BOOK BANK SYSTEM

1.Problem statement

A Book Bank lends books and magazines to members, who are 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 Old books and magazines are removed when they are out of date or poor in condition.
A member can reserve a book or magazine that is not currently available in the book bank, so
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.

2.2Tools and Technology Requirements

The following are the list of software requirements we are using to implement thisClient 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 Tomcat 8.

2.3Non-functional Requirements

Performance:

It is the response time, utilization and throughput behavior 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 steps of the product throughout its lifetime. 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 diagram **3.1 Design Document Description**

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 modeling 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
- To model a logical database schema.

ii. 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 Hows of Control by Time Ordering.

IV Collaboration Diagram

Also known as a communication diagram is an illustration of the relationships and interaction among software objects in the UML.

- objects: These are shown as rectangles with naming labels inside. The naming label follows the convention of ● object name: class.

- Links: These connect objects with actors and are depicted using a solid line between two elements. Message between objects: these are shown as a labeled arrow placed near a link.

Common Use:

Modeling Flows of Control by Time Ordering.

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.

Activities ultimately result in some action, which is made up of executable atomic computations that result in a change in state of the system or the return of a value. 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

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of a Statechart diagram is to model the lifetime of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams :

- To model the dynamic aspect of a system.
- To model the lifetime of a reactive system.
- To describe different states of an object during its lifetime.

4

- Define a state machine to model the states of an object.

v. 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

3.2 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 tests.

Each test type addresses a specific testing requirement.

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.

All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .

It is done after the completion of an individual unit before integration. 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 satisfactory, 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.

iii. 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 Exercise internal data structures to ensure their validity

- **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 a 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 database access
- Behavior or performance errors
- Initialization and termination errors.

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. 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

3.4 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 **Risk Table:**

Risk	Category	Probability(%)	Impact	
Size estimation may be significantly low	PS	60	2	
Delivery deadline will be tightened	BU	50	2	
Customer will the requirements	PS	80	2	
Technology will not meet expectations	TE	30	1	
Lack of tracking on tools	DE	80	3	
Inexperienced staff	ST	30	2	

ST-staff size and experience risk

BU-business risk

TE-technology risk

PS-project size risk

1-catastrophobia

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 tools

STAR UML:

Star uml is open source software modeling tool that supports the uml (unified modeling language)Framework for a system and software modeling .It is based on the uml version 1.4,provides different types of diagrams and it accepts UML 2.0 notation.I actively supports MDA(Model Driven Architecture)Approach by supporting UML profile concepts and allowing to generate code for multiple languages.

UML supports the following diagrams:

1.USE CASE diagrams

2.CLASS diagrams

3.SEQUENCE diagram

4.COLLABORATION diagram

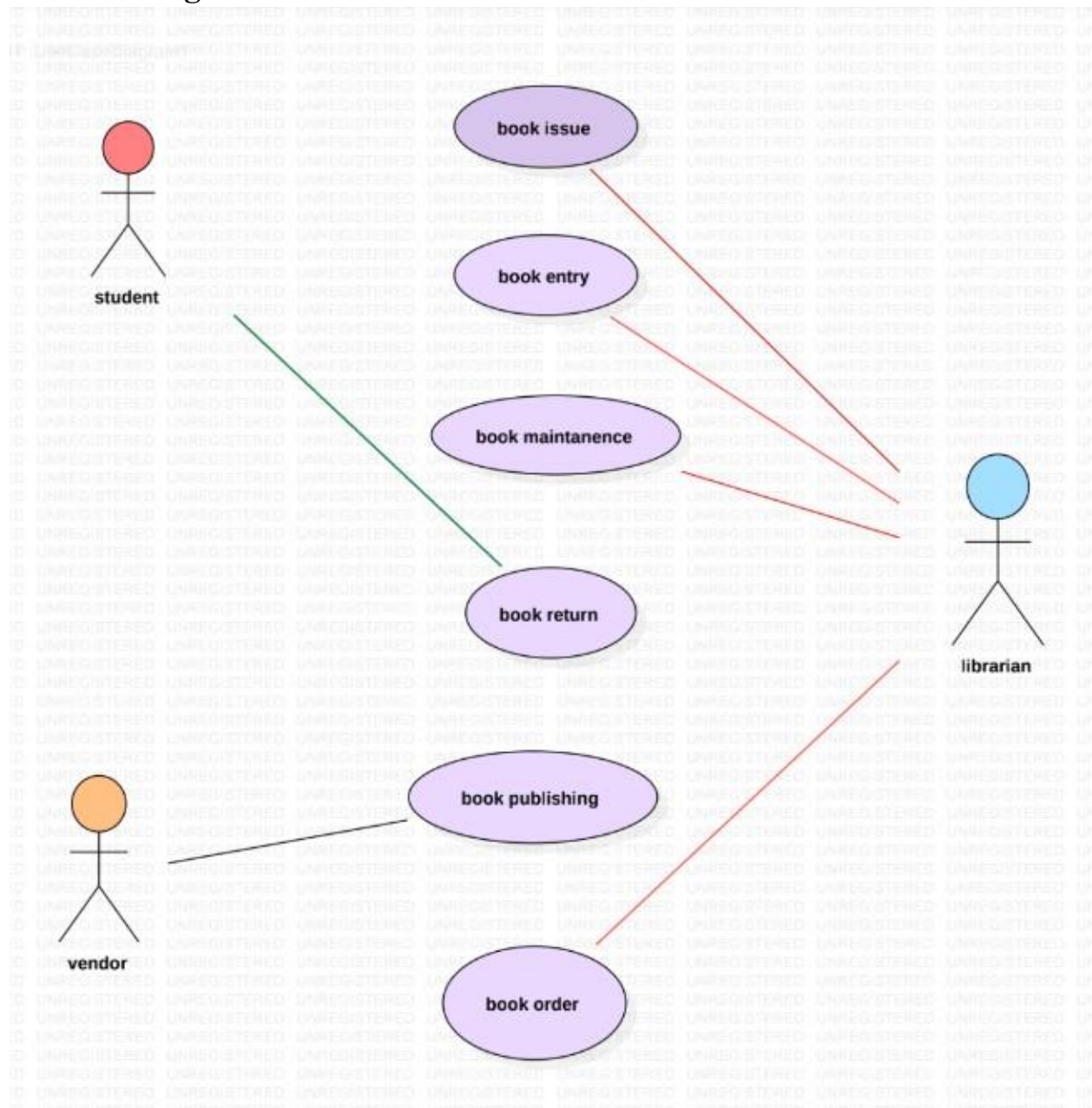5.STATE CHART diagram

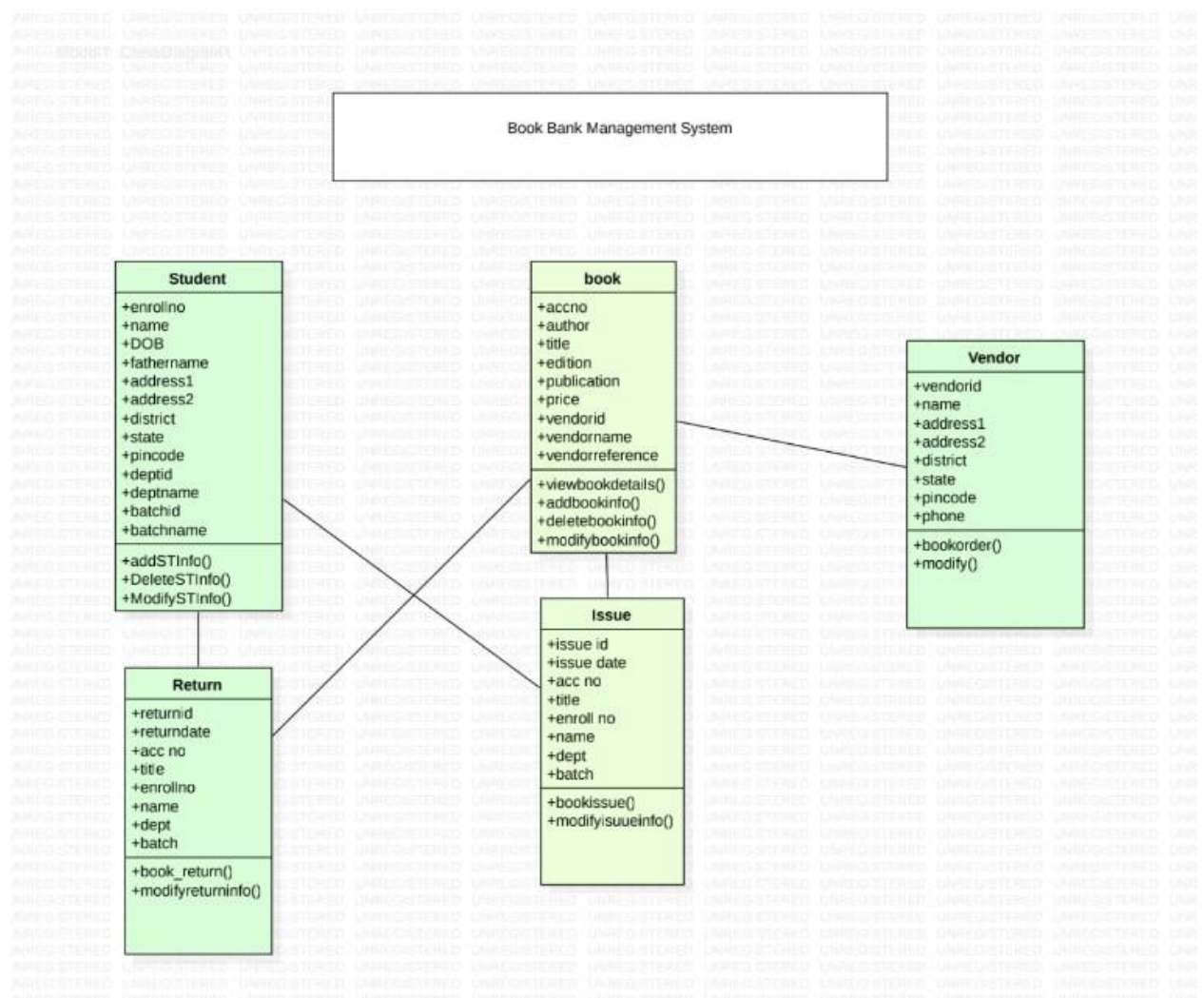6.ACTIVITY diagram

7.COMPONENT diagram

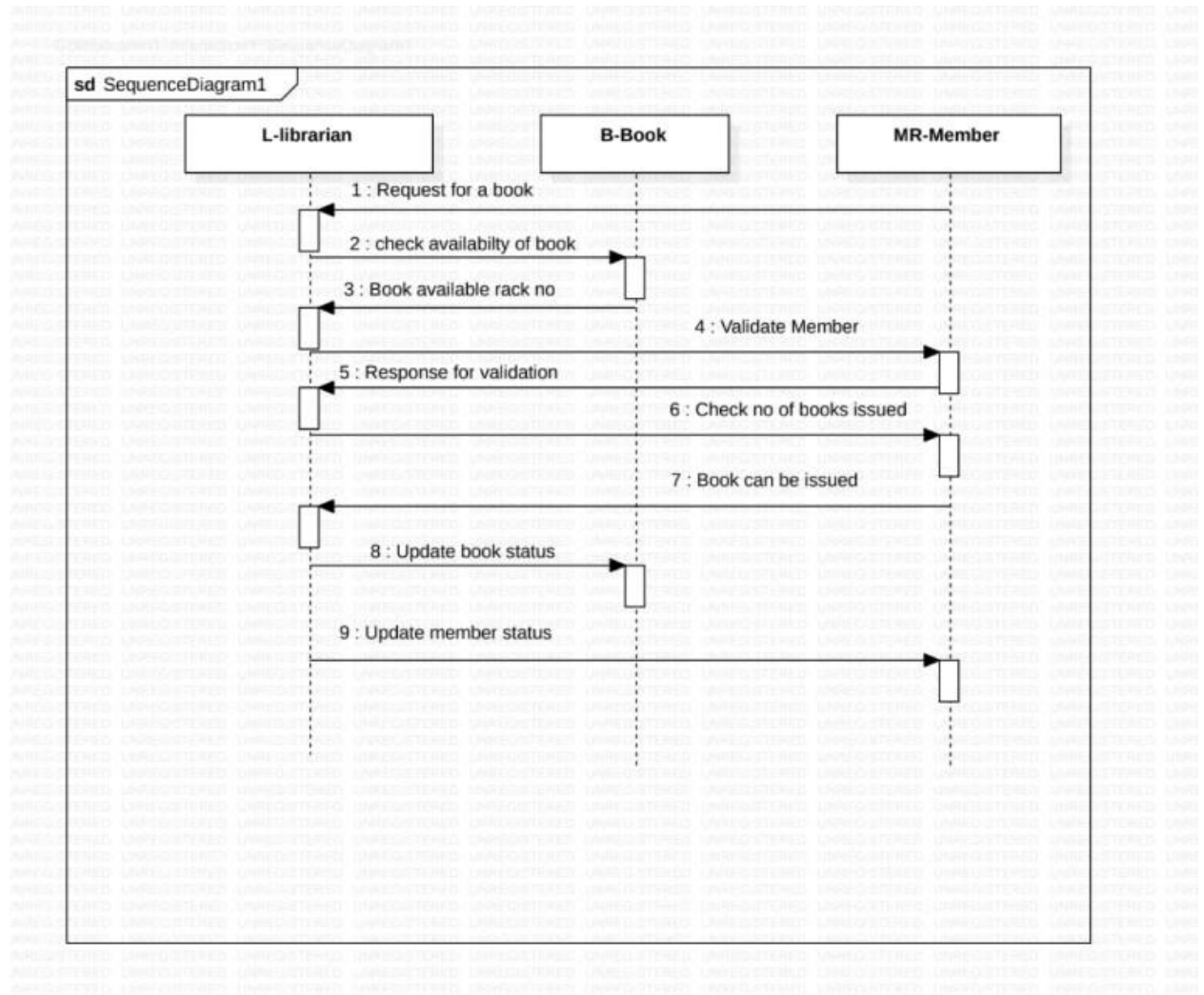8.DEPLOYMENT diagram

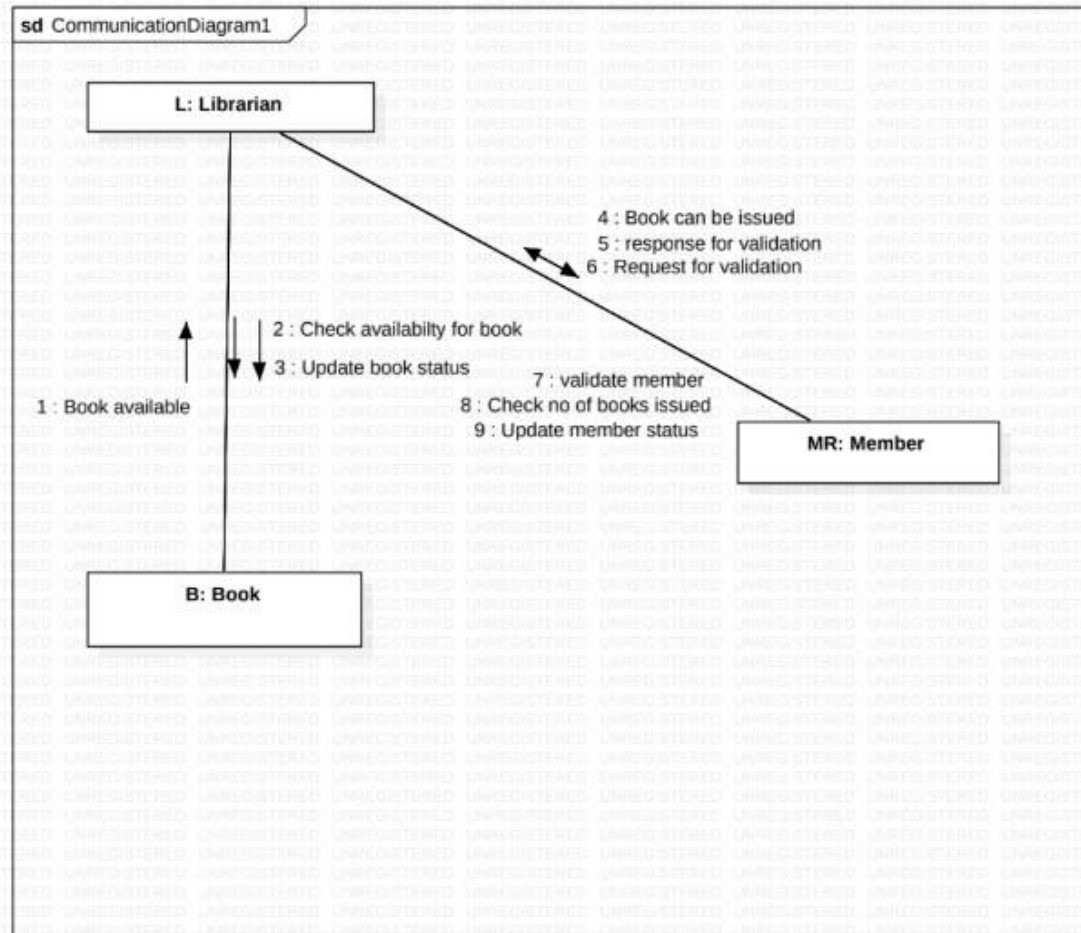9.COMPOSITE STRUCTURE diagram

5.DESIGN

Use case diagram

Class diagram

Book Bank Management System

Student

+enrollno
+name
+DOB
+fathername
+address1
+address2
+district
+state
+pincode
+deptid
+deptname
+batchid
+batchname

+addSTInfo()
+DeleteSTInfo()
+ModifySTInfo()

book

+accno
+author
+title
+edition
+publication
+price
+vendorid
+vendorname
+vendorreference

+viewbookdetails()
+addbookinfo()
+deletebookinfo()
+modifybookinfo()

Vendor

+vendorid
+name
+address1
+address2
+district
+state
+pincode
+phone

+bookorder()
+modify()

Issue

+issue id
+issue date
+acc no
+title
+enroll no
+name
+dept
+batch

+bookissue()
+modifyisuueinfo()

Return

+returnid
+returndate
+acc no
+title
+enrollno
+name
+dept
+batch

+book_return()
+modifyreturninfo()

10

Sequence diagram

sd SequenceDiagram1

L-librarian   B-Book   MR-Member

1 : Request for a book

2 : check availabilty of book

3 : Book available rack no

4 : Validate Member

5 : Response for validation

6 : Check no of books issued

7 : Book can be issued

8 : Update book status

9 : Update member status

11

Collaboration Diagram

sd CommunicationDiagram1

L: Librarian

4 : Book can be issued
5 : response for validation
6 : Request for validation

2 : Check availabilty for book

3 : Update book status

7 : validate member
8 : Check no of books issued
9 : Update member status

1 : Book available

MR: Member

B: Book

12

State chart diagram

The diagram (stm Book) shows states: Available, Issue, Return, Renew.

- Initial state → Available
- Available → Issue: Book issued to User
- Issue → Return: Book no. of days completed or (User want to Return Book)
- Return → Available: Submit the Book
- Return → Issue: Request to Issue Same Book
- Issue → Renew: Book Re-Issue
- Renew → Issue

Activity diagram

Component diagram
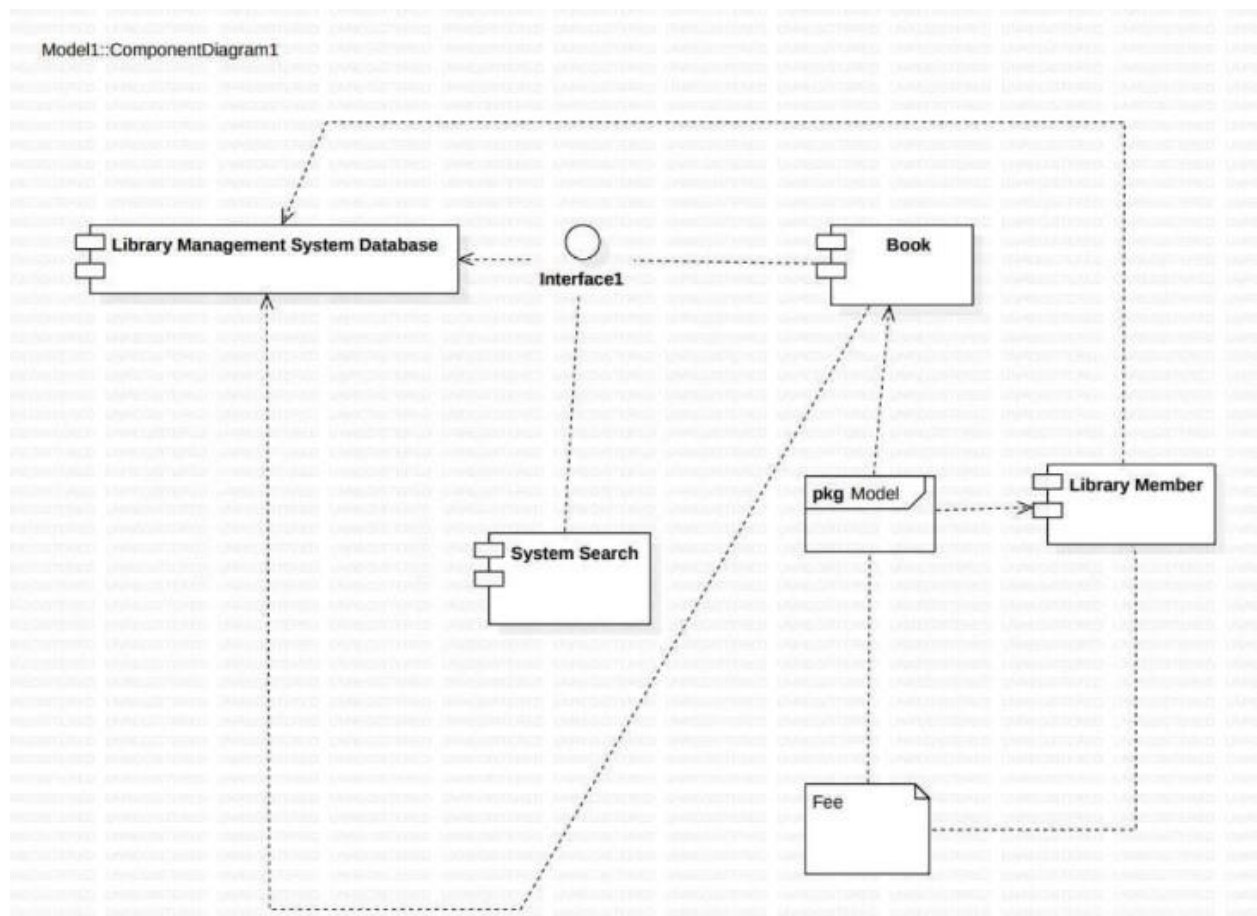
Model1::ComponentDiagram1

6. Test Case

Test cases	Input	Expected output	Actual output
Valid login	Username, password.	Successful	Successful Next page
Invalid login	Username, password.	Failed	Enter valid details, Try again

Stock Maintenance System

1.Problem statement

The process of stock maintenance system is that the customer logins to the particular site to place the order from for the customer product. The stock maintenance system are described sequentially through steps : The customer login to the particular site, they fill the customer details, they plan the orders for their product ,the vendor login and views the customer details and manage orders. Concurrently, vendors play a pivotal role by logging in to the system, granting them access to comprehensive customer details and empowering them to effectively manage incoming orders. This collaborative platform not only enhances customer satisfaction through personalized interactions but also enables vendors to make informed decisions regarding stock levels and order fulfillment, contributing to an overall streamlined and customer-centric stock maintenance experience.

z.

2. Software Requirement Specification Document

2.1 Functional Requirements

In the context of the Stock Maintenance System, the manual handling of the entire inventory management process would lead to significant delays in meeting customer demands. Given the annual increase in the number of products and customer interactions, the implementation of an Automated System is crucial to efficiently address the demand. This system employs diverse programming and database techniques to streamline the intricacies inherent in stock maintenance. Rigorously verified and validated, the system ensures satisfaction by providing an online interface for users to input personal details and submit necessary documents, possibly through scanning. The system not only reduces the workload for authorities overseeing stock management but also expedites the application processing, ensuring a swift and effective response to stock-related requirements.

2.2Tools and Technology Requirements

The following are the list of software requirements we are using to implement:
Application:
- Client Side Technologies: HTML, CSS

16

- 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 Tomcat 8.

2.3 Non-functional Requirements

Performance:

It is the response time, utilization and throughput behavior 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 steps of the product throughout its lifetime. 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 diagram

3.1 Design Document Description

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 modeling 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
- To model a logical database schema.

ii. 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 Hows of Control by Time Ordering.

IV Collaboration Diagram

Also known as a communication diagram is an illustration of the relationships and interaction among software objects in the UML.

- objects: These are shown as rectangles with naming labels inside. The naming label follows the convention of ● object name: class.
- Links: These connect objects with actors and are depicted using a solid line between two elements. Message between objects: these are shown as a labeled arrow placed near a link.

Common Use:

Modeling Flows of Control by Time Ordering.

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.

Activities ultimately result in some action, which is made up of executable atomic computations that result in a change in state of the system or the return of a value. 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

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most

important purpose of a Statechart diagram is to model the lifetime of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams :

- To model the dynamic aspect of a system.
- To model the lifetime of a reactive system.
- To describe different states of an object during its lifetime.
- Define a state machine to model the states of an object.

v. 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

3.2 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 tests.

Each test type addresses a specific testing requirement.

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.

All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .

It is done after the completion of an individual unit before integration. 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 satisfactory, 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.

iii. 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 Exercise internal

  data structures to ensure their validity

- **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 a 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 database access
- Behavior or performance errors
- Initialization and termination errors.

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. 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

3.4 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 **Risk Table:**

Risk	Category	Probability(%)	Impact	
Size estimation may be significantly low	PS	**60**	**2**	
Delivery deadline will be tightened	BU	**50**	**2**	

Customer will the requirements	PS	**80**	2	
Technology will not meet expectations	TE	**30**	1	
Lack of tracking on tools	DE	**80**	3	
Inexperienced staff	ST	**30**	2	

ST-staff size and experience risk

BU-business risk

TE-technology risk

PS-project size risk

1-catastrophobia

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 tools
STAR UML:

Star uml is open source software modeling tool that supports the uml (unified modeling language)Framework for a system and software modeling .It is based on the uml version 1.4,provides different types of diagrams and it accepts UML 2.0 notation.I actively supports MDA(Model Driven Architecture)Approach by supporting UML profile concepts and allowing to generate code for multiple languages.
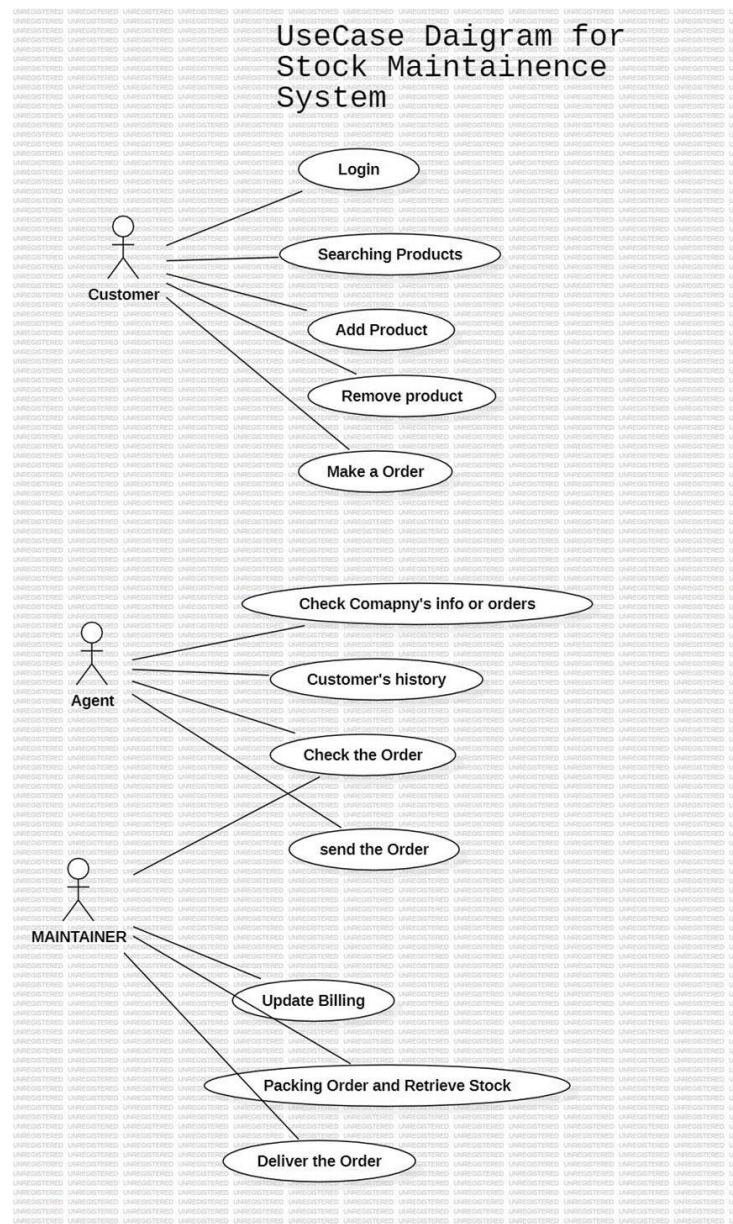
UML supports the following diagrams:

1.USE CASE diagrams

2.CLASS diagrams

3.SEQUENCE diagram

4.COLLABORATION diagram

5.STATE CHART diagram

6.ACTIVITY diagram

7.COMPONENT diagram

8.DEPLOYMENT diagram

9.COMPOSITE STRUCTURE diagram5.DESIGN

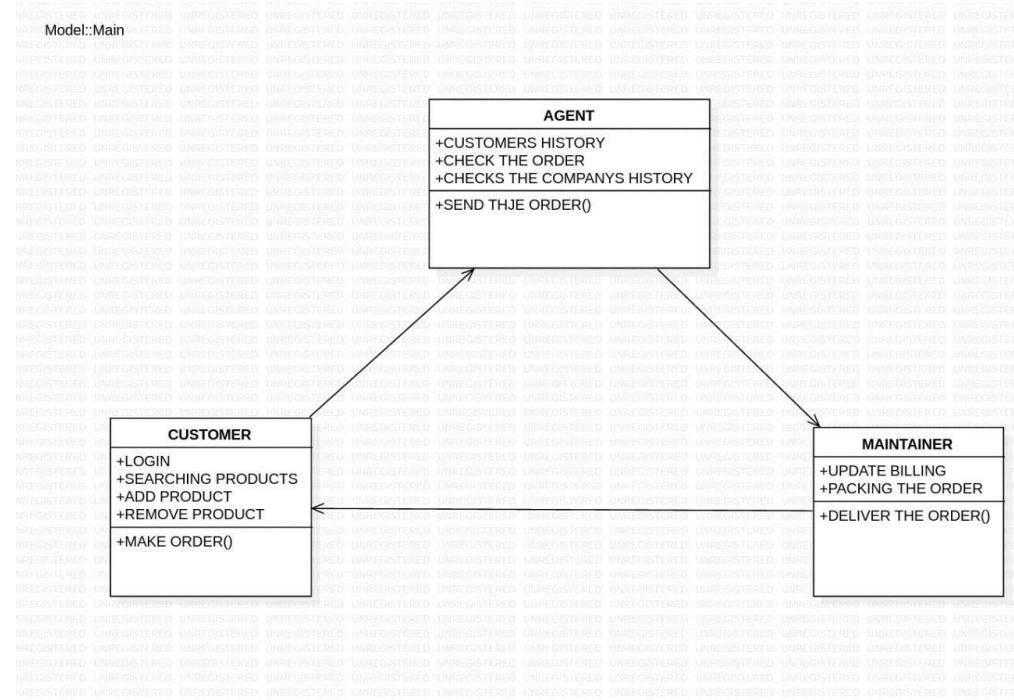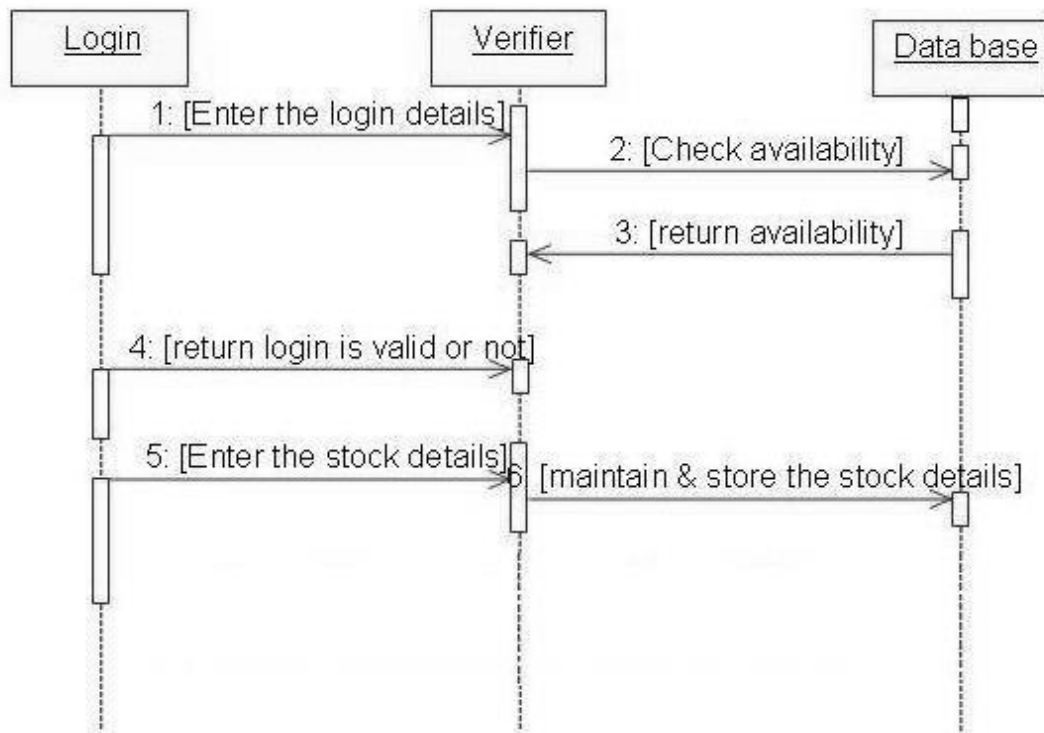# 5.Design

USE CASE DIAGRAM:

UseCase Daigram for Stock Maintainence System

Customer
- Login
- Searching Products
- Add Product
- Remove product
- Make a Order

Agent
- Check Comapny's info or orders
- Customer's history
- Check the Order
- send the Order

MAINTAINER
- Update Billing
- Packing Order and Retrieve Stock
- Deliver the Order

CLASS DIAGRAM :

Model::Main

AGENT

+CUSTOMERS HISTORY
+CHECK THE ORDER
+CHECKS THE COMPANYS HISTORY

+SEND THJE ORDER()

CUSTOMER

+LOGIN
+SEARCHING PRODUCTS
+ADD PRODUCT
+REMOVE PRODUCT

+MAKE ORDER()

MAINTAINER

+UPDATE BILLING
+PACKING THE ORDER

+DELIVER THE ORDER()

26

SEQUENCE DIAGRAM :

COLLABORATION DIAGRAM

Login

1 : Enter the login details
4 : return login is valid or not
5 : enter the stock details

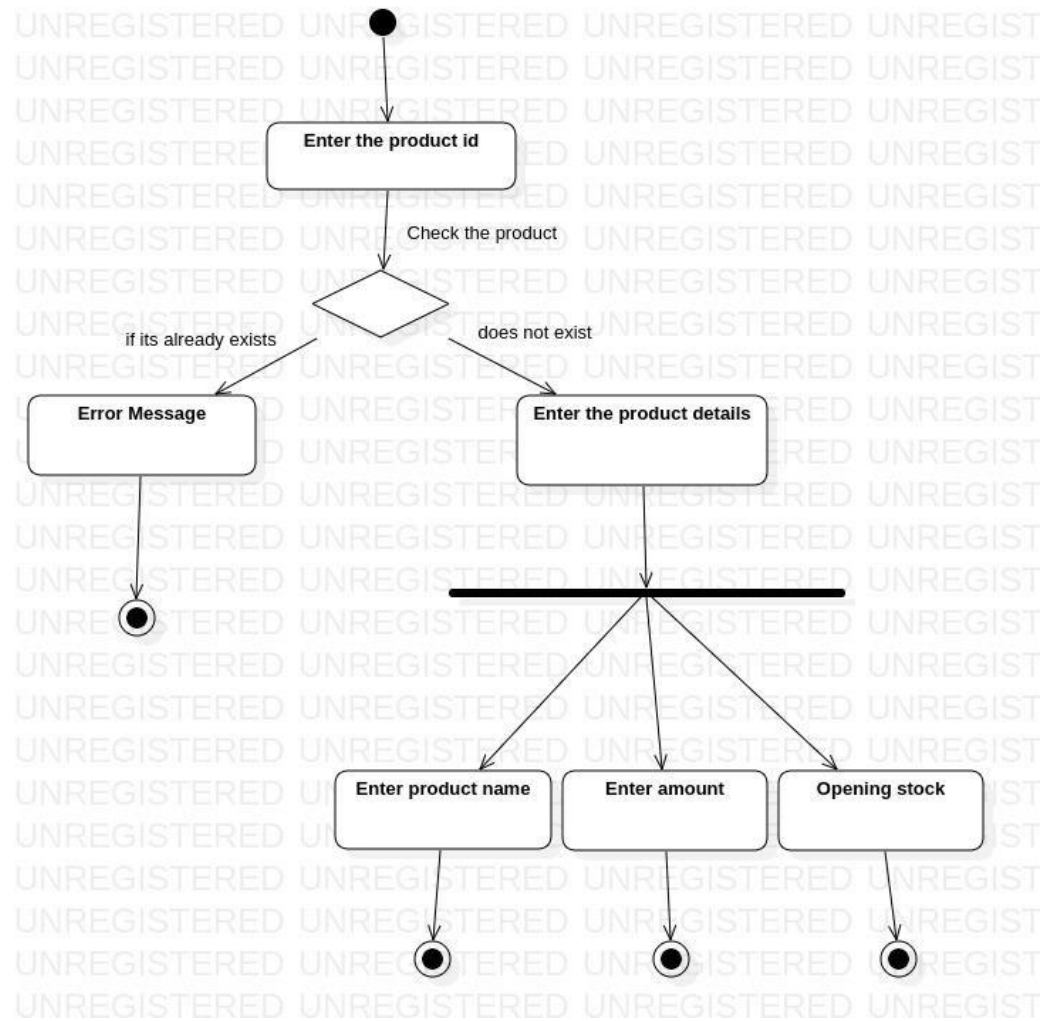Verifier

3 : return avaliability

DataBase

2 : Check availability
6 : maintain and store the stock details

ACTIVITY DIAGRAM

29
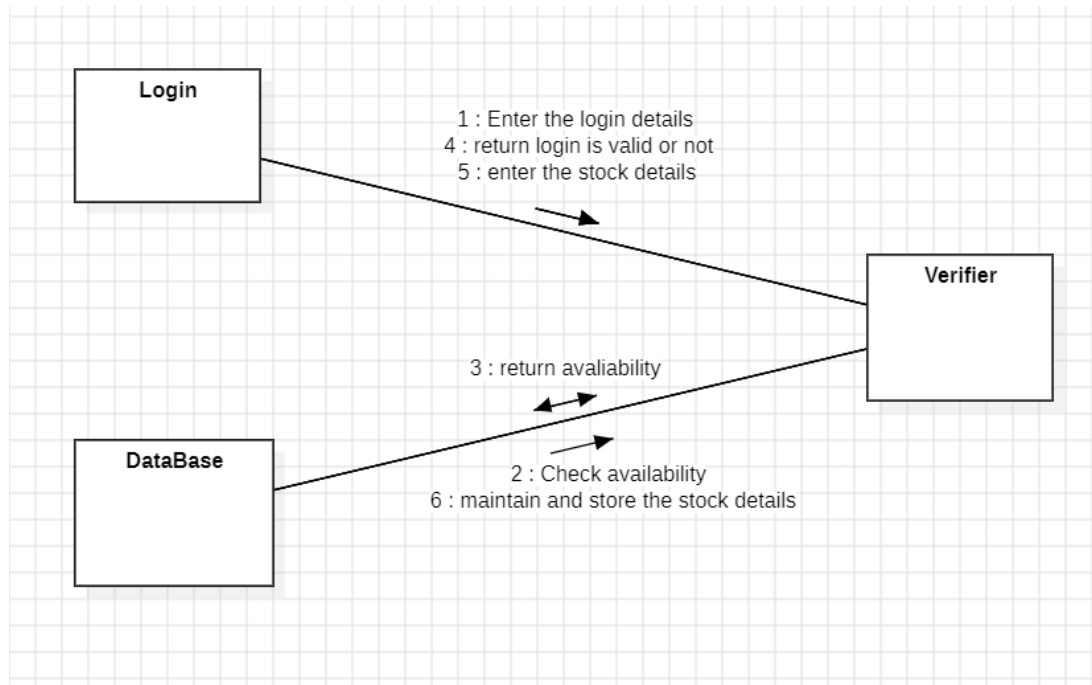
STATECHART DIAGRAM

6. Test Case

Test cases	Input	Expected output	Actual output
Valid login	Username, password.	Successful	Successful
Invalid login	Username, password.	Failed	Failed,Enter valid details
Enter Product details	Product Name	Successful	Successful

Payment	Requesting Payment	Successful	Successful

—--END—-