VISHNU
UNIVERSAL LEARNING
BVRIT H
Estd. 2012

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)

*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 poorin 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. Thebook bank can easily create, replace and delete information about the tiles, members, loans andreservations 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 theirpersonal details and submit the necessary documents (may be by scanning). Theauthority 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

Performance:

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 theproduct throughout its life time. All development will be provided with gooddocumentation.

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 amodel. 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 thesystem 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

• 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 objectssend and receive along the Y axis, in order of increasing time from top to bottom. This gives thereader a clear visual cue to the flow of control over time. Sequence diagrams have two features that distinguish them from collaborationdiagrams.

- 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, thinrectangle 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. 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 orthe return of a value. Actions encompass calling another operation, sending a signal, creatingor 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 itscontent.

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 twoways.

- To model a workflow
- To model an operation

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 anode. Component diagrams are used to visualize the organization and relationships amongcomponents 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 theprocess 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 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 softwareunits 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 expectedresults.

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

• 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 thatwill 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.

3.3 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 ManagementSystem 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

• 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

PS – Project size risk

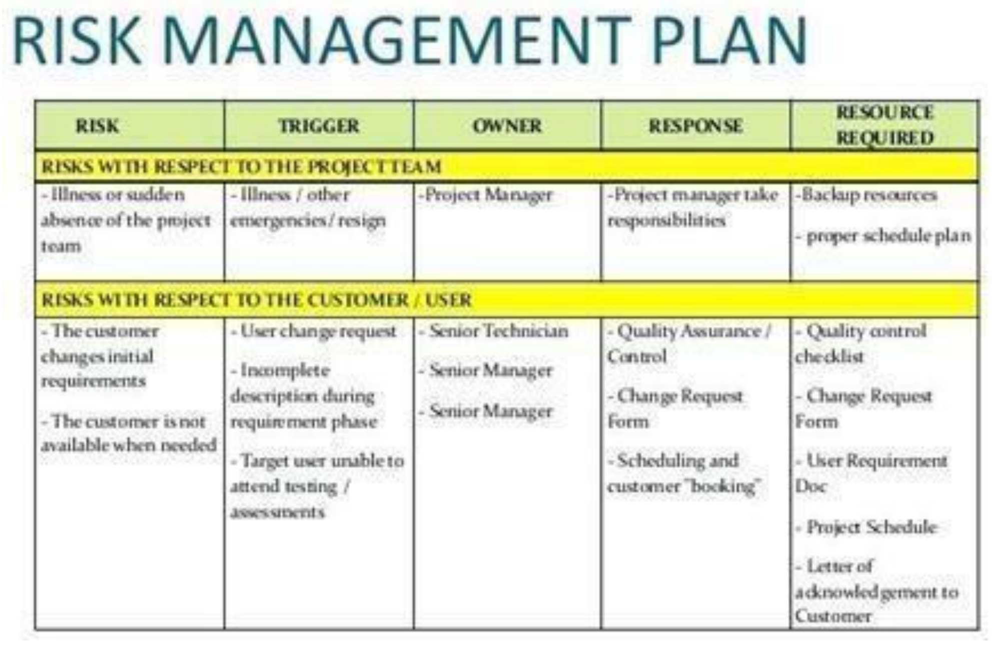
TE- Technology risk

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:

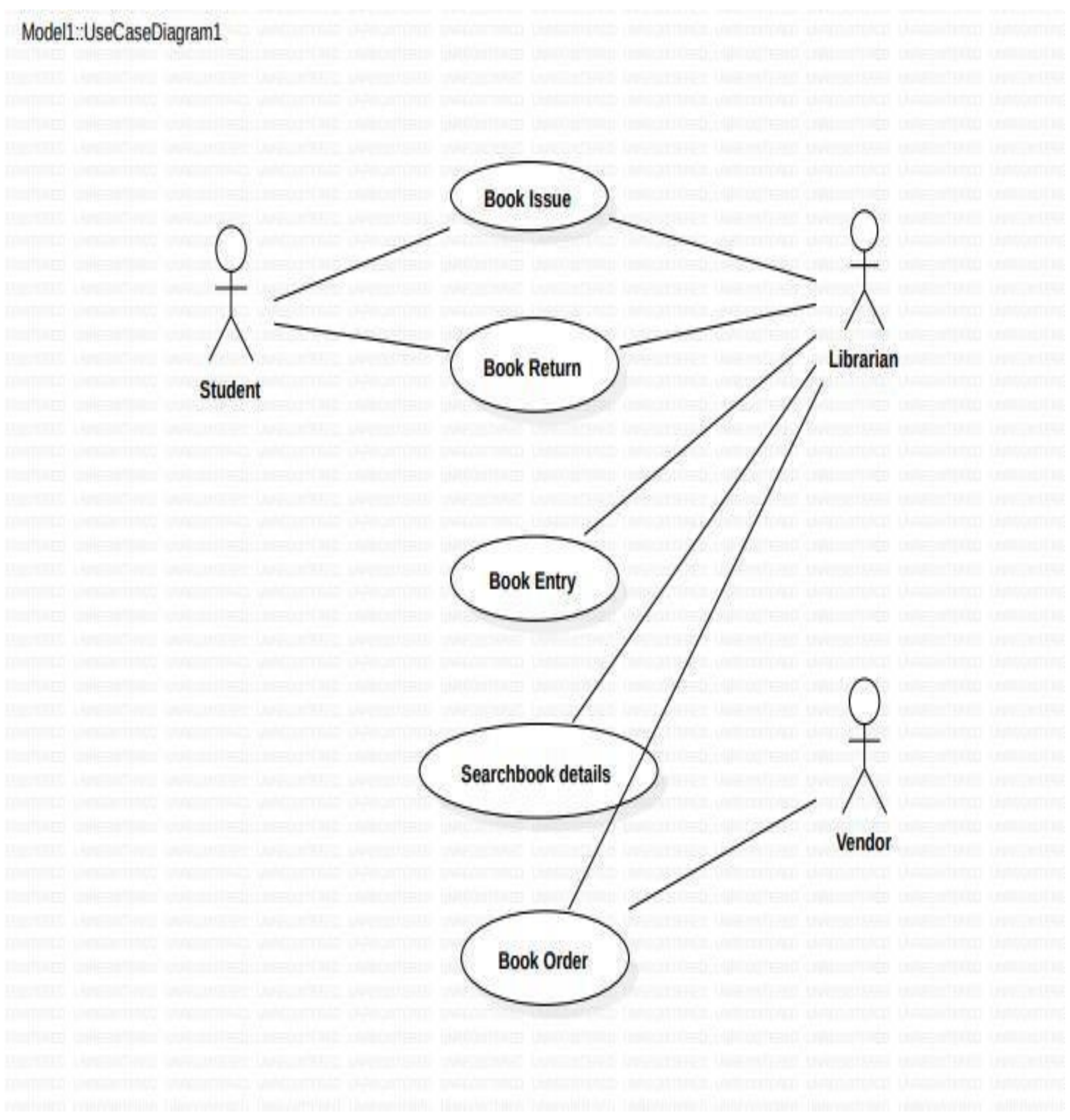
StarUML is an open source software modeling tool that supports the UML (Unified Modeling Language) framework for system and software modeling. It is based on UML version 1.4, provides different types of diagram and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture)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

StarUML supports the following diagram types:

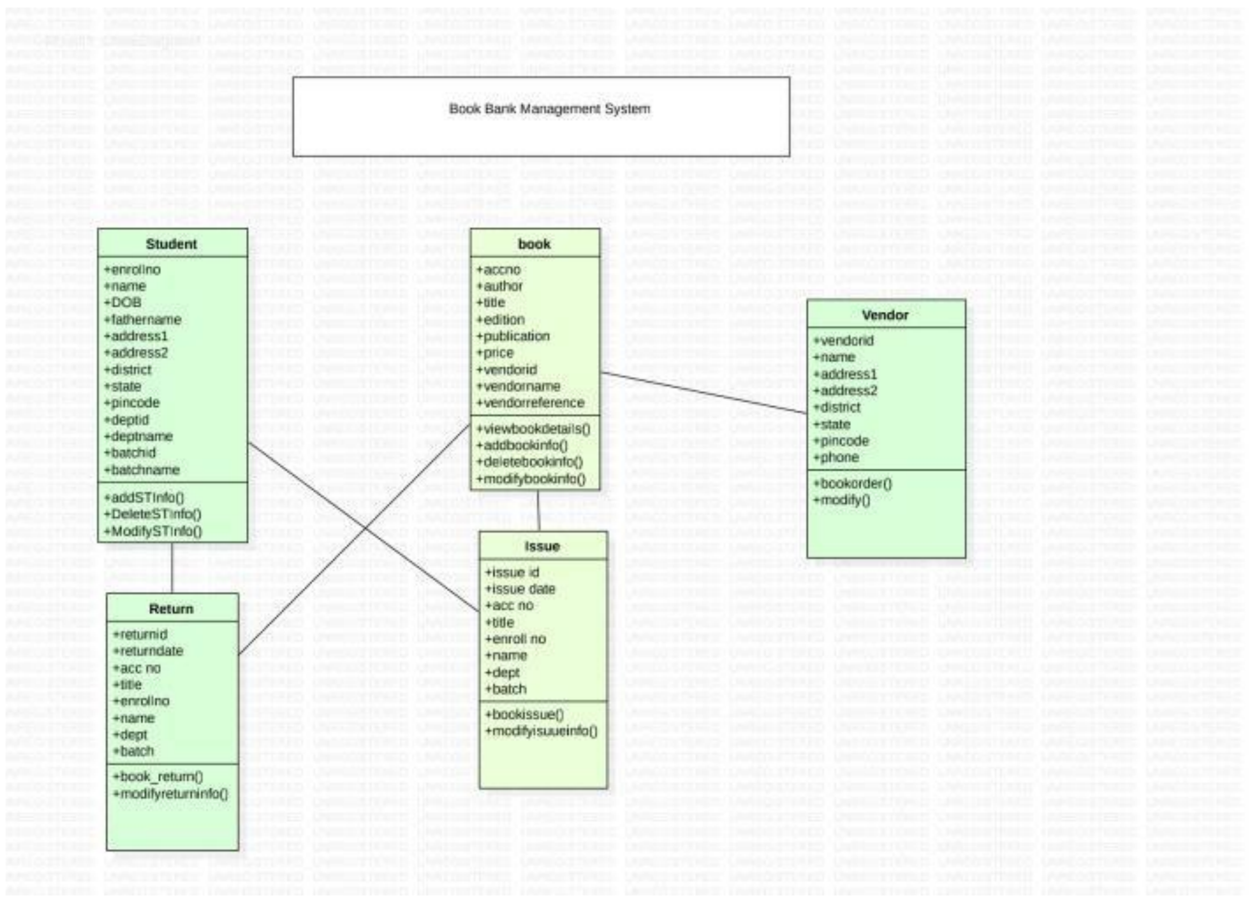
1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. Collaboration Diagram
5. State chart Diagram
6. Activity Diagram
7. Component Diagram
8. Deployment Diagram
9. Composite Structure Diagram

5. Design

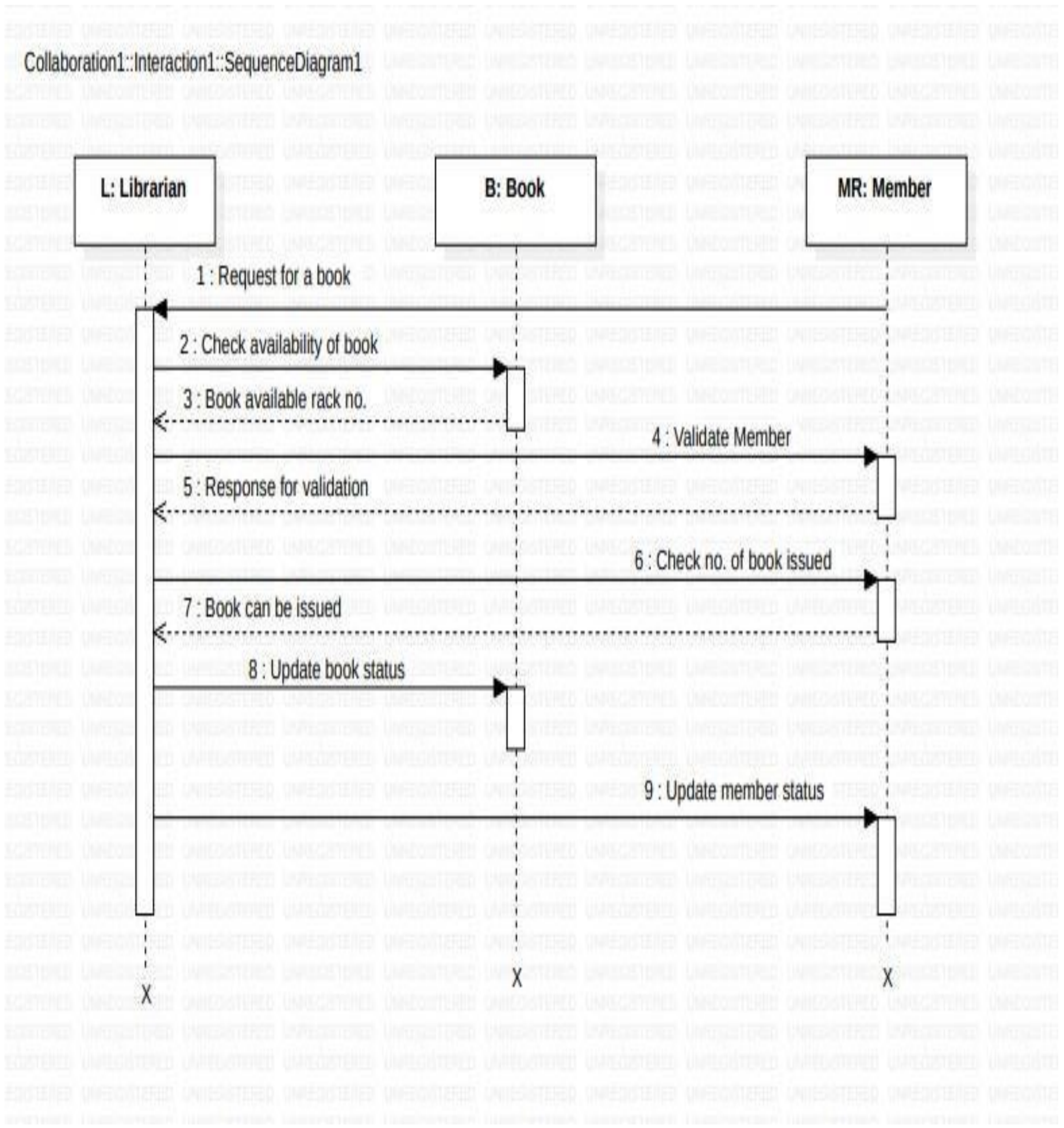
Use case diagram:



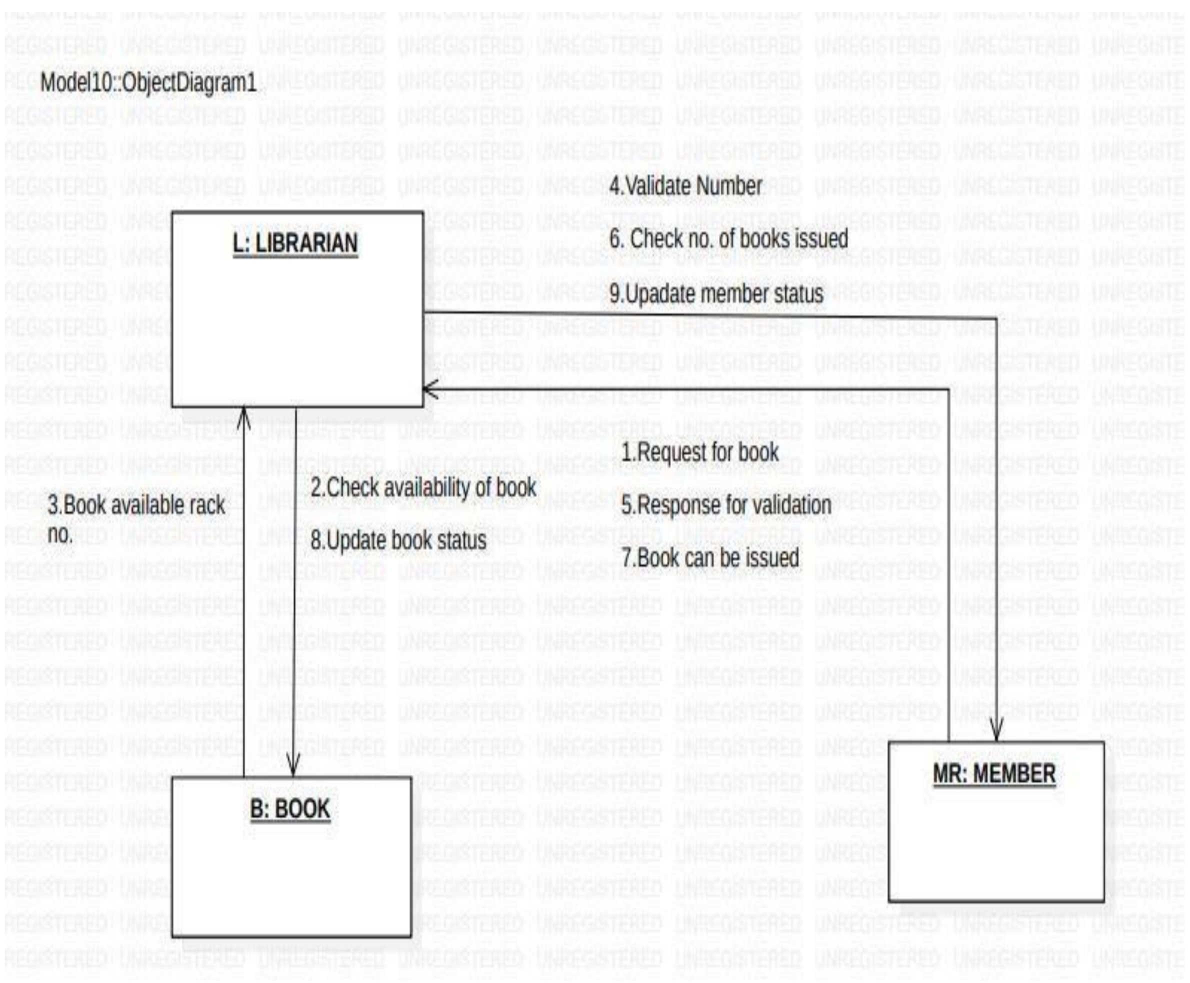
Class diagram:



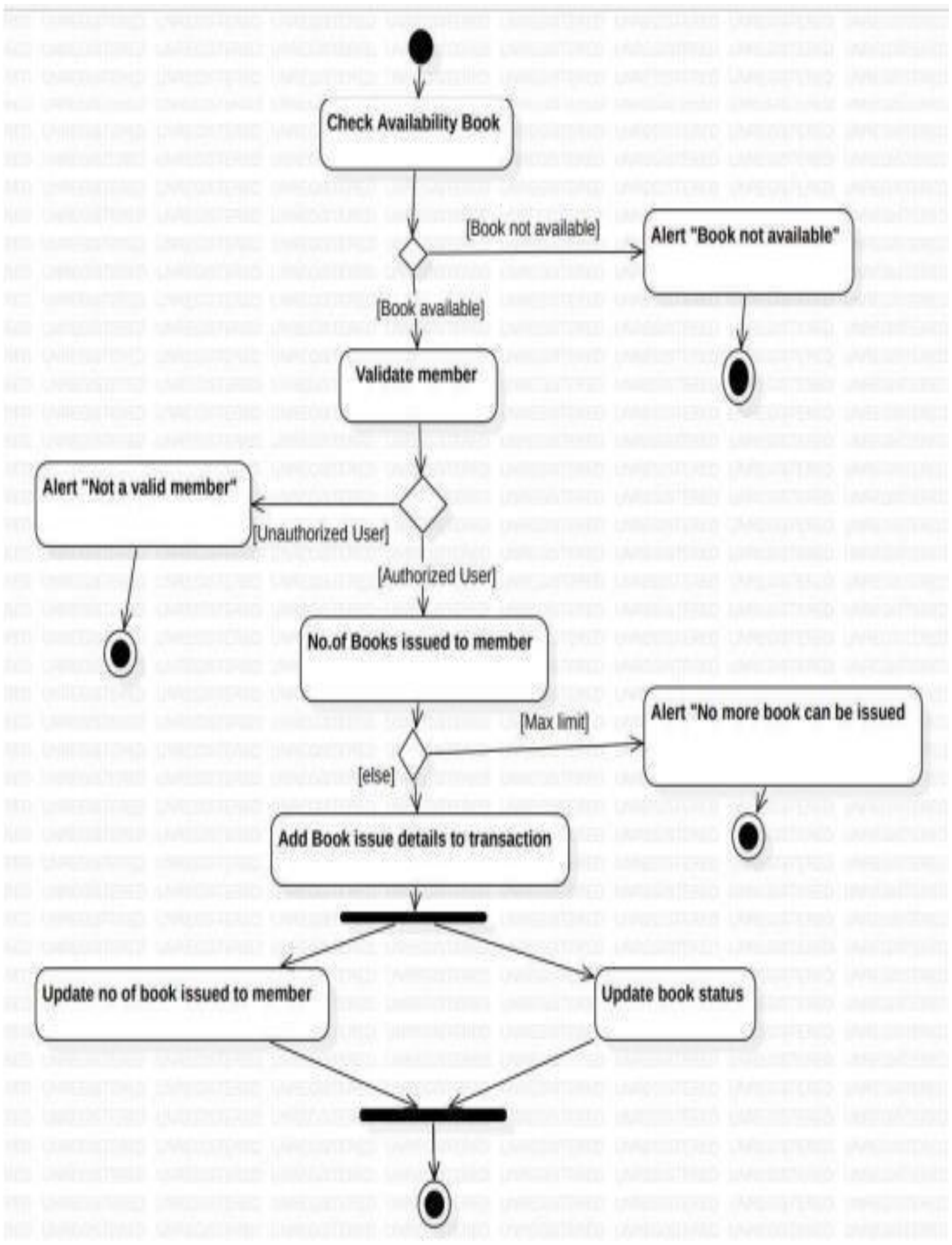
Sequence diagram:


Collaboration1::Interaction1::SequenceDiagram1
L: Librarian
B: Book
MR: Member
1 : Request for a book
2 : Check availability of book
3 : Book available rack no.
4 : Validate Member
5 : Response for validation
6 : Check no. of book issued
7 : Book can be issued
8 : Update book status
9 : Update member status
X
X
X

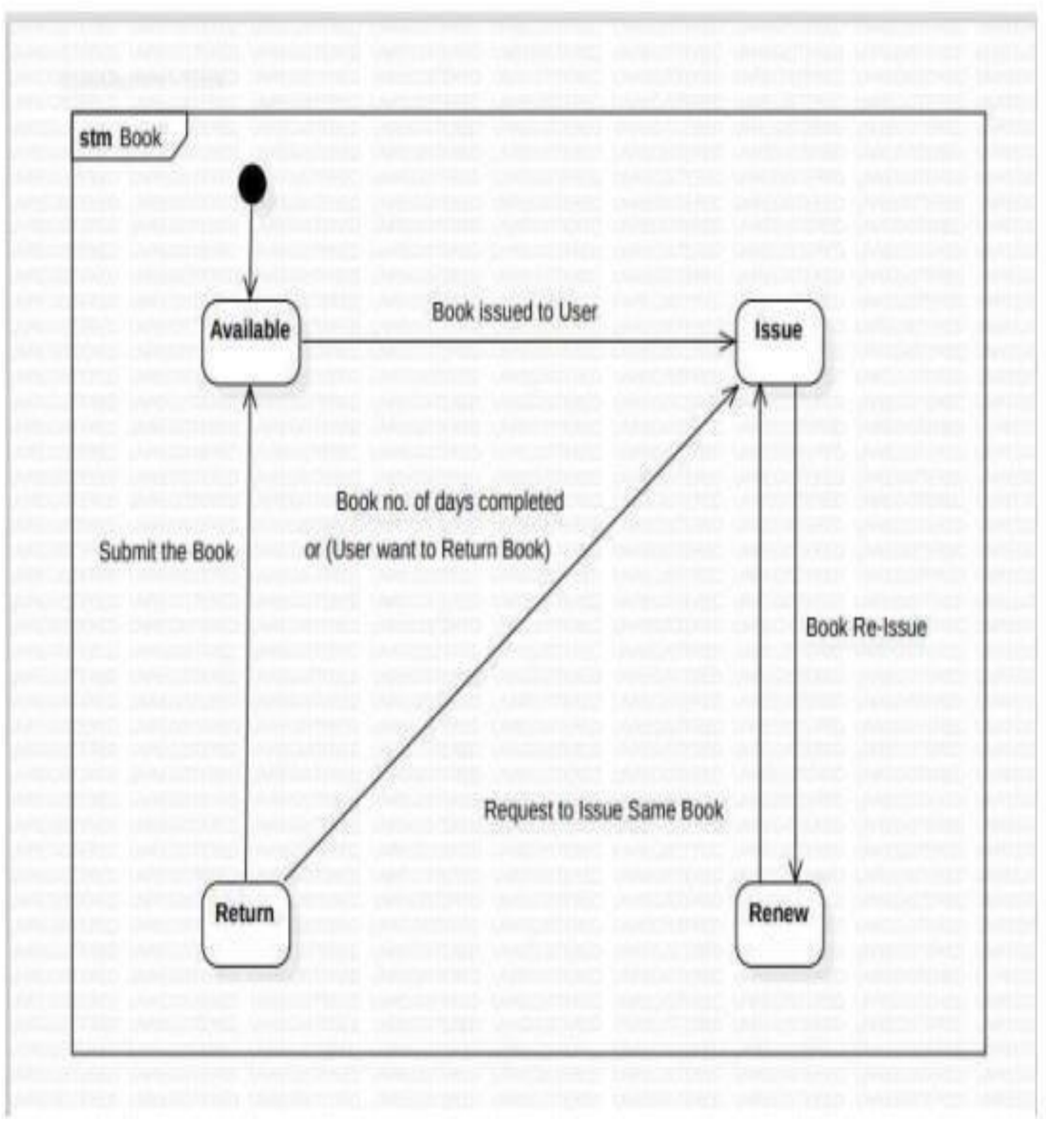
Collaboration diagram:



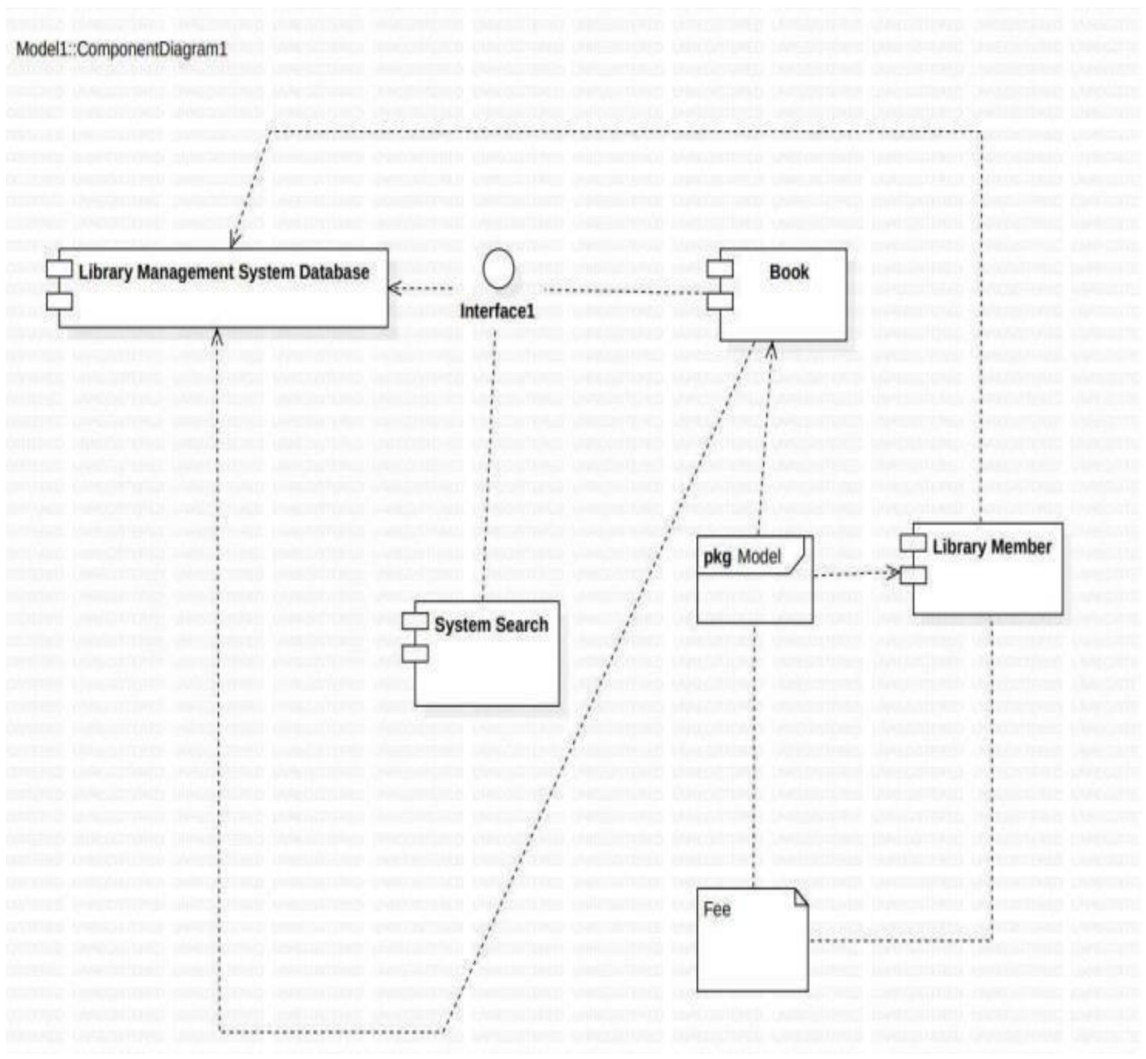
Activity diagram:


Check Availability Book
[Book not available]
Alert "Book not available"
[Book available]
Validate member
Alert "Not a valid member"
[Unauthorized User]
[Authorized User]
No.of Books issued to member
[Max limit]
Alert "No more book can be issued
[else]
Add Book issue details to transaction
Update no of book issued to member
Update book status

Statechart diagram:


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

Component diagram:



6. Test Cases

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

PASSPORT AUTOMATION SYSTEM

1. Problem statement

Passport automation system is an interface between applicant and authority which issues the passport. The system adopts a comprehensive approach to minimize the manual work and time. The applicant has to enter the details (i.e., personal, official etc.). These details will be transferred to the database server. This forms the first step in processing of passport application after first round of verification. This information is forwarded to regional administrators, ministry of external affairs office. The police verification is done manually and report is transferred to administrators. If everything is okay, then passport is issued or else it is cancelled. After issuing the passport, the original information is added to the database and stored permanently.

2. Software Requirement Specification Document

2.1 Functional Requirements

▪ It is defined as how they should react in the particular input and how the system should react in the particular situations and what the system do not do.

▪ In this project, login as functional requirement. In that functional requirement we may check the user name and password is correct or not. After checking entity of login, we can show the detail based on the type of actor.

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 : Ubuntu
- Documentation : LibreOffice Writer

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 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 theproduct throughout its life time. All development will be provided with gooddocumentation.

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 amodel. 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 thesystem 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
- 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 yourdiagram, 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 objectssend and receive along the Y axis, in order of increasing time from top to bottom. This gives thereader a clear visual cue to the flow of control over time. Sequence diagrams have two features that distinguish them from collaboration.

- First, there is the object lifeline. An object lifeline is the vertical dashed linethat 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 directlyor through a subordinate procedure.

Content:

Sequence diagrams commonly contain

- Objects
- Links
- Messages

Common Use:

Modelling Flows of Control by Time Ordering

iv. Collaboration Diagrams

A 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. 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 destroyingan 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 Use:

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
- T model an operation

vi. State Chart Diagrams

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.

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 anode.Component diagrams are used to visualize the organization and relationships amongcomponents 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 test. 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 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 atleast 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 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.

3.3 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 ManagementSystem 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
- Catastropic

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

PS – Project size risk

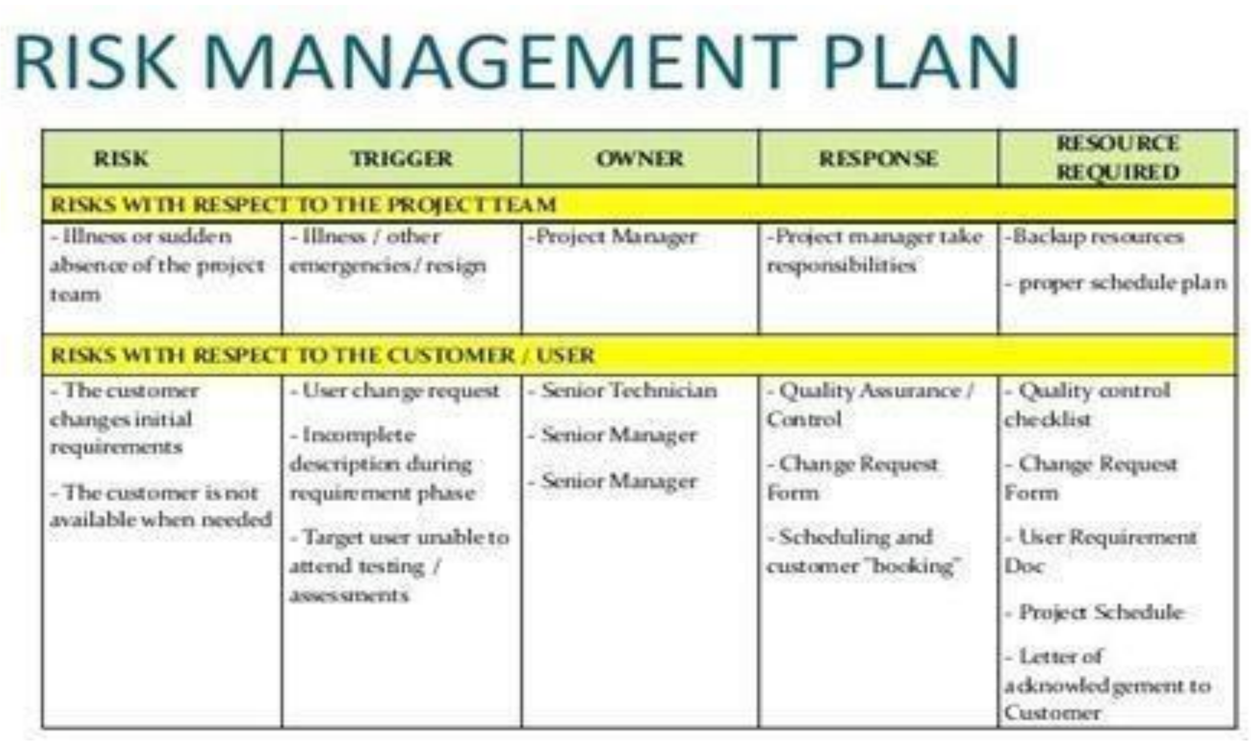
TE- Technology risk

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

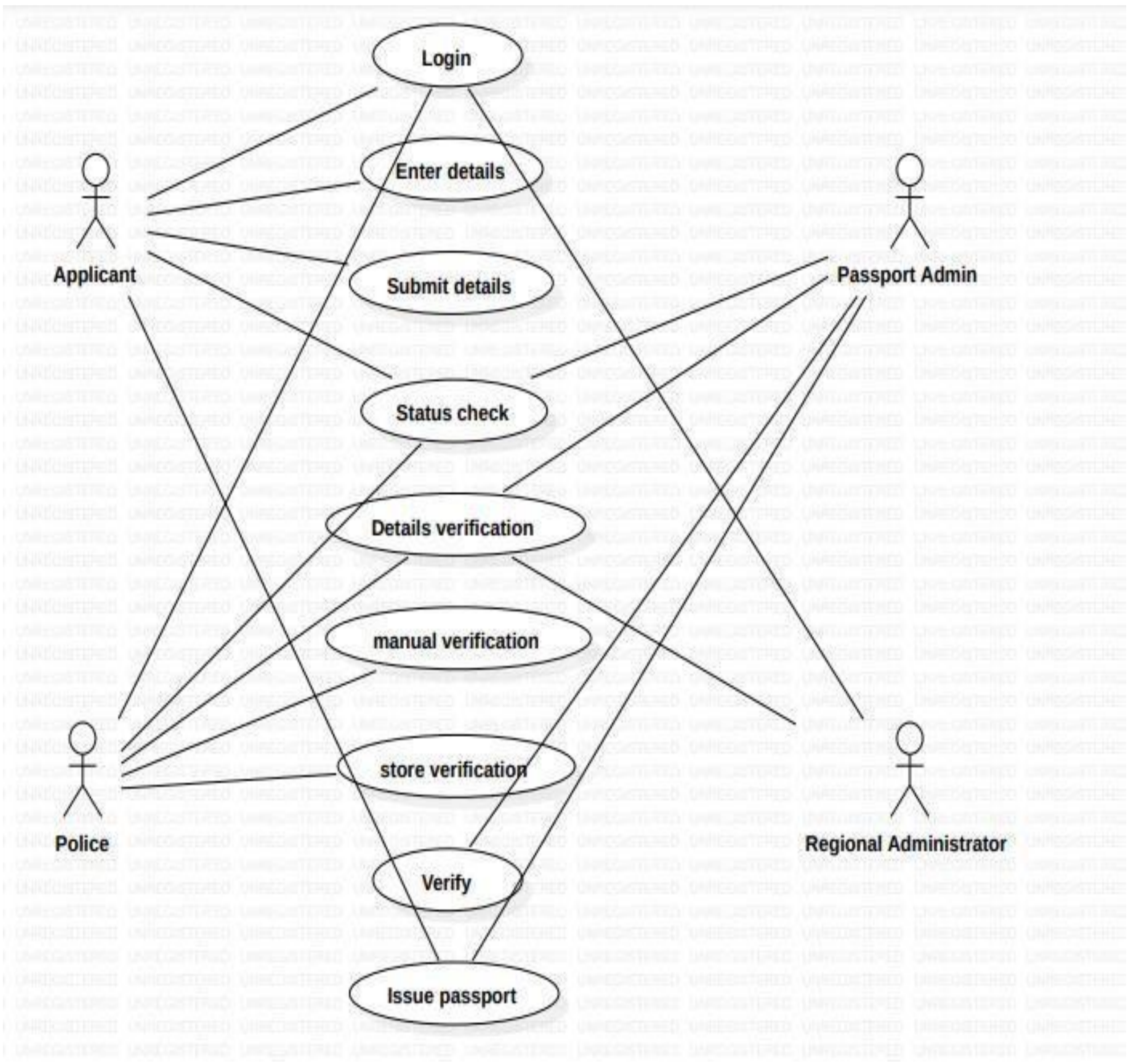
StarUML:

StarUML is an open source software modelling tool that supports the UML (Unified Modelling Language) framework for system and software modelling. 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 profileconcept and allowing to generate code for multiple languages. StarUML supports the following diagram types:

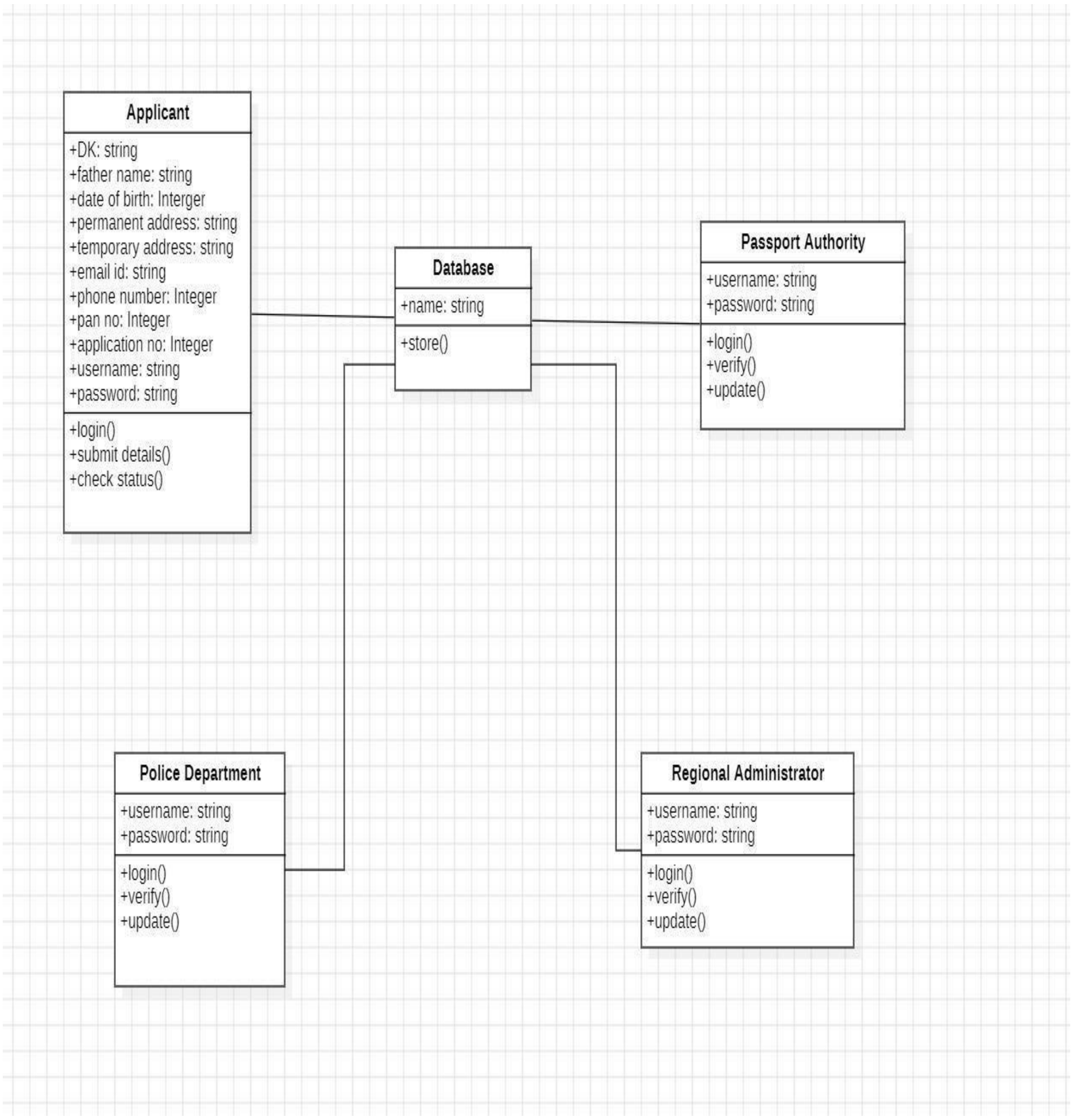
1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. Collaboration Diagram
5. State chart Diagram
6. Activity Diagram
7. Component Diagram
8. Deployment Diagram
9. Composite Structure Diagram.

5. Design

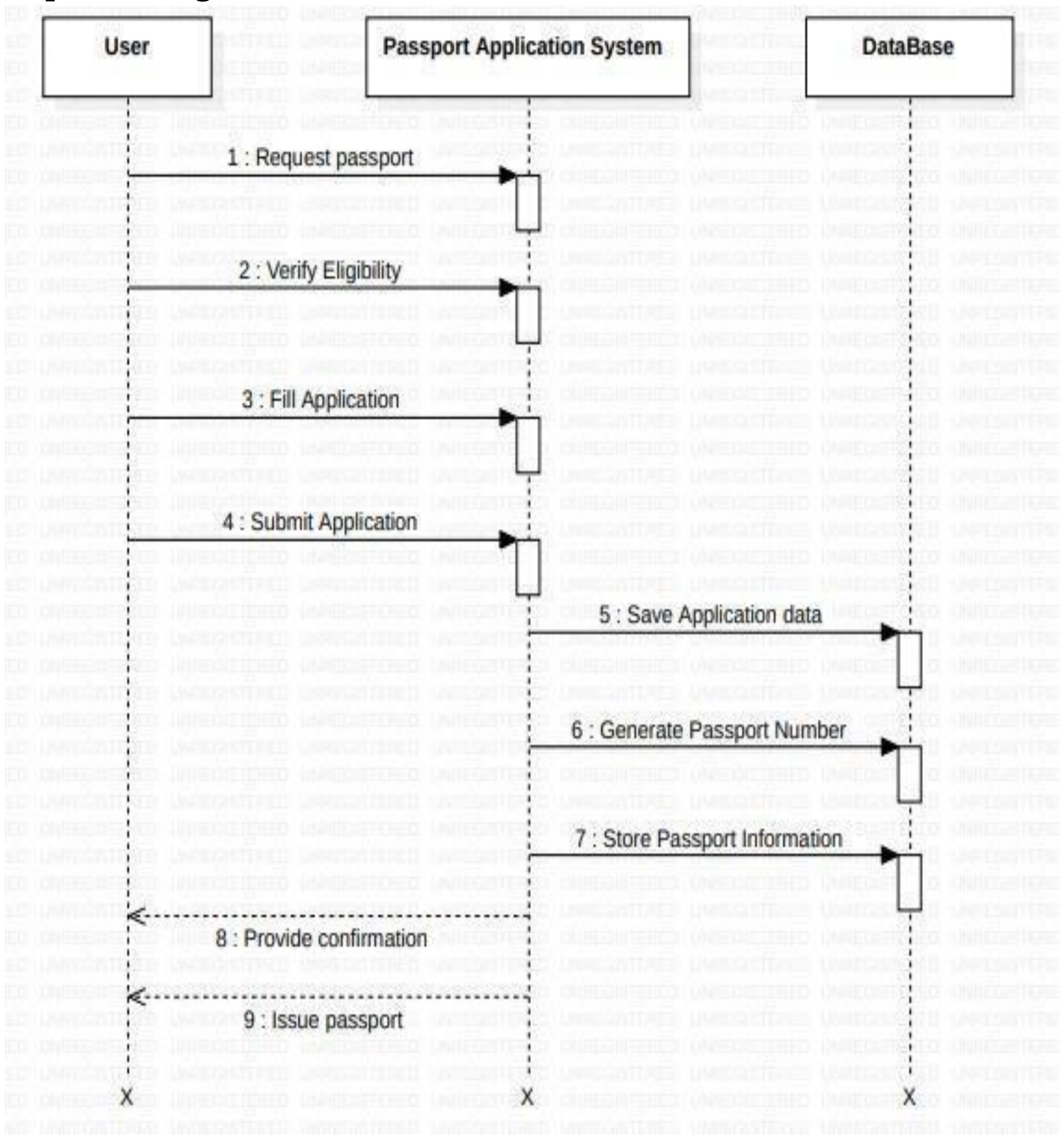
Use case diagram:



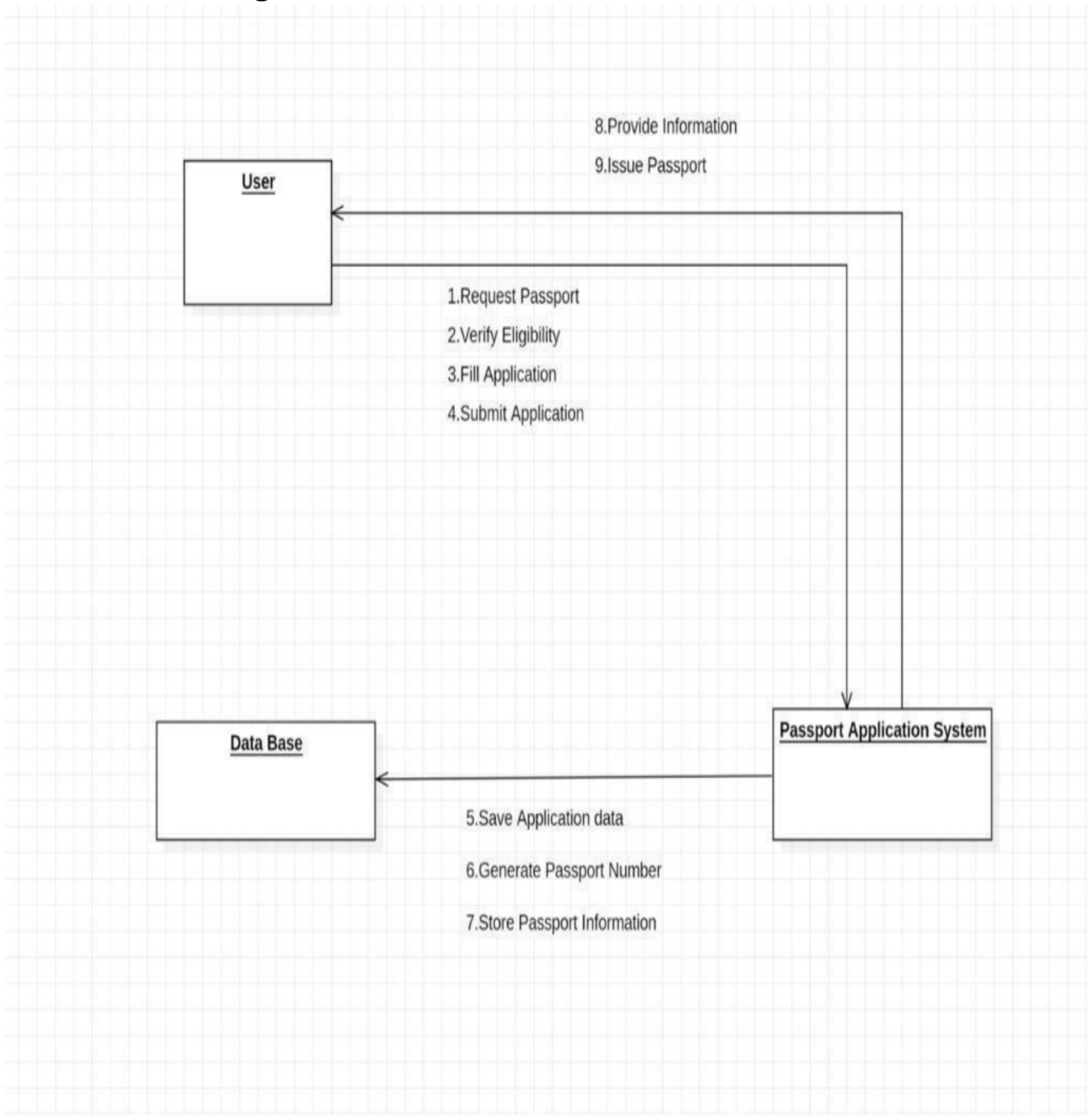
Class diagram:



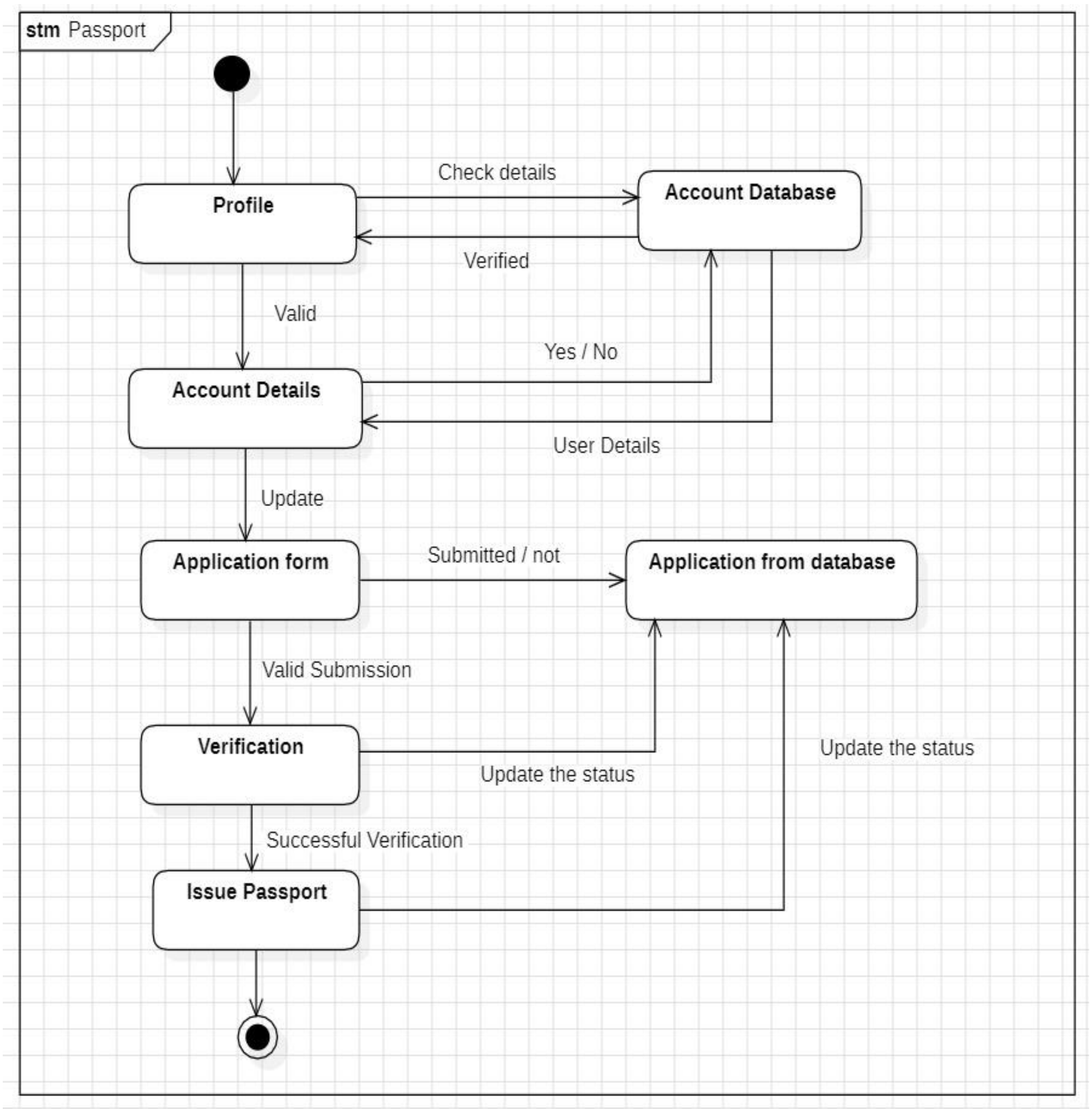
Sequence diagram:



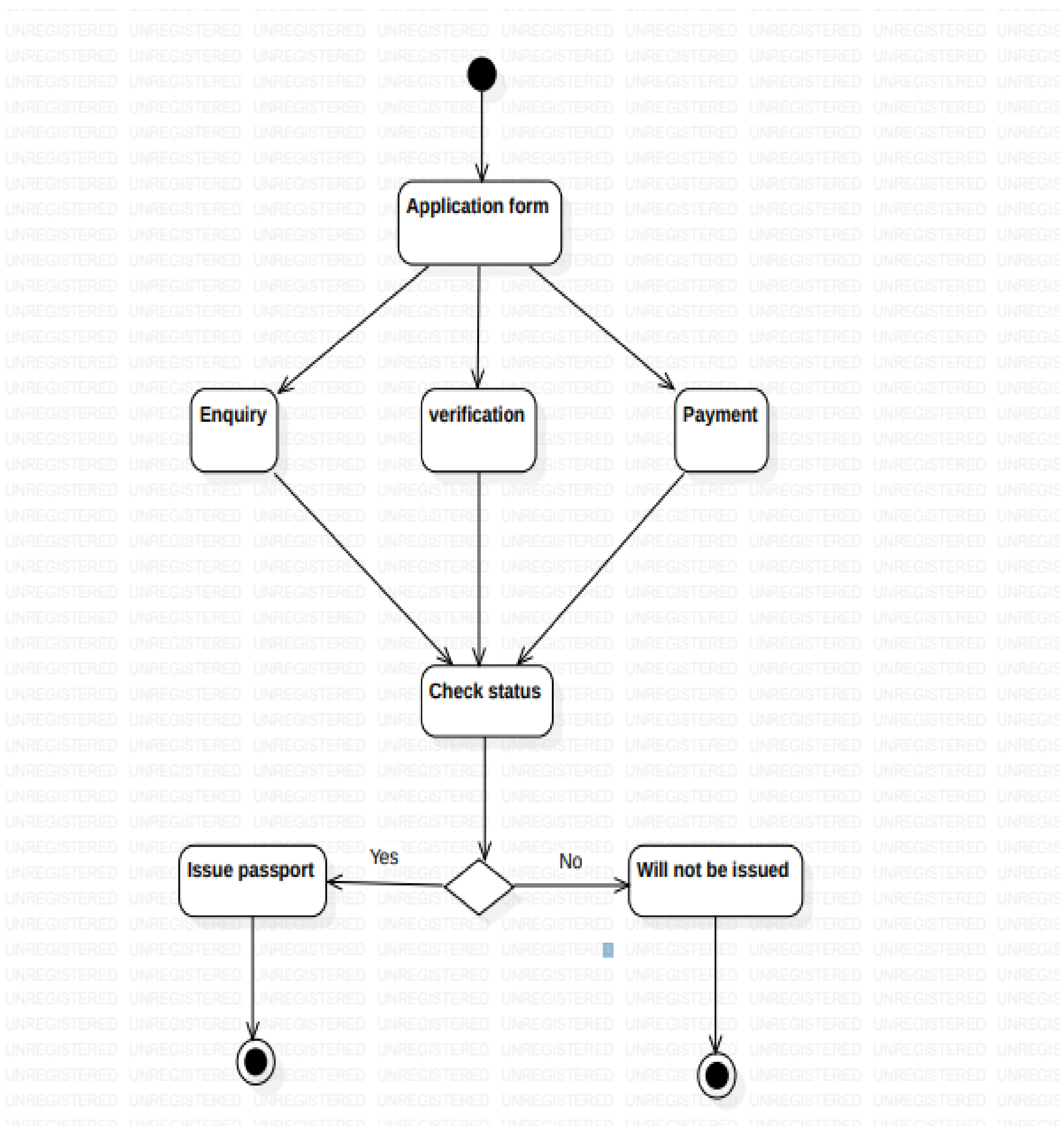
Collaboration diagram:



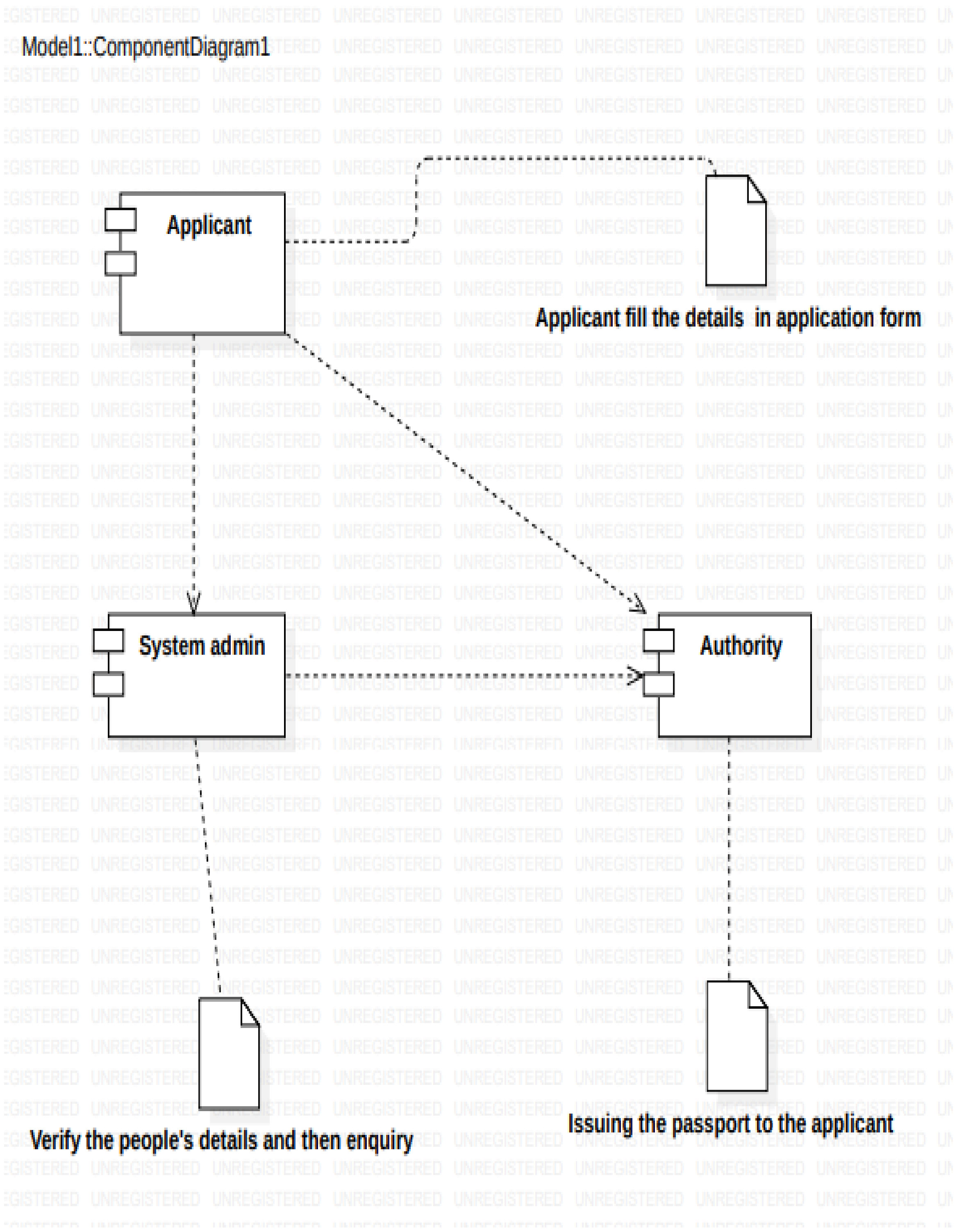
Statechart diagram:



Activity diagram:



Component diagram:


Model1::ComponentDiagram1
Applicant
Applicant fill the details in application form
System admin
Authority
Verify the people's details and then enquiry
Issuing the passport to the applicant

6. Test cases

Input	Expected Output	Actual Output
Valid Passport Application Form	Success	Success
Incomplete Passport Application Form	Error	Error
Invalid Passport photo format	Error	Error
Passport Application with Expired documents	Error	Error
Renewal request with valid documents	Success	Success
Renewal request with invalid documents	Error	Error
Pickup request with valid reference number	Ready	Ready
Pickup request with invalid reference number	Error	Error
Delivery request with valid address	Success	Success
Delivery request with incomplete address	Error	Error

--THE END--