Verification and Validation

Book Bazar

February 3, 2022

Group 6

Caleb Mech	mechc2
David Thompson	thompd10
Matthew Williams	willim36
Ahmed Al Koasmh	alkoasma
Harsh Mahajan	mahajanh

Contents

List of Figures

List of Tables

1 Table of Revisions

Version	Date(dd.mm.yyy)	Author(s)	Description
0	21.01.2022	Harsh Mahajan	First draft assembled.
1	25.01.2022	Caleb Mech	Tests added in Section 7 & 8
2	26.01.2022	Matthew Williams	Tests added in Section 7 & 8
3	31.01.2022	Ahmed Al Koasmh	Tests added in Section 8
4	03.02.2022	All team members	Worked on feedback from TA. Section 6.1 Testing Tools added and elaborated on load testing in Section 6.2.

Table 1: Table of Revisions

2 Purpose

The motivation for this project is to create an application that allows McMaster students to conveniently buy and sell textbooks in a secure marketplace. Currently, students use online websites like Facebook (`https://www.facebook.com/groups/macusedtexts`) or the campus store's buyback program (`https://campusstore.mcmaster.ca/information/guaranteed-buyback.html`, suspended because of COVID) to sell their books. These current solutions are either disorganized or take a cut of the transaction. Book Bazaar aims to create an organized marketplace that allows users to sell books for a better price than at the campus store.

The purpose of this Verification and Validation (V&V) is to provide a complete overview of the process undertaken in-order to ensure that all the requirements have been covered and tested.
NOTE: Book Bazar's website where all the books to be sold are listed, is referred to as *marketplace* in certain parts of this document.

3 Scope

The scope for the project covers the following application features:

1. An authentication and authorization platform that verifies McMaster students and allows them to log into and use the application.
2. A marketplace system that uses information from the campus store and allows users to:
   - Post about and categorize books that they would like to sell.
   - Search for books that they would like to purchase.
3. A system that allows buyers and sellers to privately contact each other, agree on a price, and choose where they would like to meet.
4. The application will be accessible from any device with a modern browser (phone, tablet, desktop, etc.) to provide the best experience.

4 Background

University students often like to buy/sell used textbooks. Sometimes it's for the notes written in the margins, but often it's because new textbooks are quite expensive.
Currently at McMaster University, students use online websites like Facebook (`https://www.facebook.com/groups/macusedtexts`) or the campus store's buyback program (`https://campusstore.mcmaster.ca/information/guaranteed-buyback.html`, suspended because of COVID) to sell their books. These current solutions are either disorganized or take a cut

of the transaction. Book Bazaar aims to create an organized marketplace that allows users to sell books for a better price than at the campus store.
The project, Book Bazar is a website that addresses these issues. McMaster community members may buy/sell textbooks with ease using Book Bazar.

5 Roadmap

Using the milestones, listed in Section 6 of the Development Process document, the team aims to have Book Bazar completed by the week of March 6, a week prior to the team's final presentation.

6 Testing Plan

Integration testing was used for all top-level modules which transitively tests all bottom level modules. This style of testing was chosen because it effectively tests how users will interact with the system instead of implementation details. Unit tests were not used as in practice they are often burdensome to write and maintain while providing minimal benefit in many cases due to much of the system being mocked out.

6.1 Testing Tools

The team will be use the following tools for implementing tests:

- Cypress – used to run all integration tests (API and UI).
- Jest – used for writing unit tests.
- Mock Service Workers – used for mocking out external service APIs.

6.2 API Tests

In the design specification the behaviour of HTTP handler modules was defined. This behaviour was used to define an expected output for a variety of inputs. This required creating tooling to build and teardown a testing database.
A serverless architecture was used for API execution which means that horizontal scaling of the system is handled by Vercel, the hosting provider. Therefore, **load testing** was not performed.

6.3 UI Tests

UI tests have been created for each way a user can interact with a page.

6.4 PASS/FAIL Criterion

The criterion used to determine PASS/FAIL of the the test cases id given below:

$$actual\_output == expected\_output$$

7 Traceability Matrix

Below tables go over the coverage of our test plan. Modules were tested multiple times and ensured that the Function Requirements implemented are correct.

7.1 UserInterface Module (M1)

Test ID	Description	Corresponding Requirement
M1T1	should allow a user to login	FR11, FR15
M1T2	should allow a user to logout	FR16
M1T3	should allow a user to delete their account	FR12
M1T4	should allow a user to change their name	FR17
M1T5	should allow a user to change their profile picture	FR17
M1T6	should allow a user to delete their posting from account page	FR2
M1T7	should allow a user to edit their posting from account page	FR19

7.2 PostHttpHandler Module (M2)

Test ID	Description	Corresponding Requirement
M2T1	should give a 200 when GETting an existing post	FR9
M2T2	should have user when GETting an existing post while AuthN	FR9
M2T3	should give a 400 when trying to GET a post with an invalid uuid	FR9
M2T4	should give a 404 when trying to GET a post that doesn't exist	FR9
M2T5	should give a 401 when trying to PUT when unauthenticated	(Pending changes to functional requirements)
M2T6	should give a 404 when PUTting to a post that doesn't exist (while AuthN)	(Pending changes to functional requirements)
M2T7	should give a 400 when trying to set the price to a negative number using PUT while AuthN	(Pending changes to functional requirements)
M2T8	should give a 400 when trying to set the price to a non-number value using PUT while AuthN	(Pending changes to functional requirements)
M2T9	should give a 200 when PUTting while AuthN with correct syntax	(Pending changes to functional requirements)
M2T10	should give a 401 when trying to DELETE when unauthenticated	FR3
M2T11	should give a 404 when trying to DELETE a post that does not exist	FR3
M2T12	should give a 403 when trying to DELETE someone else's post	FR3
M2T13	should give a 200 when DELETEing when authenticated	FR3
M2T14	should give 401 when attempting to create a post while not AuthN	FR1, FR2
M2T15	should give 200 when creating a post while AuthN	FR1, FR2
M2T16	should give 400 when attempting to create a post with bookId missing while AuthN	FR1, FR2
M2T17	should give 400 when attempting to create a post with price NaN while AuthN	FR1, FR2
M2T18	should give 400 when attempting to create a post with negative price while AuthN	FR1, FR2
M2T19	should give 400 when attempting to create a post with missing book while AuthN	FR1, FR2

7.3 BookHttpHandler Module (M3)

Test ID	Description	Corresponding Requirement
M3T1	should give a 200 when GETting an existing book, it's 4 posts, and it's course	FR4, FR9, FR10
M3T2	should give users of posts when GETting an existing book and it's posts while AuthN	FR4, FR9, FR10
M3T3	should return null for the google book data of a book that does not exist on google books	FR4, FR9, FR10
M3T4	should give a 404 when trying to GET a book with an isbn that does not exist	FR4, FR9, FR10
M3T5	should include the first 3 of 4 posts when GETting a book and it's posts with a length of 3	FR4, FR9, FR10
M3T6	should include the last post of 4 when GETting a book and it's posts with a length of 3 and page of 1	FR4, FR9, FR10
M3T7	should include zero posts when GETting a book and it's posts with a length of 3 and page of 2	FR4, FR9, FR10
M3T8	should return a default result of 4 posts for an invalid length input	FR4, FR9, FR10
M3T9	should return a default result of 4 posts for an invalid page input	FR4, FR9, FR10

7.4 CourseHttpHandler Module (M4)

Test ID	Description	Corresponding Requirement
M4T1	should give a 200 when GETting an existing course	FR5
M4T2	should give a 200 when GETting posts for an existing course	FR5
M4T3	should include user with posts when signed in when GETting an existing courses posts while AuthN	FR5
M4T4	should include first 3 of 4 posts when GETting posts for an existing course with a length of 3	FR5
M4T5	should include last post of 4 when GETting posts for an existing course with a length of 3 and page of 1	FR5
M4T6	should include zero posts when GETting posts for an existing course with a length of 3 and page of 2	FR5
M4T7	should give a 400 when trying to GET a course with an invalid uuid	FR5
M4T8	should give a 400 when trying to GET posts for a course with an invalid uuid	FR5
M4T9	should give a 404 when trying to GET a course that doesn't exist	FR5
M4T10	should give a 404 when trying to GET posts for a course with an invalid uuid	FR5

7.5 AuthHttpHandler Module (M5)

Test ID	Description	Corresponding Requirement
M5T1	should be able to send a magic link to a new user	FR11
M5T2	should be able to send a magic link to an existing user	FR15
M5T3	should be able to logout	FR16

7.6 UserHttpHandler Module (M6)

Test ID	Description	Corresponding Requirement
M6T1	should be able to get current user	FR9
M6T2	should not be able to get current user if not signed in	FR9
M6T3	should be able to get other users if logged in	FR9
M6T4	should not be able to get a user if not signed in	FR9
M6T5	should be able to update current user	FR17
M6T6	should not be able to update a user if not logged in	FR18
M6T7	should not be able to update a another user's account	FR18
M6T8	should be able to delete current user	FR12
M6T9	should not be able to delete other users	FR18
M6T10	should not be able to delete user if not logged in	FR18

8 Testing results

8.1 UserInterface Module (M1)

Test ID	Inputs	Expected Output	Actual Output	Result
M1T1	Login with MacID and visit login link	Magic link is sent and authenticates user	Magic link is sent and authenticates user	Pass
M1T2	Click logout button	User is unauthenticated	User is unauthenticated	Pass
M1T3	Visit account page and delete account	Account no longer exists	Unable to perform	Fail
M1T4	Visit account page and edit name	User's name is updated as expected	Unable to perform	Fail
M1T5	Visit account page and edit profile picture	User's profile picture is updated as expected	Unable to perform	Fail
M1T6	Visit account page and delete a post	Post no longer exists	Unable to perform	Fail
M1T7	Visit account page and edit a post	Post is updated as expected	Unable to perform	Fail

Note: Some of the above tests were unable to perform because the team is yet to implement these tests.

8.2 PostHttpHandler Module (M2)

Test ID	Inputs	Expected Output	Actual Output	Result
M2T1	Unauthenticated HTTP GET request to /api/post/{existing-post-uuid}	HTTP OK response, along with the specified post, without user information	Same as expected	Pass
M2T2	Authenticated HTTP GET request to /api/post/{existing-post-uuid}	HTTP OK response, along with the specified post and information on the user who created the post	Same as expected	Pass
M2T3	Unauthenticated HTTP GET request to /api/post/hello	HTTP BAD_REQUEST response	Same as expected	Pass
M2T4	Unauthenticated HTTP GET request to /api/post/{random-uuid}	HTTP NOT_FOUND response	Same as expected	Pass
M2T5	Unauthenticated HTTP PUT request to /api/post/{existing-post-uuid}	HTTP UNAUTHORIZED response	Same as expected	Pass
M2T6	Authenticated HTTP PUT request to /api/post/{random-uuid}	HTTP NOT_FOUND response	Same as expected	Pass
M2T7	Authenticated HTTP PUT request to /api/post/{existing-post-uuid}, where the new asking price is negative	HTTP BAD_REQUEST response	Same as expected	Pass
M2T8	Authenticated HTTP PUT request to /api/post/{existing-post-uuid}, where the new asking price is "salami"	HTTP BAD_REQUEST	Same as expected	Pass
M2T9	Authenticated HTTP PUT request to /api/post/{existing-post-uuid}, where the new description is set to "hi, mom!"	HTTP OK response, along with a copy of the post where the description has been changed to "hi, mom!"	Same as expected	Pass
M2T10	Unauthenticated HTTP DELETE request to /api/post/{existing-post-uuid}	HTTP UNAUTHORIZED response	Same as expected	Pass

Test ID	Inputs	Expected Output	Actual Output	Result
M2T11	Unauthenticated HTTP DELETE request to /api/post/{random-uuid}	HTTP NOT_FOUND response	Same as expected	Pass
M2T12	Authenticated HTTP DELETE request to /api/post/{another-persons-post-uuid} (try to delete someone else's post)	HTTP FORBIDDEN response	Same as expected	Pass
M2T13	Authenticated HTTP DELETE request to /api/post/{existing-post-uuid}	HTTP OK response	Same as expected	Pass
M2T14	Unauthenticated HTTP POST request to /api/post	HTTP UNAUTHORIZED response	Same as expected	Pass
M2T15	Authenticated HTTP POST request to /api/post, with all required params passed correctly	HTTP OK response	Same as expected	Pass
M2T16	Authenticated HTTP POST request to /api/post, with the bookId parameter missing	HTTP BAD_REQUEST response	Same as expected	Pass
M2T17	Authenticated HTTP POST request to /api/post, with the listing price set to "salami"	HTTP BAD_REQUEST response	Same as expected	Pass
M2T18	Authenticated HTTP POST request to /api/post, with the listing price set to a negative number	HTTP BAD_REQUEST response	Same as expected	Pass
M2T19	Authenticated HTTP POST request to /api/post, with the bookId parameter set to a book that doesn't exist in the database	HTTP BAD_REQUEST response	Same as expected	Pass

8.3 BookHttpHandler Module (M3)

Test ID	Inputs	Expected Output	Actual Output	Result
M3T1	HTTP GET request to /api/book/:isbn	HTTP OK response containing 4 post objects, without users included, and including expected values	Same as expected	Pass
M3T2	Authenticated HTTP GET request to /api/book/:isbn	HTTP OK response containing 4 post objects with users included	Same as expected	Pass
M3T3	HTTP GET request to /api/book/:isbn where isbn has no googleBooks data	HTTP OK response with googleBooks field set to null	Same as expected	Pass
M3T4	HTTP GET request /api/book/:isbn with invalid isbn	HTTP Not Found Response	Same as expected	Pass
M3T5	HTTP GET request to /api/book/:isbn/?length=3&page=0	HTTP OK response containing 3 post objects	Same as expected	Pass
M3T6	HTTP GET request to /api/book/:isbn/?length=3&page=1	HTTP OK response containing 1 post object	Same as expected	Pass
M3T7	HTTP GET request to /api/book/:isbn/?length=3&page=2	HTTP OK response containing 0 post objects	Same as expected	Pass
M3T8	HTTP GET request to /api/book/:isbn/?length=-1&page=0	HTTP OK response containing default options with 4 post objects	Same as expected	Pass
M3T9	HTTP GET request to /api/book/:isbn/?length=1&page=a	HTTP OK response containing default options with 4 post objects	Same as expected	Pass

8.4 CourseHttpHandler Module (M4)

Test ID	Inputs	Expected Output	Actual Output	Result
M4T1	HTTP GET request to /api/-course/id/:id	HTTP OK response	HTTP OK response	Pass
M4T2	Unauthenticated HTTP GET request to /api/-course/id/:id/posts	HTTP OK response containing 4 post objects without users included	HTTP OK response containing 4 post objects without users included	Pass
M4T3	Authenticated HTTP GET request to /api/-course/id/:id/posts	HTTP OK response containing 4 post objects with users included	HTTP OK response containing 4 post objects with users included	Pass
M4T4	HTTP GET request to /api/-course/id/:id/posts?length=3 &page=0	HTTP OK response containing 3 post objects	HTTP OK response containing 3 post objects	Pass
M4T5	HTTP GET request to /api/-course/id/:id/posts?length=3 &page=1	HTTP OK response containing 1 post object	HTTP OK response containing 1 post object	Pass
M4T6	HTTP GET request to /api/-course/id/:id/posts?length=3 &page=2	HTTP OK response containing 0 post objects	HTTP OK response containing 0 post objects	Pass
M4T7	HTTP GET request to /api/-course/id/hello	HTTP Bad Request response	HTTP Bad Request response	Pass
M4T8	HTTP GET request to /api/-course/id/hello/posts	HTTP Bad Request response	HTTP Bad Request response	Pass
M4T9	HTTP GET request to /api/-course/id/{random uuid}	HTTP Not Found response	HTTP Not Found response	Pass
M4T10	HTTP GET request to /api/course/id/{random uuid}/posts	HTTP Not Found response	HTTP Not Found response	Pass

8.5 AuthHttpHandler Module (M5)

Test ID	Inputs	Expected Output	Actual Output	Result
M5T1	HTTP POST request to /api/auth/magic containing new MacID	HTTP OK response	HTTP OK response	Pass
M5T2	HTTP POST request to /api/auth/magic containing existing MacID	HTTP OK response	HTTP OK response	Pass
M5T3	HTTP POST request to /api/auth/logout	HTTP OK response	HTTP OK response	Pass

8.6 UserHttpHandler Module (M6)

Test ID	Inputs	Expected Output	Actual Output	Result
M6T1	Authenticated HTTP GET request to /api/user	HTTP OK response containing current user object	HTTP OK response containing current user object	Pass
M6T2	Unauthenticated HTTP GET request to /api/user	HTTP NO_CONTENT response	HTTP NO_CONTENT response	Pass
M6T3	Authenticated HTTP GET request to /api/user/:id	HTTP OK response containing matching user object	HTTP OK response containing matching user object	Pass
M6T4	Unauthenticated HTTP GET request to /api/user/:id	HTTP UNAUTHORIZED response	HTTP UNAUTHORIZED response	Pass
M6T5	Authenticated HTTP PUT request to /api/user/:currentUserId containing user update	HTTP OK response containing updated user	HTTP OK response containing updated user	Pass
M6T6	Unauthenticated HTTP PUT request to /api/user/:id containing user update	HTTP UNAUTHORIZED response	HTTP UNAUTHORIZED response	Pass
M6T7	Authenticated HTTP PUT request to /api/user/:id containing other user update	HTTP FORBIDDEN response	HTTP FORBIDDEN response	Pass
M6T8	Authenticated HTTP DELETE request to /api/user/:currentUserId	HTTP OK response	HTTP OK response	Pass
M6T9	Authenticated HTTP DELETE request to /api/user/:id	HTTP FORBIDDEN response	HTTP FORBIDDEN response	Pass
M6T10	Unauthenticated HTTP DELETE request to /api/user/:id	HTTP UNAUTHORIZED response	HTTP UNAUTHORIZED response	Pass

9 More Than the Tests

9.1 Code Reviews

All code pushed to the Git repository was peer-reviewed by other members of the team. New code had to be reviewed and approved by at least one other member before merging into the main branch.
History of all peer-reviewed merge results can be found on GitHub.

9.2 Static Analysis

Static analysis was performed using eslint. Common JavaScript and React pitfalls were avoided using eslint and code styling was enforced by the Prettier code formatting tool. We run eslint as a part of our continuous integration testing to ensure that new code being added to the main repository adheres to the code style. Furthermore, type safety at the static analysis level was added by using TypeScript over vanilla JavaScript.

10 Supporting Material

Please refer to prior design documents submitted.