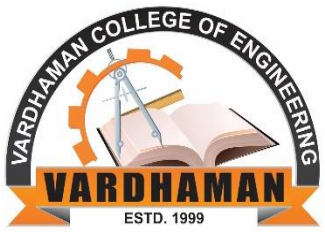
VARDHAMAN COLLEGE OF ENGINEERING, HYDERABAD
(Autonomous)
Affiliated to JNTUH, Hyderabad

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Laboratory Manual
of
SOFTWARE ENGINEERING
(III B.Tech.- I SEMESTER)
(VCE-R19)

<table>
<tr><td>Course Title</td><td colspan="3">SOFTWARE ENGINEERING</td><td colspan="2">Course Type</td><td colspan="2">Integrated</td></tr>
<tr><td>Course Code</td><td>A5605</td><td>Credits</td><td>4</td><td colspan="2">Class</td><td colspan="2">III Year I Semester</td></tr>
<tr><td rowspan="5">Course Structure</td><td>TLP</td><td>Credits</td><td>Contact Hours</td><td>Work Load</td><td colspan="2" rowspan="2">Total Number of Classes Per Semester</td><td colspan="2" rowspan="2">Assessment in Weightage</td></tr>
<tr><td>Theory</td><td>3</td><td>3</td><td>3</td></tr>
<tr><td>Practice</td><td>1</td><td>2</td><td>1</td><td>Theory</td><td>Practical</td><td>CIE</td><td>SEE</td></tr>
<tr><td>-</td><td></td><td></td><td></td><td colspan="4" rowspan="2"></td></tr>
<tr><td>Total</td><td>4</td><td>5</td><td>4</td></tr>
<tr><td rowspan="3">Course Instructors</td><td colspan="7">Course Lead: S Venu Gopal, Associate Professor, Dept. of CSE</td></tr>
<tr><td colspan="4">Theory</td><td colspan="3">Practice</td></tr>
<tr><td colspan="7">Ms N Deepika Rani, Asst. professor, Mr Kumar Baradur, Asst. professor</td></tr>
</table>

COURSE OVERVIEW

This course acts as a foundation in the field of software engineering and is aimed at helping students develop an understanding of how software systems are developed from scratch, by guiding them through the development process, adopting the fundamental principles of system development. The course will orient the students to the different software process models, software requirements engineering process, systems analysis and design as a problem-solving activity, with focus on quality.

Prerequisite(s):

- Programming for problem solving
- Object Oriented Programming

COURSE OBJECTIVE

Is to introduces software engineering to students as a discipline, discuss stages of the software life cycle, compare development models such as waterfall, prototyping and incremental/ iterative, agile process models. The course introduces Unified Modeling Language (UML) and quality management.

COURSE OUTCOMES (COs)

After the completion of the course, the student will be able to:

CO#	Course Outcomes	POs	PSOs
A5605.1	Illustrate the right process model to develop the right software system.	2, 3	1
A5605.2	Choose requirements and analyze them scientifically in order to develop the right product, besides authoring software requirements documents.	3,10	
A5605.3	Design as per functional and non-functional requirements using design principles.	2, 3	
A5605.4	Evaluate testing strategies for application being developed	4, 5	2
A5605.5	Classify the right set of umbrella activities for quality management and assurance.	9, 11	

BLOOM'S LEVEL OF THE COURSE OUTCOMES

CO#	Bloom's Level					
	Remember (L1)	Understand (L2)	Apply (L3)	Analyze (L4)	Evaluate (L5)	Create (L6)
A5605.1		✓				
A5605.2				✓		
A5605.3					✓	
A5605.4					✓	
A5605.5				✓		

COURSE ARTICULATION MATRIX

CO#/ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO 1	PSO 2
A5605.1		1	3											
A5605.2			3							2				
A5605.3		3	3											
A5605.4				1	2									
A5605.5									2		2			

Note: 1-Low, 2-Medium, 3-High

COURSE ASSESSMENT

Type of Course	Assessment									
	CIE Marks (30% Weightage)							SEE Marks (70% Weightage)		
	CAT1		CAT2		AAT		CIE Total	SEE		SEE Total
	T	P	T	P	T	P		T	P	
Integrated Course	30	10	30	10	15	5	100	75	25	100

* Assignment, Quiz, Class test, SWAYAM/NPTEL/MOOCs and etc..

LIST OF PROGRAMS FOR PRACTICE:

S No	Title of the Experiment	Tools and Techniques	Expected Skills/Ability
1.	Development of problem statement	Word Processor	Analyse requirements of client
2.	Preparation of Software Requirements specification Document, Design Documents and Test phase related documents		Prepare of design document

3.	Preparation of software Configuration Management and Risk Management related document		Evaluate risk in the project. Manage Versions of Software.
4	Design structural diagrams using CASE tools	IBM Rational rose	Able to design static aspect of the system
5	Design behavioral diagrams using CASE tools	IBM Rational rose	Able to design dynamic aspect of the system
6	Develop test cases for unit testing and integration testing	Word Processor	Manage the quality of the software project
7	Develop test cases for various system and regression testing techniques		

Sample projects for the above problems

1	Withdraw money from an Automatic Teller Machine (ATM)
2	Online bus Ticket reservation system
3	Exam registration
4	Library Management 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

LAB MANUAL CONTENTS

1. Development of problem statement.

1. Development of problem statement.

What are the key elements of a problem statement?

There are four key elements you should include when writing a problem statement:

1. **Ideal situation**
2. **Reality**
3. **Consequences**
4. **Proposal**

1. Ideal situation

The first thing your problem statement should describe is what the ideal situation would be if there wasn't a problem you needed to address. This section identifies the goals and scope of the project are. This section should create a clear understanding of what the ideal environment will be once the issue has been resolved.

2. Reality

The next section of your problem statement should describe what the current reality is for your company or organization. This section will identify what the problem is, state why it is a problem and identify who the problem is impacting. It will also describe when and where the problem was identified.

3. Consequences

The next section of your problem statement should identify what the consequences of the problem are. This section describes the effects of the problem by describing how the people affected by the problem are being impacted and quantifying how much the problem is impacting them. Common consequences can include the loss of time, money, resources, competitive advantage, productivity and more.

4. Proposal

The proposal section of a problem statement may contain several possible solutions to the problem, but it is important to remember that it does not need to identify a specific solution. The purpose of the proposal section should be to guide the project team on how they can research, investigate and resolve the problem.

1. Withdrawal money from an Automatic Teller Machine (ATM).

Automatic Teller Machine (ATM)

Problem Statement:

State Bank of India is planning to introduce Automated Teller Machine (ATM) system for its customers to provide convenient withdrawal service in everywhere, even if customer doesn't come to a bank. A software is to be designed that will control 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 (in multiples of Rs.100) and a printer for printing customer receipts. The bank providescomputer to maintain its own accounts and process transactions against them.

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.

The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. 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 also maintain an internal log of transactions to facilitate resolving ambiguities arising from a hardware failure in the middle of a transaction. Entries will be made in the log when the ATM is started up and shut down, for each message sent to the Bank (along with the response back, if one is expected), for the dispensing of cash, and for the receiving of an envelope. Log entries may contain card numbers and amounts, but for security will never contain a PIN.

To avail ATM facility, a customer is required to open/have an account in the bank and apply for the ATM card. A customer can have one or more accounts and for each account, only one ATM card will be provided. The bank also provides SMS updates for every transaction of customer's account. To obtain SMS updates, customer is required to register his/her mobile number against his account in the bank.

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

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

Aim: Understanding of SRS.

Requirements:
Hardware Requirements:
- PC with 300 megahertz or higher processor clock speed recommended; 233 MHz minimum required.
- 128 megabytes (MB) of RAM or higher recommended (64 MB minimum supported)
- 1.5 gigabytes (GB) of available hard disk space
- CD ROM or DVD Drive
- Keyboard and Mouse (compatible pointing device).

Software Requirements:
Rational Rose, Windows XP.

Theory:

An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time(usually) prior to any actual design or development work. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.

The SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an eCommerce Web site, and so on) must provide, as well as states any required constraints by which the system must abide. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans,and documentation plans, are related to it.

It's important to note that an SRS contains functional and nonfunctional requirements only; it doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements to be.

A well-designed, well-written SRS accomplishes four major goals:

- It provides feedback to the customer. An SRS is the customer's assurance that the development organization understands the issues or problems to be solved and the software behavior necessary to address those problems. Therefore, the SRS should be written in natural language (versus a formal language, explained later in this article), in an unambiguous manner that may also include charts, tables, data flow diagrams, decision tables, and so on.
- It decomposes the problem into component parts. The simple act of writing down software requirements in a well-designed format organizes information, places borders around the problem, solidifies ideas, and helps break down the problem into its component parts in an orderly fashion.

- It serves as an input to the design specification. As mentioned previously, the SRS serves as the parent document to subsequent documents, such as the software design specification and statement of work. Therefore, the SRS must contain sufficient detail in the functional system requirements so that a design solution can be devised.
- It serves as a product validation check. The SRS also serves as the parent document for testing and validation strategies that will be applied to the requirements for verification.

SRSs are typically developed during the first stages of "Requirements Development," which is the initial product development phase in which information is gathered about what requirements are needed--and not. This information-gathering stage can include onsite visits, questionnaires, surveys, interviews, and perhaps a return-on-investment (ROI) analysis or needs analysis of the customer or client's current business environment. The actual specification, then, is written after the requirements have been gathered and analyzed.

SRS should address the following:
The basic issues that the SRS shall address are the following:
a) Functionality: What is the software supposed to do?
b) External interfaces: How does the software interact with people, the system‘s hardware, other hardware, and other software?
c) Performance: What is the speed, availability, response time, recovery time of various software functions, etc.?
d)Attributes: What are the portability, correctness, maintainability, security, etc. considerations?
e) Design constraints: imposed on an implementation. Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?
Chracteristics of a good SRS
An SRS should be

a) Correct
b) Unambiguous
c) Complete
d) Consistent
e) Ranked for importance and/or stability
f) Verifiable
g) Modifiable
h) Traceable

Correct -This is like motherhood and apple pie. Of course you want the specification to be correct. No one writes a specification that they know is incorrect. We like to say -"Correct and Ever Correcting." The discipline is keeping the specification up to date when you find things that are not correct.
Unambiguous -An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. Again, easier said than done. Spending time on this area prior to releasing the SRS can be a waste of time. But as you find ambiguities -fix them.
Complete -A simple judge of this is that is should be all that is needed by the software designers to create the software.

Consistent -The SRS should be consistent within itself and consistent to its reference documents. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another.

Ranked forImportance -Very often a new system has requirements that are really marketing wish lists. Some may not be achievable. It is useful provide this information in the SRS.

Verifiable -Don't put in requirements like -"It should provide the user a fast response." Another of my favorites is -"The system should never crash." Instead, provide a quantitative requirement like: "Every key stroke should provide a user response within 100 milliseconds."

Modifiable -Having the same requirement in more than one place may not be wrong -but tends to make the document not maintainable.

Traceable -Often, this is not important in a non-politicized environment. However, in most organizations, it is sometimes useful to connect the requirements in the SRS to a higher level document. Why do we need this requirement?

A sample of basic SRS Outline

1. Introduction
- 1.1 Purpose
- 1.2 Document conventions
- 1.3 Intended audience
- 1.4 Additional information
- 1.5 Contact information/SRS team members
- 1.6 References

2. Overall Description
- 2.1 Product perspective
- 2.2 Product functions
- 2.3 User classes and characteristics
- 2.4 Operating environment
- 2.5 User environment
- 2.6 Design/implementation constraints
- 2.7 Assumptions and dependencies

3. External Interface Requirements
- 3.1 User interfaces
- 3.2 Hardware interfaces
- 3.3 Software interfaces
- 3.4 Communication protocols and interfaces

4. System Features
- 4.1 System feature A
  - 4.1.1 Description and priority
  - 4.1.2 Action/result
  - 4.1.3 Functional requirements
- 4.2 System feature B

5. Other Nonfunctional Requirements
- 5.1 Performance requirements
- 5.2 Safety requirements
- 5.3 Security requirements
- 5.4 Software quality attributes
- 5.5 Project documentation

5.6 User documentation

6. Other Requirements

Appendix A: Terminology/Glossary/Definitions list

Appendix B: To be determined

Conclusion: The SRS was made successfully by following the steps described above.

SAMPLE SRS
S O F T WA R E R E Q U I R E M E N TS S P E C I F I C AT I O N
ATM
Version 1.0
AN AUTOMATED TELLER MACHINE

Table of Contents

1.Introduction

The software ATMExcl 3.0TMversion1.0 is to be developed for Automated Teller Machines (ATM). An automated teller machine (ATM) is computerized telecommunications device that provides a financial institution's customers a secure method of performing financial transactions, in a public space without the need for a human bank teller. Through ATMExcl 3.0TM,customers interact with a user-friendly interface that enables them to access their bank accounts and perform various transactions.

1.1 Purpose

This SRS defines External Interface, Performance and Software System Attributes requirements of ATMExcl 3.0TM. This document is intended for the following group of people:-

- ✓ Developers for the purpose of maintenance and new releases of the software.
- ✓ Management of the bank.
- ✓ Documentation writers.
- ✓ Testers.

1.2 Scope

This document applies to Automated Teller Machine software ATM 3.0TM. This software facilitates the user to perform various transaction in his account without going to bank. This software offers benefits such cash withdrawals, balance transfers, deposits, inquiries, credit card advances and other banking related operations for customers. It also allows the administrator to fix the tariffs and rules as and when required.

The software takes as input the login Id and tha bank account number of the user for login purposes. The outputs then comprise of an interactive display that lets the user select the desirable function that he wants to perform..

The software is expected to complete in duration of six months and the estimated cost is Rs18 lakhs.

1.3 Definitions, Acronyms, and Abbreviations

AC	Alternate Current
AIMS.	ATM Information Management System
ATM	An unattended electronic machine in a public place, connected to a data system and related equipment and activated by a bank customer to obtain cash withdrawals and other banking services.
Braille	A system of writing and printing for blind or visually impaired people, in which varied arrangements of raised dots representing letters and numerals are identified bytouch
BMS	Bank Management Software developed by KPM Bank.
CDMA	Code Division Multiple Access, a reliable data communication protocol
CMS	Card Management Software developed by KPM Bank.
DES	Data Encryption Standard.
Dial-Up POS	A message format for low cost communications.
Electronic Journals	For easier, safer information storage, related to modem.
Internet	An interconnected system of networks that connects computers around the world via the TCP/IP protocol.
MB	Mega Bytes
Ms	Milliseconds.

Sec	Seconds
Smart Card	Card without hardware which stores the user‘s private keys within a tamper proof software guard
SRS	Software Requirements Specification
Tactile Keyboard	Special keyboard designed to aid the visually impaired.
TCP/IP	Transmission Control Protocol/Internet Protocol.
V	Volts
VGA	Video Graphics Adaptor is a display standard.

1.4 References

The references for the above software are as follows:-

i.www.google.co.in
ii.www.wikipedia.com
iii.IEEE. Software Requirements Specification Std. 830-1993.
iv.Chevy Chase Bank, UMBC Branch.
v.Russell C. Bjork Requirements Statement for Example ATM System. Online. URL: http://www.math-cs.gordon.edu/local/courses/cs211/ATMExample/

1.5 Overview

Section 1.0 discusses the purpose and scope of the software.

Section 2.0 describes the overall functionalities and constraints of the software and user characteristics.

Section 3.0 details all the requirements needed to design the software.

2. The Overall Description

2.1 Product Perspective

The ATM is a single functional unit consisting of various sub-components.

- ✓ This software allows the user to access their bank accounts remotely through an ATM without any aid of human bank teller.
- ✓ This software also allows the perform various other functions apart from just accessing his bank account such as mobile bill clearings etc.
- ✓ Some of its hardware components are cassettes, memory, drives, dispensers i.e. for receipts and cash, a card reader, printer, switches, a console, a telephone dialer port, a networking port and disks.
- ✓ The ATM communicates with the bank‘s central server through a dial-up communication link. The Memory of the system shall be 20MB.
- ✓ The Cassette capacity shall be at least 2000 notes.

2.2 Product Functions

The major functions that ATMExcl 3.0TM performs are described as follows:-

Language Selection:-After the user has logged in, the display provides him with a list of languages from which he can select any one in order to interact with the machine throughout that session. After the language selection the user is prompted with an option that whether he wants the selected language to be fixed for future use so that he is not offered with the language selection menu in future thus making the transaction a bit faster. User also has the freedom to switch to a different language mentioned in the list in between that session.

Account Maintenance:- The various functions that a user can perform with his account are as follows:-

♣**Account Type:-**The user has the freedom to select his account type to which all the transactions are made, i.e. he can select whether the account is current account or savings account etc.

♣**Withdrawal/Deposit:** The software allows the user to select the kind of operation to be performed i.e. whether he wants to withdraw or deposit the money.

♣**Amount:-**The amount to be withdrawn or deposited is then mentioned by the user.

♣**Denominations:-**The user is also provided with the facility to mention the required denominations. Once he enters his requirements the machine goes through its calculations on the basis of current resources to check whether it is possible or not. If yes, the amount is given to the user otherwise other possible alternatives are displayed.

♣**Money Deposition:-**Money deposition shall be done with an envelope. After typing the amount to be deposited and verification of the same, the customer must insert the envelope in the depositary.

♣**BalanceTransfer:-**Balance transfer shall be facilitated between any two accounts linked to the card for example saving and checking account.

♣**Balance Enquiry:-**Balance enquiry for any account linked to the card shall be facilitated.

Billing:-Any transaction shall be recorded in the form of a receipt and the same would be dispensed to the customer. The billing procedures are handled by the billing module that enable user to choose whether he wants the printed statement of the transaction or just the updating in his account

Cancelling:-The customer shall abort a transaction with the press of a Cancel key.For example on entering a wrong depositing amount. In addition the user can also cancel the entire session by pressing the abort key and can start a fresh session all over again.

Map locating other machines:-The machine also has a facility of displaying the map that marks the locations of other ATM machines of the same bank in the entire city.

Mobile Bills Clearings:-The machine also allows the user to clear off his pending mobile bills there only, if the name of his operator is mentioned there in the list. The machine displays the list of the companies supported by that bank to the user.

2.3 User Characteristics

There are different kind of users that will be interacting with the system. The intended user of the software are as follows:-

User A: A novice ATM customer. This user has little or no experience with electronic means of account management and is not a frequent user of the product. User A will find the product easy to use due to simple explanatory screens for each ATM function. He is also assisted by an intarctive teaching mechanism at every step of the transaction, both with the help of visual and audio help sessions.

User B: An experienced customer. This user has used an ATM on several occasions before and does most of his account management through the ATM. There is only a little help session that too at the beginning of the session thus making the transaction procedure more faster. Maintenance Personnel: A bank employee. This user is familiar with the functioning of the ATM. This user is in charge of storing cash into the ATM vault and repairing the ATM in case of malfunction. This user is presented with a different display when he logs in with the admninistrator‘s password and is provided with options different from that of normal

user. He has the authority to change or restrict various features provided by the software in situations of repairing.

2.4 Constraints

The major constraints that the project has are as follows:-

- ✓ The ATM must service at most one person at a time.
- ✓ The number of invalid pin entries attempted must not exceed three. After three unsuccessful login attempts, the card is seized/blocked and need to be unlocked by the bank.
- ✓ The simultaneous access to an account through both, the ATM and the bank is not supported.
- ✓ The minimum amount of money a user can withdraw is Rs 100/-and the maximum amount of money a user can withdraw in a session is Rs.10,000/-and the maximum amount he can withdraw in a day is Rs 20,000/-.
- ✓ Before the transaction is carried out, a check is performed by the machine to ensure that a minimum amount of Rs 1000/-is left in the user‘s account after the withdrawal failing which the withdrawal is denied.
- ✓ The minimum amount a user can deposit is Rs 100/-and the maximum amount he can deposit is Rs 10,000/-.
- ✓ A user can select only that cellular operator for mobile bill clearings that is supported by the bank.
- ✓ The software requires a minimum memory of 20GB⏐The database used should be Oracle7.0.
- ✓ There shall be a printer installed with the machine to provide the user with the printed statement of the transaction.
- ✓ For voice interactions, speakers should also be there to accompany the machine.

2.5 Assumptions and Dependencies

The requirements stated in the SRS could be affected by the following factors:

•One major dependency that the project might face is the changes that need to be incorporated with the changes in the bank policies regarding different services. As the policies changes the system needs to be updated with the same immediately. A delay in doing the same will result to tremendous loss to the bank. So this should be changed as and when required by the developer.

•Another constraint relating to the operating environment is that we are specific to Oracle Database.

•The project could be largely affected if some amount is withdrawn from the user‘s account from the bank at the same time when someone is accessing that account through the ATM machine. Such a condition shall be taken care of.

•At this stage no quantitive measures are imposed on the software in terms of speed and memory although it is implied that all functions will be optimized with respect tospeed and memory.

It is furthermore assumed that the scope of the package will increase considerably in the future.

3. External Interface Requirements

3.1.1User Interface Requirements

The interface provided to the user should be a very user-friendly one and it should provide an optional interactive help for each of the service listed. The interface provided is a menu driven one and the following screens will be provided:-

1. A login screen is provided in the beginning for entering the required username/pin no. and account number.

2. An unsuccessful login leads to a reattempt (maximum three) screen for again entering the same information. The successful login leads to a screen displaying a list of supported languages from which a user can select any one.

3.In case of administrator, a screen will be shown having options to reboot system, shut down system, block system, disable any service.

4. In case of reboot/ shut down, a screen is displayed to confirm the user‘s will to reboot and also allow the user to take any backup if needed.

5. In case of blocking system, a screen is provided asking for the card no. By entering the card number of a particular user, system access can be blocked for him.

6. Administrator is also provided with a screen that enables him to block any service provided to the user by enter in the name of the service or by selecting it from the list displayed.

7.After the login, a screen with a number of options is then shown to the user. It contains all the options along with their brief description to enable the user to understand their functioning and select the proper option.

8. A screen will be provided for user to check his account balance.

9. A screen will be provided that displays the location of all other ATMs of same bank elsewhere in the city.

10.A screen will be provided for the user to perform various transactions in his account.

The following reports will be generated after each session dealt with in the machine:-

1.The login time and logout time along with the user‘s pin no and account number is registered in the bank‘s database.

2.The ATM‘s branch ID through which the session is established is also noted down in the bank‘s database.

3.Various changes in the user‘s account after the transactions,if any, are reported in the database.

4.A printed statement is generated for the user displaying all the transactions he performed.

Other various user interface requirements that need to be fulfilled are as follows:-

- ✓ The display screen shall be of 10" VGA color type.
- ✓ The display screen shall have 256 color resolution.
- ✓ The display screen shall also support touchscreen facility.
- ✓ The speakers shall support Yamaha codecs.
- ✓ The keypad shall consist of 16 tactile keys.
- ✓ There shall be 8 tactile function keys.
- ✓ The keyboard will be weather resistant.
- ✓ The transaction receipt shall be 3.1" × 6".
- ✓ The statement receipt shall be 4.2" × 12".

- ✓ The deposit envelopes shall be 9" long and 4" wide.

3.1.2Hardware Interface Requirements

There are various hardware components with which the machine is required to interact. Various hardware interface requirements that need to be fulfilled for successful functioning of the software are as follows:-θ

The ATM power supply shall have a 10/220 V AC manual switch.

The ATM card should have the following physical dimensions:-

Width	-85.47mm-85.72mmo
Height	-53.92mm-54.03mmo
Thickness	-0.76mm+0.08mm

The card reader shall be a magnetic stripe reader

The card reader shall have Smart card option.

The slot for a card in thye card reader may include an extra indentation for the embossed area of the card. In effect it acts as a polarization key and may be used to aid the correct insertion orientation of the card. This is an additional characteristic to the magnetic field sensor which operates off the magnetic stripe and is used to open a mechanical gate on devices such as ATMs.

There shall be a 40 column dot matrix receipt printer.

There shall be a 40 column dot matrix statement printer.

The receipt dispenser shall be a maximum of 4" width and 0.5" thickness.

The statement dispenser shall be a maximum of 5" width and 0.5" thickness.

The envelope depository shall be a maximum of 4.5" width, 10" length and 0.5" thickness.

Screen resolution of at least 800X600-required for proper and complete viewing of screens. Higher resolution would not be a problem.

3.1.3 Software Interface Requirements

In order to perform various different functions, this software needs to interact with various other software's. So there are certain software interface requirements that need to be fulfilled which are listed as follows:-

- ✓ The transaction management software used to manage the transaction and keep track of resources shall be BMS version 2.0.
- ✓ The card management software used to verify pin no and login shall be CMS version 3.0.
- ✓ Yamaha codec 367/98 for active speakers.
- ✓ The database used to keep record of user accounts shall be Oracle version7.0.

3.1.4 Communication Interface Requirements

The machine needs to communicate with the main branch for each session for various functions such as login verification, account access etc. so the following are the various communication interface requirements that are needed to be fulfilled in order to run the software successfully:-

- The system will employ dial-up POS with the central server for low cost communication.
- The communication protocol used shall be TCP/IP.
- Protocol used for data transfer shall be File Transfer Protocol.(FTP)

4. System Features

1. Remote Banking and Account Management

Description

The system is designed to provide the user with the facility of remote banking and perform various other functions at an interface without any aid of human bank teller. The functioning of the system shall be as follows:-

At the start, the user is provided with a log in screen and he is required to enter his PIN NO. and Account details which are then verified by the machine. In case of an unsuccessful attempt a user is asked again for his credentials but the maximum number of attempt given to the user is limited to 3 only, failing which his card is blocked and need to be unblocked by the bank for any future use.

After a successful log in, the user is presented with a list of language. The user can select any one in the list for interaction with the machine for the entire session.

After the language selection the user is also asked whether he wants to fix that language for future use also so that he is never asked for language in future. In addition there is also a facility for the user to switch to any other language during that session.

After the language selection, the user is directed towards a main page that displays a set of options/services along with their brief description, enabling the user to understand their functioning. The user can select any of the listed option and can continue with the transaction.

The machine also provides the user with a number of miscellaneous services such as:

The machine lists a set of operators that are supported by the bank. A user can clear off his pending mobile phone bills be selecting his operator.The machine also has the facility to display a map that marks the location of other ATMs of the same bank in the city. This may help the user to look for the ATM nearest to his destination.

At any moment if the user wants to abort the transaction, he is provided with an option to cancel it. Just by pressing the abort button he can cancel all the changes made so far and can begin with a new transaction.

After the user is finished with his work, for security purpose, he is required to log out and then take his card out of the slot.

Validity Checks

In order to gain access to the system, the user is required to enter his/her correct user id/pin no and account no failing which his card may be blocked.

The user can access only one account at a time and can enter only one account no.

Also if the user is an administrator, he is required to enter his login id in order to access and change the facilities provided by the system.

Sequencing Information

The information about the users and their account should be entered into the database prior to any of the transactions and the backup be maintained for all account information

Error Handling/ Response to Abnormal Situations

If any of the above validation/sequencing flow does not hold true, appropriate error messages will be prompted to the user for doing the needful.

2. Receipt Generation

After each transaction user has performed, a receipt isgenerated that contains all the information about the transaction. The format of the generated receipt is as shown below:-

KPM BANK

Branch name/Id

(address)

Login Time:-

Date:-
Account No:-
User Name:-
TRANSACTIONS:

FROM	TO	TYPE	AMOUNT

Logout Time:- BARCODE
Thank You For your visit.
See you soon.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The following list provides a brief summary of the performance requirements for the software:

5.1.1 Capacity

The ATM shall provide customers a 24 hour service.

5.1.2 Dynamic requirements

- ✓ The card verification time must not exceed 0.8 sec. under normal server workload and 1 sec. under peak server workload.
- ✓ The pin number verification time must not exceed 0.3 sec. under normal server workload and 0.5 sec. under peak server workload.
- ✓ Account balance display time must not exceed 2 sec. under normal server workload and 3 sec. under peak server workload.
- ✓ Account balance transfer time must not exceed 3 sec. under normal server workload and 4 sec. under peak server workload.
- ✓ Cash withdrawal transaction time must not exceed 4 sec. under normal server workload and 5 sec. under peak server workload.
- ✓ Deposit transaction time after insertion of the deposit envelope must not exceed 5 sec. under normal server workload and 6 sec. under peak server workload.
- ✓ Receipt printing time after must not exceed 3 sec. under normal server and peak server workload.
- ✓ Touch screen and button response time must not exceed 5000ms.
- ✓ Credit card advance time must not exceed 6 sec. under normal traffic and server and peak traffic and server workload.

5.1.3 Quality–The primary objective is to produce quality software. As the quality of a piece of software is difficult to measure quantitatively, the following guidelines will be used when judging the quality of the software:

1.Consistency –All code will be consistent with respect to the style. (This is implied when adhering to the standard).

2.Test cases –All functionality will be thoroughly tested

5.2 Software System Attributes

5.2.1 Reliability

The data communication protocol shall be such that it ensures reliability and quality of data and voice transmission in a mobile environment.For example, CDMA.

The memory system shall be of non-volatile type

5.2.2 Availability

The product will have a backup power supply in case of power failures.
Any abnormal operations shall result in the shutting down of the system.
After abnormal shutdown of the ATM, the system shall have to be manually restarted by a maintenance personnel.
There should be no inconsistency introduced in the account during whose transaction the system is abnormally shut down.

5.2.3Security

- The system shall be compatible with AIMS security standards.
- The system shall have two levels of security i.e. ATM card and pin verification both authenticated by the CMS software.
- The Encryption standard used during pin transmission shall be triple DES.
- The password shall be 6-14 characters long.
- Passwords shall not contain name of customers as they are easy to be hacked.

✓ Passwords can contain digit, hyphen and underscore.

- User should be provided with only three attempts for login failing which his card needs to be blocked.
- There shall be a security camera installed near the ATM. θThere shall be a secured cash vault with a combination locking system.
- The product cabinet cover shall be manufactured using Fiber glass for security purposes.

5.2.4Maintainability

The system components i.e. modem, memory, disk, drives shall be easily serviceable without requiring access to the vault.
The system should have the mechanism of self-monitoring periodically in order to detect any fault.
The system should inform the main branch automatically as soon as it detects any error. The kind of fault and the problem being encountered should also be mentioned by the system automatically.

5.3 Business Rules

The business rules for the software are as follows:

•The Administrator has the authority to fix the rules and regulations and to set or update the policies as and when required.

•The staff at the bank performs the following:

a. Making the entries in the system regarding all the details of the bank account of the user.

b.Keeping the bank account of the user updated as soon as changes are encountered so that the data is in consistent state.

c.Blocking or seizing of the account of user on discovery of any illegal transaction.

d.Unblocking of ATM card that got blocked due to more than three unsuccessful login attempt.

e.Blocking of a lost/stolen ATM card on complaint of the user, only if he presents his verification and a FIR filed for that case.

f.Constantly monitor all the ATMs in the city to check whether any one of them is encountering any fault.

g.Immediately correct any fault discovered in any of the ATM.

h.Maintain the backup of all the accounts for reliability purposes.

i.Rollback all the changes made in an account during whose transaction an ATM got abnormal shutdown.

- In case of loss of the ATM card. The user has to lodge a First Investigation Report(FIR) and present its one copy to bank officials for card blocking purposes.
- A log of the following annexure is generated by the system:
- User bank account details.
- Updations made in the user account along with date, time and the changes made.
- Schedule of fixed assets.

6 Other Requirements

None.

Appendix A: Glossary

AIMS -	ATM Information Management System.
ATM -	An unattended electronic machine in a public place, connected to a data system and related equipment and activated by a bank customer to obtain cash withdrawals and other banking services.
Braille-	A system of writing and printing for blind or visually impaired people, in which varied arrangements of raised dots representing letters and numerals are identified by touch.
CDMA -	Code Division Multiple Access, a reliable data communication protocol.
CMS -	Card Management Software developed by KPM Bank.
Dial-Up -	A message format for low cost communications.
POSInternet -	An interconnected system of networks that connects computers around the world via the TCP/IP protocol.
Smart Card -	Card without hardware which stores the user‘s private keys within a tamper proof software guard.
Tactile Keyboard -	Special keyboard designed to aid the visually impaired.
TCP/IP -	Transmission Control Protocol/Internet Protocol.

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

Preparation of Software Configuration Management and risk Management related documents.

The purpose of software configuration management is to plan, organize, control and coordinate the identification, storage and change of software through development, integration and transfer (SCM02). Every project must establish a software configuration management system. All softwareitems,forexampledocumentation,sourcecode,executablecode, files, tools, test software and data, must be subjected to SCM (SCM01). Software configuration management must ensurethat:

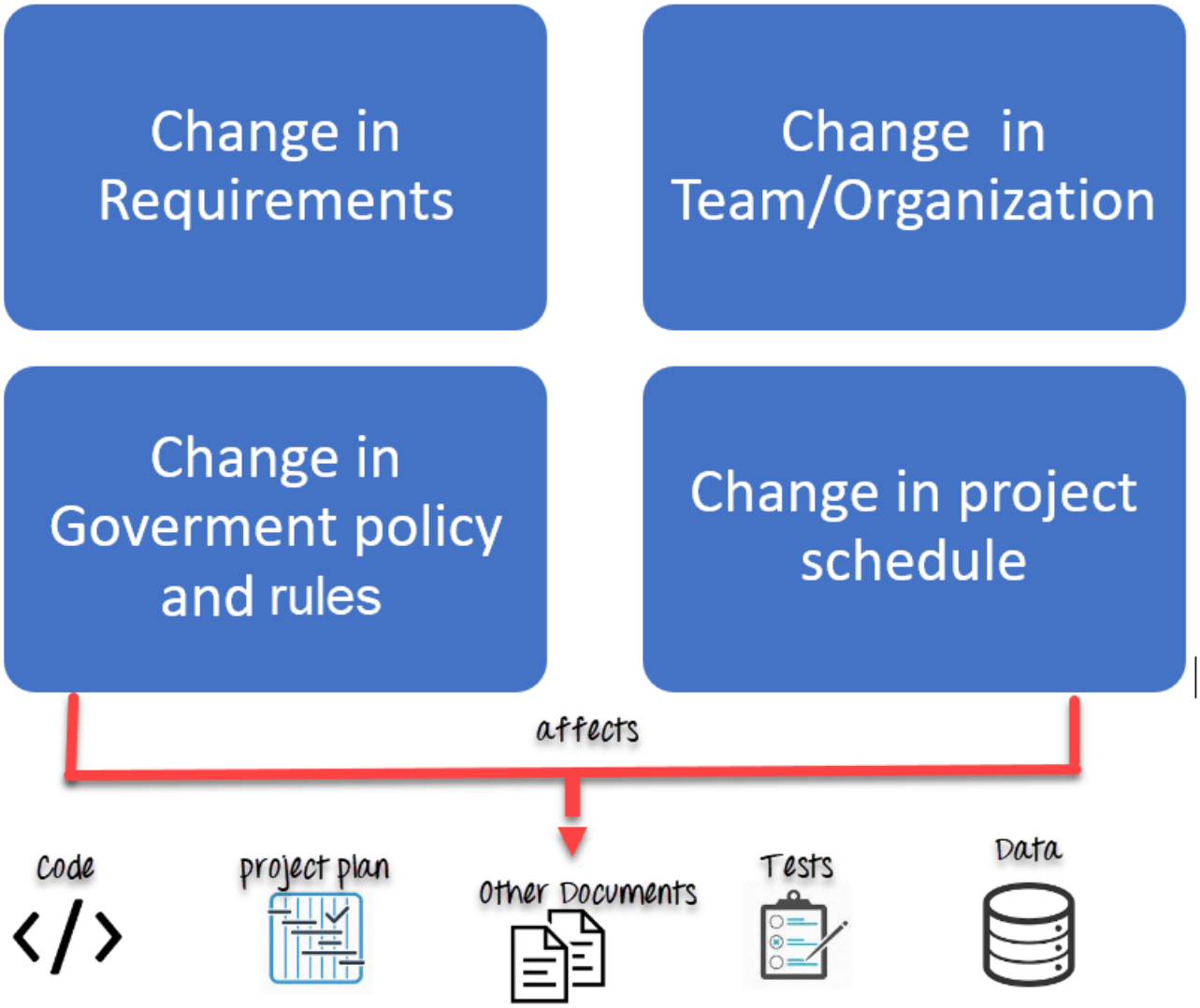
1. software components can beidentified;
2. software is built from a consistentset of components;
3. software components are available andaccessible;
4. softwarecomponentsnevergetlost(e.g.aftermediafailureoroperator error);
5. everychangetothesoftwareisapprovedanddocumented;
6. changesdonotgetlost(e.g.throughsimultaneousupdates);
7. itisalwayspossibletogobacktoapreviousversion;
8. ahistoryofchangesiskept,sothatisalwayspossibletodiscoverwho did what andwhen.

All configuration items, from modules to entire software releases, must be defined as early as possible and systematically labeled when they are created. Software development can only proceed smoothly if the development team are able to work on correct and consistent configuration items. Responsibilities for the creation and modification of configuration items must be allocated to individuals and respected by others. Although this allocation is defined in the Work Breakdown Structure (WBS) of the Software Project Management Plan (SPMP), the Software Configuration Management Plan (SCMP) defines the procedures that coordinate the efforts of the developers.

Why do we need Configuration management?

The primary reasons for Implementing Software Configuration Management System are**:**

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



Any change in the software configuration Items will affect the final product. Therefore, changes to configuration items need to be controlled and managed.

Tasks in SCM process

a. Configuration Identification
b. Baselines
c. Change Control
d. Configuration Status Accounting
e. Configuration Audits and Reviews

Configuration Identification:

Configuration identification is a method of determining the scope of the software system. With the help of this step, you can manage or control something even if you don't know what it is. It is a description that contains the CSCI type (Computer Software Configuration Item), a project identifier and version information.

Activities during this process:

- Identification of configuration Items like source code modules, test case, and requirements specification.
- Identification of each CSCI in the SCM repository, by using an object-oriented approach
- The process starts with basic objects which are grouped into aggregate objects. Details of what, why, when and by whom changes in the test are made
- Every object has its own features that identify its name that is explicit to all other objects
- List of resources required such as the document, the file, tools, etc.

Example:

Instead of naming a File login.phpits should be named login_v1.2.php where v1.2 stands for the version number of the file

Instead of naming folder "Code" it should be named "Code_D" where D represents code should be backed up daily.

Baseline:

A baseline is a formally accepted version of a software configuration item. It is designated and fixed at a specific time while conducting the SCM process. It can only be changed through formal change control procedures.

Activities during this process:

- Facilitate construction of various versions of an application
- Defining and determining mechanisms for managing various versions of these work products
- The functional baseline corresponds to the reviewed system requirements
- Widely used baselines include functional, developmental, and product baselines

In simple words, baseline means ready for release.

Change Control:

Change control is a procedural method which ensures quality and consistency when changes are made in the configuration object. In this step, the change request is submitted to software configuration manager.

Activities during this process:

- Control ad-hoc change to build stable software development environment. Changes are committed to the repository
- The request will be checked based on the technical merit, possible side effects and overall impact on other configuration objects.

- It manages changes and making configuration items available during the software lifecycle

Configuration Status Accounting:

Configuration status accounting tracks each release during the SCM process. This stage involves tracking what each version has and the changes that lead to this version.

Activities during this process:

- Keeps a record of all the changes made to the previous baseline to reach a new baseline
- Identify all items to define the software configuration
- Monitor status of change requests
- Complete listing of all changes since the last baseline
- Allows tracking of progress to next baseline
- Allows to check previous releases/versions to be extracted for testing

Configuration Audits and Reviews:

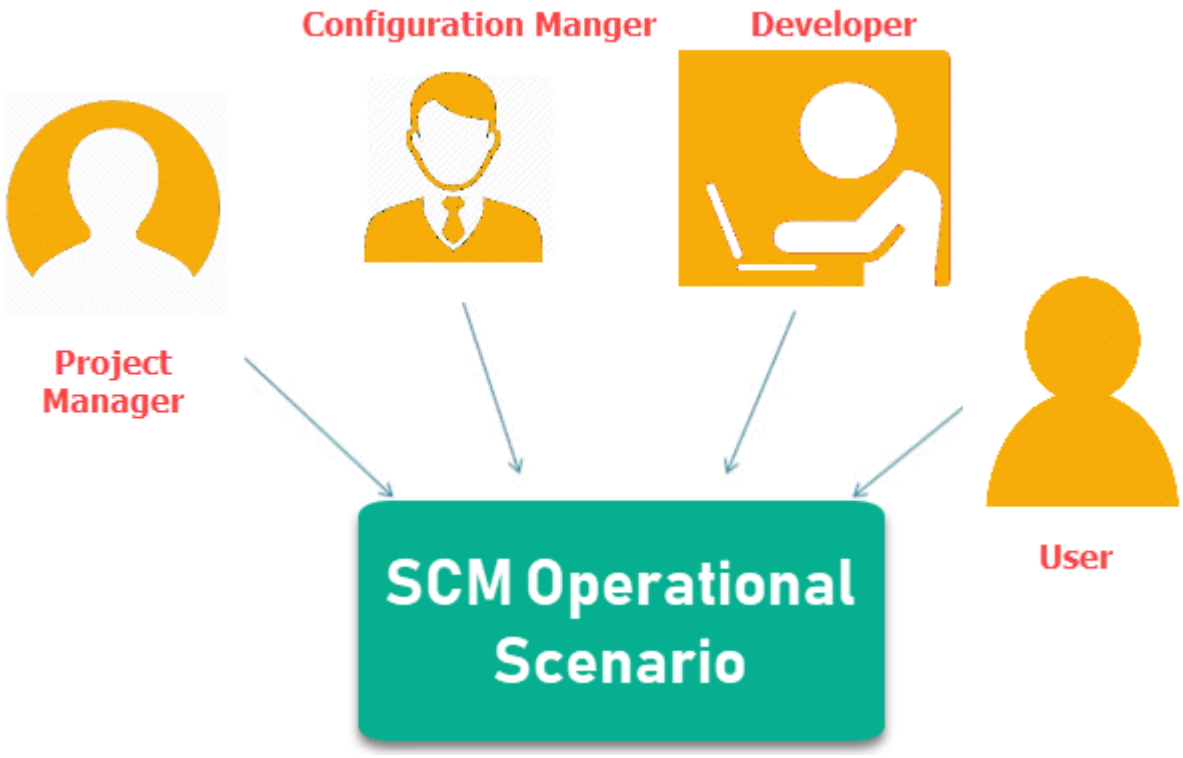
Software Configuration audits verify that all the software product satisfies the baseline needs. It ensures that what is built is what is delivered.

Activities during this process:

- Configuration auditing is conducted by auditors by checking that defined processes are being followed and ensuring that the SCM goals are satisfied.
- To verify compliance with configuration control standards. auditing and reporting the changes made
- SCM audits also ensure that traceability is maintained during the process.
- Ensures that changes made to a baseline comply with the configuration status reports
- Validation of completeness and consistency

Participant of SCM process:

Following are the key participants in SCM



1. Configuration Manager

- Configuration Manager is the head who is Responsible for identifying configuration items.
- CM ensures team follows the SCM process
- She/he needs to approve or reject change requests

2. Developer

- The developer needs to change the code as per standard development activities or change requests. He is responsible for maintaining configuration of code.
- The developer should check the changes and resolves conflicts

3. Auditor

- The auditor is responsible for SCM audits and reviews.
- Need to ensure the consistency and completeness of release.

4. Project Manager:

- Ensure that the product is developed within a certain time frame
- Monitors the progress of development and recognizes issues in the SCM process
- Generate reports about the status of the software system
- Make sure that processes and policies are followed for creating, changing, and testing

5. User

The end user should understand the key SCM terms to ensure he has the latest version of the software

Software Configuration Management Plan

The SCMP (Software Configuration management planning) process planning begins at the early phases of a project. The outcome of the planning phase is the SCM plan which might be stretched or revised during the project.

- The SCMP can follow a public standard like the IEEE 828 or organization specific standard
- It defines the types of documents to be management and a document naming. Example Test_v1
- SCMP defines the person who will be responsible for the entire SCM process and creation of baselines.
- Fix policies for version management & change control
- Define tools which can be used during the SCM process
- Configuration management database for recording configuration information.

Software Configuration Management Tools

Any Change management software should have the following 3 Key features:

Concurrency Management:

When two or more tasks are happening at the same time, it is known as concurrent operation. Concurrency in context to SCM means that the same file being edited by multiple persons at the same time.

If concurrency is not managed correctly with SCM tools, then it may create many pressing issues.

Version Control:

SCM uses archiving method or saves every change made to file. With the help of archiving or save feature, it is possible to roll back to the previous version in case of issues.

Synchronization:

Users can checkout more than one files or an entire copy of the repository. The user then works on the needed file and checks in the changes back to the repository.They can synchronize their local copy to stay updated with the changes made by other team members.

Following are popular tools

1. Git: Git is a free and open source tool which helps version control. It is designed to handle all types of projects with speed and efficiency.

Download link:https://git-scm.com/

2. Team Foundation Server: Team Foundation is a group of tools and technologies that enable the team to collaborate and coordinate for building a product.

Download link:https://www.visualstudio.com/tfs/

3. Ansible: It is an open source Software configuration management tool. Apart from configuration management it also offers application deployment & task automation.

Download link:https://www.ansible.com/

Software Development Risk Management Plan with Examples

Most software engineering projects are risky because of the range of serious potential problems that can arise. The primary benefit of risk management is to contain and mitigate threats to project success. You have to identify and plan, and then be ready to act when a risk arises—drawing upon the experience and knowledge of the entire team to minimize the impact to the project.

Software Risk management includes the identification and classification of technical, programmatic and process risks, which become part of a plan that links each to a mitigation strategy. The project manager monitors risk during the project. If any materialize, a specific owner implements a mitigating action. In this article, we explain the elements of an effective software risk management plan and provide examples of plan elements.

Software Risk Management Plan

After cataloging risks according to type (technical, project, process, organizational), the software development project manager crafts a plan to record and monitor these risks. As part of a larger, comprehensive project plan, the risk management plan outlines the response that will be taken for each risk—if it materializes. The core of the risk management plan is the risk register, which describes and highlights the most likely threats to a software project.

Software Development Risk Register

To ensure that risks remain in the forefront of project management activities, it's best to keep the risk management plan as simple as possible. For both conventional and agile software project management methodologies, a **risk register** is a proven tool for organizing and referring to known projects risks. A comprehensive risk register would contain consist of the following attributes:

Description of risk — Summary description of the risk—easy to understand.

Recognition Date — Date on which stakeholders identify and acknowledge the risk.

Probability of occurrence — Estimate of probability that this risk will materialize (%).

Severity —The intensity of undesirable impact to the project—if the risk materializes.

Owner —This person monitors the risk and takes action if necessary.

Action —The contingent response if the risk materializes.

Status — current team view of the risk: potential, monitoring, occurring, or eliminated.

Loss Size —Given in hours or days, this is a measure of the negative impact to the project.

Risk Exposure —Given in hours or days, this is is a product of *probability* and *loss size*.

Priority (optional) **—**This is either an independent ranking, or the product of *probability* and *severity*. Typically, a higher-severity risk with high probability has higher relative priority.

Software Risk Register Example

For the purpose of illustration, we provide an example of a risk register that includes four of the attributes given above. It is sorted according to the probability of occurrence, and the total risk exposure is a sum of all the individual risk exposures.

Risk Description	**Probability of Occurrence**	**Loss Size (Days)**	**Risk Exposure (Days)**
Insufficient QA time to validate on all browsers and OS types.	45%	6	2.7
Lack of verifiable sample data may affect the ability of the primary external stakeholder to validate end product.	35%	18	6.3
Inadequate staff available from external stakeholders until very late in cycle.	25%	7	1.8
Following end-user testing, more effort on the user guide may be necessary.	25%	18	4.5
Backup and restore requires 3rd-party solutions (not evaluated yet).	20%	12	2.4
Insufficient time for external stakeholders to submit feedback on layout and composition of reports.	10%	5	0.5
		Total Risk Exposure	**18.2**

Software risk management is a balance of risk and reward; therefore, it is essential that—as the team reviews the requirements (user stories in the product backlog)—it must also evaluate the risk for each one. In software, a high risk often does not correspond with a high reward. Instead, the driving question for managing risk should be: Does the potential reward for each story or requirement warrant the level of risk that the team is assuming as it proceeds with

development? By considering alternatives, a development team can often achieve (nearly) the same level of reward without nearly as much risk. Adoption of this posture will help improve requirements prioritization.

RISK REGISTER TEMPLATE

PROJECT NAME	Library Management System
CLIENT	Vardhaman College of Engineering
POINT OF CONTACT	Librarian
VERSION NUMBER	1
DATE PREPARED	01-02-20
AUTHOR	Manager
BEGIN DATE	01-01-20
END DATE	01-03-20
DURATION	2 Months

RISK OVER VIEW				
RISK ID	**DESCRIPTION**	**OWNER**	**REPORT DATE**	**LAST UPDATED**
R101	Loss of data Storage devices	Admin	01-02-20	02-02-20
R102	People	Customer	02-02-20	02-02-20
R103	Schedule	Manager	03-02-20	03-02-20
R104	Cost	Manager	01-02-20	01-02-20
R105	Development Risk	Developer	03-02-20	03-02-20
R106	Data Communication	Admin	02-02-20	02-02-20

RISK ASSESSMENT				
RISK ID	**IMPACT LEVEL**	**IMPACT DESCRIPTION**	**OCCURRENCE LIKELIHOOD**	**PLAN STATUS**
R101	HIGH	It might be possible that the data stored might get lost due to damage of hard disk.	HIGH	PLAN IDLE
R102	MEDIUM	Fewer people than necessary are available. People with specific skills that are not available	MEDIUM	PLAN IDLE

R103	LOW	The underestimation of schedule also happens due to inexperience or optimism	LOW	PLAN IN PROGRESS
R104	HIGH	The degree of uncertainty that the project budget will be maintained. Underestimating the cost drivers.	HIGH	PLAN IN PROGRESS
R105	MEDIUM	Availability and quality of the tool used to make the project	HIGH	PLAN IN PROGRESS
R106	HIGH	Communication gap between the developing members of the project.	HIGH	PLAN IDLE

RISK RESPONSERESPONSE				
RISK ID	**PLANNED ACTION**	**COMPLETED ACTION**	**RESPONSE STATUS**	**DATE CLOSED**
R101	Work done should have at least two backups	No	RISK OPEN	03-02-20
R102	Training for critical areas of the project should provide	No	RISK OPEN	04-02-20
R103	Proper time management should be maintained	Yes	RISK OPEN	05-02-20
R104	Proper cost estimation should be there	Yes	RISK CLOSED	04-02-20
R105	Make the software available and have proper training on it.	No	RISK OPEN	10-03-20
R106	Frequent meetings should be organized for better	No	RISK OPEN	05-02-20

Development Risk Management Table

Impact Levels	HIGH	LOW	MEDIUM	
Response Status	RISK OPEN	RISK CLOSED		
Plan Status	PLAN IDLE	NO PLAN	PLAN IMPLEMENTED	PLAN FAILED
RISK OVER VIEW				
RISK ID	**DESCRIPTION**	**OWNER**	**REPORT DATE**	**LAST UPDATED**
D101	Interface	Developer	02-02-20	02-02-20
D102	Login	Front End Developer	10-02-20	10-02-20
D103	Display of Text books	Database Admin	02-02-20	02-02-20
RISK ASSESSMENT				
RISK ID	**IMPACT LEVEL**	**IMPACT DESCRIPTION**	**OCCURRENCE LIKELIHOOD**	**PLAN STATUS**
D101	HIGH	Interface error may display unwanted data to the client	HIGH	PLAN IDLE
D102	MEDIUM	After registration Login page is unable to display	MEDIUM	PLAN IDLE
D103	LOW	When client requests, the requested test books are not displaying properly	LOW	PLAN IN PROGRESS

RISK RESPONSE				
RISK ID	**PLANNED ACTION**	**COMPLETED ACTION**	**RESPONSE STATUS**	**DATE CLOSED**
D101	Change the internal code or use external API's to solve Interface errors	No	RISK OPEN	05-02-20

D102	Display Login page and connect database properly	No	RISK OPEN	02-02-20
D103	Write query to display text books	Yes	RISK OPEN	02-02-20

Design Structural Diagrams using CASE tool

4.Design structural diagrams using CASE tools

4. Withdraw money from an Automatic Teller Machine (ATM)

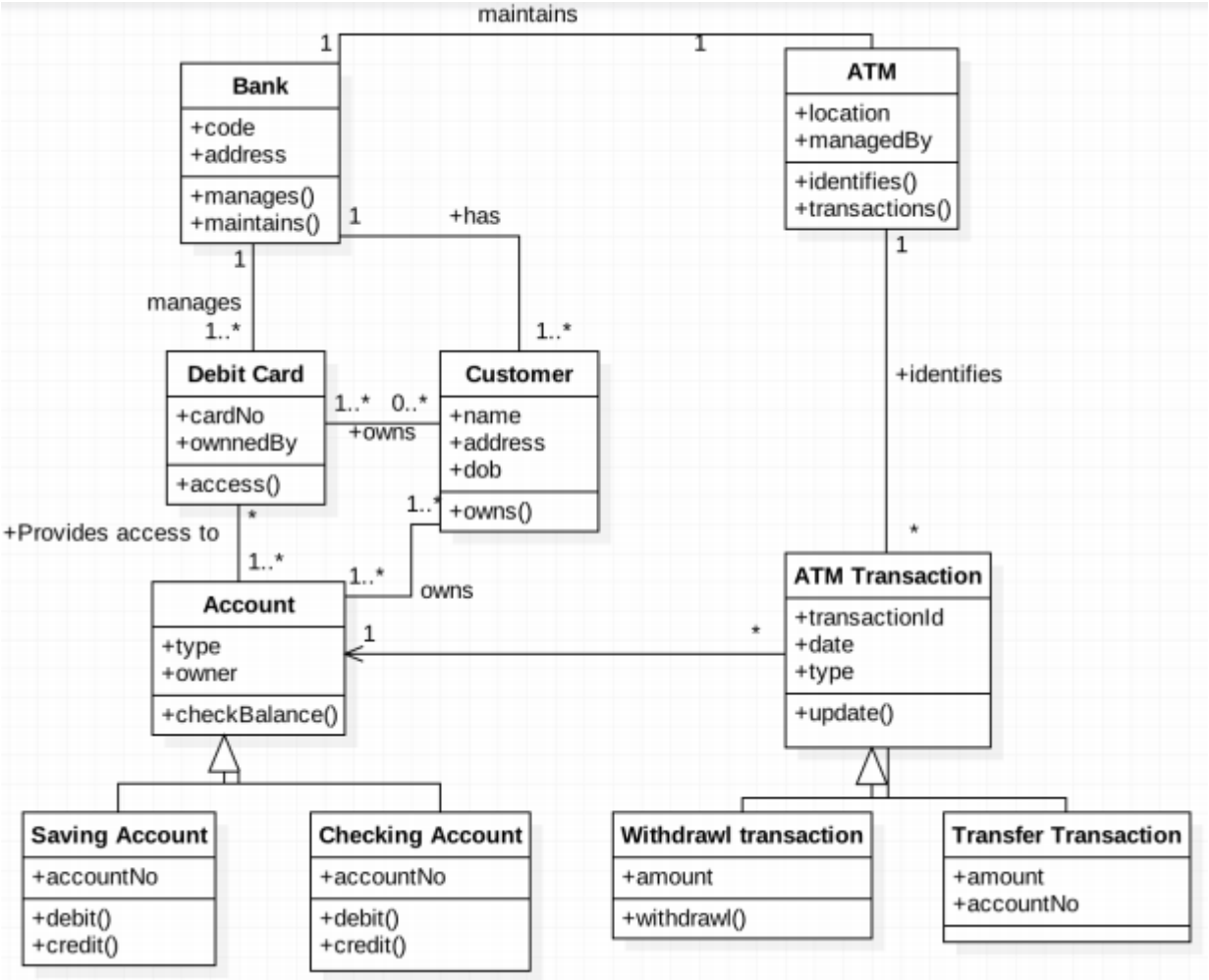
Class Diagram:

Class Diagram:-

Class diagrams describe the static structure of a system, or how it is structured rather than how it behaves. These diagrams contain the following elements:

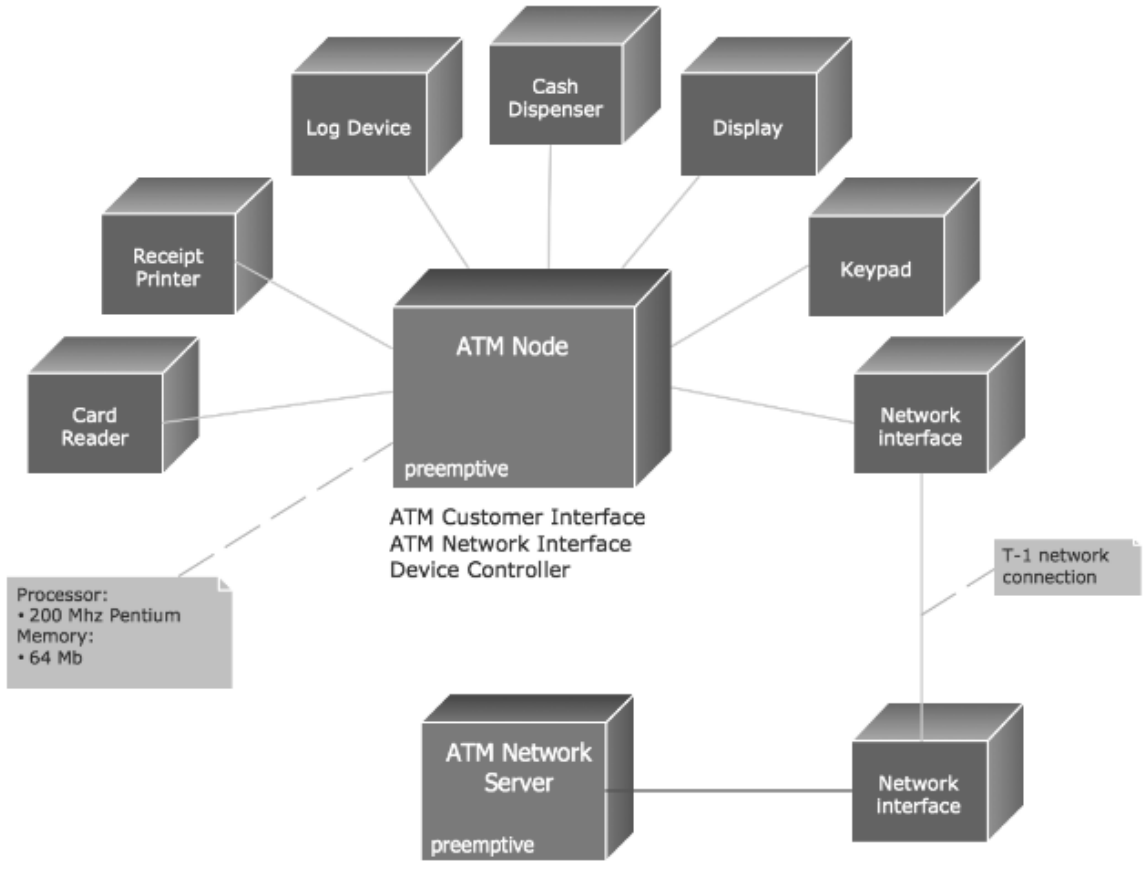
1. Classes , which represent entities with common characteristics or features. These features include attributes, operations, and associations.

2. Associations , which represent relationships that relate two or more other classes where the relationships have common characteristics or features. These features include attributes and operations.

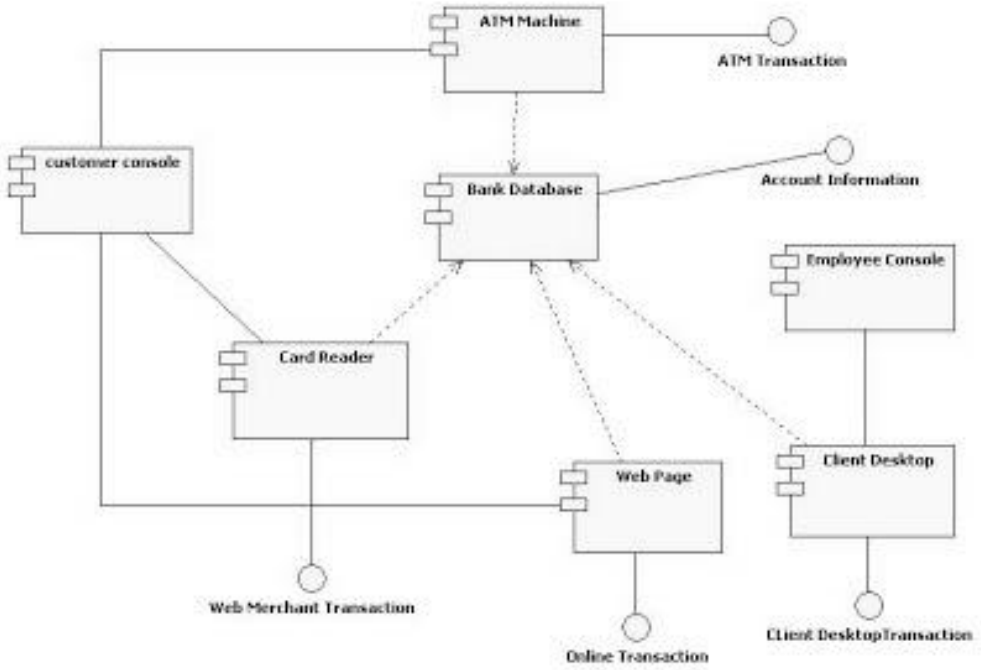


UML Deployment Diagram ATM System UML diagrams

The UML Deployment Diagram is used for visualization of elements and components of a program, that exist at the stage of its execution. It contains graphical representations of processors, devices, processes, and relationships between them. The UML Deployment Diagram allows to determine the distribution of system components on its physical nodes, to show the physical connections between all system nodes at the stage of realization, to identify the system bottlenecks and reconfigure its topology to achieve the required performance. The UML Deployment diagram is typically developed jointly by systems analysts, network engineers and system engineers.



UML Component Diagram ATM System UML diagrams



5. Design behavioral diagrams using CASE tools

Use Case Diagram: Use case diagrams describe the functionality of a system and users of the system. They contain the following elements:

1. Actors , which represent users of a system, including human users and other systems

2. Use cases , which represent functionality or services provided by a system to users Here, is a use case diagram for the ATM System.

5. Design Behavioral Diagrams using CASE tools

Sequence Diagram:

Sequence diagrams typically show the flow of functionality through a use case, and consist of the following components:

1. Actors , involved in the functionality

2. Objects , that a system needs to provide the functionality

3. Messages , which represent communication between objects Here, is an example of Sequence diagram for withdrawing amount from ATM.

Deposit Money (Bank Customer- ATM System)

Description The Deposit Money use-case for the customer is for the ATM System. Its primary function is to allow the user to deposit money into an account by means of the ATM System.

Actors

1. Bank Customer

2. ATM System

Main Flow

1. The user is presented with the main menu screen.

2. The user will request deposit option from the ATM by clicking on the deposit button of the main menu.

3. The user is presented with a screen that prompts the user for the account to where the money will be deposited and the amount of money to deposit.

4. The user enters the correct information and then clicks on the confirm button.

5. The ATM system sets updateBalanceNeeded on the users account.

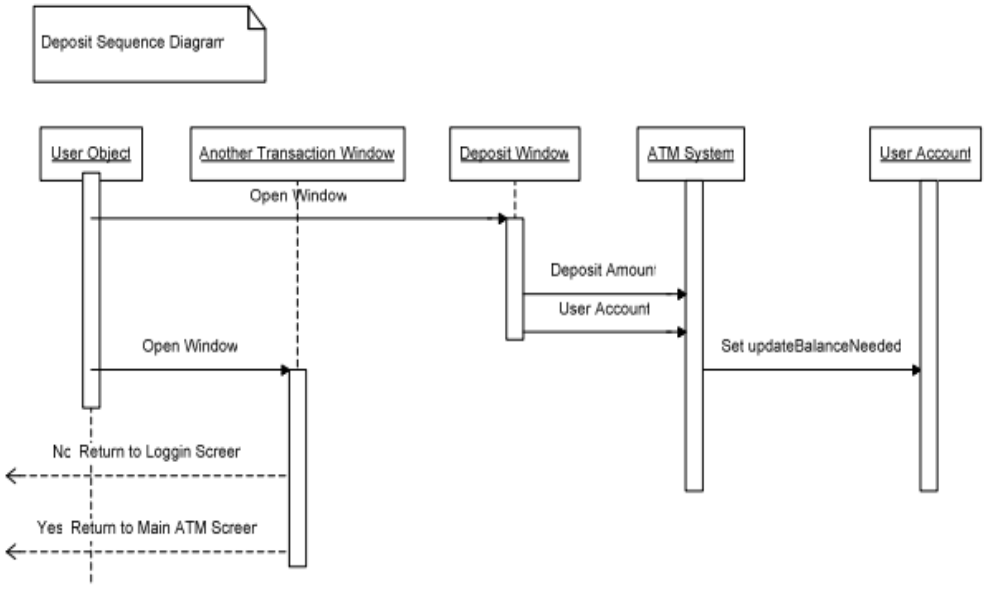
6. The ATM system will prompt the user for another transaction. If the user input is yes then the ATM system will reset to main menu screen. If the user input is No then the system will logout the user and return to the login screen.

Pre-Conditions

1. The user has been authenticated properly.

Post-Conditions

1. UpdateBalanceNeeded flag has been set for the appropriate account.


Deposit Sequence Diagram
User Object
Another Transaction Window
Deposit Window
ATM System
User Account
Open Window
Deposit Amoun
User Account
Open Window
Set updateBalanceNeeded
Nc Return to Loggin Screer
Yes Return to Main ATM Screer

Transfer Funds (Bank Customer – ATM system)

Description The Transfer Funds use-case for the customer is for the ATM System. Its primary function is to allow the user to transfer funds from one account to another.

Actors

1. Bank Customer

2. ATM System

Main Flow

1. The user is presented with the main menu screen.

2. The user requests transfer funds from ATM system main menu screen.

3. The user selects the accounts for outgoing funds and incoming funds.

4. The user enters the dollar amount of funds to transfer.

5. The ATM system queries the bank server to validate fund transfers vs account balances.

6. The ATM System queries the bank server to complete funds transfer.

7. The user is prompted for another transaction.

If the user input is Yes then go to main menu screen. If the user input is No then go to login screen.

Alternate Flow

1. If the user doesn't have the necessary funds to complete a funds transfer then the user is notified.

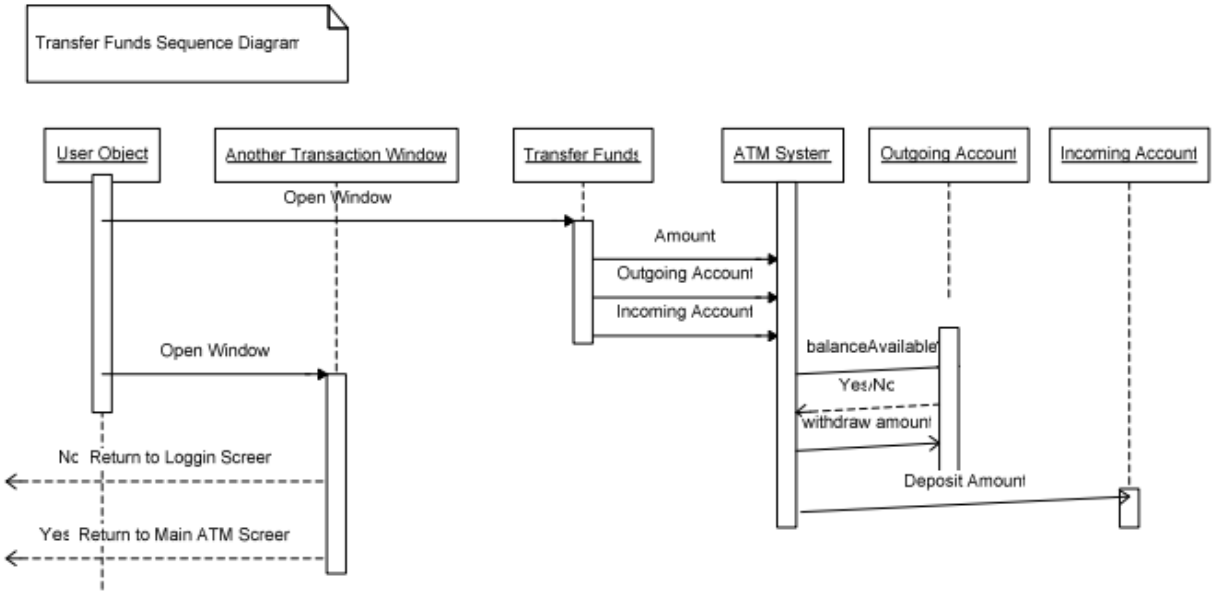
2. The user is prompted for another transaction. If the user input is Yes then go to main menu screen. If the user input is No then go to login screen.

Pre-Condition

1. The user has been authenticated properly.

Post-Condition

1. The funds are transferred from the outgoing account into the incoming account.


Transfer Funds Sequence Diagram
User Object
Another Transaction Window
Transfer Funds
ATM System
Outgoing Account
Incoming Account
Open Window
Amount
Outgoing Account
Incoming Account
Open Window
balanceAvailable
Yes/No
withdraw amount
No Return to Loggin Screen
Deposit Amount
Yes Return to Main ATM Screen

Authenticate (Bank Customer-ATM System)

Description The Authenticate use-case for the customer is for the ATM System. Its primary function is to ensure that the user is accessing their information and account and to ensure that no unauthorized user access the intended user's account.

Actors

1. Bank Customer

2. ATM System

Main Flow

1. System is at the welcome screen when user comes to the Authenticate UseCase. The welcome screen has a sign in prompt.

2. The welcome screen will prompt the user for his / her account number and password.

3. The user enters his / her information

4. The information is validated against the bank database.

5. If the validation process returns true, then the user is granted access to the ATM System.

Alternate Flows

1. If the validation returns false, then the user is informed and the ATM returns to the welcome screen.

2. If the user enters their information incorrectly three times, then their account is locked and the Bank Officer must unlock the account before the user can use the ATM again.

3. Any time the user tries to access their account before it is unlocked they will receive an message telling them that they account is still locked.

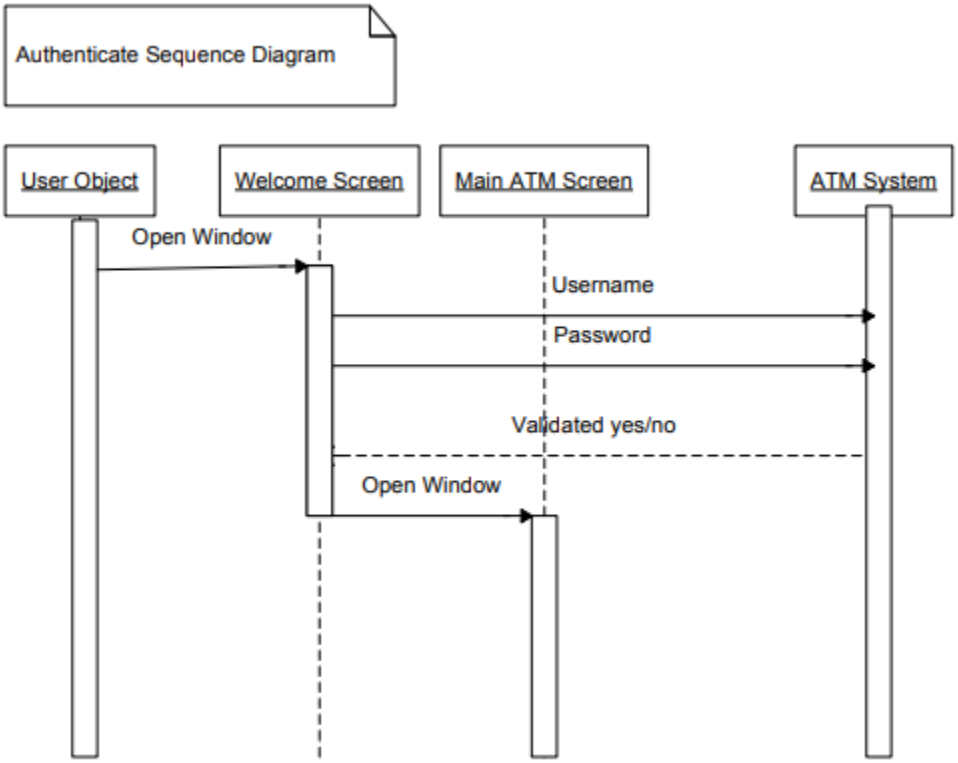
Pre-conditions

1. The ATM System is at the welcome screen when the user approaches the ATM.

Post-condition

1. If the user is validated, the ATM System is logged in and ready for the user to access their account.

2. If the user was not validated, then the ATM System is at the welcome screen waiting for a new customer.


Authenticate Sequence Diagram
User Object
Welcome Screen
Main ATM Screen
ATM System
Open Window
Username
Password
Validated yes/no
Open Window

Withdraw Money (Bank Customer-ATM System)

Description The Withdraw Money use-case is on the ATM System. Its primary function is to allow the user, once logged in, to remove money from the account of their choosing.

Actors

1. Bank Customer

2. ATM System

Main Flow

1. The user selects the Withdraw option from the main ATM Screen.

2. The user selects from a list of his / her accounts which one to withdraw from.

3. The user specifies in the amount box how much he / she wants to withdraw.

4. The ATM System verifies that the amount is in increments of $10.

5. The ATM System verifies that the user has the amount available in their account.

6. The balance is subtracted from the user's account, and they are deposited the cash.

7. The Another Transaction screen will appear and the user will select yes to return to the main ATM Screen or no to exit the ATM System. The user will be logged out automatically.

Alternate Flows

1. If the user does not enter an amount in increments of $10 then the user will be informed that the amount must be in increments of $10. The Withdraw screen will appear again for the user to retry their withdrawal.

2. If the desired balance is not available in the user's account then the Insufficient Funds window will appear to inform the user of the problem. The user can select OK and the system will return to the Withdraw screen.

Pre-conditions

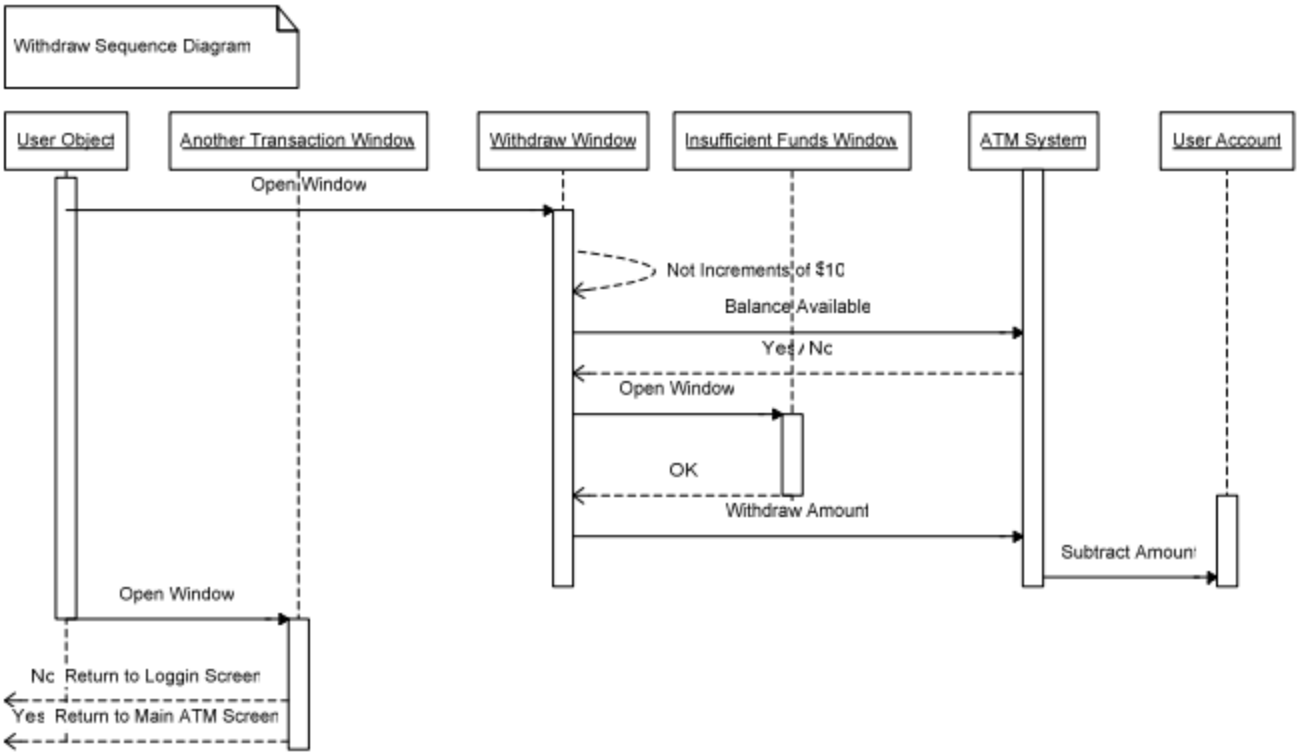
1. The user has logged into their account.

2. The user is at the main ATM Screen.

Post-condition

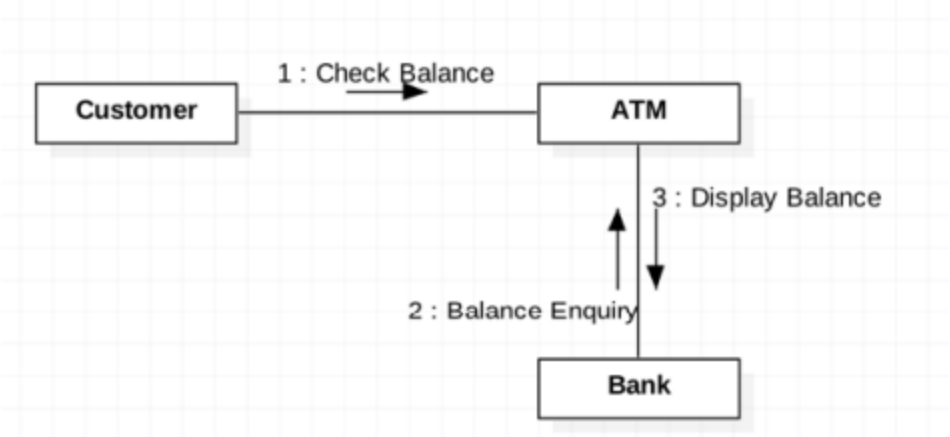
1. The user has withdrawn the money from their account if they entered the correct amount and if their account had that amount in it.

2. The user sees the main ATM Screen if they selected to do another transaction, or the ATM System has logged the user out and the user sees the welcome screen.

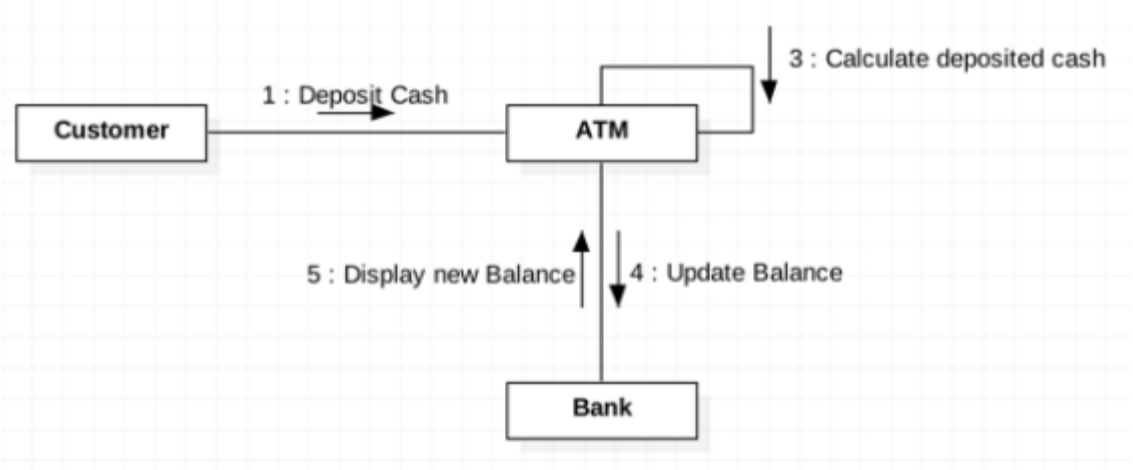

Withdraw Sequence Diagram
User Object
Another Transaction Window
Withdraw Window
Insufficient Funds Window
ATM System
User Account
Open Window
Not Increments of $10
Balance Available
Yes / Nc
Open Window
OK
Withdraw Amount
Subtract Amoun
Open Window
Nc Return to Loggin Screen
Yes Return to Main ATM Screen

Communication/Collaboration Diagrams:
A Communication or Collaboration diagram, as shown is a directed graph that uses objects and actors as graph nodes. The focus of the collaboration diagram is on the roles of the objects as they interact to realize a system function. Directional links are used to indicate communication between objects. These links are labeled using appropriate messages. Each message is prefixed with a sequence number indicating the time ordering needed to realize the system function.

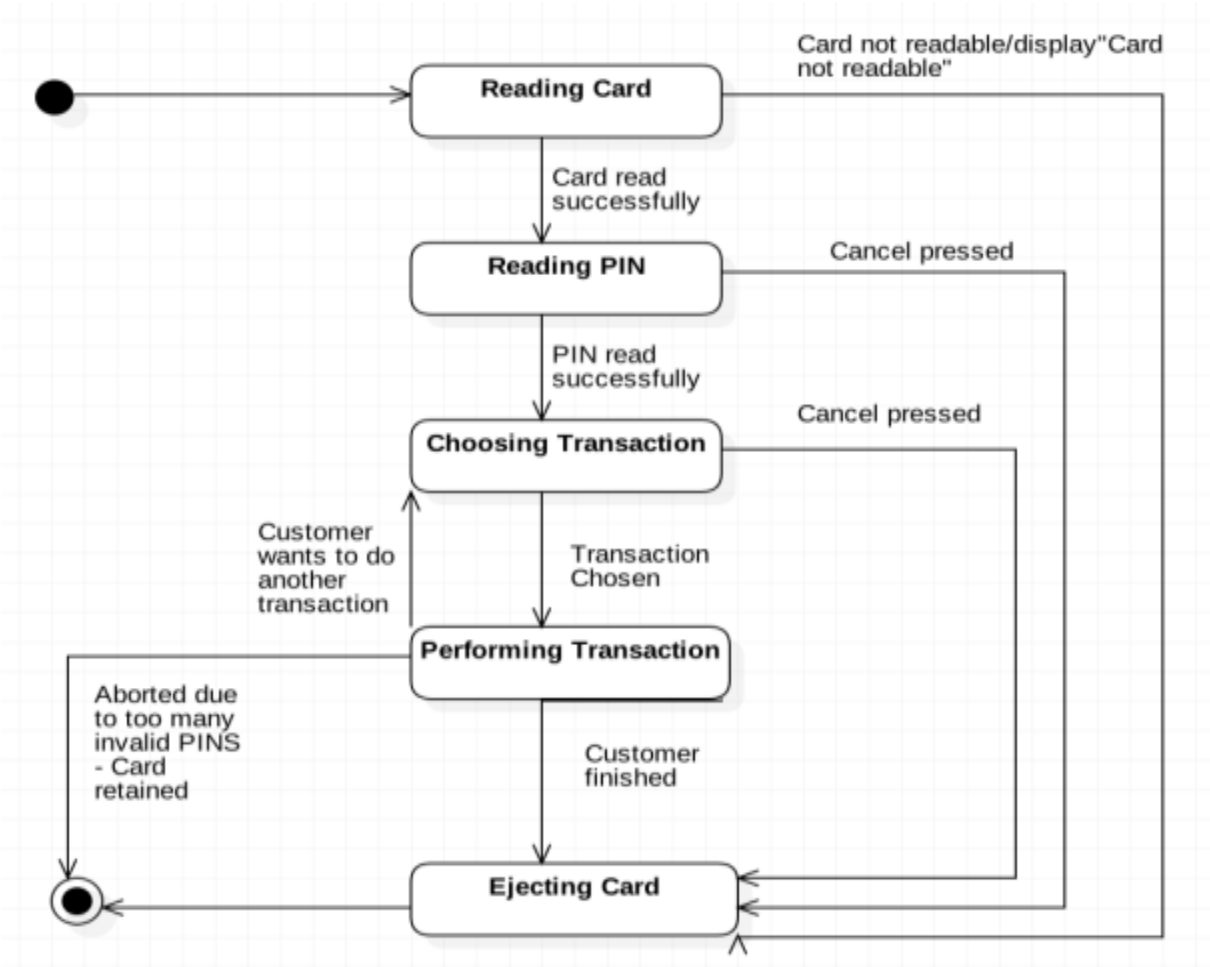
Here is an example of the Check Balance communication diagram:



Here is an example of the Deposit Cash communication diagram:



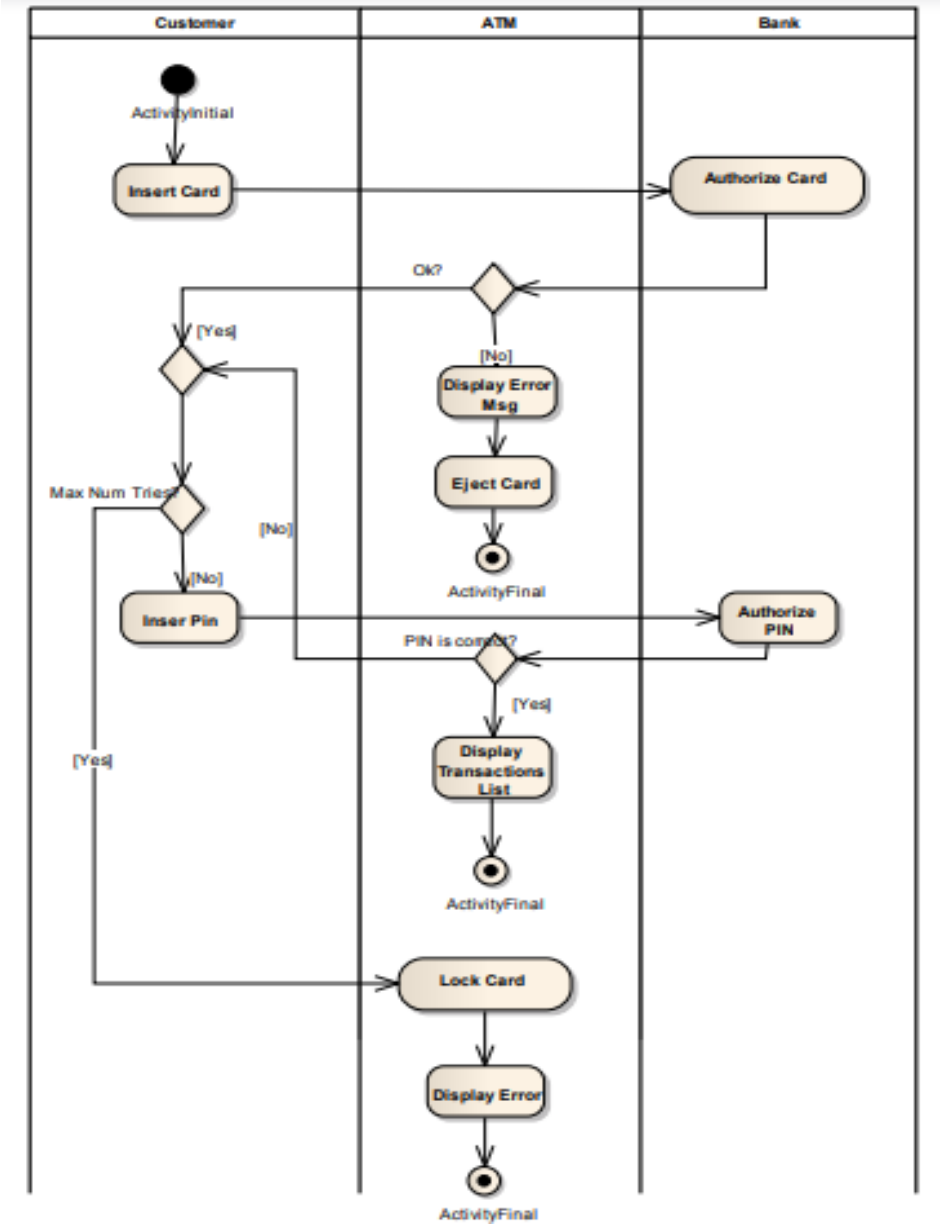
State Diagram:-
For behavior: State, Activity Diagram State Diagram:- State transition diagrams provide a way to model the various states in which an object can exist. While the class diagram show a static picture of the classes and their relationships, state transition diagrams model the dynamic behavior of a system in response to external events (stimuli). State transition diagrams consist of the following:
1. States , which show the possible situations in which an object can find itself
2. Transitions , which show the different events which cause a change in the state of an object. Here, is an example of the state diagram for the session of ATM.

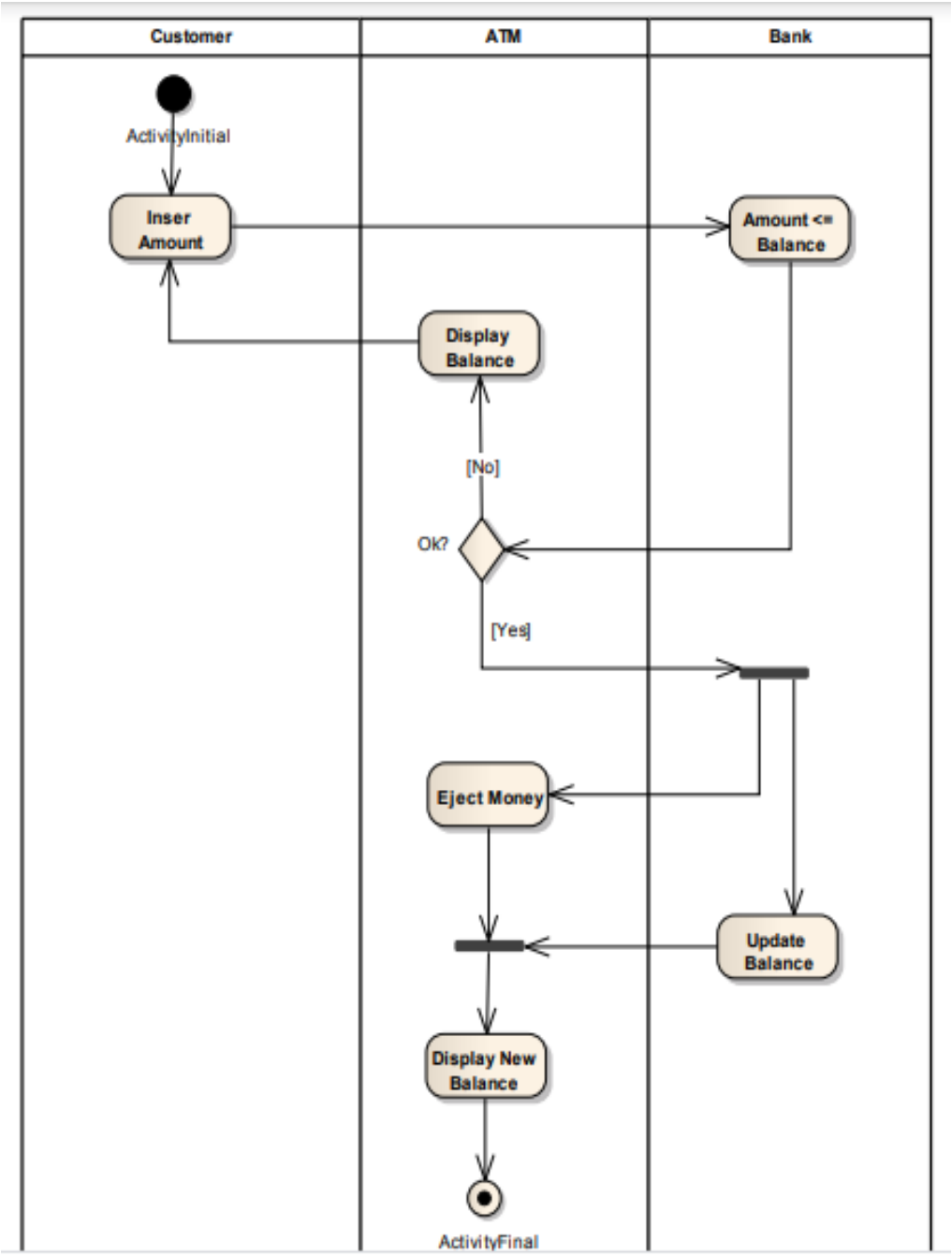

Reading Card
Card not readable/display"Card not readable"
Card read successfully
Reading PIN
Cancel pressed
PIN read successfully
Choosing Transaction
Cancel pressed
Customer wants to do another transaction
Transaction Chosen
Performing Transaction
Aborted due to too many invalid PINS - Card retained
Customer finished
Ejecting Card

Activity Diagram:-

Activity diagrams describe the activities of a class. They are similar to state transition diagrams and use similar conventions, but activity diagrams describe the behavior/states of a class in response to internal processing rather than external events. They contain the following elements:

1. Swimlanes , which delegate specific actions to objects within an overall activity
2. Action States , which represent uninterruptible actions of entities, or steps in the execution of an algorithm
3. Action Flows , which represent relationships between the different action states on an entity
4. Object Flows , which represent utilization of objects by action states, or influence of action states on objects. Following are the examples of Login, Withdraw Activity Diagrams.




Customer
ATM
Bank
ActivityInitial
Inser Amount
Amount <= Balance
Display Balance
[No]
Ok?
[Yes]
Eject Money
Update Balance
Display New Balance
ActivityFinal

6. Develop test cases for unit testing and integration testing

What is Unit Testing?

UNIT TESTING is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing. Unit testing is a WhiteBox testing technique that is usually performed by the developer. Though, in a practical world due to time crunch or reluctance of developers to tests, QA engineers also do unit testing.

Why Unit Testing?

Unit Testing is important because software developers sometimes try saving time doing minimal unit testing and this is myth because inappropriate unit testing leads to high costdefectfixing during system Testing, Integration Testing and even Beta Testing after application is built. If proper unit testing is done in early development, then it saves time and money in the end.

Unit Testing Levels

1. Unit tests help to fix bugs early in the development cycle and save costs.
2. It helps the developers to understand the code base and enables them to make changes quickly
3. Good unit tests serve as project documentation
4. Unit tests help with code re-use. Migrate both your code andyour tests to your new project. Tweak the code until the tests run again.

How to do Unit Testing

In order **to do Unit Testing**, developers write a section of code to test a specific function in software application. Developers can also isolate this function to test more rigorously which reveals unnecessary dependencies between function being tested and other units so the dependencies can be eliminated. Developers generally use UnitTest framework to develop automated test cases for unit testing.

Unit Testing is of two types

- Manual
- Automated

Unit testing is commonly automated but may still be performed manually. Software Engineering does not favor one over the other but automation is preferred. A manual approach to unit testing may employ a step-by-step instructional document.

Under the automated approach-

- A developer writes a section of code in the application just to test the function. They would later comment out and finally remove the test code when the application is deployed.
- A developer could also isolate the function to test it more rigorously. This is a more thorough unit testing practice that involves copy and paste of code to its own testing environment than its natural environment. Isolating the code helps in revealing unnecessary dependencies between the code being tested and other units or data spaces in the product. These dependencies can then be eliminated.
- A coder generally uses a UnitTest Framework to develop automated test cases. Using an automation framework, the developer codes criteria into the test to verify the correctness of the code. During execution of the test cases, the framework logs failing test cases. Many frameworks will also automatically flag and report, in summary, these failed test cases. Depending on the severity of a failure, the framework may halt subsequent testing.
- The workflow of Unit Testing is 1) Create Test Cases 2) Review/Rework 3) Baseline 4) Execute Test Cases.

Unit Testing Techniques

The **Unit Testing Techniques** are mainly categorized into three parts which are Black box testing that involves testing of user interface along with input and output, White box testing that involves testing the functional behavior of the software application and Gray box testing that is used to execute test suites, test methods, test cases and performing risk analysis.

Code coverage techniques used in Unit Testing are listed below:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Condition Coverage
- Finite State Machine Coverage

Unit Testing Best Practices

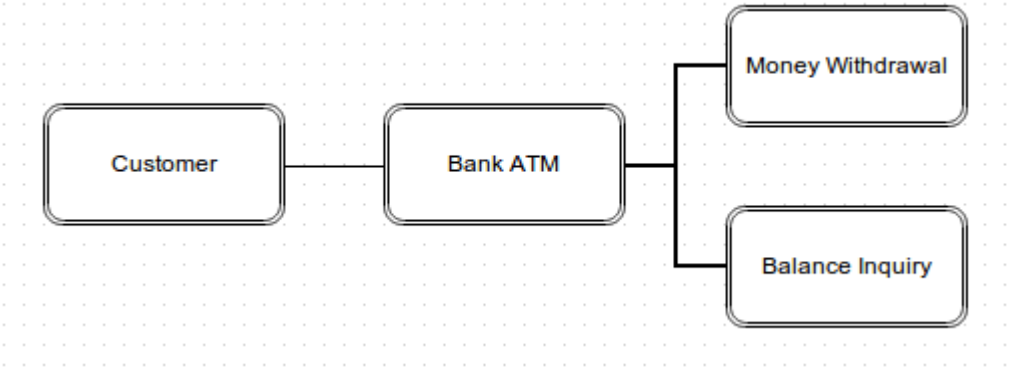
- Unit Test cases should be independent. In case of any enhancements or change in requirements, unit test cases should not be affected.
- Test only one code at a time.
- Follow clear and consistent naming conventions for your unit tests
- In case of a change in code in any module, ensure there is a corresponding unit Test Case for the module, and the module passes the tests before changing the implementation
- Bugs identified during unit testing must be fixed before proceeding to the next phase in SDLC
- Adopt a "test as your code" approach. The more code you write without testing, the more paths you have to check for errors.

Test Cases for ATM (Test Scenarios ATM Machine)

ATM machines work 24×7 and its related to money. We need to test it for accuracy, reliability, and performance.

Here we will list most of the test scenarios of ATM by covering major aspects of the ATM machine. Following are the test cases for ATM Machine. The list consists of both Positive and Negative test scenarios login page.

ATM Machine Test Cases Flow



ATM test cases may vary from one bank to another. Each back has its process. You can add test cases based on your companies requirement document.

Manual Test Cases for ATM:

1.Verify the ‘ATM Card Insertion Slot’ is as per the specification
2.Verify the ATM machine accepts card and PIN details
3.Verify the error message by inserting a card incorrectly
4.Verify the error message by inserting an invalid card (Expired Card)
5.Verify the error message by entering an incorrect PIN
6.Verify that the user is asked to enter the PIN after inserting a valid ATM Card
7.Verify that PIN is encrypted
8.Verify that there is an action like blocking of card occurs when the total no. of incorrect PIN attempts get surpassed
9.Verify the user is allowed to do only one cash withdrawal transaction per PIN request
10.Verify the machine logs out of the user session immediately after successful withdrawal
11.Verify the message when there is no money in the ATM
12.Verify the language selection functionality
13.Verify the cash withdrawal functionality by entering some valid amount
14.Verify the cash withdrawal functionality by entering an amount less than 100
15.Verify the cash withdrawal functionality by entering an amount greater than the total available balance in the account.
16.Verify the cash withdrawal functionality by entering an amount greater than per day limit
17.Verify the user is allowed to enter the amount again in case the amount entered is not valid. proper message should be displayed.
18.Verify the ATM machine successfully takes out the money.
19.Verify the ATM machine takes out the balance printout after the withdrawal
20.Verify the font of the text displayed in ATM screen
21.Verify the text on the screen buttons visible clearly.
22.Verify the functionality of all the buttons on the keypad
23.Verify the text on the buttons visible clearly.
24.Verify that touch of the ATM screen is smooth and operational
25.Verify the user is allowed to choose different account types like Savings, Current etc.,
26.Verify the different combinations of operation and check if there will be an electricity loss in the middle of the operation. If there is an electricity loss in the middle of the transaction then the transaction should be marked as null and the amount shouldn’t be disclosed to others.
27.Verify the functionality of the cash dispenser
28.Verify the functionality of the receipt printer
29.Verify whether the printed data is correct or not in the receipt
30. Verify how much time the system takes to log out.

Format for Test Case Design:

Project Name:	[redacted]
Module Name:	[redacted]
Reference Document:	[redacted]
Created by:	[redacted]
Date of creation:	[redacted]
Date of review:	[redacted]

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/ FAIL)
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	Successful login	Gmail inbox is shown		
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					

Test Cases for ATM

TEST CASE NAME	TEST CASE DESCRIPTION	P	STEP NAME	STEP DESCRIPTION	TEST DATA	EXPECTED RESULT
TC01_Bank_ Money withdrawal_ verify card insertion with valid cards	This test case to validate card insertion functionality with valid card	P0	Step1	Insert valid card in the insertion point of atm	Valid atm card	Atm should display language page with following objects ENLISH,TELUGU,HINDI
TC02_Bank_ Money withdrawal_ verify card insertion with invalid cards	This test case to validate card insertion functionality with invalid card		Step1	Insert invalid card of other bank card in the insertion point of atm	Invalid card like other bank card	Atm should not accept the card and display a message "please insert valid atm card"
			Step 2	Insert invalid card of expired atm card in the insertion point of atm	Invalid card like expired atm card	Atm should not accept the card and display a message "Sorry unable to process your request code 1234"
TC03_Bank_ Money withdrawal_ verify card insertion with valid cards in wrong angle	This test case to validate card insertion functionality with valid card in wrong angle		Step1	Insert invalid card of expired atm card in the insertion point of atm in wrong angle	Valid atm card	Atm should not accept the card and display a message " Sorry unable to process your request code 1222"
TC04_Bank_ Money withdrawal_ verify language selection	This test case to verify the language selection functionality		Step1	Insert valid card in the insertion point of atm	Valid atm card	Atm should display language page with following objects ENLISH,TELUGU,HINDI
			Step 2	Click on corresponding language to be used		Atm should display the pin number page in corresponding language selected
TC04_Bank_ Money withdrawal_ verify pin number entry with valid pin number	This test case is to verify the functionality of pin number functionality with valid pin number		Step1	Insert valid card in the insertion point of atm	Valid atm card	Atm should display language page with following objects ENLISH,TELUGU,HINDI
			Step 2	Click on corresponding language to be used		Atm should display the pin number page in corresponding languge selected
			Step 3	Enter the valid pin number	Valid pin number	Atm should display the account type selction page
TC05_Bank_ Money withdrawal_ verify pin number entry with invalid pin number	This test case is to verify the functionality of pin number functionality with invalid pin number		Step1	Insert valid card in the insertion point of atm	Valid atm card	Atm should display language page with following objects ENLISH,TELUGU,HINDI
			Step 2	Click on corresponding language to be used		Atm should display the pin number page in corresponding languge selected
			Step 3	Enter the invalid pin number	InValid pinnumber	Atm should display meaning full message" Sorry unable to process your request code 1222"

TC06_Bank_ Money withdrawal_ verify pin number entry with invalid pin number upto 3 times	This test case is to verify the functionality of pin number functionality with invalid pin number upto 3 times		Step1	Insert valid card in the insertion point of atm	Valid atm card	Atm should display language page with following objects ENLISH,TELUGU,HINDI
			Step 2	Click on corresponding language to be used		Atm should display the pin number page in corresponding languge selected
			Step 3	Enter the invalid pin number	Invalid pinnumber	Atm should display meaning full message"enter valid pinnumber"
			Step 4	Enter the invalid pin number	Invalid pinnumber	Atm should display meaning full message"enter valid pinnumber"
			Step 5	Enter the invalid pin number	Invalid pinnumber	Atm should display meaning full message" Sorry unable to process your request code 1236"
TC07_Bank_ Money withdrawal_ verify account type selection with correct account type	This test is to verify the functionality of the account selection type with correct account type		Step1	Insert valid card in the insertion point of atm	Valid atm card	Atm should display language page with following objects ENLISH,TELUGU,HINDI
			Step 2	Click on corresponding language to be used		Atm should display the pin number page in corresponding language selected

Note: as a student you can follow any test case format

7. Develop test cases for various system and regression testing techniques

9 Tips for Selecting Test Cases for Regression Testing

Regression testing can be defined as testing done to verify if the code changes in a software/app have any kind of impact on the existing functionality of the product.

1. Select test cases for Regression testing where there are recent code changes or functional changes

Testers should first concentrate on all those test cases wherever there are recent changes in code/functionality of the application. There are higher chances of encountering issues in these areas. In the test case documentation, maintain the history of functionality changes in order to identify test cases which can be used in the regression suite later.

2. Select test cases that map to the business requirements

It should be ensured that the functionality of the application is working fine as per the software requirements specified by the client.

3. Select test cases for Regression testing in areas with frequent bugs/defects

Tester has to select the test cases in the areas where there are frequent bugs or defects. There might be some areas in the software which usually fail even with a small coding change. There are more chances of getting multiple issues in these areas.

4. Select test cases for Regression testing of the areas which are visible to the user

Suppose there is a web application on 'School Management'. There might be several modules in it. Suppose in one of the modules, the logo of the application is incorrect. That's an issue.

The issue is low in severity but has high priority from a user's point of view. The tester has to select the test cases at the areas which are visible to the user.

5. Select all integration test cases for Regression testing

Integration testing is a type of software testing in which individual software modules are combined and they are tested in a group. Suppose there is 1 application that has 2 modules which are integrated in a way that module 2 uses data of module 1.

Now, whenever there is any update in the data of module 1, the data of module 2 doesn't get updated.

This can be due to integration issues in the interface which is not allowing data transfer from module 1 to module 2. Testers need to check for these types of integration issues. All such integration test cases need to be selected during the regression testing.

6. Select all complex test cases for Regression testing

Software testing is becoming complex day by day. Most of the time, on executing complex test cases, the application gets crashed or the performance gets affected. This can become a serious issue.

To test the complexity efficiently, testers should use different methods and techniques to ensure software quality. Testers have to make sure that all complex test cases are included in the regression testing suite.

7. Select test cases based on priorities for Regression testing

With time, the regression test suite may become big and take a significant amount of time for execution. In such scenarios, to reduce the testing time and efforts, first select all the test cases which are of high priority from the user's/customer's point of view.

Make sure that these test cases are working fine followed by other test cases which are of medium and low priorities if time permits.

8. Select test cases for Regression testing based on criticality and impact of bug fixes

If the criticality and impact of bug fixes is low, then the tester can select just the high priority test cases for the impacted functionalities.

If the criticality and impact of bug fixes is high, then the tester needs to select all test cases from high, medium and low priorities.

9. Select a sample of successful and failed test cases for Regression testing

During regression testing, the tester needs to select some of the test cases which were earlier passed or failed. A test case that fails for a reason unrelated to the functionality of the application is termed as ‘failed test case’. As some areas in an application are more prone to errors, they usually fail even after making a small code change.

ONLINE RESOURCES

OBJECTIVE

To help students on acquiring more practice on the course using various online resources

LINKS

https://en.wikipedia.org/wiki/Software_configuration_management

https://en.wikipedia.org/wiki/Software_requirements_specification

http://www.math-cs.gordon.edu/local/courses/cs211/ATMExample/

http://www.math-cs.gordon.edu/local/courses/cs211/ATMExample/

https://nptel.ac.in/courses/106/105/106105087/

http://www.softwaremag.com/

https://www.coursera.org/learn/software-processes

POSSIBLE VIVAQUESTIONS

POSSIBLE VIVA QUESTIONS

Description

Possible viva questions includes, questions whose answers are provided. The student can have a practice of them.

	What is a class? A class is a set of objects that share a common structure and a common behavior.
	What is the need of an Object diagram? An object diagram is used to show the existence of objects and their relationships in the logical design of a system.
	Define Static model? It can be viewed as a snapshot of a system's parameters at rest or a specific point in time. They are needed to represent the structural or static aspect of a system.
	Define Dynamic model? It can be viewed as a collection of procedures or behaviors that taken together reflect the behavior of a system over time. Dynamic modeling is the most useful during the design and implementation phases of the system development.
	What is a use case? Use cases are scenarios for understanding system requirements. A use case is an interaction between users and a system.
	Name the three types of relationships in a use case diagram. Communication, Uses, extends.
	Write the two types of Implementation diagram? Component diagram, deployment diagram.
	What is an activity? An activity is a set of operations that is executing during the entire period an object is in a state.
	What is the need of a Class diagram? A class diagram is used to show the existence of classes and their relationships in the logical view of a system.
	What is Behavior of an object? Behavior is how an object acts and reacts in terms of its state changes and message passing.
	What is verification and validation Verification is a term that refers to the set of activities which ensure that software implements a specific function. Validation refers to the set of activities which ensure that software that has been built according to the need of clients.
	What is Software configuration management? Software configuration management is a process of tracking and controlling changes that happen in the software.

	What are software requirements? Software requirements are a functional description of a proposed software system. It is assumed to be the description of the target system, its functionalities, and features.
	What are functional and non-functional requirements? Functional requirements are functional features which are expected by users from the proposed software product. Non-functional requirements are related to security, performance, look, and feel of the user interface.
	What is black-box and white-box testing? Black-box testing checks if the desired outputs are produced when valid input values are given. It does not verify the actual implementation of the program. White-box testing not only checks for desired and valid output when valid input is provided but also it checks if the code is implemented correctly.
	Define Software Prototyping. Software prototyping is represented as rapid software development for validating the requirements.
	What is Software Quality Assurance? Software Quality Assurance is a set of auditing and documenting functions that assess the effectiveness and completeness of quality control activities.
	What is SRS? SRS stands for Software Requirement Specification (SRS) document. It is a document to capture all the functional and non-functional requirements of a product. Not all SDLC models need to follow SRS documents, some models capture requirements in the form of user stories, whereas some models in the form of excel sheets, etc.
	What is change control management? Change control is a function which ensures that all changes made into the software system are consistent and created using organizational rules and regulations

KNOWLEDGE BASE

KNOWLEDGE BASE

Description

Knowledge base includes possible viva questions whose answers are not provided. The student has to explore for the answers.

POSSIBLE VIVA QUESTIONS (WITHOUT ANSWERS)

1. What do you understand by the term 'Data Modeling'?
2. What are the different types of data models?
3. What are the important types of relationships in a data model and explain them?
4. What is a Surrogate key?
5. What are Forward Engineering and Reverse Engineering in a data model?
6. What is the Logical data model, Logical data modeling, Physical data model and Physical data modeling?
7. What is the main difference between Snow Flake Schema and Star Flake Schema?
8. Describe Data Sparsely and how does it impact on aggregation?
9. Describe the subtype entity and super type entity?
10. What are Relational data modeling, conceptual Data model, and conceptual data modeling?
11. What is computer software?
12. Can you differentiate computer software and computer program?
13. What is software engineering?
14. When you know programming, what is the need to learn software engineering concepts?
15. What is software process or Software Development Life Cycle (SDLC)?
16. What are SDLC models available?
17. What are various phases of SDLC?
18. Which SDLC model is the best?
19. What is software project management?
20. Who is software project manager?
21. What does software project manager do?
22. What is software scope?
23. What is project estimation?
24. How can we derive the size of software product?
25. What are function points?

26. What are software project estimation techniques available?
27. What is baseline?
28. What is Software configuration management?
29. What is change control?
30. How can you measure project execution?
31. Mention some project management tools.
32. What are software requirements?
33. What is feasibility study?
34. How can you gather requirements?
35. What is SRS?
36. What are functional requirements?
37. What are non-functional requirements?
38. What is software measure?
39. What is software metric?
40. What is modularization?
41. What is concurrency and how it is achieved in software?
42. What is cohesion?
43. What is coupling?
44. Mentions some software analysis & design tools?
45. What is level-0 DFD?
46. What is the difference between structured English and Pseudo Code?
47. What is data dictionary?
48. What is structured design?
49. What is the difference between function oriented and object oriented design?
50. Briefly define top-down and bottom-up design model.
51. What is the basis of Halstead's complexity measure?
52. Mention the formula to calculate Cyclomatic complexity of a program?
53. What is functional programming?
54. Differentiate validation and verification?
55. What is black-box and white-box testing?
56. Quality assurance vs. Quality Control?
57. What are various types of software maintenance?
58. What is software re-engineering?

59. What are CASE tools?

60.What are the important categories of software?

61.What is the main difference between a computer program and computer software?

62.What is software re-engineering?

63.Describe the software development process in brief:

64.What are SDLC models available?

65.What is verification and validation?

66.In software development process what is the meaning of debugging?

67.How can you make sure that your code is both safe and fast?

68.Name two tools which are used for keeping track of software requirements?

69.What is the main difference between a stubs, a mock?

70.What language do you like to write programming algorithms?

71.What is computer software?

72.According to you which SDLC model is the best?

73.Who is software project manager? What is his role?

74.What is mean by software scope?

75.How to find the size of a software product?

76.What are function points?

77.What are software project estimation techniques available?

78.What is Software configuration management?

79.How can you measure project execution?

80.Tell me about some project management tools.

81.What are software requirements?

82.What is feasibility study?

83.What are functional and non-functional requirements?

84.What is software metric?

85.What is modularization?

86.What is cohesion?

87.Mentions some software analysis & design tools?

88.What is mean by level-0 Data flow diagram?

89.What is the major difference between structured English and Pseudo Code?

90.What is structured design?

91.What is functional programming?

92.What is Quality Assurance vs. Quality Control?

93.What are CASE tools?

94.Which process model removes defects before software get into trouble?

95.Solve this problem

96.How you can make sure that your written code which can handle various kinds of error situation?

97.Explain the differences between a Thread and a Process?

98.Tell me the difference between an EXE and a DLL?

99.What is strong-typing and weak-typing? Which is preferred? Why?

100.Describe the difference between Interface-oriented, Object-oriented and Aspect-oriented programming.

101.Why using catch (exception) is always a bad idea?

102.What type of data is passed via HTTP Headers?

103.How do you prioritize requirements?

104.Give me differences between object-oriented and component-based design?

105.When do you use polymorphism?

106.What is the difference between stack and queue?

107.What is essential for testing the quality of the code?

108.Do you think that the maintenance of software is expensive?

109.Give me differences between tags and branches?

110.Where is a protected class-level variable available?

111.Is it possible to execute multiple catch blocks for a single try statement?

112.When do you need to declare a class as abstract?

113.Develop an algorithm that output your current location and a list of ATMs locations in that area. Get you the closest K ATMs to your location.

114. What Is SDLC?

115.Name five Models used in SDLC

116.Explain Phases of the waterfall model

117.States the importance of Design phase?

118.What are the tasks performed in Coding phase?

119.What is feasibility study?

120.What are the Maturity levels in CMM?

121.Give some benefits of using V model?

123.What is the 'scope' of a project?

124.According to you, when should users be trained on a new system?

125.Name the phase where the performance of the new system monitored?

126.What is a computer-based information system?

127.Explain Low Level Or Detailed Design concerning SDLC

128.What is the use of JAD session?

129.State the Difference Between SDLC And Stlc

130.Who are the people involved in the phases of Waterfall Model

131.What is level-0 DFD?

132.Explain the team Requirement Gathering concerning SDLC

133.Briefly explain Testing Phase

134.What are problems faced in the waterfall model?

135.What is the details study of the existing system is called?

136.What is the main aim of prototyping aim?

137.In which step of SDLC project early termination could be done?

138.According to you which is most creative and challenging phase of system life cycle?

139.Name the type of feasibility where the cost saving and additional profits will exceed the investment required.

140.Can bug fixes also include software maintenance?

141.Cost of error correction is least in which stage of SDCL life cycle?

142.What is an Interface?

143. What is an API?

144. What are main differences between API and Web Service?

145. What are the advantages of API Testing?

146.What are the common API testing types?

147.What is the difference between the QA and software testing?

148. What is Testware?

149.What is the difference between build and release?

150.What are the automation challenges that SQA(Software Quality Assurance) team faces while testing?

151.What is bug leakage and bug release?

152. What is data driven testing?

153. Explain the steps for Bug Cycle?

154.What does the test strategy include?

155.Mention the different types of software testing?

156.What is branch testing and what is boundary testing?

157.What are the contents of test plans and test cases?

158.What is Agile testing and what is the importance of Agile testing?

159.What is Test case?

160.What is the strategy for Automation Test Plan?

161.What is quality audit?

162.What are the tools used by a tester while testing?

163.Explain stress testing, load testing and volume testing?

164.What are the five common solutions for software developments problems?

165.What is a 'USE' case and what does it include?

166.What is CRUD testing and how to test CRUD?

167.What is thread testing?

168.What is configuration management?

169.What is Ad Hoc testing?

170.List out the roles of Software Quality Assurance engineer?

171.Explain what is Bug triage?

172.List out various tools required to support testing during development of the application?

173.What is a cause effect graph?

174.What is a Test Metric in software testing and what information does it contains?

175.Explain what is traceability matrix?

176.Explain what is the difference between Regression testing and Retesting?

177.List out the software quality practices through the software development cycle?

178.Explain what is the rule of a "Test Driven Development"?

179.Mention what are the types of documents in SQA?

180.Explain what should your QA documents include?

181.Explain what is MR and what information does MR consists of?

182.What does the software QA document should include?

183.Mention how validation activities should be conducted?

184. Which of these are not among the eight principles followed by Software Engineering Code of Ethics and Professional Practice ?

185.Which of these does not account for software failure ?