K. J. Somaiya College of Engineering, Mumbai-77

(A constituent college of Somaiya Vidyavihar University)

Batch : A2 **Roll No. : 16010121045**

Experiment / assignment / tutorial No. 1

TITLE: Requirement **S**pecification Document

AIM: To learn and understand the way of analysing the gathered information in the previous phase for the development process and prepare requirement specification document. A concept of software engineering.

Expected Course outcome of Experiment:

Process of gathering requirements and converting them into specifications.
Document created will be used by both, the customer and the developer, to understand WHAT is going to be developed.

Books/ Journals/ Websites referred:

1. Roger Pressman, Software Engineering: A practitioners Approach, McGraw Hill, 2010, 6th edition
2. Ian Somerville, Software Engineering, Addison Wesley, 2011, 9th edition
3 http://en.wikipedia.org/wiki/Software_requirements_specification

Pre Lab/ Prior Concepts:

Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. It is an early stage in the more general activity of requirements engineering which encompasses all activities concerned with eliciting, analyzing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Conceptually, requirements analysis includes three types of activities:

- **Eliciting requirements**: the task of identifying the various types of requirements from various sources including project documentation, (e.g. the project charter or definition), business process documentation, and stakeholder interviews. This is sometimes also called requirements gathering.
- **Analysing requirements**: determining whether the stated requirements are clear, complete, consistent and unambiguous, and resolving any apparent conflicts.
- Recording requirements: Requirements may be documented in various forms, usually including a summary list and may include natural-language documents, use cases or process specifications.

New systems change the environment and relationships between people, so it is important to identify all the stakeholders, taken into account all their needs and ensure they understand the implications of the new systems. Analysts can employ several techniques to elicit the requirements from the customer. These may include the development of scenarios, the identification of use cases, the use of workplace observation or ethnography, holding interviews, or focus groups (more aptly named in this context as requirements workshops, or requirements review sessions) and creating requirements lists. Prototyping may be used to develop an example system that can be demonstrated to stakeholders. Where necessary, the analyst will employ a combination of these methods to establish the exact requirements of the stakeholders, so that a system that meets the business needs is produced

Different types of Requirements

- Functional requirements
- Usability requirements
- Reliability requirements
- Performance requirements
- Security requirements

A typical SRS document template is shared subsequently. This document acts as a reference and will be used by both, the customer (for whom the software system is to be developed), and the organization which develops the solution.Typically, prepared by the development organization at the early stage of development by the professionals after interacting with the customer.

K. J. Somaiya College of Engineering, Mumbai-77

(A constituent college of Somaiya Vidyavihar University)

Software Requirements Specification for:

CodeCat – Online Complier

Version 1.0

Prepared by Pargat Singh

K.J Somaiya College of Engineering

23rd August 23, 2023

Table of Contents

Introduction

Purpose

The primary purpose of developing CodeCat is to create an all-inclusive online coding platform that addresses the challenges faced by programmers and coding enthusiasts. This platform aims to provide an intuitive and user-friendly interface for writing, compiling, and executing code in multiple programming languages. CodeCat seeks to enhance the coding experience by offering advanced features such as syntax highlighting, error-tolerant terminal screens, customizable input/output support, and both dark and light modes. The platform's development is motivated by the need to improve upon existing online compilers and coding tools, particularly in educational settings. CodeCat aims to empower users to efficiently code in their language of choice, overcome coding challenges, and ultimately enhance their coding skills. By providing a versatile and accessible coding environment, CodeCat intends to support learning, collaboration, and productivity for programmers at various skill levels.

Product Scope

CodeCat is an online coding platform designed to empower programmers by providing an intuitive interface for coding in multiple languages. It offers a suite of advanced features, including syntax highlighting, real-time code execution, and error-tolerant terminals. CodeCat's scope encompasses comprehensive language support, efficient coding tools, and user-friendly design, all geared towards enhancing the coding experience and promoting learning. With a focus on accessibility, customization, and collaboration, CodeCat aims to become a go-to platform for coding education, skill development, and project collaboration.

References

- Figma Design for CodeCat UI/UX and wireframing
- Judge0 API documentation for real-time code execution - https://ce.judge0.com/
- GitHub repository for CodeCat development - https://github.com/Pargat-Dhanjal/Mini-Project.git
- Microsoft Azure - https://azure.com/
- Monaco Editor - https://microsoft.github.io/monaco-editor/

Overall Description

Product Perspective

CodeCat is a standalone, self-contained online coding platform developed to meet the needs of programmers and coding enthusiasts. It is not a replacement for any existing system but rather a new and innovative solution to address the challenges faced by users while coding online. CodeCat operates as an independent platform with its own set of features, functionalities, and user interfaces. Majorly depends upon various Open source libraries maintained my the developer community.

Product Functions

- Multi-language support for coding in various programming languages.
- Syntax highlighting to enhance code readability.
- Real-time code execution and compilation for efficient testing.
- Error-tolerant terminal screen for quick identification of coding issues.
- Customizable input/output support for comprehensive testing.
- Dark and light mode options for personalized coding environments.
- User authentication for secure access to projects and features.
- Progressive web application design for cross-device accessibility.

Operating Environment

CodeCat operates in a web-based environment. Users can access the platform through modern web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge. The platform is compatible with various operating systems, including Windows, macOS, and Linux. No additional software installation is required, as CodeCat is a progressive web application that functions seamlessly on desktops, laptops, tablets, and smartphones.

Design and Implementation Constraints

- CodeCat development adheres to modern web technologies and best practices.
- The platform is designed to be responsive, ensuring optimal user experience on different screen sizes and devices.
- Code execution and compilation utilize the Judge0 API to ensure real-time processing.
- The chosen tech stack includes React.js for the front-end, Firebase for the database, and Azure for hosting.
- It also uses firebase auth to authenticate users via Google or Github, following a simplistic and modern authentication standard.

User Documentation

User documentation for CodeCat will include online help guides, tutorials, and a user manual. These resources will guide users through platform features, functionalities, and how to effectively utilize the coding environment. The documentation will be provided in digital format, accessible within the platform and downloadable for offline use. It focuses on a user first approach, the entire application was design with user flow in mind to ease the convenience of using a compiler.

Assumptions and Dependencies

- CodeCat assumes a stable and reliable internet connection for users to access the platform.
- Dependencies include the proper functioning of third-party services such as Judge0 for real-time code execution.
- The project assumes users have a basic understanding of coding concepts and programming languages.
- It uses Microsoft Azure platform to deploy the self-hosted judge0 api.
- It also uses Vercel hosting services to host the frontend of the application.

External Interface Requirements

User Interfaces

CodeCat offers an intuitive and user-friendly user interface designed to enhance the coding experience. While detailed interface design specifications are documented separately, here's an overview of the logical characteristics:

- **Home Page:** Provides quick access to create, manage, and view coding projects along with showing stats of your GitHub project.
- **Code Editor:** Offers syntax highlighting, code completion, and formatting tools.
- **Compile & Execute:** Users can run code in real-time, with error feedback and output displayed.
- **Terminal Screen:** Displays error messages and debugging information for rapid issue resolution.
- **Input/Output Windows:** Multi-line windows for testing various input and output scenarios.
- **Dark/Light Mode:** Users can choose between two modes for personalized viewing.
- **Authentication:** Secure login using GitHub and Google credentials for personalized experience.

Hardware Interfaces

CodeCat operates as a web-based platform, requiring no direct hardware interfaces. It functions on a variety of devices including desktops, laptops, tablets, and smartphones. The most important hardware requirement of all is an active internet connection on the device of choice.

Software Interfaces

- **React.js Frontend:** The user interface is built using React.js, incorporating various components for a seamless experience.
- **Firebase Backend:** CodeCat interfaces with Firebase for user authentication, project storage, and data management.
- **Judge0 API:** Real-time code execution is facilitated through the Judge0 API, enabling compilation and output display.
- **Web Browsers:** CodeCat is accessible through modern web browsers such as Chrome, Firefox, and Edge.
- **Microsoft Azure:** Azure is used to host the sdk of judge0 api on a personal cloud server as a means to serve the api.

Communications Interfaces

- **HTTP/HTTPS Protocols:** CodeCat communicates with users' browsers using standard HTTP/HTTPS protocols.
- **Judge0 API:** Real-time code execution communication is established via HTTP requests to the Judge0 API.
- **GitHub OAuth:** User authentication and login are facilitated through GitHub OAuth services.

System Features

Feature 1: Multi-Language Support 4.1.1 Description and Priority Enables users to code in various programming languages. High priority as it's a core offering of the platform, essential for user flexibility and learning.

4.1.2 Stimulus/Response Sequences User selects the desired programming language in the code editor. System responds by enabling syntax highlighting and relevant autocomplete suggestions.

4.1.3 Functional Requirements REQ-1: The platform must provide a dropdown list of supported languages. REQ-2: When a language is selected, the code editor should adjust syntax highlighting accordingly. REQ-3: The code editor must provide autocomplete suggestions specific to the selected language. REQ-4: An error message should be displayed if an unsupported language is chosen.

Feature 2: Real-Time Code Execution 4.1.1 Description and Priority Allows users to compile and run their code instantly. High priority for immediate code testing and debugging.

4.1.2 Stimulus/Response Sequences User clicks the "Run" button in the code editor. System compiles and executes the code in the background. Output or error messages are displayed in the terminal screen.

4.1.3 Functional Requirements REQ-5: The platform must include a "Run" button in the code editor. REQ-6: The system should trigger code compilation and execution upon button click. REQ-7: If compilation fails, relevant error messages must be displayed in

the terminal screen. REQ-8: Successful compilation should show the code's output in the terminal screen.

Feature 3: Error-Tolerant Terminal Screen 4.1.1 Description and Priority Displays error messages and debugging information for efficient issue resolution. Medium priority for smooth coding experience.

4.1.2 Stimulus/Response Sequences User runs code that contains errors. System detects errors during compilation. Error messages and line numbers are displayed in the terminal screen.

4.1.3 Functional Requirements REQ-9: The terminal screen must display relevant error messages with line numbers. REQ-10: If there are multiple errors, the terminal screen should list all errors. REQ-11: The system should differentiate between syntax errors and runtime errors. REQ-12: Users should have the option to clear the terminal screen.

Feature 4: Dark/Light Mode Support 4.1.1 Description and Priority Allows users to switch between dark and light modes based on preference. Medium priority for personalized visual experience.

4.1.2 Stimulus/Response Sequences User toggles between dark and light mode in settings. System adjusts the color scheme of the interface accordingly.

4.1.3 Functional Requirements REQ-13: The platform must include dark and light mode options in settings. REQ-14: User preferences for mode should be saved for future sessions. REQ-15: Interface colors and contrasts should change based on the selected mode. REQ-16: The mode toggle should be easily accessible from any page.

Feature 5: User Authentication 4.1.1 Description and Priority Enables secure login using GitHub credentials for personalized experience. High priority for ensuring user data security.

4.1.2 Stimulus/Response Sequences User clicks the "Login with GitHub" button. System redirects the user to GitHub's authentication page. After successful authentication, user is granted access to their account.

4.1.3 Functional Requirements REQ-17: The platform must include a "Login with GitHub" button. REQ-18: Users should be redirected to GitHub's authentication page upon clicking. REQ-19: Upon successful authentication, the system must grant access to

the user's account. REQ-20: Failed authentication attempts should be handled with appropriate error messages.

Feature 6: Input/Output Testing 4.1.1 Description and Priority Allows users to test various input/output scenarios for comprehensive code evaluation. Medium priority for enhancing code functionality.

4.1.2 Stimulus/Response Sequences User inputs test cases in the input window. System runs the code using the provided input. Output or error messages are displayed in the terminal screen.

4.1.3 Functional Requirements REQ-21: The platform must provide multi-line input windows for users to input test cases. REQ-22: Users should be able to submit multiple test cases for one code execution. REQ-23: The system should execute the code using each test case's input. REQ-24: The output or error messages for each test case should be displayed in the terminal.

Feature 7: Project Management 4.1.1 Description and Priority Enables users to create, manage, and save coding projects. Medium priority for organized project handling.

4.1.2 Stimulus/Response Sequences User creates a new project and names it. System creates a new project file and interface for coding.

4.1.3 Functional Requirements REQ-25: The platform must provide an option to create new projects. REQ-26: Users should be able to name and save their projects. REQ-27: Each project should have a dedicated code editor interface. REQ-28: Projects should be easily accessible from the user's dashboard.

Feature 8: Progressive Web App (PWA) 4.1.1 Description and Priority Ensures CodeCat functions as a web app on various devices. Medium priority for cross-device compatibility.

4.1.2 Stimulus/Response Sequences User accesses CodeCat via a mobile device's browser. System loads the platform with a responsive design.

4.1.3 Functional Requirements REQ-29: The platform must be accessible through web browsers on desktops and mobile devices. REQ-30: CodeCat's user interface should adapt to different screen sizes. REQ-31: The platform should have a "Add to Home Screen" option on mobile devices.

Other Nonfunctional Requirements

Performance Requirements

- The platform should respond to user actions, such as code execution, within 2 seconds to ensure a smooth user experience.
- Concurrent users (up to 100) should experience minimal slowdown in platform responsiveness.
- The application must be web responsive for all platforms and should provide the same user experience.
- A seemless user authentication is a must for the whole system.

Safety Requirements

- **Code Safety:** The platform must prevent the execution of code containing malicious elements or posing security risks to users' devices. This includes disallowing harmful system calls or unauthorized access to sensitive resources.
- **Data Encryption:** User data, including GitHub or Google credentials and personal information, must be encrypted using industry-standard encryption techniques while in transit and storage. Firebase's built-in encryption features will be leveraged for secure data handling.
- **Authentication Security:** To ensure secure access, user authentication will be handled through OAuth using GitHub credentials. This prevents unauthorized access attempts and enhances overall platform security.

Security Requirements

- User data, including GitHub credentials, must be securely encrypted during transmission and storage.
- User authentication should be robust, requiring valid GitHub credentials to access the platform.
- The platform should adhere to industry best practices for security, including regular security audits and updates.
- The platform also provides a method to sign in using the email, password combination. The passwords stored on the database are hashed following the industry standard of SHA-256 hashing algorithm to ensure the privacy of the users.

Software Quality Attributes

- Usability: The platform should be user-friendly, allowing users of varying skill levels to code efficiently.
- Maintainability: The codebase should be well-organized and documented for ease of maintenance.
- Portability: CodeCat should function seamlessly on different web browsers and devices.

- Reliability: The platform should have a high uptime percentage to ensure consistent availability.

Business Rules

- **Authenticated Project Access:** Only users with valid GitHub credentials and successful authentication can create, save, and access coding projects on the platform. This ensures that users' projects are associated with their unique identities and provides a personalized coding experience.

- **Code of Conduct:** Users are required to adhere to the platform's acceptable use policies. This includes refraining from executing harmful or malicious code that could compromise the platform's security or impact the experience of other users. Violations of the code of conduct may result in account suspension or termination to maintain a safe and collaborative coding environment.

- **Responsible Code Sharing:** Users are encouraged to share their code and collaborate with others, but they are also expected to respect intellectual property rights and give appropriate credit to others' work. Plagiarism and unauthorized sharing of copyrighted code are strictly prohibited.

- **Feedback and Collaboration:** Users are encouraged to provide constructive feedback to enhance the platform's functionality and user experience. Collaboration and mutual respect within the platform's community are encouraged, fostering a supportive learning environment.

- **Continuous Learning:** CodeCat aims to provide a platform for continuous learning and skill development. Users are expected to engage in coding activities that contribute to their personal growth as programmers and adhere to ethical coding practices.

Other Requirements

- The platform should be designed to handle increased traffic during peak usage times, such as exam periods.
- Internationalization: The interface should be designed to support multiple languages in the future.
- Legal Requirements: The platform should comply with relevant data protection and privacy regulations.
- Reuse Objectives: Any external libraries or tools used should be properly attributed and comply with their respective licenses.

Appendix A: Glossary

- **CodeCat:** The online coding platform being developed, offering a user-friendly interface for coding, compiling, and executing code in various programming languages.
- **GitHub:** A widely-used web-based platform for version control and collaborative software development, often used by developers to host and manage code repositories.
- **OAuth:** Open Authorization, a secure protocol that allows third-party applications limited access to a user's account without exposing their credentials. Used for secure user authentication in CodeCat.
- **Firebase:** A cloud-based platform developed by Google that offers various tools and services for building and managing web and mobile applications, including authentication and real-time database capabilities.
- **Malicious Code:** Code that is intentionally designed to compromise the security or functionality of a computer system or network, often with harmful intent.
- **Acceptable Use Policies:** Guidelines and rules established by the platform to outline acceptable behavior and usage for users, ensuring a safe and respectful community environment.

Appendix B: Analysis Models

- System Architecture: The project is built on React JS. For the online Compiler , we have used the Judge0 Api. Judge0 is a Robust, scalable, and open-source online code execution system Api that can be used to build a wide range of applications that need online code execution features. The judge0 servers restrict the API calls to their servers at only 50 per day. Due to this reason we choose to host the API live on our own Azure cloud server which allows unlimited access to the API.
- Test Plan: The test plan for CodeCat outlines the strategies and procedures for ensuring the quality and functionality of the online coding platform. The testing process aims to identify and rectify any issues, ensuring a smooth user experience and reliable performance. The plan includes various testing stages, such as unit testing, integration testing, system testing, and user acceptance testing (UAT).
- Software Project Management: The software project management plan outlines the strategies, processes, and roles necessary to effectively manage the development and deployment of CodeCat. This plan ensures that the project is executed efficiently, on time, and within budget, while meeting the specified quality standards.

Fig : System Architecture

Fig: Test Plan

Test case	Description	Intended result	Actual result	Completed by
Front-end Design	To have a check if the webpage designed is as same as the figma design created	The figma design had to be coded using react.js	The webpage resembled the design and had a very similar look	Pargat
Checking if all languages are supported	Boiler plates of every language where to be added and checked if its functioning properly	Every language need to be supported and complied with a initial example code	All languages where supported and example code of each language where seen.	Meet
Code Validation	Check whether the complier shows a error or not for a false input by the user	The intended result was the error must be thrown with a specified line number	The result was achieved	Vishrut
Platform Check (Pwa test)	Check where the complier works properly on a portable mobile device	To create a progressive web app and check if works as per the desire of the team	Yes Pwa was successful and user friendly too.	Pargat
Login and register validation	To check if mail entered while login are in a proper format or not. (eg. abc@gmail.com)	The system should throw error if the mail format is not correct and user is successfully registered.	Validated	Meet and Vishrut

Fig: Software Management Plan

Task	Member1 Pargat Singh	Member2 Meet Gala	Member 3 Name Vishrut Deshmukh
UI:			
Design	X		
Coding	X		
Database:			
Table design		X	
Query design		X	X
Coding	X		
Program:			
Log-on System	X		
Execution / Api connection	X		X
Input Validation	X	X	
Boiler-plate for each language		X	
Progressive Web App	X		
Testing:			
Testing approach		X	
Front End positioning	X		X
Database creation and insertion operations	X		
Input Validation		X	X
Presentation			
Report and Documentation	X	X	X

Post Laboratory Activity:

1. You are required to prepare SRS document for any project. (It could be the mini project you have completed in semester IV
2. Prepare questionnaire for the allotted project considering your lab instructor is the client for requirement gathering.
3. Consider following scenario: An institute is interested in developing a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

   The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (e.g. passwords) is stored in plain text.

   Prepare SRS document for the same in the format discussed in the write-up.

Library Information System Software Requirements Specification (SRS) Document:

Software Requirements Specification (SRS) for Library Information System (LIS)

1. **Introduction**

1.1 **Intent**

This document serves to outline the requisites and particulars essential for the construction of the Library Information System (LIS) web application. The system is designed to furnish a streamlined and user-centric platform, enabling students, staff, and library personnel to effectively oversee book transactions within the institute's Local Area Network (LAN).

1.2 **Extent**

LIS will empower users to execute the following functions:

- Conduct online searches and exploration of books (accessible to both members and non-members).
- Facilitate the borrowing and returning of books (limited to authorized members).
- Allow extensions of borrowing durations if the book is not reserved by another user.
- Empower the librarian to administer book archives, encompassing the addition of new books and the removal of books from circulation.

1.3 **Definitions, Acronyms, and Abbreviations**

- LIS: Library Information System
- LAN: Local Area Network

2. **Comprehensive Overview**

2.1 **Product Context**

LIS is a self-contained web application that operates within the institute's Local Area Network (LAN). It interfaces with a backend database to store and retrieve book and user data. The system ensures the secure storage of sensitive information, such as passwords, avoiding plaintext storage.

2.2 **User Categories and Characteristics**

LIS encompasses three primary user categories:

1. **Members:** Comprising students and staff who can engage in book borrowing and returning activities.
2. **Library Staff:** Encompassing staff responsible for daily book transactions and aiding members.
3. **Librarian:** Encompassing administrative personnel with full control over the system, charged with managing book records.

2.3 **Operational Surroundings**

LIS will manifest as a web application fashioned using HTML5 and other compatible web technologies. It will be hosted on a secure web server within the confines of the institute's LAN.

2.4 **Design and Implementation Limitations**

- The system's chief priority is data security, mandating the avoidance of sensitive information storage in plain text.
- The application's usability is paramount, ensuring cross-device and cross-browser accessibility.
- Responsive design is indispensable, guaranteeing a seamless experience across desktop and mobile platforms.

3. **Particular Requisites**

3.1 **Functional Prerequisites**

3.1.1 **User Enrollment and Verification**

- The system must facilitate user registration via institute-provided credentials.
- User passwords must be securely hashed and stored in the database.
- User authentication is obligatory prior to accessing book borrowing and returning functionalities.

3.1.2 **Book Exploration and Search (Open to All)**

- The system is obliged to provide a search mechanism for users to locate books based on titles, authors, or categories.
- All users, inclusive of both members and non-members, are permitted to browse the book catalog sans authentication.

3.1.3 **Book Borrowing and Returning (Restricted to Members)**

- Members must be enabled to borrow books by selecting their desired books and confirming the transaction.
- The return of borrowed books must be feasible by marking them as returned in the system.

3.1.4 **Book Prolongation (Members Exclusive)**

- Members must be capable of requesting extensions for borrowing durations, conditioned on the absence of other reservations for the specific book.
- Limitations on the number of extension requests for each book are mandated.

3.1.5 **Management of Book Records (Exclusive to Librarians)**

- Librarians must wield administrative privileges to introduce new book records into the system.
- The removal of book records from the system must be viable for librarians, especially if a book is no longer available.

3.2 **Non-Functional Prerequisites**

3.2.1 **Security**

- Secure communication between clients and the server shall be upheld using HTTPS.
- Industry-standard algorithms must be employed to securely hash user passwords.
- Access to book borrowing and returning functionalities shall be confined to authenticated users solely.

3.2.2 **Performance**

- The system must be prompt and responsive, delivering swift responses to user interactions.
- Database queries must be optimized to ensure efficient data retrieval and management.

3.2.3 **User-Friendliness**

- The user interface shall be instinctive and user-friendly, necessitating minimal user training.
- The design must adapt responsively, ensuring compatibility across diverse devices and browsers.

3.2.4 **Dependability**

- The system is required to maintain data integrity consistently during book transactions.
- Regular data backups must be executed to avert data loss.

3.2.5 **Portability**

- The web application shall be platform-agnostic, accessible from assorted devices within the institute's LAN.

Hardware Prerequisites:

1. **Server:**
    - Processor: Multi-core processor with ample processing power (e.g., Intel Xeon or AMD Ryzen series).
    - RAM: Minimum of 8GB RAM, with higher capacity recommended for enhanced performance.
    - Storage: Adequate storage for web application files and database (e.g., SSD for optimal performance).
    - Network Interface Card: Gigabit Ethernet adapter for accelerated LAN communication.
2. **Network Infrastructure:**
    - A well-structured Local Area Network (LAN) with appropriate cabling and switches to ensure steadfast communication between clients and the server.
3. **Client Devices:**
    - Desktops, laptops, tablets, or mobile devices capable of running modern web browsers (e.g., Chrome, Firefox, Safari).

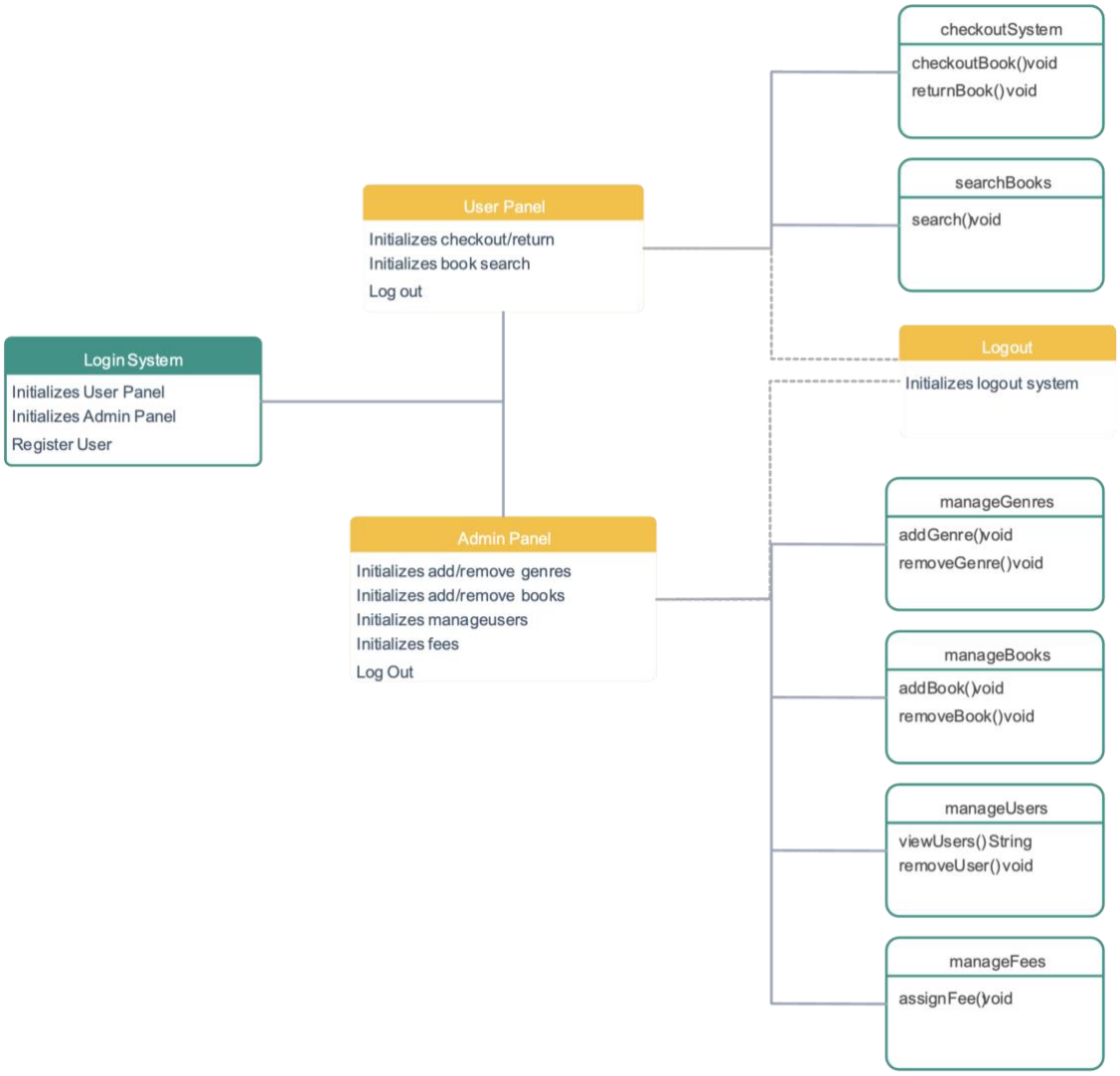
Software Prerequisites:

1. **Operating System:**
    - Server: Linux (e.g., Ubuntu, CentOS) or Windows Server for web application hosting.
    - Clients: Windows, macOS, or any OS compatible with contemporary web browsers.

2. **Web Server:**
    - Apache or Nginx: Esteemed and dependable web servers for web application hosting.
3. **Database:**
    - MySQL, PostgreSQL, or another relational database management system (RDBMS) for book and user data storage.
4. **Programming Languages and Frameworks:**
    - HTML5: Structure for web application content and user interface.
    - CSS3: Styling for the web application and enhanced user interface.
    - JavaScript: Integration of interactive elements and client-side functions.
    - Server-side Language: PHP, Python, Node.js, or other backend languages for server-side logic.
    - Web Application Framework: Utilization of frameworks like Flask, Django, Laravel, or Express.js for streamlined development.
5. **Security:**
    - HTTPS: SSL/TLS certificate for secure client-server communication.
    - Robust Password Hashing: Integration of libraries or modules for secure password hashing and storage.
6. **Supplementary Libraries and Tools:**
    - JavaScript libraries like jQuery for enriched user interactions.
    - Front-end frameworks like Bootstrap for responsive design.
    - AJAX for asynchronous data exchange between client and server.
    - Database Connector/ORM: Libraries or frameworks for database interaction.
7. **Development Software:**
    - Code Editor: Text editors or Integrated Development Environments (IDEs) such as Visual Studio Code, Sublime Text, or PyCharm.
    - Version Control: Git and platforms like GitHub or GitLab for version management and collaboration.

4. Appendix

This section may comprise additional materials like mockups



Post Lab Descriptive Questions answers must be handwritten and to be submitted BEFORE the next tern.

1. **What are different techniques to gather information for software development?**
   - There are several techniques used to gather information for software development:
   - a. **Interviews:** Direct discussions with stakeholders, users, and subject matter experts to collect insights and requirements.
   - b. **Questionnaires and Surveys:** Structured forms distributed to users to gather their needs and preferences.
   - c. **Observations:** Directly observing users while they perform tasks to understand their workflow and pain points.
   - d. **Workshops:** Collaborative sessions where stakeholders come together to discuss and define requirements.

- e. **Prototyping:** Developing mockups or prototypes to visualize the software and gather feedback early in the process.
- f. **Focus Groups:** Small groups of users discussing their needs and opinions about the software.
- g. **Document Analysis:** Reviewing existing documents, manuals, and reports to extract requirements.
- h. **Brainstorming:** A creative technique where team members generate ideas collectively.
- i. **Contextual Inquiry:** Observing users in their natural environment and discussing their needs.
- j. **Ethnographic Studies:** In-depth study of users in their real-world context to uncover hidden needs.

2. List verification and validation techniques for requirements.

a. **Inspection:** Careful review of requirements documents to identify errors, inconsistencies, and ambiguities.
b. **Walkthroughs:** A group review process where participants go through the requirements together, discussing potential issues.
c. **Prototyping:** Creating a working model to demonstrate the requirements and gather feedback on their accuracy and completeness.
d. **Simulation:** Creating a simulation to test how the software would behave based on the requirements.
e. **Test Cases:** Developing test cases that verify whether each requirement is met by the software.
f. **Traceability Analysis:** Ensuring that each requirement is traceable through various stages of development and testing.
g. **Use Cases:** Describing how users will interact with the software to validate that it fulfills their needs.
h. **Peer Review:** Having colleagues review the requirements for errors and inconsistencies.
i. **Formal Methods:** Applying mathematical techniques to formally prove the correctness of requirements.
j. **User Acceptance Testing (UAT):** Letting end-users validate that the software meets their needs.