BVRIT HYDERABAD College of Engineering for Women

(Approved by AICTE | Affiliated to JNTUH | Accredited by NAAC with Grade 'A' &NBA for CSE, ECE, EEE, & IT)

Bachupally, Hyderabad-500090

Department of CSE(Artificial Intelligence and Machine Learning)

Department of Computer Science and Engineering(AIML)

Certified that this is the bonafide record of the work done by Miss _____ *Registration no.* _____ *of Class* AIML *Year* II *Semester* I *in Software Engineering Laboratory.*

Date: *Head of the Dept.* *Staff In charge*

Regd. no. .....

Submitted for the University Practical Examination held on .....

Internal Examiner *External Examiner*

INDEX

INDEX

BOOK BANK SYSTEM

1. Problem statement

A Book Bank lends books and magazines to member, who is registered in the system. Also it handles the purchase of new titles for the Book Bank. Popular titles are brought into multiple copies. Old books and magazines are removed when they are out or date or poor in condition. A member can reserve a book or magazine that is not currently available in the book bank, so that when it is returned or purchased by the book bank, that person is notified. The book bank can easily create, replace and delete information about the tiles, members, loans and reservations from the system.

2. Software Requirement Specification Document

2.1 Functional Requirements

If the entire process of 'Issue of Books or Magazines' is done in a manual manner then it would take several months for the books or magazines to reach the applicant. Considering the fact that the number of students for Book Bank is increasing every year, an Automated System becomes essential to meet the demand. So this system uses several programming and database techniques to elucidate the work involved in this process. The system has been carefully verified and validated in order to satisfy it. The System provides an online interface to the user where they can fill in their personal details and submit the necessary documents (may be by scanning). The authority concerned with the issue of books can use this system to reduce his workload and process the application in a speedy manner.

2.2 Tools and Technology Requirements

The following are the list of software requirements we are using to implement this application.

- Client Side Technologies : HTML, CSS
- Scripting Language : JavaScript
- Business Logic Development Language : JSP
- Database Connectivity : JDBC
- Database : MYSQL
- Operating System : Windows 10
- Documentation : MS-Office

Hardware Requirements:

The following are the hardware requirements with minimum configuration to get better performance of our application.

· Processor : Pentium-IV Systems
· RAM : 512MB or above
· Hard Disk : 20GB or above
· Input and Output Devices : Keyboard, Monitor

Deployment Requirements:

· Front end : Java 1.8
· Technologies : JSP and JDBC
· Database : MYSQL server
· Web Server : Apache Tomact 8.5

2.3 Non-functional Requirements

It is the response time, utilization and throughput behaviour of the system. Care is taken so as to ensure a sytem with comparatively high performance.

Maintainability:

All the modules must be clearly separate to allow different user interfaces to be developed in future. Through thoughtful and effective software engineering, all steps of the product throughout its life time. All development will be provided with good documentation.

Reliability: The software should have less failure rate.

3. Design Documents

The purpose of a design is to describe how the enhancements will be incorporated into the existing project. It should contain samples of the finished product. This could include navigational mechanism screenshots, example reports, and UML diagrams.

i. Use case diagrams

A use case diagram is a diagram that shows a set of use cases and actors and their relationships.

Common Properties:

A use case diagram is just a special kind of diagram and shares the same common properties as do all other diagrams - a name and graphical contents that are a projection into a model. What distinguishes a use case diagram from all other kinds of diagrams is its content.

Contents:

Use case diagrams commonly contain

· Use cases

· Actors

· Dependency, generalization, and association relationships

Common Uses:

The use case diagrams are used to model the static use case view of a system. This view primarily supports the behavior of a system - the outwardly visible services that the system provides in the context of its environment.

ii. Class Diagrams

A class diagram is a diagram that shows a set of classes, interfaces, and collaborations and their relationships.

Class diagram commonly contain the following things:

· Classes

· Interfaces

· Collaborations

· Dependency, generalization and association relationships

Common Uses:

Class diagrams are used to model the static design view of a system. While modelling the static design view of a system, class diagrams are used in one of the three ways:

· To model the vocabulary of a system

· To model simple collaborations

iii. Sequence Diagrams

A sequence diagram emphasizes the time ordering of messages. Sequence diagram is formed by first placing the objects that participate in the interaction at the top of your diagram, across the X axis. Typically, you place the object that initiates the interaction at the left, and increasingly more subordinate objects to the right. Next, you place the messages that these objects send and receive along the Y axis, in order of increasing time from top to bottom. This gives the reader a clear visual cue to the flow of control over time. Sequence diagrams have two features that distinguish them from collaboration diagrams.

· First, there is the object lifeline. An object lifeline is the vertical dashed line that represents the existence of an object over a period of time.

· Second, there is the focus of control. The focus of control is a tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure.

Content:

Sequence diagrams commonly contain

· Objects

· Links

· Messages

Common Use:

Modeling Flows of Control by Time Ordering.

iv. Collaboration Diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

v. Activity Diagrams

An activity diagram shows the flow from activity to activity. An activity is an ongoing non atomic execution within a state machine. Actions encompass calling another operation, sending a signal, creating or destroying an object, or some pure computation, such as evaluating an expression. Graphically, an activity diagram is a collection of vertices and arcs.

Common Properties:

An activity diagram is just a special kind of diagram and shares the same common properties as do all other diagrams - a name and graphical contents that are a projection into a model. What distinguishes an interaction diagram from all other kinds of diagrams is its content.

Content: Activity diagrams commonly contain

· Activity states and action states

· Transitions

· Objects

Common Uses:

Activity diagrams are used to model the dynamic aspects of a system. When you model the dynamic aspects of a system, you will typically use activity diagrams in two ways.

· To model a workflow

· To model an operation

vi. Statechart Diagrams

There are various types of test. Each test type addresses a specific testing requirement. Statechart diagram is used to describe the states of different objects in its life cycle. Emphasis is placed on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately. Statechart diagrams are very important for describing the states. Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

vii. Component Diagrams

Component diagrams are used to model the physical aspects of a system. Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node. Component diagrams are used to visualize the organization and relationships among components in a system. Component diagram commonly contain:

· Components
· Interfaces
· Relationships

3.1 Testing Document

i. Overview of Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing .

ii. Stages of Testing:

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs .This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Types of testing:

White-box testing:

White-box testing, sometimes called glass-box testing, is a test case design method that uses the control structure of the procedural design to derive test cases. These test cases

Guarantee that all independent paths within a module have been exercised at least once
Exercise all logical decisions on their true and false sides
Execute all loops at their boundaries and within their operational bounds

Black box testing:

Also called behavioral testing, focuses on the functional requirements of the software. It enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black-box testing is not an alternative to white-box techniques, but it is complementary approach.

Black box testing attempts to find errors in the following categories:

Incorrect or missing functions
Interface errors
Errors in data structures or external data base access
Behavior or performance errors
Initialization and termination errors.

3.2 Software Configuration Management

Software Configuration Management is defined as a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. It is abbreviated as the SCM process in software engineering. The primary goal is to increase productivity with minimal mistakes .The primary reasons for Implementing Software Configuration Management System are:

There are multiple people working on software which is continually updating
It may be a case where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently
Changes in user requirement, policy, budget, schedule need to be accommodated.
Software should able to run on various machines and Operating Systems
Helps to develop coordination among stakeholders
SCM process is also beneficial to control the costs involved in making changes to a system

3.3 Risk Management

Risk management assists a project team in identifying risks, assessing their impact and probability and tracking risks throughout a software project. Categories of risks:

Project risks
Technical risks
Business risks Risk Components:
Performance risk
Cost risk
Support risk
Schedule risk Risk Drivers:
Negligible
Marginal
Critical
Catastrophic

Issue	Department	Impact (Likelihood x Severity)	Priority
Size estimate may be significantly low	PS	60	2
Delivery deadline will be tightened	BU	50	2
Customer will change requirements	PS	80	2
Technology will not meet expectations	TE	30	1
Lack of training on tools	DE	80	3
Inexperienced staff	ST	30	2

ST- Staff size and experience risk BU – Business risk

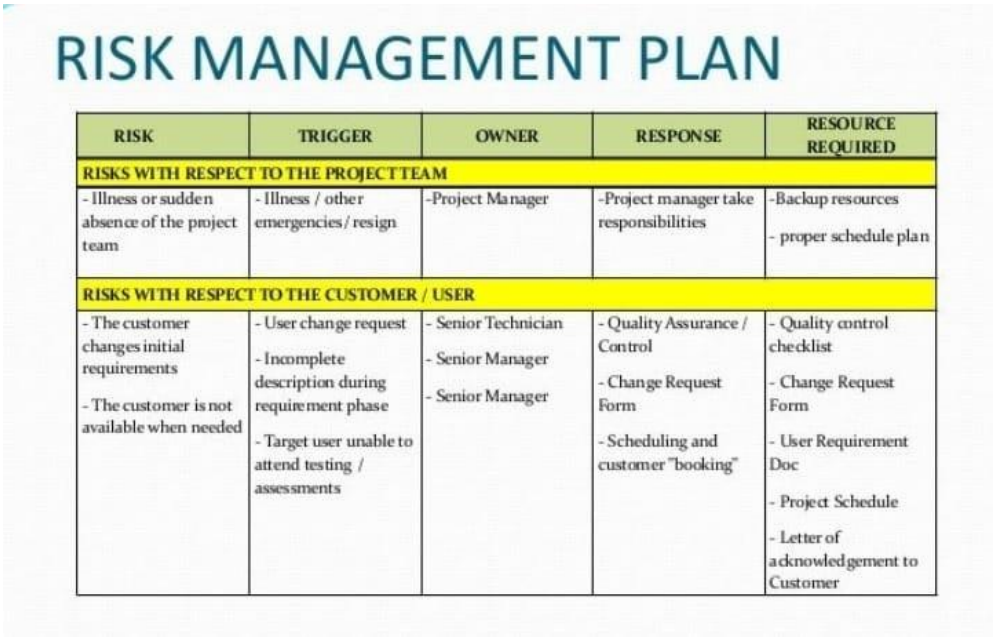
PS – Project size risk TE- Technology risk

1- Catastrophic

2- Critical

3- Marginal

4- Negligible



RISK MANAGEMENT PLAN

RISK	TRIGGER	OWNER	RESPONSE	RESOURCE REQUIRED
RISKS WITH RESPECT TO THE PROJECT TEAM				
- Illness or sudden absence of the project team	- Illness / other emergencies / resign	-Project Manager	-Project manager take responsibilities	-Backup resources - proper schedule plan
RISKS WITH RESPECT TO THE CUSTOMER / USER				
- The customer changes initial requirements - The customer is not available when needed	- User change request - Incomplete description during requirement phase - Target user unable to attend testing / assessments	- Senior Technician - Senior Manager - Senior Manager	- Quality Assurance / Control - Change Request Form - Scheduling and customer "booking"	- Quality control checklist - Change Request Form - User Requirement Doc - Project Schedule - Letter of acknowledgement to Customer

4. Design Phase tool

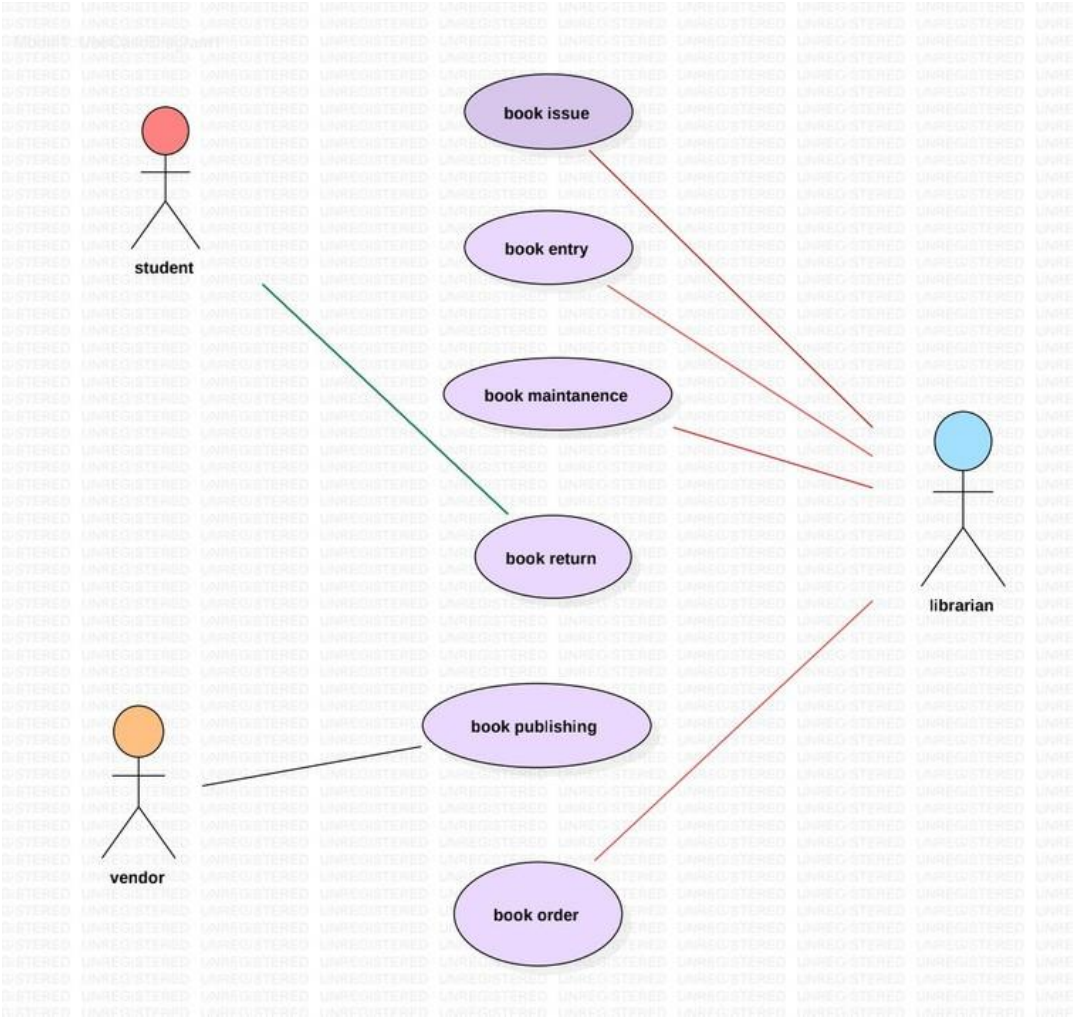
StarUML:

StarUML is an open source software modeling tool that supports the UML (Unified Modeling Language) framework for system and software modeling. It is based on UML version 1.4, provides different types of diagram and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept and allowing to generate code for multiple languages. StarUML supports the following diagram types

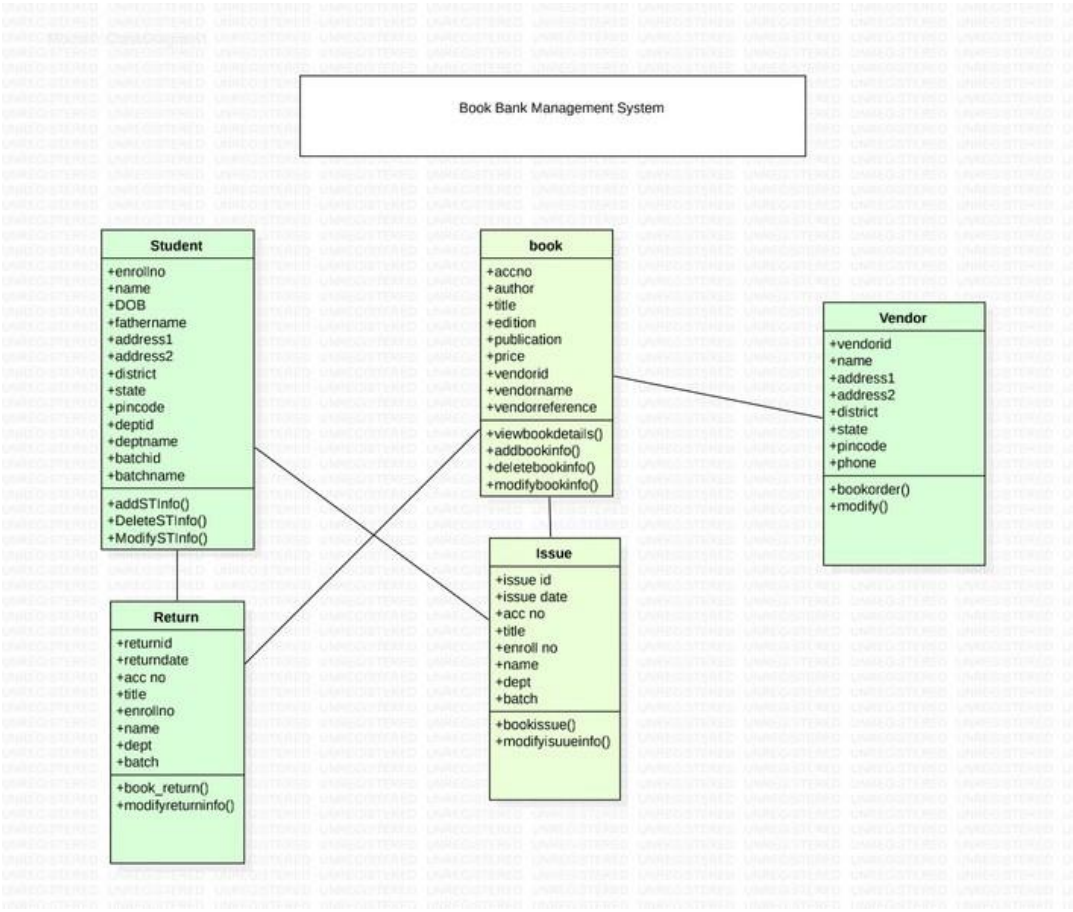
1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. Collaboration Diagram
5. Activity Diagram
6. State Chart Diagram
7. Component Diagram

5.Design Model

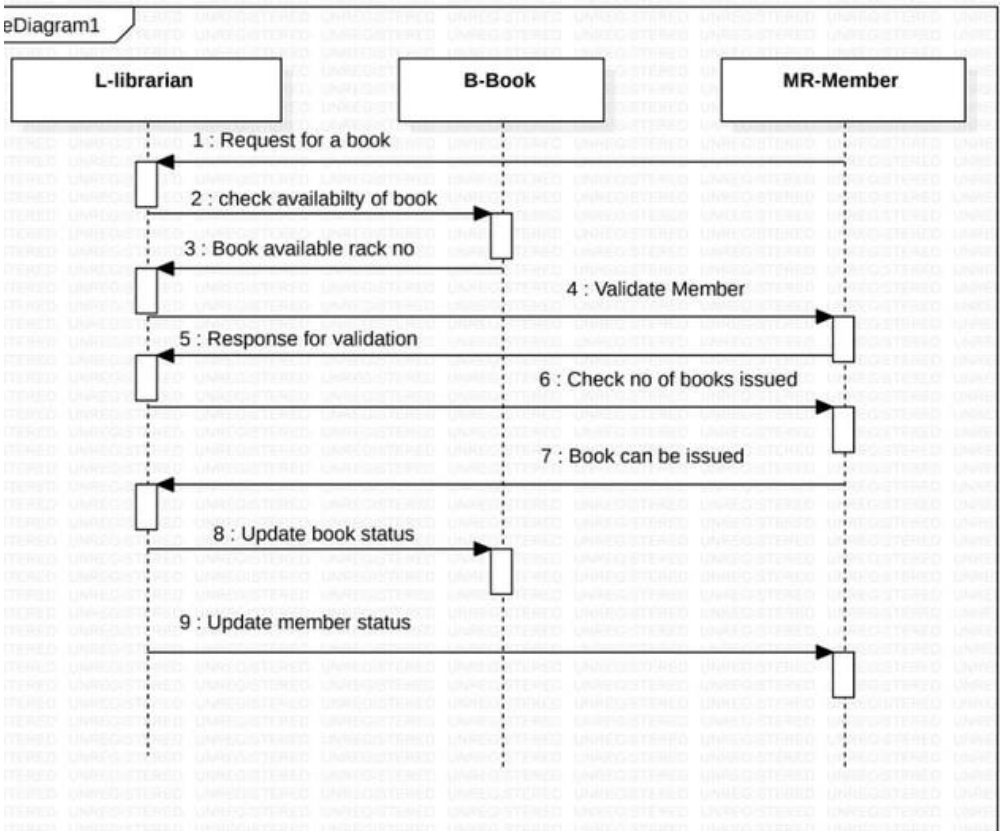
i . USECASE DAIGRAM



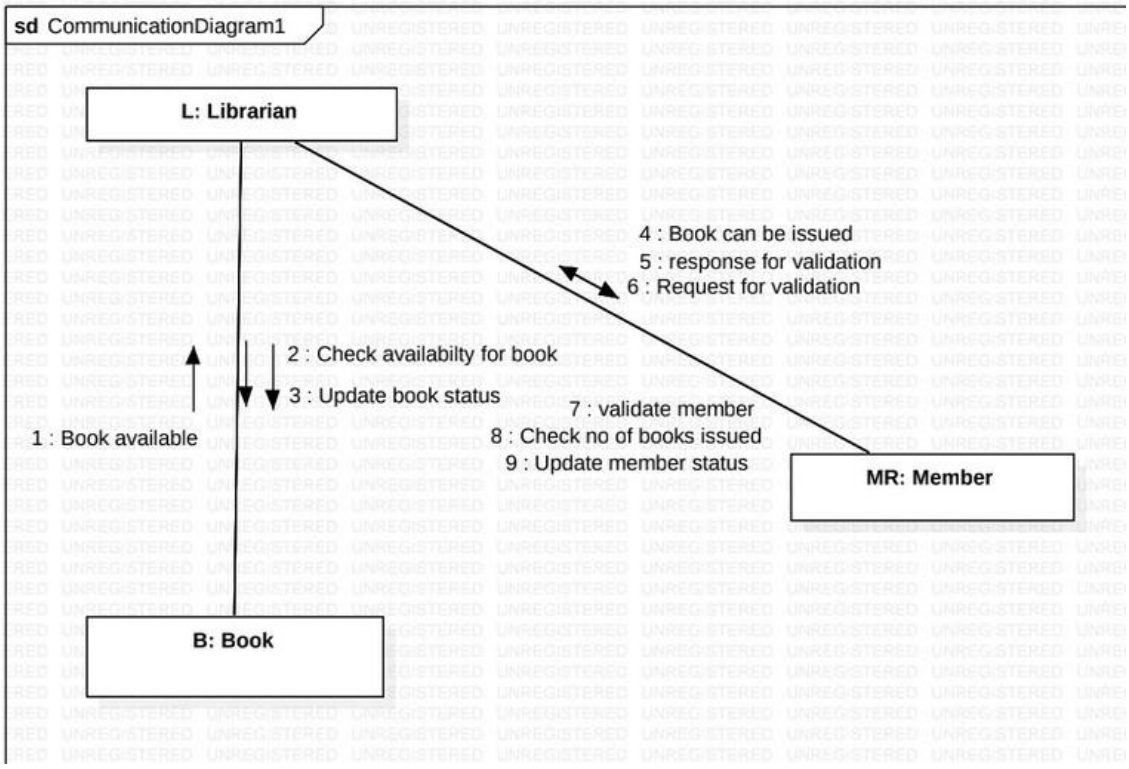
ii . CLASS DIAGRAM



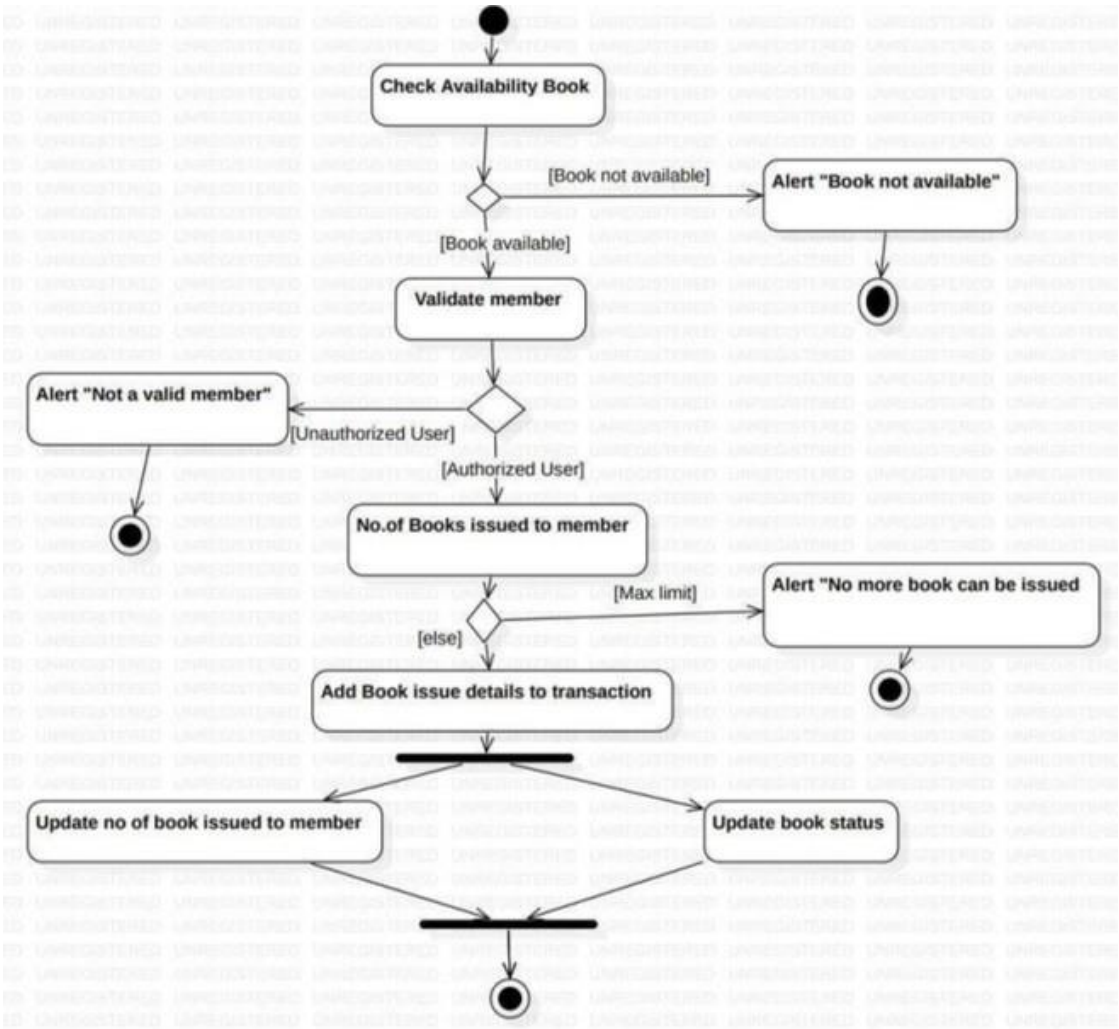
iii.SEQUENCE DIAGRAM



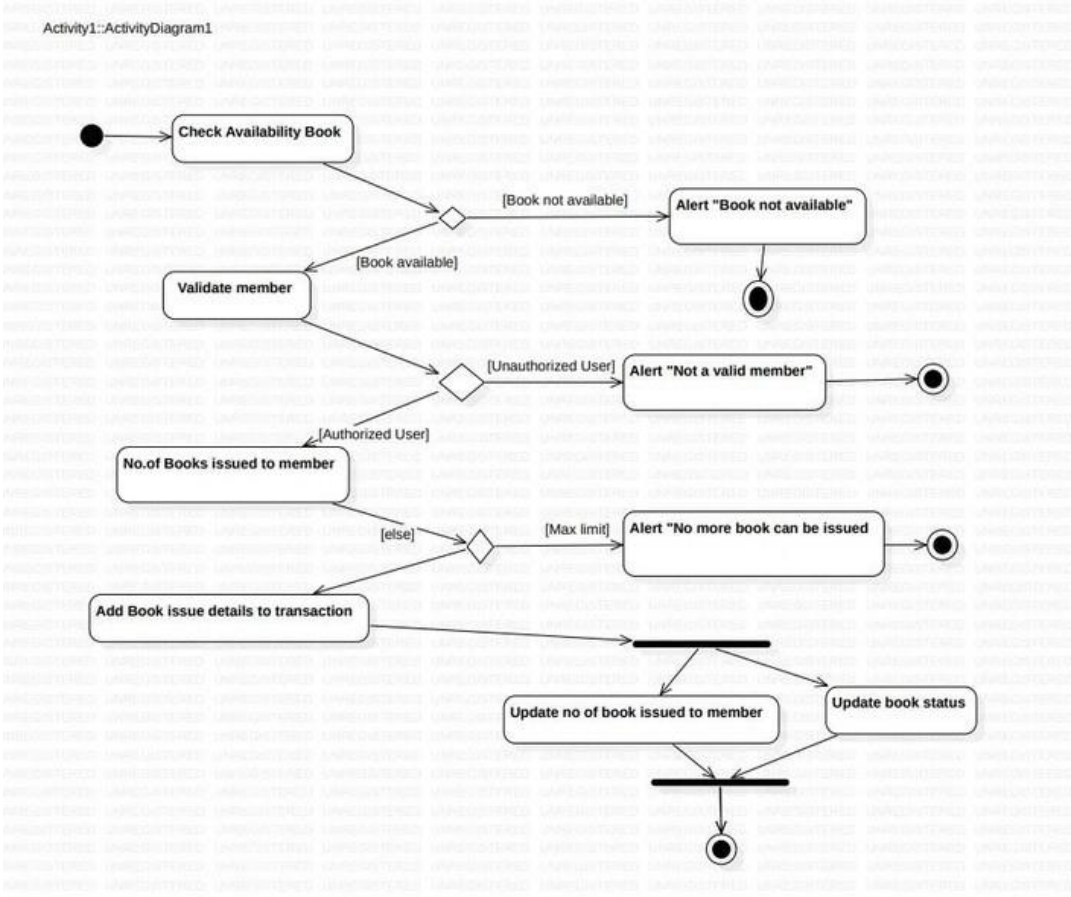
iv.COLLABARATION DIAGRAM



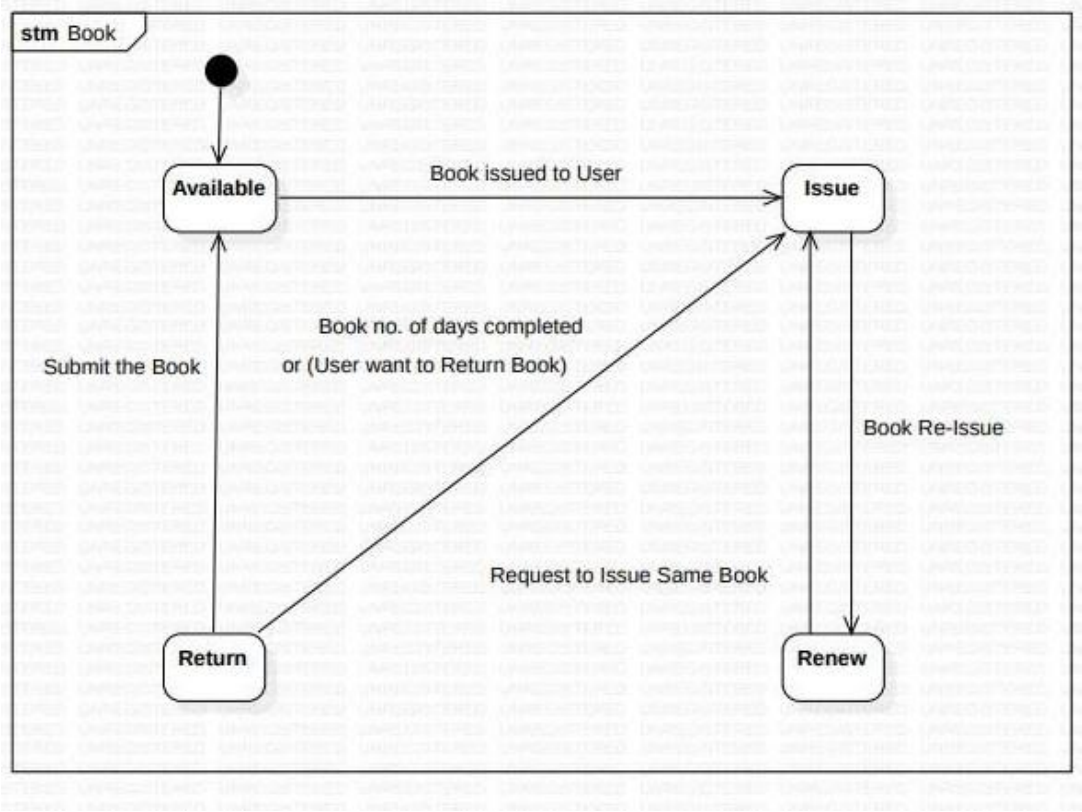
V.ACTIVITY DIAGRAM FOR BOOK ISSUE



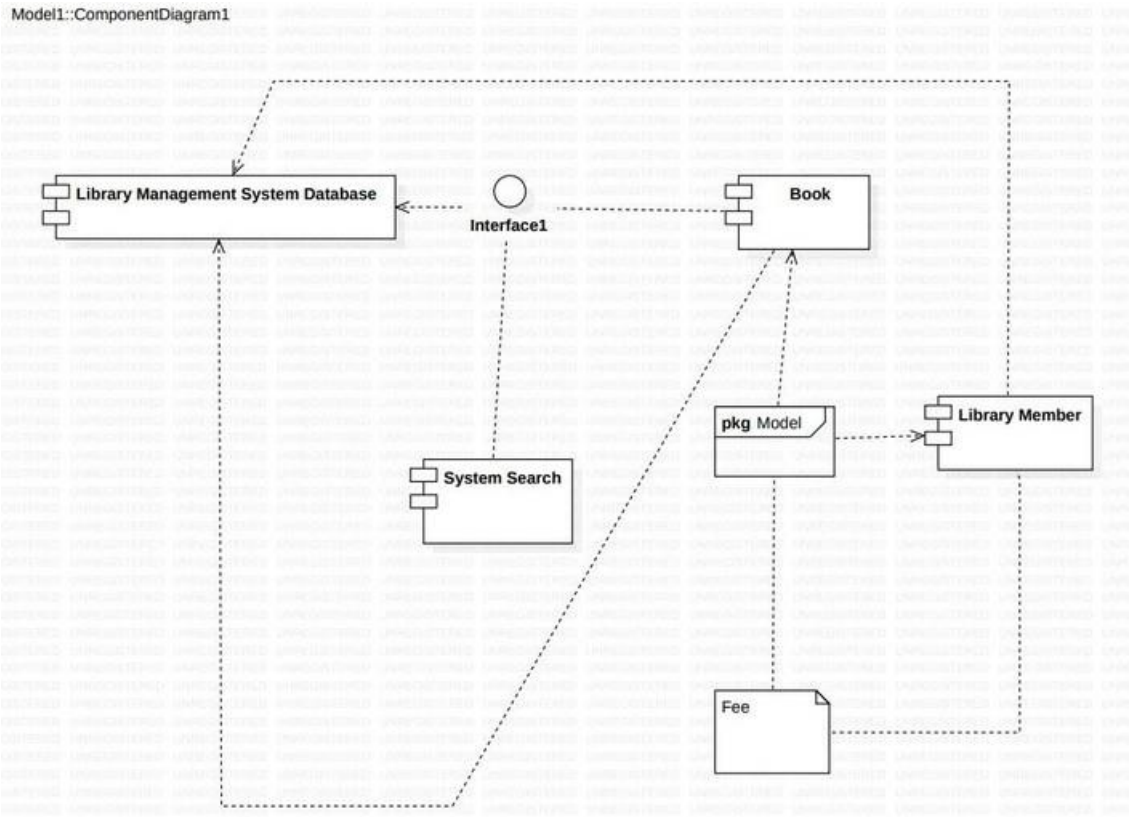
vi.SWIMLANE DIAGRAM FOR BOOK ISSUE



vii.STATE CHART DIAGRAM



viii.COMPONENT DIAGRAM



6.Test Cases :

Test Case	Input	Expected Output	Actual Output	Description
Valid login	Username, Password, Type	Success	Success	Login successful.
Invalid Login	Username, Password, Type	Failed	Failed	Login unsuccessful. Try again.

ONLINE COURSE RESERVATION SYSTEM

1. Problem statement

The online Course Reservation system is built to be used by students and managed by an administrator. The student and employee have to login to the system before any processing can be done. The student can see the courses available to him/her and register for the course he/she wants. The administrator can maintain the course details and view all the students who have registered to any course.

2. Software Requirement Specification Document

2.1 Functional Requirements

• Secure Reservation of information by the Students.

• SMS and Mail updates to the students by the Registrar.

• Registrar can generate reports from the information and is the only authorized personnel to add the eligible application information to the database.

• Applicant - They are the person who desires to obtain the course and submit the information to the database.

• Administrator - He has the certain privileges to add the course status and to approve the issue of course. He may contain a group of persons under him to verify the documents and give suggestion whether to approve the dispatch of course.

17

2.2 Tools and Technology Requirement

The following are the list of software requirements we are using to implement this application.

- Client Side Technologies: HTML, CSS
- Scripting Language: JavaScript
- Business Logic Development Language: JSP
- Database Connectivity: JDBC
- Database: MYSQL
- Operating System: Windows 10
- Documentation: MS-Office

Hardware Requirements:

The following are the hardware requirements with minimum configuration to get better performance of our application

. • Processor: Pentium-IV Systems

- RAM: 512MB or above
- Hard Disk: 20GB or above
- Input and Output Devices: Keyboard

Monitor Deployment Requirements:

- Front end: Java 1.8
- Technologies: JSP and JDBC
- Database: MYSQL server
- Web Server: Apache Tomact 8.5

2.3 Non-functional Requirements

Performance:

It is the response time, utilization and throughput behaviour of the system. Care is taken so as to ensure a system with comparatively high performance.

Maintainability:

All the modules must be clearly separate to allow different user interfaces to be developed in future. Through thoughtful and effective software engineering, all stepsof the product throughout its life time. All development will be provided with good documentation.

Reliability:

The software should have less failure rate.

3. Design Documents

3.1 UML Diagrams

i. Use Case Diagram

A use case diagram is a diagram that shows a set of use cases and actors and their relationships.

Common Properties:

A use case diagram is just a special kind of diagram and shares the same common properties as do all other diagrams – a name and graphical contents that are a projection into a model. What distinguishes a use case diagram from all other kinds of diagrams is its content.

Contents:

Use case diagrams commonly contain

- Use Cases
- Actors
- Dependency, generalization, and association relationships

Common Uses:

The use case diagrams are used to model the static design view of a system. This view primarily supports the behavior of a system - the outwardly visible services that the system provides in the context of its environment.

ii. Class Diagram

A Class diagram is a diagram that shows a great classes and interfaces, and collaborations and their relationships.

Class diagram contain the following things:

- Classes
- Interfaces
- Collaborations
- Dependency, generalization

- Association relationships.

Common Uses:

Class diagrams are used to model the static design view of a system. While modelling the static design view of a system, class diagram are used in oneof the three ways:

- To model the vocabulary of a system.
- To model simple collaborations.
- To model a logical database schema.

iii. Sequence Diagrams

A sequence diagram emphasizes the time ordering of messages. Sequence diagram is formed by first placing the objects that participate in the interaction at the top of your diagram, across the X axis. Typically, you place the object that initiates the interaction at the left, and increasingly more subordinate objects to the right. Next, you place the messages that these objects send and receive along the Y axis, in order of increasing time from top to bottom. This gives the reader a clear visual cue to the flow of control over time.

Sequence diagrams have two features that distinguish them from collaboration diagrams.

- First, there is the object lifeline. An object lifeline is the vertical dashed line that represents the existence of an object over a period of time.
- Second, there is the focus of control. The focus of control is a tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure.

Content:

- Sequence diagrams commonly contain
- Objects
- Links

- Messages

Common Use:

- Modelling Flows of Control by Time Ordering

iv. Collaboration Diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

v. Activity Diagram

In an activity diagram, several key elements are utilized to represent the dynamic aspects of a system or process. The primary components include Activities, which are actions or operations denoted by rounded rectangles and labeled with a verb.

Content:

The flow of activities is depicted through Transitions or Flows, represented by arrows connecting activities. Decision points are illustrated using Decision or Choice Nodes, denoted by diamond shapes, indicating where the flow can take different paths based on conditions. For parallel flows, the diagram employs Fork and Join Nodes, with forks splitting a single flow into multiple parallel flows and joins merging them back into a single flow. This Activity diagram begins with a Start Node, typically a circle, and concludes with an End Node, represented by a solid or concentric circle. To organize activities based on roles or responsibilities, the diagram may incorporate Swimlanes, either vertically or horizontally partitioning the activities.

Common Uses:

- Business Process modelling
- System Analysis and Design
- Use case scenario

vi. State Chat Diagram

A state chart is a UML diagram that represents the dynamic behaviour of a system in response to external stimuli. It primarily consists of the following elements.

Content:

- States
- Transitions and events
- Initial State and Final State

Common Uses:

- Software Modelling
- Emended Modelling
- User Interface Design
- Control Design

vii. Component Diagram

A component diagram is a type of UML diagram that visually represents the high-level structure of a system, focusing on the components, their interactions, and the relationships between them. It provides a static view of the system architecture.

Content:

- Component

- Interface
- Dependencies and Relationships
- Associations

Common Uses:

- System Architecture Design
- System Maintenance
- Component level testing
- Communication

3.2 Testing Document

1. Overview of Testing

Testing in software engineering for an online course reservation system is a critical phase to ensure the reliability, functionality, and security of the software. The testing process involves various stages, each focusing on different aspects of the system. Here's an overview of the testing activities typically performed:

Requirement Analysis and Validation:

Ensure that the requirements for the online course reservation system are clear, complete, and feasible. Validate requirements with stakeholders to avoid misunderstandings and misinterpretations.

Unit Testing:

Test individual components or units of code in isolation. Verify that each module or function works as expected. Identify and fix any bugs at the code level.

Integration Testing:

Test the interaction between different components or modules. Verify that integrated modules function correctly together. Address issues related to data flow and communication between components.

System Testing:

Evaluate the entire system's functionality against the specified requirements. Ensure that the online course reservation system behaves as expected in various scenarios. Test different user roles and their interactions with the system.

2. Stages of Testing

Testing for an online course reservation system involves several crucial stages in software engineering:

Performance Testing:

Assess the system's performance under different conditions, including peak loads and concurrent users.

System Testing:

Identify and address potential security vulnerabilities to ensure the protection of user data.

Usability Testing:

Evaluate the user interface (UI) and overall user experience to ensure ease of use.

Compatibility Testing:

Verify that the system works seamlessly across various browsers, devices, and operating systems.

Regression Testing:

Confirm that new code changes or enhancements do not adversely affect existing functionalities.

Automated Testing:

Implement automated testing tools to streamline repetitive and time-consuming test cases.

Continuous Testing:

Integrate testing into the continuous integration/continuous deployment (CI/CD) pipeline for ongoing assessment.

3. Types of Testing

Load Testing:

Assess how the system performs under anticipated loads to ensure it can handle the expected number of users.

Stress Testing:

Push the system beyond its limits to identify the breaking point and assess its behaviour under extreme conditions.

Automated Testing:

Implement automated testing for repetitive and critical test scenarios, improving efficiency and consistency.

Accessibility Testing:

Ensure that the online reservation system is accessible to users with disabilities, meeting relevant accessibility standards.

Database Testing:

Verify the integrity, accuracy, and performance of the database operations, including data retrieval and storage.

Scalability Testing:

Assess the system's ability to scale and handle increased loads as the user base grows.

Cross-Browser Testing:

Confirm that the online reservation system functions correctly across different web browsers.

3.3 Software Configuration Management

Software Configuration Management is defined as a process to systematically manage,organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. System Configuration Management (SCM) is an arrangement of exercises that controls change by recognizing the items for change, setting up connections between those things, making/characterizing instruments for overseeing diverse variants, controlling the changes being executed in the current framework, inspecting and revealing/reporting on the changes made. It is essential to control the changes in light of the fact that if the changes are not checked legitimately then they may wind up undermining a well-run programming. It involves the following activities:

1. Identification and Establishment – Identifying the configuration items from products that compose baselines at given points in time a baseline is a set of mutually consisten. Establishing relationships among items, creating a mechanism to manage multiple levels of control and procedure for the change management system.

2. Version control – Creating versions/specifications of the existing product to build new products with the help of the SCM system.

3. Change control – Controlling changes to Configuration items (CI).

4. Configuration auditing – A software configuration audit complements the formal technical review of the process and product.

5. Reporting – Providing accurate status and current configuration data to developers, tester, end users, customers and stakeholders through admin guides, user guides, FAQs, Release notes, Memos, Installation Guide, Configuration guide etc .

The primary reasons for Implementing Software Configuration Management System are:

1. There are multiple people working on software which is continually updating.
2. It may be a case where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently.
3. Changes in user requirement, policy, budget, schedule need to be accommodated.
4. Software should able to run on various machines and Operating Systems.
5. Helps to develop coordination among stakeholders.
6. SCM process is also beneficial to control the costs involved in making changes to a system.

3.4 Risk Management

- Security: Ensure that the online registration system has robust security measures in place to protect sensitive data from unauthorized access, breaches, and cyber-attacks. This includes using encryption, firewalls, and other security technologies.

- Data Accuracy: Implement measures to ensure the accuracy of data collected through the online registration system. This includes validating data inputs, checking for errors, and ensuring that data is properly categorized and stored.

- System Performance: Ensure that the online registration system can handle peak registration periods and has the necessary capacity and scalability to support high volumes of traffic and data.

- User Experience: Design the online registration system to be user-friendly and intuitive, with clear instructions and feedback mechanisms to help users navigate the system and avoid errors.

- Integration: Ensure that the online registration system can integrate with other systems and tools used in the registration process, such as payment gateways, CRM systems, and email marketing tools.

- Compliance: Ensure that the online registration system complies with relevant laws and regulations, such as data protection and privacy laws, and any industry-specific regulations.

- Contingency Planning: Develop contingency plans for potential risks, such as system downtime, data breaches, or user errors, and ensure that there are backup and recovery procedures in place.

- Continuous Monitoring: Continuously monitor the online registration system for potential risks and issues, and have processes in place for reporting and addressing any issues that arise.

- Training and Support: Provide training and support to users and administrators of the online registration system to ensure that they are able to use the system effectively and safely.

- Documentation: Maintain accurate and up-to-date documentation of the online registration system, including technical specifications, user guides, and system logs, to support ongoing maintenance and improvement.

i. Risk Table

Risk ID	Trigger Points	Description	Impact	Category	Likelihood	Mitigation Strategy	Owner	Status
1	Inadequate security measures	Weak passwords and lack of two-factor authentication	Data breach and loss of user trust	Security	High	Implement strong password policies and two-factor authentication	IT	In Progress
2	System downtime	Server overload during peak registration periods	Inability for users to register and loss of revenue	Availability	Medium	Increase server capacity and implement load balancing	IT	To Do
3	User error	Users entering incorrect information during registration	Inaccurate data and need for manual correction	Data integrity	Implement data validation checks and user feedback messages	Development	To Do	
4	Integration issues	Compatibility issues with third-party systems	Inability to exchange data	Integration	Medium	Test and validate integrations	IT	To Do
5	Legal and compliance	Registration process may not comply with data protection regulations	Fines and legal action	Compliance	Medium	Consult with legal experts and implement necessary changes	IT	To Do

Risk Identification

Trigger Points: Insufficient data security measures, inadequate system performance, and lack of stakeholder engagement.

Impacts: Data breaches, system downtime, and ineffective risk management plan implementation.

Categories: Technical, operational, and stakeholder engagement.

RISK ASSESSMENT :

Likelihood: Medium for data breaches, high for system downtime, and low for ineffective risk management plan implementation.

Impact: High for all risks.

RISK MITIGATION STARTAGIES :

Technical Risks: Implement robust data security measures, including encryption, access controls, and regular system updates. Optimize system performance through scaling, load balancing, and fault tolerance mechanisms.

Operational Risks: Develop a comprehensive training and communication plan for stakeholders, including instructions for using the online registration system and updates on risk management plan progress.

Stakeholder Engagement Risks: Establish clear communication channels and protocols, including regular meetings and progress reports.

4. Design Phase

StarUML:

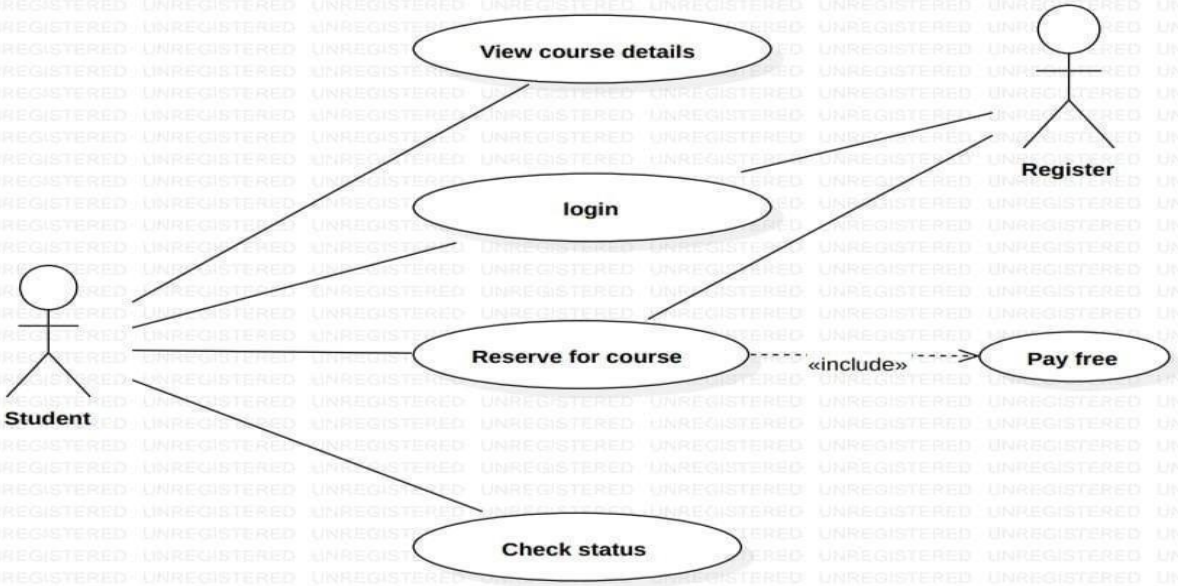
StarUML is an open source software modeling tool that supports the UML (UnifiedModeling Language) framework for system and software modeling. It based on UML version 1.4, provides different types of diagram and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept and allowing to generate code for multiple languages.

StarUML supports the following diagram types

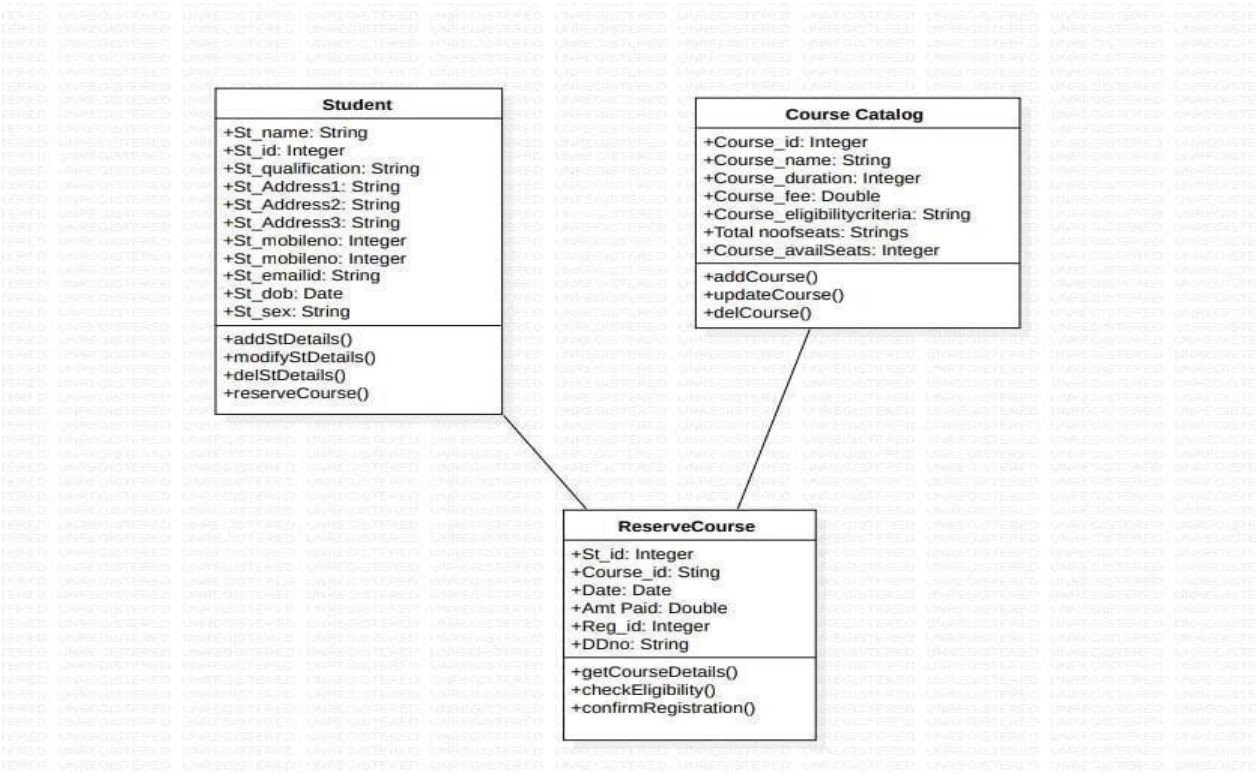
1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. Collaboration Diagram
5. State chart Diagram
6. Activity Diagram
7. Component Diagram
8. Deployment Diagram
9. Composite Structure Diagram

5. Design Model

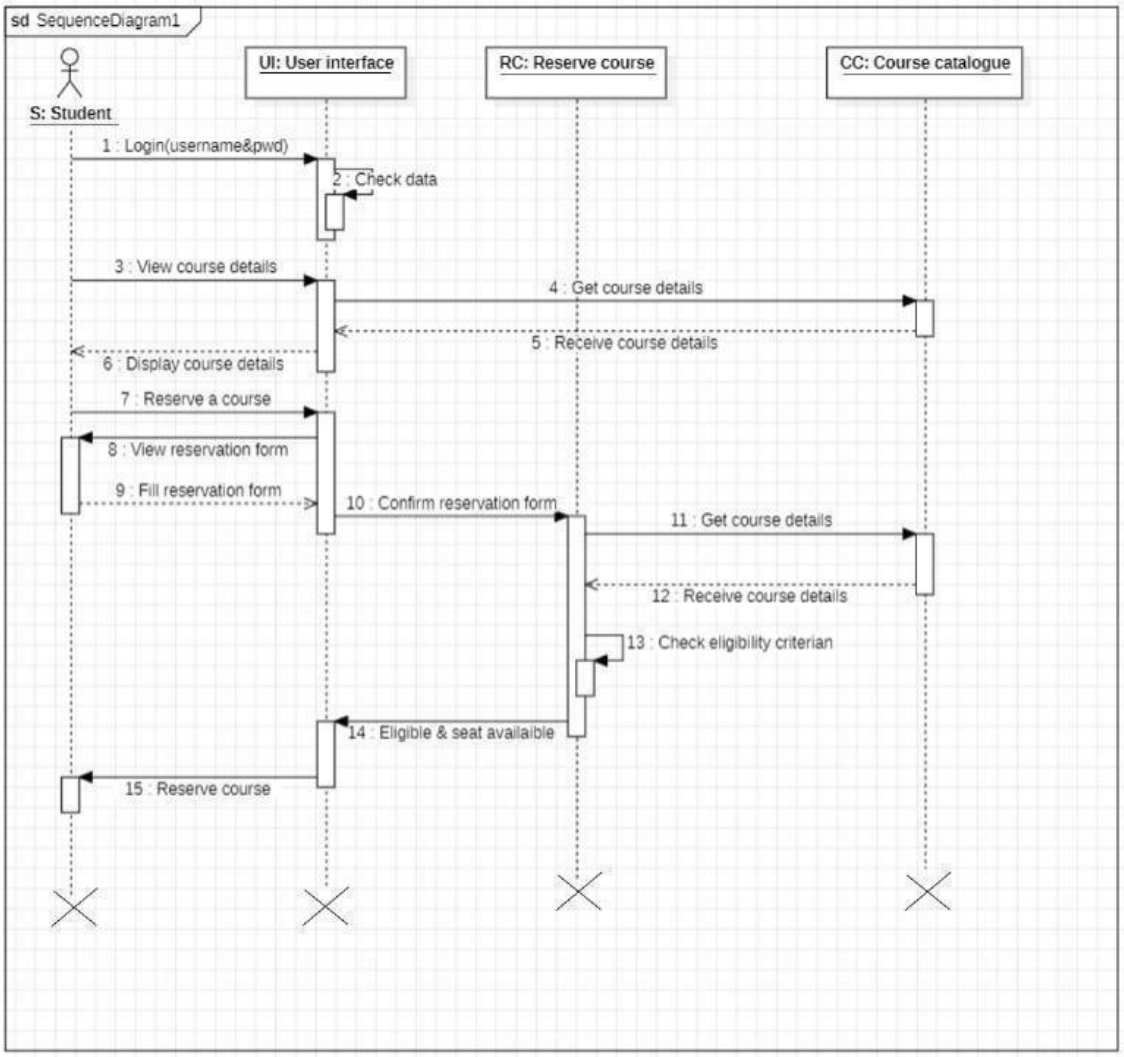
i. USECASE DIAGRAM



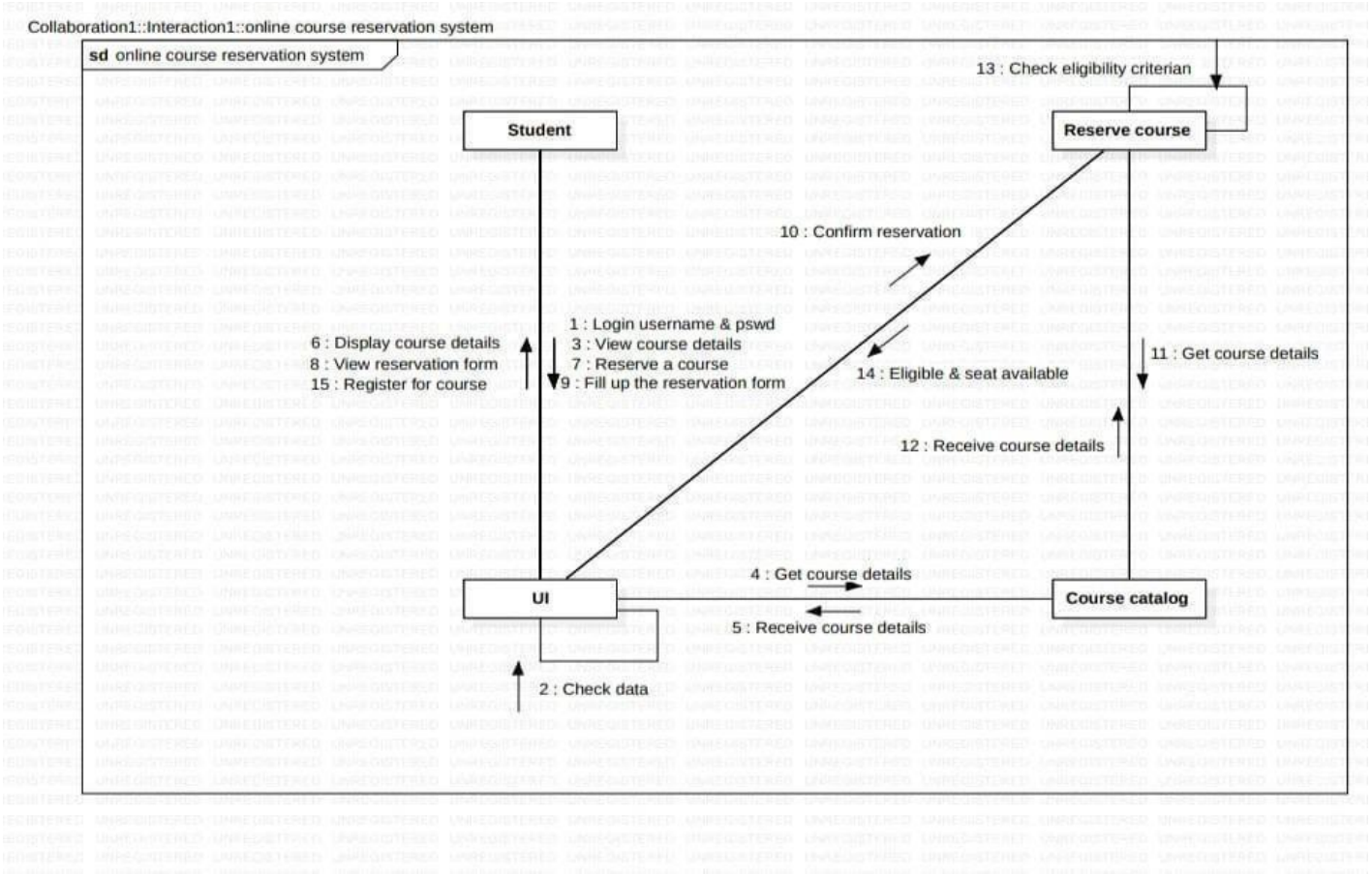
ii. CLASS DIAGRAM



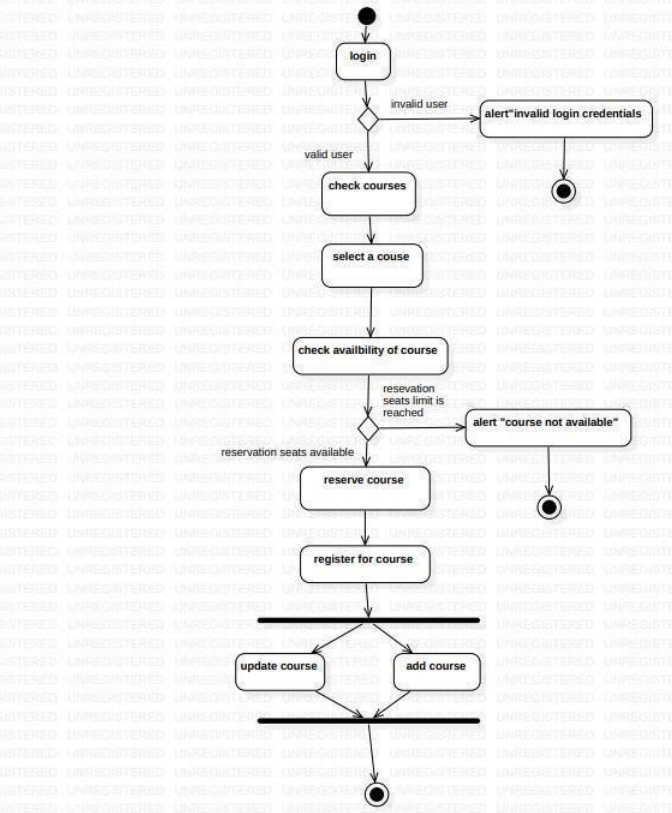
iii. SEQUENCE DIAGRAM



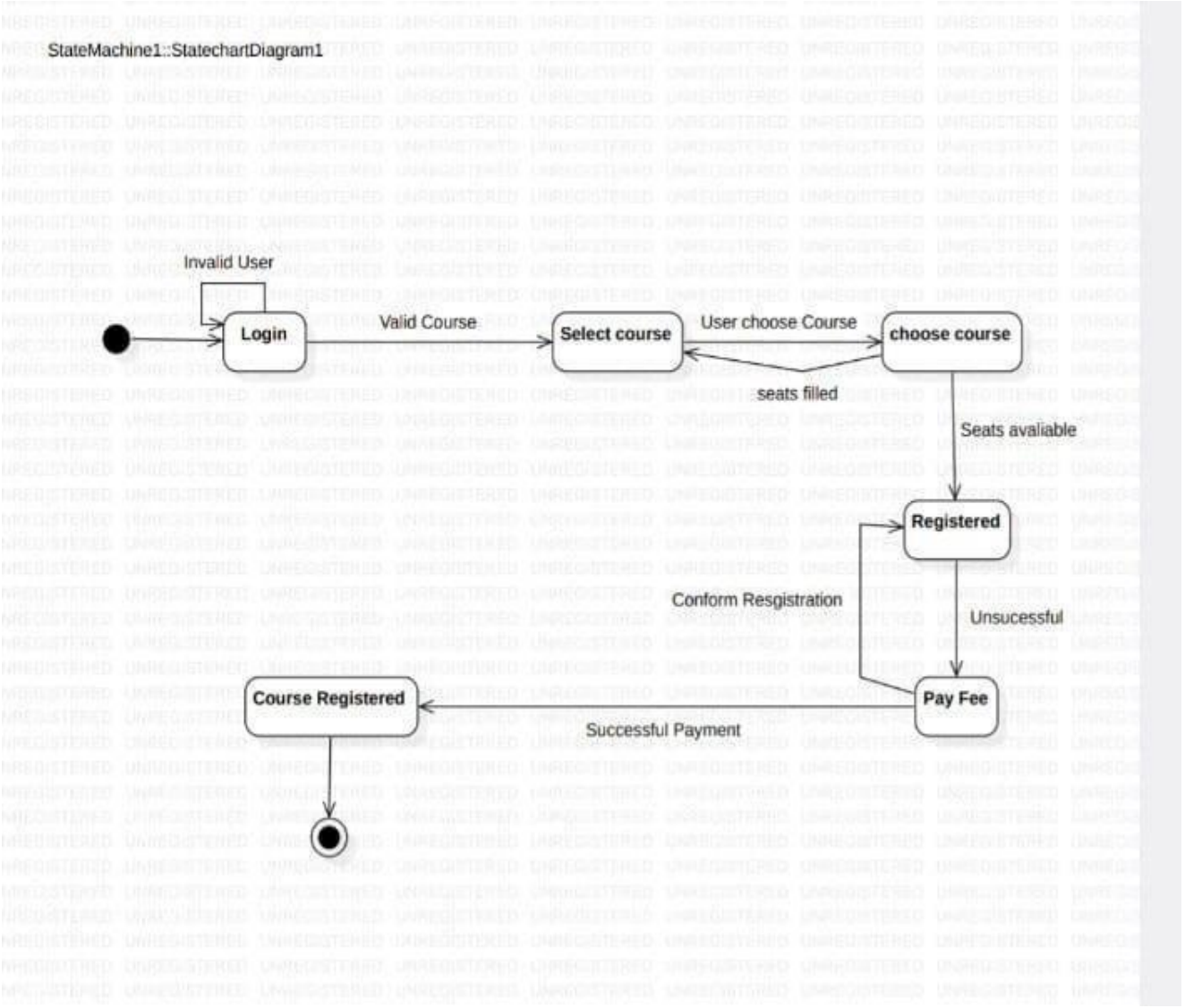
iv. COLLABORATION DIAGRAM



v. ACTIVITY DIAGRAM



vi. STATE CHART DIAGRAM



6. Test Cases:

Test Case	Input	Expected output	Actual output
Valid Login	Username and password	Success	Success
Invalid Login	Username and password	Success	Failure
Seat available for reserving course	Course	Success	Success
Seat unavailable for the reserving course	Course	Success	Failure
Able to register to the course	Course	Success	Success
Unable to register to the course	Course	Success	Failure