SOFTWARE ENGINEERING

LAB MANUAL

(R22A0585)

B.TECH - II Year I Sem
(2023-24)

PREPARED BY
K.SWETHA
K.N.KOUSHIL REDDY

DEPARTMENT OF INFORMATION TECHNOLOGY

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)
Recognized under 2(f) and 12 (B) of UGC ACT 1956
(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

DEPARTMENT OF INFORMATION TECHNOLOGY

Vision & Mission

Vision

- To acknowledge quality education and instill high patterns of discipline making the students technologically superior and ethically strong whichinvolves the improvement in the quality of life in human race.

Mission

- To achieve and impart holistic technical education using the best of infrastructure, outstanding technical and teaching expertise to establish the students in to competent and confident engineers.
- Evolving the center of excellence through creative and innovative teaching learning practices for promoting academic achievement to produce internationally accepted competitive and world class professionals.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – ANALYTICAL SKILLS

- To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2 – TECHNICAL SKILLS

- To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and researchbasedon their interest.

PEO3 – SOFT SKILLS

- To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4 – PROFESSIONAL ETHICS

- To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting themselves to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Computer Science and Engineering, the graduates will have the following Program Specific Outcomes:

1. Fundamentals and critical knowledge of the Computer System:- Able to Understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyze the software and hardware aspects of it .

2. The comprehensive and Applicative knowledge of Software Development: Comprehensive skills of Programming Languages, Software process models,methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. Applications of Computing Domain & Research: Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates should possess the following:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design / development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriateconsideration for the public health and safety, and the cultural, societal, and environmentalconsiderations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, andsynthesis of the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilitiesrelevant to the professional engineering practice.

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

Maisammaguda, Dhulapally Post, Via Hakimpet, Secunderabad – 500100

Department of Information Technology

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time),those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with thesynopsis / program / experiment details.
3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm,Procedure, Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) neededin the lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allottedto you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if

anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is keptproperly.

HEAD OF THE DEPARTMENT PRINCIPAL

S.NO	Date	Name of the Activity / Experiment	Grade / Marks	Faculty signature

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

II Year B. TECH IT – I - SEM **L/T/P/C**

-/-/2/1

(R22A0585) SOFTWARE ENGINEERING LAB

Prerequisites

- A course on "Programming for Problem Solving".

Co-requisite

- A Course on "Software Engineering".

Course Objectives:

- Discuss and analyze how to develop software requirements specifications for a given problem.
- To understand Software development as a process.
- To implement various software designs, data flow models and UML diagrams.
- To apply various testing techniques like white box testing & black box testing techniques.
- To have hands on experience in developing a software project by using various software engineering principles and methods.

Course Outcomes:

- Ability to translate end-user requirements into system and software requirements
- Ability to generate a high-level design of the system from the software requirements
- Will have experience and/or awareness of testing problems and will be able to develop a simple testing report.
- Understand and develop various structural and behavioral UML diagrams.
- Explain the knowledge of project management tool

List of Experiments

Do the following exercises for any two projects given in the list of sample projects or any other Projects:

1. Development of Problem Statements.
2. Preparation of Software Requirement Specification Document, Design Documents and Testing Phase related documents.
3. Preparation of Software Configuration Management and Risk Management related documents.
4. Study and usage of any Design phase CASE tool
5. Performing the Design by using any Design phase CASE tools.
6. Develop test cases for unit testing and integration testing
7. Develop test cases for various white box and black box testing techniques.

Sample Projects:

1. Passport automation System
2. Book Bank
3. Online Exam Registration
4. Stock Maintenance System
5. Online course reservation system
6. E-ticketing
7. Software Personnel Management System
8. Credit Card Processing
9. E-book management System.
10. Recruitment system

TEXT BOOKS:

1. Software Engineering, A practitioner's Approach- Roger S. Pressman, 6th edition, McGraw Hill International Edition.
2. Software Engineering-Sommerville, 7thedition, Pearson Education.
3. The unified modeling language user guide Grady Brooch, James Rambaugh, Ivar Jacobson, Pearson Education.

REFERENCE BOOKS:

1. Software Engineering, an Engineering approach- James F. Peters, Witold Pedrycz, John Wiley.
2. Software Engineering principles and practice-Waman SJawadekar

CONTENTS

Task 1: Development of problem statements

***Passport Automation System*:**

Problem Statement: Passport Automation System is used in the effective dispatch of passport to all the eligible applicants after several rounds of processing and verification. This system adopts a comprehensive approach to minimize the manual work and schedule resources, time in a lucid manner.

The core of the system is to get the online registration form (with details such as name, address etc.,) filled by the applicant whose testament is verified for its genuineness by the Passport Automation System with respect to the already existing information in the database. After the first round of verification done by the system, the information is in turn forwarded to the regional administrator's Ministry of External Affairs office. The application is then processed manually based on the report given by the system, and any forfeiting identified can make the applicant liable to penalty as per the law. The system forwards the necessary details to the police for its separate verification whose report is then presented to the administrator. After all the necessary criteria have been met, the original information is added to the database and the passport is sent to the applicant.

Here are a few issues with the manual system that could be addressed through the development of a Passport Automation System:

Manual Processing Inefficiencies:
The current passport issuance process relies heavily on manual data entry, paper-based forms, and lengthy queues at passport offices. This leads to delays, errors, and frustrated applicants.

Data Security and Privacy Concerns:
The manual handling of sensitive personal information during the passport application process raises concerns about data security and privacy breaches. Unauthorized access to personal data is a significant risk.

Inconsistent Application Processing:
Passport offices in different regions or countries may have varying procedures and processing times. This results in inconsistent experiences for applicants and a lack of centralized oversight.

Long Processing Times:
The existing process often requires applicants to wait for extended periods before receiving their passports. This delay can be problematic for individuals who need to travel urgently.

Lack of Application Tracking:
Applicants have limited visibility into the status of their passport applications. This lack of tracking capability makes it challenging for them to plan their travel arrangements.

Manual Verification Processes:
Verification of submitted documents and applicant information is time-consuming due to manual processes. This can lead to errors in verifying authenticity and identity.

Limited Accessibility:
Not all applicants can easily access physical passport offices due to geographical constraints. This limits access to passport services for remote or rural populations.

Paper-Based Documentation:
The reliance on physical documents creates challenges in terms of storage, retrieval, and sharing of information. There's a need for a more efficient digital document management system.

Lack of Integration with other Systems:
Existing passport systems might not be integrated with other government or immigration systems, leading to redundant data entry and inefficiencies.

Complex Documentation Requirements:
Applicants often struggle to understand and full fill the documentation requirements for passport applications, leading to rejections and repeated visits.

Inefficient Appointment Scheduling:
The current system might not offer efficient appointment scheduling mechanisms, causing long waiting times and dissatisfaction among applicants.

Manual Payment Processing:
Payment processes for passport fees might involve manual cash handling, leading to accounting discrepancies and potential corruption.

These problems/issues highlight the key challenges and inefficiencies that a Passport Automation System could address. The system aims to streamline processes, enhance security, improve applicant experiences, and provide efficient tracking and management of passport applications.

RECORD NOTES:

Task 2: Preparation of Software Requirement Specification Document, Design Documents and Testing Phase related documents

1. Software Requirement Specification (SRS) Document:

The SRS document outlines the detailed requirements for the Passport Automation System. It defines the system's functionalities, constraints, and user expectations. The SRS document typically includes the following sections:

Introduction: Provides an overview of the document, its purpose, and the scope of the Passport Automation System.
System Overview: Describes the system's high-level architecture, components, and interactions with external systems.

Functional Requirements: Specifies detailed functionalities of the system. Each requirement should be clear, concise, and unambiguous. Use cases and user stories can be helpful here.

Non-Functional Requirements: Addresses system qualities like performance, scalability, security, usability, and reliability.

User Interfaces: Describes the user interfaces, including mockups or wireframes if available.

Data Requirements: Defines the data structures, databases, and data flow within the system.

System Architecture: Provides an overview of the system's technical architecture, including hardware and software components.

Dependencies: Lists external systems, libraries, or APIs that the system will depend on.

Constraints: Specifies any limitations or restrictions that the system must adhere to.

Assumptions: Documents any assumptions made during requirement gathering.

2. Design Documents:

Design documents elaborate on the technical architecture, system components, and implementation details of the Passport Automation System. These documents aid the development process. Common design documents include:

High-Level Design (HLD): Outlines the overall system architecture, component interactions, and major modules.

Low-Level Design (LLD): Provides in-depth details about each module, including data structures, algorithms, and interfaces.

Database Design: Describes the database schema, relationships, and data manipulation methods.

User Interface (UI) Design: Provides detailed information about the user interfaces, including layouts, styles, and user interactions.

API Design: Documents the APIs, endpoints, request-response formats, and authentication mechanisms.

3. <u>Testing Phase Related Documents</u>:

The testing phase requires a set of documents to ensure comprehensive testing and validation of the system:

Test Plan: Outlines the testing approach, test objectives, scope, testing strategies, and resources required for testing.

Test Cases: Provides detailed scenarios, inputs, expected outcomes, and procedures for testing various system functionalities.

Test Scripts/Automation: If applicable, provides automated scripts for executing test cases.

Test Data: Specifies the data needed to perform tests effectively.

Defect Reports: Documents any defects found during testing, including steps to reproduce and severity levels.

Traceability Matrix: Links test cases back to the specific requirements, ensuring all requirements are tested.

Test Summary/Report: Summarizes the testing process, results, and any open issues.

Remember that these documents should be continuously updated throughout the development process to reflect any changes or additions. The documents will serve as valuable references for developers, testers, and stakeholders, ensuring a well-organized and successful development and testing process for the Passport Automation System.

SOFTWARE REQUIREMENT SPECIFICATION (SRS)
FOR
PASSPORT AUTOMATION SYSTEM

PURPOSE

If the entire process of 'Issue of Passport' is done in a manual manner then it would takes several months for the passport to reach the applicant. Considering the fact that the number of applicants for passport 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. As this is a matter of National Security, the system has been carefully verified and validated in order to satisfy it.

SCOPE

- The System provides an online interface to the user where they can fill in their personal details and submit the necessary documents (maybe by scanning).
- The authority concerned with the issue of passport can use thissystem to reduce his workload and process the application in aspeedy manner.
- Provide a communication platform between the applicant and theadministrator.
- Transfer of data between the Passport Issuing Authority and the Local Police for verification of applicant's information.

Users/Applicants will come to know their status of application and the date in which they must subject themselves for manual documentverification.

DEFINITIONS, ACRONYMS AND ABBREVIATIONS

- **Administrator**

  Refers to the super user who is the Central Authority with the privilege to manage the entire system. It can be any higher official in the Regional Passport Office of Ministry of External Affairs.
- **Applicant**

  One who wishes to obtain the Passport?
- **PAS**

  Refers to this Passport Automation System.
- **HTML**

  Markup Language used for creating web pages.

- **J2EE**

  Java 2 Enterprise Edition is a programming platform java platform for developing and running distributed java applications.

- **HTTP**

  Hyper Text Transfer Protocol.

- **TCP/IP**

  Transmission Control Protocol/Internet Protocol is the communication protocol used to connect hosts on the Internet.

TECHNOLOGIES TO BE USED

- HTML
- JSP
- JavaScript
- Java

TOOLS TO BE USED

- Eclipse IDE (Integrated Development Environment)
- Rational Rose tool (for developing UML Patterns)

OVERVIEW

SRS includes two sections overall description and specific requirements **Overall Description** will describe major role of the system components andinter- connections.

Specific Requirements will describe roles & functions of the actors.

OVERALL DESCRIPTION

PRODUCT PERSPECTIVE

The PAS acts as an interface between the 'applicant' and the 'administrator'. This system tries to make the interface as simple as possible and at the same time not risking the security of data stored in. This minimizes the time duration in which the user receives the passport.

SOFTWARE INTERFACE

- **Front End Client** - The applicant and Administrator online interface is built using JSP and HTML. The Administrator's localinterface is built using Java.
- **Web Server** – Apache Tomcat application server (OracleCorporation).
- **Back End** – Oracle11g database.

HARDWARE INTERFACE

The server is directly connected to the client systems. The client systems have access to the database in the server.

SYSTEM FUNCTIONS

- Secure Registration of information by the Applicants.
- Schedule the applicants an appointment for manual verification oforiginal documents.
- Panel for Passport Application Status Display by the Administrator.
- SMS and Mail updates to the applicants by the administrator.
  Administrator can generate reports from the information and is the only authorized personnel to add the eligible application informationto the database.

USER CHARACTERISTICS

- **Applicant**

  These are the person who desires to obtain the passport and submit the information to the database.

- **Administrator**

  He has the certain privileges to add the passport status and to approve the issue of passport. He may contain a group of persons under him toverify the documents and give suggestion whether or not to approve the dispatch of passport.

- **Police**

  He is the person who upon receiving intimation from the PAS, perform a personal verification of the applicant and see if he has any criminal case against him before or at present. He has been vetoed with the power to decline an application by suggesting it to the Administrator if he finds any discrepancy with the applicant. He communicates via this PAS.

CONSTRAINTS

- The applicants require a computer to submit their information.
- Although the security is given high importance, there is always a chance of intrusion in the web world which requires constant monitoring.

  The user has to be careful while submitting the information. Much careis required.

ASSUMPTIONS AND DEPENDENCIES

- The Applicants and Administrator must have basic knowledge of computers and English Language.
- The applicants may be required to scan the documents and send.

RECORD NOTES

Task 3: Preparation of Software Configuration Management and Risk Management related documents.

Software Configuration Management (SCM) Document:

Introduction:
Brief overview of the document's purpose and scope.

Configuration Management Plan:
Outline the overall approach to configuration management.
Define roles and responsibilities of team members involved in SCM.
Specify the tools and resources to be used for configuration management.

Version Control:
Specify version control practices for source code, documents, and other artifacts.
Describe how branching, merging, and release management will be handled.

Configuration Identification:
Define naming conventions for components, files, and versions.
Explain how to uniquely identify and label each configuration item.

Configuration Control:
Detail the change management process, including how changes are requested, reviewed, and approved.
Describe the change tracking mechanism and how it's integrated with development activities.

Configuration Status Accounting:
Explain how configuration status will be tracked and reported.
Define the format and frequency of status reports.

Configuration Auditing:
Describe the process for conducting configuration audits to ensure compliance with standards and requirements.

Baseline Management:
Define how and when baselines will be established for different project phases.
Explain the process for managing changes after baselines are established.

Backup and Recovery:
Detail the backup and recovery strategy for configuration items and project data.

Release Management:

Describe the process for packaging, distributing, and deploying releases to different environments.

Risk Management Document:

Introduction:
Briefly explain the purpose and importance of risk management in the project.

Risk Management Plan:
Outline the overall approach to risk management throughout the project lifecycle.
Define roles and responsibilities of team members involved in risk management.

Risk Identification:
List and categorize potential risks related to the Passport Automation System project.
Include risks related to technical, organizational, and external factors.

Risk Assessment:
Evaluate and prioritize identified risks based on their impact and likelihood.
Create a risk matrix to visualize the severity of each risk.

Risk Mitigation Strategies:
Define strategies to mitigate high-priority risks.
Specify preventive and corrective actions for each risk.

Risk Monitoring and Control:
Explain how risks will be monitored throughout the project.
Describe the process for tracking risk triggers and implementing mitigation plans.

Risk Communication:
Define how risks and mitigation plans will be communicated to stakeholders.
Include a communication plan for addressing risks in a transparent manner.

Contingency Planning:
Detail contingency plans for handling risks that cannot be fully mitigated.

Lessons Learned:
Discuss how lessons learned from risk events will be documented and shared for future projects.
Remember, both SCM and Risk Management documents should be tailored to your project's specific needs, and they should be reviewed and updated as the project progresses to ensure their continued relevance and effectiveness.

RECORD NOTES:

Task 4: Study and usage of any Design phase CASE tool

A Computer-Aided Software Engineering (CASE) tool can significantly aid the design phase of a project like a Passport Automation System. One popular type of CASE tool is a Unified Modeling Language (UML) tool, which helps in creating visual representations of the system's architecture, components, interactions, and more. Let's consider an example of using a UML tool for the design phase of a Passport Automation System:

The UML is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML gives you a standard way to write a system's blueprints, covering conceptual things, such as business processes and system functions, as well as concrete things, such as classes written in a specific programming language, database schemas, andreusable software components.

Model

A model is a simplification of reality. A model provides the blueprints of a system. A model may be structural, emphasizing the organization of the system, or it may be behavioral, emphasizing the dynamics of the system.

Why do we model

We build models so that we can better understand the system we are developing.

Through modeling, we achieve four aims.

1. Models help us to visualize a system as it is or as we want it to be.
2. Models permit us to specify the structure or behavior of a system.
3. Models give us a template that guides us in constructing a system.
4. Models document the decisions we have made.

We build models of complex systems because we cannot comprehend such a system in its entirety.

Principles of Modeling

There are four basic principles of model

1. The choice of what models to create has a profound influence on how a problem is attackedand how a solution is shaped.
2. Every model may be expressed at different levels of precision.
3. The best models are connected to reality.
4. No single model is sufficient. Every nontrivial system is best approached through a small set ofnearly independent models.

An Overview of UML

- The Unified Modeling Language is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system.
- The UML is appropriate for modeling systems ranging from enterprise information systems to distributed Web-based applications and even to hard real time embedded systems. It is a very expressive language, addressing all the views needed to develop and then deploy such systems.

The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting

- **Visualizing** The UML is more than just a bunch of graphical symbols. Rather, behind each symbol in the UML notation is a well-defined semantics. In this manner, one developer can write a model in the UML, and another developer, or even another tool, can interpret that model unambiguously

- **Specifying** means building models that are precise, unambiguous, and complete.

- **Constructing** the UML is not a visual programming language, but its models can be directly connected to a variety of programming languages
- **Documenting** a healthy software organization produces all sorts of artifacts in addition to raw executable code.

To understand the UML, you need to form a **conceptual model of the language**, and this requires learning three major elements:

1. Things
2. Relationships
3. Diagrams

Things in the UML

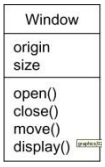
There are four kinds of things in the UML:

Structural things
Behavioral things
Grouping things
Annotational things

Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. In all, there are seven kinds of structural things.

1. Classes
2. Interfaces
3. Collaborations
4. Use cases
5. Active classes
6. Components
7. Nodes

Class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces. Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations.



Interface

Interface is a collection of operations that specify a service of a class or component.
An interface therefore describes the externally visible behavior of that element.
An interface might represent the complete behavior of a class or component or only a part of that behavior.

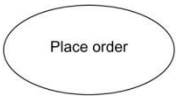
An interface is rendered as a circle together with its name. An interface rarely stands alone. Rather, it is typically attached to the class or component that realizes the interface

Collaboration defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior that's bigger than the sum of all the elements. Therefore, collaborations have structural, as well as behavioral, dimensions. A given class might participate in several collaborations.

Graphically, a collaboration is rendered as an ellipse with dashed lines, usually including only its name

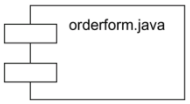
Use case

- Use case is a description of set of sequence of actions that a system performs that yields an observable result of value to a particular actor
- Use case is used to structure the behavioral things in a model.
- A use case is realized by a collaboration. Graphically, a use case is rendered as an ellipse with solid lines, usually including only its name

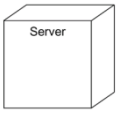


Active class is just like a class except that its objects represent elements whose behavior is concurrent with other elements. Graphically, an active class is rendered just like a class, but with heavy lines, usually including its name, attributes, and operations

Component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces. Graphically, a component is rendered as a rectangle with tabs



Node is a physical element that exists at run time and represents a computational resource, generally having at least some memory and, often, processing capability. Graphically, a node is rendered as a cube, usually including only its na



Behavioral Things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space. In all, there are two primary kinds of behavioral things

Interaction
state machine

Interaction

Interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose
An interaction involves a number of other elements, including messages, action sequences and links
Graphically a message is rendered as a directed line, almost always including the name of its operation

State Machine

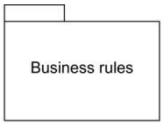
State machine is a behavior that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its responses to those events
State machine involves a number of other elements, including states, transitions, events and activities
Graphically, a state is rendered as a rounded rectangle, usually including its name and its substates

Grouping Things:-

1. are the organizational parts of UML models. These are the boxes into which a model can bedecomposed
2. There is one primary kind of grouping thing, namely, packages.

Package:-

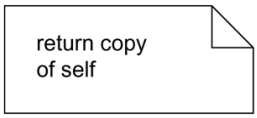
- A package is a general-purpose mechanism for organizing elements into groups. Structural things,behavioral things, and even other grouping things may be placed in a package
- Graphically, a package is rendered as a tabbed folder, usually including only its name and, sometimes, its contents



Annotational things are the explanatory parts of UML models. These are the comments you may apply to describe about any element in a model.

A note is simply a symbol for rendering constraints and comments attached to an element or a collection of elements.

Graphically, a note is rendered as a rectangle with a dog-eared corner, together with a textual orgraphical comment

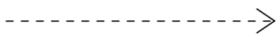


Relationships in the UML: There are four kinds of relationships in the UML:

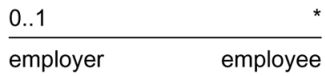
1. Dependency
2. Association
3. Generalization
4. Realization

Dependency:-

Dependency is a semantic relationship between two things in which a change to one thing mayaffect the semantics of the other thing
Graphically a dependency is rendered as a dashed line, possibly directed, and occasionallyincluding a label



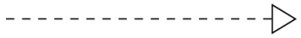
Association is a structural relationship that describes a set of links, a link being a connection amongobjects.
Graphically an association is rendered as a solid line, possibly directed, occasionally including a label,and often containing other adornments, such as



multiplicity and role names

Aggregation is a special kind of association, representing a structural relationship between a whole and its parts. Graphically, a generalization relationship is rendered as a solid line with a hollow arrowhead pointing to the parent

Realization is a semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carry out. Graphically a realization relationship is rendered as a cross between a generalization and a



dependency relationship

Diagrams in the UML

- **Diagram** is the graphical presentation of a set of elements, most often rendered as a connectedgraph of vertices (things) and arcs (relationships).
- In theory, a diagram may contain any combination of things and relationships.

- For this reason, the UML includes nine such diagrams:
    - Class diagram
    - Object diagram
    - Use case diagram
    - Sequence diagram
    - Collaboration diagram
    - State chart diagram
    - Activity diagram
    - Component diagram
    - Deployment diagram

RECORD NOTES

Task 5: Performing the Design by using any Design phase CASE tools

Designing a Use Case Diagram for Passport Automation System:
Step 1: Create a New Diagram:
Using preferred UML CASE tool.
Create a new diagram and select "Use Case Diagram" from the UML diagram category.
Step 2: Identify Actors:
Add actors to the diagram. These are external entities that interact with the system. For the Passport Automation System, typical actors could be "Applicant" and "Passport Officer."
Step 3: Add Use Cases:
Add use cases to the diagram. These represent specific actions or functionalities that the system provides.
Connect the use cases to the actors using lines to show interactions. For instance, connect the "Apply for Passport" use case to the "Applicant" actor.
Step 4: Define Relationships:
Add associations between use cases to show dependencies or interactions between them. For instance, connect "Verify Documents" and "Issue Passport" use cases to "Apply for Passport" to indicate that these use cases are related.
Step 5: Add System Boundary:
Draw a system boundary around the actors and use cases to visually represent the scope of the system.
Step 6: Add Labels and Annotations:
Label the actors and use cases with appropriate names to clearly indicate their roles and functionalities.
Add notes or annotations to provide additional context or explanations where necessary.
Step 7: Save and Share:
Save your diagram and share it with team members or stakeholders for review and feedback.

Important Considerations:

Accurate Representation: Ensure that your use case diagram accurately represents the interactions and functionalities of the Passport Automation System. Use descriptive names for actors and use cases.

Clarity: Keep the diagram clear and uncluttered. Avoid excessive details that might hinder understanding.

Consistency: Follow UML conventions and guidelines for symbols, notations, and relationships.

Collaboration: CASE tools often support collaboration features, allowing multiple team members to work on the same diagram simultaneously.

Remember, this is just one aspect of the design phase. Depending on the complexity of the system, you would create various other diagrams like class diagrams, sequence diagrams, and more, to comprehensively capture the design details of the Passport Automation System. The goal is to create a clear and detailed representation of the system's architecture, components, and interactions.

USECASE DIAGRAM:

The Passport Automation system use cases are:

1. Login
2. Registration
3. Verification
4. Check status
5. Enquiry
6. Dispatch Passport

ACTORS INVOLVED:

1. Applicant
2. Passport Officer
3. Police

USE-CASE NAME: LOGIN

The applicant login to the system to obtain a passport

USE-CASE NAME: REGISTRATION

The Applicant enters his name and details for applying a Passport. The applicant initially give his/ her details for registration.

USE-CASE NAME: VERIFICATION

The system verifies the applicant mandatory information given by him/her.

USE-CASE NAME: CHECK STATUS

The Applicant tries to check the status in which category applied. The system displays the message to the applicant.

USE-CASE NAME: ENQUIRY

The police receive intimation from the PAS, perform a personal verification of the applicant and see if he has any criminal case against him before or at present. He has been vetoed with the power to decline an application by suggesting it to the Administrator if he finds any discrepancy with the applicant. He communicates via this PAS.

USE-CASE NAME: DISPATCH PASSPORT

The administrator check or process the application which are submitted by applicant .Process the application means the data which are given by the applicant is processed to create a passport for the applicant and finallydispatches the passport to the applicant.

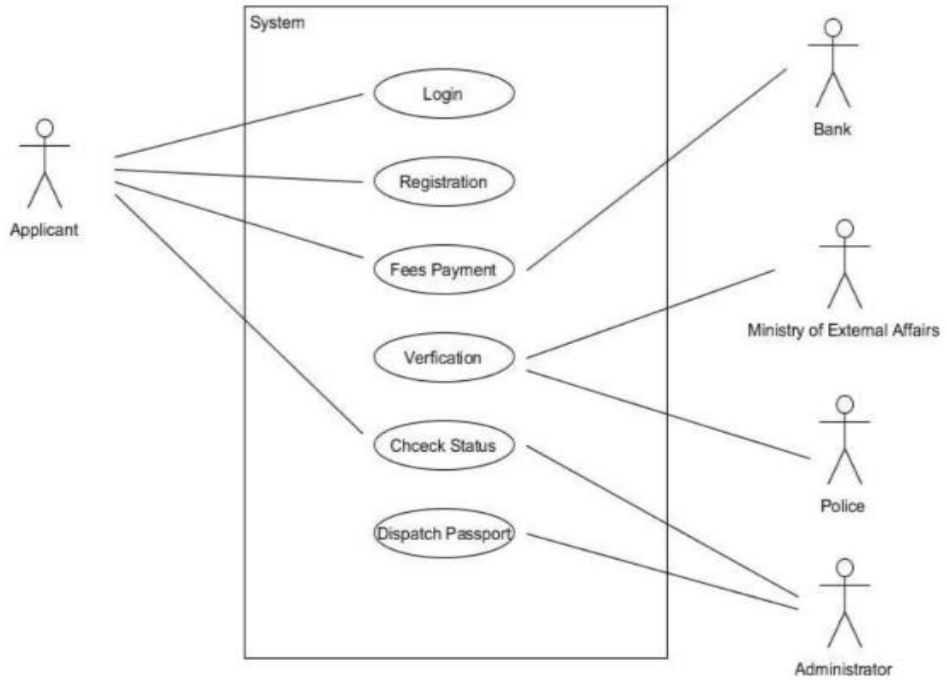


Fig.1. USECASE DIAGRAM FOR PASSPORT AUTOMATION SYSTEM

ACTIVITY DIAGRAM:

The activity diagram represents the series of activities that are occurring between the objects. Following is activity diagram which represents the Software personnel management system process.

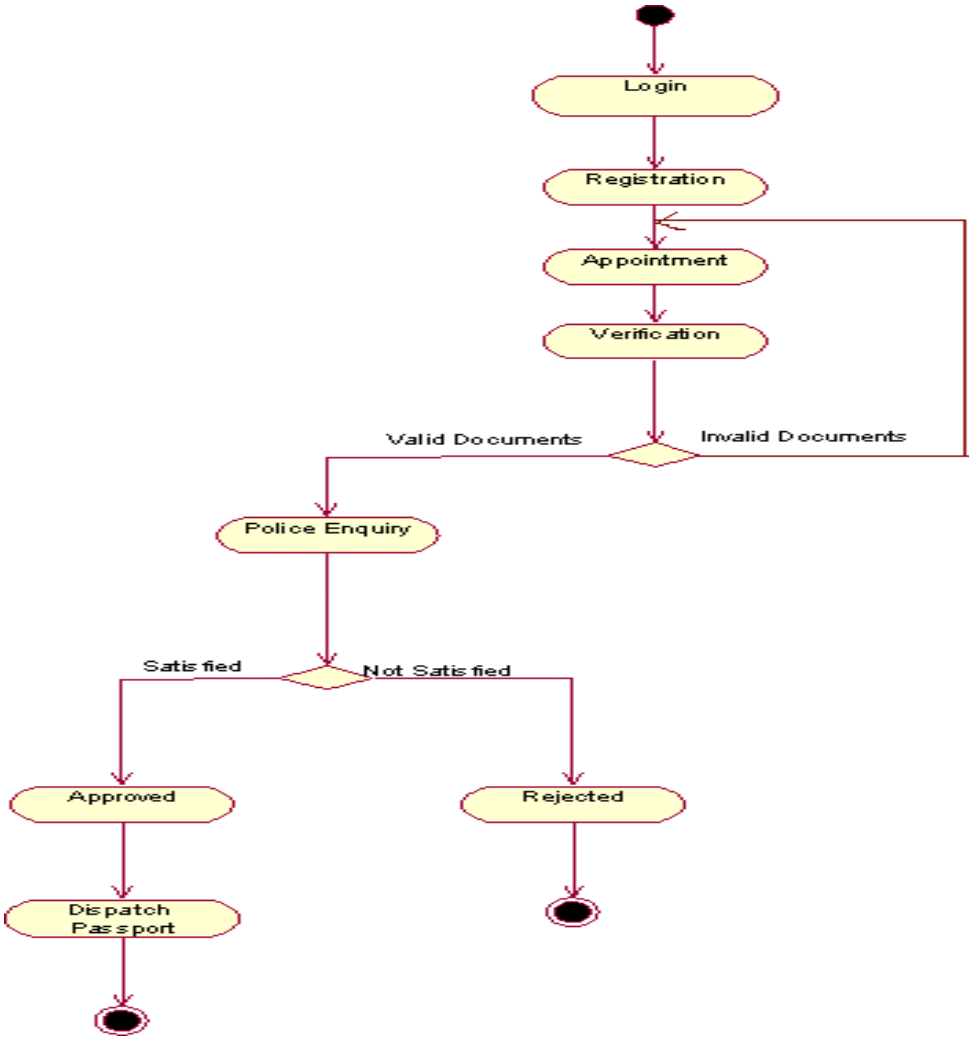


Fig.2. ACTIVITY DIAGRAM FOR PASSPORT AUTOMATION SYSTEM

CLASS DIAGRAM:

The class diagram is referred as object modeling in the static analysis diagram. The main task of object modeling is to graphically show what each object will do in the problem domain. The problem domain describes the structure and the relationships among objects.

The Passport Automation system class diagram consists of five classes

1. Login class
2. Appointment class
3. Registration class
4. Authority class
5. Verification class

1) LOGIN CLASS:

It consists of two attributes and two operations. The attributes are user name, and password. The operations of this class are creating login ( ), sign in ( ).

2) APPOINTMENT CLASS:

The attributes of this class are appointment id, applicant id, date, time, and description. The operation of this class are get appointment ( ), get appointment status ( ), Modify ( ), cancel ( ).

3) REGISTRATION CLASS:

The attributes are applicant id, name, dob, gender, birthplace, father name, addr1, addr2, district, state, country, pin code, mobile, email id, qualification. The operation are add ( ), modify( ), view( ).

4) AUTHORITY CLASS:

The attributes of this class are officered, name, designation, andpassword. The operations are search( ).

5) VERIFICATION CLASS:

The attributes of this class are verification id, appointment id, applicant id, officer id, status id, description. The operation are verify( ).

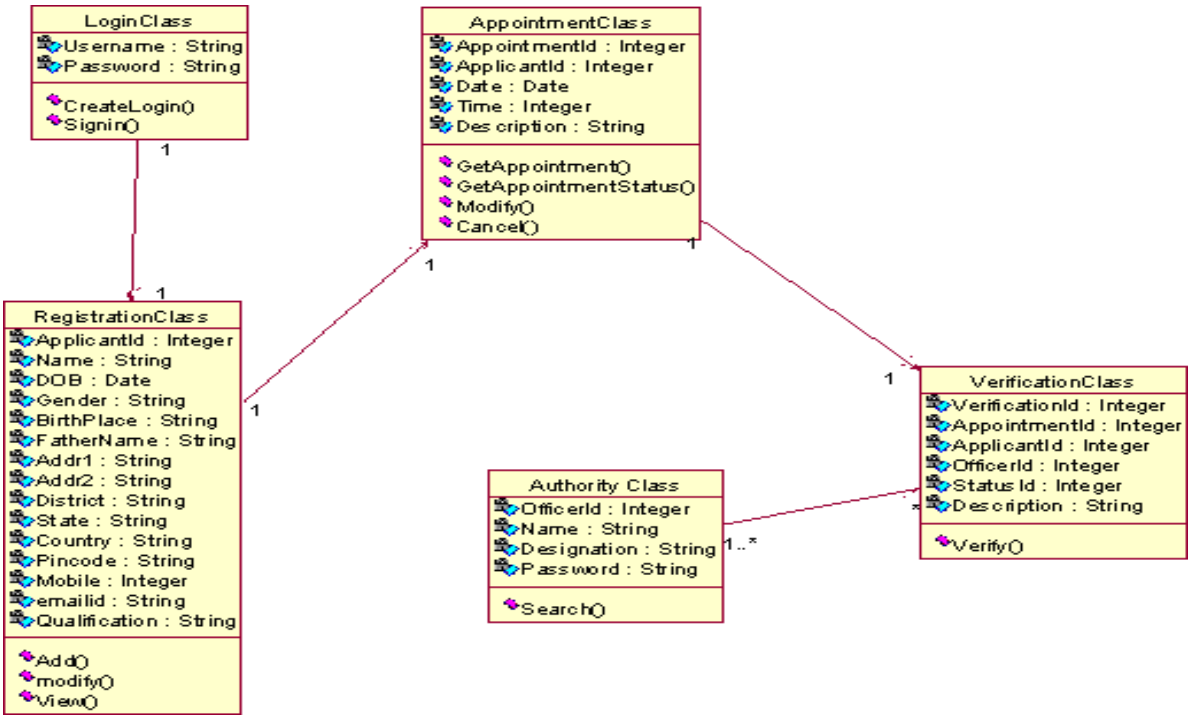


Fig.3.CLASS DIAGRAM FOR PASSPORT AUTOMATION SYSTEM

INTERACTION DIAGRAM:

1. A sequence diagram represents the sequence and interactions of a given USE-CASE or scenario. Sequence diagrams can capture most of the information about the system.
2. Most object to object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.
3. An event also is considered to be any action by an object that sends information.
4. The event line represents a message sent from one object to another, in which the "form" object is requesting an operation be performed by the "to" object. The "to" object performs the operation using a method that the class contains.
5. It is also represented by the order in which things occur and how the objects in the system send message to one another.
6. The sequence diagram for each USE-CASE that exists when a user administrator, check status and new registration about passport automation system are given

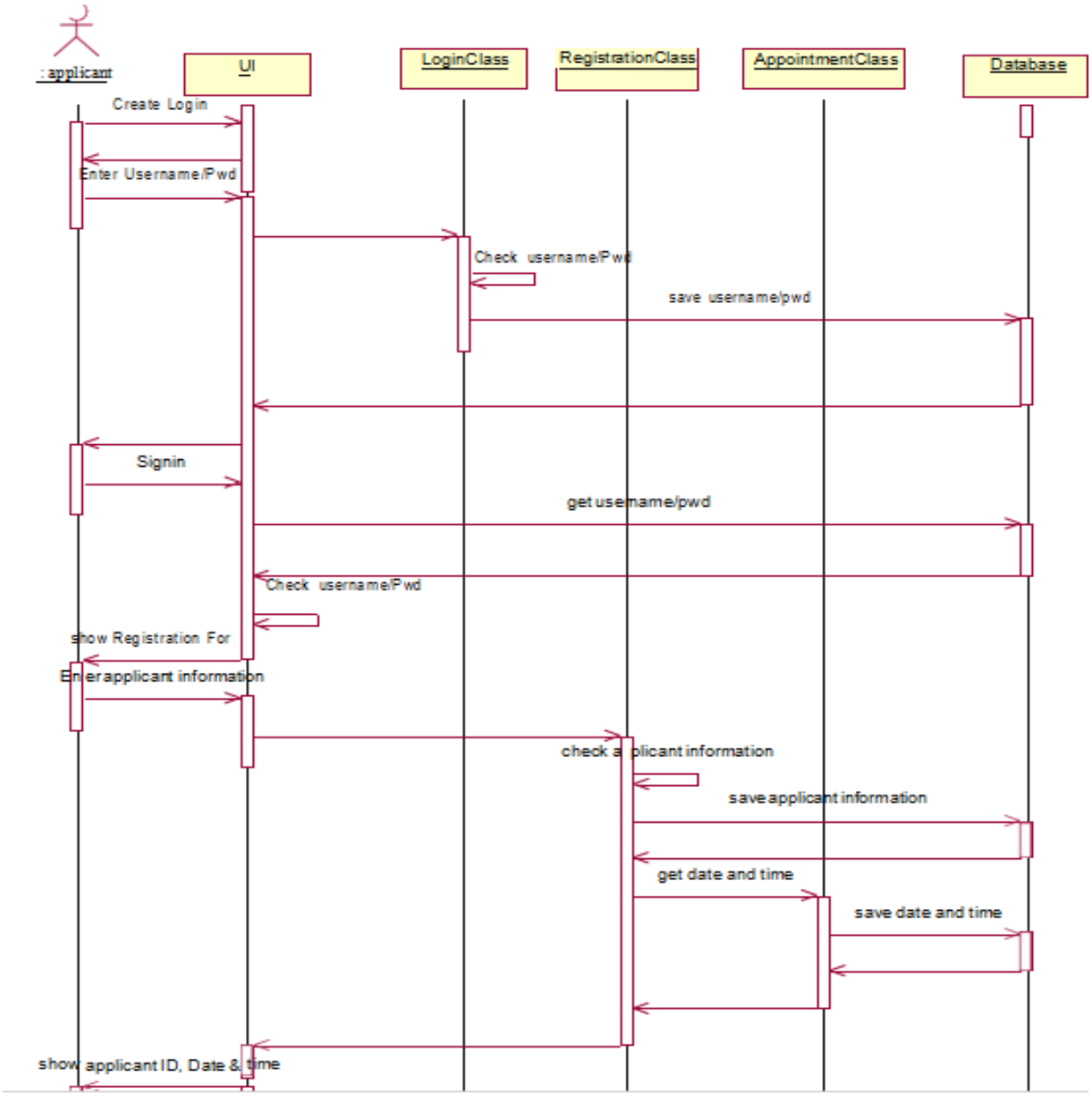


Fig.4.SEQUENCE DIAGRAM FOR LOGIN AND VERIFICATION

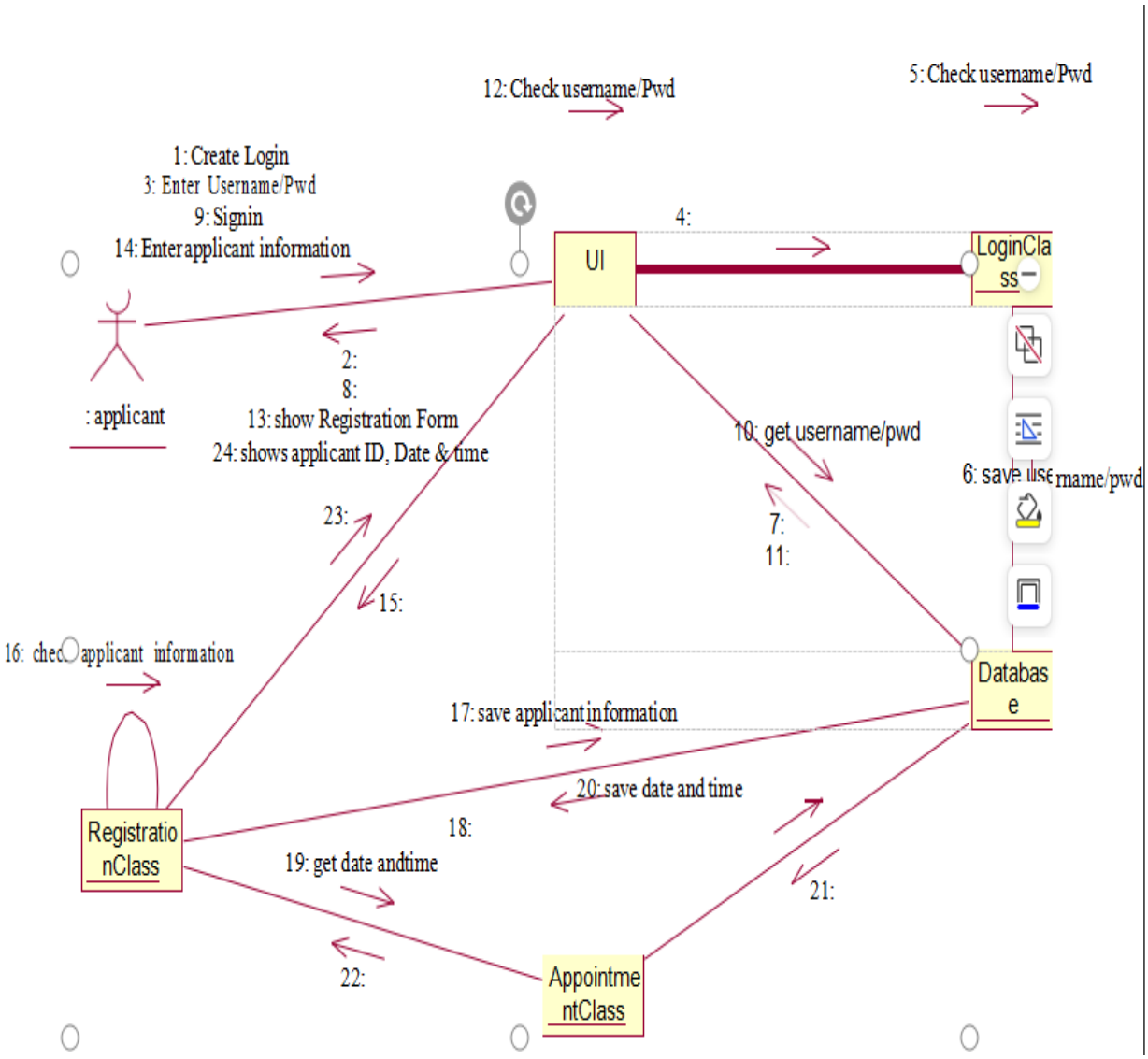


Fig.5.COLLABORATION DIAGRAM FOR LOGIN AND VERIFICATION

- The diagrams show the process done by the Passport Authority to the Passport Automation system. The applicant has to enter his details.
- The details entered are verified by the Passport Authority and the applicant is approved if the details match then the passport is dispatch, otherwise an appropriate error message is displayed

STATE CHART DIAGRAM:

- Every object undergoes through some state and on receiving some event the state gets changed. This transition of the state can be represented by the state transition diagram.

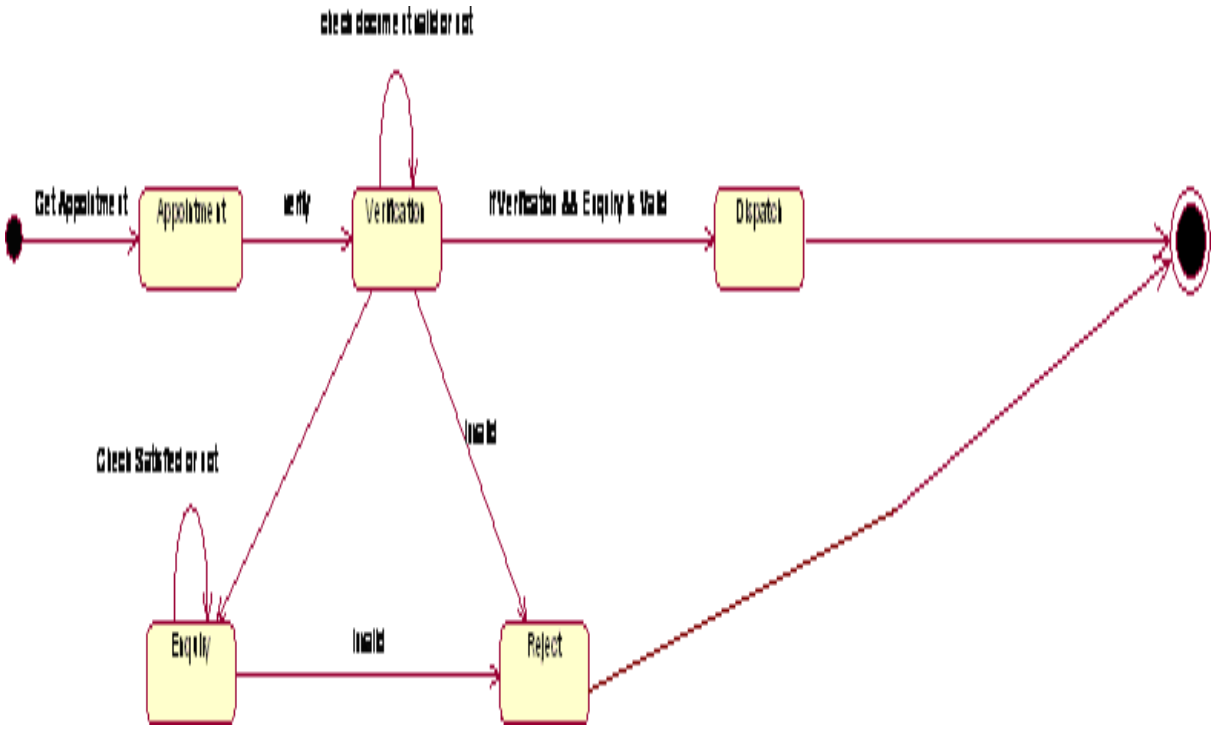


Fig.6.STATE CHART DIAGRAM FOR PASSPORT AUTOMATION SYSTEM

DEPLOYMENT DIAGRAM AND COMPONENT DIAGRAM

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

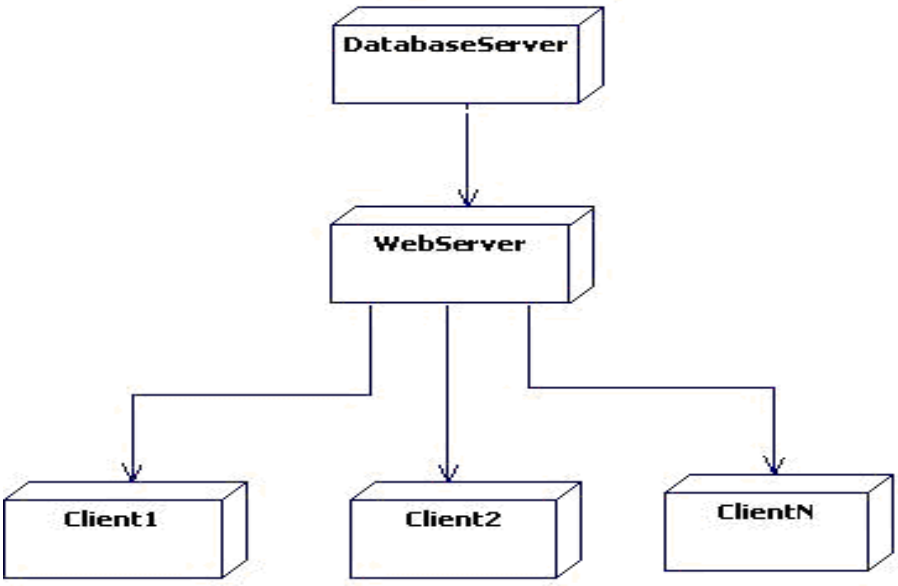


Fig.7.DEPLOYMENT DIAGRAM FOR PASSPORT AUTOMATION SYSTEM

COMPONENT DIAGRAM

Component diagrams are used to visualize the organization and relationshipsamong components in a system.

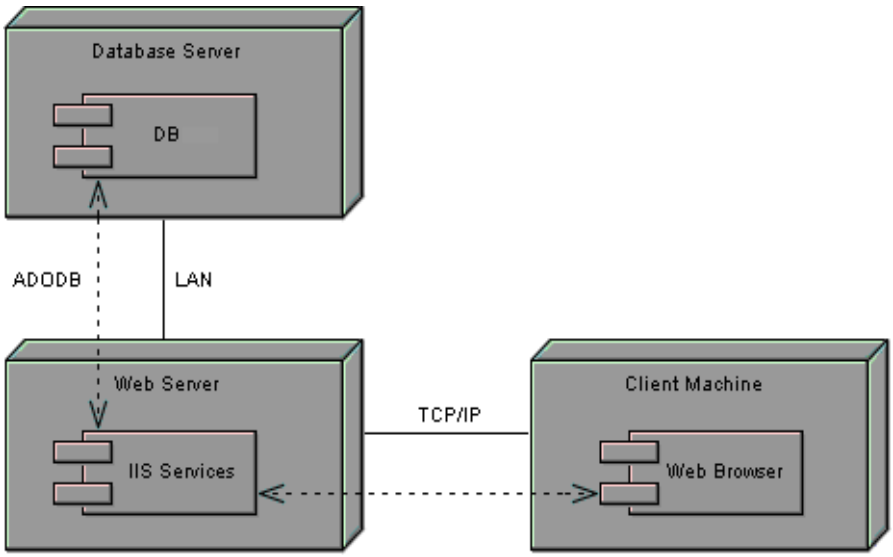
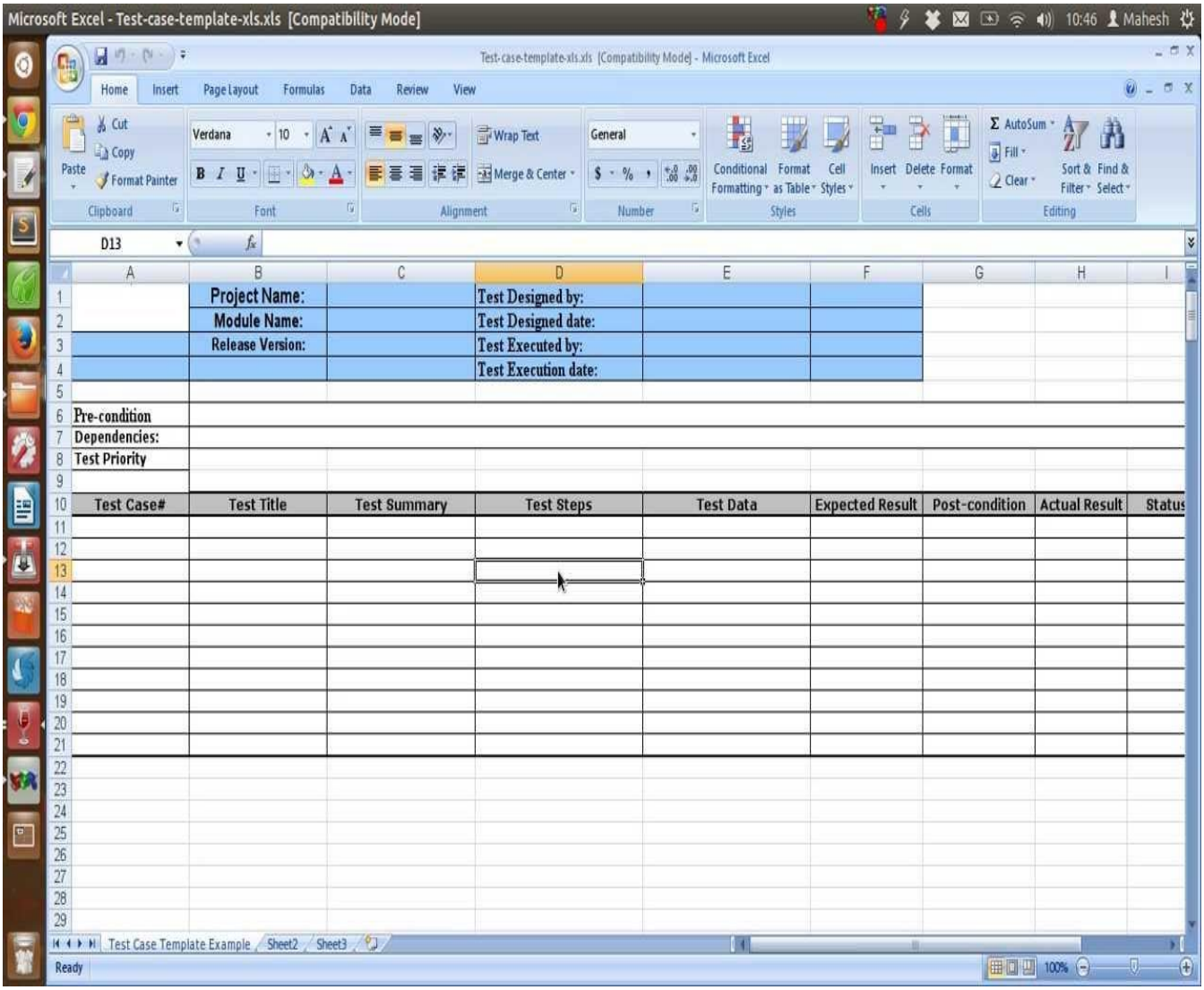


Fig.8.COMPONENT DIAGRAM FOR PASSPORT AUTOMATION SYSTEM

RECORD NOTES

Task 6: Develop test cases for unit testing and integration testing

A **Test Case** is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.



Unit Testing Test Cases:

Test Case: Verify Applicant Information Validation

Description: Ensure that the system properly validates and sanitizes applicant information.
Input: Applicant's name, address, date of birth, etc.
Expected Outcome: Valid inputs are accepted, and invalid inputs are rejected with appropriate error messages.

Test Case: Test Document Verification Logic

Description: Test the logic that verifies submitted documents against predefined criteria.
Input: Sample set of valid and invalid documents.
Expected Outcome: Valid documents pass verification, while invalid ones fail with corresponding error messages.

Test Case: Test Passport Issuance Process

Description: Verify the process of issuing a passport once all requirements are met.
Input: Approved applicant details and verified documents.
Expected Outcome: A new passport is generated and associated with the applicant**.**

Integration Testing Test Cases:

Test Case: Verify Applicant Application Flow
Description: Test the integration of various application components for a seamless user experience.
Steps:
Applicant submits an application.
Documents are verified.
Passport is issued upon successful verification.
Expected Outcome: Each step integrates smoothly, ensuring a complete application process.

Test Case: Test Data Exchange Between Modules

Description: Verify that data is correctly passed between different system modules.
Steps:
Applicant details are shared from the application module to the verification module.
Verification results are passed back to the application module.
Expected Outcome: Data is accurately exchanged between modules without loss or corruption.

Test Case: Test System Performance Under Load

Description: Evaluate the system's performance when handling multiple concurrent applications.
Input: Simulated concurrent application submissions.
Expected Outcome: The system should handle the load without significant degradation in response time or errors.

Test Case: Test Integration with External Systems

Description: Validate interactions with external systems, such as government databases for background checks.
Input: Simulated requests and responses from external systems.
Expected Outcome: The system should correctly send requests and process responses from external systems.

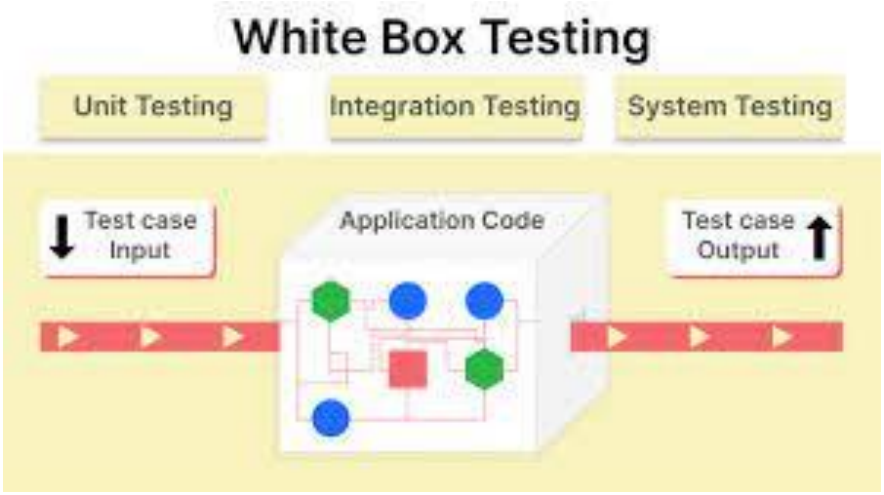
Test Case: Test Error Handling and Logging

Description: Verify how the system handles unexpected errors and logs relevant information.
Steps:
Trigger a scenario that causes an error (e.g., database connection failure).
Check if the system logs the error and provides meaningful error messages.
Expected Outcome: Errors are appropriately logged, and users are presented with informative error messages.

RECORD NOTES

Task 7: Develop test cases for various white box and black box testing techniques

White box testing is a software testing technique that tests a system's internal design, source code structure, data structures used, and working details. Its primary objective is to improve the software's design, input-output flow, usability, and security. It is also called transparent testing, structural testing, and glass box testing.



White Box Testing Techniques:

1. Statement Coverage:

Test Case: Ensure that each line of code is executed at least once during the application process.

Steps:

Submit a passport application with valid details.

Verify the documents.

Check the status of the application.

Expected Outcome: All code lines related to application submission, document verification, and status tracking are executed.

2. Branch Coverage:

Test Case: Verify that all possible branches within decision points are taken during testing.

Steps:

Submit an application with valid details and valid documents.

Submit an application with valid details but invalid documents.

Submit an application with missing details.

Expected Outcome: All possible execution paths through decision points (like document validation) are tested.

3. Path Coverage:

Test Case: Ensure that all possible paths through the system are tested.

Steps:

Submit an application with valid details and documents.

Verify documents successfully.

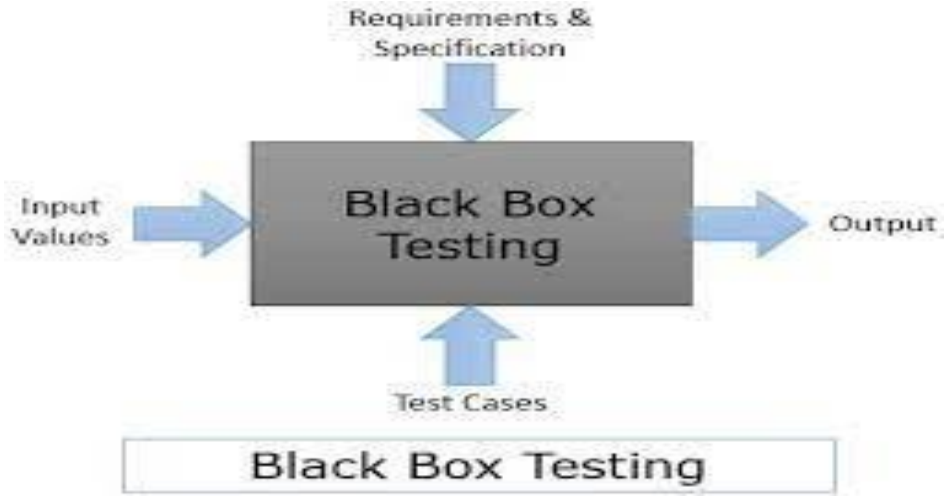
Issue a passport.

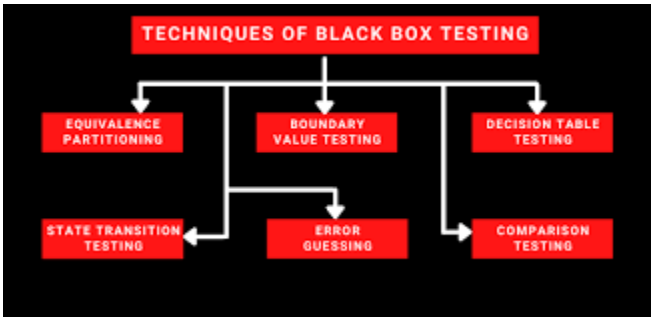
Expected Outcome: The entire path from application submission to passport issuance is tested.

Black Box Testing Techniques:

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed





1. Equivalence Partitioning:

Test Case: Validate input validation by partitioning input domains.

Steps:

Submit an application with valid details.

Submit an application with invalid characters in the name field.

Submit an application with an age above the valid range.

Expected Outcome: Input partitions are tested, including valid inputs and common invalid inputs.

2. Boundary Value Analysis:

Test Case: Test values at the boundaries of valid and invalid ranges.

Steps:

Submit an application with the minimum required age.

Submit an application with an age one unit below the minimum.

Submit an application with the maximum allowed age.

Expected Outcome: System handles boundary values correctly and doesn't break at extreme inputs.

3. Decision Table Testing:

Test Case: Test combinations of conditions and actions in decision tables.

Steps:

Submit an application with valid details and valid documents.

Submit an application with valid details but invalid documents.

Submit an application with missing details.

Expected Outcome: Different combinations of conditions and actions are tested.

4. Scenario-based Testing:

Test Case: Test end-to-end scenarios based on user stories.

Steps:

Submit a passport application, verify documents, and issue a passport.

Submit an application with incomplete details and verify the response.

Expected Outcome: User stories are tested to ensure the system meets user requirements.

RECORD NOTES