Software Requirements Specification for an Online Book-Store and Exchanging Application.

Project : Bookbin

nurolmahadi0132@gmail.com

Daffodil Smart City, Ashulia Savar

Version 5.0

1. PROJECT OVERVIEW

An online Book Exchanging Platform. Users have to create an account and log in. Afterwards they can log in to their accounts and display their books for exchanging and put on requests.

- BookBin is a online book exchanging app.
- Users can search though libraries and purchase new books like any other normal online book store. But the main focus of this app is exchanging books between users.
- Users would have to create accounts and can build their own profile. They can display what books they have for exchanging and also what books they are searching for.
- Users can buy, sell, exchange books.
- Users can interact with each other, talk and negotiate with each other while exchanging (like any social media app).
- The app also has E-Book options. Users will have an option to subscribe to a yearly subscription system for the E-Book section. The yearly subscription includes access to all of the E-Books in the E-Book library section. However, users without the subscription will have to buy each E-Book they want to access and download.
- We won't have any physical location to store books. Every exchange would be done through users.
- Selling point of the new books would be libraries that we are partnered with. Thus, when any user puts on a order to buy books, that order would be placed to the libraries that we are partnered with and they will process that order. The users can also request books from the libraries if their stock don't have those certain books.

1.1 INTENDED AUDIENCE

Bookbin is designed for a diverse audience, including:

- Faculty Members.

- Casual Book readers.

- Academic Students.

2. USER FEATURES IN DETAILS:

- Dashboard
    - Home
    - Events
    - Properties
        - On Click A Properties View Project Details
            - Exchange

            - Buy Now
            - Cart

    - Contact
    - Login
- Application System
    - User
        - Login
            - User Id and Password are provide by admin
        - Payment Status
        - Payment History
        - Exchange History
        - Cart
        - Download Receipt
    - Admin
        - Create User
        - Submit User Payment
        - See User Payment Status

- Payment History
- Generated Receipt

3. TECHNOLOGY REQUIREMENTS (APPLICATION AND HARDWARE)

- Framework: JFrame.
- Database: MySQL.
- Design: Standard.
- Coding Architecture: OOP.
- Security: Standard.

4. SPECIFIC REQUIREMENTS

4.1. USER REGISTRATION AND AUTHENTICATION:

- Users can sign up using email, social media accounts.

- Secure authentication mechanisms for user safety.

4.2. USER PROFILES:

- Users can create profiles with profile pictures, bios, and personal preferences.
- Users can customize their account based on their preference of books and genre.
- AI-driven profile overview based on user’s search and preferences.

4.3. INTERACTION AND ENGAGEMENT:

- Comments, and reviews are available for everyone.
- Live chat interactions for negotiations and book quality.

4.4. SEARCH AND DISCOVERY:

- Users can search for specific books, genres, or authors.
- AI-driven recommendations suggest relevant books based on user preferences.

4.5 NOTIFICATIONS:

- Users receive notifications for new books, followers, and interactions.
- Customizable notification settings ensure user control.

5.SYSTEM REQUIREMENTS

- Platform Compatibility : iOS, Android, Windows, and macOS platform compatibility.

- Internet Connection.

Accessibility: The app should conform to accessibility standards, ensuring usability for all users.

6. DATA MANAGEMENT

- User Data : Store user profiles, authentication credentials, and interaction history securely.

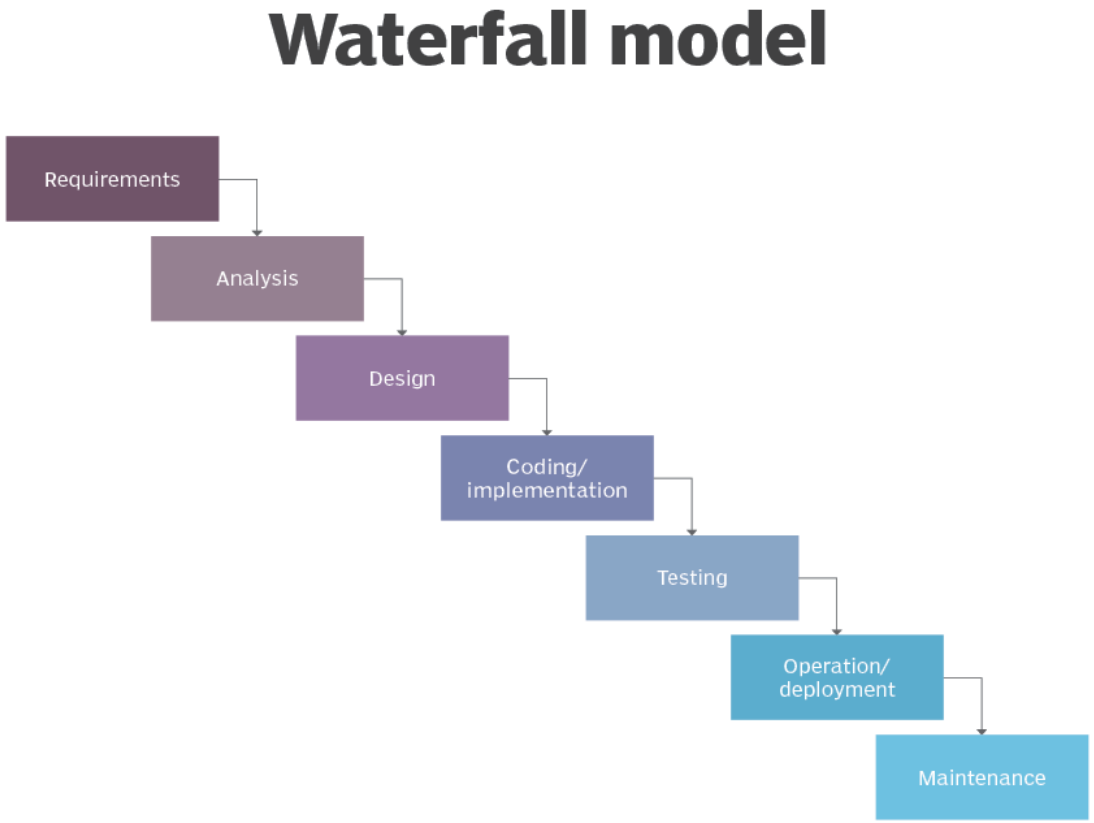
- Interaction Data : Store chat messages, comments, reactions, and analytics data.

7. SDLC

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

7.1 SOFTWARE DEVELOPMENT METHODOLOGIES

7.1.1 WATERFALL MODEL

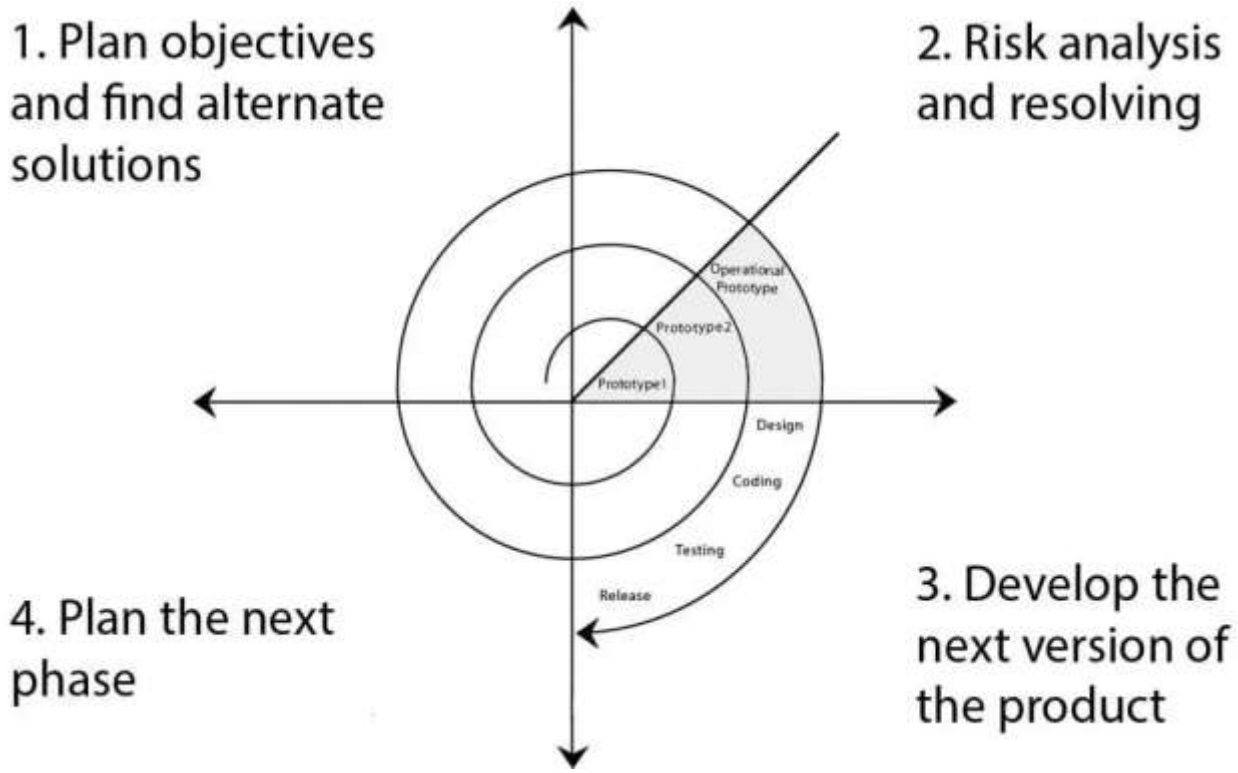


Waterfall model uses a sequential steps for a project, similar to the fall of water from a cliff. This SDLC model divides the life cycle into a set of phases. And these phases can’t be revisited afterwards. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow of the waterfall.

7.1.1.1 PROS AND CONS OF WATERFALL METHODOLOGY

Advantages	Disadvantages
Simple and easy to understand	No working software is produced until late during the life cycle.
Works well for smaller projects where requirements are very well understood.	Cannot accommodate changing requirements
Easy to manage due to the rigidity of the model.	High amounts of risk and uncertainty.
. Each phase has specific deliverables and a review process	Adjusting scope during the life cycle can end a project
Phases are processed and completed one at a time.	Not a good model for complex and object-oriented projects.
Well understood milestones	
Works well for smaller projects where requirements are very well understood.	Poor model for long and ongoing projects.

7.1.2 SPIRAL MODEL

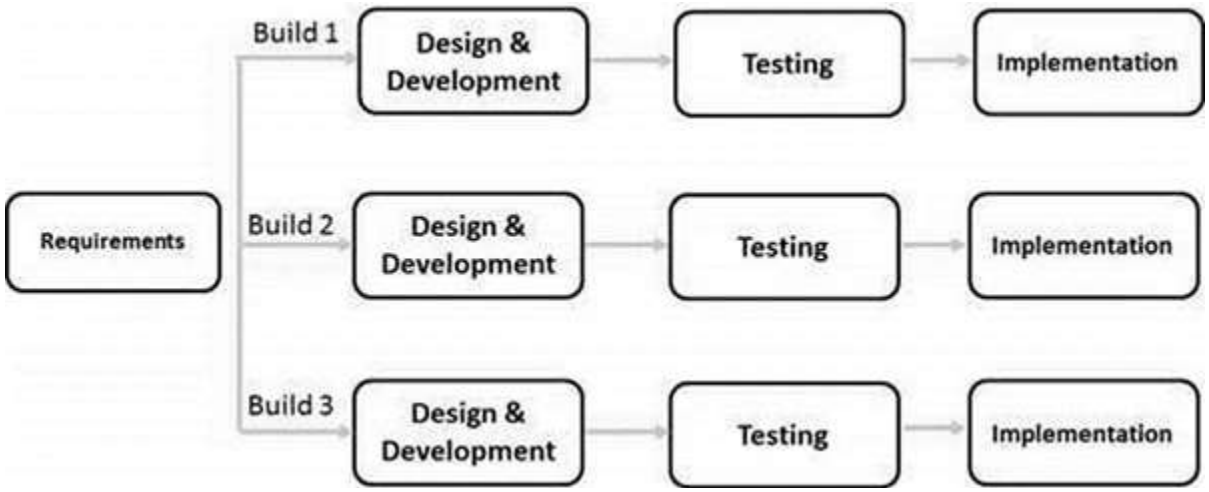


A Spiral Model of Software Development and Enhancement as an iterative and risk-driven model of software development. The entire process of development is divided into four different stages which keep on repeating until the entire project is completed. These stages are determining objectives, evaluating risks, developing the product and planning the next phase. This entire process is represented in a spiral diagram and thus known as the spiral model. The Spiral Model provides support for risk handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process.

7.1.2.1 PROS AND CONS OF SPIRAL METHODOLOGY

Advantages	Disadvantages
Changing requirements can be accommodated.	Management is more complex.
Users see the system early	Process is complex
Allows extensive use of prototypes.	End of the project may not be known early.
Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.	Spiral may go on indefinitely
Requirements can be captured more accurately.	Not suitable for small or low risk projects and could be expensive for small projects.

7.1.3.0 ITERATIVE MODEL



The Iterative model is all about repetition. Instead of starting out with a comprehensive overview of the requirements, development teams build the software piece by piece and identify further requirements as they go along. As a result, each new phase in the Iterative model produces a newer, more-refined version of the software under development.

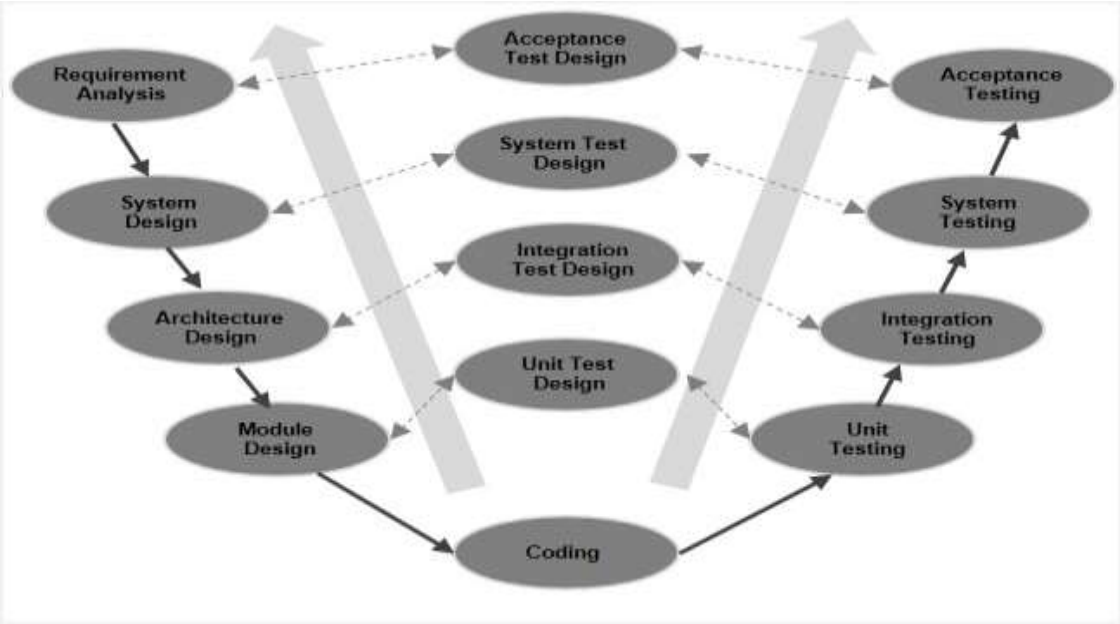
Iterative allows developers and testers to identify functional or design flaws early, and can easily adapt to the ever-changing needs of the client. Like Spiral, Iterative suits larger projects and requires more management and oversight to work well.

7.1.3.1 PROS AND CONS OF ITERATIVE METHODOLOGY

Advantages	Disadvantages
Results are obtained early and periodically.	More resources may be required.
With every increment, operational product is delivered.	Defining increments may require definition of the complete system.

Parallel development can be planned.	Although cost of change is lesser, but it is not very suitable for changing requirements.
Testing and debugging during smaller iteration is easy.	System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
With every increment, operational product is delivered.	More management attention is required.
Results are obtained early and periodically.	

7.1.4.0 V-MODEL



Similar to the waterfall methodology where testing is done at the end of a project, with the V-model, testing happens at every stage of development. The next stage of the V-model starts only when the previous stage is entirely finished.As part of the V-Model, a software tester has to verify if the requirements of a specific development phase are met. They also have to validate that the system meets the needs of the user, customer or other stakeholders, which includes both verifications and validations.

7.1.4.1 PROS AND CONS OF V-MODEL METHODOLOGY

Advantages	Disadvantages
This is a highly-disciplined model and Phases are completed one at a time.	High risk and uncertainty.
Works well for smaller projects where requirements are very well understood.	Not a good model for complex and object-oriented projects.
Simple and easy to understand and use.	Poor model for long and ongoing projects.
Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.	Not suitable for the projects where requirements are at a moderate to high risk of changing.

7.1.5.0 AGILE MODEL

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations

7.1.5.1 ADVANTAGES OF AGILE METHODOLOGY

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

7.1.6.0 RAD (RAPID APPLICATION DEVELOPMENT) MODEL

Rapid application development is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

In the RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process.

RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.

8.0 AGILE MODEL FOR BOOKBIN APPLICATION DEVELOPMENT

Agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.

Customer Interaction is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location

8.1 WHY AGILE IS SUITABLE.

Agile is based on the adaptive software development methods, whereas the traditional SDLC models like the waterfall model is based on a predictive approach. Predictive teams in the traditional SDLC models usually work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle.

Predictive methods entirely depend on the requirement analysis and planning done in the beginning of cycle. Any changes to be incorporated go through a strict change control management and prioritization.

Agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.

Customer Interaction is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location.

8.3 AGILE VS LEAN

- Frequent Change and Uncertainty: Agile is particularly effective in environments where requirements change frequently, and there is a need for flexibility and rapid adaptation to customer feedback.

- Customer Collaboration: Having direct access to customers or end-users who can provide continuous feedback and you want to involve them closely in the development process, Agile's customer-centric approach is beneficial.

- Iterative and Incremental Delivery: Delivering incremental value in short iterations aligns with your project's goals and stakeholder expectations, Agile is a good choice.

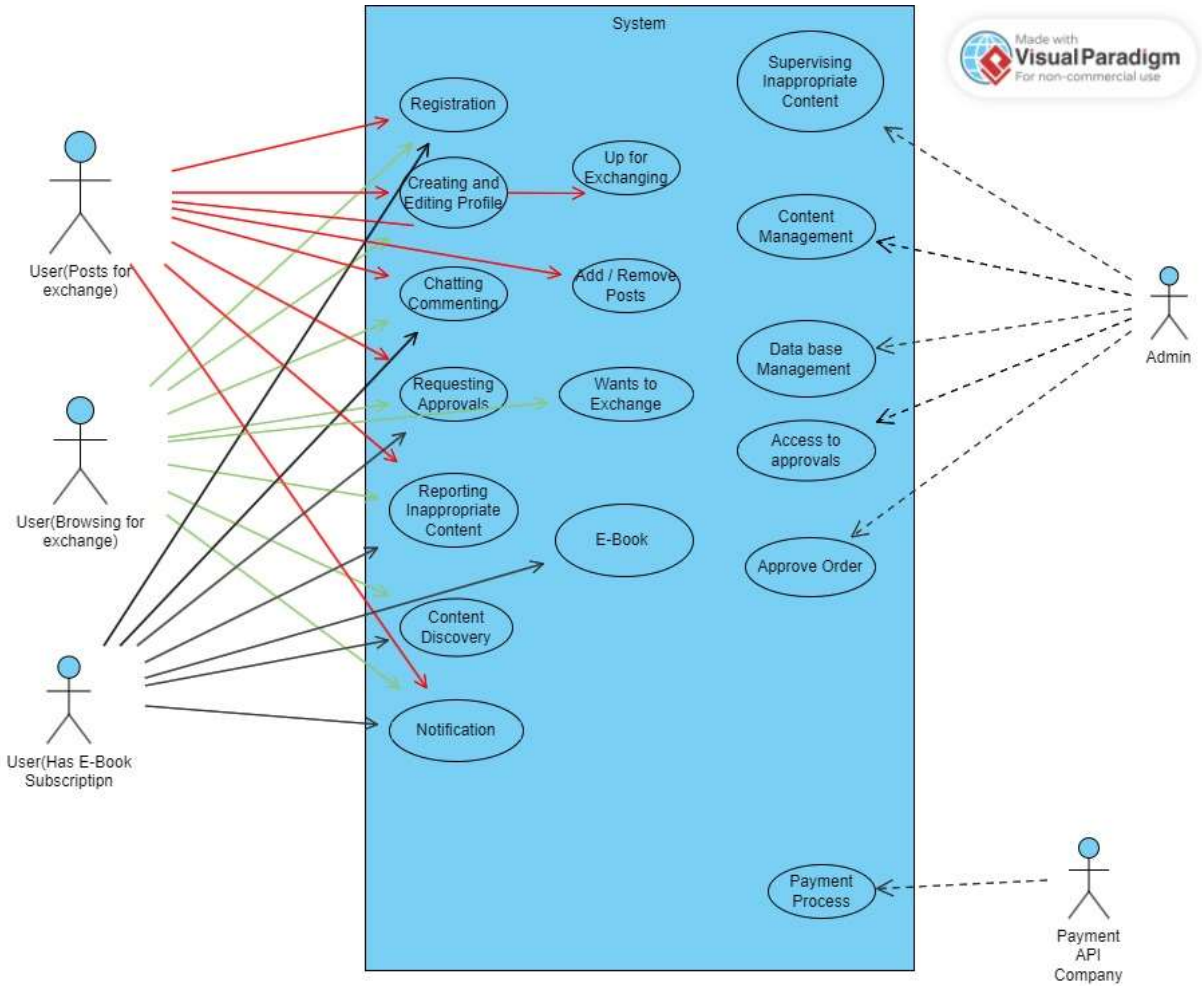
- Cross Functional Team : Agile teams are typically cross-functional, with members responsible for various aspects of the app.

- Adaptability : Well suited to projects with evolving requirements. Allows for flexibility in responding to changing user needs.

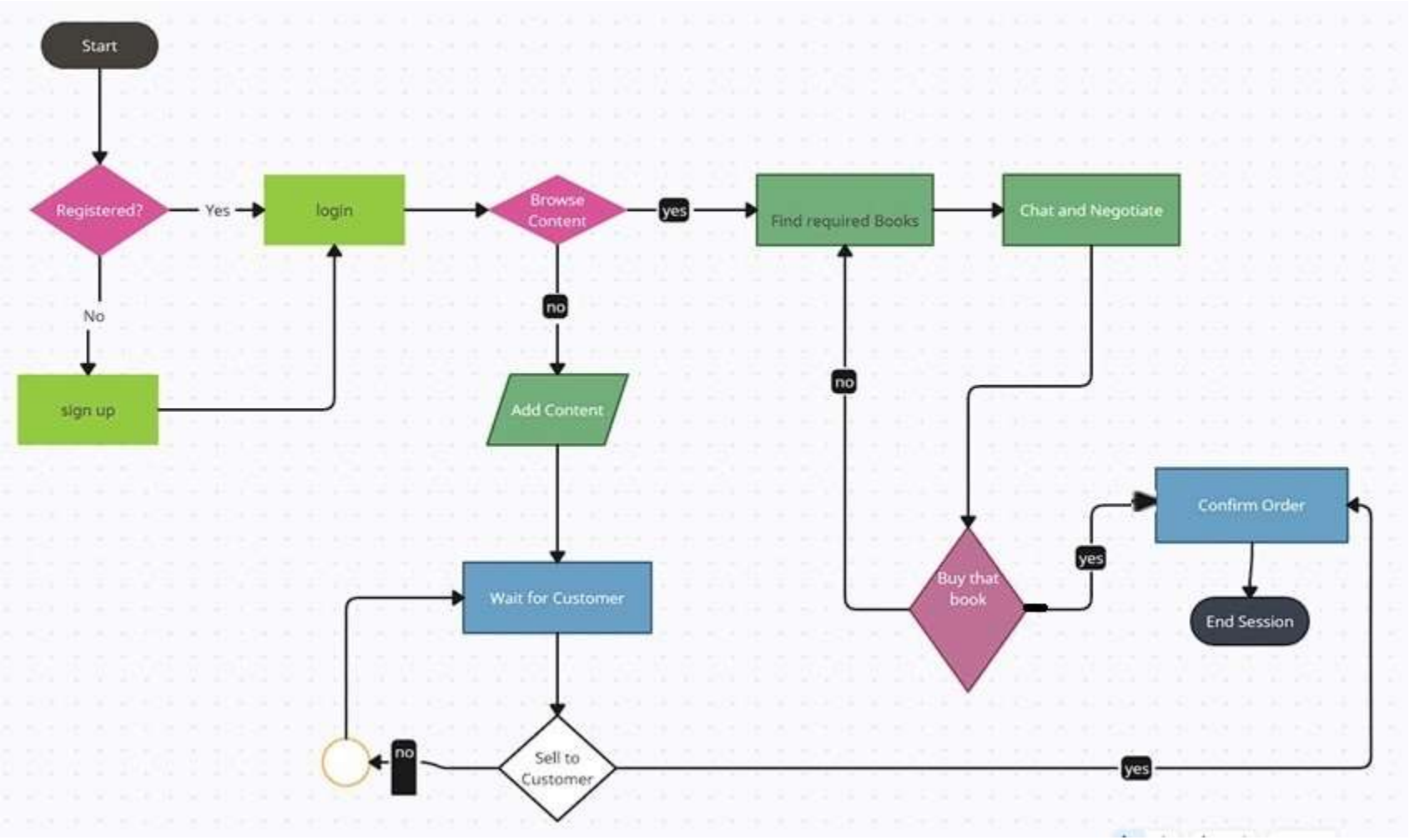
- Provides a dynamic environment with evolving user needs .

9. MIND MAP

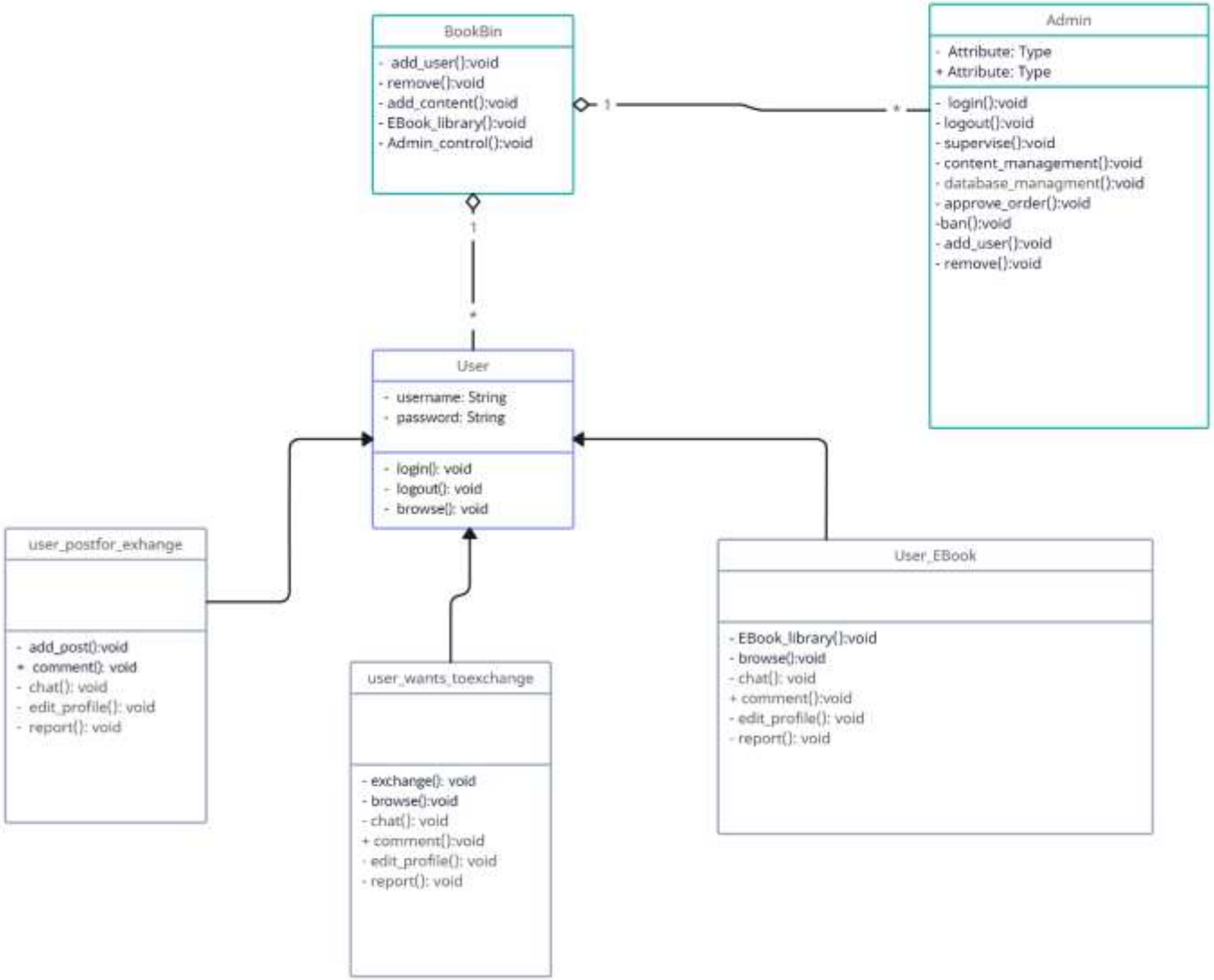
9.1 USE CASE DIAGRAM



9.2 FLOWCHART



9.3 UML DIAGRAM



10. TESTING

10.1.0 BOX APPROACH

10.1.1 BLACK BOX TESTING

It focuses on the functionality and user experience without requiring knowledge of the internal structure of the application. Testers simulate user actions to validate the features, usability, and overall functionality of the library. Black box testing is crucial for ensuring that the user-facing aspects of the library, such as browsing books, adding items to the cart, checkout process, search functionality, and account management, work as intended.

10.1.2 WHITE BOX TESTING

This approach dives into the internal structure of the application, analyzing the code paths, logic, and system architecture. It ensures that the underlying code operates efficiently, handles edge cases, and is free from errors. White box testing can be particularly useful for critical functionalities like payment processing, database interactions, security features, and ensuring proper error handling.

10.1.3 GREY BOX TESTING

Combines elements of both black box and white box testing, offering insights into the application's functionality while also understanding some internal workings. Testers might have limited knowledge of the codebase but enough to create targeted test scenarios.

10.1.4 CHOOSING GREY BOX APPROACH

We would move forward with a combination of the above testing approaches. Black box testing will be performed to ensure that the user experience and functional aspects are working smoothly. Then, to complement this with white box testing we would delve into critical components and security measures .And finally, Grey box testing can fill in gaps where a partial understanding of internal workings is beneficial for specific tests or scenarios. We are moving forward with Grey Box Testing because the testing procedure involved both Black Box and White Box testing

10.2 TESTING LEVEL

Testing level will include all three levels

- Unit testing- Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors
- Integration testing-Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.
- Acceptance testing-Acceptance testing commonly includes the following four types:

  -User acceptance testing (UAT)
  -Operational acceptance testing (OAT)
  -Contractual and regulatory acceptance testing
  -Alpha and beta testing

10.3 TEST TYPES AND TECHNIQUES

We propose to implement the following Test types and techniques in this system

- **Regression testing-**Regression testing involves retesting the previously tested parts of the software to verify that existing functionalities still work correctly after a change has been made, be it in the form of bug fixes, enhancements, or new features.
- **Acceptance testing-**Acceptance testing performed by the customer, often in their lab environment on their own hardware, is known as user acceptance testing (UAT). Acceptance testing may be performed as part of the hand-off process between any two phases of development
- **Alpha testing-**Alpha testing is a form of acceptance testing where a software application is tested in a real environment by end-users or internal employees, usually before the software is made available to the public
- **Beta testing-**Beta Testing is done to evaluate the software's performance, functionality, usability, and compatibility in a real-world environment under diverse user scenarios.

- **Continuous testing-** Continuous testing ensures code changes are continuously and automatically tested, providing quick feedback to developers about potential issues or defects introduced by new code.
- **Software performance testing-** Performance testing is generally executed to determine how a system or sub-system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.
- **Security testing-** Security testing is essential for software that processes confidential data to prevent system intrusion by hackers.
- **A/B testing-** A/B testing is a method of running a controlled experiment to determine if a proposed change is more effective than the current approach

Other various Testing Types are also given below

- **Output comparison testing-** This cannot detect failures automatically and instead requires that a human evaluate the output for inconsistencies.
- **Conformance testing or type testing-** In software testing, conformance testing verifies that a product performs according to its specified standards. Compilers, for instance, are extensively tested to determine whether they meet the recognized standard for that language.
- **Property testing-** Property testing is a testing technique where, instead of asserting that specific inputs produce specific expected outputs, the practitioner randomly generates many inputs, runs the program on all of them.

10.4 TESTING PROCESS

We choose to implement Agile or XP development model in this system. Agile Testing process is iterative and flexible that emphasize adaptability, collaboration, and customer satisfaction. It involves short development cycles called sprints. Testing occurs throughout these cycles, not just at the end. It is based on user stories and their acceptance criteria. Testers and developers collaborate to define tests that ensure each user story meets the specified criteria. The tests are expected to fail initially. Each failing test is followed by writing just enough code to make it pass. This means the test suites are continuously updated as new failure conditions and corner cases are discovered, and they are integrated with any regression tests that are developed. Unit tests are maintained along with the rest

of the software source code and generally integrated into the build process. Testing evolves incrementally with the software. Each iteration adds new functionalities and features, along with corresponding tests.

10.5 TESTING TOOLS

The test will be done using the following tools

- Automated testing- It is a continuous integration software will run tests automatically every time code is checked into a version control system
- Instruction set simulator
- Hypervisor
- Program animation
- Code coverage
- Automated functional Graphical User Interface
- Benchmarks

10.6 MEASUREMENT IN SOFTWARE TESTING (HIERARCHY OF TESTING DIFFICULTY)

Quality measures include such topics as correctness, completeness, security, capability, reliability, efficiency, portability, maintainability, compatibility, and usability.

Determining the difficulty level of testing can vary based on various factors, including the type of testing being conducted, the complexity of the system, the expertise of the testers, and the tools and resources available

a hierarchy of testing difficulty has been proposed.It includes the following testability classes:

- **Class I:** Finite Complete Test Suite: This class represents a scenario where a finite test suite exists that can completely test a system. It covers all possible scenarios and states, ensuring every aspect of the system is tested.

- **Class II:** Finite Partial Distinguishing Rate: In this class, while the test suite is finite, it may not cover every possible scenario comprehensively. However, it has the capability to distinguish between correct and incorrect systems to a certain extent even with incomplete testing.
- **Class III:** Countable Complete Test Suite: Here, there exists a test suite that's countable (could be infinite but countable) and is capable of completely testing the system. Despite being countable, it covers all possible scenarios or states.
- **Class IV:** Complete Test Suite: This class represents the existence of a complete test suite that thoroughly tests the system, covering all aspects, scenarios, and states. It ensures that every possible behavior of the system is tested.
- **Class V:** All Cases : This class implies that the test suite covers every conceivable case or scenario. It represents the highest level of completeness in testing, encompassing all possible situations, inputs, and states.

11 TEST PLAN

- Application will be tested by PHPUnit.
- Application also tested By Codeception.

The testing process shall be as follows:

- 1. Introduction: Overview of the test plan, its purpose, and intended audience.
- 2. Scope and Objectives: Scope of testing: What functionalities and modules will be covered in testing (e.g., user registration, book listing, exchanges, user profiles). Objectives: Clear goals and expectations from the testing effort (e.g., ensuring functionality, usability, security).
- 3. Testing Approach: Methodologies and techniques used for testing (e.g., manual testing, automated testing, exploratory testing). Types of testing to be performed (e.g., functional, non-functional, regression).
- 4. Test Environment: Description of the testing environment (e.g., hardware, software, browsers, devices). Configuration details for servers, databases, and other necessary components.
- 5. Test Deliverables: List of documents, reports, and artifacts to be produced during testing (e.g., test cases, test reports, defect logs).

- 6. Test Cases: Detailed test cases covering various functionalities of the platform (e.g., adding books, searching, initiating exchanges, user authentication).
  Each test case includes steps, expected results, pass/fail criteria, and necessary preconditions.
- 7. Test Execution Schedule: Timeline and schedule for executing test cases and testing activities. Milestones, phases, and their durations.
- 8. Resource Planning: Roles and responsibilities of team members involved in testing (e.g., testers, QA leads). Allocation of resources (human and infrastructure) for testing activities.
- 9. Risk Analysis and Mitigation: Identification of potential risks associated with testing and the platform. Mitigation strategies for managing and minimizing identified risks.
- 10. Reporting and Metrics: Reporting mechanisms for documenting and communicating test results. Key metrics to be measured during testing (e.g., test coverage, defect density, pass/fail rates).
- Entry criteria: Conditions that must be met before testing starts.
- Exit criteria: Conditions for determining completion of testing activities.

12. TEST CASE

Test Case Title: Adding a Book to User's Inventory

- Objective: To verify that a user can successfully add a book to their inventory on the online book exchange platform.

- Preconditions: (i) User is logged in with valid credentials.
  (ii) User has navigated to their profile or inventory management section.
- Test Steps:

- Step 1 - Accessing Inventory:
  Action: Log in to the online book exchange platform.
  Expected Result: User successfully lands on their profile/dashboard.
- Step 2 - Adding a Book:
  Action: Click on the "Add Book" or similar button/icon to add a book.

Expected Result: The system presents a form or interface to add book details.

- Step 3 - Entering Book Details:
  Action: Fill in the required details (Title, Author, ISBN, Condition, etc.) for the book to be added.
  Expected Result: The user successfully inputs all necessary book information without errors.
- Step 4 - Uploading Book Image:
  Action: Upload an image or cover picture of the book (if applicable).
  Expected Result: The system accepts the image upload and displays the book cover successfully.
- Step 5 - Saving the Book:
  Action: Click on the "Save" or "Add Book" button after entering all details.
  Expected Result: The book is successfully added to the user's inventory without any system errors or delays.
- Step 6 - Verifying Book in Inventory:
  Action: Navigate to the inventory section or refresh the page.
  Expected Result: The added book is displayed in the user's inventory list with the correct details and uploaded image.
- Pass/Fail Criteria:
  Pass: The book is added successfully to the user's inventory, and all entered details are accurately displayed.
  Fail: Any step results in an error, system crash, or the book details not being correctly saved/displayed in the inventory.

13. LANGUAGE TABLE

Component	**Languages**	**Functionality**	**Advantages**
Frontend	Javascript, Html, CSS	Creating User Interface	Create interactive and responsive web applications. With the evolution of frameworks and libraries, JavaScript has become even more powerful, enabling developers to build complex

			and feature-rich applications for the web.
Backend	Node.js		allows use JavaScript on the server-side, which can streamline development when already using JavaScript for the front end. Combined with frameworks like Express.js, it's powerful for handling asynchronous operations and creating RESTful APIs.
Database	MySQL	Data Organization (data insertion, deletion , storing)	Reliable and efficient database solution for a wide range of applications, from small-scale projects to enterprise-level systems.
Security	HTTPS	Provide security of user data. Ensures sensitive information like login credentials ,payment details and personal information remains confidential.	Uses Encryption to secure transmitted data between user’s browser and the website’s server. Encrypts data to prevent unauthorized access. Ensures sensitive information like login credentials ,payment details and personal information remains confidential.

14.MILESTONES AND REPORTING

Milestone	Tasks	Reporting	Time
Analysis		Design Submission	3 days
Requirements Collection	Data Submission		7 days
Development Stage		Work Review	30 days
Testing Stage	Testing before Deployment		10 days
Deployment Stage	Server Deployment Process	Review Final Work	5 days
Delivery		Activation of Live Server	5 days

15. DEPLOYMENT

The Application would be based on the requirements given by your company. During the development stage we would focus on researching and developing the project. The client would have to pay according to their needs and necessities.

16. PRICING

Our fee for seeing the project through from start to completion will be Twenty Thousand Taka only (20000Tk).

17. PAYMENT TERMS

We propose the following payment terms:

10% (10%)

Paid on acceptance of this proposal.

40% (50%)

Paid on signing of our Application development agreement.

25% (75%)

Paid at 70% Application Demonstration.

25% (100%)

Paid at completion the Application.

18. SUPPORT

24/7 support system.

19. CONTACT

Phone:

+8801721716297

MD Nurol Amin

Email

nurolmahadi0132@gmail.com

Website

www.XYZ.com

By post

Room-E_517, Younus Khan Scholars Garden, Old Building

Daffodil Smart City,

Ashulia Savar, Dhaka

We look forward to hearing from you soon!

Agreement Signed By:

……

Client Signature

NUR ALVI

Managing Director

ABC

……
…

Authority Signature

MD Nurol Amin

Managing Director

XYZ