HofTAPS

Online Textbook Exchange

Project Plan

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

Marcus Wiesenhart

Version 1 - 2/24/25

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, book name, or author and place bids on the ones they are interested in purchasing. Once a bid is accepted, the transaction is finalized, both parties receive an email confirmation for their records.

To enhance the user experience, the platform incorporates AI-powered search and recommendation systems. The search functionality is optimized using Natural Language Processing (NLP) models, allowing users to find books even if they enter partial or misspelled queries. Additionally, AI-based recommendations suggest popular and relevant books based on user behavior and market trends. This ensures a smooth and intuitive book discovery process.

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. Budget overruns are another concern, as the project relies on third-party services for search engine optimization and email verification, requiring careful financial planning to cover multiple development phases.

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.

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 Code Regression Risk

In order to mitigate any changes in the codebase that may cause software defects, the team implements Automated Regression Testing through the Testim browser extension; this allows the team to develop new user scripts and ensure functionality. By using automation, the team does not need to rely on manual testing due to the limiting human resource availability.

3.3 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.

3.4 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. The team also implements tools such as Markdown to store documentation alongside the codebase.

3.5 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 Slack and Discord to ensure there is clear communication between team members with no confusion on what tasks each team member is responsible for.

3.6 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 regularly rotates credentials to prevent unauthorized reuse.

3.7 Dependency Risk

Since the team is utilizing external services including Google Books API, the team will use dependency management tools such as Dependabot that alert team members when one of our software dependencies includes a known vulnerability. In addition, the team creates contingency plans, including utilizing Google Books API as it is open while Amazon API requires a key and contains more fetch errors.

4. Activity Delegation

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.

5. Activity Charts

5.1 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; S#: Security

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	B9	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	B17	3	Med
B7	Method to Remove Textbook	Textbooks are automatically removed from the database when a sale is completed	B17	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	B2	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	B6	3	Med

B12	Search Filters	Limits search results based on criteria such as condition or price	B13	3	Med
B13	Sort Searches	Sorts search results based on criteria such as price or title	B11	3	Med
B14	Compile Featured Items	List of textbooks that have been viewed most often	B17	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	B2	2	Low
B17	Store/Access Listing Data	Allows listing data to be stored in stored and referenced	B3	4	High
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	B12	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	B16	2	Low
F4	Create Account History Page	Page where the user can see textbooks they have viewed, purchased, and sold	F3	2	Low
F5	Create Notifications Page	Page that displays all of a user's notifications	F4	2	Low
F6	Create Wishlist Page	Page that displays a user's wishlist	B10	2	Low
F7	Create Listings Page	Page which displays the listings from a particular user search	B14	2	Med
F8	Create Individual Listings Page	Page for an individual listing, displaying the textbook's information and allowing the user to purchase	F7	2	Med
F9	Create Sell Page	Page which allows a user to sell a textbooks using the add textbook method	B6	2	Low
F10	Create Settings Page	Page which allows the user to edit their account settings	F5	1	Low

F11	Fluid Grid Layout	Allows the site to adjust to screen size	F2, F6, F10 F13, F14, F16, F17	5	Med
F12	Breakpoints		F11	3	Med
F13	Search Bar	Allows the user to search for textbooks using the search method	F1	3	Med
F14	Left Bar	Navigation bar on each page which allows access to other site pages	F13	2	Med
F15	Featured Items	Displays featured items from the featured list on the home page	F8	3	Low
F16	Textbook Carousel	Allows the user to scroll across the featured textbooks	F15	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	F12	5	Med
S1	Password Encryption	Ensures user passwords are encrypted	B2	3	Med
S2	Notification Encryption	Ensures notifications to users are encrypted	S3	3	Med
S3	User Encryption	Ensures all users' information is encrypted	S1	3	Med
S4	Email Verification Link	An email is sent to a user creating an account with a verification code	S2	2	Low

5.2 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.

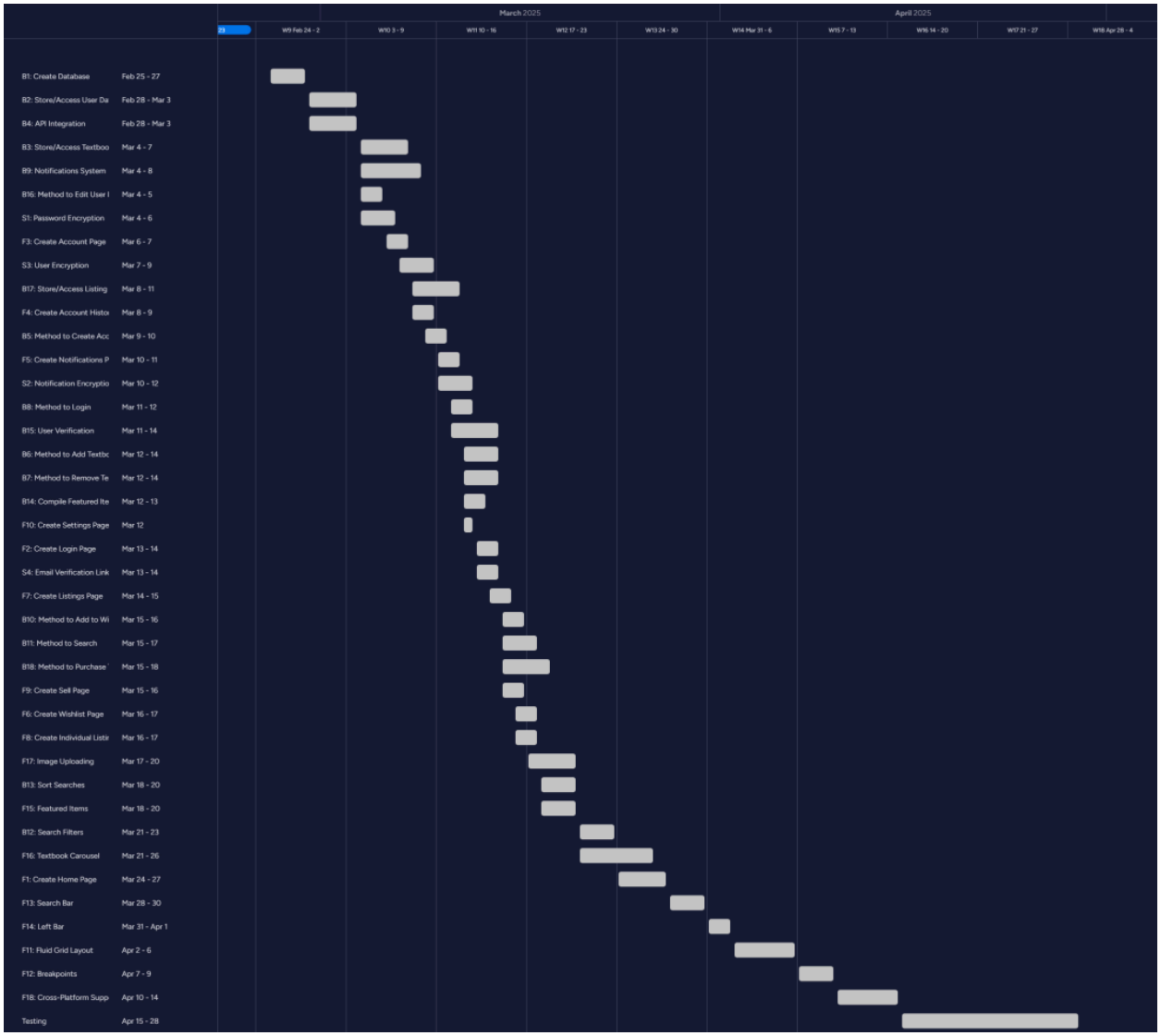
Tag	Name	Dependency	Duration	ES	EF	LS	LF	Slack	Delegation	Man Days	Start Day	End Day
B1	Create Database	None	3	1	3	1	3	0	6 people	18	2/25	2/27
B2	Store/Access User Data	B1	4	4	7	18	21	14	3 people	12	2/28	3/3
B3	Store/Access Textbook Data	B4	4	8	11	8	11	0	1 person	4	3/4	3/7
B4	API Integration	B1	4	4	7	4	7	0	3 people	12	2/28	3/3
B5	Method to Create Account	B9	2	13	14	27	28	14	1 person	2	3/9	3/10
B6	Method to Add Textbook	B17	3	16	18	16	18	0	1 person	3	3/12	3/14
B7	Method to Remove Textbook	B17	3	16	18	43	45	27	2 people	6	3/12	3/14
B8	Method to Login	B5	2	15	16	33	34	18	1 person	2	3/11	3/12
B9	Notifications System	B2	5	8	12	22	26	14	2 people	10	3/4	3/8
B10	Method to Add to Wishlist	B15	2	19	20	33	34	14	1 person	2	3/15	3/16
B11	Method to Search	B6	3	19	21	19	21	0	2 people	6	3/15	3/17
B12	Search Filters	B13	3	25	27	25	27	0	2 people	6	3/21	3/23
B13	Sort Searches	B11	3	22	24	22	24	0	1 person	3	3/18	3/20
B14	Compile Featured Items	B17	2	16	17	22	23	6	1 person	2	3/12	3/13
B15	User Verification	B5	4	15	18	29	32	14	2 people	8	3/11	3/14

B16	Method to Edit User Data	B2	2	8	9	26	27	18	1 person	2	3/4	3/5
B17	Store/Access Listing Data	B3	4	12	15	12	15	0	3 people	12	3/8	3/11
B18	Method to Purchase Textbook	B7	4	19	22	46	49	27	2 people	8	3/15	3/18
F1	Create Home Page	B12	4	28	31	28	31	0	2 people	8	3/24	3/27
F2	Create Login Page	B8	2	17	18	35	36	18	1 person	2	3/13	3/14
F3	Create Account Page	B16	2	10	11	28	29	18	1 person	2	3/6	3/7
F4	Create Account History Page	F3	2	12	13	30	31	18	1 person	2	3/8	3/9
F5	Create Notifications Page	F4	2	14	15	32	33	18	2 person	4	3/10	3/11
F6	Create Wishlist Page	B10	2	21	22	35	36	14	1 person	2	3/16	3/17
F7	Create Listings Page	B14	2	18	19	24	25	6	1 person	2	3/14	3/15
F8	Create Individual Listings Page	F7	2	20	21	26	27	6	1 person	2	3/16	3/17
F9	Create Sell Page	B6	2	19	20	31	32	12	1 person	2	3/15	3/16
F10	Create Settings Page	F5	1	16	16	24	26	10	1 person	1	3/12	3/12
F11	Fluid Grid Layout	F2, F6, F10 F14, F16, F17	5	37	41	37	41	0	6 people	30	4/2	4/6
F12	Breakpoints	F11	3	42	44	42	44	0	6 people	18	4/7	4/9
F13	Search Bar	F1	3	32	34	32	34	0	2 people	6	3/28	3/30
F14	Left Bar	F13	2	35	36	35	36	0	2 people	4	3/31	4/1
F15	Featured Items	F8	3	22	24	28	30	6	1 person	3	3/18	3/20
F16	Textbook Carousel	F15	6	25	30	31	36	6	3 people	18	3/21	3/26

F17	Image Uploading	F9	4	21	24	33	36	12	2 people	8	3/17	3/20
F18	Cross-Platform Support	F12	5	45	49	45	49	0	6 people	30	4/10	4/14
S1	Password Encryption	B2	3	8	10	39	41	31	1 person	3	3/4	3/6
S2	Notification Encryption	S3	3	14	16	45	47	31	1 person	3	3/10	3/12
S3	User Encryption	S1	3	11	13	42	44	31	1 person	3	3/7	3/9
S4	Email Verification Link	S2	2	17	18	48	49	31	1 person	2	3/13	3/14
Total Man Days:										**273**		

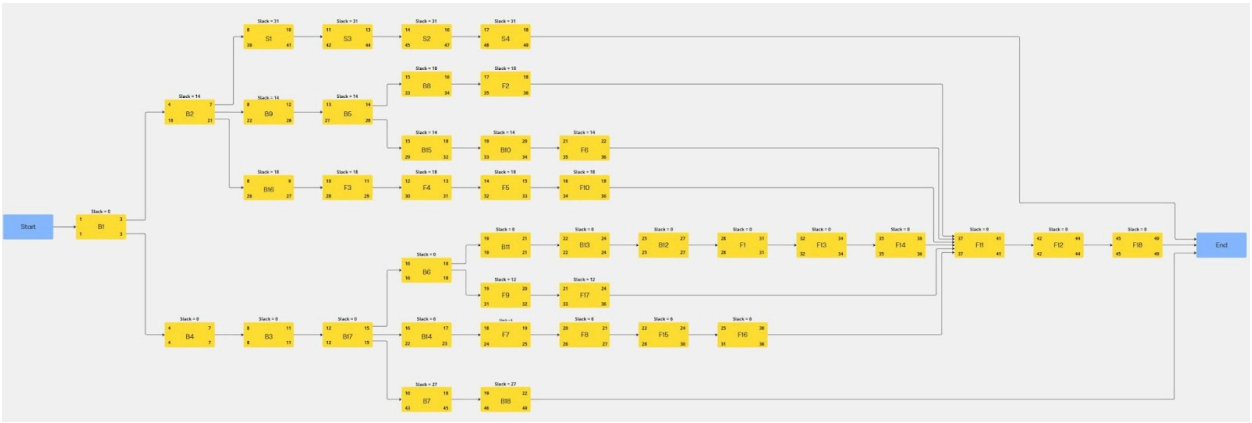
5.3 Gantt Chart

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



5.4 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, S1, S3, S2, S4: 18 Days
B1, B2, B9, B5, B8, F2, F11, F12, F18: 30 Days
B1, B2, B9, B5, B15, B10, F6, F11, F12, F18: 44 Days
B1, B2, B16, F3, F4, F5, F10, F11, F12, F18: 28 Days
B1, B4, B3, B17, B6, B11, B13, B12, F1, F13, F14, F11, F12, F18: 49 Days *Critical Path
B1, B4, B3, B17, B6, F9, F17, F11, F12, F18: 36 Days
B1, B4, B3, B17, B7, B18: 22 Days
B1, B4, B3, B17, B14, F7, F8, F15, F16, F11, F12, F18: 42 Days

6. Configuration Management Strategy

6.1 Versioning Convention

The project will start with its first version at V0.0.1, indicating a very early test of the program. When more functionality is implemented, the second digit will be bumped up and will count up from V0.1.0. Upon the project's completion with all of its features implemented, the

leftmost digit will be bumped up and the project will count from V1.0.0, indicating that the project is fully functional.

6.2 Branching Strategy

The project will have a Master Branch which houses a stable build of the project. This branch will not be altered until a large amount of changes are made, and until said changes are thoroughly tested and proven to be stable for both developers and the end user. While the Master Branch houses the latest stable code, new features will be worked on through an Experimental Branch, which will be altered much more frequently. New changes will be merged with the Experimental Branch upon completion, so that if something goes wrong, the whole project does not suffer as a result. When a large chunk of new features are ready, and have been tested thoroughly, the Master Branch will be updated to reflect the changes in the Experimental Branch.

6.3 Change Management

Change requests will be made through Github's pull requests, and must be reviewed by all members of the team before being added. When a pull request is approved, it will be merged into the Experimental Branch; pull requests will not be made to the Master Branch under any circumstance. These features will be added to the Master Branch later down the line after being tested to ensure their stability.

6.4 Automated Build Process

Github will handle the automated build process, specifically the Github Actions feature offered on the site. This will automate adding the code submitted in the pull requests into the Experimental Branch. As such, new code can be pushed into the Experimental Branch without having the potential to disrupt the Master Branch.

6.5 Merge Strategy

The Experimental Branch will be tested to ensure that new features are working as intended, and are stable. Once these features are confirmed to work reliably without flaws that harm the user experience, they will be merged with the Master Branch. Each merge with the Master Branch would constitute changing the version number, such as from 0.0.1 to 0.0.2, and if the change is large enough, it may be changed from 0.0.x to 0.1.x.

6.6 Documentation and Review:

Each new feature and each new change made to the Experimental Branch should be given an explanation that provides insight on its purpose, and how it impacts the user experience. These changes should be explained with as much detail as possible in order to make it clear how each aspect of the project functions. Each aspect of the code should be explained with enough detail that someone with no prior relation to the project can read it and not only be able to get a sense of what's going on, but to also understand the inner workings of the project in its entirety. In order to meet the goal of having our project be understandable to anyone picking it up, each feature of the code should be explained with as much detail as possible.

7. Quality Assessment Planning

7.1 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.

7.2 Testing Resources

The front-end development team is using Bootstrap, which is an open-source framework that includes languages like CSS, Javascript and HTML. Bootstraps 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. Firebase supports Java, which is our backend language of choice. Firebase also has API integration support which makes this resource very useful for the HofTAPS team.

7.3 Testing Methods

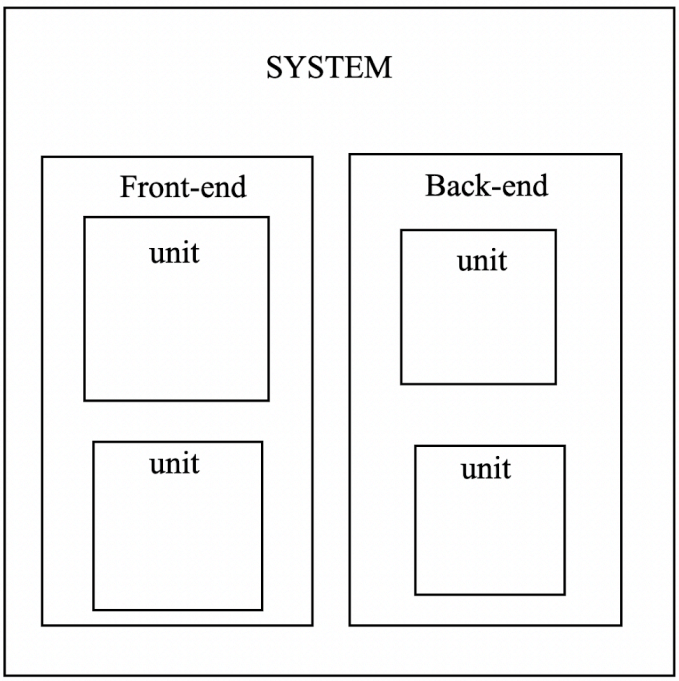
There are several testing methods that will be utilized throughout the duration of this project. We can break these into 5 different components:

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.

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.

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.

System Testing: When the whole system is up and running, testing will be 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 fine, and user interface components are implemented correctly.



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.

7.4 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. Testing in this way guarantees a fair benchmark for acceptable code.

Iit 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.