HofTAPS

Online Textbook Exchange

Project Plan

Awab Abedin, Robbie Bernstein, Thomas Catania, Sanjit Menon, James Nastasi, Marcus Wiesenhart

Version 3 - 4/24/25

Table of Contents

1. Introduction

The Textbook Exchange Application is an online marketplace where students and book enthusiasts can buy and sell used textbooks efficiently. The platform allows sellers to upload books they want to sell by providing details such as book title, author, ISBN, condition, and price. Once listed, buyers can search for books using ISBN, title, or author and purchase a textbook. Once the seller accepts the purchase, the sale is complete.

Future enhancements could include a chat system for direct negotiations, a user rating system to build trust, AI-powered price suggestions, and options for local pickup. By creating an efficient, cost-effective way for students to buy and sell textbooks, this platform can help reduce the financial burden of purchasing books while ensuring a seamless and intelligent marketplace experience.

2. Risk Analysis

Developing the Textbook Exchange Application involves various risks that must be carefully managed to ensure a successful outcome. One major project risk is scope creep which can lead to delays, resource strain, and increased costs. Additionally, unrealistic deadlines can emerge when tasks are underestimated, making it difficult to meet project milestones. Reliance on third-party services such as Firebase and Google Books means that outages impact our service. HofTAPS could not run some or all functionalities if these were to have outages.

From a technical perspective, software flaws pose a significant risk, making it essential to conduct comprehensive functional and non-functional testing to identify and resolve bugs early. System failures could lead to downtime, poor user experience, and security vulnerabilities, requiring robust infrastructure, monitoring, and error-handling mechanisms. If not properly addressed, these challenges could impact the overall performance, reliability, and security of the platform. Lacking technical skills with certain applications also poses a risk as it becomes more difficult to troubleshoot and find solutions to issues that may arise.

3. Risk Mitigation Strategy

3.1 Unplanned Changes Risk

To mitigate unplanned changes in the project scope, the team conducts impact analysis before accepting changes in the project code base. This helps assess risks and determine whether such changes will cause significant delays, introduce new dependencies, or impact any existing functionalities. In addition, the team establishes version control to help developers in the team track and revert to pre-existing versions of code to ensure functionality.

3.2 Branch Naming Convention Non-Adherence Risk

The team utilizes GitHub Flow that permits consistency through naming conventions. The team also includes periodic peer code reviews to ensure the naming conventions between each team member remains consistent. This limits the amount of confusion and permits greater collaboration amongst the team members in a way that each member can review each others' code and ensure consistency.

3.3 Inadequate Documentation Risk

Poor documentation may lead to informational gaps, creating discrepancies between developers and stakeholders to understand the codebase. To mitigate this, the team enforces documentation as a mandatory part of the code review process. Therefore, any reforms in the codebase are documented by said team member.

3.4 Miscommunication Risk

To prevent project delays due to a misalignment in requirements, the team conducts regular meetings to synchronize team members. This maintains an up-to-date project blueprint in order to improve the team coordination. The team also utilizes collaboration tools such as iMessage and Discord to ensure there is clear communication between team members with no confusion on what tasks each team member is responsible for.

3.5 Unauthorized Access Risk

To prevent data leaks or security breaches within the team, access logs and permissions are accessed with GitHub to determine which team members access certain parts of the project at specific times. Since the team is utilizing API keys and service accounts, the team ensures these are not shared publicly and only those on the team are able to use them.

3.6 Dependency Risk

Since the team is utilizing external services including Google Books API, the team needs to be aware of when these services are impacted by outages. Ensuring that some usability remains during an outage enhances the user experience. For example, a Google Books outage would only impact textbook uploading and not purchasing. It is important that functionalities that are not impacted by outages remain available.

4. Activity Delegation and Activity Charts

The activities (denoted by tags for traceability) were split up into numerous components that create an up and running textbook exchange platform. Each activity is selected and assigned to the team members based on how efficient each person would be at that specific task. We as a team are focused on high risk tasks and have more people on said tasks, since that is a bigger concern for us.

4.1 Activity Descriptions Chart

The following chart depicts each activity with its description, any dependencies, its duration in days, and the risk (high, medium, low). Each activity has a unique tag that is consistent across all documentation.

Key: B#: Backend; F#: Frontend

Tag	Activity	Description	Dependencies	Duration (Days)	Risk
B1	Create Database	The database stores all user and textbook data	None	3	High
B2	Store/Access User Data	Allows user information to be stored and referenced	B1	4	High
B3	Store/Access Textbook Data	Allows textbook information to be stored and referenced	B4	4	High
B4	API Integration	Retrieve textbook information, such as title, author, and cover images, and store them within the database	B1	4	High
B5	Method to Create Account	Allows users to create an account by entering their name, Hofstra email, ID number, and a password	B2	2	Med
B6	Method to Add Textbook	Allows users to add a textbook to the database by entering the textbooks title, author, ISBN, condition, and price	B3	3	Med
B7	Method to Remove Textbook	Textbooks are automatically removed from the database when a sale is completed	B3	3	High
B8	Method to Login	Allows users to sign into the site to make purchases or sell	B5	2	Low

B9	Notifications System	Sends in-site and email notifications to users	B10	5	High
B10	Method to Add to Wishlist	Creates a lists of textbooks that a user is interesting in purchasing	B15	2	Low
B11	Method to Search	Allows users to search for textbooks by keywords, title, or ISBN	F14	3	Med
B12	Search Filters	Limits search results based on criteria such as condition or price	B11	3	Med
B13	Sort Searches	Sorts search results based on criteria such as price or title	B12	5	Med
B14	Compile Featured Items	List of textbooks that have been viewed most often	F8	2	Low
B15	User Verification	Verifies a user by sending an email with a verification code	B5	4	Med
B16	Method to Edit User Data	Allows users to edit their information	B18	2	Low
B18	Method to Purchase Textbook	Allows a user to purchase a textbook	B7	4	Med
F1	Create Home Page	The home page is the page users first see when entering the site and allows for site navigation	B6	4	Med
F2	Create Login Page	Page where users can login using the user login method	B8	2	Low
F3	Create Account Page	Page where users can create an account using the create account method	B2	2	Low
F5	Create Notifications Page	Page that displays all of a user's notifications	F3	2	Low
F6	Create Wishlist Page	Page that displays a user's wishlist	F2	2	Low
F7	Create Listings Page	Page which displays the listings from a particular user search	B3	2	Med
F8	Create Individual Listings Page	Page for an individual listing, displaying the textbook's information and allowing the user to purchase	F16	2	Med
F9	Create Sell Page	Page which allows a user to sell a textbooks using the add textbook method	B6	2	Low
F13	Search Bar	Allows the user to search for	F1	3	Med

		textbooks using the search method			
F14	Left Bar	Navigation bar on each page which allows access to other site pages	F13	2	Med
F16	Textbook Carousel	Allows the user to scroll across the featured textbooks	F7	6	Low
F17	Image Uploading	Allows the user to upload images of their textbook when they are selling a textbook	F9	4	Med
F18	Cross-Platform Support	Ensures seamless support of the site across desktop and mobile	B13	6	Med
F19	AWS Hosting	Allows users to connect to the site from any device	F18	6	High

4.2 Project Schedule

The following chart depicts the project schedule. Each activity is listed alongside its unique tag, any dependencies, and its duration. The early start/finish, late start/finish, and slack are also listed with anticipated start and end dates based on the early start and finishes. Each activity also contains a delegation of man power and the corresponding man days.

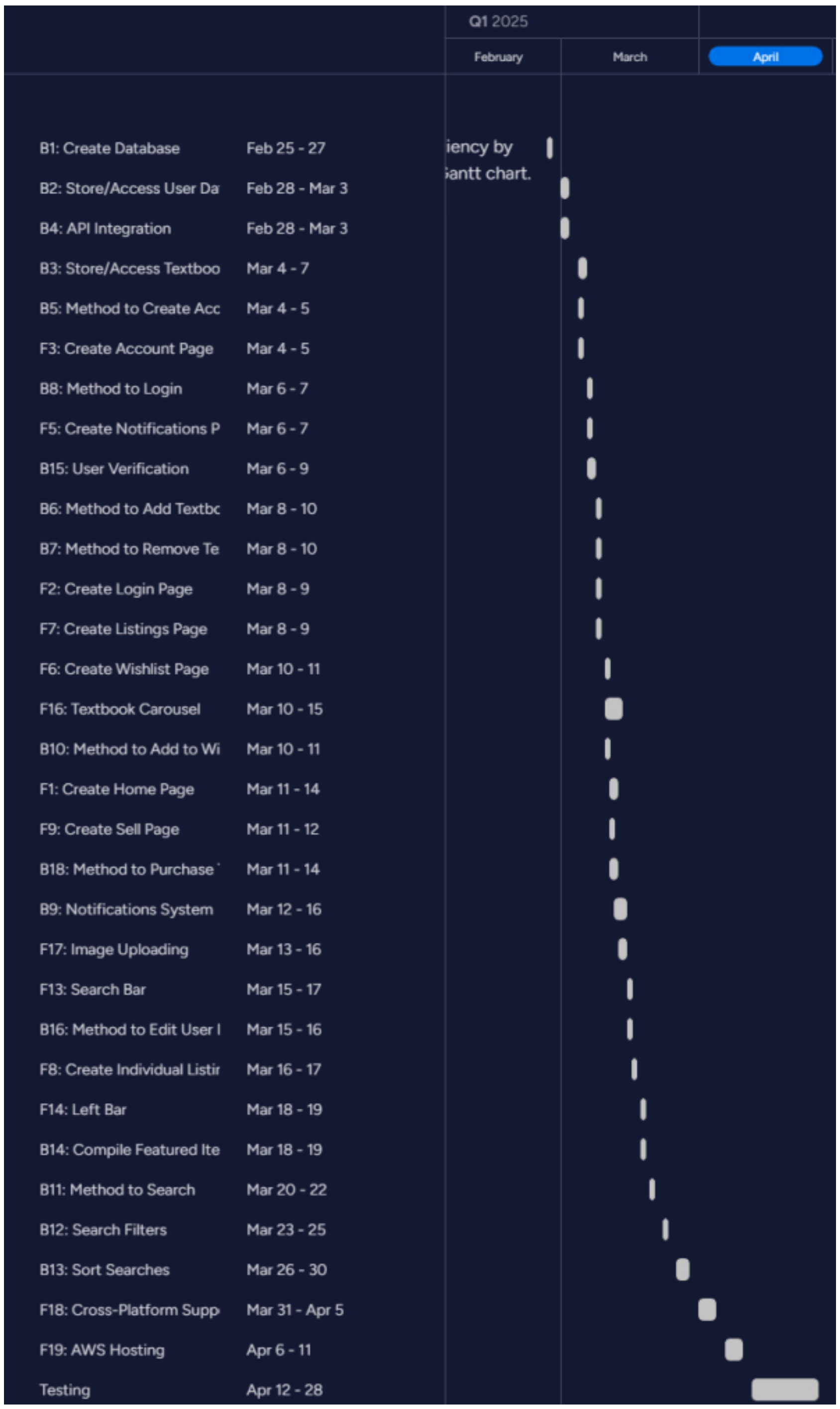
Tag	Name	Dependency	Duration	ES	EF	LS	LF	Slack	Delegation	Man Days	Start Day	End Day
B1	Create Database	None	3	1	3	1	3	0	Robbie	3	2/25	2/27
B2	Store/Access User Data	B1	4	4	7	27	30	23	Robbie	4	2/28	3/3
B3	Store/Access Textbook Data	B4	4	8	11	8	11	0	Robbie	4	3/4	3/7
B4	API Integration	B1	4	4	7	4	7	0	Sanjit	4	2/28	3/3
B5	Method to Create Account	B2	2	8	9	31	32	23	Thomas	2	3/4	3/5
B6	Method to Add Textbook	B3	3	12	14	12	14	0	Robbie	3	3/8	3/10

B7	Method to Remove Textbook	B3	3	12	14	35	37	23	Robbie	3	3/8	3/10
B8	Method to Login	B5	2	10	11	38	39	28	Thomas	2	3/6	3/7
B9	Notifications System	B10	5	16	20	39	43	23	Sanjit, Marcus	10	3/12	3/16
B10	Method to Add to Wishlist	B15	2	14	15	37	38	23	Thomas	2	3/10	3/11
B11	Method to Search	F14	3	24	26	24	26	0	Sanjit, Awab	6	3/20	3/22
B12	Search Filters	B11	3	27	29	27	29	0	James, Sanjit	6	3/23	3/25
B13	Sort Searches	B12	5	30	34	30	34	0	James	5	3/26	3/30
B14	Compile Featured Items	F8	2	22	23	42	43	20	Marcus, Thomas	4	3/18	3/19
B15	User Verification	B5	4	10	13	33	36	23	Awab, Thomas	8	3/6	3/9
B16	Method to Edit User Data	B18	2	19	20	42	43	23	Thomas	2	3/15	3/16
B18	Method to Purchase Textbook	B7	4	15	18	38	41	23	Sanjit, Marcus	8	3/11	3/14
F1	Create Home Page	B6	4	15	18	15	18	0	James, Sanjit	8	3/11	3/14
F2	Create Login Page	B8	2	12	13	40	41	28	Thomas	2	3/8	3/9
F3	Create Account Page	B2	2	8	9	40	41	32	James	2	3/4	3/5
F5	Create Notifications Page	F3	2	10	11	42	43	32	James, Marcus	4	3/6	3/7
F6	Create Wishlist Page	F2	2	14	15	42	43	28	Thomas	2	3/10	3/11
F7	Create Listings Page	B3	2	12	13	32	33	20	James	2	3/8	3/9
F8	Create Individual Listings Page	F16	2	20	21	40	41	20	James	2	3/16	3/17

F9	Create Sell Page	B6	2	15	16	38	39	23	James	2	3/11	3/12
F13	Search Bar	F1	3	19	21	19	21	0	Sanjit, Marcus	6	3/15	3/17
F14	Left Bar	F13	2	22	23	22	23	0	Sanjit and James	4	3/18	3/19
F16	Textbook Carousel	F7	6	14	19	34	39	20	James, Awab, Thomas	18	3/10	3/15
F17	Image Uploading	F9	4	17	20	40	43	23	Sanjit	8	3/13	3/16
F18	Cross-Platform Support	B13	6	35	40	35	40	0	Robbie	6	3/31	4/5
F19	AWS Hosting	F18	6	41	43	41	43	0	Robbie	6	4/6	4/11
Total Man Days:										**145**		

4.3 Gantt Chart

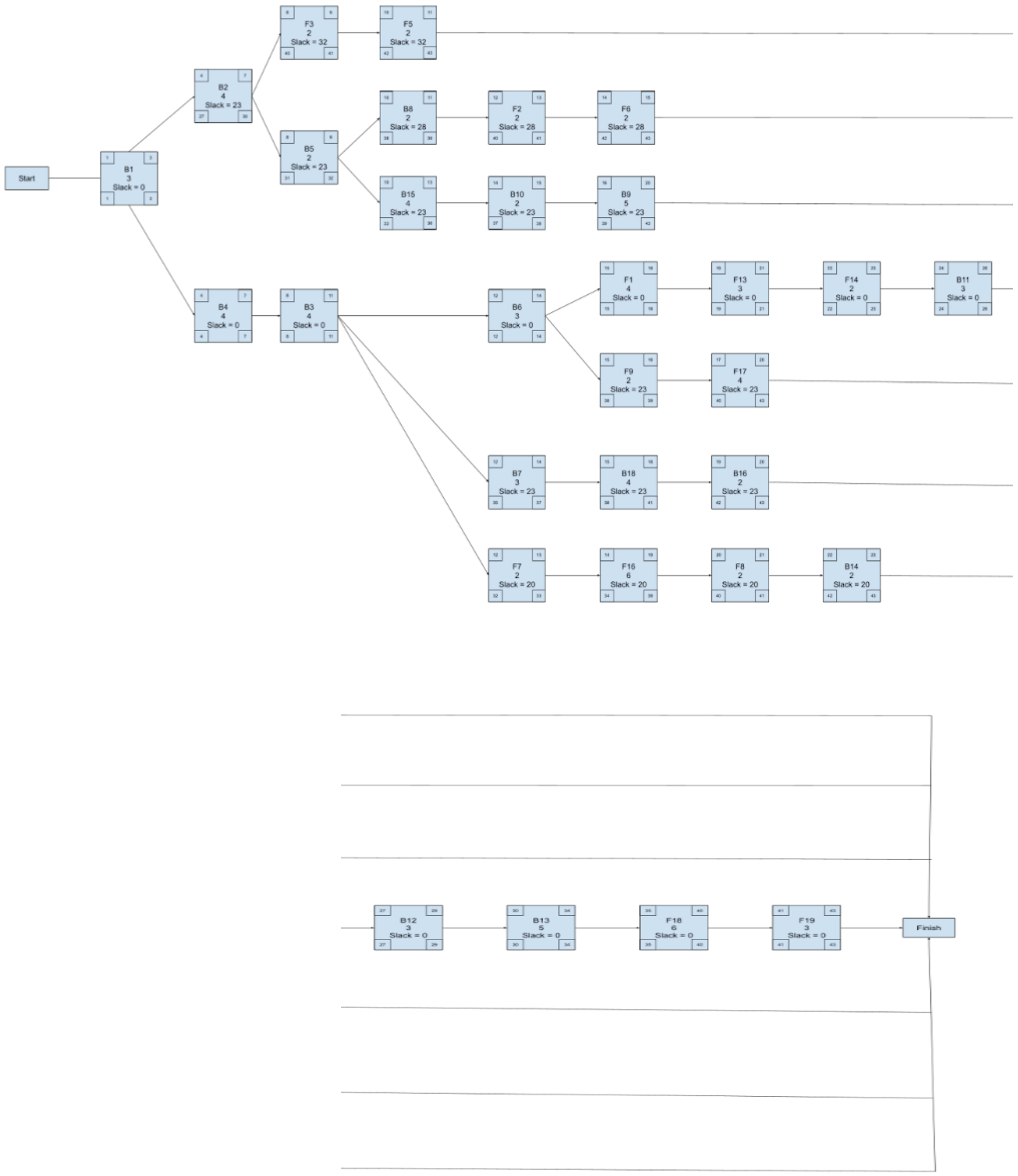
The following is the Gantt Chart for this project. Each start and end date as well as any activity overlaps can be seen here:



B1: Create Database	Feb 25 - 27
B2: Store/Access User Da	Feb 28 - Mar 3
B4: API Integration	Feb 28 - Mar 3
B3: Store/Access Textboo	Mar 4 - 7
B5: Method to Create Acc	Mar 4 - 5
F3: Create Account Page	Mar 4 - 5
B8: Method to Login	Mar 6 - 7
F5: Create Notifications P	Mar 6 - 7
B15: User Verification	Mar 6 - 9
B6: Method to Add Textbc	Mar 8 - 10
B7: Method to Remove Te	Mar 8 - 10
F2: Create Login Page	Mar 8 - 9
F7: Create Listings Page	Mar 8 - 9
F6: Create Wishlist Page	Mar 10 - 11
F16: Textbook Carousel	Mar 10 - 15
B10: Method to Add to Wi	Mar 10 - 11
F1: Create Home Page	Mar 11 - 14
F9: Create Sell Page	Mar 11 - 12
B18: Method to Purchase	Mar 11 - 14
B9: Notifications System	Mar 12 - 16
F17: Image Uploading	Mar 13 - 16
F13: Search Bar	Mar 15 - 17
B16: Method to Edit User I	Mar 15 - 16
F8: Create Individual Listir	Mar 16 - 17
F14: Left Bar	Mar 18 - 19
B14: Compile Featured Ite	Mar 18 - 19
B11: Method to Search	Mar 20 - 22
B12: Search Filters	Mar 23 - 25
B13: Sort Searches	Mar 26 - 30
F18: Cross-Platform Supp	Mar 31 - Apr 5
F19: AWS Hosting	Apr 6 - 11
Testing	Apr 12 - 28

4.4 Project Activity Chart

The following is the project activity chart, showing each activity with its tag, early start/finish, late start/finish, slack, and all project paths:



The project paths are as follows:

B1, B2, F3, F5: 11 Days

B1, B2, B5, B8, F2, F6: 15 Days

B1, B2, B5, B15, B10, B9: 20 Days

B1, B4, B3, B6, F1, F13, F14, B11, B12, B13, F18, F19: 43 Days* Critical Path

B1, B4, B3, B6, F9, F17: 20 Days

B1, B2, B3, B7, B18, B16: 20 Days

B1, B4, B3, F7, F16, F8, B14: 23 Days

5. Configuration Management Strategy

5.1 Versioning Convention

Each document follows the same versioning convention. Versions are listed starting at Version 1, and are increased by 1 for each version. Small updates may be viewed as less than a version, so in this case a .1 may be added to the version number. There are not many versions of each document to be expected, so more detailed versioning is not necessary.

5.2 Branching Strategy

The project will have a Main Branch which houses a stable build of the project. This branch is not altered until features are tested locally on the corresponding team member's branch before being pushed onto the main branch. This prevents faulty or buggy code from becoming part of the main source. Team members may utilize additional branches for testing purposes or to build alternate versions of the application. Any changes to these branches are later merged into the main branch when they are set to become part of the main build.

5.3 Change Management

Change requests are made through GitHub’s pull requests. Users pull existing builds from the main branch and make changes on their local machines. These changes are then merged back into the main branch once sufficient testing and verification has occurred. GitHub automatically checks for conflicts to ensure that new code does not conflict with existing code.

5.4 Merge Strategy

A proper merge strategy ensures new code is compatible with existing code. Local branches or any additional branches are kept up-to-date with changes made to the main branch. Team members regularly merge main branch changes into their personal branches to ensure they are working with the latest version of the application. Merges into the main branch occur in two ways. Pushes from local machines into the main branch only occur once sufficient testing has been carried out on the local machine to make sure the main branch includes code that works. Merges from separate GitHub branches into the main branch occur when features on those branches need to be brought over to the main branch. The same testing and verification requirements are necessary for this merge to occur, with these testing requirements discussed in further detail in Section 6. Any conflicts that exist between merging branches are pointed out by GitHub automatically and are adjusted before the final merge.

5.5 Documentation

Each push or merge to the main branch includes a short description of what changes were made in the build. This allows all team members to know what changes were made in a merge and to know which changes they need to merge into their local machines. All pushes, pulls, and merges are listed in the HofTAPS GitHub allowing team members to see all changes that have been made over time. They can see when the change was made, who pushed the changes, and what files were modified.

6. Quality Assessment Planning

6.1 Importance of Testing

The inclusion of consistent testing during and after the development of HofTAPS is tied to the success of carrying out this project plan. Testing not only ensures that the required specifications laid out for the client are met, but also assists in updating the software to newer versions if new developers are assigned to work on the website. Quality assurance testing allows developers to discover any inconsistencies or bugs within the project while maintaining a clear roadmap for the future work.

6.2 Testing Resources

The front-end development team is using Bootstrap, which is an open-source framework that includes languages like CSS, JavaScript and HTML. Bootstrap provides the HofTAPS team with consistent UI features, as well as some pre-built components which helps assist the team members when testing on the front-end. This platform also prevents bugs from being created due to its well-established resources that make front-end development seamless for the HofTAPS team. HTML and CSS are relatively new for us, so utilizing bootstrap mitigates any concerns when it comes to testing.

The back-end development team is using Firebase, which is a Google owned open-source database tool that allows users to develop software effectively. Not only does Firebase have its own testing studio that provides testing resources, there is also generative AI integrated into Firebase which facilitates writing and testing new code.

6.3 Testing Methods

There are several testing methods that are utilized throughout the duration of this project. We can break these into 5 different components:

6.3.1 Regression Testing: Regression testing is one of the most important forms of testing, as failure to regularly perform this method can result in a full project breakdown. We must make sure as developers that any new code or assets added to the project do not break the foundation that had been built for HofTAPS. A large part of our regression testing is centered around making sure the database still works properly, as that part of the website is essential for making sure the textbook exchange platform runs as planned.

6.3.2 Unit Testing: For each new piece of code that we write, the HofTAPS team tests in the form of “unit testing” which means that testing will be done down to every new function that is created for the project. This ensures that the new additions to the project are bug free on their own.

6.3.3 Integration Testing: Since we are working in a team environment, integrating some of the team member’s assets together and making sure they work collectively is essential for the development of HofTAPS. Regular integration testing is done shortly after new assets from different team members are uploaded to the GitHub repository, and even if one piece of code isn’t meshing well with the other components, it is changed immediately. Code compatibility between different team members is one of the highest priorities we have in this project.

6.3.4 System Testing: When the whole system is up and running, system testing is done to make sure the whole project is functioning as it is intended to. This involves making sure the frontend and backend components work together, successful integration of the API, making sure the database is working, and user interface components are implemented correctly.

6.3.5 User Acceptance Testing: User acceptance testing is conducted during the multiple demo periods throughout the semester. We have several features that are able to be tested by the clientele and our group accounts for as many possibilities as we can so that any user testing our software has a seamless and bug-free experience. This form of testing provides outside feedback that our team uses to make the software even better.

6.4 Acceptance Criteria for Tests

Testing is accepted when the code is proven to have met the interface requirements, the system features match, and when the non-functional requirements are met. This allows for the code to run smoothly and efficiently while also maintaining the original specified requirements that the client expects.

Most importantly, testing is accepted when the client is satisfied with the product. Ultimately, the client decides if a product meets their needs and it is therefore required that the client is fully satisfied with the product and all of their requirements have been met for testing to be complete. Testing in this way guarantees a fair benchmark for acceptable code.

It is very important that we have acceptable criteria for the testing phases because the site needs to have a good interface with fast page response time, minimal breakpoints and live updating regarding listings. In addition, we need to have good test criteria so that the system features of the main page, listing page, buy page and account page all function as planned. The non-functional requirements like security, maintainability and usability must be met in order for our test criteria to be acceptable. HofTAPS needs to be efficient, usable and user-friendly, and have high standards for our acceptance testing criteria.