionnaire answer page

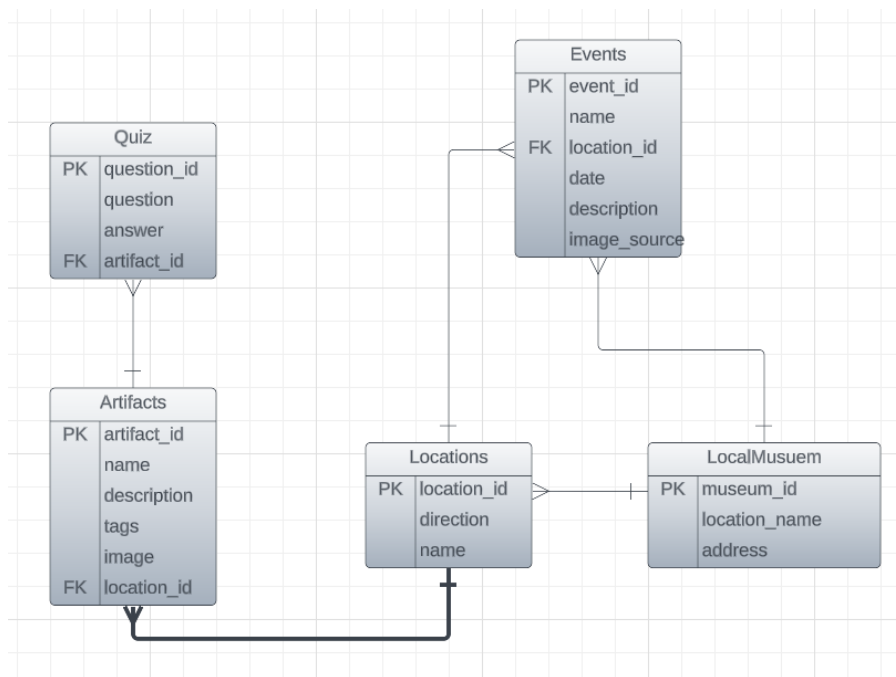
This is where the admin can see all the answers that users have sent to the admins of the museum when giving feedback on the questionnaire page. It shows the name, the email address, and some info about how the person felt about the museum and it shows all the answers that have been entered. The admin can look at the feedback and then decide what to do with the information given as a result.

Installation Manual

Our system uses Bootstrap and jquery which are external links to grab the resource hosted online, thus requiring a constant internet connection. Node.js was also used with the following packages:

- bcrypt@5.1.0
- cors@2.8.5
- express@4.18.2
- mongoose@7.0.4
- uuid@9.0.0

Along with MongoDB database clusters with the corresponding schemas to reflect the following database layout dependencies:



Our data for the artifacts came from an open source website that indexes a vast variety of open source Canadian museum data listed here:

<https://ingeniumcanada.org/collection-research/artifact-open-data-set-mash-up>

Within our source code exists a “data_importer.html” page which takes in a file argument to be uploaded at which point it will clean the file uploaded and insert the corresponding artifacts into the database.

After these requirements have been installed and implemented in the system, the last step is to ensure the node.js database file “database.js” is executed and running. With all these steps completed the website is fully functional and browseable.

Deployment Manual

To deploy this project any dedicated server platform that supports Node.js will work and the admin verification will have to be edited to check for the domain rather than the local host. All local host tags will have to be adjusted when deploying.

A MongoDB Cluster must also be created and connected to the server with the proper schema and fields included.

Test Documents/Results

Analyzing burndown charts

Within Microsoft Planner our scrum master would check to see which tasks have been completed and estimate the remaining workload to be completed by the end of the sprint. This allowed us to determine our velocity and see if we are meeting deadlines or not. Naturally with us all having many other courses to do amongst this project, we had a few times where some tasks had to get bumped into the next sprint, but overall this was a consistent way of keeping our group reaching our deadlines on time.

Sanity testing

This testing technique was very prevalent during our sprint process since our development team would perform sanity tests on features that were recently added or changed to ensure that they are working as anticipated according to the product owner and ensuring that they don't affect any other components.

Generally during our scrum meetings, we would be able to discuss the new features that each person was working on and figure out how we want to take on the remaining tasks to ensure they get completed.

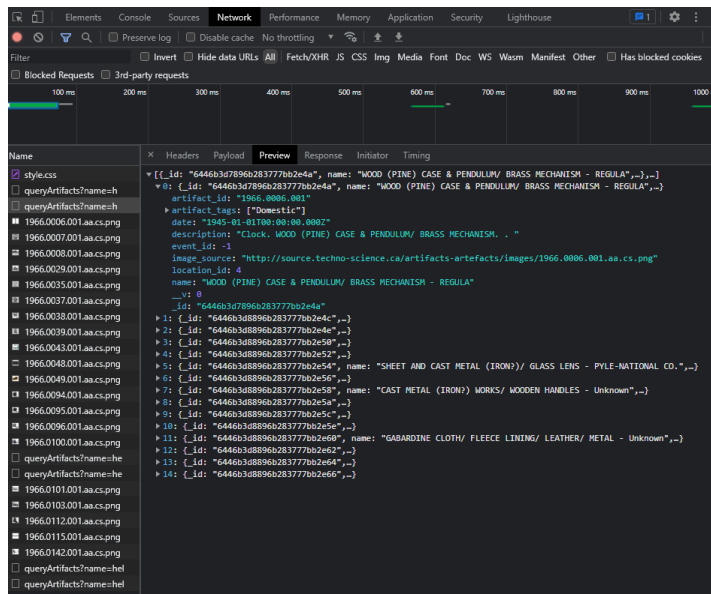
UI testing

Our product owner would relay their input to the team about how our progress is coming in terms of visual aspects along with functional aspects.

When any member of the scrum team was conducting UI tests we would also perform regression tests to ensure that any modified features would still allow previous components to continue functioning properly.

Manual testing for debugging backend calls

Many test cases consisted of analyzing the chrome developer tools to check the payload being passed from the backend AJAX calls into the front. This technique ensured that we were querying data that was consistent with what we had existing in our database.



Many variables within Javascript were printed out to the console during testing phases within the development to ensure that correct variables were being passed around or modified according to user input accordingly.

