Department of Computer Science
COSC 4P02 - Software Engineering - II

Final Report

Instructor: Naser Ezzati-Jivan
Date: April 28, 2024

Contents

1 Overview

This report provides a comprehensive insight into our team's compact and reliable learning management system, *BITES*. This report not only covers details about using iterative procedures in sprints but also how they are used to efficiently plan and complete tasks. The technical and personal difficulties faced by the team while developing this project are explored and every member's contributions are also discussed. Several logs and charts from Github and Jira are given to showcase the details of contributions and sprints. Moreover, details are also provided on how and why certain technical tools were implemented for developing this project.

2 User and Installation Manual

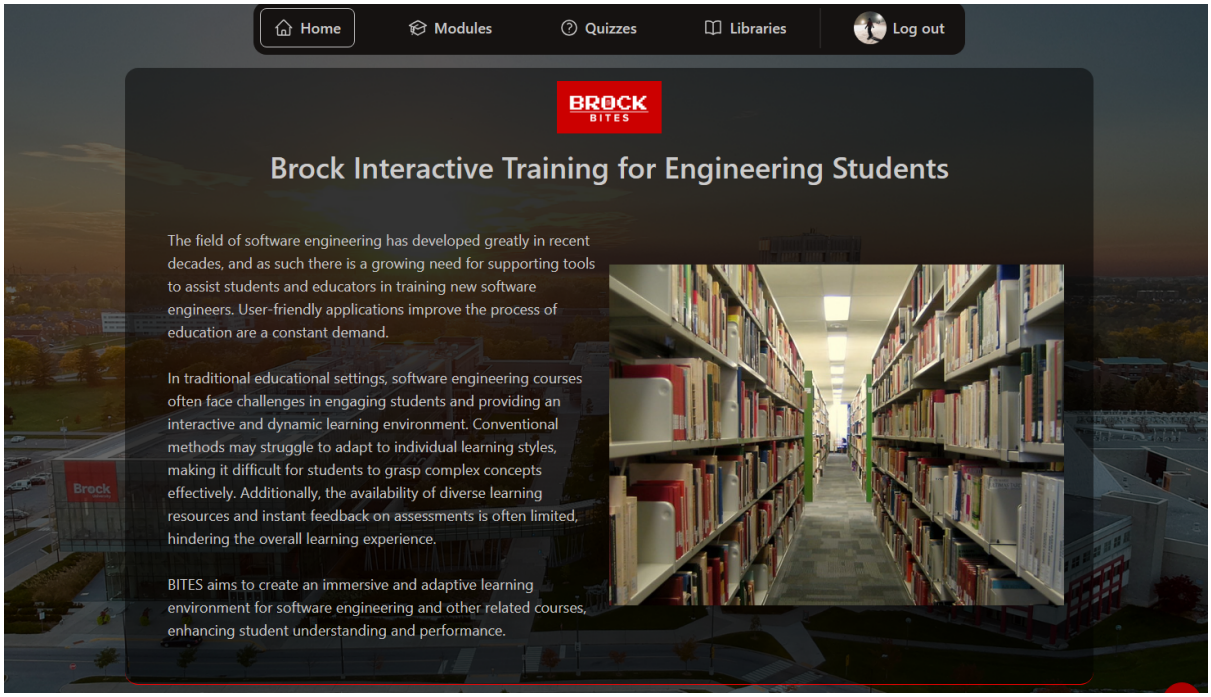
2.1 Accessing the web app

The website is available at https://bitesapp.org/. The website can also be run locally on your machine by following the instructions found in the installation manual below.

2.2 Features and Functionalities

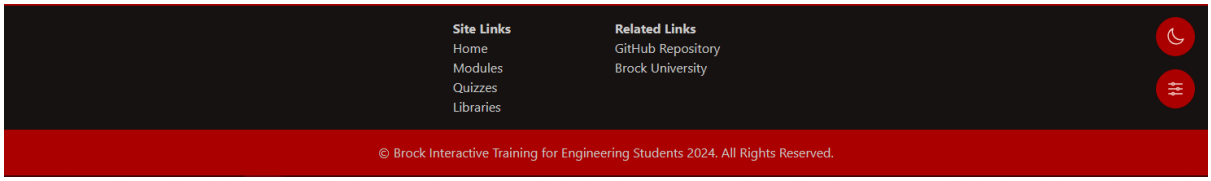
2.2.1 Landing Page

The landing page is intended to be brief and informative for visitors who are interested in learning more about the purpose of this website. Upon initial visit, users can read about the goals of our web page, see a demo of the user experience, and see an outline of content to be studied for software engineering.



2.2.2 Footer

The footer component can be found at the bottom of each page. It includes all the links you can find on the navigation bar, and additionally includes a link to the GitHub Repository for this project, as well as the link to the official Brock University website.



2.2.3 Modules Navigation

The modules tab is a main feature and is the only method of accessing any informative PDF slideshows and videos. The left column contains a collection of modules dedicated to teaching the fundamentals of software engineering. As the names suggest, the *Slides* button toggles the PDF viewer and the *Video* button toggles the video viewer (ex. YouTube).

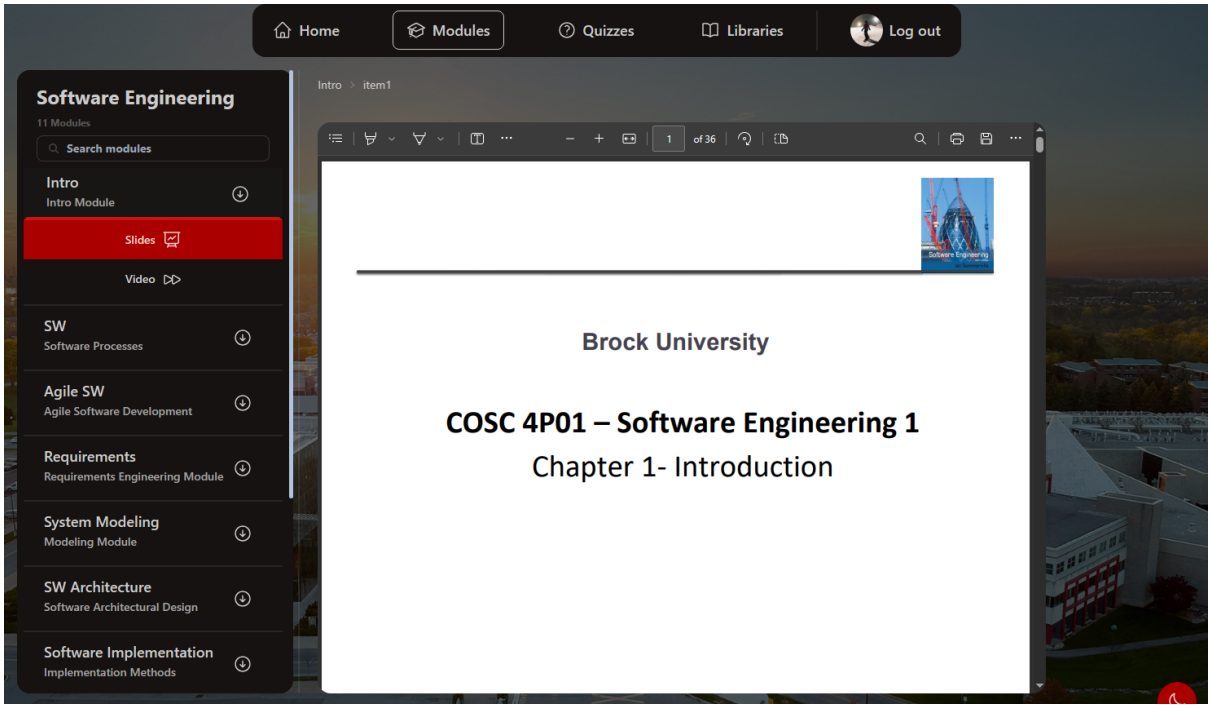


Figure 2.1: Modules PDF viewer

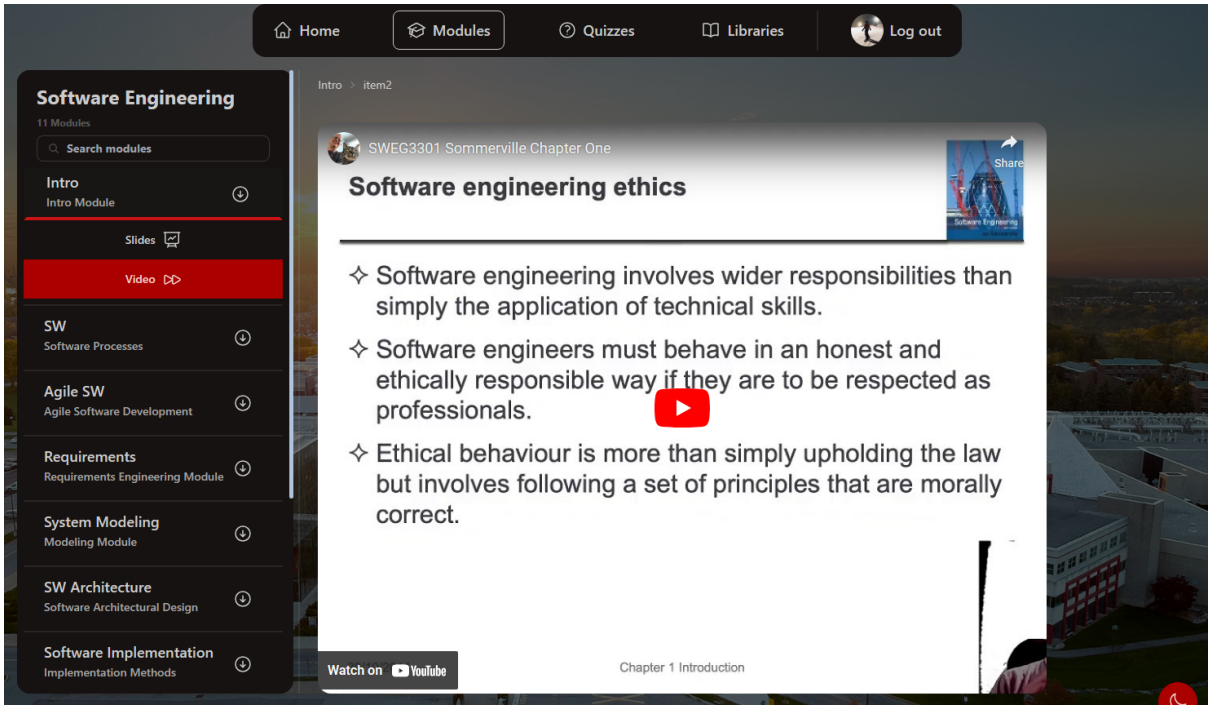


Figure 2.2: Modules video viewer (with YouTube)

Additionally, the user can use the search bar and query through all the modules to find a specific module to read on, allowing an easier and convenient experience to find the content the user needs.

Figure 2.3: Using the search bar to find a specific module

2.2.4 Quizzes Navigation

The quizzes tab is another main feature as it allows users to evaluate their learning progress and compare their results with others. The questions shown in figure 2.4 are randomly picked from a larger pool of questions. Users can see which option they choose based on the red highlight around it.

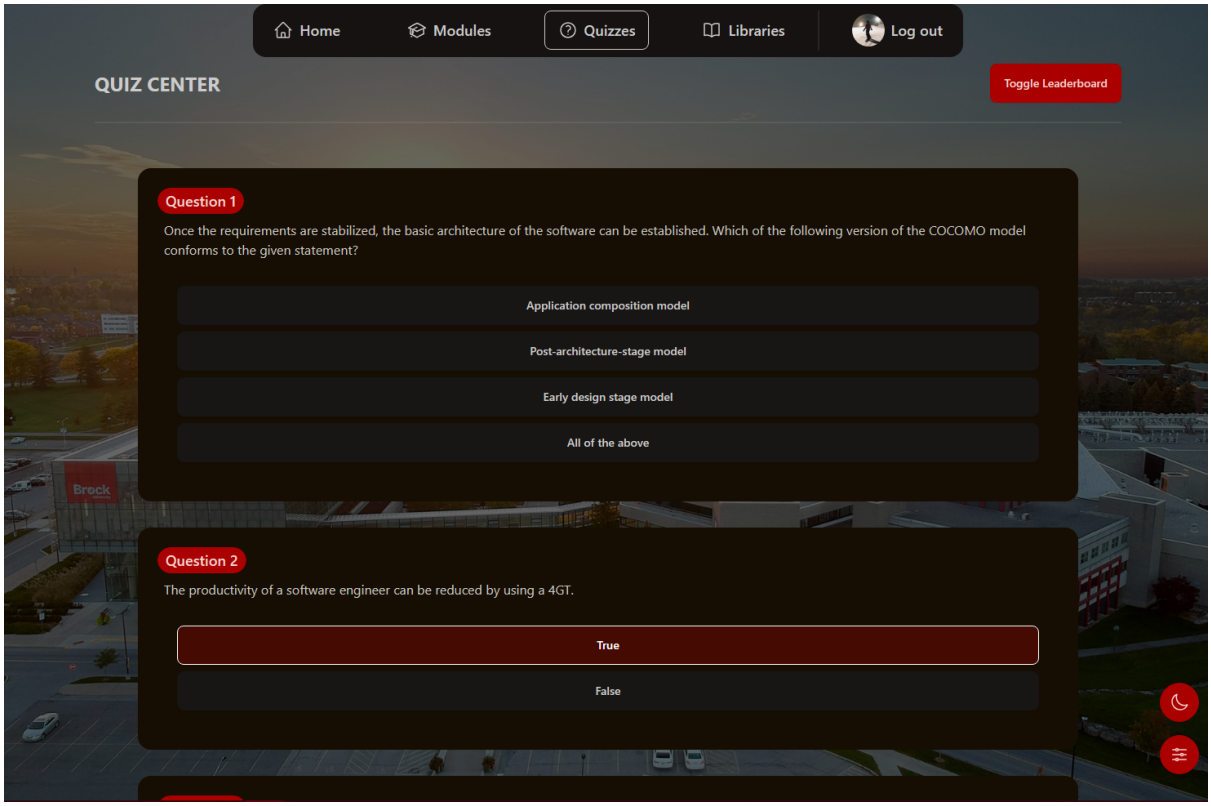


Figure 2.4: Top of quizzes page

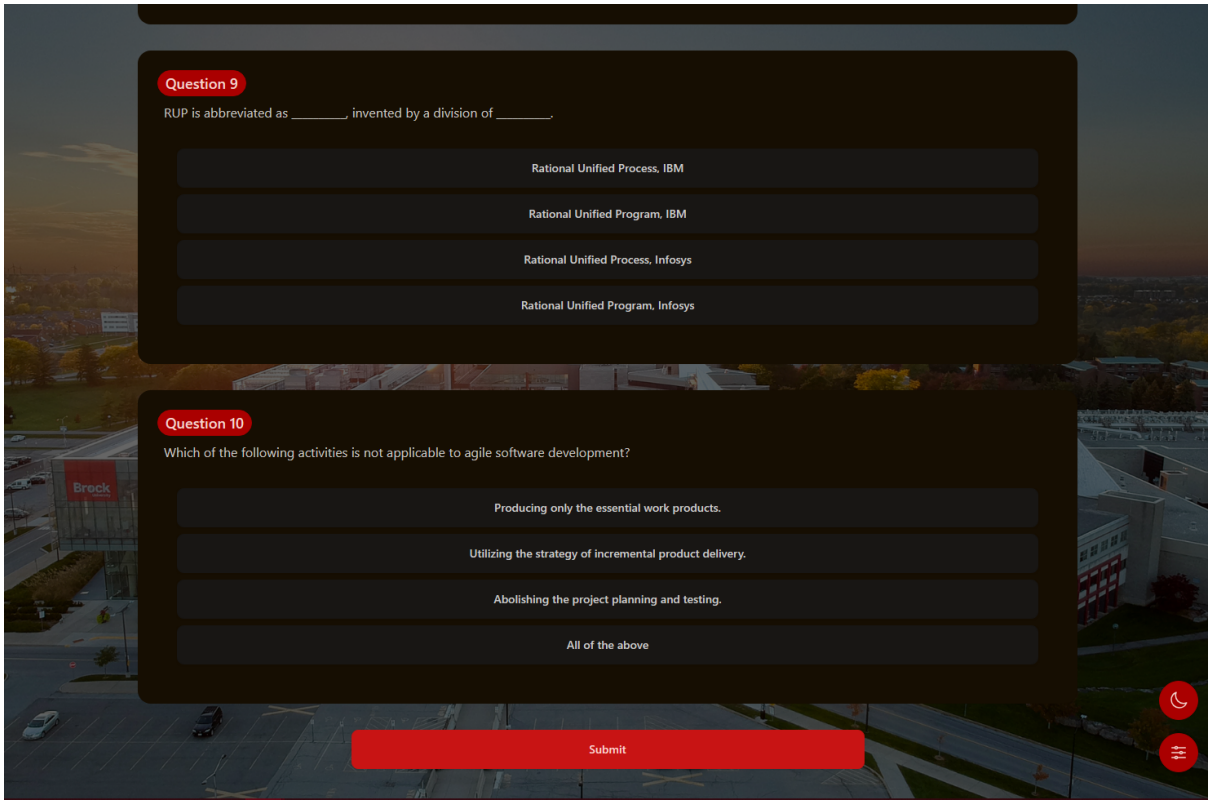


Figure 2.5: Bottom of quizzes page

Once a user submits the answers for their questions, a leaderboard subsequently pops up displaying the current score of the user, as well as a listing of other users' scores ordered from highest to lowest. This popup can only be closed by the X button on the top-right corner.

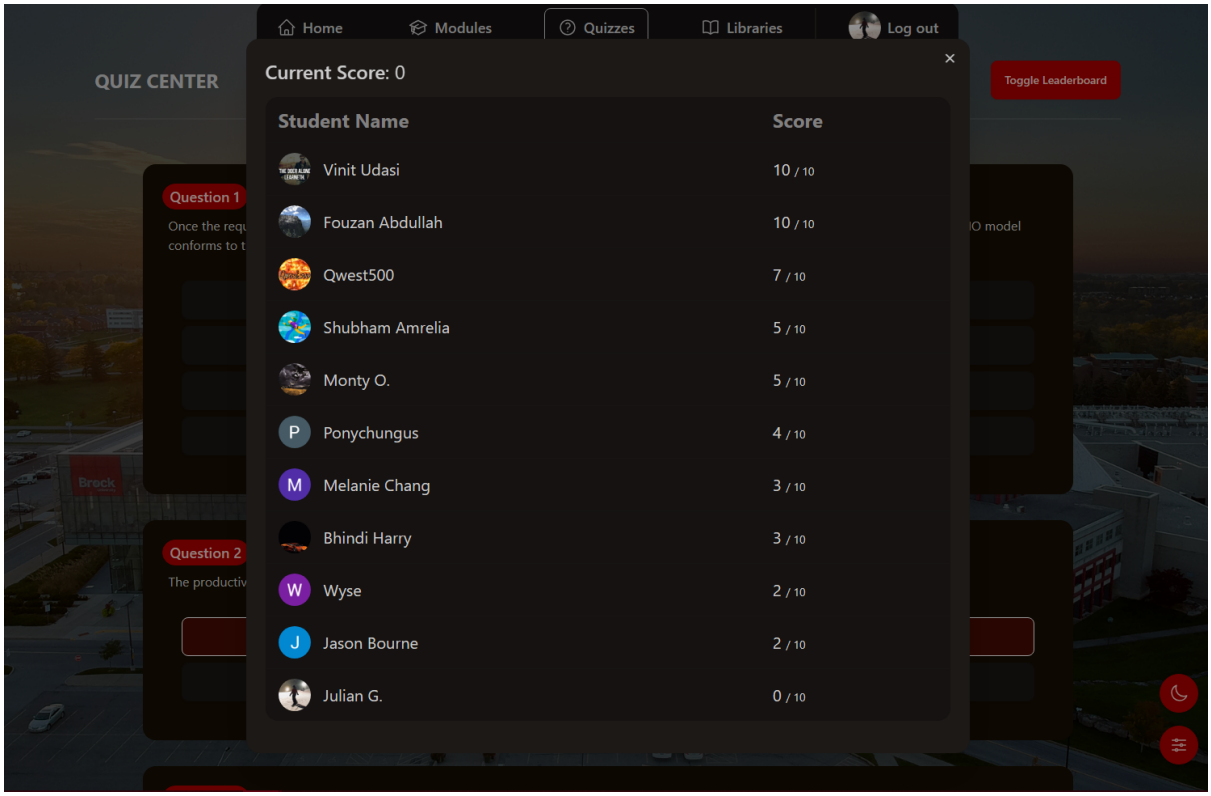


Figure 2.6: Leaderboard pop-up view

2.2.5 Library Usage

The library tab gives users additional resources to study from that are not directly included in the modules page. As shown below, a user can check through different tabs for different resources to use, which range from software engineering fundamentals to relevant job interview tips.

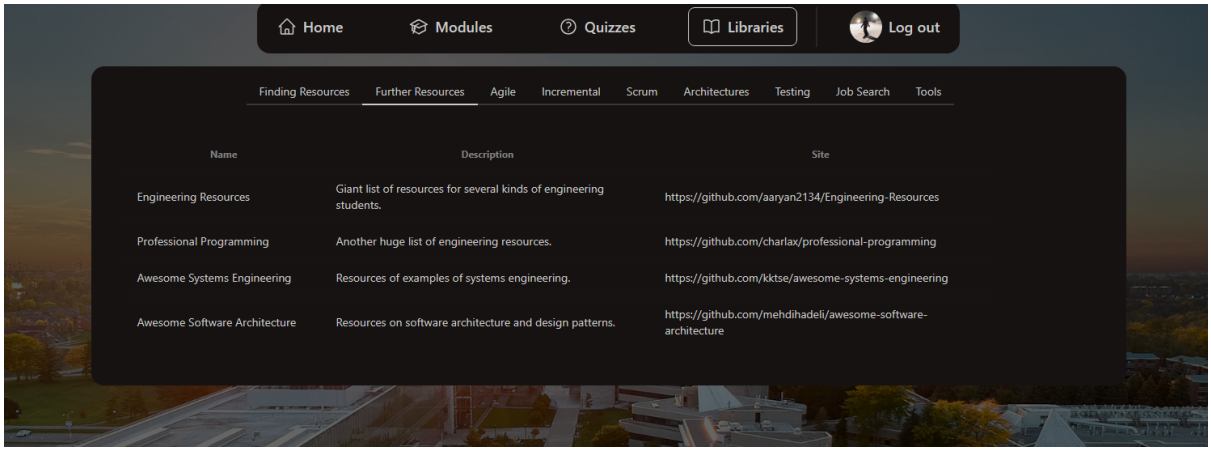


Figure 2.7: Libraries view

2.2.6 Dark/Light Theme Toggle

The user can switch between the dark (default) mode and light mode by pressing the red button with the crescent moon symbol located in the bottom-right corner of each page. When the symbol has a crescent moon, it indicates that it is in dark mode, and if it has a sun, it indicates it is in light mode. The dark and light modes are shown in figures 2.8 and 2.9, respectively.

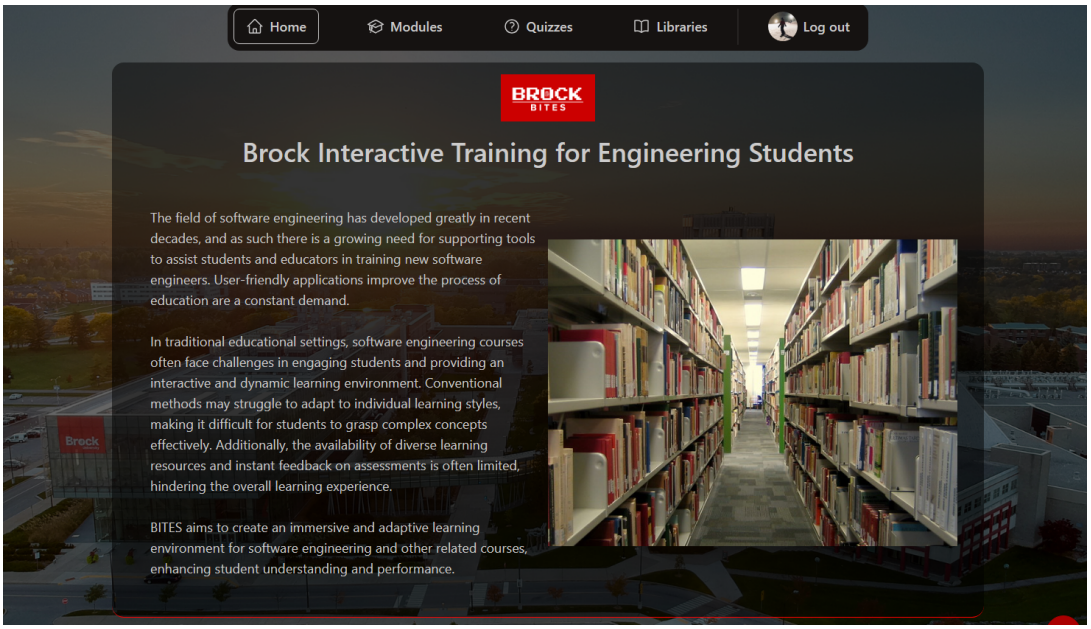


Figure 2.8: Dark (default) theme

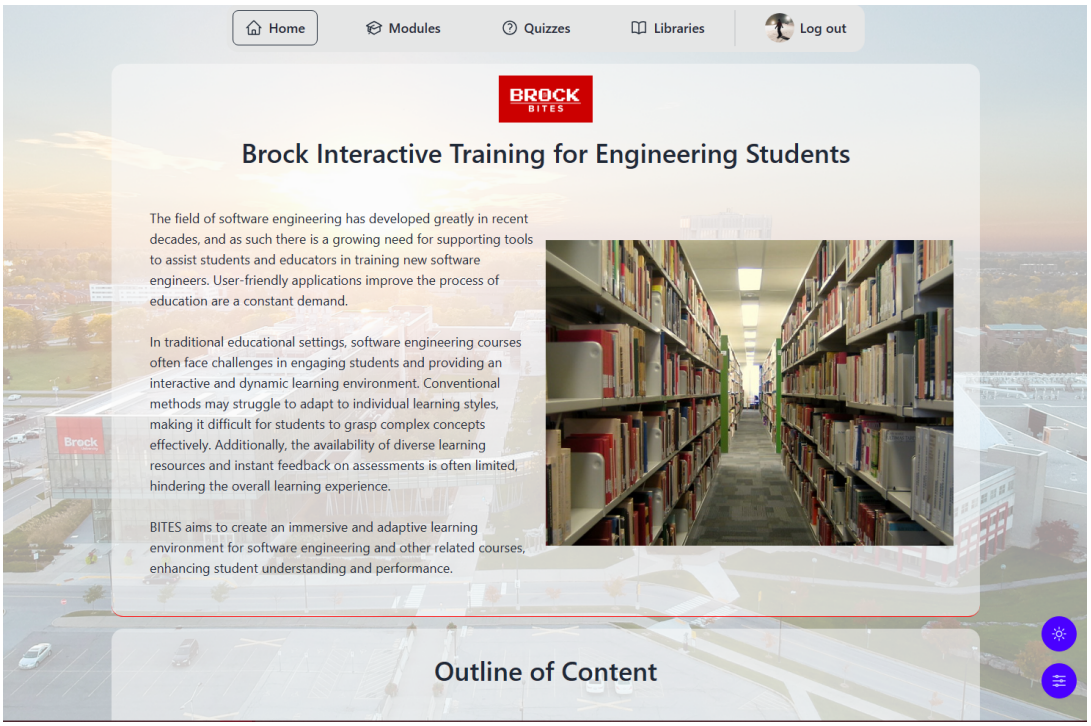


Figure 2.9: Light theme

2.2.7 Accessibility Menu

Below the dark/light theme button is the accessibility menu which is represented by 3 sliders. Clicking the button displays the accessibility options on top of the current page. These include a font size slider, dark/light theme button, and drop down menu of other web themes.

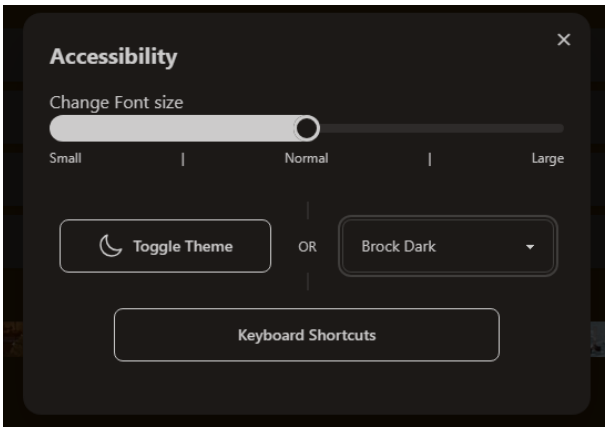


Figure 2.10: Accessibility pop-up options

The final button at the button brings up an alert that gives a list of keyboard shortcuts, as seen in figure 2.11.

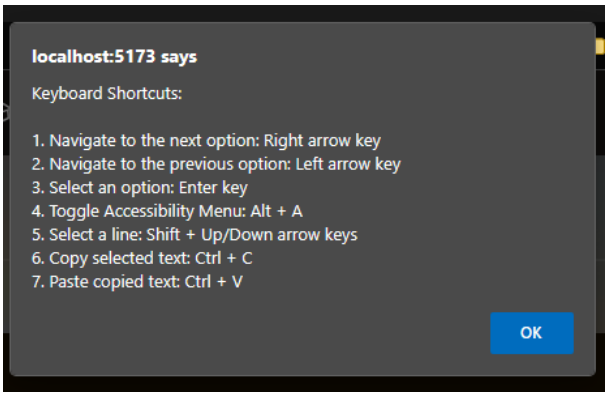


Figure 2.11: Alert pop-up

2.2.8 User Authentication and Navigation Bar

When a user visits the site they are assigned to be a guest by default.

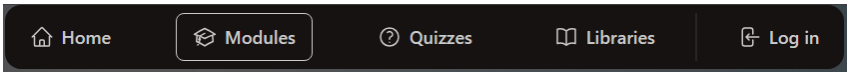


Figure 2.12: Navigation bar with the "Log In" prompt

If the guest tries to access the Modules or Quizzes tab, they are prompted to log in.

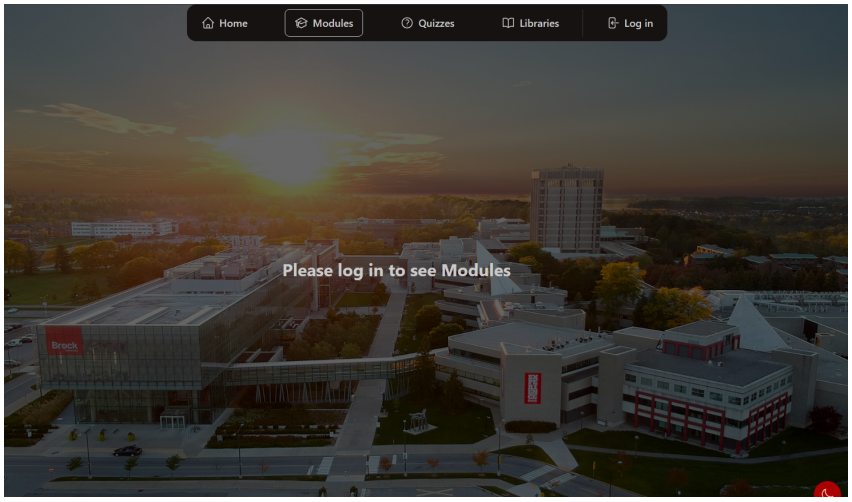


Figure 2.13: Accessing the Modules page as a Guest

When a user signs into an existing Google account, they are treated as a logged in student. Tabs such as modules and quizzes become accessible as previously shown. The navigation bar is updated with the user's account profile picture and a Sign Out option.

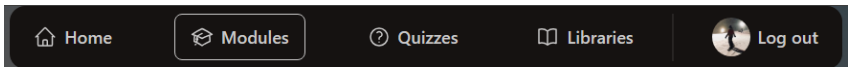


Figure 2.14: Accessing the Modules page as a Guest

When a user who is an instructor logs into their account, the navigation bar remains the same. However they are capable of adding to or editing the list of modules in the modules page.

The selector lets the instructor upload a file and a link to a video that is on an online server (it doesn't necessarily need to be a YouTube video but needs to be supported by iFrame). Once the instructor uploads the module, it will be shown on the sidebar and an entry will be created in Firestore, our database, for that module.

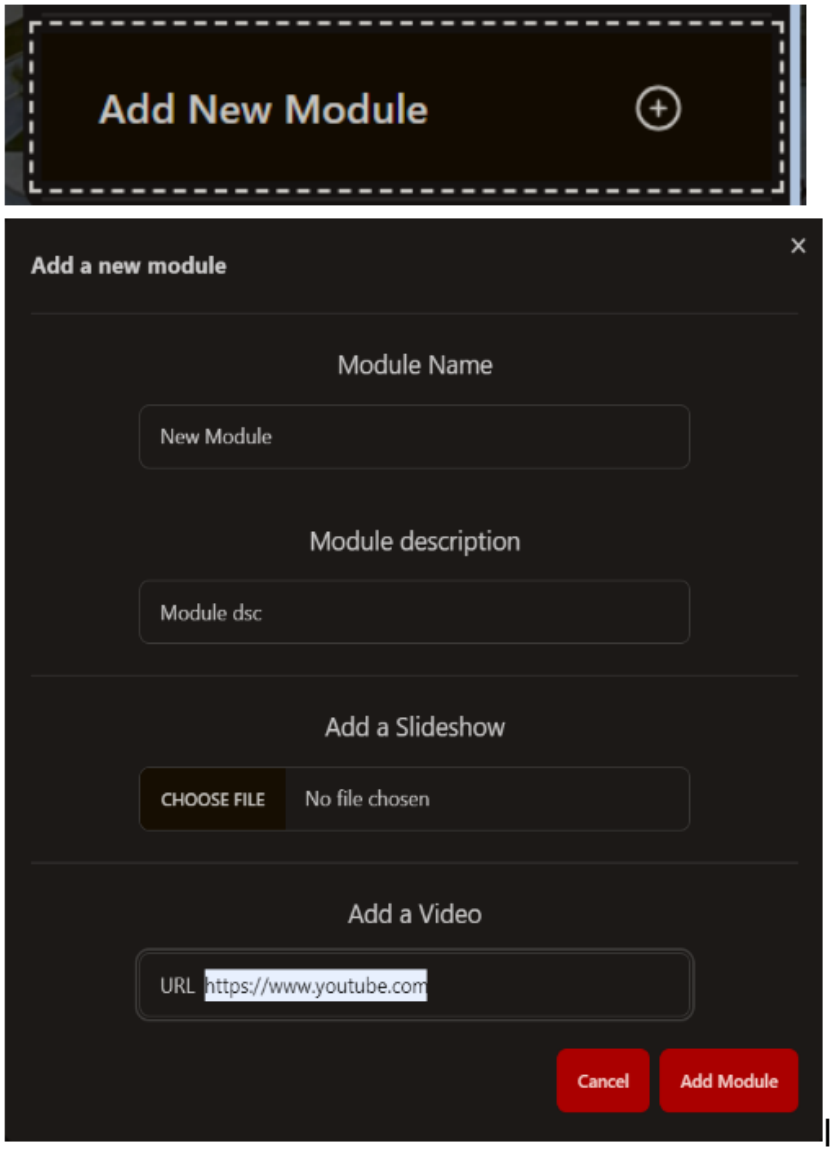


Figure 2.15: Accessing the Modules page as an Instructor

2.3 Installation Manual

The *BITES* application is hosted on our GitHub repository. The process of installing and using it is shown below. Installation requires Node version 18+, Typescript, and Firebase on your machine.

- Install the firebase command line interface with **Node Package Manager**:

  ```
  npm i firebase -g
  ```

- Clone the repository from GitHub

  ```
  git clone https://github.com/SWE-2024/COSC-4P02.git
  ```

- Navigate into the folder using the ***cd*** command

```
cd bites-app/
```

- Install dependencies with **Node Package Manager**

```
npm i
```

- Run Svelte (in development mode)

```
npm run dev
```

Alternatively, the project can be installed using other package managers such as yarn.

3 System Summary

Our team has developed an interactive learning system for software engineering fundamentals. This system is website based and supported for both desktop and mobile use. Upon visiting the site, the user is defaulted to being a guest, and is able to log in with an existing Google account to access all the intended features. Once logged in, the user is able to access the *quizzes* and *modules* pages which help users efficiently learn about software engineering fundamentals.

The user can access our primary features as briefly described in the *home* page. The *modules* page can be accessed for reading PDF slideshows and watching video lectures, while the *quizzes* page allows users to evaluate their knowledge and compare with other users through a leaderboard. The *libraries* page provides the user with a comprehensive list of resources to further their learning. Lastly, there is a collection of *accessibility* options located on the bottom-right corner of the website which enables options for a better viewing experience for certain users.

In terms of the software stack, we chose *SvelteKit* over other frameworks like *React* because it has superior performance and simpler syntax. It offers server-side rendering and built-in routing out of the box which reduces boilerplate and speeds up development. We used *TailwindCSS* which is different from vanilla CSS in that it easily adds mobile responsiveness and enables us to quickly style using pre-made classes, allowing us to focus more on design. For handling logins, we chose *Firebase Authentication* since it allows seamless integration with authentication methods—allowing users to log in with existing Google accounts. For storing files, we chose *Firestore* since it offers real-time data synchronization and is simple to set up, unlike other databases.

3.1 Login

The login feature allows the user to log into the system and access features such as the lecture module. Visiting the website the user will be greeted with the landing page shown on Figure 3.1

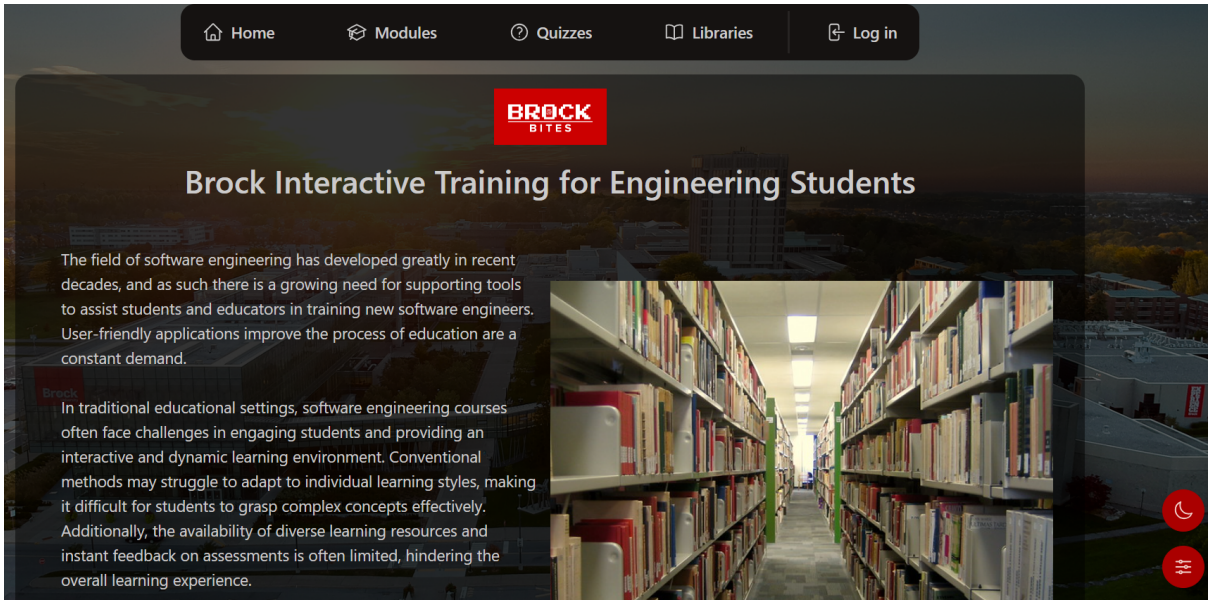


Figure 3.1: The home page

If the user tries to access the modules or the quizzes without being signed in, they will be asked to log in, as shown in Figure 3.2.

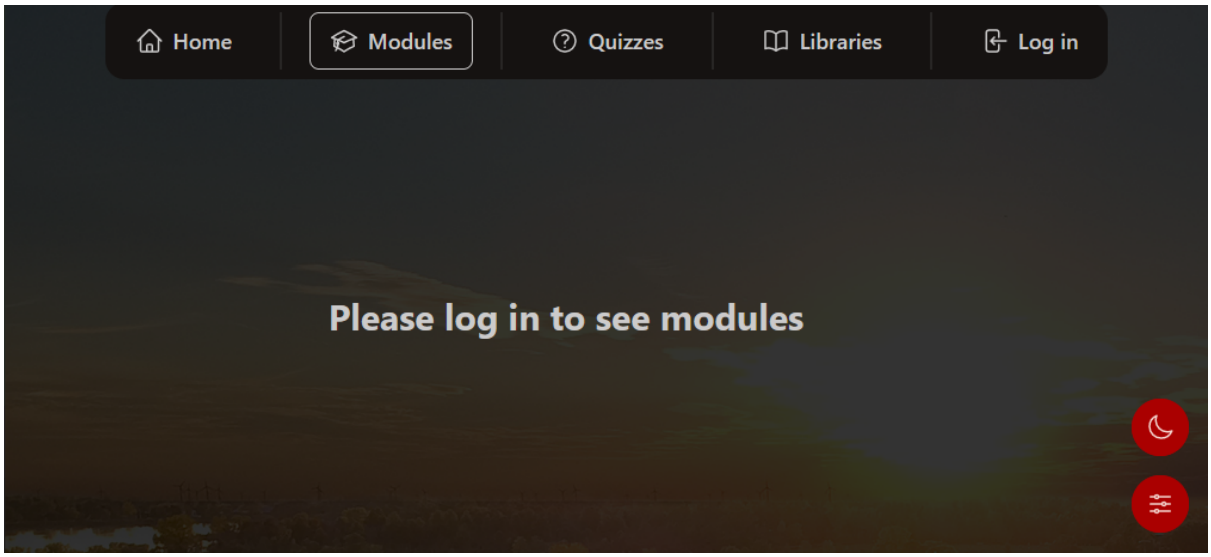


Figure 3.2: The modules page without being logged in

Logging in can be done by clicking the ***Log in*** button, located at the very right on the navigation bar. Clicking it opens a login modal which can be seen in Figure 3.3.

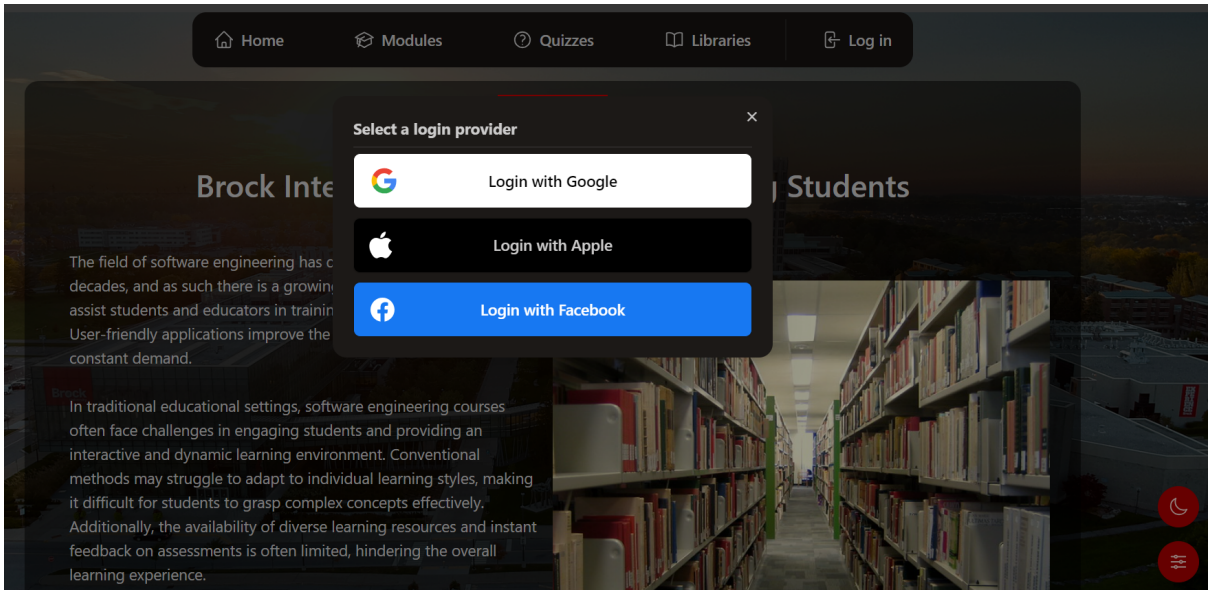


Figure 3.3: The login modal.

There are three login providers the user can choose from. We decided not to store login information ourselves and to offload that burden to one of these providers.

3.2 Modules

Once the user selects a provider and logs in, they can access one of the main features such as the modules, as shown in Figure 3.4.

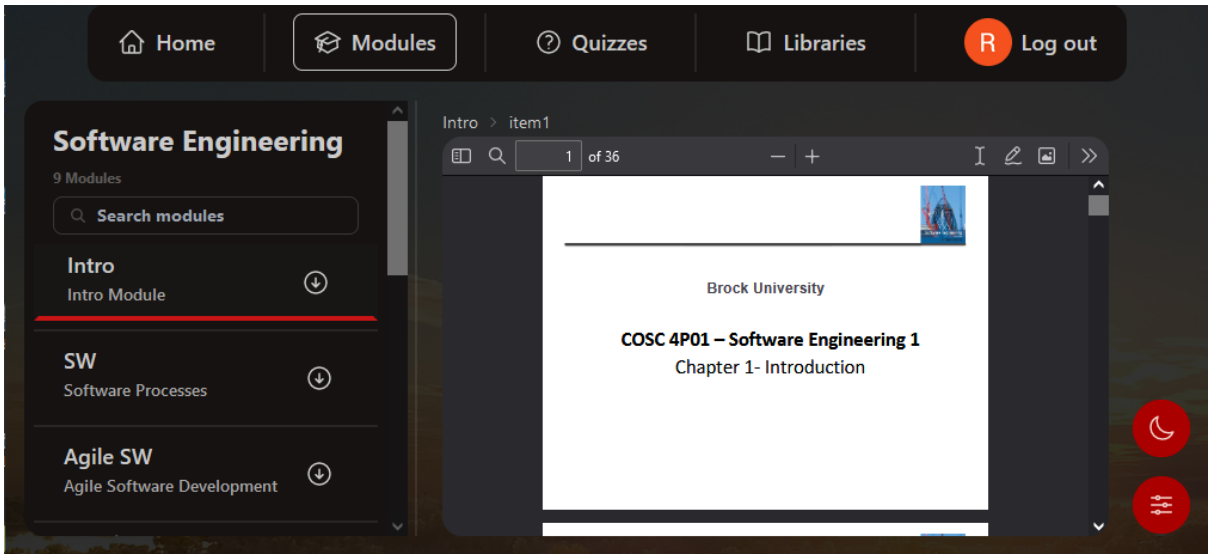


Figure 3.4: The modules page now that a user has logged in.

There are 9 modules that the user can use to learn the concepts of Software Engineering. These are:

- Introduction

- Software Processes
- Agile and Scrum
- Requirements Engineering
- System Modelling
- Software Architecture
- Software Implementation
- Software Testing
- Software Evolution

The user has the option to read through the lecture PDFs and also watch a *YouTube* video for each of these topics. The module items include an icon besides them, indicating whether it is a lecture PDF or a YouTube video as shown in Figure 3.5. For ease of use, the user can also search any module by its name using the search bar located at the top of the list of modules as shown in Figure 3.6.

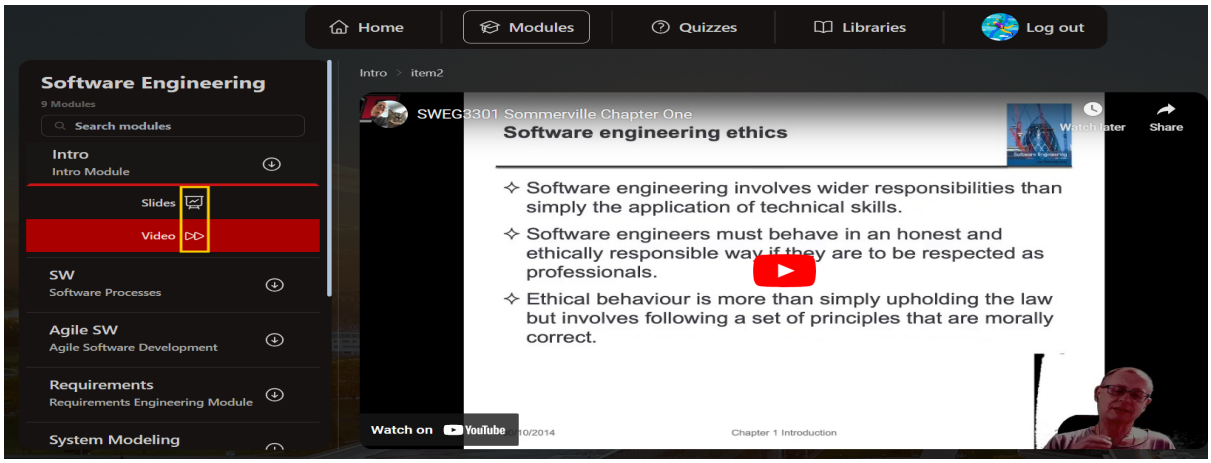


Figure 3.5: Icon of the module items

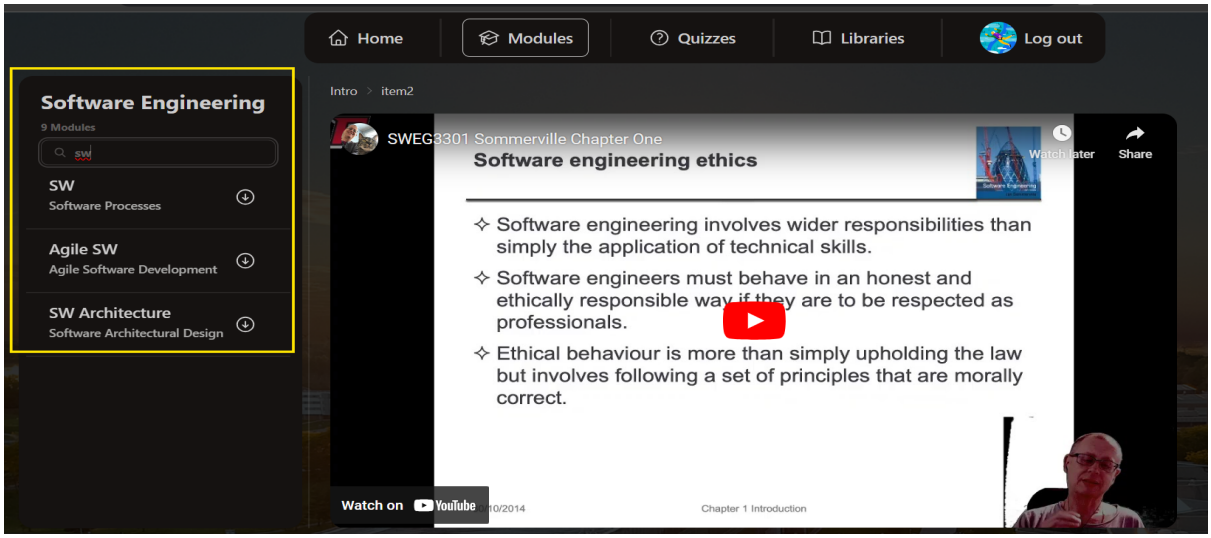


Figure 3.6: Search bar for Modules

3.3 Quizzes

After learning the content from the available modules, the user can test their knowledge by completing a quiz on the *Quizzes* page. Once the user completes a quiz, it gets graded immediately on the leaderboard for the user to check their score. The leaderboard can be accessed by clicking the "Toggle Leaderboard" button seen in Figure 3.7. The leaderboard has a display for the current score, but primarily contains the best scores of each user, sorted in a descending order. As shown in Figure 3.8, the current score is different than the one on the leaderboard for user ***Shubham Amrelia***. If the current score is higher than the *all-time high* score for the user, then the leaderboard score gets updated with the new highest score.

Moreover, there is a pool of 60 questions related to software engineering in our system, and every time a user visits the quizzes page a set of 10 randomly selected questions from the pool is used for the quiz.

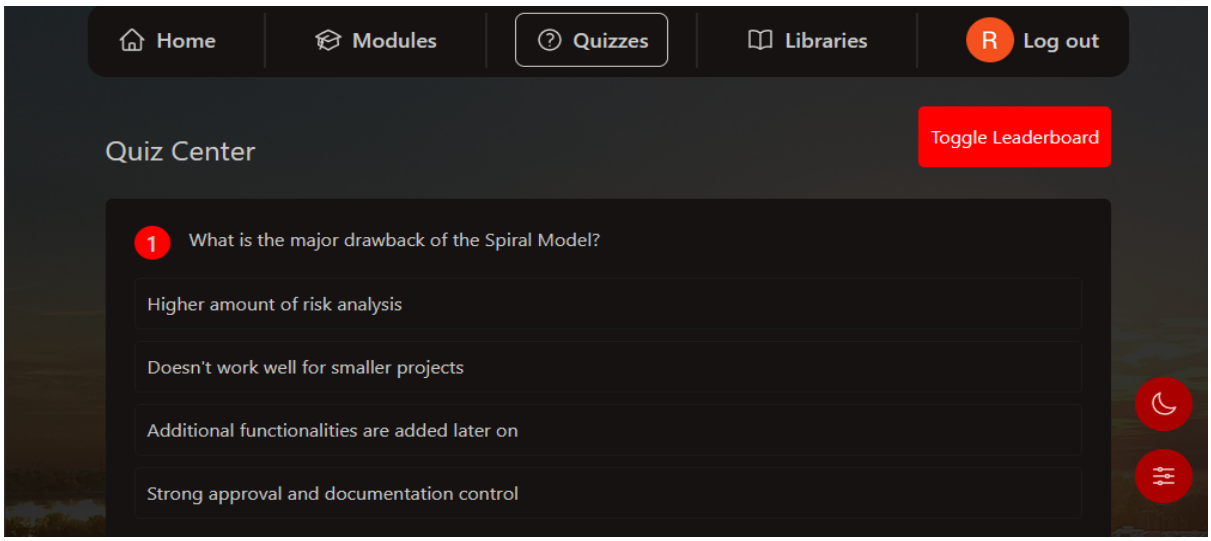


Figure 3.7: The quizzes page.

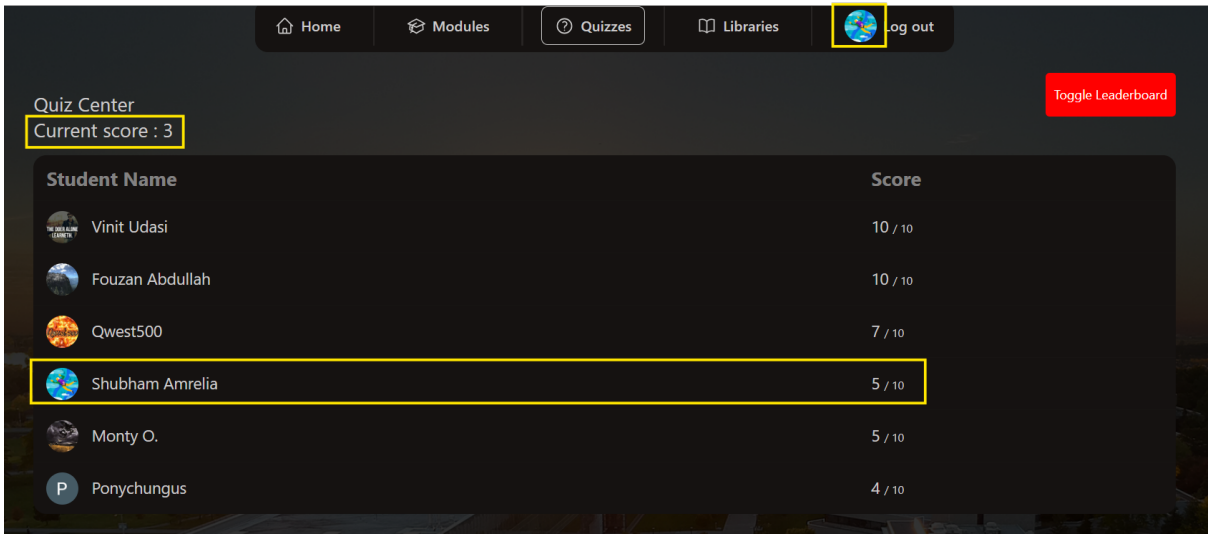


Figure 3.8: The quizzes leaderboard

```
{
    "question": "The agile software development model is built based on _____.",
    "options": [
        "Linear Development",
        "Incremental Development",
        "Iterative Development",
        "Both Incremental and Iterative Development"
    ],
    "answer": 3,
    "selectedOption": 100
},
```

Figure 3.9: JSON structure of quiz's questions

3.4 Libraries

The *libraries* page is made to provide the user with a list of resources that they can use for further learning. For ease of use, these resources are divided into many categories as can be seen in Figure 3.13. While most of the resources are related to the material from the modules, there are also things that can help the user in learning different skills, such as using tools like Git and Docker. There is also a section in the libraries page, *Job Search*, that can be used to practice technical questions frequently asked in interviews for SWE jobs.

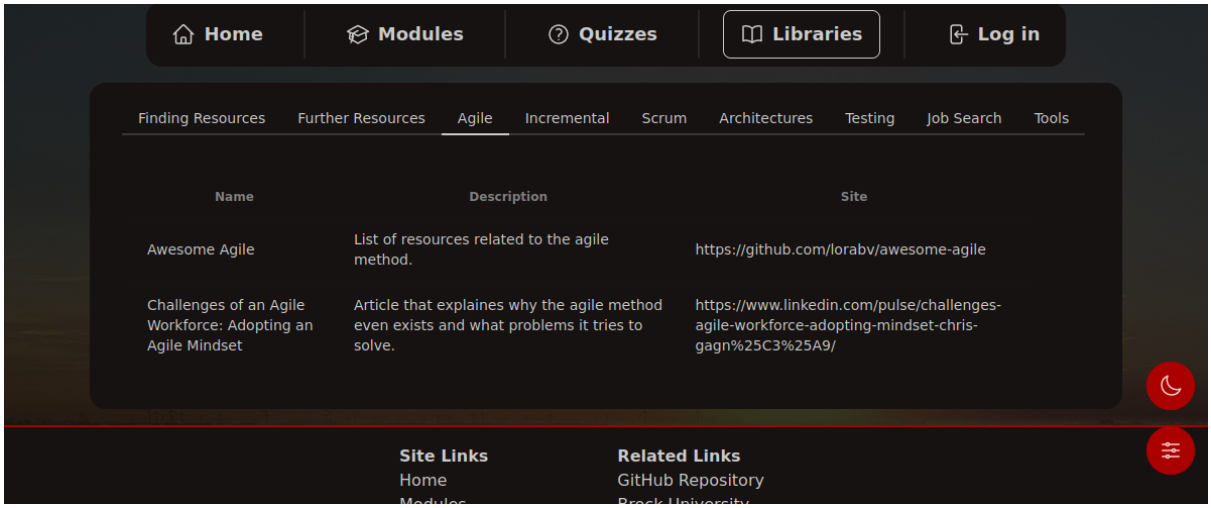


Figure 3.10: The libraries page.

3.5 Accessibility

An accessibility menu was implemented as a modal menu that can be opened by clicking a button that floats at the bottom-right of the screen labelled with an icon as depicted in Figure 3.11. Once opened it includes options for selecting a site theme, resizing the website's text, and displaying a list of keyboard shortcuts. Selecting a theme can be done either by toggling between the default dark and light themes of website by clicking "Toggle Theme" or by selecting a specific theme in the drop-down menu as seen in 3.12.

More on these themes will be discussed in section 3.6. The websites font size is updated to scale with the slider on the menu.

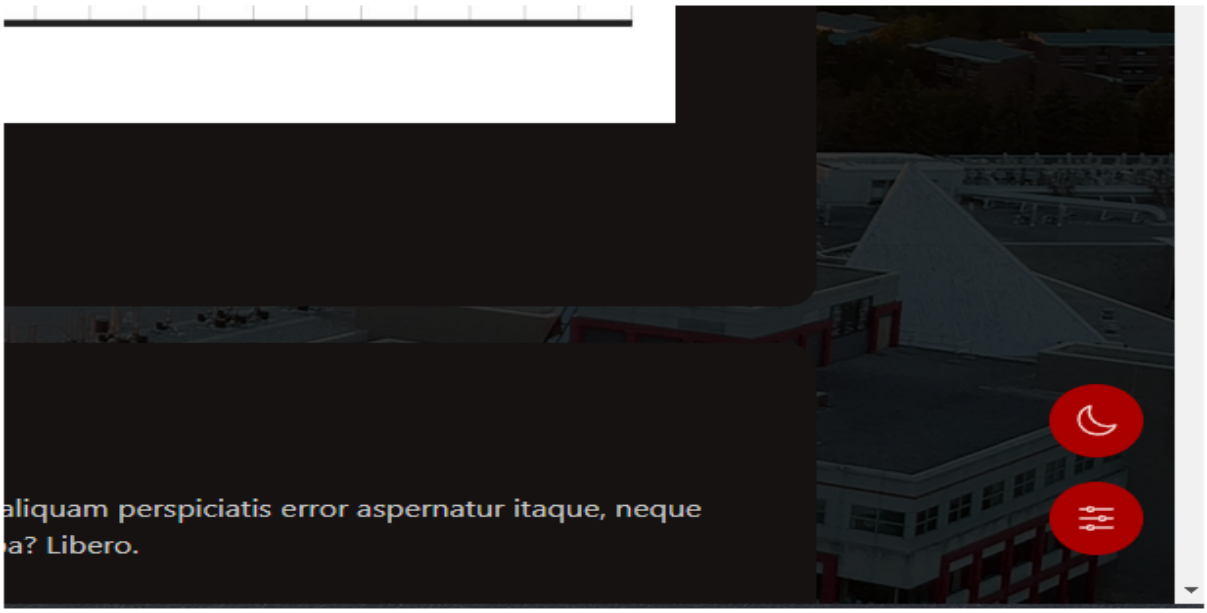


Figure 3.11: The accessibility menu and dark/light mode toggle button.

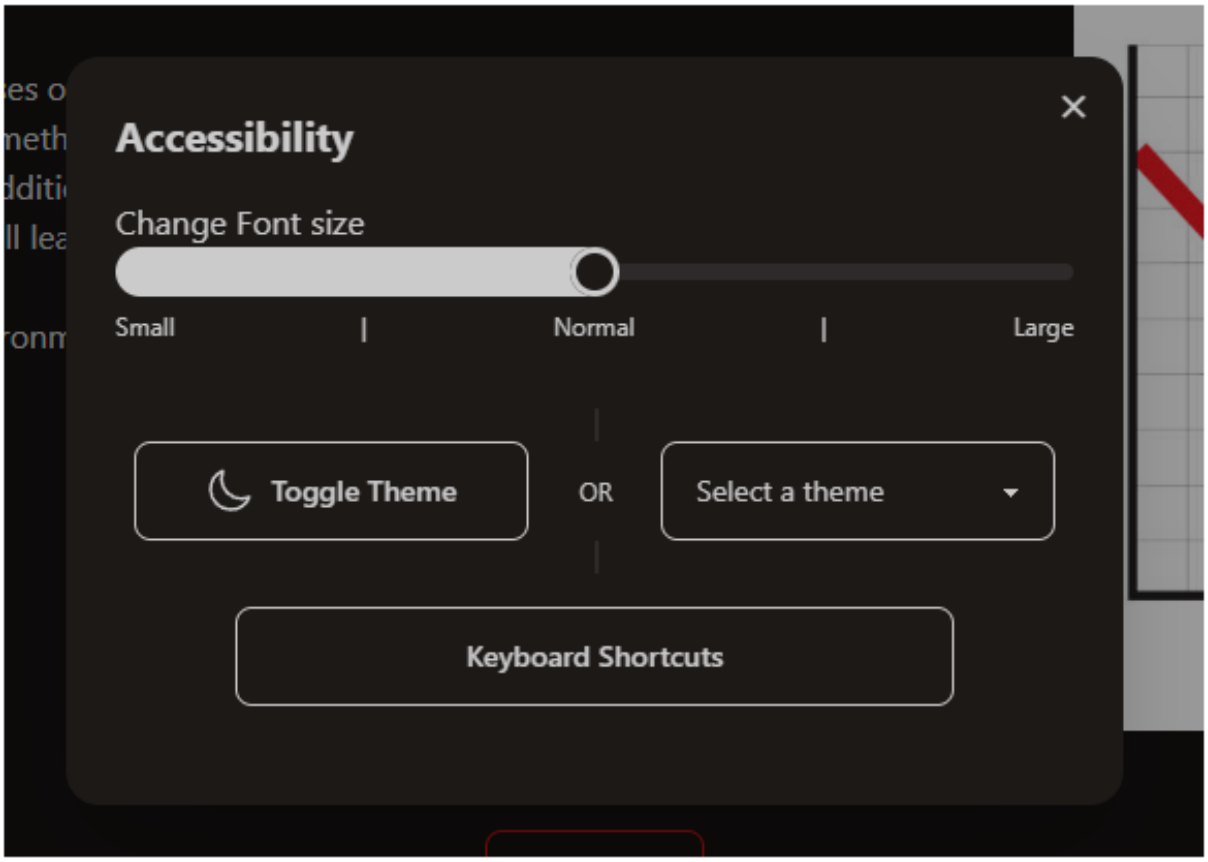


Figure 3.12: The accessibility menu

3.6 Overall Aesthetics

The website received some front-end aesthetic changes that affected all pages. These include an updated image of Brock Campus as a background image to make the site feel less flat, and unique color themes. Themes can either be changed in the accessibility menu, or using the toggle theme button above it (see figure 3.11). The toggle-able button shifts the page to and from the default dark and light themes, while the accessibility menu's theme selector allows changing to additional non-default themes.

3.7 Application Architecture

We primarily followed a microservice based architecture where the different parts of the project were split up into their own entities that communicated via API calls. These microservices will be discussed in the Technology Stack section 3.8.

Additionally, this project can be viewed as a layered architecture where frontend stack elements like TailwindCSS and Svelte make up the user interface. Firebase makes up our backend and represents the business logic and data layers.

3.7.1 Firebase Firestore

One of the services that we used was Firestore. Figure 3.13 shows a summary of all databases created and some fields from one of them.

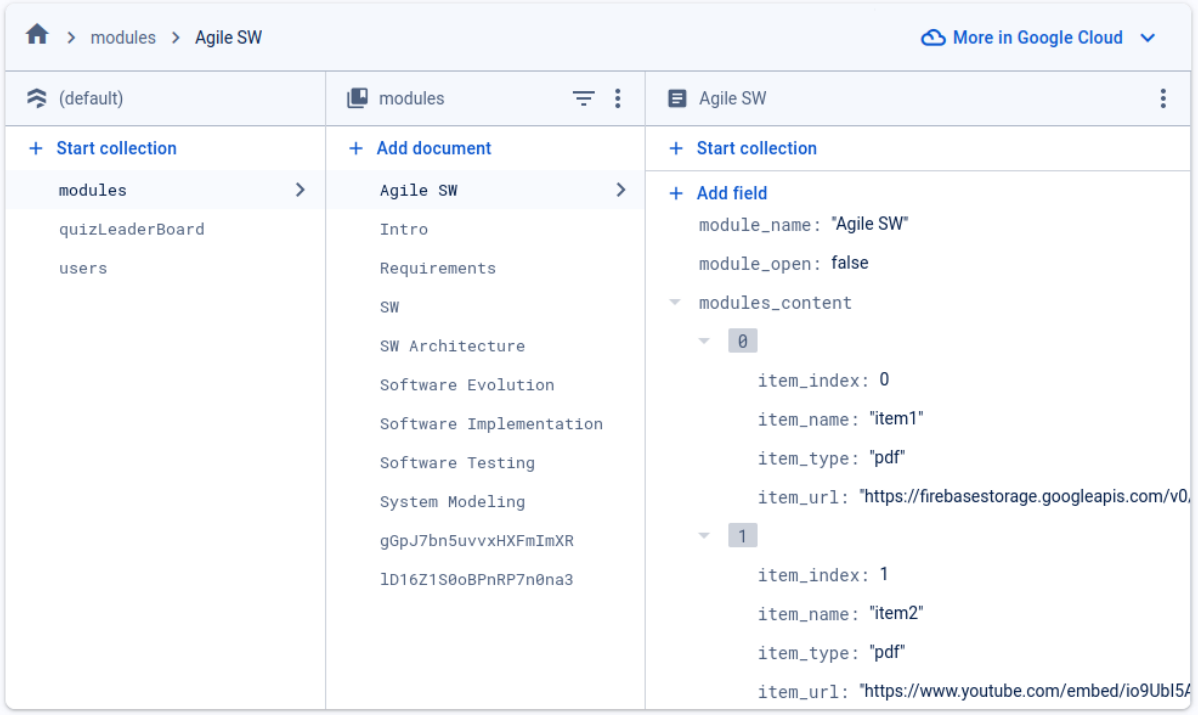


Figure 3.13: Three databases, each with their own modules and each module has its own schema.

3.7.2 Firebase Auth

Logging in was handled by Firebase Auth. Users can select a third party login provider, such as Google, and then automatically be signed in. Examples of users who have logged in through Google can be seen in figure 3.14.

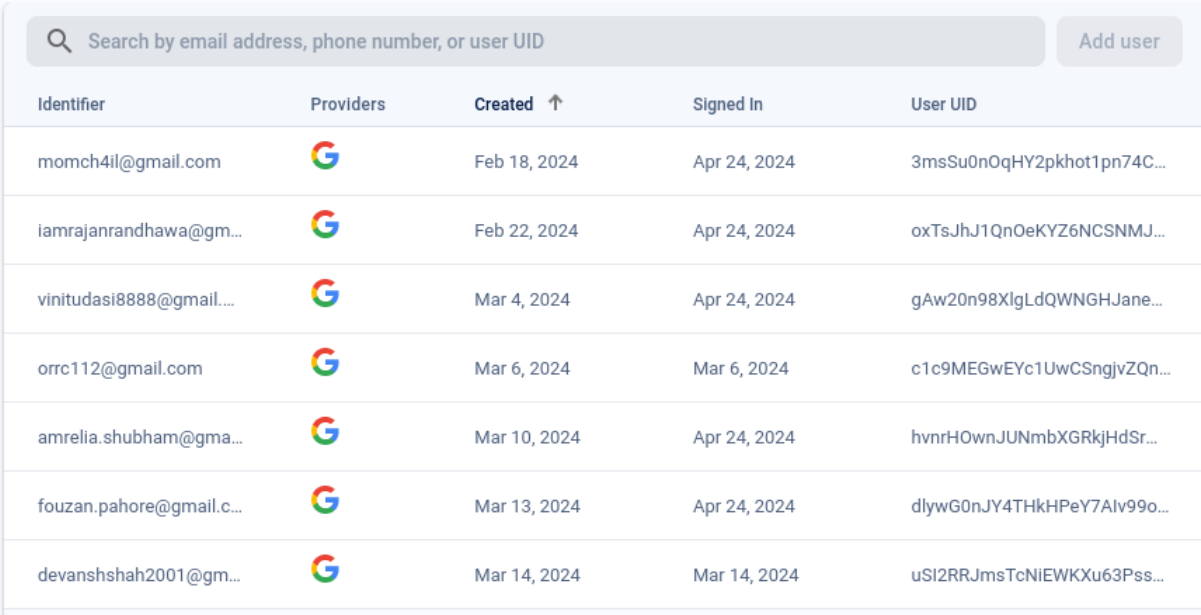


Figure 3.14: List of Google users who have logged into the site.

3.8 Technology Stack

3.8.1 Tailwind

- More Premade CSS styles than you'd ever need and more modularized than CSS.
- CSS requires writing a lot of unnecessary and niche markup styles. This can pile on when you have to use it consistently. This becomes a large time sink.
- Offers the the option for extensions.
- Simplifies frontend further by providing more specific and functional components along.
- Forces developers to use styles that look good together resulting in a more cohesive UX.

3.8.2 SvelteKit

- Svelte is our front-end JavaScript meta framework.
- Compiles javascript and HTML to the DOM.
- Functional webpage components can be naturally created within svelte files.
- Relatively easy to learn.
- Since Svelte is compiled, it runs faster at runtime than other meta frameworks like React and Angular did when the project was started.

- Since this project was started, other frameworks like React and Angular have announced their intent to adopt compilers, showing that Svelte has been very influential in this regard.

3.8.3 Vite

- Vite is the development server for the project. It uses Rollup for bundling.
- Rollup converts source code to elements a browser can read.
- It is easier to use than alternatives such as webpack and highly optimized.

3.8.4 Firebase Firestore

- Backend for storing databases and files.
- Images for the site and PDF files for the modules were stored in Firestore.
- A database was then created which was used to fetch links to these files.
- Firestore simplifies the need for a backend.

3.8.5 Firebase Auth

- Handled user login authentication.
- Has a database containing the history of user logins.
- Allowed for multiple providers such as Google, Facebook, and Github.

3.8.6 Typescript

- Javascript but with explicit typing.
- Helps catch errors very early in the development process.
- Introduces many Object Oriented Programming constructs into Javascript which are familiar to many programmers.

4 Software Engineering Process

4.1 Overview

We decided to go with scrum for our Software Engineering Process. A project like this is bound to have several massive unplanned changes and many more smaller ones, and we believe that in such a situation an agile framework like scrum would be the best choice. Our development consisted of two-week long sprints with meetings every Wednesday at noon. The plan was to create new increments of the project by working on it little-by-little every day. At the end of each sprint we would conduct a sprint retrospective meeting, assign new tasks, and initiate a new sprint. We used Jira to store and assign the backlog of tasks to each person during our sprints.

The following subsection details everything that has occurred throughout the sprints. These notes were mostly taken by Basim. Some were taken by Monty, Vinit, and others.

4.2 Sprint Details

4.2.1 Meeting 1

A problem was that we installed pure Svelte when we should have used Sveltekit for our project. Monty fixed that by installing Sveltekit. Another problem was that our new install didn't have a properly configured *.gitignore* file, which meant that several dozen node modules were all committed to the repository. That also was fixed. Vinit set up routing. Once routing was set up he also created the modules page. Everyone else was given a rundown of Sveltekit to catch them up to speed. Some things to do next were make the navigation bar on the top look nicer and set up a quizzes page.

4.2.2 Meeting 2

Raj set up the firebase console and created the libraries page. Shubham worked on the modules page: setting up the PDF and YouTube viewer. Monty gave the styling on the navigation bar a makeover. Julian set up the home page. Chris added button handlers to the home page. Basim and Fouzan worked on the quizzes page. Vinit made general UI changes to make it more appealing.

To do: Vinit will create a database for all our media, Monty will add login-with-Google, Julian will add a footer and a daisyUI theme, Fouzan will implement reactive quizzes, Basim will add accessibility options, Chris will add a search function, Shubh will make an interface for embedding videos, and Raj will compile resources for the libraries page.

4.2.3 Meeting 3

We faced merge conflicts with the search function and some other mistakes that came from inexperience with using tools like Git, as well as general time management issues. Other than that everyone pretty much did their respective jobs.

This week's jobs were similar as before with some new ones: Vinit making the backend more dynamic and Monty getting the login feature working with the database.

4.2.4 Meeting 4

We were behind schedule so some tasks were moved to the third sprint. To do: Monty will finish up with registering our authentication to the database, Vinit will give the modules page a proper makeover and then finish it off, Chris will add scaling images to the background and implement lazy loading for content, Basim will continue looking at the accessibility menu, Raj will help with login and authentication, and Fouzan will continue working on the quizzes page.

4.2.5 Meeting 5

During this reporting period, the team remained dedicated to ongoing tasks initiated in the previous sprint. The focus was primarily on advancing existing tasks. Below is a brief summary of individual contributions:
Monty: Focused efforts on finalizing the authentication registration within the database.

Chris: Continued the implementation of scaling images for background usage and integrated lazy loading functionality for images.
Vinit: Maintained focus on database creation and initiated work on developing a dynamic view to facilitate module interaction.
Basim: Continued development work on the accessibility menu, ensuring its effectiveness and usability.
Raj: Provided assistance in login and authentication-related tasks, supporting the team's progress in this area.
Fouzan: Further improved and refined the quiz page, incorporating additional enhancements for user engagement.

4.2.6 Meeting 6

Due to other obligations like tests and assignments taking up the team's time and focus, Sprint 3 ended with little progress. As a result, project work did not progress much during this time.

4.2.7 Meeting 7

Some issues faced this time around were: Raj had a weird bug with the login button, Monty had to figure out backend stuff which is always a pain, Julian needs more time, and Basim had difficulty making the accessibility menu function on other pages. The accessibility menu in general was hard to deal with since it still had some residual CSS that hadn't been converted to tailwind or daisyUI yet.

On the bright side Chris added some background images, a logo, content for the home page, and changed some colors.

4.2.8 Meeting 8

Monty added a dark-light theme switcher for the accessibility menu, changed CSS to tailwind mainly on the modules page, added inline documentation, refactored the README files, added feedback while authentication is loading, and fixed the modules page to make it look better after the tailwind move.

Basim worked on keyboard shortcuts for the accessibility menu, Shubh gathered question and worked on the logic for the quizzes page, Raj worked on getting other login providers, and Julian also helped with the dark-light theme.

Chris had a merge issue and possibly has to redo some things in his branch before merging it with main.

4.2.9 Meeting 9

Monty finished the theme switcher, switched more CSS to tailwind, added a slider for resizing text, cleaned up the modules page to make it look nicer, and some other general front end stuff on pretty much every page.

Raj worked on adding tabs for the libraries page. Shubham added visual cues on the

modules page. Chris fixed his branch and readded the background images and a new logo to the home page using Tailwind instead of CSS.

Problems faced: the quizzes page was pulling more documents than it was supposed to, quiz questions were stored locally so you could easily cheat, the footer was breaking on certain pages, some merges had to be reverted because they solely contained pure CSS and no tailwind, and inconsistent indentation also lead to some issues.

4.2.10 Meeting 10

Tests were done on the current main build to see how slides and videos were loaded by the modules page. Only one slide URL is loaded at any given moment and a new one is loaded when it is selected from the drop-down menu, which is good. YouTube videos are also only loaded when they are selected, even though multiple versions of the video are loaded at the same time—embedded, original, href, etc.

4.2.11 Meeting 11

Focused on designing the logo for the project. Acknowledged that little progress was made overall due to everyone being occupied with finals and last week's assignments.

4.3 Sprint Timeline

Dates	**Task**
Jan 9 - Jan 16	Created **Project Proposal**
Jan 17 - Jan 23	Generated user stories and created **Product and Sprint Backlog**
Jan 23 - Jan 25	Finalize **Product and Sprint Backlog** due **Jan 26**; Meet with TA
Jan 27	Begin **Sprint 1**
Feb 7	Sprint 1 Retrospective Meeting and Create **Progress Report 1**;Begin Sprint 2
Feb 21	Sprint 2 Retrospective Meeting and Submit **Progress Report 1**;Begin Sprint 3; Meet with TA
March 6	Sprint 3 Retrospective Meeting and Create **Progress Report 2**; Begin Sprint 4
March 20	Sprint 4 Retrospective Meeting and Submit **Progress Report 2**; Meet with TA
April 10 - April 28	Web App Deployment and Submission of **Final Report**; Final Presentation

Table 1: Project Timeline

4.4 Testing

1. Self testing

    - Done by developers to make sure that their code runs as intended
    - Developers were asked to think like users while self-testing
    - Resulted in better code quality
    - Highly effective as each developer knows their weak points better

2. Peer Testing

    - Peer testing was implemented within the developers to test each other's code
    - This approach helped in identifying defects early in the development
    - Each developer's pull requests were scrutinized before merging it to the main branch
    - Product owner was asked to test a feature once it gets deployed, to make sure it meets the requirements

3. User testing

    - We reached out to friends and colleagues to mimic user centered testing
    - This way, we had 4 users throughout the project as shown via quiz leaderboard in fig 4.3
    - To keep the tests fair, the users did not know any technical details about the system
    - Users also helped with selecting themes for the website, in the accessibility menu
    - Found and resolved issues in quiz's leaderboard via user feedback

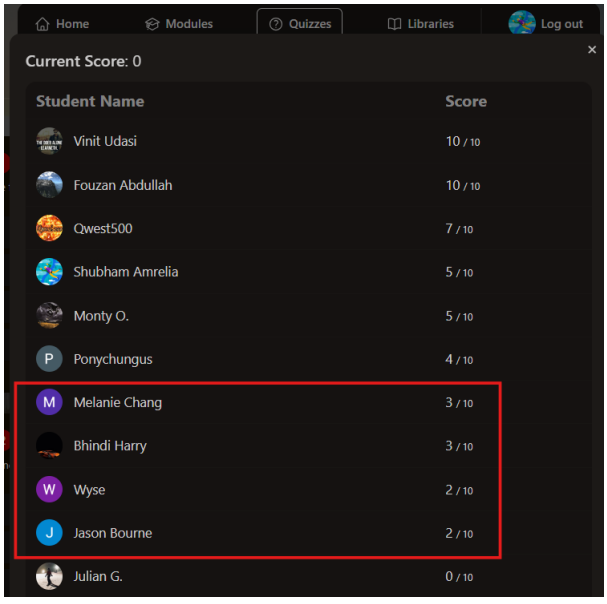


Figure 4.1: Users of our system

4. Unit Testing

- Throughout development, we did component testing and end to end testing before merging it to the main branch.
- Once all the components were tested and we had developed a Minimum Viable Product, we used ***Playwright*** to do the unit tests throughout the Application.
- Some unit tests we wrote included testing **Cross-Browser Compatibility, UI Components, Ease of Use of Buttons, Authentication**.
- One problem we discovered using these tests is that user couldn't sign in sometimes, a snapshot is attached at below.

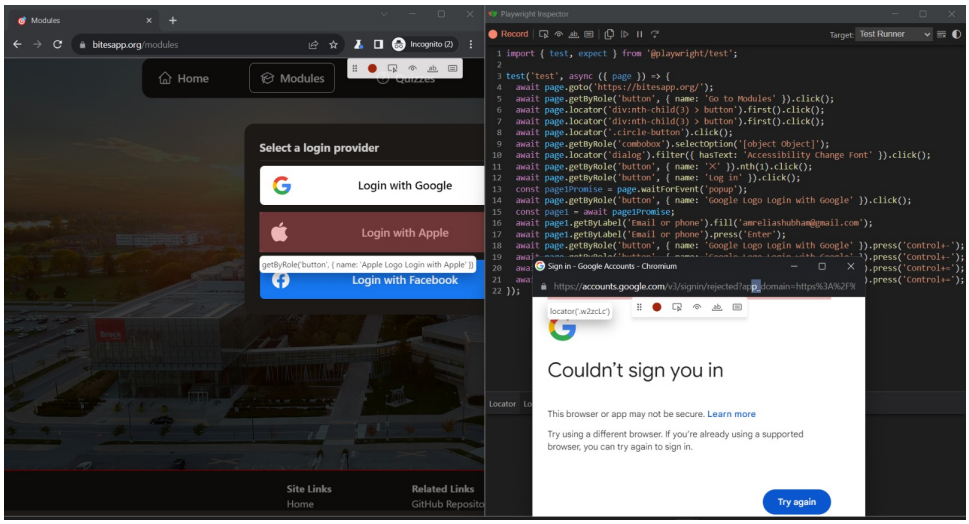


Figure 4.2: Sign-In error

```
ASUS@LAPTOP-339PJHKB MINGW64 ~/Desktop/4p02/4p02_new/bites-app (testing)
$ npx playwright test

Running 24 tests using 6 workers
  Slow test file: [firefox] › togLeadBoard.spec.ts (28.4s)
  Slow test file: [firefox] › quizSubmit.spec.ts (28.0s)
  Slow test file: [firefox] › openModule.spec.ts (27.1s)
  Slow test file: [firefox] › goToMod.spec.ts (24.0s)
  Slow test file: [chromium] › togLeadBoard.spec.ts (22.4s)
  Consider splitting slow test files to speed up parallel execution
  24 passed (1.5m)

To open last HTML report run:

  npx playwright show-report


ASUS@LAPTOP-339PJHKB MINGW64 ~/Desktop/4p02/4p02_new/bites-app (testing)
$
```

Figure 4.3: Test cases

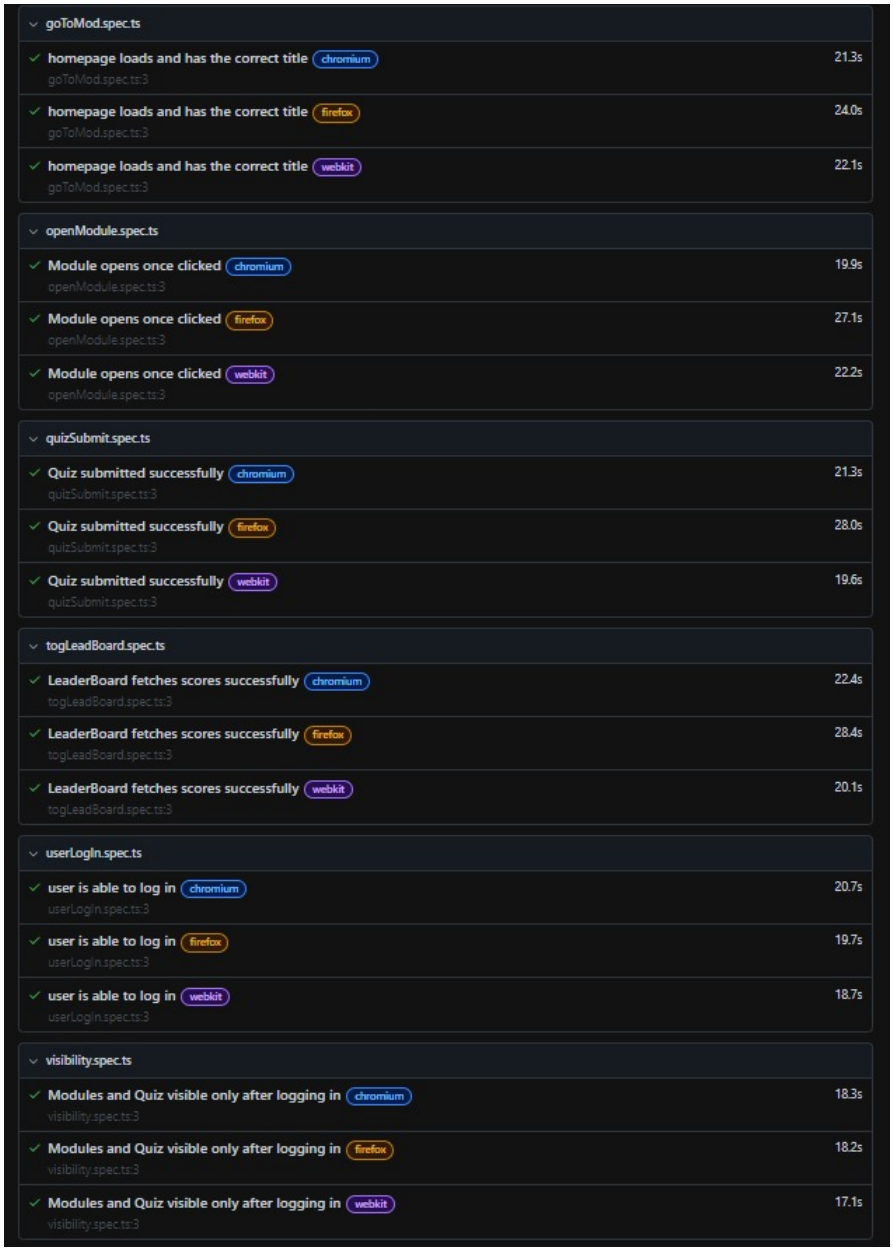


Figure 4.4: Test Case Logs

5 Challenges

Just like all the projects ever made, our BITES projects also came with its own set of challenges. This section provides an insight into these challenges while also providing information on how they were handled by the team.

- **Implementation:** The first problem we faced was implementing a pure Svelte framework for this project. Implementing pure Svelte is an efficient way for web development if the website being made is single paged i.e. it is good for only front-end UI development. SvelteKit, in contrast, offers the capability to do full-stack development. For instance, it has built-in, file based routing as compared to pure Svelte that requires additional libraries. It also has native support for server side

rendering (SSR)! So, to solve this issue, pure Svelte was replaced by SvelteKit for this project.

- **Accidental Pushes/Commits:** Next, a the member accidentally pushed NodeJS libraries to the main branch, because a '*.gitignore*' file was missing from the repository. Since we did not have this file, git committed *all* files into our repository. The majority of these files were Node libraries and thus over a million lines of "code" were added to the repository that day. The *.gitignore* file was updated on the repo after reverting the changes to the branch. The branch had to be reverted with a hard reset and forced push to keep this from influencing github's commit history.

- **Code Review:** One major challenge our group faced was ensuring code quality and consistency while implementing effective collaboration techniques such as pair programming.

- **Documentation:** Finding documentation on how to properly secure Firebase was difficult in this project. Firebase has had many different versions and has a huge community. As a result of this, advice and standards for security can vary significantly making it difficult for engineers with limited experience working on backend systems to learn. To solve this issue, learning about security from a single source was emphasized.

- **Personal Challenges:** One of the main challenges for the team was time management. Not everyone was able to commit a fixed number of hours to the project but everyone made up for it by catching up and getting upto speed with what had been discussed in previous meetings. Additionally, there was some communication gap which hindered development for some time but there was a learning curve of that and throughout the course, it became more efficient.

6 Activity

6.1 Contributions

Each quantifiable contribution for every member.

6.1.1 Basim

- Scheduled and held weekly Scrum meetings
- Made notes for weekly meetings based on the assigned tasks, finished tasks, and challenges faced
- Kept track of project's development timeline, achievements, challenges, and future plans
- Helped in designing the initial accessibility menu
- Added buttons for changing font size and keyboard shortcuts

6.1.2 Fouzan

- Prioritized tasks relating to setting up initial dependencies, creating the frontend, and documentation based on user needs and requirements in the product backlog
- Collaborated with team members to gather requirements for different features as per stakeholder feedback and needs
- Ensured that the documentation provided such as the README, meeting notes, and instructions were ample enough to support the development process and assisted in the setup of the project
- Also assisted in the development of a basic skeleton layout for the quizzes page
- Helped take meeting notes

6.1.3 Monty

- Designed the project stack and helped other members get the project running on their machines
- Polished and finalized client-side components and pages, including modules, quizzes, accessibility, and website background
- Updated these components to use TailwindCSS where applicable
- Created the front-end for login components, theme selector and toggles, navbar, and administrator view for adding new modules
- Implemented authentication to the project, alongside related functions on the back-end, like registering user metadata in the database
- Added other UI improvements for the accessibility menu
- Wrote readmes and comments as documentation for the project
- Worked on making the website mobile-friendly
- Helped work on the reports
- Performed code review, providing feedback like requesting changes or approval
- Helped manage repository

6.1.4 Vinit

- Set up routing so the users can easily navigate through all the pages
- Worked with Monty and Raj to integrate Firebase, allowing for seamlessly fetching and displaying of media
- Made the app look and feel better by tweaking UI for aesthetics and usability
- Made improvements to the backend to add more interactive features and make the app run smoother
- Wrote the entire first progress report with Monty
- Gave the admin view a makeover, adding new tools and refining existing ones
- Contributed to building out the Modules page, adding code, features, and functionality
- Helped create the quizzes page, providing ideas and support throughout the process
- Provided code reviews to make sure everything was consistent, offering feedback

and suggestions to keep the project on track

6.1.5 Shubh

- Worked mostly on the logic and technicalities of the *Modules* and the *Quizzes* page
- Gathered the PDFs and videos for the *Modules* page
- Set up PDF and YouTube viewer for *Modules* page
- Made the JSON structure of the list of questions (ref Fig 3.9)
- Made the *Quizzes* page, and the leaderboard logic with Vinit as pair programmer
- Drafted the final presentation, later reviewed by the team
- Helped in writing the final report
- Participated in code review

6.1.6 Julian

- Designed the basic layout of the landing page
- Created the footer and made it persist across each page
- Designed the website logo and screen-recorded the module demo
- Assisted in making the website mobile-friendly
- Polished some client-side components such as the landing and quizzes page
- Translated some CSS to Tailwind + DaisyUI to match the program stack

6.1.7 Chris

- Attended meeting to organize Svelte Implementation and project setup
- Integrated button OnClick Handlers to Landing Page.
- Designed search feature for Modules page to filter Module Results
- Implemented scaling background images and lazy loading
- Updated CSS to Tailwind + DaisyUI to match the project
- Conducted thorough testing of modules page, ensuring video and slide content is only loaded when a specific module is selected

6.1.8 Raj

- Created the libraries page
- Gathered supplementary resources for libraries and modules
- Wrote the entire second progress report
- Polished login function frontend
- Deployed the site to Cloudflare pages and bought a domain for it
- Participated in code review
- Helped manage Github repository by deleting old branches

We all collectively wrote the final report.

6.2 Github Log

6.2.1 Github Contribution Plots

Figures 6.1 through 6.8 are plots that represent summaries of the total commits each person has contributed.

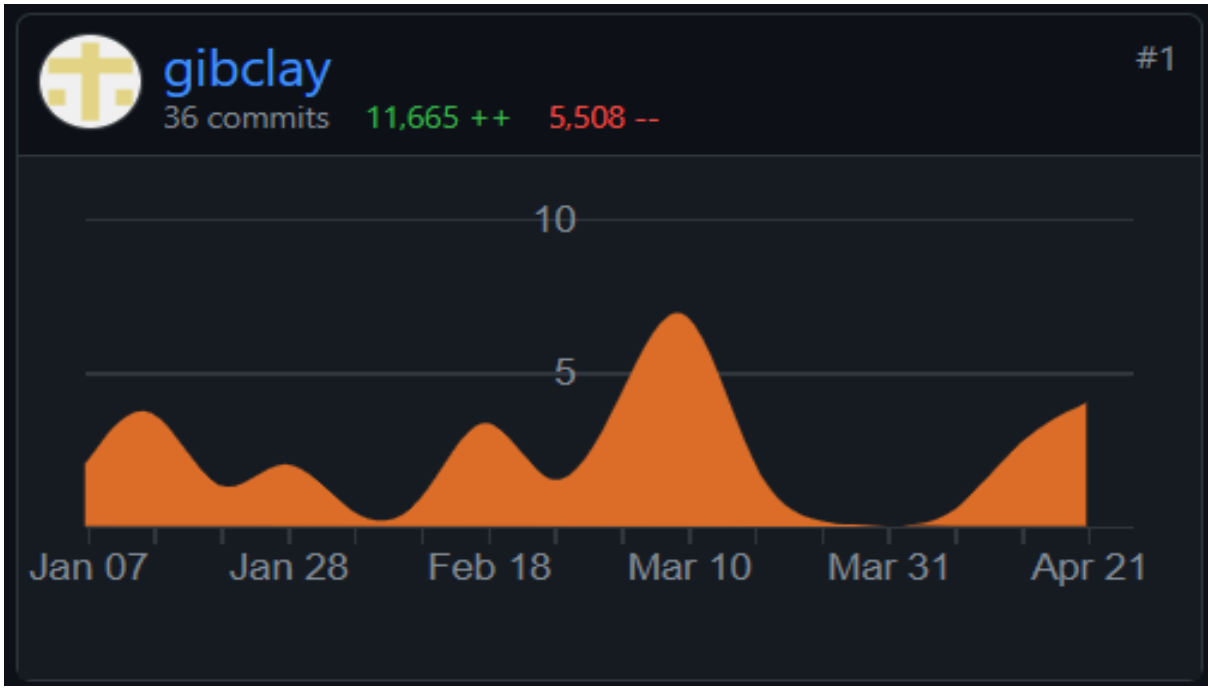


Figure 6.1: Monty's commits summary.

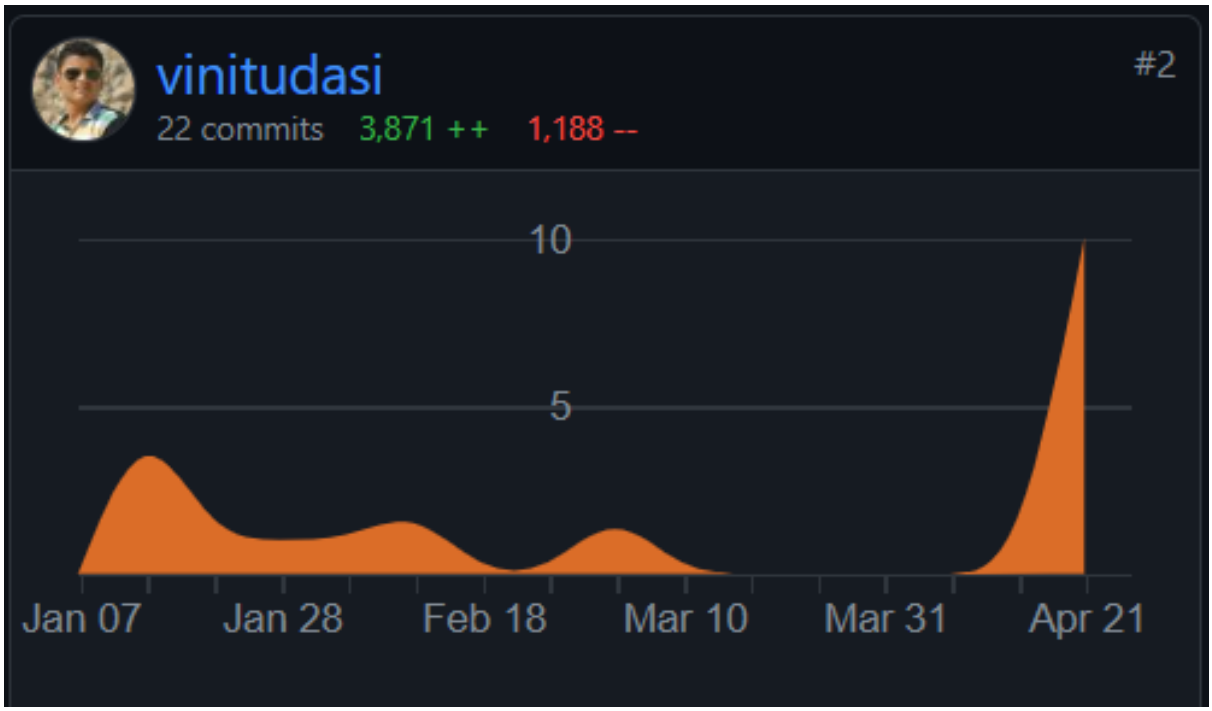


Figure 6.2: Vinit's commits summary.

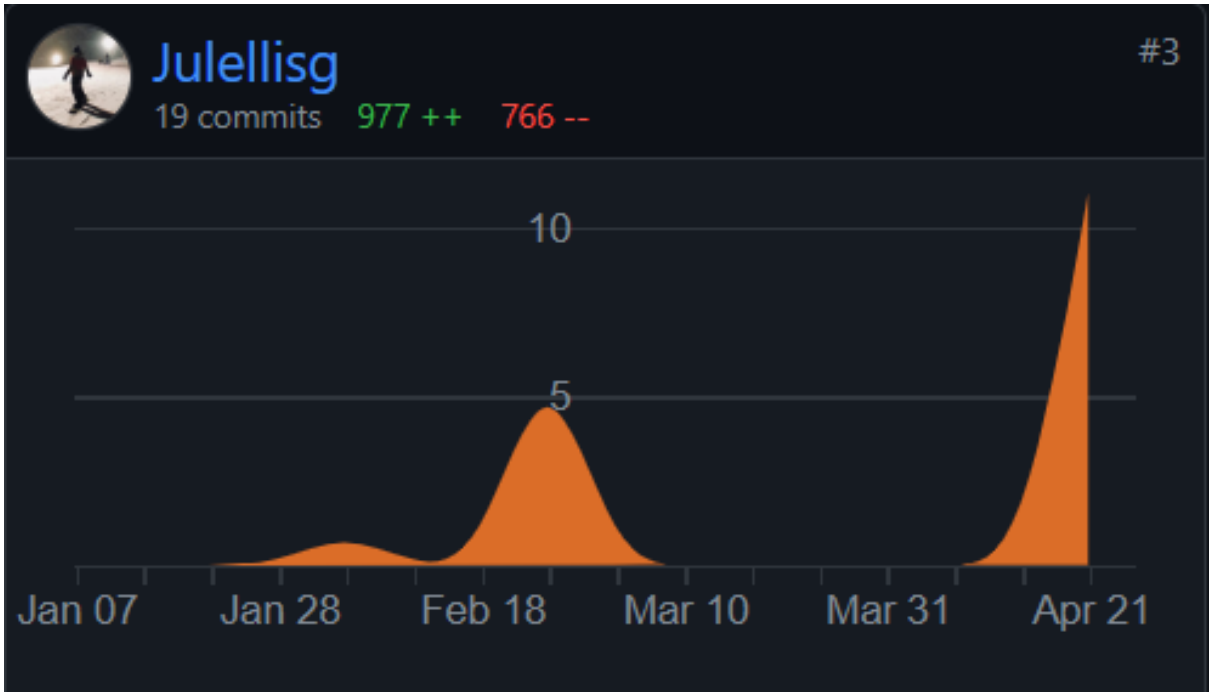


Figure 6.3: Fouzan's commits summary.

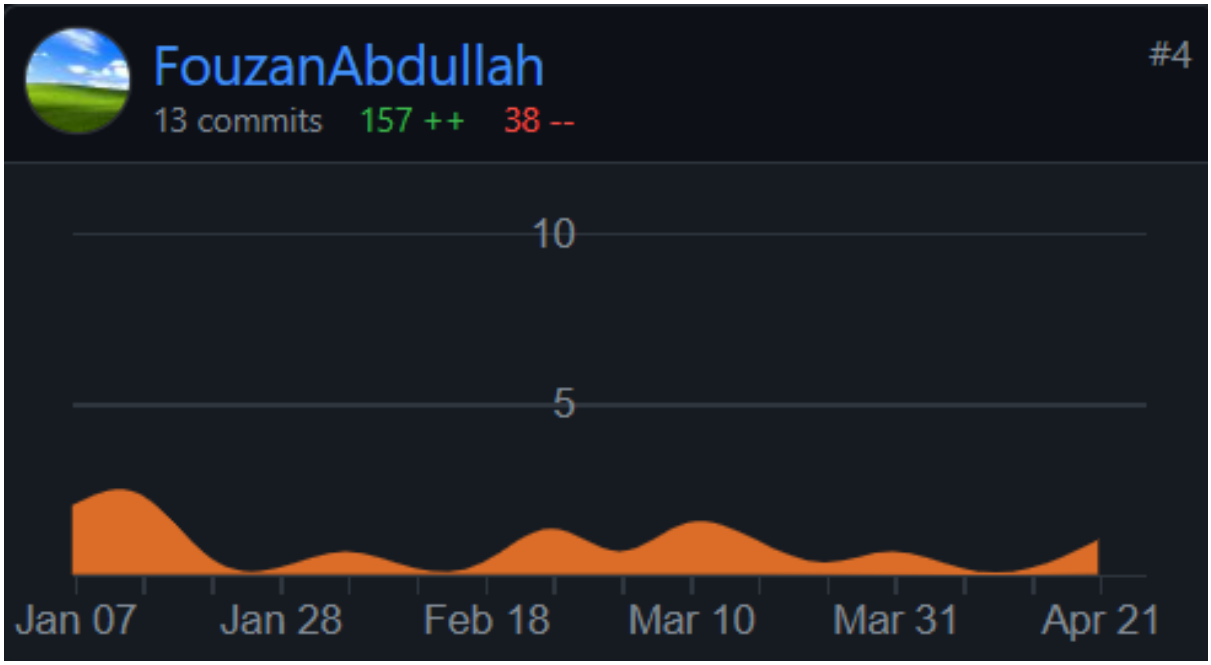


Figure 6.4: Raj's commits summary.

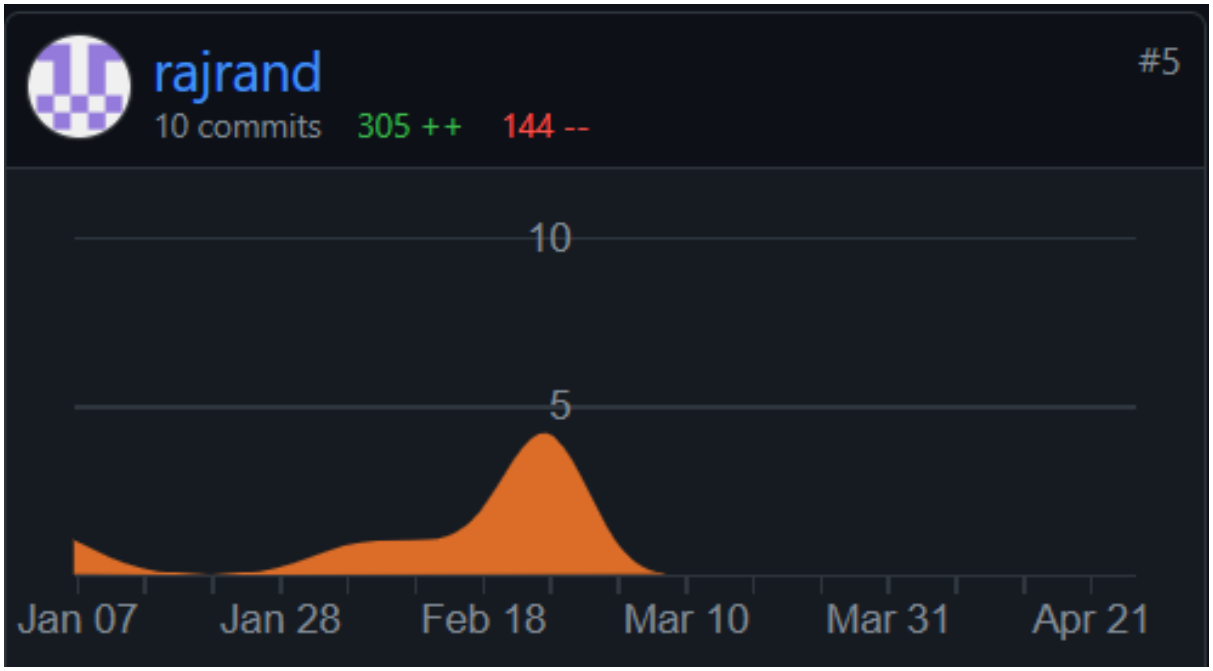


Figure 6.5: Julian's commits summary.

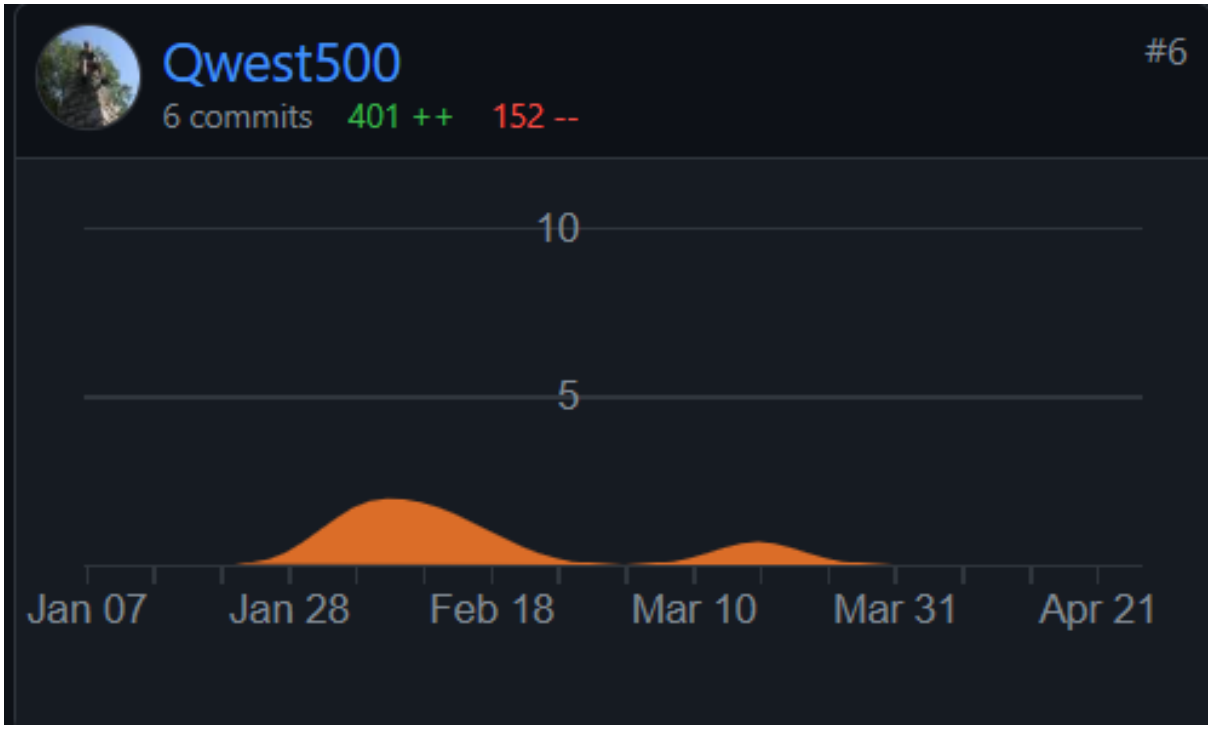


Figure 6.6: Chris' commits summary.

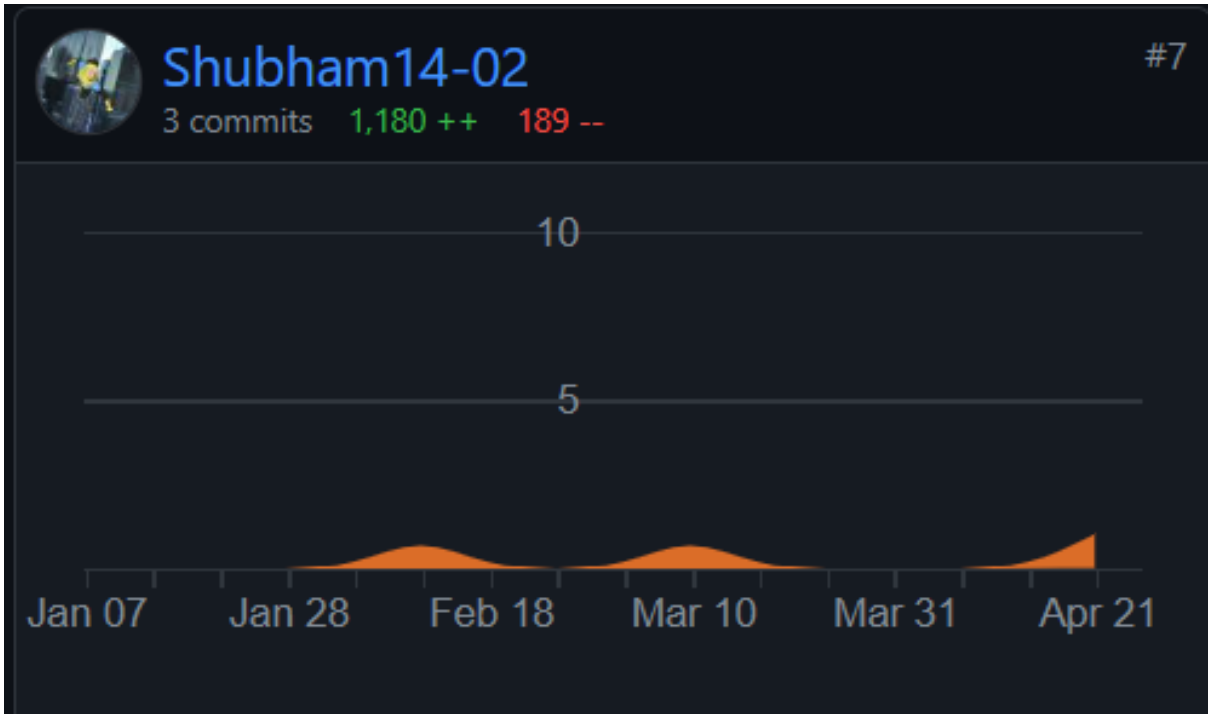


Figure 6.7: Shubh's commits summary.

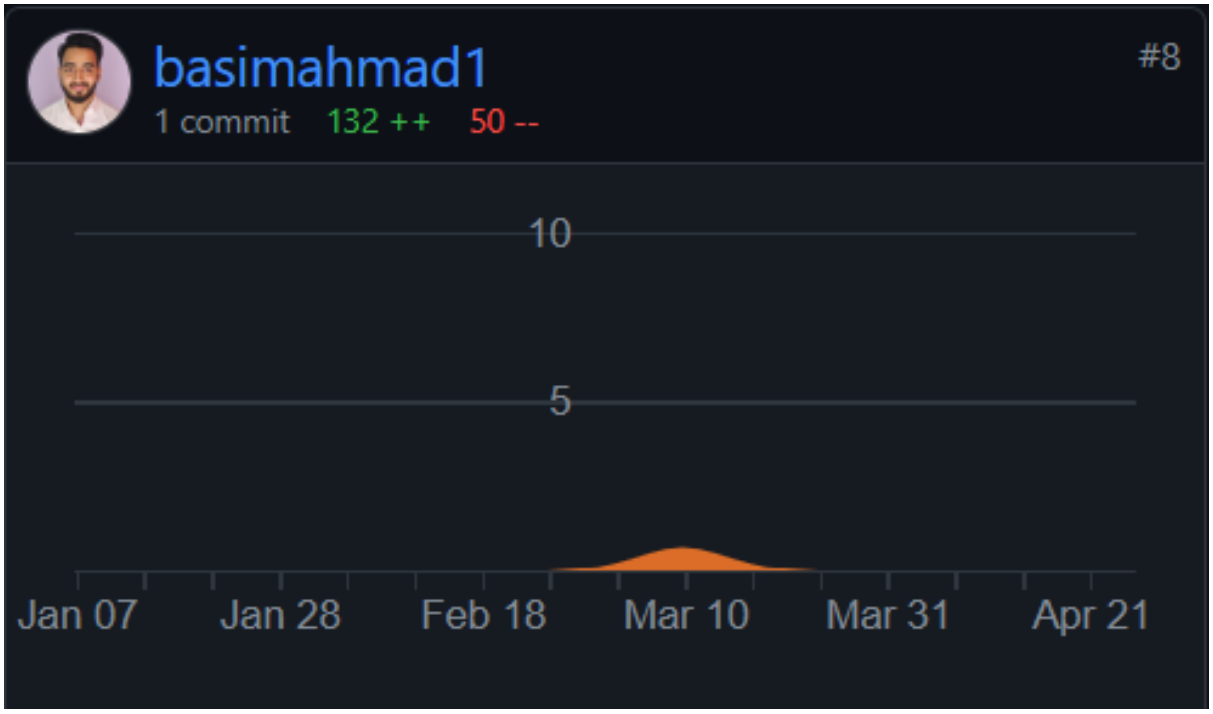


Figure 6.8: Basim's commits summary.

6.2.2 Performance Reports

The next figures summarize how much of our backlog we were able to complete.

Figure 6.9 is the sprint commitment which shows the average tasks from our backlog

completed per sprint.

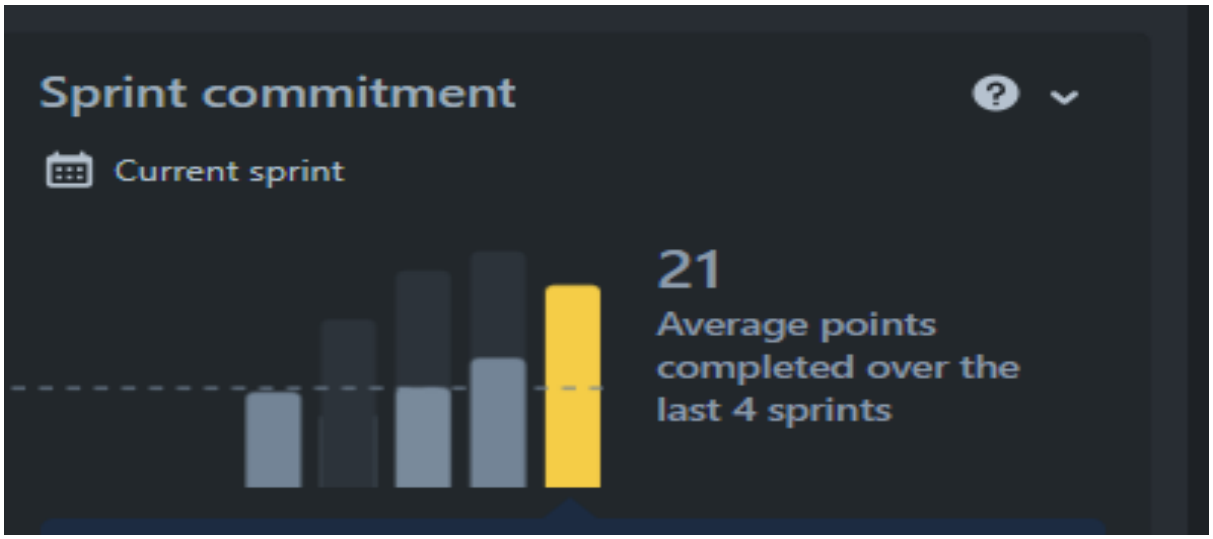


Figure 6.9: Sprint commitment chart.

The burnout charts, which represent completion of tasks over time compared with expected completion, for sprints 3 and 4 can be seen in figures 6.10 and 6.11, respectively.

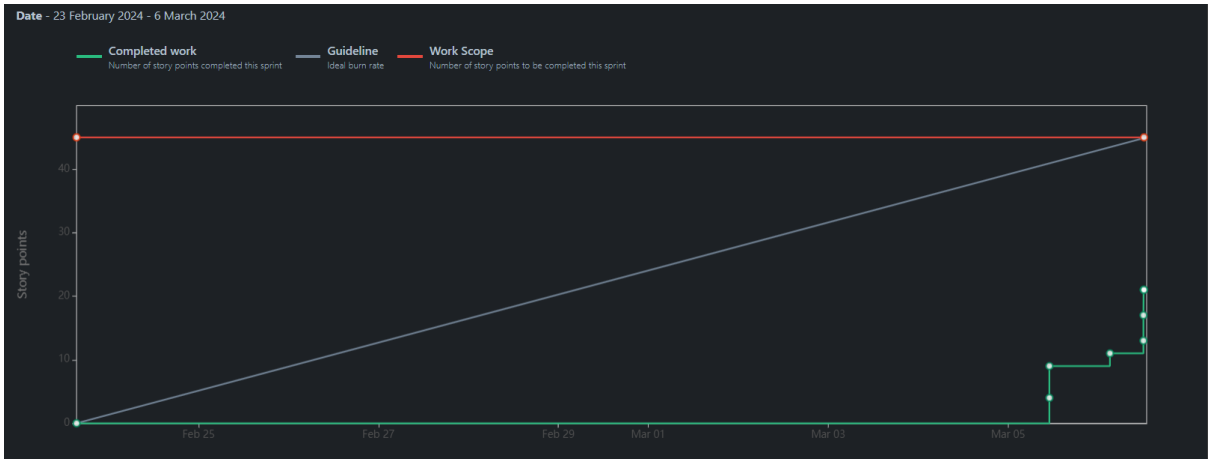


Figure 6.10: The Sprint 3 burnup chart.

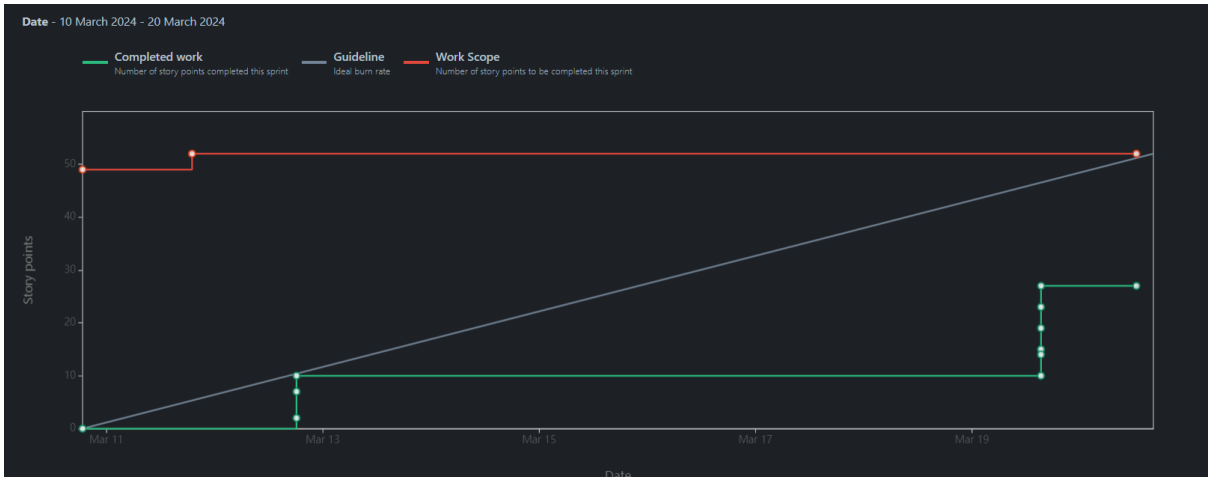


Figure 6.11: The Sprint 4 burnup chart.

The velocity chart for sprint 4, which is basically the same thing but represented as a Bar Graph, can be seen in 6.12.

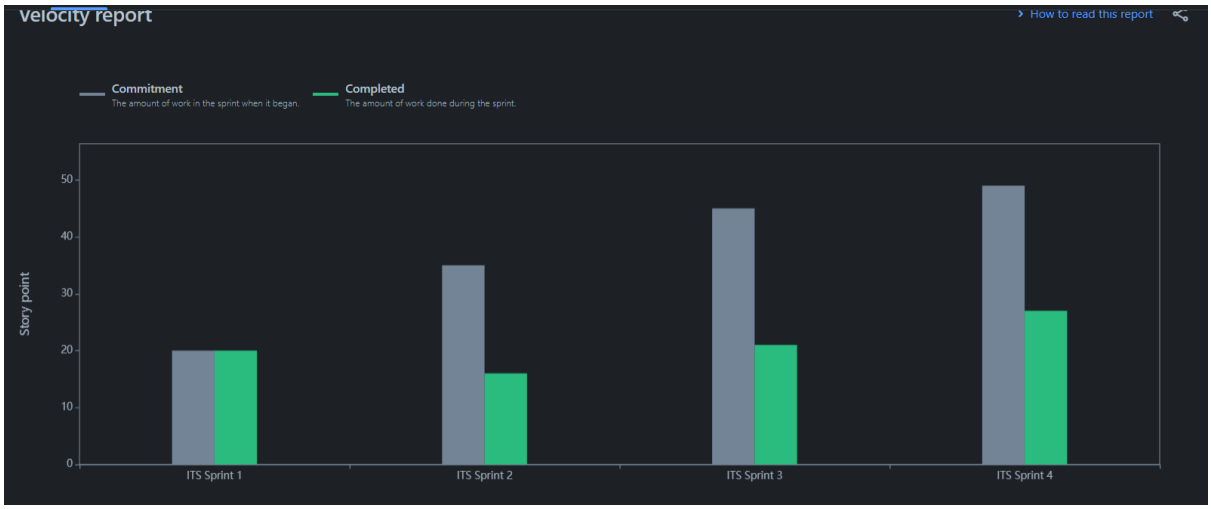


Figure 6.12: The Sprint 4 velocity chart.

The cumulative flow diagram, which shows progress done compared with to-do over time, can be seen by referring to figure 6.13.

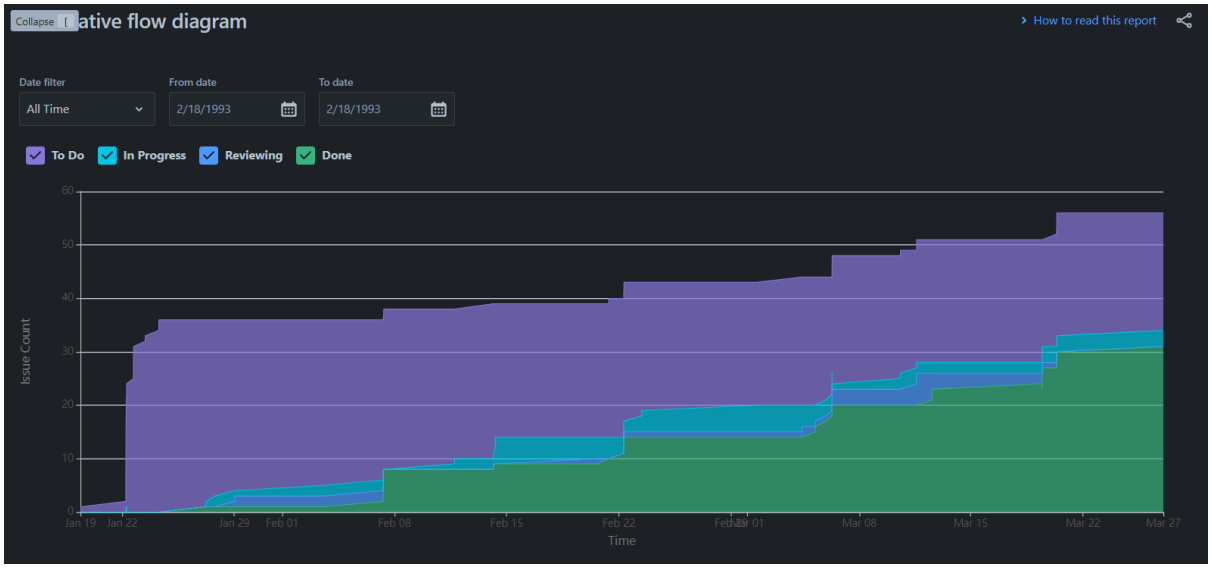


Figure 6.13: The cumulative flow diagram.

7 Github Repository for the Project

https://github.com/SWE-2024/COSC-4P02

8 Team

1. **Fouzan Abdullah**
   Student Number: 6840797
   Role: Product Owner

2. **Basim Ahmed**
   Student Number: 7022494
   Role: Developer and Scrum Master

3. **Vinit Udasi**
   Student Number: 6847800
   Role: Developer

4. **Shubham Amrelia**
   Student Number: 6846877
   Role: Developer

5. **Julian Ellis Geronimo**
   Student Number: 6756597
   Role: Developer

6. **Monty Oshinov**
   Student Number: 6759286
   Role: Developer

7. **Rajan Randhawa**
   Student Number: 6996441
   Role: Developer

8. **Chris Orr**
   Student Number: 6755383
   Role: Developer