YEAR: III B. TECH

SEMESTER: I

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

OBJECT ORIENTED SOFTWARE ENGINEERING LAB

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

OBJECT ORIENTED SOFTWARE ENGINEERING LAB

Course Code: GR18A3052 **L/T/P/C: 0/0/3/1.5**

Course objectives

- To impart state-of-the-art knowledge on Object oriented Software Engineering and UML.
- Practice object oriented software engineering principles for real time problems.
- Design various types of diagrams for real time problems.
- Learn test case generation.
- Demonstrate object oriented software engineering methodologies for various real time problems.

Course outcomes

- Analyze and identify requirements for real time problems.
- Design and implement various software design models.
- Usage of modern engineering tools for specification, design and implementation.
- Provide appropriate solutions for the real time problems using object oriented software engineering methodology.
- Design test cases for various real time problems.

Software's Used: StarUML /Umbrello & JUNIT

Develop the following applications using object oriented software engineering methodologies.

1. Unified Library System 2. Online Railway Reservation System 3. Data Warehouse Management System.

Task1. Prepare the problem statement for above applications.

Task2. Develop Software Requirement Specification (SRS) for above applications.

Task3. Design the class and object diagrams for above applications.

Task 4: Design the Use-case diagrams for the above applications.

Task 5: Design the interaction diagrams for the above applications.

Task 6: Design the activity diagrams for the above applications.

Task 7: Design the State-chart diagrams for the above applications.

Task 8: Perform forward and Reverse Engineering, and generate documentation of the project.

Task 9: Design the component diagrams for the above applications.

Task 10: Design the behavioral diagrams for a case study of student's choice.

Task 11: Implement a Junit Test program and design test cases to find the maximum of an array of numbers.

Task 12: Implement a Junit Test program and design test cases to count the number of elements in array of numbers.

TEXT BOOKS:

1. The Unified Modeling Language User Guide, Grady Booch, James Rumbaugh, Ivar Jacobson, Pearson Education.

2. UML 2 Toolkit, Hans-Erik Eriksson, Magnus Penker, Brian Lyons, David Fado, WILEYDreamtech India Pvt. Ltd.

REFERENCE BOOKS:

1. Bernd Bruegge& Allen H. Dutoit, "Object-Oriented Software Engineering", 2009.
2. Ivar Jacobson, "Object-Oriented Software Engineering", Pearson Education, 2009.
3. Shekhar Gulati, Rahul Sharma "Java Unit Testing with JUnit 5: Test Driven Development with JUnit
   5", Apress, 2017.

TABLE OF CONTENTS

S.No	Content	Page No.

Task 1
Prepare problem statement for any application.

The problem statement is the initial starting point for a project. It is basically a one to three page statement that everyone on the project agrees with that describes what will be done at a high level. The problem statement is intended for a broad audience and should be written in nontechnical terms. It helps the non-technical and technical personnel communicate by providing a description of a problem. It doesn't describe the solution to the problem. The input to requirement engineering is the problem statement prepared by customer. It may give an overview of the existing system along with broad expectations from the new system. The first phase of requirements engineering begins with requirements elicitation i.e. gathering of information about requirements. Here, requirements are identified with the help of customer and existing system processes. So from here begins the preparation of problem statement. So, basically a problem statement describes what needs to be done without describing how.

The library management system is a software system that issues books and magazines to registered students only. The student has to login after getting registered to the system. The borrower of the book can perform various functions such as searching for desired book, get the issued book and return the book.

Software has to be developed for **automating the manual reservation system** of railway. The system should be standalone in nature. It should be designed to provide functionalities like booking of tickets in which a user should be able to apply for tickets of any train and of any class. A limitation is imposed when the numbers of tickets for which user apply is greater than available seats or no seats are available. If seats are not available then put user transaction in the waiting queue. If the tickets are available then it is issued to the user and it must be updated in the database concurrently. The system generates the receipt for the same. The software takes the current system date and time as the date of issue and calculates the amount to be paid by the user. It also provide the functionality of cancellation of tickets .If the user wants to cancel the tickets, he/she must enter the details. The system checks the records from the database if it is matched with the user entered details, and then it cancels the tickets. The system also calculates the amount to be return to the user after deductions. The system must update the database for the same. After that system must check for waiting passenger for that train, if any then these tickets are issued to waiting passenger and update the database. The system displays the details of train of which user enter the name. The information is saved and the corresponding updating take place in the database. In the enquiry, the system should be able to provide information like the availability of tickets of particular train, train schedule. The system should be able to reserve a ticket for a particular user if the tickets are not currently available. The corresponding print outs for each entry (issue/cancel) in the system should be generated.

There should be proper information if the waiting list ticket is confirmed, through mail or via sms. It should tell us as to which all stations it haults and current status of the train should be informed. Security provisions like the login authenticity should be provided. Each user should have a user id and a password. Record of the users of the system should be kept in the log file. Provision should be made for full backup of the system.

The client for the project is interested in receiving an upgrade for their **Warehouse management system.** The mission of the project is to modernize, increase efficiency and reduce errors that take place in their supply chain system. Currently the clients are using a fax based system to send and receive orders. This system is crucial to the company because they need it to receive parts that are required in their product. The company is looking to modernize this system and move their records into a database. Their reason for wanting to do this is because currently they are experiencing manual errors caused by employees. They would like for us to increase processing speed, reduce errors and create more standardization.

Task 2
Develop Software Requirement Specification (SRS) for above applications.

Procedure:
Step 1:
Introduction:
Purpose

A project has a purpose, certain benefits that are targeted. In order to achieve the benefits, certain deliverables need to be produced during the project. Both these goals and

objectives should be defined early in the project. The project purpose statement is often the first thing that someone interested in learning more about a project might read or hear from a project team member or another project stakeholder. This purpose statement, like the proverbial elevator speech, should in a few words describe what the project is all about. The purpose statement is part of the project charter that is created during the project initiation phase. A project purpose statement is relatively short, typically just a few sentences. This is what makes writing one hard, because these few sentences often must describe a project that is fairly complex. A great way to start writing a project purpose statement is by writing down the problems and/or opportunities the project is supposed to resolve. When writing these make sure you demonstrate how the problems/opportunities affect the business.

Project Scope

Project scope is the part of project planning that involves determining and documenting a list of specific project goals, deliverables, tasks, costs and deadlines. The documentation of a project's scope, which is called a scope statement or terms of reference, explains the boundaries of the project, establishes responsibilities for each team member and sets up procedures for how completed work will be verified and approved.
During the project, this documentation helps the project team remain focused and on task. The scope statement also provides the team with guidelines for making decisions about change requests during the project.

It is important to pin down the scope early in a project's life cycle as it can greatly impact the schedule or cost (or both) of the project down the track.
A project scope statement is a written document that includes all the required information for producing the project deliverables. The project scope statement is more detailed than a statement of work; it helps the project team remain focused and on task. The scope statement also provides the project team leader or facilitator with guidelines for making decisions about change requests during the project.

Step 2:

Overall Description

The overall description gives an overview of the requirements and other subsections. The requirements will be described in greater detail in the specific requirements section. The function of the overall description is to consider determining factors that impact the requirements.

Subsections of the overall description are product perspective, design constraints, product functions, user characteristics and constraints, assumptions, and dependencies. These all have to do with anticipating the needs and challenges that stand in the way of completing the requirements. Design constraints, for example, includes everything from consideration of software compliance to hardware constraints.

Product Perspective

Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.

Product Features

Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, is often effective.

User Classes and Characteristics

Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.

Operating Environment

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

Design and Implementation Constraints

Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).

Step 3:

System Features

This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

System Feature 1

Don't really say "System Feature 1." State the feature name in just a few words.

1 Description and Priority

Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component

ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).

2 Stimulus/Response Sequences

List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.

3 Functional Requirements

Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary.

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:
REQ-2:

Step 4:

External Interface Requirements

User Interfaces

Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.

Hardware Interfaces

Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

Software Interfaces

Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

Communications Interfaces

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

Non functional Requirements:

Performance Requirements

If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

Safety Requirements

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.

Security Requirements

Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

Software Quality Attributes

Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

Other Requirements

Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.

SRS Report for Library Management System

INTRODUCTION

With the increase in the number of readers, better management of libraries system is required. The Library management system focuses on improving the management of libraries in a city or town. “What If you can check whether a book is available in the library through your phone?” or “what if instead of having different library cards for different libraries you can just have one ?” or “you can reserve a book or issue a book from your phone sitting at your home!”. The Integrated Library Management system provides you the ease of issuing, renewing, or reserving a book from an library within your town through your phone. The Integrated Library Management system is developed on the android platform which basically focuses on issuing, renewing and reserving a book.

PURPOSE

The purpose of the project is to maintain the details of books and library members of different libraries. The main purpose of this project is to maintain a easy circulation system between clients and the libraries, to issue books using single library card, also to search and reserve any book from different available libraries and to maintain details about the user (fine, address, phone number).Moreover, the user can check all these features from their home.

SCOPE

- Manually updating the library system into an android based application so that the user can know the details of the books available and maximum limit on borrowing from their computer and also through their phones.

- The LM System provides information's like details of the books, insertion of new books, deletion of lost books, limitation on issuing books, fine on keeping a book more than one month from the issued date.

- Also user can provide feedback for adding some new books to the library.

Definition, Acronyms, Abbreviation:

- JAVA -> platform independence
- DFD -> Data Flow Diagram
- ER -> Entity Relationship
- IDE -> Integrated Development Environment
- SRS -> Software Requirement Specification

OVERALL DESCRIPTION

PRODUCT PRESPECTIVE

The proposed Library Management System will take care of the current book detail at any point of time. The book issue, book return will update the current book details automatically so that user will get the update current book details.

FUNCTIONAL REQUIREMENT

R.1:Register

Description : First the user will have to register/sign up. There are two different type of users.
The library manager/head : The manager have to provide details about the name of library ,address, phone number, email id.
Regular person/student : The user have to provide details about his/her name of address, phone number, email id.

R.1.1: Sign up

Input: Detail about the user as mentioned in the description.
Output: Confirmation of registration status and a membership number and password will be generated and mailed to the user.

Processing: All details will be checked and if any error are found then an error message is displayed else a membership number and password will be generated.
R.1.2 : Login
Input: Enter the membership number and password provided.
Output : User will be able to use the features of software.

R.2: Manage books by user.

R.2.1: Books issued.
Description : List of books will be displaced along with data of return.

R.2.2: Search
Input: Enter the name of author's name of the books to be issued.
Output: List of books related to the keyword.

R.2.3 : Issues book
State : Searched the book user wants to issues.
Input : click the book user wants.
Output : conformation for book issue and apology for failure in issue.
Processing : if selected book is available then book will be issued else error will be displayed.

R.2.4 : Renew book
State : Book is issued and is about to reach the date of return.
Input : Select the book to be renewed.
Output : conformation message.
Processing : If the issued book is already reserved by another user then error message will be send and if not then conformation message will be displayed.

R.2.5 : Return
Input ; Return the book to the library.
Output : The issued list will be updated and the returned book will be listed out.

R.2.6 ; Reserve book
Input ; Enter the details of the book.
Output : Book successfully reserved.

Description : If a book is issued by someone then the user can reserve it ,so that later the user can issue it.

R.2.6 Fine
Input : check for the fines.
Output : Details about fines on different books issued by the user.
Processing : The fine will be calculated, if it crossed the date of return and the user did not renewed if then fine will be applied by Rs 10 per day.

R.3 Manage book by librarian

R.3.1 Update details of books

R.3.1.1 Add books
Input : Enter the details of the books such as names ,author ,edition, quantity.
Output : confirmation of addition.

R.3.1.2 Remove books
Input : Enter the name of the book and quantity of books.
Output : Update the list of the books available.

NON FUNCTIONAL REQUIREMENTS

Availability Requirement

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

Efficiency Requirement
Mean Time to Repair (MTTR) - Even if the system fails, the system will be recovered back up within an hour or less.

Accuracy

The system should accurately provide real time information taking into consideration various concurrency issues. The system shall provide 100% access reliability.

Performance Requirement

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

Reliability Requirement

The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week, 24 hours a day.

USER CHARACTERSTICS

We have 3 levels of users :

User module: In the user module, user will check the availability of the books.
Issue book
Reserve book
Return book
Fine details

Library module:

Add new book
Remove books
Update details of book

Administration module:

The following are the sub module in the administration module:
Register user
Entry book details
Book issue

CONSTRAINTS

Any update regarding the book from the library is to be recorded to have update & correct values, and any fine on a member should be notified as soon as possible and should be correctly calculated.

Task 3: Class diagrams for Library, Railway Reservation and Warehouse Management System

Overview:

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

Purpose:

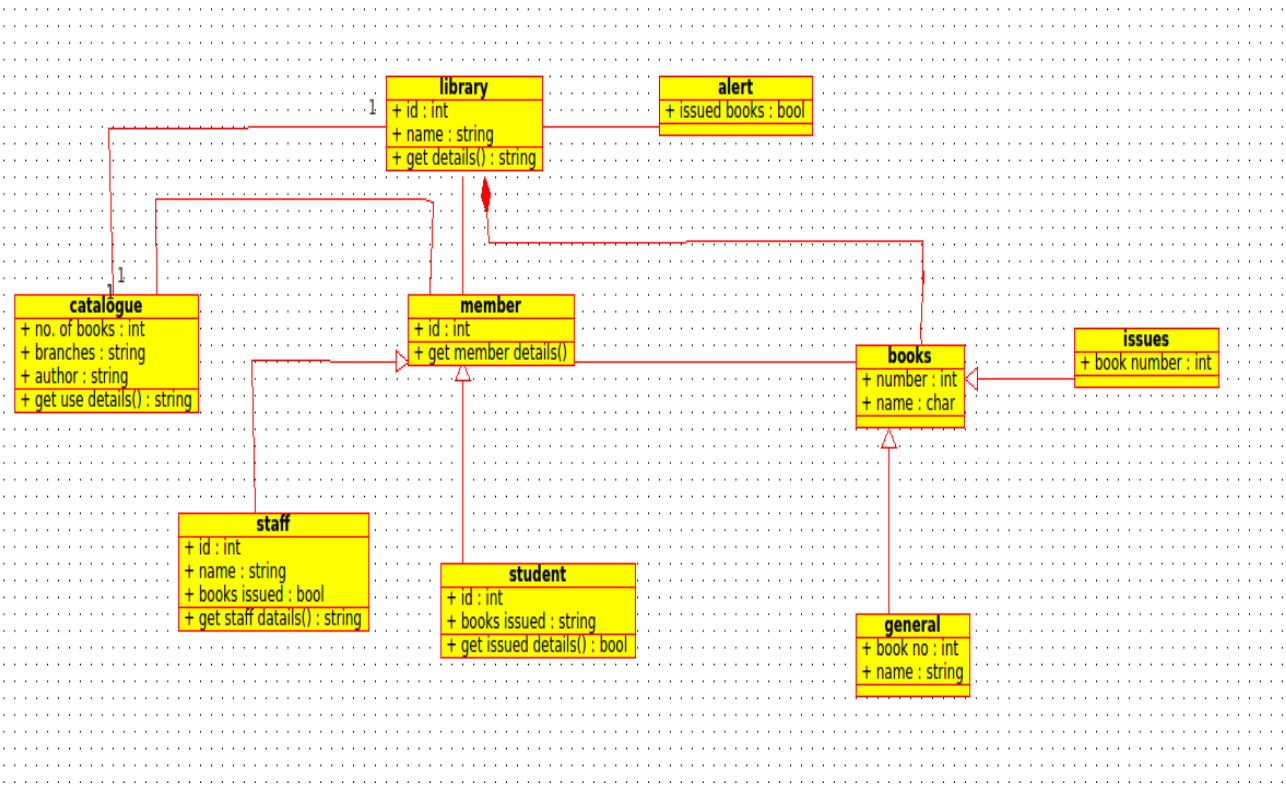
The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction. The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community. So the purpose of the class diagram can be summarized as:

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

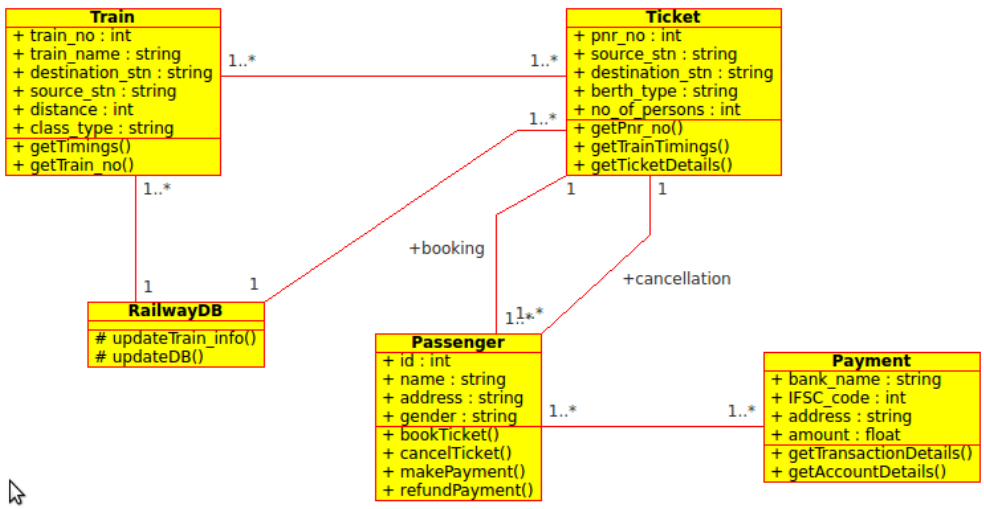
Contents:

Class diagrams commonly contain the following things

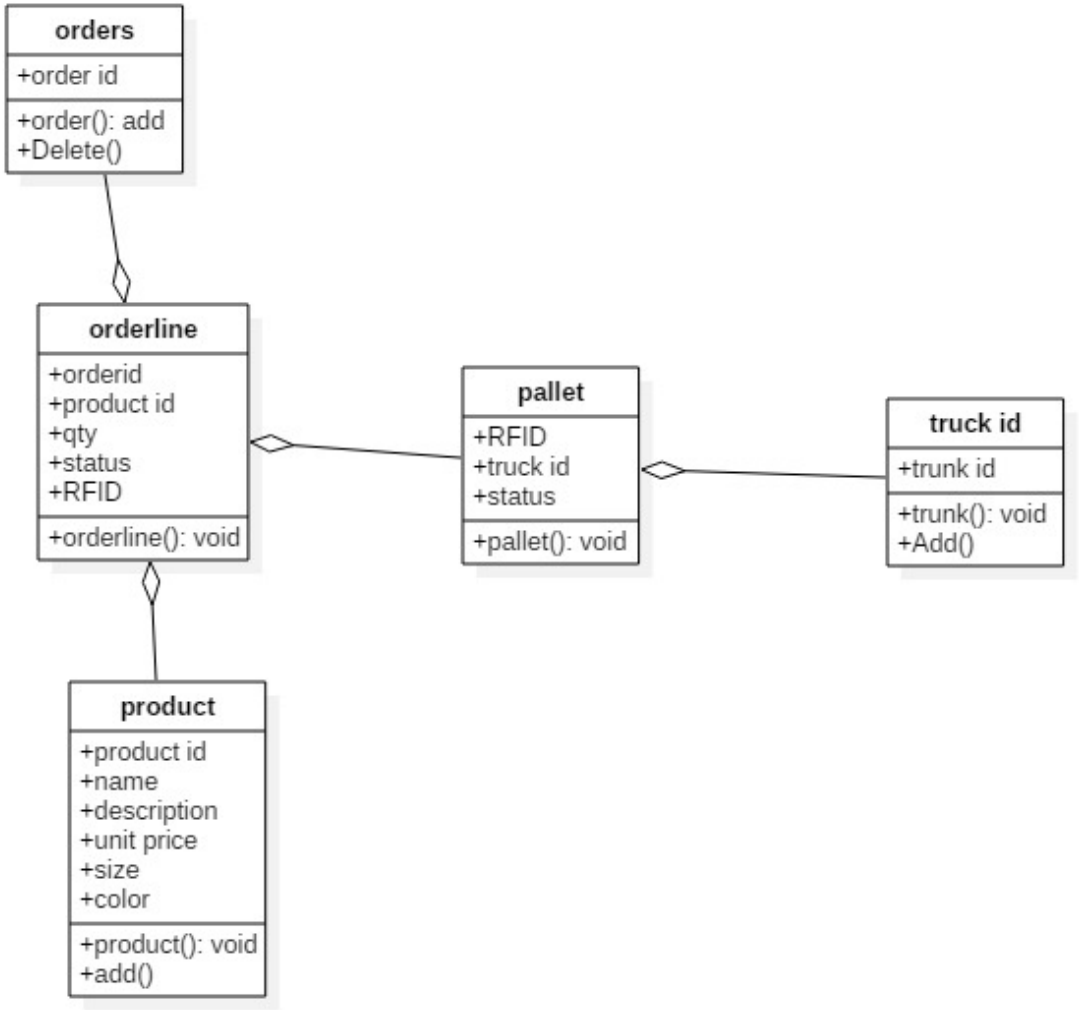
- Classes
- Interfaces
- Collaborations
- Dependency, generalization and association relationships



Class diagram for Library Management System



Class diagram for Railway Reservation System



Class diagram for Warehouse Management System

Task 4: Use case diagram for Library, Railway Reservation and Warehouse Management System

Overview:

To model a system the most important aspect is to capture the dynamic behaviour. Use case diagram is dynamic in nature there should be some internal or external factors for making the interaction. These internal and external agents are known as actors.

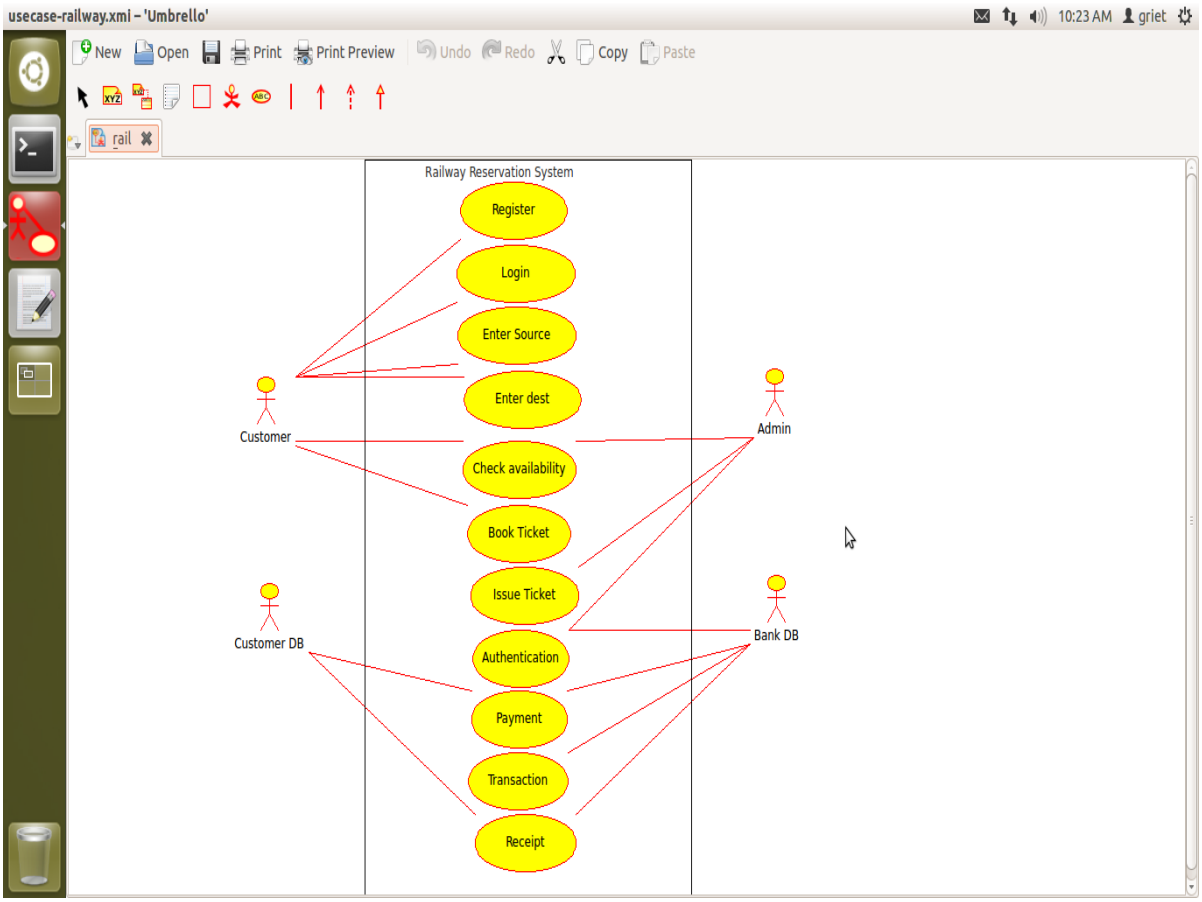
Contents: actors, use cases and their relationships.

The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Purpose:

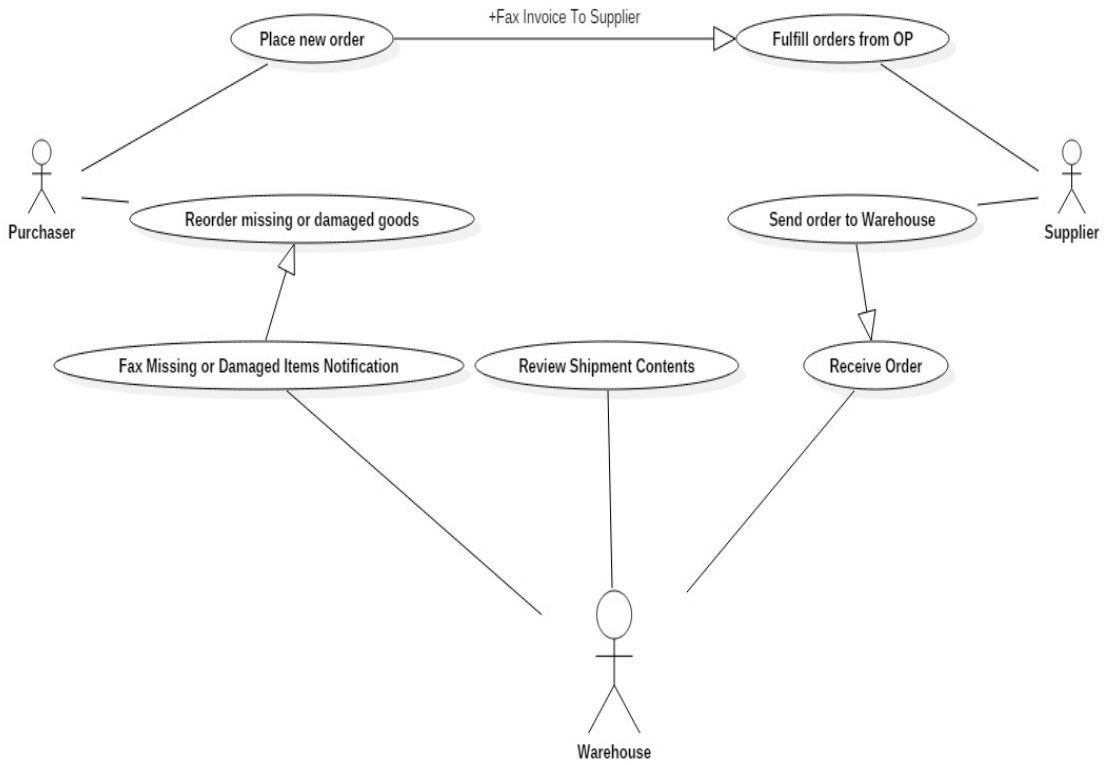
The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Because other four diagrams (activity, sequence, collaboration and Statechart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.
So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors



Usecase diagram for Railway management system

Usecase diagram for Library



Usecase diagram for Warehouse Management

Task 5: Interaction diagrams for Library, Railway Reservation and Warehouse Management System

Interaction it is clear that the diagram is used to describe some type of interactions among the different elements in the model. So this interaction is a part of dynamic behaviour of the system.

This interactive behaviour is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purposes of both the diagrams are similar.

Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

Purpose:

The purposes of interaction diagrams are to visualize the interactive behaviour of the system. Now visualizing interaction is a difficult task. So the solution is to use different types of models to capture the different aspects of the interaction.

That is why sequence and collaboration diagrams are used to capture dynamic nature but from a different angle.

So the purposes of interaction diagram can be describes as:

- To capture dynamic behaviour of a system.
- To describe the message flow in the system.
- To describe structural organization of the objects.
- To describe interaction among objects.

The second interaction diagram is collaboration diagram. It shows the object organization as shown below. Here in collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram.

The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.

Following are the usages of interaction diagrams:

- To model flow of control by time sequence.
- To model flow of control by structural organizations.
- For forward engineering.
- For reverse engineering.

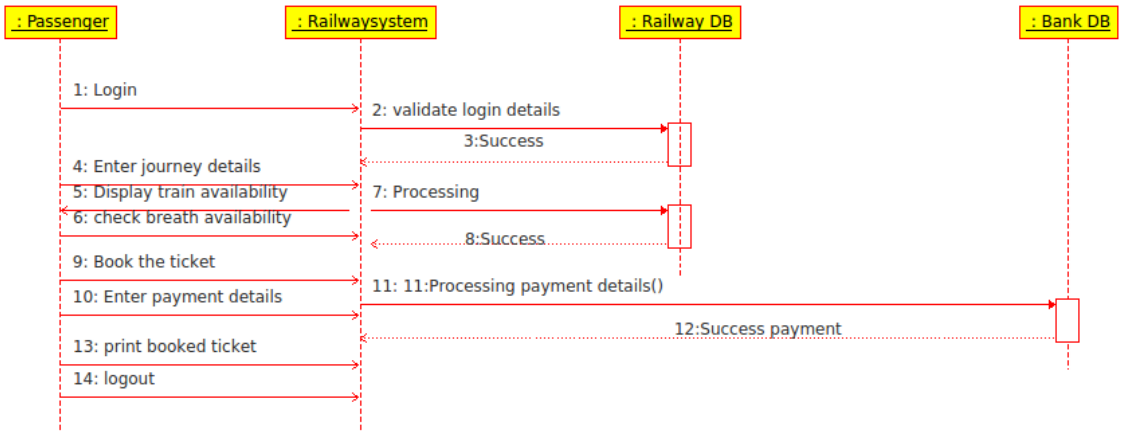


Fig: Sequence diagram for Railway Reservation System

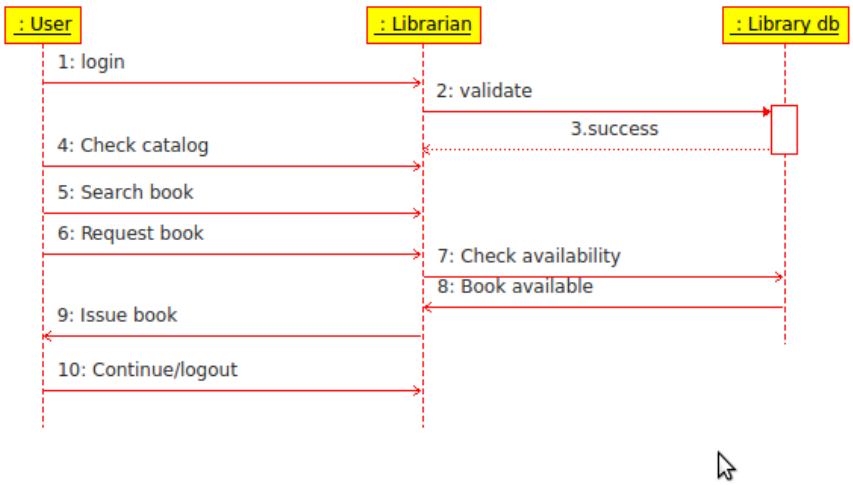


Fig: Sequence diagram for Library Management System

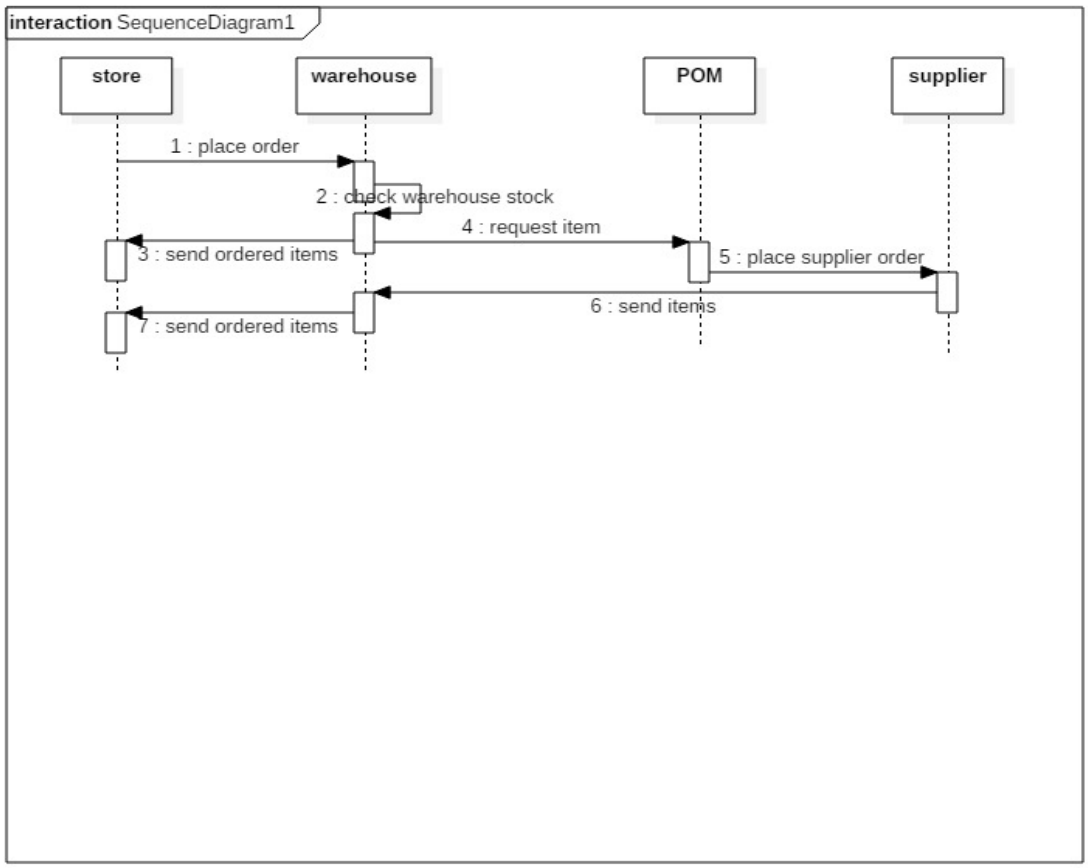


Fig: Sequence diagram for Warehouse Management System

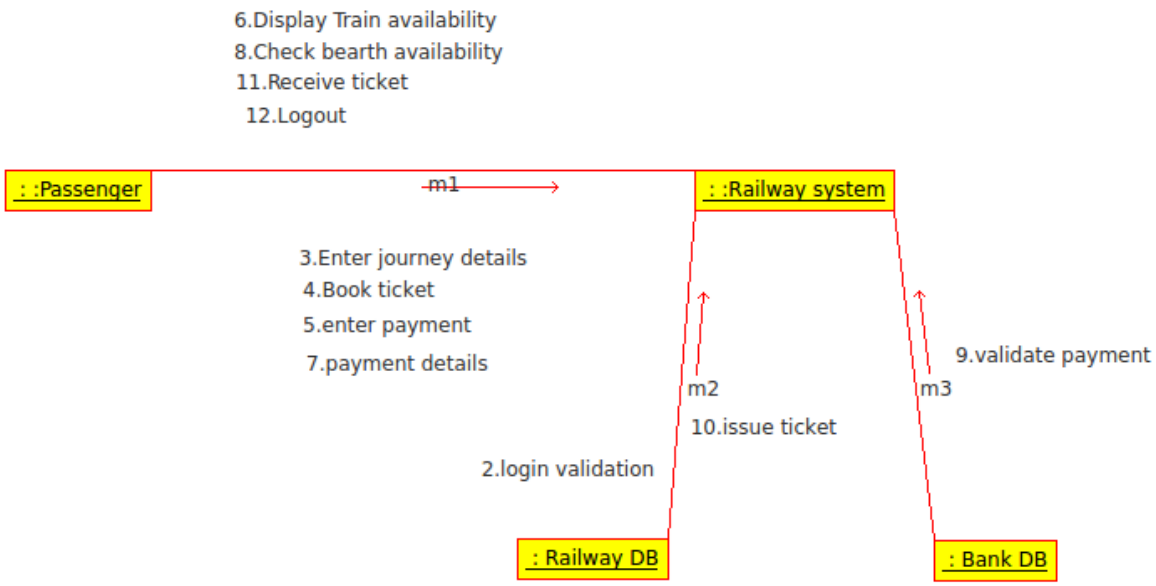


Fig: Collaboration diagram for Railway Reservation System

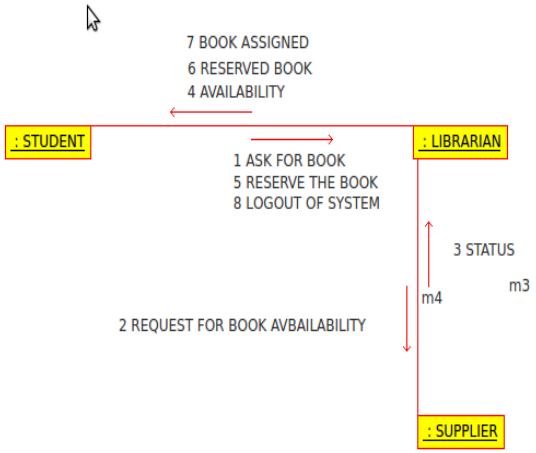


Fig: Collaboration diagram for Library Management System

Task 6: Activity diagram for Library, Railway Reservation and Warehouse Management System

Activity diagram is basically a flow chart to represent the flow form one activity to another . The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow by using elements like fork, join etc.

Contents

Initial/Final State, Activity, Fork & Join, Branch, Swimlanes

Fork

A fork represents the splitting of a single flow of control into two or more concurrentFlow of control. A fork may have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control. Below fork the activities associated with each of these path continues in parallel.

Join

A join represents the synchronization of two or more concurrent flows of control. A join may have two or more incoming transition and one outgoing transition. Above the join the activities associated with each of these paths continues in parallel.

Branching

A branch specifies alternate paths takes based on some Boolean expression Branch is represented by diamond Branch may have one incoming transition and two or more outgoing one on each outgoing transition, you place a Boolean expression shouldn’t overlap but they should cover all possibilities.

Swimlane:

Swimlanes are useful when we model workflows of business processes to partition the activity states on an activity diagram into groups. Each group representing the business organization responsible for those activities ,these groups are called Swimlanes .

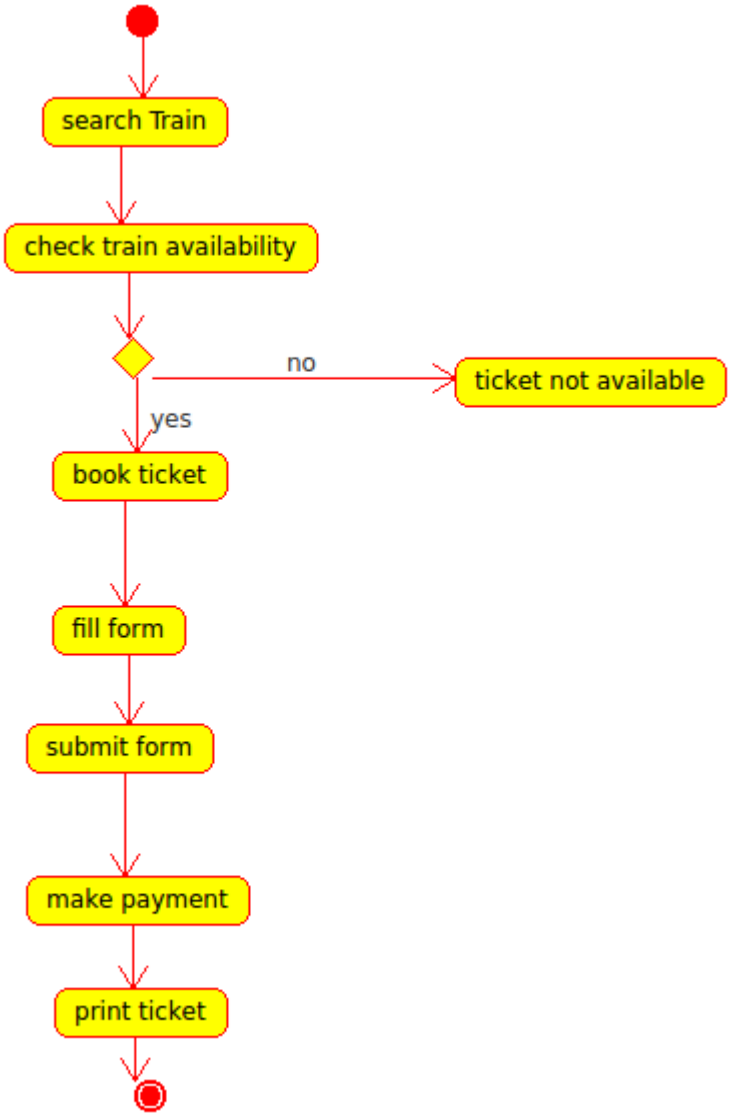


Fig: Activity diagram for Railway Reservation System

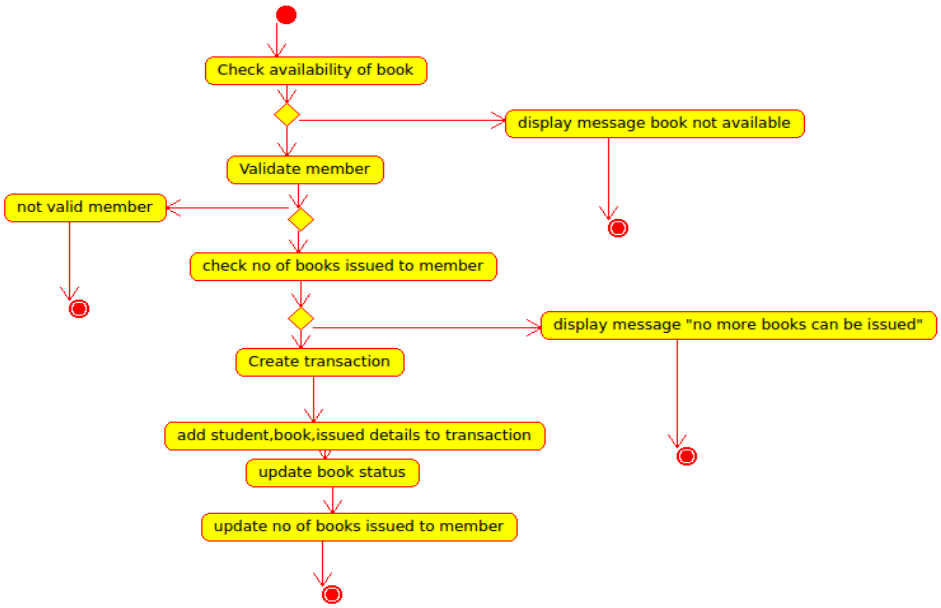


Fig: Activity diagram for Library

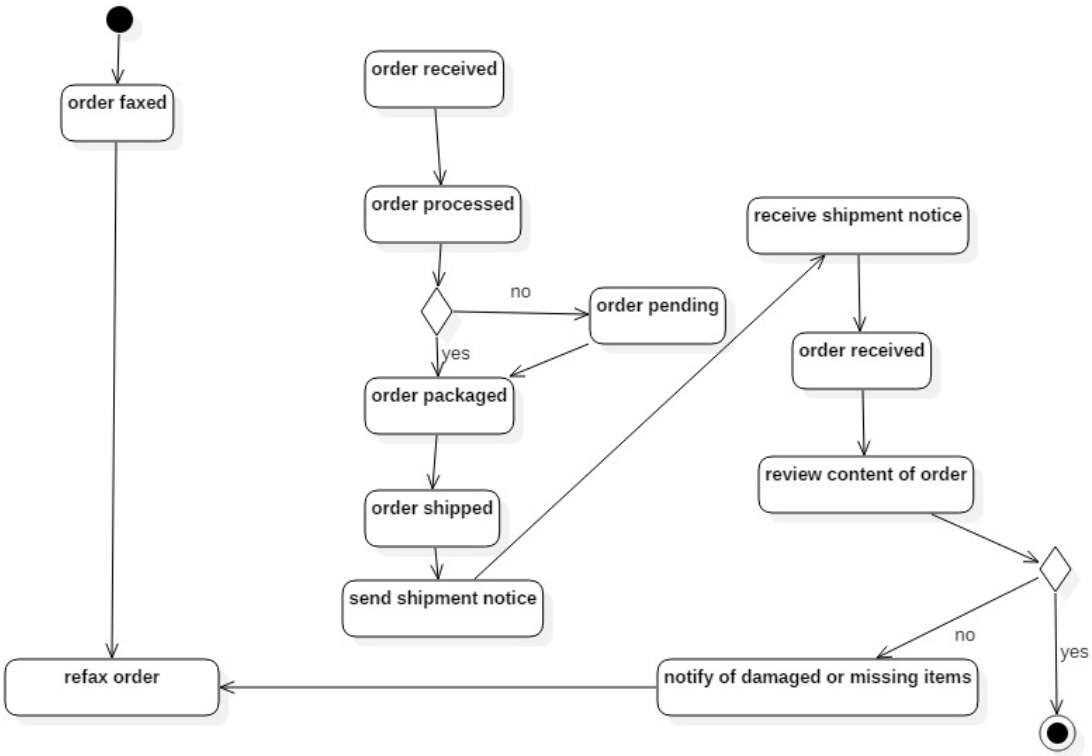


Fig: Activity diagram for Warehouse Management

Task 7: State chart diagram for Library, Railway Reservation and Warehouse Management System

State chart diagram is used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. So Statechart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

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. So the most important purpose of Statechart diagram is to model life time of an object from creation to termination.

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

Following are the main purposes of using Statechart diagrams:

1. To model dynamic aspect of a system.
2. To model life time of a reactive system.
3. To describe different states of an object during its life time.
4. Define a state machine to model states of an object.

Contents

Simply state and composite states

Transitions, including events and actions

Common use

They are use to model the dynamic aspects of a system.

Event ordered behavior of any kind of objects, to model reactive objects.

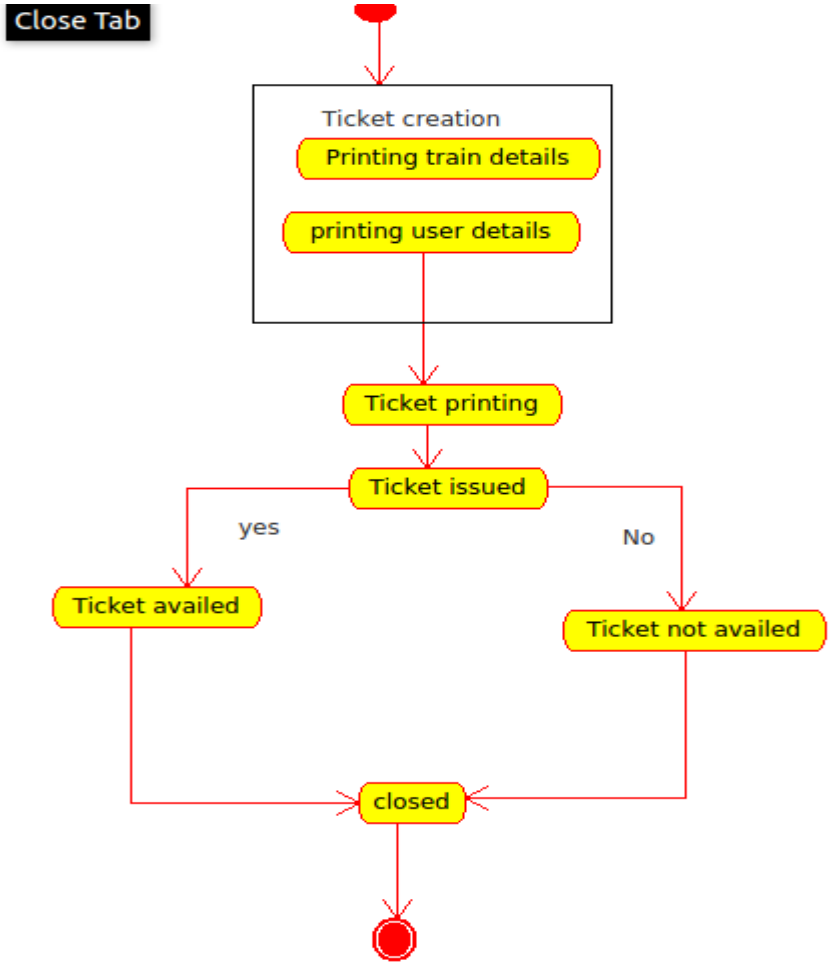


Fig: State chart diagram for Railway Reservation System

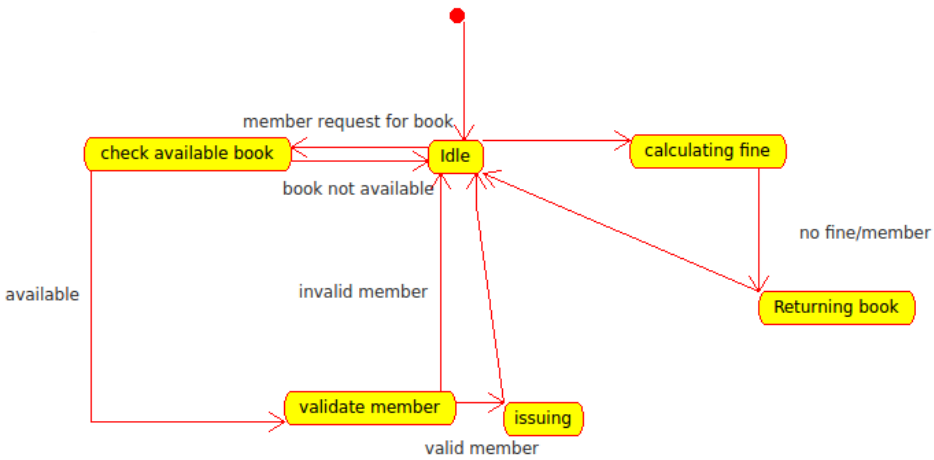


Fig: State chart diagram for Library Management System

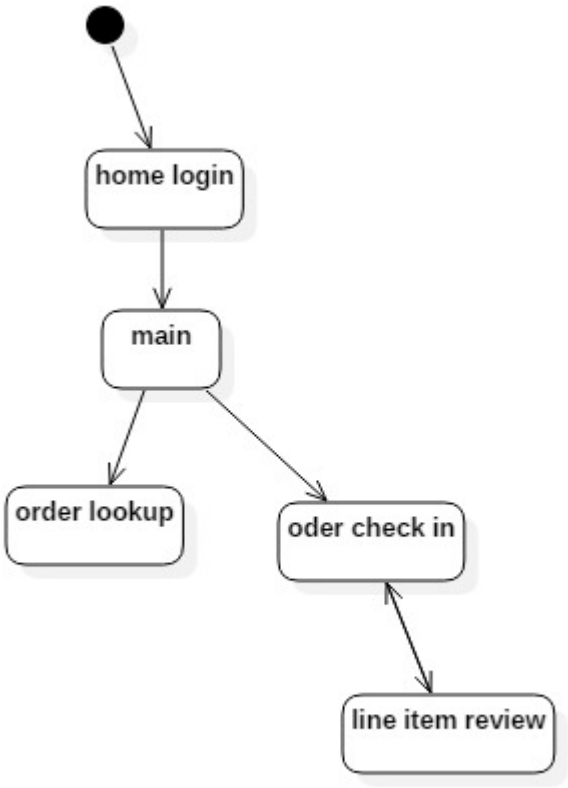


Fig: State chart diagram for Warehouse Management System

Task 8: Perform forward and Reverse Engineering, and Generate documentation of the project.

Forward and Reverse Engineering:

Forward engineering is the process of transforming a model into code through a mapping to an implementation language. Forward engineering results in a loss of information, because models written in the UML are semantically richer than any current object-oriented programming language. In fact, this is a major reason why you need models in addition to code. Structural features, such as collaborations, and behavioral features, such as interactions, can be visualized clearly in the UML, but not so clearly from raw code.

To forward engineer a class diagram

• Identify the rules for mapping to your implementation language or languages of choice. This is something you'll want to do for your project or your organization as a whole.

• Depending on the semantics of the languages you choose, you may want to constrain your use of certain UML features. For example, the UML permits you to model multiple inheritance, but Smalltalk permits only single inheritance. You can choose to prohibit developers from modeling with multiple inheritance (which makes your models language-dependent), or you can develop idioms that transform these richer features into the implementation language (which makes the mapping more complex).

• Use tagged values to guide implementation choices in your target language. You can do this at the level of individual classes if you need precise control. You can also do so at a higher level, such as with collaborations or packages.

• Use tools to generate code.

```
public abstract class EventHandler {
 EventHandler successor;
private Integer currentEventID;
private String source; EventHandler() {}
public void handleRequest() {}
 }
```

Reverse engineering is the process of transforming code into a model through a mapping from a specific implementation language. Reverse engineering results in a flood of information, some of which is at a lower level of detail than you'll need to build useful models. At the same time, reverse engineering is incomplete. There is a loss of information when forward engineering models into code, and so you can't completely recreate a model from code unless your tools encode information in the source comments that goes beyond the semantics of the implementation language.

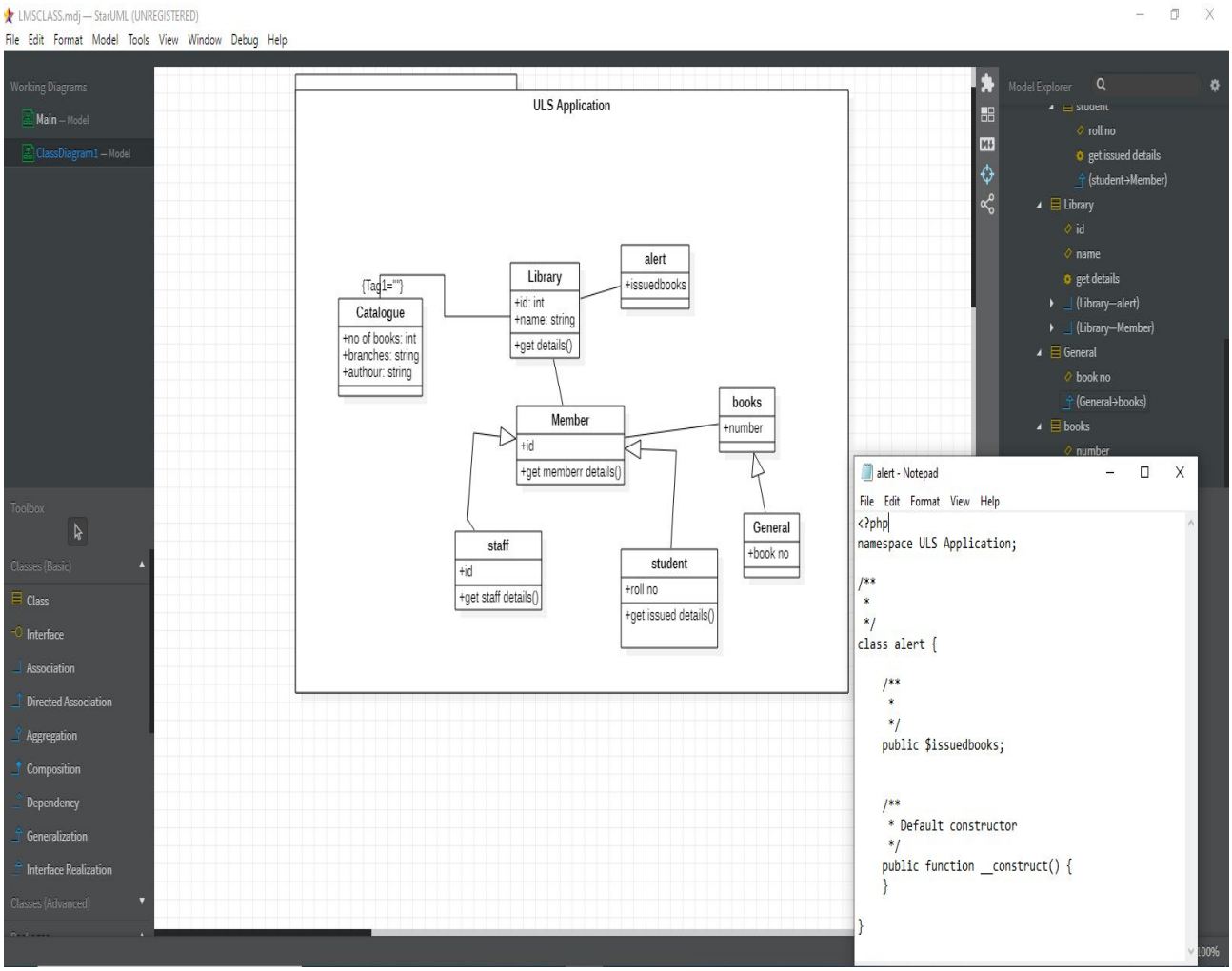
To reverse engineer a class diagram

•Identify the rules for mapping from your implementation language or languages of choice. This is something you'll want to do for your project or your organization as a whole.

• Using a tool, point to the code you'd like to reverse engineer. Use your tool to generate a new model or modify an existing one that was previously forward engineered. It is unreasonable to expect to reverse engineer a single concise model from a large body of code. You need to select portion of the code and build the model from the bottom.

• Using your tool, create a class diagram by querying the model. For example, you might start with one or more classes, then expand the diagram by following specific relationships or other neighboring classes. Expose or hide details of the contents of this class diagram as necessary to communicate your intent.

• Manually add design information to the model to express the intent of the design that is missing or hidden in the code.



Forward Engineer a Class Diagram for Library Management System

Umbrello UML Modeller is a UML modelling tool, and as such its main purpose is to help you in the analysis and design of your systems. However, to make the transition between your design and your implementation, Umbrello UML Modeller allows you to generate

source code in different programming languages to get you started. Also, if you want to start using UML in an already started C++ project, Umbrello UML Modeller can help you create a model of your system from the source code by analysing your source code and importing the classes found in it.

Code Generation

Umbrello UML Modeller can generate source code for various programming languages based on your UML Model to help you get started with the implementation of your project. The code generated consists of the class declarations, with their methods and attributes so you can "fill in the blanks" by providing the functionality of your classes' operations.

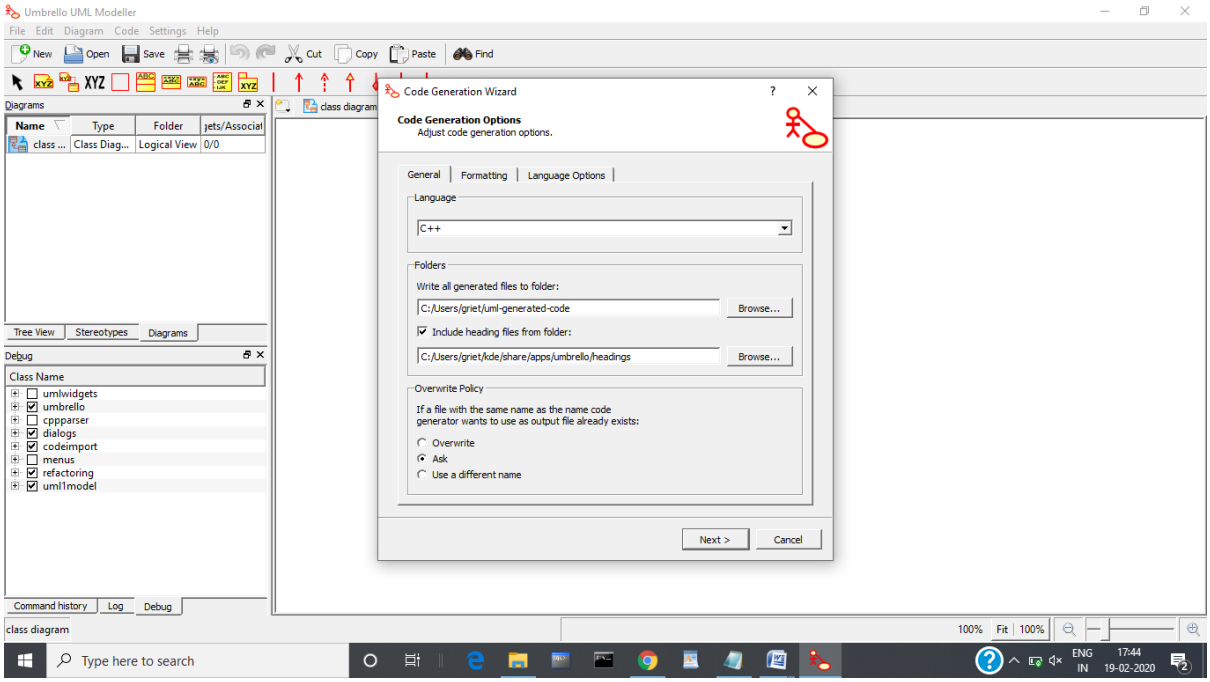
Umbrello UML Modeller 2 comes with code generation support for ActionScript, Ada, C++, C#, D, IDL, Java™, JavaScript, MySQL and Pascal.

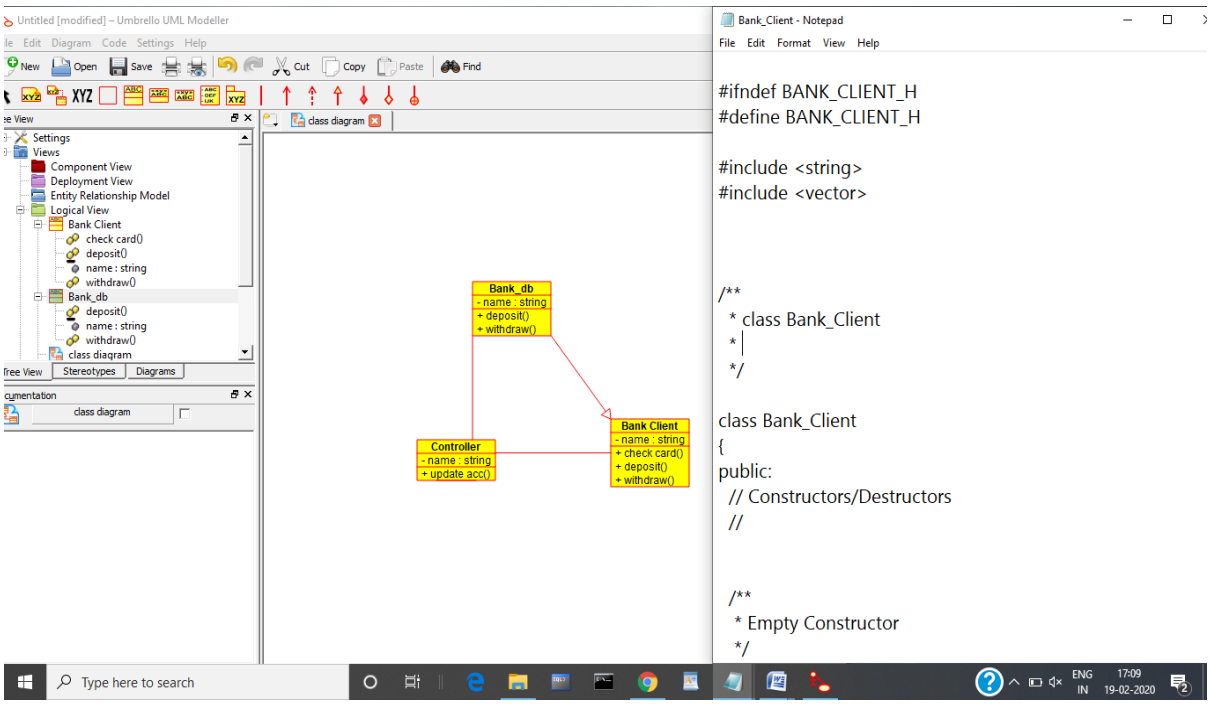
Generating Code

In order to generate code with Umbrello UML Modeller, you first need to create or load a Model containing at least one class. When you are ready to start writing some code, select the Code Generation Wizard entry from the Code menu to start a wizard which will guide you trough the code generation process.

The first step is to select the classes for which you want to generate source code. By default all the classes of your model are selected, and you can remove the ones for which you do not want to generate code by moving them to the left-hand side list.

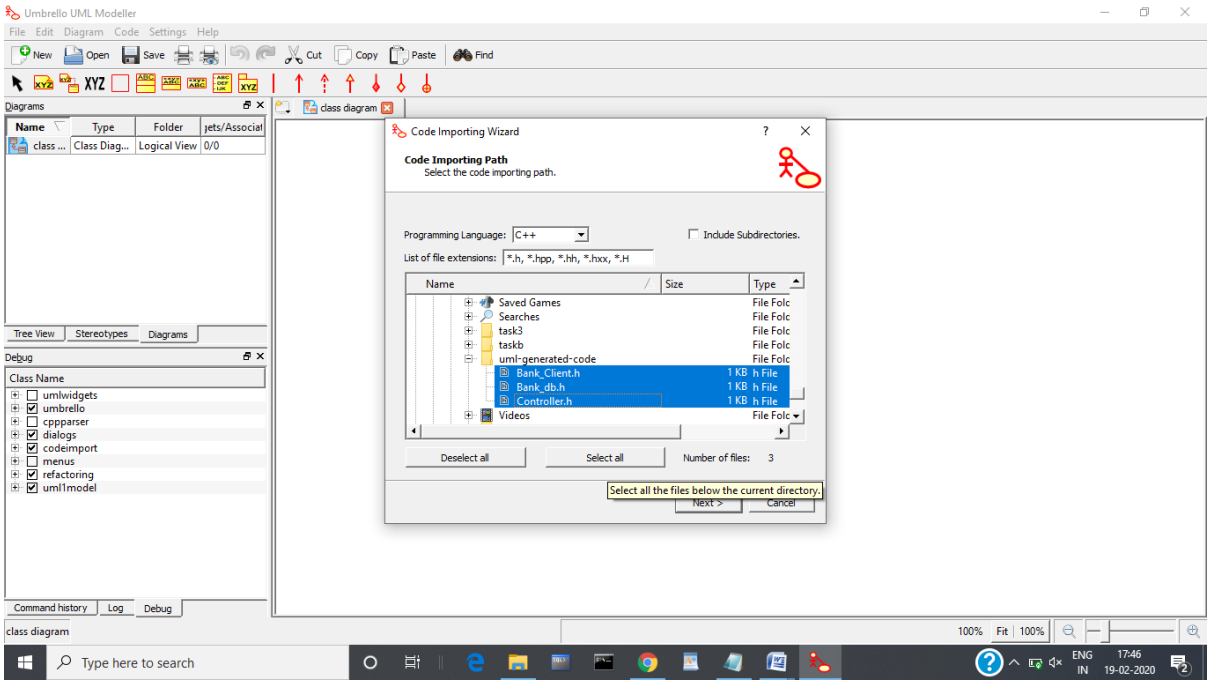
The next step of the wizard allows you to modify the parameters the Code Generator uses while writing your code. The following options are available:



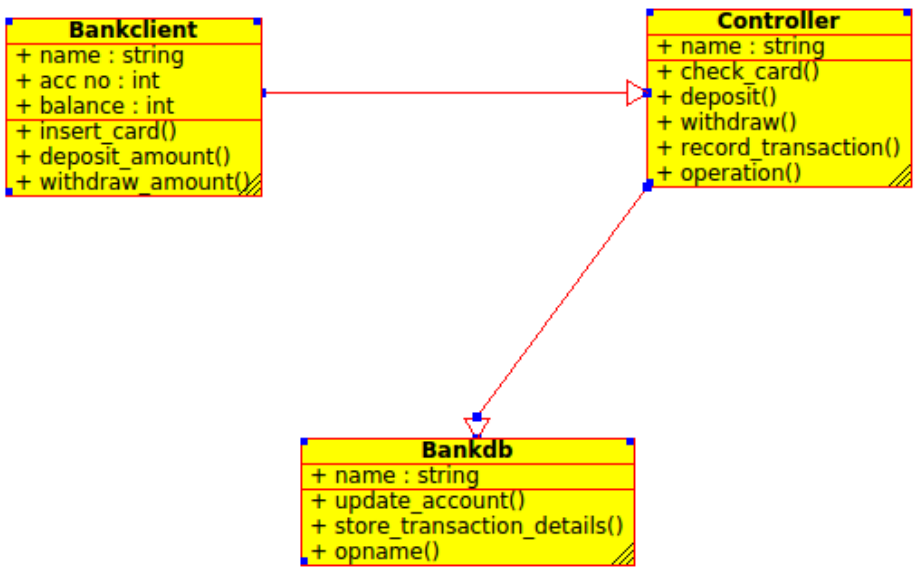


Forward Engineer a Class Diagram for ATM

Umbrello UML Modeller can help you create a model of your system from the source code by analysing your source code and importing the classes found in it.



Code Importing Path Using Umbrello UML Modeller for Reverse Engineering



Class diagram for ATM

Task 9: Component Diagram for Library, Railway Reservation and Warehouse Management System

Component diagrams are used to model physical aspects of a system. Now the question is what are these physical aspects? Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node. So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose:

Component diagrams can be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of

diagrams are used to represent the whole.

Before drawing a component diagram the following artifacts are to be identified clearly:

- Files used in the system.
- Libraries and other artifacts relevant to the application.
- Relationships among the artifacts.
- Now after identifying the artifacts the following points needs to be followed:
  - Use a meaningful name to identify the component for which the diagram is to be drawn.
  - Prepare a mental layout before producing using tools.
  - Use notes for clarifying important points.

Now the usage of component diagrams can be described as:

1. Model the components of a system.
2. Model database schema.
3. Model executables of an application.
4. Model system's source code.

Contents

Components, Interfaces, Relationships

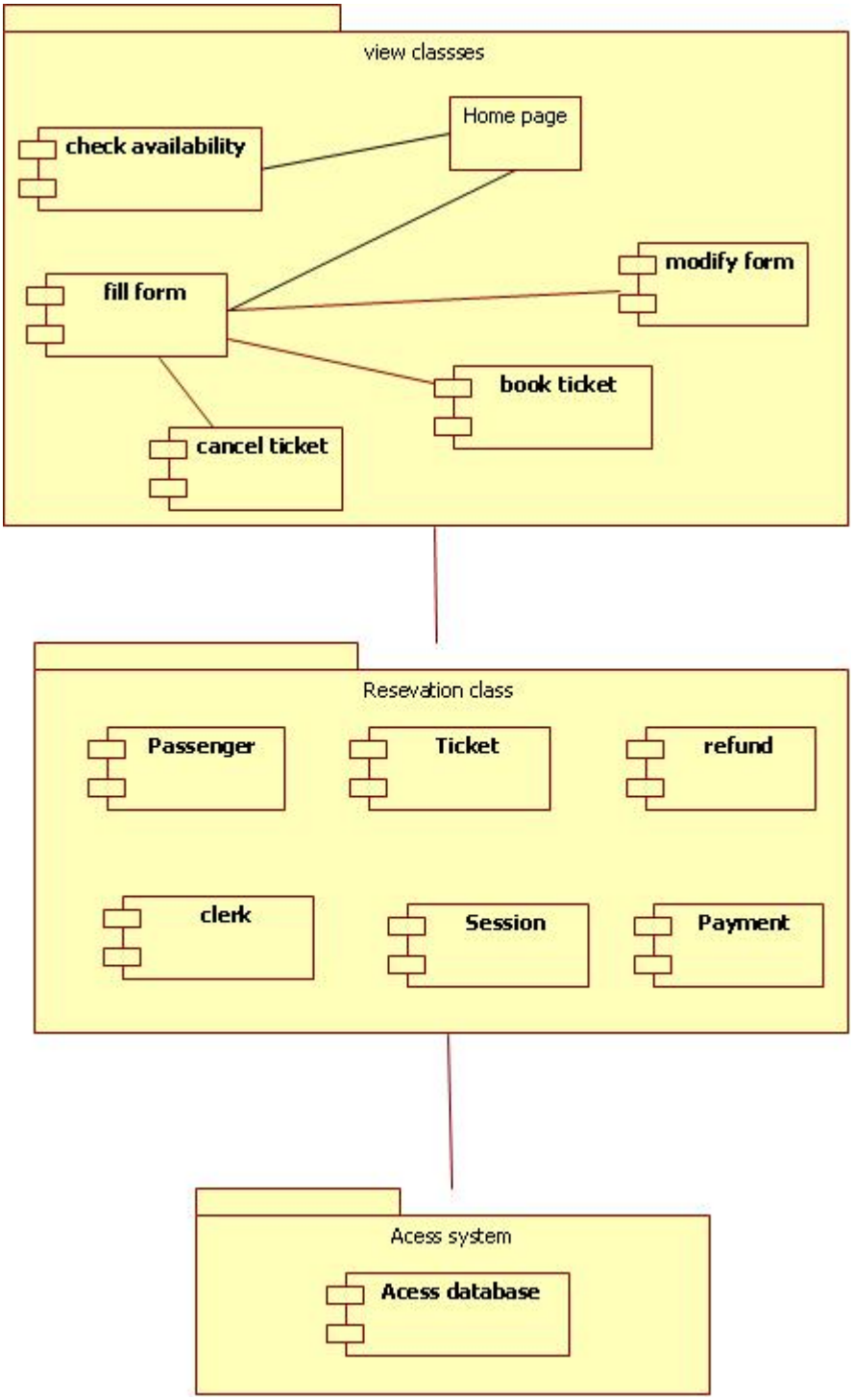


Fig: Component diagram for Railway Reservation System

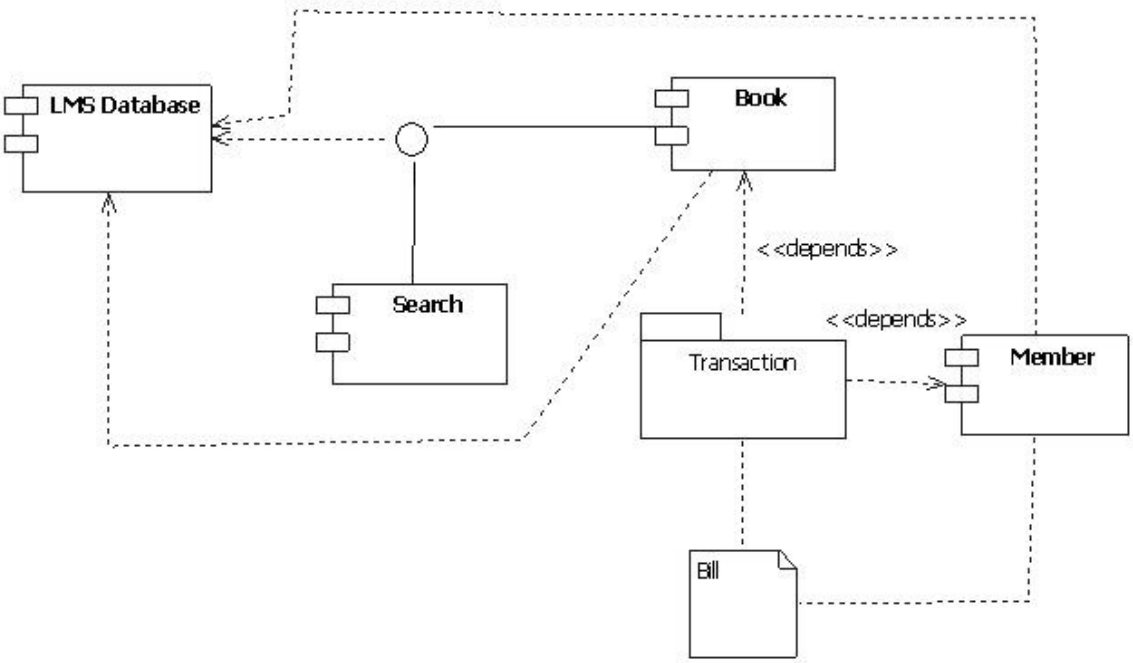


Fig: Component diagram for Library Management System

Task 9: Design the behavioral diagrams for a case study of student's choice.

Automatic Teller Machine

Description of ATM System

The software to be designed will control a simulated automated teller machine (ATM) having a magnetic stripe reader for reading an ATM card, a customer console (keyboard and display) for interaction with the customer, a slot for depositing envelopes, a dispenser for cash , a printer for printing customer receipts, and a key-operated switch to allow an operator to start or stop the machine. The ATM will communicate with the bank's computer over an appropriate communication link. (The software on the latter is not part of the requirements for this problem.)

The ATM will service one customer at a time. A customer will be required to insert an ATM card and enter a personal identification number (PIN) - both of which will be sent to the bank for validation as part of each transaction. The customer will then be able to perform one or more transactions. The card will be retained in the machine until the customer indicates that he/she desires no further transactions, at which point it will be returned - except as noted below.

The ATM must be able to provide the following services to the customer:

1. A customer must be able to make a cash withdrawal from any suitable account linked to the card. Approval must be obtained from the bank before cash is dispensed.
2. A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope. The customer will enter the amount of the deposit into the ATM, subject to manual verification when the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically accepting the envelope.
3. A customer must be able to make a transfer of money between any two accounts linked to the card.
4. A customer must be able to make a balance inquiry of any account linked to the card.
5. A customer must be able to abort a transaction in progress by pressing the Cancel key instead of responding to a request from the machine.

The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. Ordinarily, a transaction will be considered complete by the bank once it has been approved. In the case of a deposit, a second message will be sent to the bank

indicating that the customer has deposited the envelope. (If the customer fails to deposit the envelope within the timeout period, or presses cancel instead, no second message will be sent to the bank and the deposit will not be credited to the customer.)

If the bank determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a transaction can proceed. If the customer is unable to successfully enter the PIN after three tries, the card will be permanently retained by the machine, and the customer will have to contact the bank to get it back.

If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.

The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers).

The ATM will have a key-operated switch that will allow an operator to start and stop the servicing of customers. After turning the switch to the "on" position, the operator will be required to verify and enter the total cash on hand. The machine can only be turned off when it is not servicing a customer. When the switch is moved to the "off" position, the machine will shut down, so that the operator may remove deposit envelopes and reload the machine with cash, blank receipts, etc.

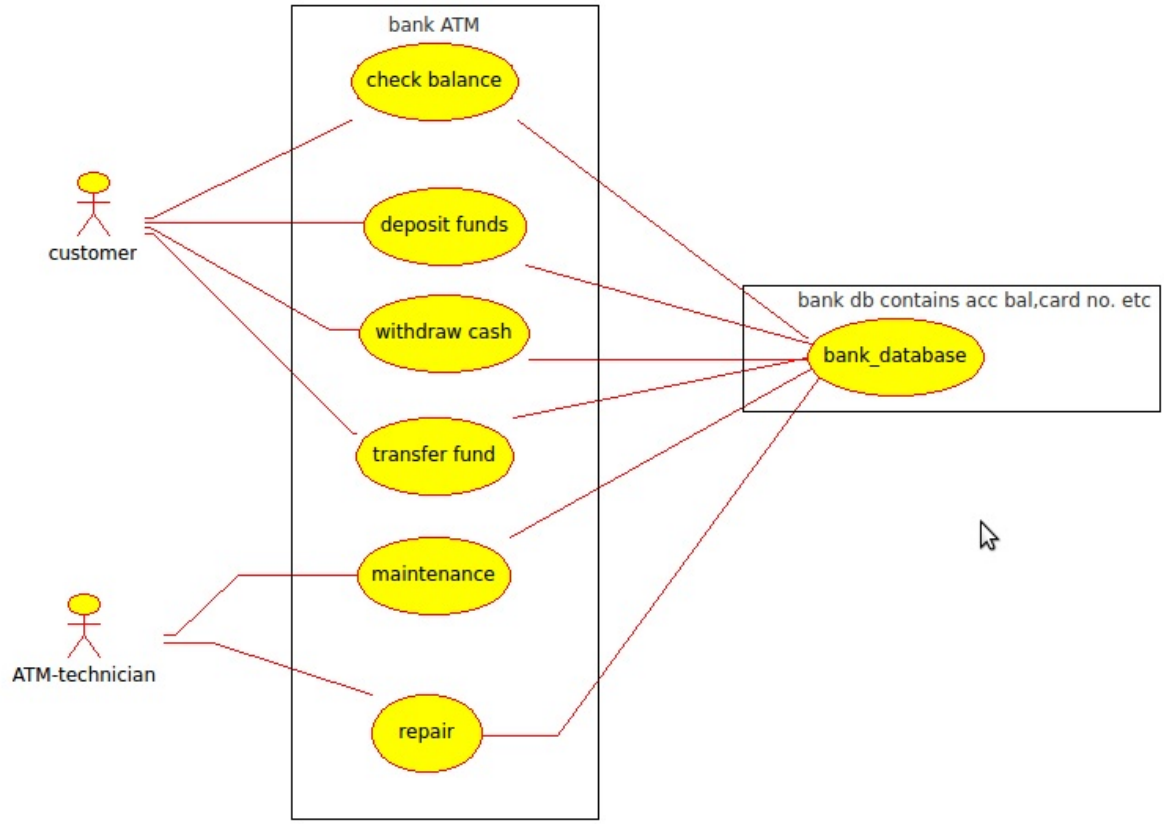


Fig: Usecase diagram for ATM

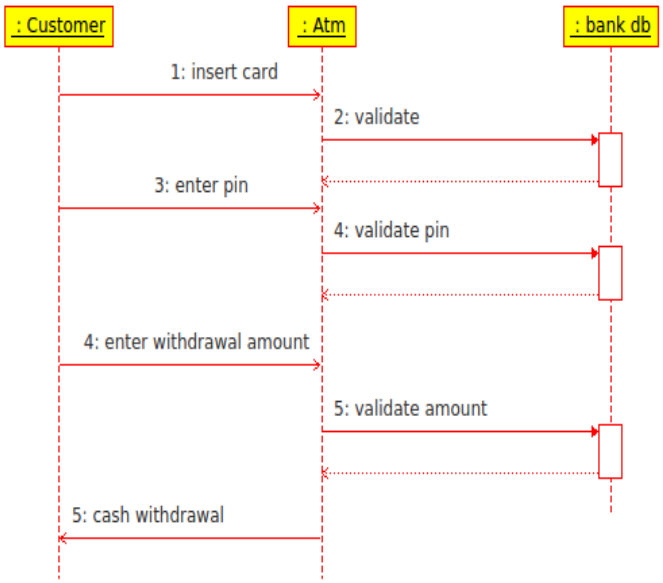


Fig: Sequence diagram for ATM

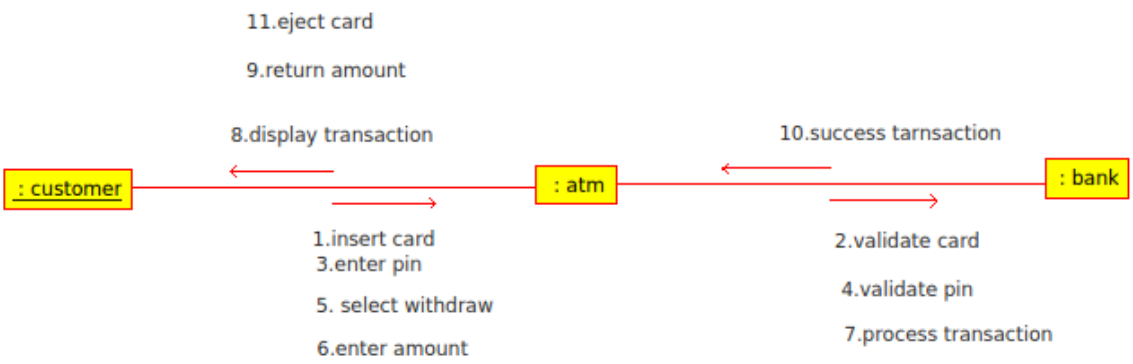


Fig: Collaboration diagram for ATM

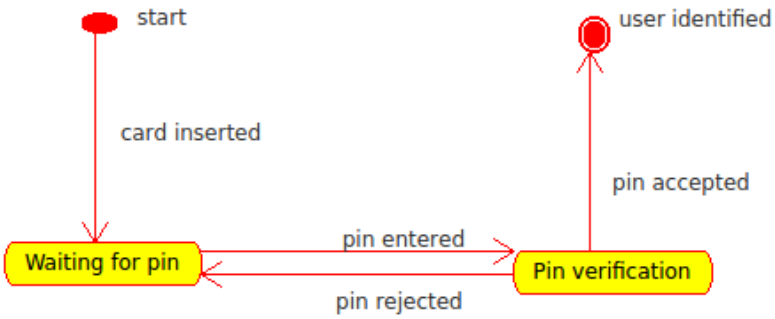


Fig: State chart diagram for ATM

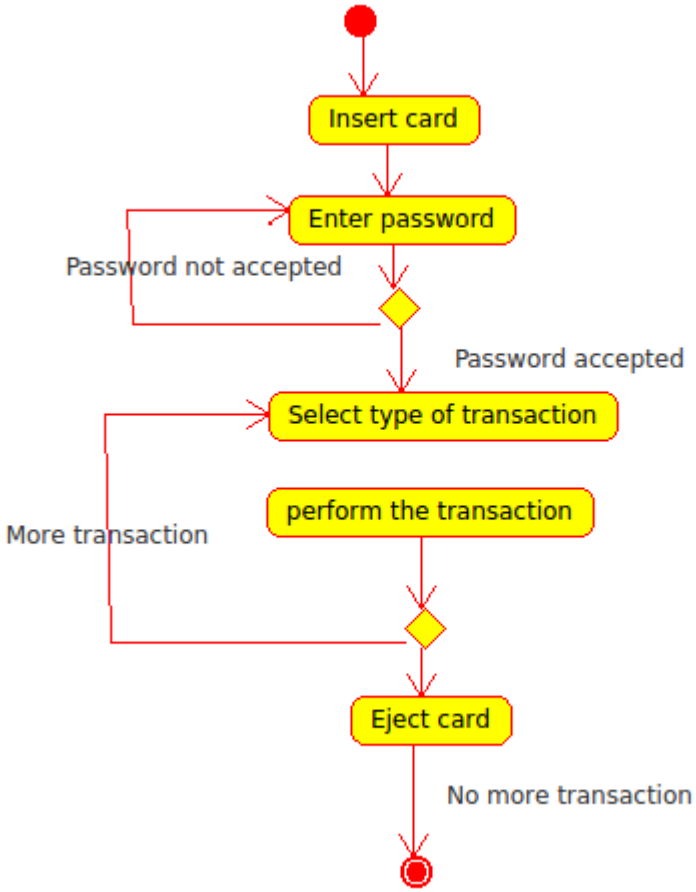


Fig: Activity diagram for ATM

Task 11: Implement a Junit Test program and design test cases to find the maximum of an array of numbers.

TEST LOGIC

```
packagecom.javatpoint.logic;
public class Calculation {

public static intfindMax(intarr[]){
int max=arr[0];//arr[0] instead of 0
for(inti=1;i<arr.length;i++){
if(max<arr[i])
max=arr[i];
      }
return max;
   }
}
```

TEST CASE

```
import static org.junit.Assert.*;
importorg.junit.Test;
public class TestLogic {
    @Test
public void testFindMax(intarr[])
{
assertEquals(4,Calculation.findMax(new int[]{1,3,4,2}));
assertEquals(-1,Calculation.findMax(new int[]{-12,-1,-3,-4,-2}));
   }
}
```

Task 12: Implement a Junit Test program and design test cases to count the number of elements in array of numbers.

TEST LOGIC

```
public class CountArray {
public static void count( ) {
    //Initialize array
int [] a = new int [] {1, 2, 3, 4, 5};
     //Number of elements present in an array can be found using the length
System.out.println("Number of elements present in given array: " +
arr.length);
   }
} }
```

TEST CASE

```
import static org.junit.Assert.*;
importorg.junit.Test;
public class TestLogic {
    @Test
public void testcount(){
assertEquals(4,Calculation.count(new int[]{1,3,4,2}));
assertEquals(2,Calculation.count(new int[]{2,3}));
assertEquals(0,Calculation.count(new int[]{}));

   }
}
```