**Software Requirements**

**Specification**

**for**

**Bank Management System**

**Version 1.0 approved**

**Siddharth Misra ( TE-COMP-B-65)**

**Sahil ParanjapeTE-COMP-B-50)**

**Dwij Patel( TE-COMP-B-42)**

**17 Feb 2022**

# Table of Contents

Table of Contents [ii](#_bookmark0)

1. Introduction [1](#_bookmark1)

Purpose [1](#_bookmark2)

Document Conventions [1](#_bookmark3)

Intended Audience and Reading Suggestions [1](#_bookmark4)

Product Scope [1](#_bookmark5)

[References 2](#_bookmark6)

1. Overall Description [2](#_bookmark7)

Product Perspective [2](#_bookmark8)

Product Functions [2](#_bookmark9)

[User Classes and Characteristics 3](#_bookmark10)

[Operating Environment 3](#_bookmark11)

[Design and Implementation Constraints 4](#_bookmark12)

[User Documentation 4](#_bookmark13)

[Assumptions and Dependencies 4](#_bookmark14)

1. [External Interface Requirements 5](#_bookmark15)

[User Interfaces 5](#_bookmark16)

[Hardware Interfaces 5](#_bookmark17)

[Software Interfaces 5](#_bookmark18)

[Communications Interfaces 6](#_bookmark19)

1. [System Features 6](#_bookmark20)

[Functional Requirements 6](#_bookmark21)

[Reservation/Booking 6](#_bookmark22)

[Food 6](#_bookmark23)

[Management 6](#_bookmark24)

1. [Other Nonfunctional Requirements 7](#_bookmark25)

[Performance Requirements 7](#_bookmark26)

[Safety Requirements 7](#_bookmark27)

[Security Requirements 7](#_bookmark28)

[Software Quality Attributes 7](#_bookmark29)

[Business Rules 8](#_bookmark30)

1. [Other Requirements 8](#_bookmark31)

Appendix A: Glossary 9

Appendix B: Analysis Models 10

Appendix C: To Be Determined List 10

1. **INTRODUCTION**

**1.1 Purpose**

The main purpose that banks have been serving since their inception is keeping our money safe for us. While keeping our money safe, they also let us earn a certain amount of interest on the money deposited with them. Traditional banks have been doing this, and internet banks continue the same function. The only difference is in the way the transactions are made.

We all know about internet banking but few of us actually understand about the history of internet banking and how it all came out. Knowing history of internet banking can be incredibly useful, especially since it will allow us to have more respect for the little things that we take for granted internet banking has been around for quite a few years now, but has really only become prominent over the past year or so in particular. Internet banking offers an array of different advantages to the user, including account balances and history including year-to date information, the ability to transfer money from one account to another and to payees for bill payments, check history, reorders, and stop payments, check credit card balances and statements, complete online loan applications, secure interactive messaging with staff and much more.

Internet banking basically allows you to be able to do everything that you can do it all right from the convenience of your own home.

The aim of this project is to develop a secured online banking system with the following objectives:

* Create a banking system that is easily accessible by customers from the comfort of their homes, offices etc.
* Reduce the flow of human traffic and long queues at banks.
* Reduce the time wasted in going to banks to stay on queues.
* Promote efficient and effective banking for the banks by focusing on those services that still require physical presence at the banking hall.

**1.2 Document Conventions.**

🡪HTTP : Hyper Text Transfer Protocol.

🡪HTML : Hyper Text Markup Language.

🡪TCP/IP: Transmission Control Protocol/Internet Protocol.

**1.3 Intended Audience & Reading Suggestions.**

The different types of readers are:-

(a). Customers.

🡪 Project Scope.

🡪 Security Available.

(b). Employers.

(c). Developers.

🡪 Project Scope.

🡪 Use Case Module.

(d). Project Manager.

🡪 System Features.

🡪 Hardware Requirement.

🡪 Software Requirement.

🡪 Interface Requirement.

**1.4 Definations, Acronyms & Abbreviation.**

🡪Account Details.

🡪ATM Request Form.

🡪New Account Opening Form

Abbreviation Used:-

PAN – Permanent Account Number.

DD - Demand Draft

SB - Saving Bank

CA - Current Account.

ATM – Automatic Teller Machine.

**1.5 Scope.**

The Scope of this project is limited to the activities of the operation units of the banking system which include opening of account, deposit of funds, withdrawal of funds & transfer.

🡪 Any bank can use this application to provide better service to their customers.

🡪 Customers can access his/her all accounts present in various branches of the same bank at one click.   
🡪 Bank can publish various upcoming plans for customers through this application.  
🡪 Manager can access all accounts present in the bank through this application.  
🡪 Reduction in work load of all employees will possible through this application as transaction rights are provided online to customer.  
🡪It can be extended for global communication between all banks in the world.

**1.6 References.**

🡪 References from following website:

[www.bankofbaroda.com](http://www.bankofbaroda.com)

[www.youtube.com](http://www.youtube.com)

🡪 Reference from our text book “Software Engineering” of Genius Publication.

**2. OVERALL DISCRIPTION.**

**2.1 Product Perspective.**

The client will have client interface in which he can interact with the banking system. It is a web based interface which will be the web page of the banking application. Starting a page is displayed asking the type of customer he is whether ordinary or a corporate customer. Then the page is redirected to login page where the user can enter the login details. If the login particulars are valid then the user is taken to a home page where he has the entire transaction list that he can perform with the bank. All the above activities come under the client interface. The administrator will have an administrative interface which is a GUI so that he can view the entire system. He will also have a login page where he can enter the login particulars so that he can perform all his actions. This administrative interface provides different environment such that he can maintain database & provide backups for the information in the database. He can register the users by providing them with username, password & by creating account in the database. He can view the cheque book request & perform action to issue the cheque books to the clients.

**2.2 Product Features.**

The Internet banking system consists of following modules :

1) Login Process

This module allows valid customers to access the functionalities provided by the bank.

2) Balance Enquiry

🡪This module maintains the balance details of a particular account.

3) Update Profile

🡪This module allows the customer to update profile of their account.

4) Funds Transfer

🡪This module allows the customers to transfer funds from one account to another within the same bank.

5) Change of Password

🡪This module allows customers to change their password.

6) Mini Statements

This module allows customers to view their transaction details.

2.2 User Classes & Characteristic

**Customers :** The normal users will have an account of fixed or savings and should have a minimum balance of Rs. 500. He can transfer funds to another account of the same bank & may view his monthly or annual statements.

**Industrialists, Entrepreneur, Organizations academicians:** These users will have all the three accounts & should have a minimum balance of 20,000 Rs. He can view the statements of his organization or industry.

**2.3 Operating Environment.**

**Server Side**

Hard Drive 🡪 More than 160 GB.

RAM 🡪 More than 1 GB.

Processor 🡪 Pentium 4 or Higher.

Client Side

Hard Drive 🡪 More than 160 GB.

RAM 🡪 More than 1 GB.

Processor 🡪 Pentium 4 or Higher.

**2.5 Design & Implementation Constraint.**

•

🡪 This system works only on a single server.

•

🡪 This is designed in ASP.Net.

🡪 Language used is C#.

•

🡪 Limited to HTTP/HTTPS protocols.

**2.6 User Documantation.**

A registered user can have following facilities:

🡪 Accounts and accounts status.

🡪 The balance enquiry.

🡪 The fund transfer standards.

🡪 Cheque Book Request.

🡪 Password Changing.

**2.7 Assumption & Dependencies.**

**Assumption:**

The details of customers such as username, password, account type and their corresponding authority details should be manually entered by the administrator before using this system.

•

Every user should be comfortable of working with computer and net browsing.

He should be aware of the banking system.

He must have basic knowledge of English too.

**3. EXTERNAL INTERFACE REQUIREMENT.**

**3.1 User Interface**

There are four different ways for a user to interact with the system:

**Viewers:**

Many unknown persons or un-authenticated persons visit the Bank official site via internet. They collect the information and search what are the schemes are available in the bank web page. Those viewers or visitors became the customer of the bank.

**New User:**

Who all visited that Bank webpage or heard about the bank those persons getting ready to start account in bank. They register the bank application form, submit and start account in the nearest bank.

**Existing User:**

The Existing user is the most typical user of the Online Banking system. Each Users have their own account and registered or authorized login access. The Existing user can login in online to their account perform the operation of deposit, withdrawn, transfer, balance queries and transactions. All the operation of the banking do in online it helpful for user because save time and efficient process.

**Administrator:**

Admin is master user of the system because they are main role of the system. Admin grant and maintain the database of the existing user and grant the permissions to users. It over rules all other users

**3.2 Hardware Interface**

*Client Side:*

**User on Internet** :  Web Browser, Operating System (any)

**Application Server** : WAS

**Data Base Server** : DB2

**Network** : Internet

**Development Tools** :  ASP.Net, HTML,OS(Windows).

*Server Side:*

Processor : Pentium IV or higher.

RAM : 1 GB or more.

Disk Space : More than 160 GB.

**3.3 Software Interface.**

**User on Internet** :  Web Browser, Operating System (any)

**Application Server** : WAS

**Data Base Server** : DB2

**Network** : Internet

**Development Tools** :  ASP.Net, HTML, OS(Windows),

**3.4 Communication Interface.**

APNA Bank website i:e [www.apnabank.com](http://www.apnabank.com) offers all banking facilities anywhere any time through internet facility. It also offers Mobile Banking facility which is a SMS based service.

🡪 Client on Internet will be using HTTP/HTTPS protocol.

🡪 Client on Intranet will be using TCP/IP protocol

🡪 A Web Browser such as IE 6.0 or equivalent

**4. Other Non-functional Requirement.**

**4.1 Performance Requirement.**

System can withstand even though many no. of customers request the desired service. Access is given to only valid users of bank who requires the services such as balance enquiry, update profile, funds transfer, mini statements, and request for stop payments and for cheque book

It is available during whole week for all 24 hours.

**4.2 Safety Requirement.**

By incorporating a secure database and proven DB2 UDB into the system, reliable performance and integrity of data is ensured. There must be a power backup for server system. Since the product is of 24x7 availability there should be power backup for server which provides the information . Every day the data should be backup even when the operation of an user is not successful i.e., while performing the operation power failure occurs then data should be backup.

**4.3 Security Requirement**

Sensitive data is protected from unwanted access by user’s appropriate technology and implementing strict user-access criteria.

Facility of unique user number and Password in such a way that unauthorized user cannot log in. Operational rights for each user/terminal can be defined. Thus, a user can have access to specific terminals and specific options only.

Online Banking uses the SSL (Secure Socket Layer) protocol for transferring data. SSL is encryption that creates a secure environment for the information being transferred between customer browser and Bank. Online Banking uses 128-bit digital certificate from VeriSign for encryption of the Secure Sockets Layer (SSL) session. SSL is the industry standard for encrypted communication and ensures that customer's interaction with the Bank over the Internet is secure. Secure Socket Layer (SSL) protects data in three key ways:

**Authentication:-** ensures that you are communicating with the correct server. This prevents another computer from impersonating Bank.

**Encryption :-** scrambles transferred data.

**Data integrity :-** verifies that the information sent by customer to Bank wasn't altered during the transfer. The system detects if data was added or deleted after customer sent the message. If any tampering has occurred, the connection is dropped

**4.4 Software Quality Attributes.**

**4.4.1. Usability.**

The users of the system are members and the administrators who maintain the system. The members are assumed to have basic knowledge of the computers and Internet browsing. The administrators of the system to have more knowledge of the internals of the system and is able to rectify the small problems that may arise due to disk crashes, power failures and other catastrophes to maintain the system. The proper  user interface, user’s manual, online  help and

the guide to use and maintain the system must be sufficient to educate the users on how to use the system without any problems.

**4.4.2 Reliability.**

The system is safety critical. If it moves out of normal operation mode, the requirement to drop to the next lower floor and open its doors is given priority. This emergency behaviour shall not occur without reason.

The system has to be very reliable due to the importance of data and the damages incorrect or incomplete data can do.

**4.4.3 Availability.**

When in normal operating conditions, request by a user for an servicer shall be handled within 1second. Immediate feedback of the systems activities shall be communicated to the user by link  page clicked. At peek system load, individual users at either the server in the security office, at the links or inside the banking system shall not experience any delay in the service response to their commands longer than 1 second.

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

**4.4.4 Security.**

There shall be no security mechanisms in place to keep unwanted users out of the system. However, all users of the system shall not be able to perform actions or request actions from the Banking system, which will cause harm to any person or damage to the system or its environment.

**4.4.5 Maintainability.**

There shall be design documents describing the internal works of the software. There shall bean access on the control panel and servers for the purpose of upgrading the software or flashing any firmware.

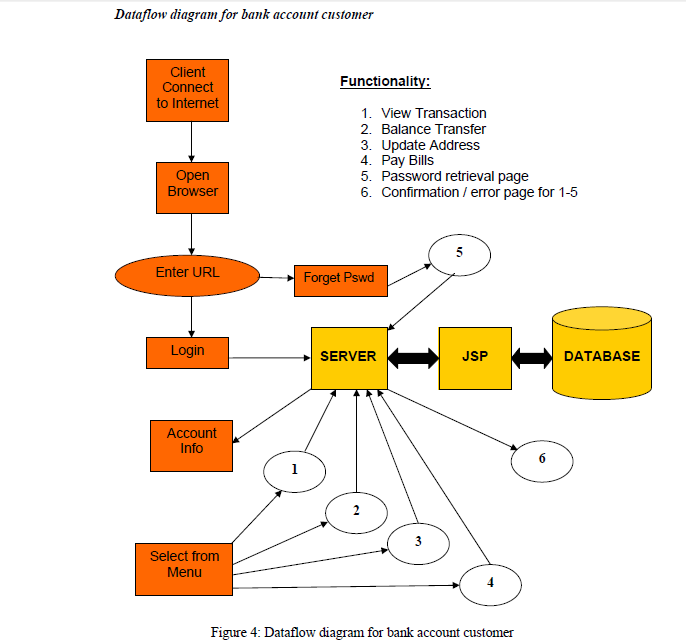
**4.4.6 Portability.**

There are no portability requirements. Requirement Organization: All requirements shall be organized according to object. First general requirements for all service types shall be described. Following are sections for each service type and their special requirements. Last are requirements related to other objects like the users view pages and any other.

**5. System Design.**

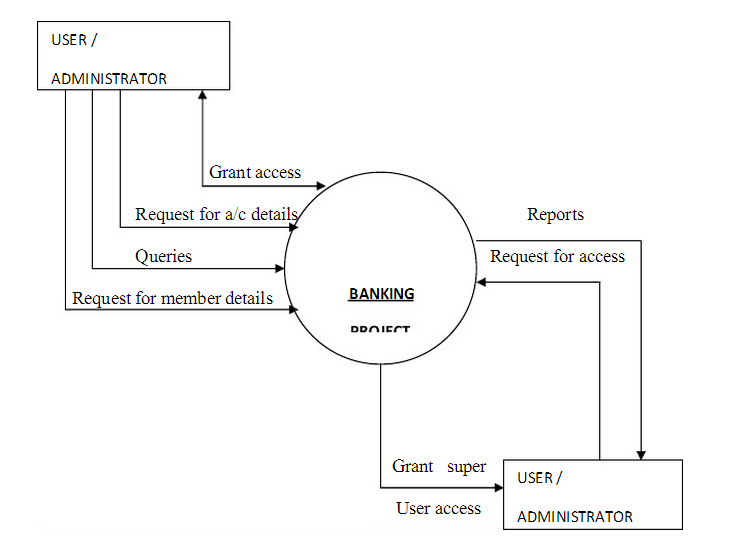
**5.1 Process Description(Data Flow Diagram).**

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system modeling its *process* aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated.



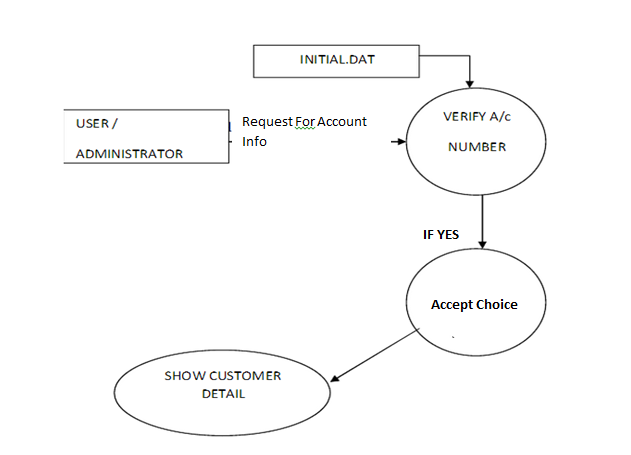
ASP

“COMMON DFD OF BANK SYSTEM

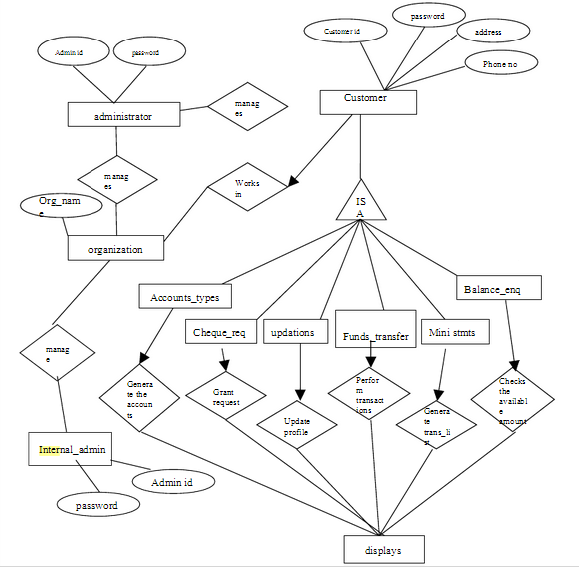


BANK SYSTEM

**“ZERO LEVEL DFD”**

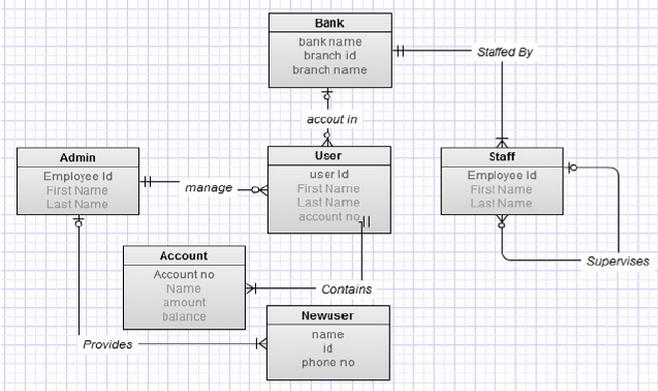


**“1ST LEVEL DFD”**

****

**5.2 Relation Between Entities(Entity Relationship Diagram).**

An entity-relationship diagram is a data modelling technique that creates a graphical representation of the entities, and the relationships between entities, within an information system.



**“Entity Relationship Diagram”**

**6. SNAPSHOTS**

**6.1 HOME :-**

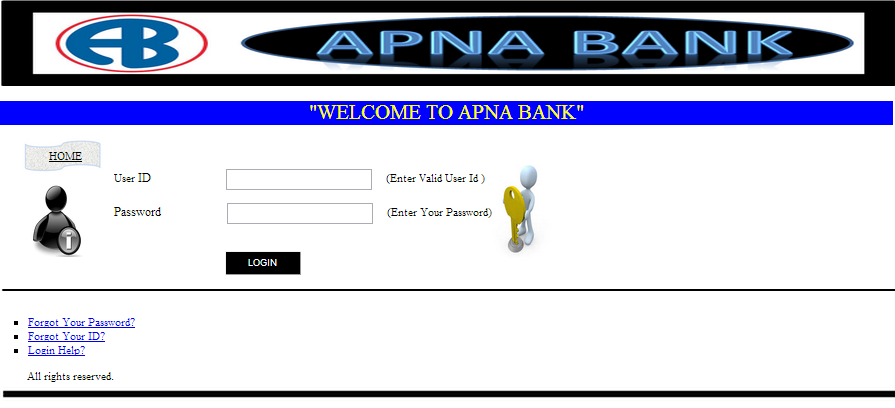
This page tell u about the bank. It tells about the various facilities available on this bank website so that user can choose his desired requirement & can go through it for results.



**6.2 LOGIN**

This page provides a form to login to the Internet banking website.

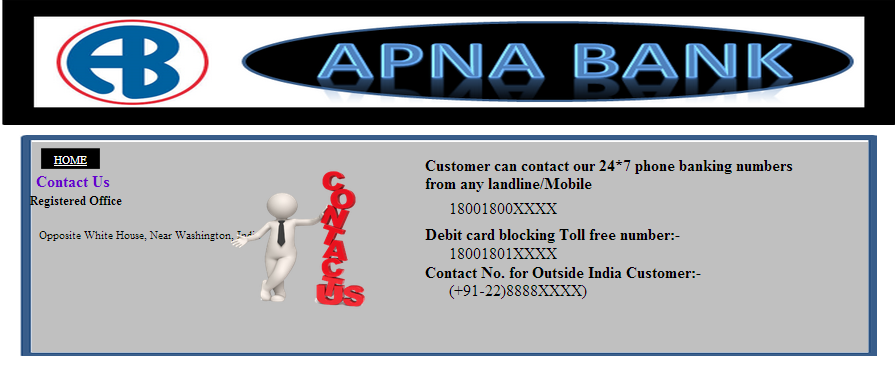
A registered user can login with a valid User ID and password.



**6.3 CONTACT US:**

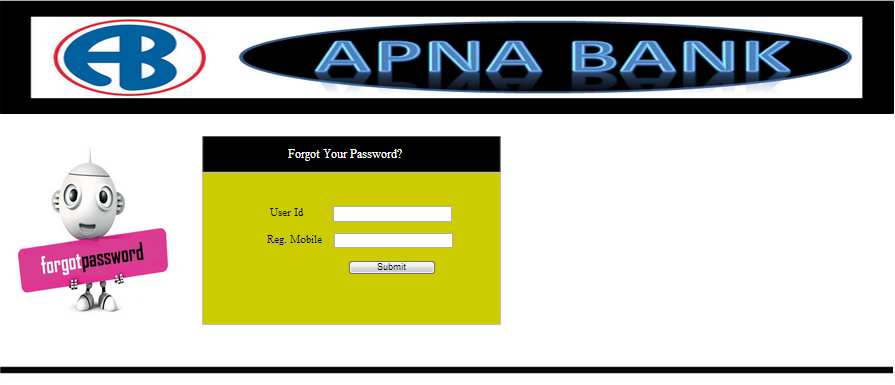
This page let u know about the contact details of the bank.

It let u know about the bank address & also about the banking customer care numbers.

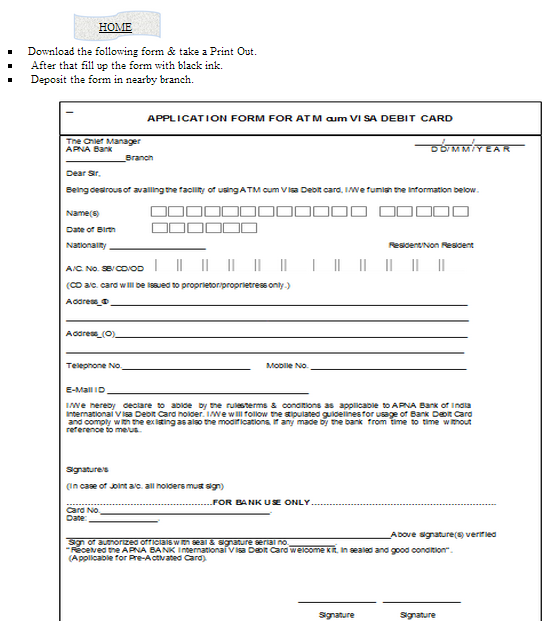


**6.4 FORGOT PASSWORD :**

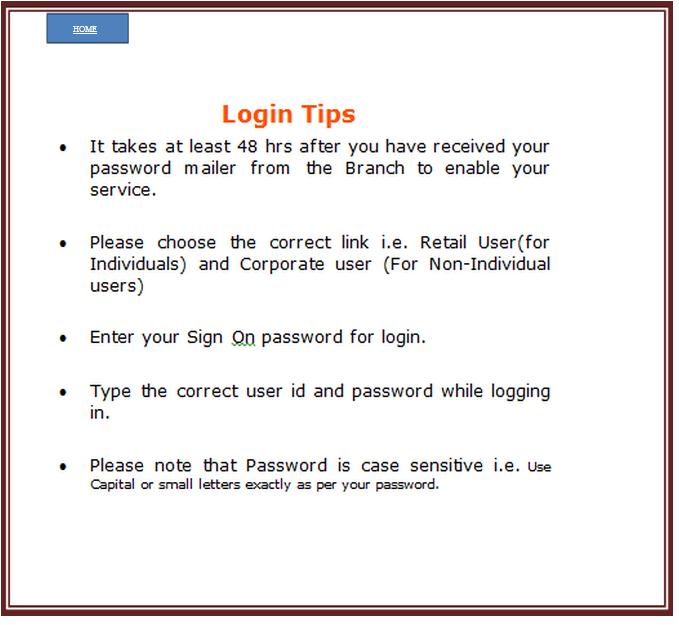
If you have forgot the password of our net banking account, then this page will help you to recover our password using your registered mobile number.



**6.5 ATM application page:**

If you want your own atm card then this page is required to be filled.

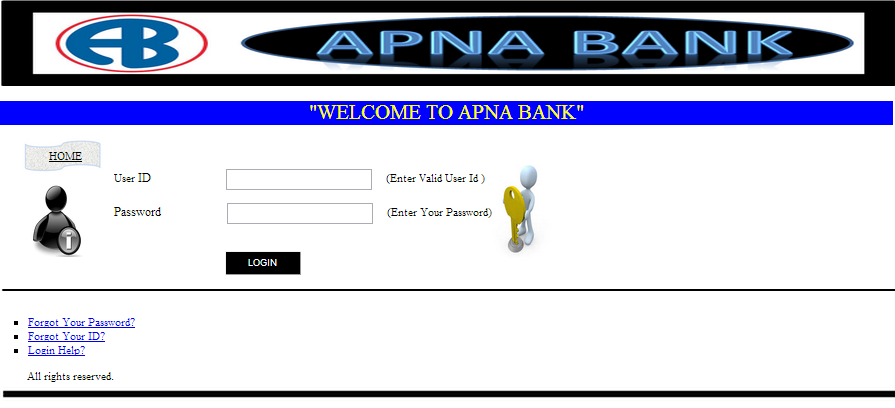
**6.6 Help Page:**

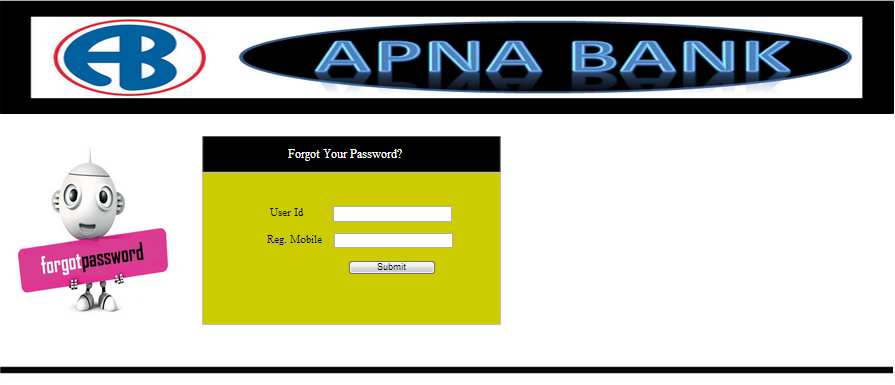
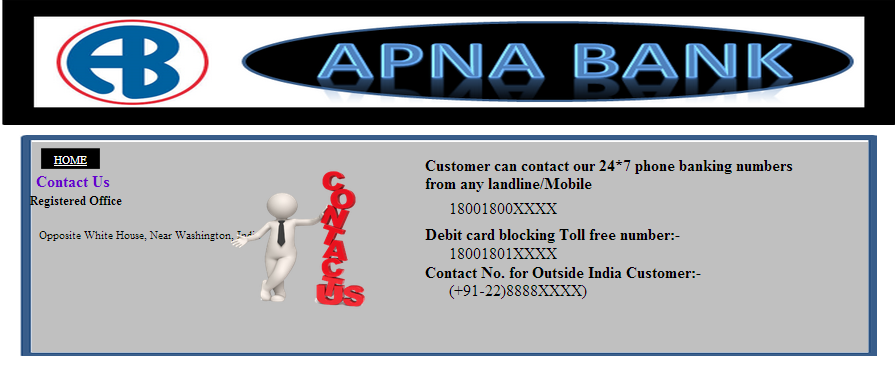
If any customer is having any problem with his/her account login then this page let them help about it.

**6.7 TRANSACTION** :

This page tells the customer about their transaction, profile, account details etc.



**6.8 All Snapshots**





**7.**

**CODING.**

**🡪 Coding for login page.**

using System;

using System.Data;

using System.Configuration;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

public partial class \_Default : System.Web.UI.Page

{

conclass c1 = new conclass();

protected void Page\_Load(object sender, EventArgs e)

{

lbl\_msg.Text = "";

}

protected void btn\_submit\_Click(object sender, EventArgs e)

{

String str;

str = " SELECT COUNT(\*) FROM dbo.Admin\_Login WHERE ";

str += " LoginID= '" + txt\_login.Text + "' AND";

str += " Passwd= '" + txt\_passwd.Text + "'";

if (Convert.ToInt32(c1.selectScalar(str)) > 0)

{

Session.Add("LoginID", txt\_login.Text);

Response.Redirect("~/administrator/homeadmin.aspx");

}

else

{

lbl\_msg.Text = "Invalid User.... ";

}

}

protected void WUC\_topmenu1\_Load(object sender, EventArgs e)

{

}

protected void txt\_login\_DataBinding(object sender, EventArgs e)

{

}

protected void WUC\_footer1\_Load(object sender, EventArgs e)

{

}

}

🡪Coding for Database Connectivity.

using System;

using System.Data;

using System.Configuration;

using System.Web;

using System.Data.SqlClient;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

/// <summary>

/// Summary description for conclass

/// </summary>

public class conclass

{

String conStr = "SERVER=AKSHAY;DATABASE=ONLINE BANKING SYSTEM; trusted\_connection=yes; ";

public conclass()

{

//

// TODO: Add constructor logic here

//

}

SqlConnection con = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataAdapter adpt = new SqlDataAdapter();

public DataSet selectDS(String sql)

{

DataSet ds = new DataSet();

con.ConnectionString = conStr;

con.Open();

cmd.CommandText = sql;

cmd.Connection = con;

adpt.SelectCommand = cmd;

adpt.Fill(ds);

con.Close();

return ds;

}

public SqlDataReader selectRD(String sql)

{

SqlDataReader RD;

con.ConnectionString = conStr;

con.Open();

cmd.CommandText = sql;

cmd.Connection = con;

RD = cmd.ExecuteReader();

return RD;

}

public String selectScalar(String str)

{

string v;

con.ConnectionString = conStr;

con.Open();

cmd.CommandText = str;

cmd.Connection = con;

v = Convert.ToString(cmd.ExecuteScalar());

con.Close();

return v;

}

public void executeQry(String sql)

{

con.ConnectionString = conStr;

con.Open();

cmd.CommandText = sql;

cmd.Connection = con;

cmd.ExecuteNonQuery();

con.Close();

}

}

**8. Software Testing.**

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate.

Software testing, depending on the testing method employed, can be implemented at any time in the development process, however the most test effort is employed after the requirements have been defined and coding process has been completed.

**Testing Methods:-**

* Black Box Testing.
* White Box Testing.

**Black Box Testing.**

Black box testing treats the software as a "blackbox," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

**White Box Testing.**

White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these)

Types of white box testing:-

**🡪 Api testing -**

Testing of the application using Public and Private APIs.

Code coverage - creating tests to satisfy some criteria of code coverage. For example, the test designer can create tests to cause all statements in the program to be executed at least once.

**🡪 Fault injection methods.**

**🡪 Mutation testing methods.**

**🡪 Static testing** –

White box testing includes all static testing.

**9. ADVANTAGES & DISADVANTAGES.**

**Advantages**

🡪Opening & closing of accountes

🡪Make the payments  of  merchandise  transaction

through Debit & Credit cards.

🡪 It gives reliefs to their customer from carrying heavy cash.

🡪 Enables prompt & speedy operation to clients.

🡪 It saves lot of time to their customers &convenient to access.

🡪 It brings efficiency in CRM(Customer relationship management).

**Disadvantages**

🡪 Customer may have to face risky transaction & fraud.

🡪 Failure of power supply cause to break down of system.

🡪 Loss of heavy income at times of settlement of higher magnitude.

🡪 Cost involved in trainning staff may not be profitable

specially in times of attrition.

🡪 Sometimes may leads to threats from outside.

**10. CONCLUSION.**

Thus reaching to the conclusion of our project we observe that Traditional  banks offer many services to their customers, including accepting customer’s   money, deposits, providing

various  banking  services to  customers,  and making

loans  to  individuals  and  companies.  Compared  with

traditional channels of offering banking services through physical branches, e-banking uses the Internet to deliver traditional banking services to their customers,such as opening accounts, transferring funds, and electronic bill payment.

E-banking services are delivered to customers through the Internet and the web using Hypertext Markup Language (HTML). In order to use the e-banking services, customers  need

Internet access and web browser software.

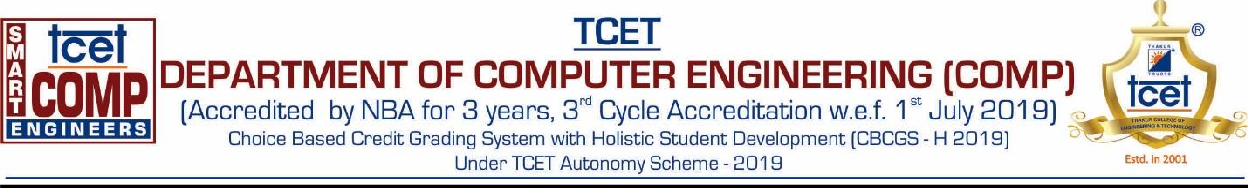
Multimedia  information in HTML format from

online banks can be displayed in web browsers. The heart of the e-banking application is the computer system,

which includes web servers, database management

systems, and web application programs that can generate dynamic HTML pages.

The range of e-banking services is likely to increase in the future. Some banks plan to introduce electronic money and electronic checks. Electronic money can be stored in computers or smart cards and consumers can use the electronic money to purchase small value items over the Internet.



**EXPERIMENT NO:02**

**Aim:** Sketch a DFD Diagram up to 2 level for our project

**Theory:** A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

Data flow diagrams were popularized in the late 1970s, arising from the book *Structured Design*, by computing pioneers Ed Yourdon and Larry Constantine. They based it on the “data flow graph” computation models by David Martin and Gerald Estrin. The structured design concept took off in the software engineering field, and the DFD method took off with it. It became more popular in business circles, as it was applied to business analysis, than in academic circles.

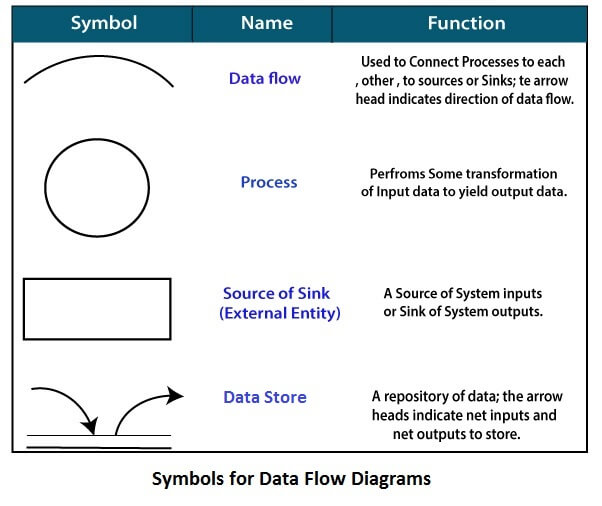
Also contributing were two related concepts:

* Object Oriented Analysis and Design (OOAD), put forth by Yourdon and Peter Coad to analyze and design an application or system.
* Structured Systems Analysis and Design Method (SSADM), a waterfall method to analyze and design information systems. This rigorous documentation approach contrasts with modern agile approaches such as Scrum and Dynamic Systems Development Method (DSDM.)

Three other experts contributing to this rise in DFD methodology were Tom DeMarco, Chris Gane and Trish Sarson. They teamed up in different combinations to be the main definers of the symbols and notations used for a data flow diagram.

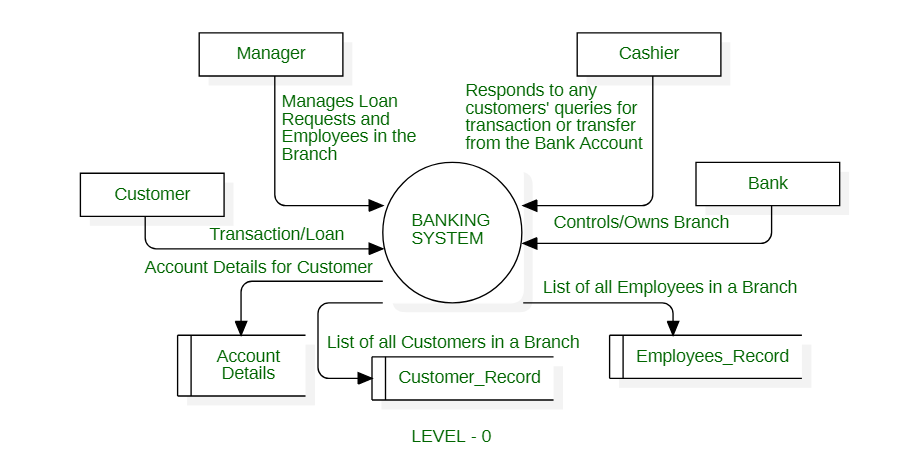
Using any convention’s DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

1. **External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
2. **Process:**any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”
3. **Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”
4. **Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

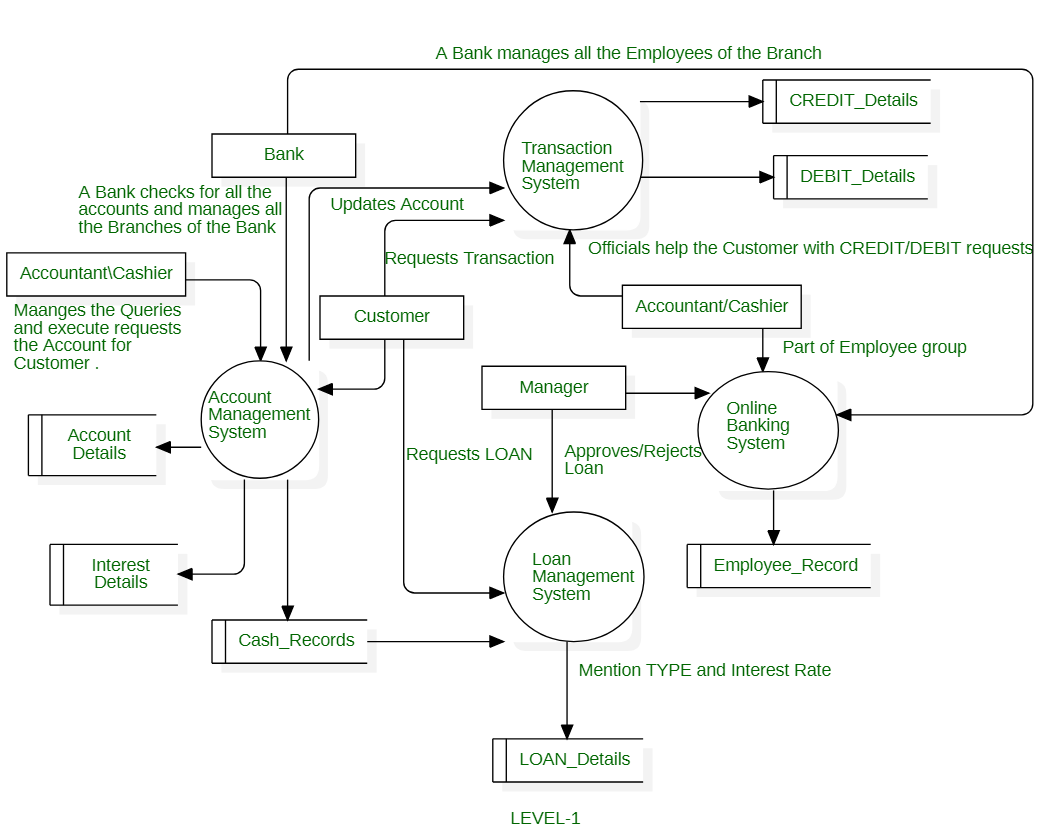


* **DFD Diagram for Bank Management System:**

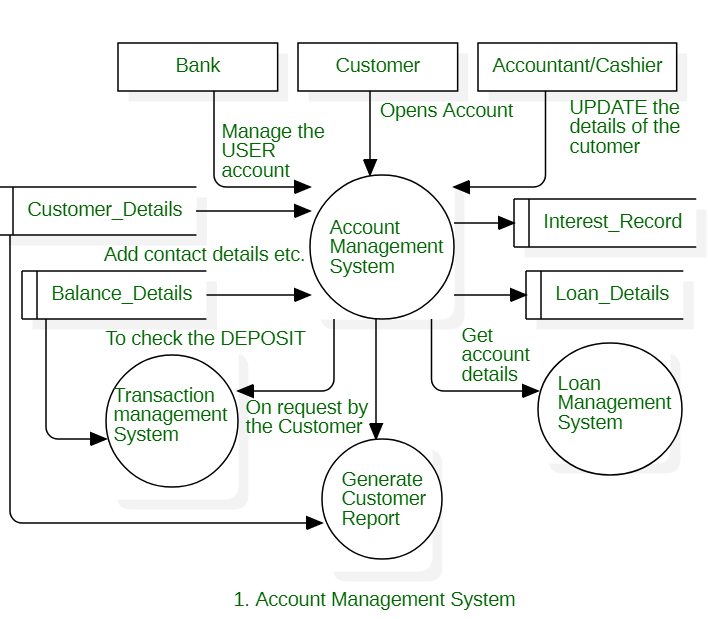
**DFD Diagram for level 0**

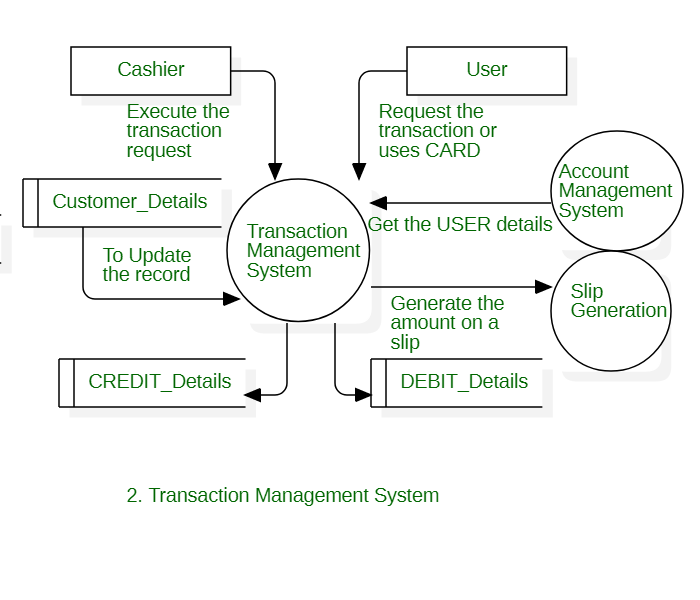
****

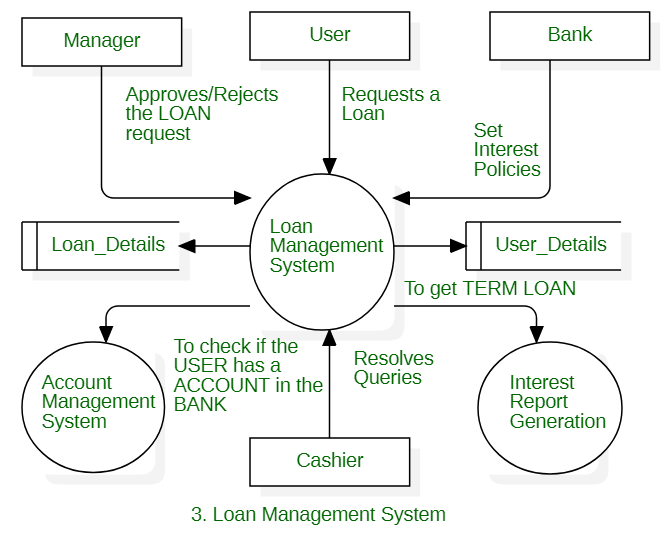
**DFD Diagram for level 1**

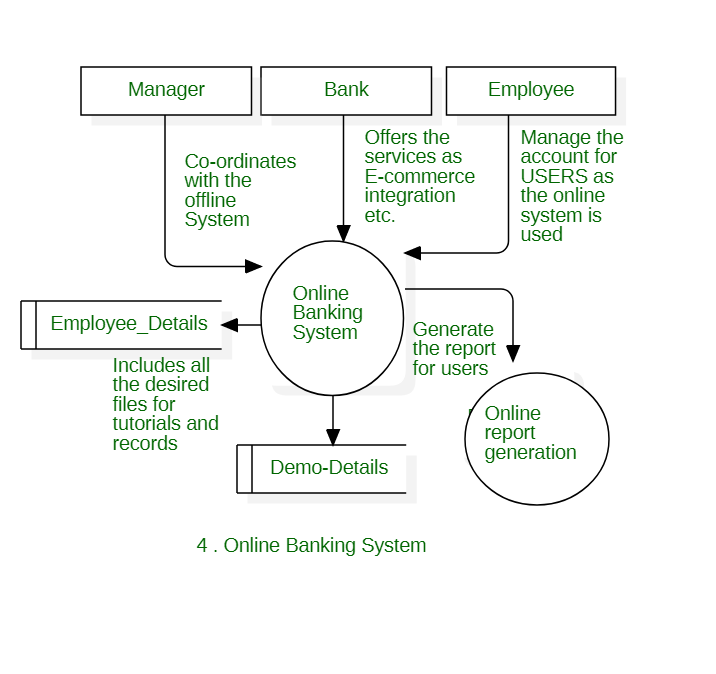
****

**DFD Diagram for level 2**

****

****

****

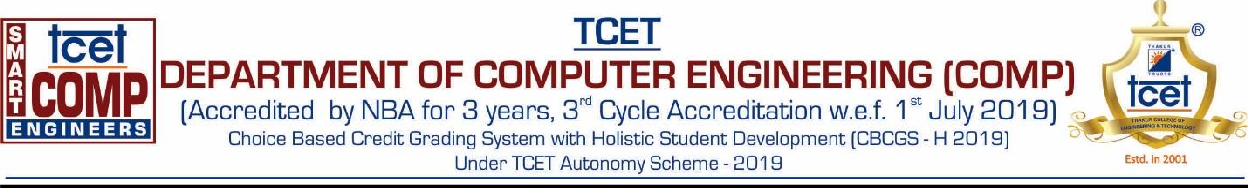
****

**Conclusion:**

Data Flow Diagrams are highly useful tools to maintain these aspects of the operation. They provide critical insights into the systems and ways the information passes through it. DFD helps structure every element of the system, keep them logically intact and interconnected. Thus, we have successfully sketch the DFD Diagrams for our project.

ForFacultyUse

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CorrectionParameters** | **FormativeAssessment[40%]** | **Timely completionofPractical[40%]** | **Attendance /LearningAttitude[20%]** |  |
| **MarksObtained** |  |  |  |



**EXPERIMENT NO:03**

**Aim:** Sketch the UML Use Case Diagram for the project

**Tools:** IBM Rational Rose

**Theory:** In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

* Scenarios in which your system or application interacts with people, organizations, or external systems
* Goals that your system or application helps those entities (known as actors) achieve
* The scope of your system

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

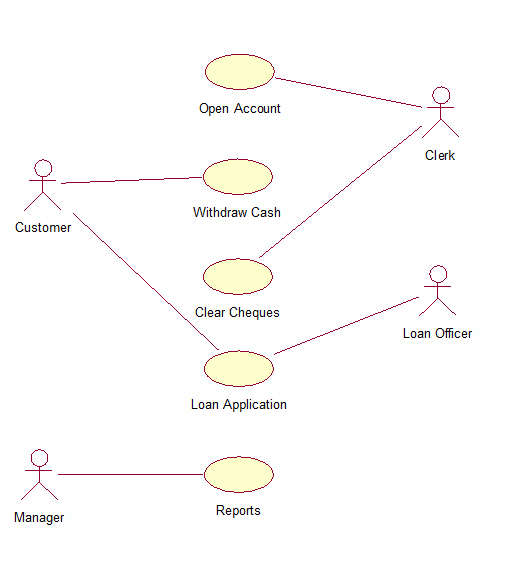
UML is the modelling toolkit that you can use to build your diagrams. Use cases are represented with a labelled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

* Representing the goals of system-user interactions
* Defining and organizing functional requirements in a system
* Specifying the context and requirements of a system
* Modelling the basic flow of events in a use case
* **Use case diagram components:**
* **Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
* **System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
* **Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.
* **Use case diagram symbols and notation**

The notation for a use case diagram is pretty straightforward and doesn't involve as many types of symbols as other UML diagrams. You can use this guide to learn how to draw a use case diagram if you need a refresher. Here are all the shapes you will be able to find in Lucid chart:

* **Use cases:** Horizontally shaped ovals that represent the different uses that a user might have.
* **Actors:** Stick figures that represent the people actually employing the use cases.
* **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
* **System boundary boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
* **Packages:** A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.
* **UML Use Case Diagram for our Project:**

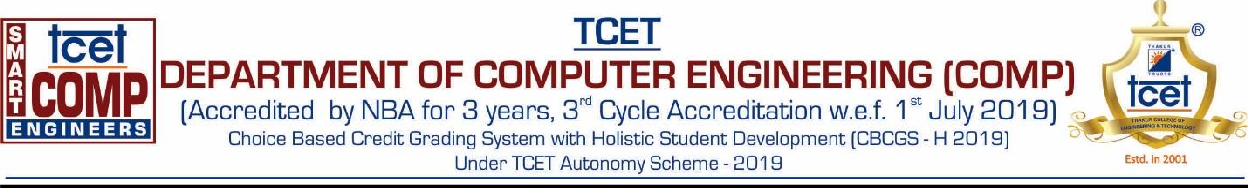


**Conclusion:**

A use case diagram is drawn to capture the functional requirement insoftware development. It also shows the interactions between requirements are actors. But,use case diagram never describes how they are implemented. It only shows the flow of theproject.

ForFacultyUse

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CorrectionParameters** | **FormativeAssessment[40%]** | **Timely completionofPractical[40%]** | **Attendance /LearningAttitude[20%]** |  |
| **MarksObtained** |  |  |  |



**EXPERIMENT NO:04**

**Aim:** Sketch Activity Diagram & State Transition Diagram for our Project

**Tools:** IBM Rational Rose

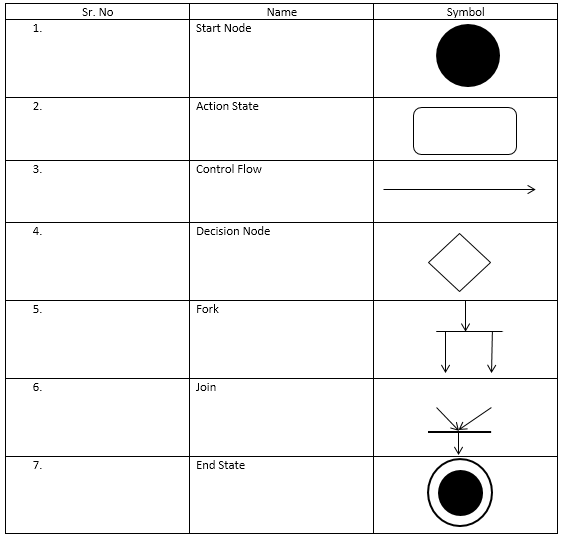
**Theory:** The Unified Modelling Language includes several subsets of diagrams, including structure diagrams, interaction diagrams, and behaviour diagrams. Activity diagrams, along with usecase and state machine diagrams, are considered behaviour diagrams because they describe what must happen in the system being modelled.

Stakeholders have many issues to manage, so it's important to communicate with clarity and brevity. Activity diagrams help people on the business and development sides of an organization come together to understand the same process and behaviour. You'll use a set of specialized symbols—including those used for starting, ending, merging, or receiving steps in the flow—to make an activity diagram, which we’ll cover in more depth within this activity diagram guide.

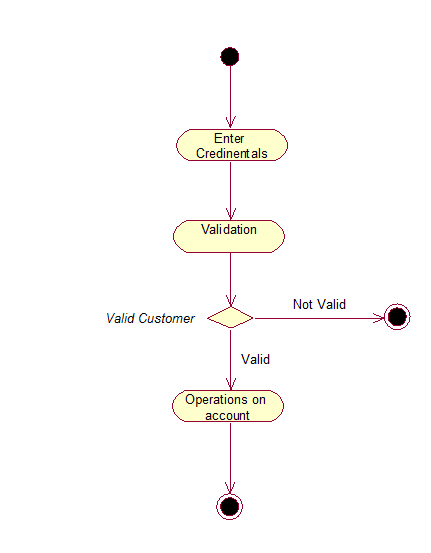
Activity diagrams present a number of benefits to users. Consider creating an activity diagram to:

* Demonstrate the logic of an algorithm.
* Describe the steps performed in a UML use case.
* Illustrate a business process or workflow between users and the system.
* Simplify and improve any process by clarifying complicated use cases.
* Model software architecture elements, such as method, function, and operation.
* **Basic components of an activity diagram**
* **Action:**A step in the activity wherein the users or software perform a given task. In Rational Rose, actions are symbolized with round-edged rectangles.
* **Decision node:** A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.
* **Control flows:** Another name for the connectors that show the flow between steps in the diagram.
* **Start node:** Symbolizes the beginning of the activity. The start node is represented by a black circle.
* **End node:** Represents the final step in the activity. The end node is represented by an outlined black circle.
* **Activity diagram symbols**

These activity diagram shapes and symbols are some of the most common types you'll find in UML diagrams.



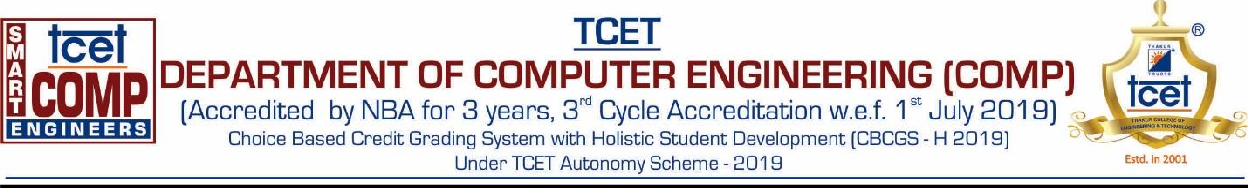
* **Activity Diagram for Bank Management System**



**Conclusion:**Activity diagrams are fairly easy to get the hang of, and will be useful for most projects because they "simply and quite plainly show how things work. Unlike many diagramming techniques, activity diagrams also enable the depiction of multiple choices and actors within a workflow, and they are easy

For Faculty Use

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CorrectionParameters** | **FormativeAssessment[40%]** | **Timely completionofPractical[40%]** | **Attendance /LearningAttitude[20%]** |  |
| **MarksObtained** |  |  |  |



**EXPERIMENT NO:05**

**Aim:** Sketch State Transition Diagram for our Project.

**Tools:** IBM Rational Rose

**Theory**: A state machine is any device that stores the status of an object at a given time and can change status or cause other actions based on the input it receives. States refer to the different combinations of information that an object can hold, not how the object behaves. In order to understand the different states of an object, you might want to visualize all of the possible states and show how an object gets to each state, and you can do so with a UML state diagram.

Each state diagram typically begins with a dark circle that indicates the initial state and ends with a bordered circle that denotes the final state. However, despite having clear start and end points, state diagrams are not necessarily the best tool for capturing an overall progression of events. Rather, they illustrate specific kinds of behaviour—in particular, shifts from one state to another.State diagrams mainly depict states and transitions. States are represented with rectangles with rounded corners that are labelled with the name of the state. Transitions are marked with arrows that flow from one state to another, showing how the states change. Below, you can see both these elements at work in a basic diagram for student life.

* **State diagram applications**

Like most UML diagrams, state diagrams have several uses. The main applications are as follows:

* Depicting event-driven objects in a reactive system.
* Illustrating use case scenarios in a business context.
* Describing how an object moves through various states within its lifetime.
* Showing the overall behaviour of a state machine or the behaviour of a related set of state machines.
* **State diagram symbols and components**
* **Composite state**

A state that has substates nested into it. See the university state diagram example below. “Enrollment” is the composite state in this example because it encompasses various substates in the enrolment process.

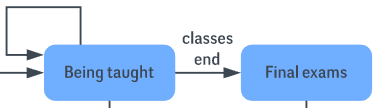
* **Choice pseudo state**

A diamond symbol that indicates a dynamic condition with branched potential results.



* **Event**

An instance that triggers a transition, labeled above the applicable transition arrow. In this case, “classes end” is the event that triggers the end of the “Being taught” state and the beginning of the “Final exams” state.



* **Exit point**

The point at which an object escapes the composite state or state machine, denoted by a circle with an X through it. The exit point is typically used if the process is not completed but has to be escaped for some error or other issue.

State Diagram Symbols - Exit Point

* **First state**

A marker for the first state in the process, shown by a dark circle with a transition arrow.



* **Guard**

A Boolean condition that allows or stops a transition, written above the transition arrow.

* **State**

A rectangle with rounded corners that indicates the current nature of an object.



* **Substate**

A state contained within a composite state's region. In the university state machine diagram found below, “Open for enrollment” is a substate in the larger “Enrollment” composite state.

* **Terminator**

A circle with a dot in it that indicates that a process is terminated.



* **Transition**

An arrow running from one state to another that indicates a changing state.

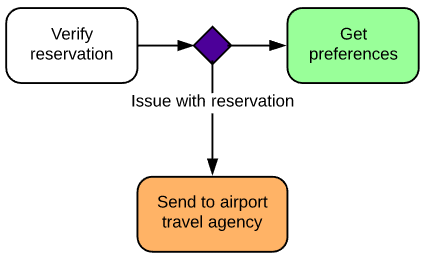


* **Transitional behaviour**

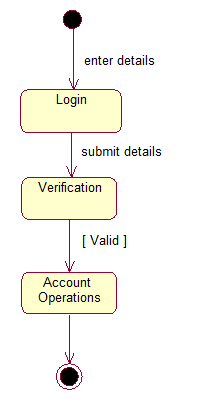
A behaviour that results when a state transitions, written above the transition arrow.

* **Trigger**

A type of message that actively moves an object from state to state, written above the transition arrow. In this example, “Issue with reservation” is the trigger that would send the person to the airport travel agency instead of the next step in the process.



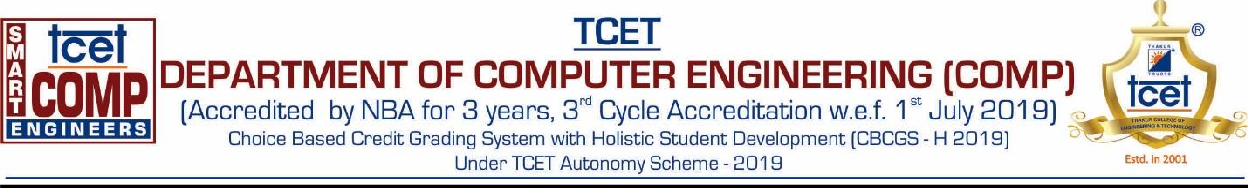
* **State Transition Diagram for Bank Management System**



**Conclusion:** A state transition diagram is used to represent a finite state machine. These are used to model objects which have a finite number of possible states and whose interaction with the outside world can be described by its state changes in response to a finite number of events. Thus, we have successfully understood and sketch the state transition diagram for our project.

For Faculty Use

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CorrectionParameters** | **FormativeAssessment[40%]** | **Timely completionofPractical[40%]** | **Attendance /LearningAttitude[20%]** |  |
| **MarksObtained** |  |  |  |



**EXPERIMENT NO:06**

**Aim:** Sketch Sequence and Collaboration Diagram for Our Project

**Tools:** IBM Rational Rose

**Theory:**

Sequence diagrams are a popular dynamic modelling solution in UML because they specifically focus on *lifelines*, or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends. Along with our UML diagramming tool, use this guide to learn everything there is to know about sequence diagrams in UML.To understand what a sequence diagram is, it's important to know the role of the Unified Modelling Language, better known as UML. UML is a modelling toolkit that guides the creation and notation of many types of diagrams, including behavior diagrams, interaction diagrams, and structure diagrams.A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.  
Note that there are two types of sequence diagrams: UML diagrams and code-based diagrams. The latter is sourced from programming code and will not be covered in this guide. Lucid chart’s UML diagramming software is equipped with all the shapes and features you will need to model both.

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

* Represent the details of a UML use case.
* Model the logic of a sophisticated procedure, function, or operation.
* See how objects and components interact with each other to complete a process.
* Plan and understand the detailed functionality of an existing or future scenario.

**Use cases for sequence diagrams**

The following scenarios are ideal for using a sequence diagram:

* Usage scenario: A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.
* Method logic: Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to explore the logic of any function, procedure, or complex process.
* Service logic: If you consider a service to be a high-level method used by different clients, a sequence diagram is an ideal way to map that out.
* Sequence diagram Visio - Any sequence diagram that you create with Visio can also be uploaded into IBM Rational Rose. IBM Rational Rose supports.vs. and .vex file import and is a great Microsoft Visio alternative. Almost all of the images you see in the UML section of this site were generated using IBM Rational Rose.

**Common message symbols**

Use the following arrows and message symbols to show how information is transmitted between objects. These symbols may reflect the start and execution of an operation or the sending and reception of a signal.

| Symbol | Name | Description |
| --- | --- | --- |
|  | Synchronous message symbol | Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply. |
|  | Asynchronous message symbol | Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram. |
|  | Asynchronous return message symbol | Represented by a dashed line with a lined arrowhead. |
|  | Asynchronous create message symbol | Represented by a dashed line with a lined arrowhead. This message creates a new object. |
|  | Reply message symbol | Represented by a dashed line with a lined arrowhead, these messages are replies to calls. |
|  | Delete message symbol | Represented by a solid line with a solid arrowhead, followed by an X. This message destroys an object. |

**Collaboration Diagram:**

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.

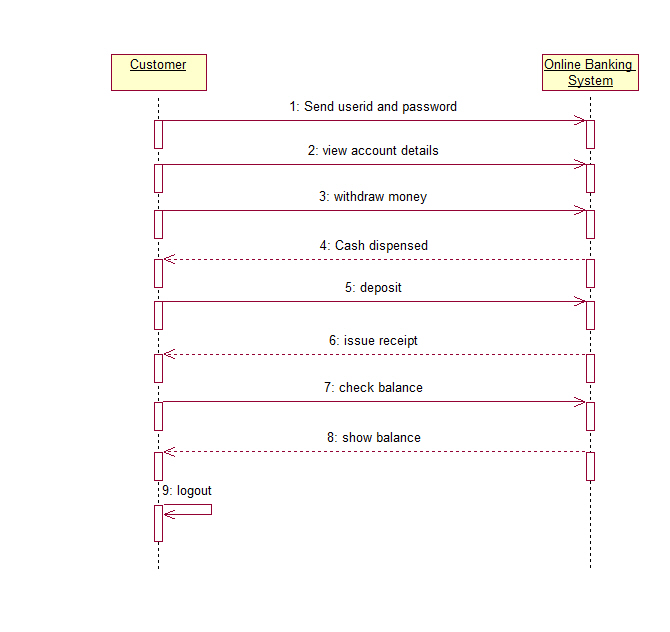
**Notations of a collaboration diagram**

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. The four major components of a collaboration diagram are:

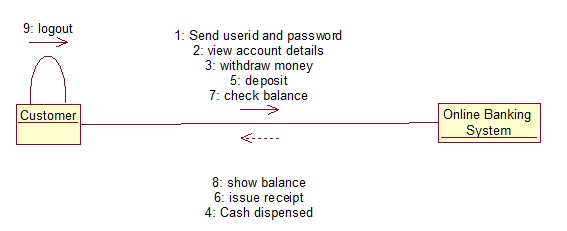
1. Objects- Objects are shown as rectangles with naming labels inside. The naming label follows the convention of object name: class name. If an object has a property or state that specifically influences the collaboration, this should also be noted.
2. Actors- Actors are instances that invoke the interaction in the diagram. Each actor has a name and a role, with one actor initiating the entire use case.
3. Links- Links connect objects with actors and are depicted using a solid line between two elements. Each link is an instance where messages can be sent.
4. messages- Messages between objects are shown as a labelled arrow placed near a link. These messages are communications between objects that convey information about the activity and can include the sequence number.

The most important objects are placed in the center of the diagram, with all other participating objects branching off. After all objects are placed, links and messages should be added in between.

* **Sequence Diagram for Bank Management System**

****

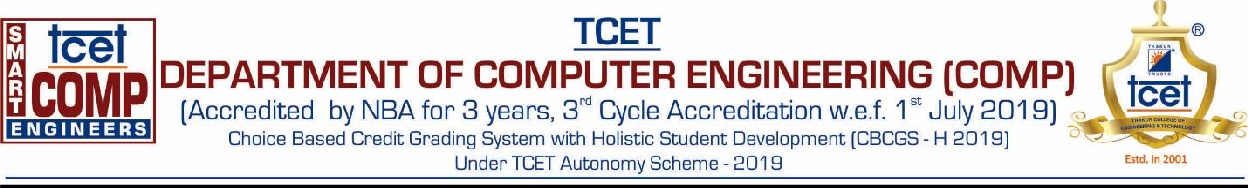
* **Collaboration Diagram for Bank Management System**

****

* **Conclusion:** Thus, we have successfully understood Sequence and Collaboration Diagram and how to sketch this diagram for our Project.

For Faculty Use

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CorrectionParameters** | **FormativeAssessment[40%]** | **Timely completionofPractical[40%]** | **Attendance /LearningAttitude[20%]** |  |
| **MarksObtained** |  |  |  |



**Experiment No:07**

**Aim:** Prepare a timeline Chart for our project

**Tools:** Microsoft Excel

**Theory:**

A timeline chart is a visual rendition of a series of events. It can be created as a chart or a graph. Timeline charts can be created for anything that occurred over a period of time. You might see a timeline chart for World War II or major events of the 20th century. As a business owner, you might have a timeline chart of new product development or a sales campaign.

A timeline chart makes it easier to conceptualize a process or a sequence of events. It can make it easier for you to understand the intricacies of a project or why something seems to take so long to accomplish

* **Different Kinds of Timelines**

**Gantt Chart:**

A Gantt chart is a fantastic organizational tool and most commonly used for project management. It will feature variously sized bars spread across a timeline, each bar representing the start date, duration and end date of an activity. Once studying the chart, you will be able to see the whole sum of the projects within its time frame. Where do activities overlap? How long will the whole project take? Remember our Gantt charts can be easily updated if your running off schedule or god forbid ahead of time! Here is an example of a Gantt Chart in the form of a housing refurbishment schedule.

**Standard Timeline Chart:**

What’s a timeline without a nice simple one!? Rather than bars graphically showing event lengths like in a Gantt Chart, a standard timeline chart shows pinpointed single events with accompanying text and/or imagery. These are classically found in History Classrooms and Museums but actually have a wind range of uses. Here is the timeline of the short but beautiful life of a Monarch Butterfly.

**Time Series Graph:**

A time series graph uses an X and Y axis to illustrate a quantity/measurement change throughout time. In most cases the horizontal axis (X) of the chart is used to mark increments in time and the vertical axis (Y) pinpoints values of the variable which is being measured. When the values are connected in chronological order by a line, it creates a series of peaks and troughs. These peaks and troughs can tell a story of progression or decline and are useful in visualizing data to create reports and forecasts. They are quite often created from financial and scientific data.

* **How to create and use timeline charts for project management**

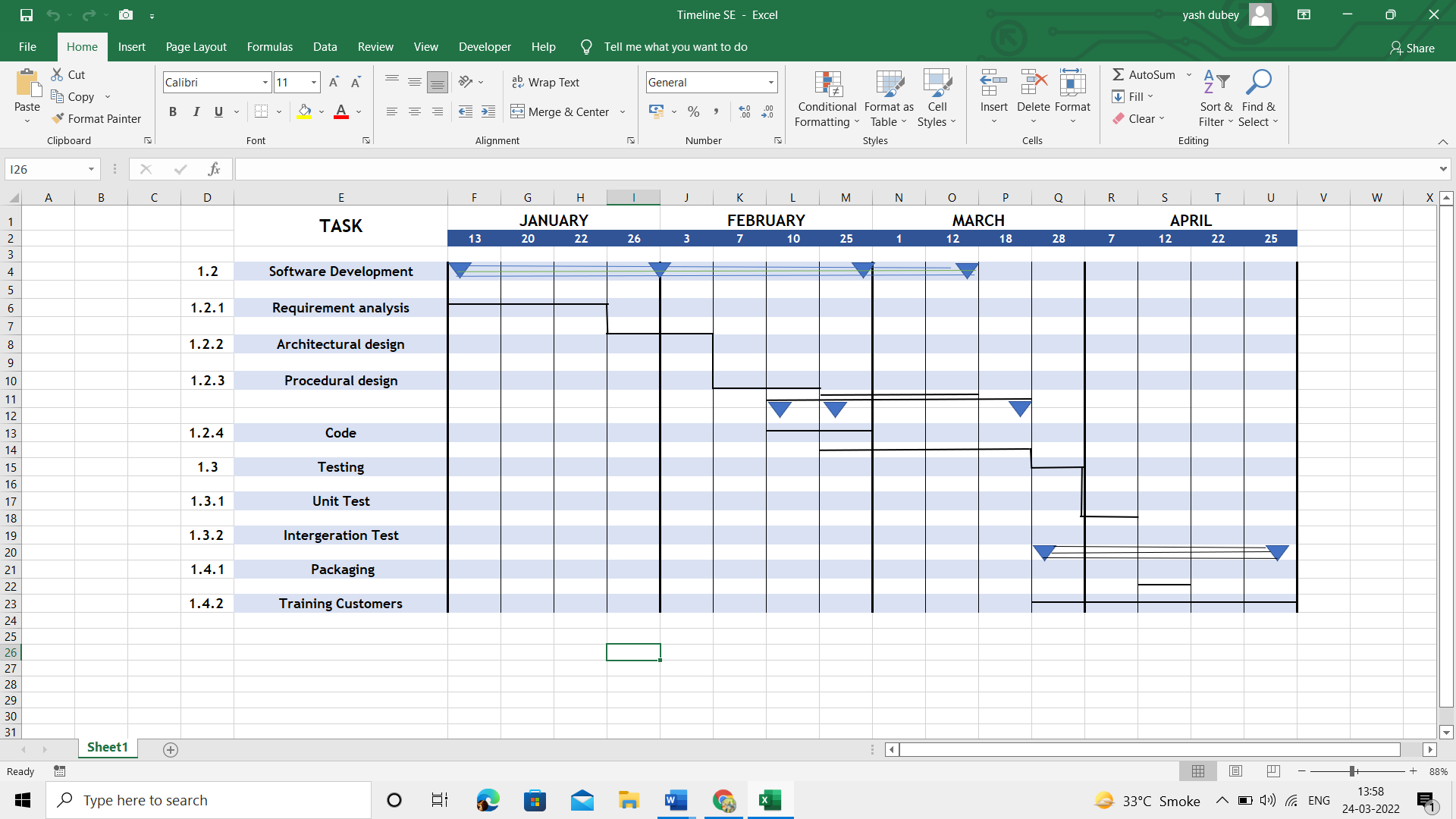
**Step 1:** Create your project.

**Step 2:** Add tasks to your project

**Step 3:** Schedule your tasks

**Step 4:** Create task dependencies

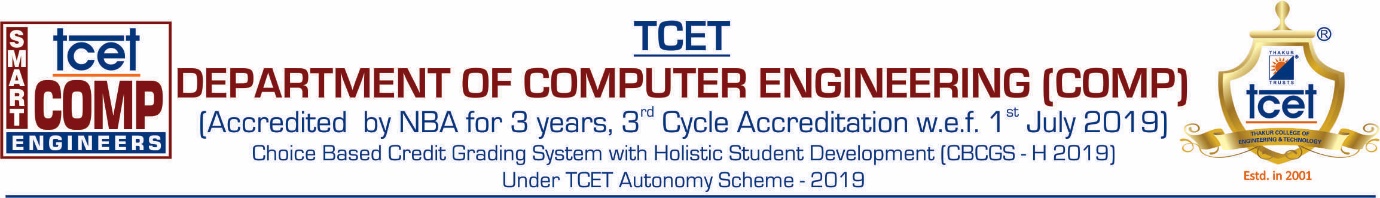
**Timeline Chart for our project**



**Conclusion:** Thus, we have successfully prepared a Timeline Chart for Bank Management System System.

For Faculty Use

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CorrectionParameters** | **FormativeAssessment[40%]** | **Timely completionofPractical[40%]** | **Attendance /LearningAttitude[20%]** |  |
| **MarksObtained** |  |  |  |

****

**EXPERIMENT NO:08**

**Aim:** Change specification and use any SCM Tool to make different versions for the project.

**Tools:**  GitHub

**Theory:**

Software configuration management: The traditional software configuration management (SCM) process is looked upon by practitioners as the best solution to handling changes in software projects. It identifies the functional and physical attributes of software at various points in time, and performs systematic control of changes to the identified attributes for the purpose of maintaining software integrity and traceability throughout the software development life cycle.

Software configuration management is a part of software engineering, which focuses mainly on maintaining, tracking and controlling the changes done to the software configuration items.

Configuration management is present in all phase of software development. The configuration items can be all the objects which come as an output of the development process e.g. coding phase produces source code, exes and obj files. The various configuration items can be:

1. Source code,

2. Documents

3. Data used in the programs

In Configuration management, there can be multiple versions created for any configuration item (Source code/ documents). Each version can be identified by unique configuration or an attribute which is associated with each version. E.g. the version number.

**Terminologies used in version control**

1. SCI – Software configuration items, i.e. the documents and code which will be having version number and saved.

2. Repository- it is the system where all the SCIs will be stored.

3. Check in- to store the tested and qualified source code.

4. Checkout- to get a copy of the stored SCI from the repository.

5. Add – Add to the local repo and keep ready for commit

6. Commit- to save the file in repository and create a version

Advantages

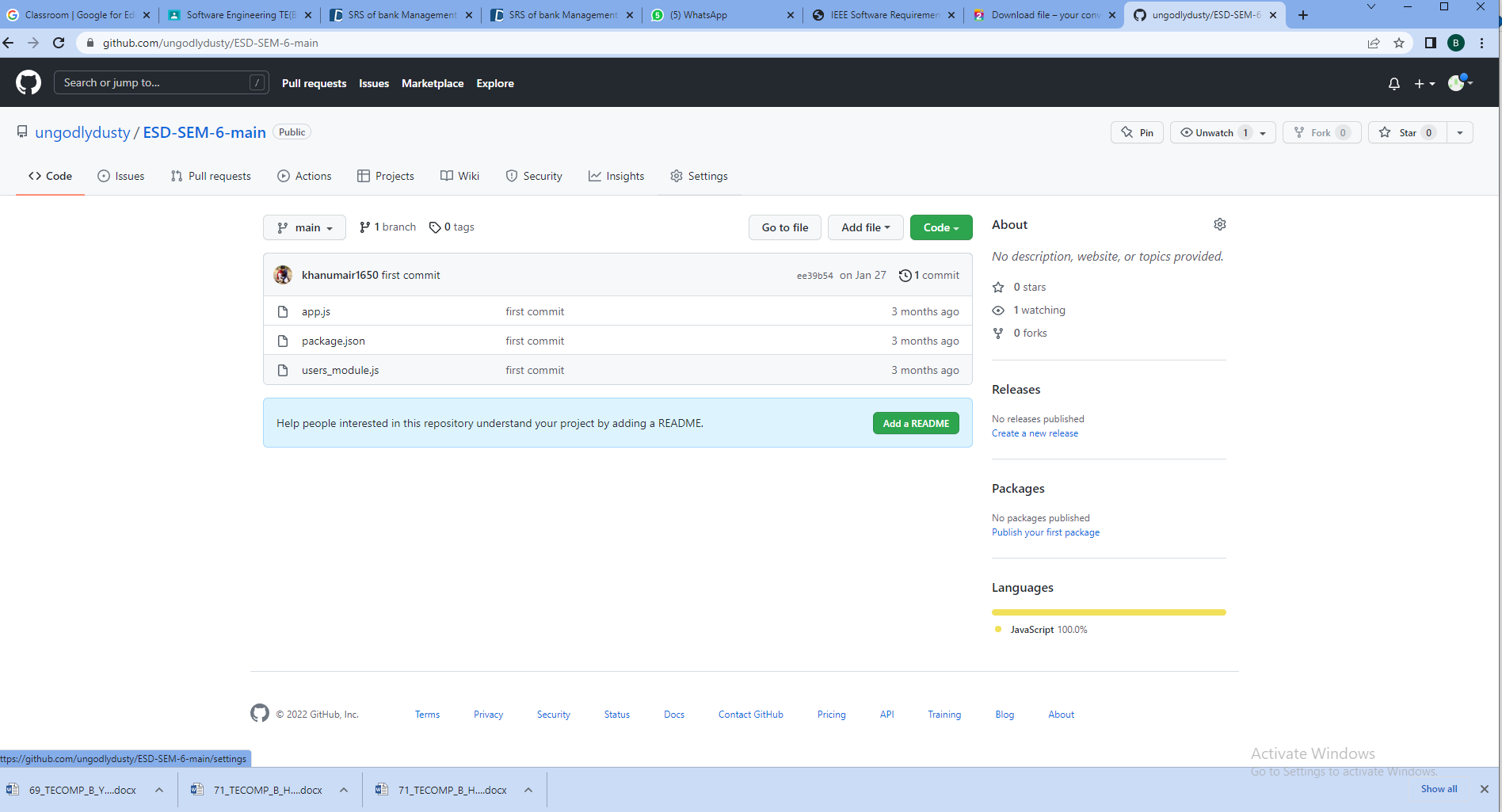
1) The versions are stored in the repository; hence they are available as backups.

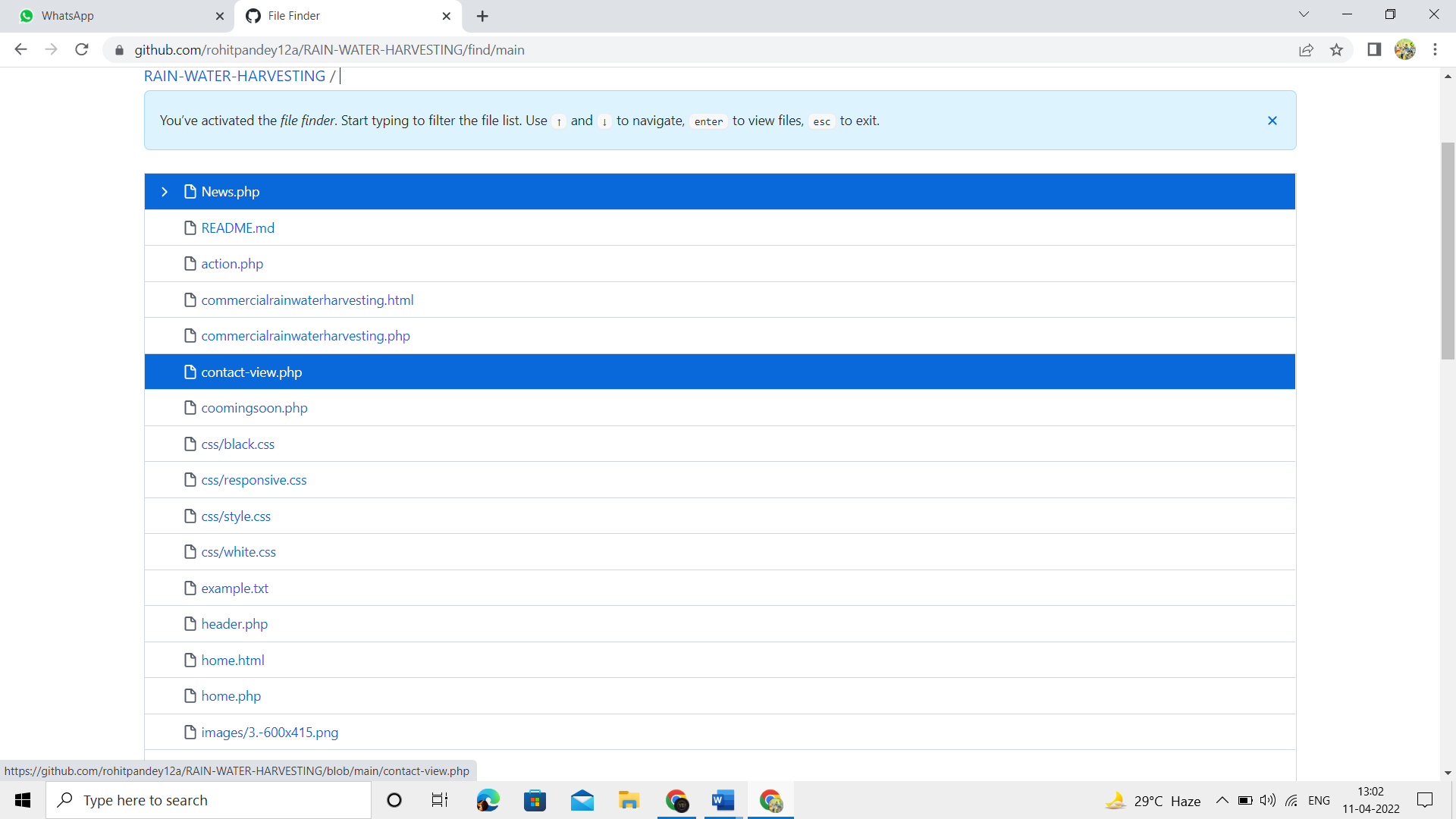
2) Multiple people can work simultaneously on same files/source code, without losing the changes made by other developers

3) It is easy to find the files with specifications as a versions are stored with version numbers

GitHub offers all of the [distributed revision control](https://en.wikipedia.org/wiki/Distributed_revision_control) and [source code management](https://en.wikipedia.org/wiki/Source_code_management) (SCM) functionality of [Git](https://en.wikipedia.org/wiki/Git_(software)) as well as adding its own features. Unlike Git, which is strictly a [command-line](https://en.wikipedia.org/wiki/Command-line) tool, GitHub provides a [Web-based graphical interface](https://en.wikipedia.org/wiki/Web_application) and desktop as well as mobile integration. It also provides [access control](https://en.wikipedia.org/wiki/Access_control) and several collaboration features such as [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system), [feature requests](https://en.wikipedia.org/wiki/Software_feature), [task management](https://en.wikipedia.org/wiki/Task_management) for every project.

**Output:**

****



**Learning Outcomes:** Students should have the ability to

**LO1:** to understand the need of doing configuration management.

**LO2**: Identify the dissimilarity between version and variant.

**LO3**: provide the knowledge of the benefits of using version control.

**LO4**: To understand the types of version control system.

**Outcomes:** Upon completion of the course students will be able to create versions for the project.

**Conclusion:** Thus, we have successfully understood and change the specifications using GitHub which is also a SCM tool and made different version of our project.

For Faculty Use

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** |  |
| **Marks Obtained** |  |  |  |

**Experiment 9:** **Apply the knowledge of test cases for the project using white box testing**.

**Learning Objective:** Students will able to create unit test cases

**Tools:**  Junit

**Theory:**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

**Unit Testing:**

Unit testing focuses on the building blocks of the software system, that is, objects and subsystems. The specific candidates for unit testing are chosen from the object model and the system decomposition. In principle, all the objects developed during the development process should be tested, which is often not feasible because of time and budget constraints. The minimal set of objects to be tested should be the participating objects in the use cases. Subsystems should be tested after each of the objects and classes within that subsystem have been tested individually Unit testing focuses verification effort on the smallest unit of software design—the software component or module.The unit test is white-box oriented. . In Unit testing the following are tested,

1. The module interface is tested to ensure that information properly flows into and out of the program unit under test.
2. The local data structure is examined to ensure that data stored temporarily maintains its integrity.
3. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
4. All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once.
5. And finally, all error handling paths are tested

Write a program to calculate the square of a number in the range 1-100

#include <stdio.h>

int main()

{

    int n, res;

    printf("Enter a number: ");

    scanf("%d", &n);

    if (n >= 1 && n <= 100)

    {

res = n \* n;

printf("\n Square of %d is %d\n", n, res);

    }

    else if (n<= 0 || n > 100)

printf("Beyond the range");

    return 0;

}

|  |  |  |
| --- | --- | --- |
| Sr no | Input | Output |
| 1 | -2 | Beyond the range |
| 2 | 0 | Beyond the range |
| 3 | 1 | Square of 1 is 1 |
| 4 | 100 | Square of 100 is 10000 |
| 5 | 101 | Beyond the range |
| 6 | 4 | Square of 4 is 16 |
| 7 | 62 | Square of 62 is 3844 |

Test Cases

Test case 1 : {I1 ,O1}

Test case 2 : {I2 ,O2}

Test case 3 : {I3, O3}

Test case 4 : {I4, O4}

Test case 5 : {I5, O5}

Test case 6 : {I6, O6}

Test case 7 : {I7, O7}

**Result:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Name** | **Test Data** | **Input** | **Expected Output** | **Actual Output** | **Test Status** |
| **TC\_1** | User Login | User will login using Login ID and Password | Login ID: ABC  Password: abc@123 | Successfully Logged in | Successfully Logged in | Pass |
| **TC\_2** | Home Page | Home page should be displayed | - | Home page display | Home page display | Pass |
| **TC\_3** | Balance Enquiry | Balance: Check the balance  Display: Display the Cash | Click on Balance Enquiry tab | Balance page opens | Balance page opens | Pass |
| **TC\_4** | Withdrawal | Users should be able to withdraw money | Click on withdraw tab  Display1: Limit of per day transaction (<25000)  Display2: Amount should be between 500 to 25000 | Withdrawal successfull | Withdrawal successfull | Pass |
| **TC\_5** | Fund Transfer | Users should be able to transfer fund | Enter account details to whom the transfer is to be made | Transfer succesfull | Transfer succesfull | Pass |
| **TC\_6** | Home Page | Return to home page | Click on Home button/tab | Home page display | Home page display | Pass |
| **TC\_7** | Account Log | Access the monetary transaction in account log | Account log button/tab | Display account log | Display account log | Pass |
| **TC\_8** | Customer Details | Go to customer detail page | Click on Customer button/tab | Displays customer details | Displays customer details | Pass |
| **TC\_9** | Update  Customer  Details | Update customer and their details in the log | -Enter new details of the the user | Log created | Log not created | Fail |
| **TC\_10** | User Logout | Logout from the system | Click Logout button | Successfully logged out | Successfully logged out | Pass |

**Learning Outcomes:** Students should have the ability to

**LO1:** Students will be able to understand Software Testing Concepts and the various Software standards.

**LO2**: to test a software with the help ofJunit

**LO3**: create test cases

**LO4**: To understand different tools for testing

**Outcomes:** Upon completion of the course students will be able to write test cases for the project.

**Conclusion:**

Thus we designed a few test Cases for our chosen project.

**Viva Questions:**

1. **What is difference between git and Github?**
2. **What is version control? Why is it required?**
3. **What are other tools for version control?**
4. **What are different types of version control?**

For Faculty Use

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** |  |
| **Marks Obtained** |  |  |  |  |

**Experiment 10:** **Study a project that is relevant to your project and prepare a case study.**.

**TECOMPB 50 Sahil Paranjape**

**TECOMPB 53 Dwij Patel**

**TECOMPB 65 Siddharth Mishra**

**Learning Objective:** Students will study and understand the project, thus creating the case study on the same.

**Tools:**  MS Word

**Theory:** The main aim of the project under consideration is the management of the database of a bank management system. This project is an insight into the design and implementation of a Bank Management System. This is done by creating a database of the available banks in the city. The primary aim of this bank management system is to improve efficiency and provide ease and control in banking processes. The aim of this project is to develop software for the effective management of a bank. We have developed this software while ensuring effective and accurate statistics are used to develop the software.

**Description on the topic:**

The main purpose that banks have been serving since their inception is keeping our money safe for us. While keeping our money safe, they also let us earn a certain amount of interest on the money deposited with them. Traditional banks have been doing this, and internet banks continue the same function. The only difference is in the way the transactions are made.

**Problem Definition:**

We all know about internet banking but few of us actually understand about the history of internet banking and how it all came out. Knowing history of internet banking can be incredibly useful, especially since it will allow us to have more respect for the little things that we take for granted internet banking has been around for quite a few years now, but has really only become prominent over the past year or so in particular. Internet banking offers an array of different advantages to the user, including account balances and history including year-to date information, the ability to transfer money from one account to another and to payees for bill payments, check history, reorders, and stop payments, check credit card balances and statements, complete online loan applications, secure interactive messaging with staff and much more.

**Objectives**

-> Primary objective

* Create a banking system that is easily accessible by customers from the comfort of their homes, offices etc.
* Reduce the flow of human traffic and long queues at banks.
* Reduce the time wasted in going to banks to stay on queues.
* Promote efficient and effective banking for the banks by focusing on those services that still require physical presence at the banking hall.

**Hardware and software tools:**

The system services and goals are established by consultation with system users. They are then defined in details and serve as a system specification. System requirements are those on which the system runs.

**Hardware Requirements:**

o Computer with either Intel Pentium processor or AMD processor.

o 1GB+ DDR RAM

o 40GB hard disk drive

**Software Requirements:**

o Windows/ MacOS/ Linux operating system.

o JRE and JDK.

o MySQL server (WAMP or XAMPP or any)

**Database Design** is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems.

It helps produce database systems:

o That meet the requirements of the users

o Have high performance.

**Architecture Description**

Data Flow Diagram (Common)

Diagram

Description automatically generated

Entity Relationship Diagram

Diagram

Description automatically generated

Fig 1: ER Diagram

An entity–relationship model describes interrelated things of interest in a specific domain of knowledge (Refer Fig 1). It is composed of entity types and specifies relationships that can exist between instances of those entity types.

**CONCLUSIONS AND FUTURE SCOPE**

Thus reaching to the conclusion of our project we observe that Traditional  banks offer many services to their customers, including accepting customer’s   money, deposits, providing various  banking  services to  customers,  and making loans  to  individuals  and  companies.  Compared  with traditional channels of offering banking services through physical branches, e-banking uses the Internet to deliver traditional banking services to their customers, such as opening accounts, transferring funds, and electronic bill payment. E-banking services are delivered to customers through the Internet and the web using Hypertext Markup Language (HTML). In order to use the e-banking services, customers  need Internet access and web browser software.

Multimedia  information in HTML format from online banks can be displayed in web browsers. The heart of the e-banking application is the computer system, which includes web servers, database managementsystems, and web application programs that can generate dynamic HTML pages. The range of e-banking services is likely to increase in the future. Some banks plan to introduce electronic money and electronic checks. Electronic money can be stored in computers or smart cards and consumers can use the electronic money to purchase small value items over the Internet. The Scope of this project is limited to the activities of the operation units of the banking system which include opening of account, deposit of funds, withdrawal of funds & transfer.

🡪 Any bank can use this application to provide better service to their customers.

🡪 Customers can access his/her all accounts present in various branches of the same bank at one click.   
🡪 Bank can publish various upcoming plans for customers through this application.  
🡪 Manager can access all accounts present in the bank through this application.  
🡪 Reduction in work load of all employees will possible through this application as transaction rights are provided online to customer.  
🡪It can be extended for global communication between all banks in the world.

For Faculty Use

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** |  |
| **Marks Obtained** |  |  |  |