

INDRA GANDHI NATIONAL OPEN UNIVERSITY

PROJECT REPORT

BCSP-064

ON

ONLINE BANKING SYSTEM

By

Kaustav Bhattacharya (176568741)

UNDER GUIDANCE

OF

Mr. Sanjeev Kumar

**Submitted to the School of Computer and Information Science in
partial fulfilment of the Requirements for the degree of****Bachelor****Of****Computer Applications****Indira Gandhi National Open University****MaidanGarhi****New Delhi-110068**



**INDIRA GANDHI NATIONAL OPEN UNIVERSITY
Regional Centre Noida**

C-53, Sector-62, Institutional Area, Noida- 201305 (UP), India
Phone: 0120-2405012, 2405013, Fax: 2405013
Email Id: rnoidea@ignou.ac.in

F. No. : IG/RCN/MCA/BCA/Pr/2020/3040
Dated:-29/09/20

Dr. M. A. Laskar
Deputy Director

To,
KAUSTAV BHATTACHARYA
 16/15 GULMOHUR ROAD
 SHIPRA SUN CITY
 INDIRAPURAM-201014
 UTTAR PRADESH

**Sub: Evaluation of Synopsis of MCA/BCA (MCSP-060/BCSP-064) in r/o Kaustav
Bhattacharya (Enrolment No. 176568741) Prno-B390001F20**

Find below the status your evaluated project proposal received from the evaluator. The status is as follows:

Status of Synopsis	Remarks/ Comments
APPROVED	Follow BCSP-064 Guidelines, Ensure security, Do the system study at a Bank

You are advised to take note of the observations (if any) made by the evaluator in the proposal before submitting the project report. You are required to enclose the approved project proposal (in original) and submit your project report to IGNOU Regional Centre-Noida within the stipulated time frame. (15th October 2020)

Best Wishes,

(M. A. Laskar)

IX CERTIFICATE OF ORIGINALITY

This is to certify that the project report entitled _____

Submitted to **Indira Gandhi National Open University** in partial fulfilment of the requirement
for the award of the degree of **BACHELOR OF COMPUTER APPLICATIONS (BCA)**, is
an original work carried out by Mr./ Ms. _____

Enrolment No.: _____ under the guidance of Mr./ Ms. _____

The matter embodied in this project is a genuine work done by the student and has not been
submitted whether to this University or to any other University / Institute for the fulfilment of the
requirement of any course of study.

Signature of the Student

Name and Address
of the student :

Enrolment No.:

Sanjeev Kumar

Signature of the Guide

Name, Designation and
Address of the Guide

*SANJEEV KUMAR
SOFTWARE DEVELOPER*

*SANJEEV KUMAR
C/O KUNDAN KUMAR
H.NO: 285, GROUND FLOOR,
TYPE-3, SEC-3, SADIQUE NABAR.
NEW DELHI-110049*

TABLE OF CONTENTS

<u>S.No.</u>	<u>CONTENTS</u>	<u>PAGE NO.</u>
1.	GUIDE BIODATA, MARKSHEET, PAN CARD	5 - 12
2.	SYNOPSIS	13 - 40
3.	PROJECT REPORT	41 – 336
4.	BIBLIOGRAPHY	337

Curriculum Vitae**Sanjeev Kumar**

Qualification:- MCA
Experience:- 10 Years 6 Months
Mobile No.:- +91-9718143197
Email:- sanjeev.srvstv@gmail.com

Objective

- ✓ To provide the valuable technical expertise in any technology in process-oriented organization with learning opportunities provide mean opportunity to grow personality and acquire knowledge.

Personal Skills

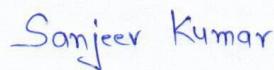
- ✓ Appetite to learn new things, a person waiting to take new challenging assignments that helpfully to utilize my skills and abilities.

Educational Qualifications

Degree	School/College	University/Board	Academic year
M.C.A	U.I.C.T New Delhi	I.G.N.O.U New Delhi	2012
B.C.A	U.I.C.T New Delhi	I.G.N.O.U New Delhi	2010
12 th	M.J.K College Bettiah Bihar	B.I.E.C Patna Bihar	2007
10 th	Raj High School Bettiah Bihar	B.S.E.B Patna Bihar	2005

Professional Experience

- Indian Council of Medical Research(ICMR), Medical Enclave, Near AIIMS Delhi-110029 Working as a Computer Programmer (Grade-B) on Burden Of Non-communication Diseases Project from (04-September-2017 to Till Date)
- National Council of Educational Research and Training Sri Aurobindo Marg, New Delhi-110016, worked as a Project Fellow(IT) on Adolescence Education Programme funded by United Nation Population fund Association (UNFPA) From(01-July-2014 to 03-September-2017)
- Stucare Solutions Private Limited, 553,3rd floor Aggarwal, Chamber-III Veer Savarkar Block Sakarpur, Delhi-110092 Worked as Software Developer from (01-January-2010 to 30-June-2014)



Professional Expertise	
Programing Languages	Web-based technologies : CORE PHP5, Java Script,Python JQUERY, Angular JS, Moodle, Drupal, Wiki, OJS, PHP-BB,PHP,WordPress, Joomla, CakePHP.CoreJava, Advance Java(Hibernet + Spring),Struts, Hibernet,Visual Basic, C, C++ and Java, Unified Modeling Language (UML),SAP business software applications,
Databases	ORACLE 11g,MY SQL (5.0+ above), PL/SQL,SQL, Sequel, SQLi, Sql Server,Maria DB
Web Technology	Wordpress,Drupal,HTML, DHTML,XHTML,XML,XSLT,CSS,AJAX,Angular JS, BOOTSTRAP Node JS, React JS, Taraform,Cloud Front, Git Hub,OOPS, MVC, WCF, Concepts,Symphonic, BeanStalk,Jira, Phpununit,AWS etc.
Web Servers	Apache,Wamp 8.1,Xamp 1.8.2,3 Lamp 0.0.12
PHP Frameworks	Zend,CakePHP,Codeigniter
Web Services used Technology	Zend Framework, Codeigniter, Amazon Web Services(AWS)
Build Tool	Phing
Editor's	Eclipse,Netbeans>Editplus,Dreamweaver
Open Source Software(CMS)	Drupal 6,7,8,Wordpress,Joomla,Magento
Browser Scripting	JavaScript, AJAX, JQuery, JSON
Mobile apps	Android,ios Iphone
Documentary Skill	Algorithm Design and Understand, SRS
Testing Skill	Software Testing and Test Cases & Unit Testing
Outside of PHP	PG Degree in Computer Application with Above 9 years experience IT Experience in Software Development and knowledge of designing and developing client/server applications, skill in application programming and systems analysis and related programming support functions and should have the proven knowledge of relational database management software such as MS SQLserver and Oracle/ERP/accounting software in IT/ICT Systems implementation.Should have experience in development and implementation of web based applications and mobile applications Expertise in MS office including Word, Excel Power Point and Ms-Outlook. Good oral and written communication skills both in English and Hindi Above 9 years experience in IT System Implementation Experience in managing large scale technology implementation or health programs in Government Experience of working with Government/Government Organizations Experience in mobile technology for community health or nutrition (Health) programs Experience in managing a technical team Lead the L-2 Support Team; provide directions to the members for ensuring technical support issues are addressed satisfactory and promptly Support Ministry in technical discussions and technical analyses. Support in resolving complex problems. Interface with

Sanjeev Kumar

	NIC/NICSI and Infrastructure team in regards to implementation of ICT-RTM. Monitor the work being done by the Infrastructure team. Support the NNRC-CPMU in planning for scale up of ICT-RTM. Support the NNRC-CPMU in creating complex custom data analyses. Conduct refresher training of Central-government-level and State-level CAS users, as needed. Support the maintenance of the CAS System and certain, minor updates of CAS content enable through that CAS web administration site like users of CAS system, using the CAS web administration site. Any other related activities of the project that may be assigned by the Executive Director.
Operating Systems	• OS: Windows, Linux, SPSS, Bilog, UML Analysis Software, IRT (Core PHP and J2EE), Facebook API Bulk SMS API, Twitter API, LinkedIn API, Google API, Off page optimization, OnPage optimization, Upgrade of Sites in Drupal with Custom theme integration with custom plugin development in both technologies (Drupal & Wordpress), IRT, Agile/Responsive Designs, Code Integration, Code Optimization and validations, Web Development, Web Maintenance and UI Integration, OOPs concepts, experience in writing PL/SQL queries, to be a good team player.

Web Developer (PHP, Drupal Developer) With Ecommerce Experience And Strong Background In open Source Technologies Including,

Drupal 8, Drupal 7, 6 Linux ,Platform ,MySQL ,Jquery Apache ,PHP, Wordpress

1. Experience In database design software engineering and development.
2. Proven ability to jump in new project and learn New Technologies quickly.
3. Extensive Drupal Experience and very active in the Drupal community Developing New modules.
4. Experience With ecommerce on Drupal Utilizing Ubercart and Drupal commerce.
5. Support Developer & Maintainer of Drupal Modules.
6. Payment gateways

Projects	
1.	ICMR Employees Paybill Software online (http://bic.icmr.org.in/paybill/index.php)
Role : Team Member(Developer) Description: The Indian Council of Medical Research (ICMR), the apex body in India for the formulation, coordination and promotion of biomedical research, is one of the oldest and largest medical research bodies in the world. The ICMR is funded by the Government of India through the Department of Health Research, Ministry of Health ICMR Employees Paybill Software online providing Monthly Salary Slip of all employees of ICMR along with individual login id & Password. Environment: Core PHP, MySql, Ajax, JS, Jquery, HTML, CSS	

Santosh Kumar

Responsibilities:

- Developing the code as per the requirements.
- Handled different types of issues.
- Writing code and Integrate the modules

Tools: Eclipse

OS:Linux using open source software (Ubuntu 13.4+LTS)

2. NIMS Employees Paybill Software online

(<http://14.139.60.53/paybillnims/Loginindex.php>)

<http://218.248.6.39/nutritionatlas/dashbord/cd.php>

<http://218.248.6.43:8080/CountWhatYouEat/Home.do>

Role : Team Member(Developer)

Description: *National Institute of Medical Statistics (NIMS)* is one among 28 institutes of Institutes of Indian Council of Medical Research, (ICMR), New Delhi & known as National Institutes of Medical Statistics(NIMS). The Institute came into existence in the year 1977 with the mandate of the Institute to provide technical expertise on research methodology, NIMS Employees Paybill Software online providing Monthly Salary Slip of all employees of NIMS along with individual login id & Password.

Environment::-Core PHP, MySql, Ajax, JS, Jquery, HTML, CSS, Java Script.

Responsibilities:

- Developing the code as per the requirements.
- Handled different types of issues.
- Writing code and Integrate the modules

Tools: Eclipse

OS:Linux using open source software (Ubuntu 13.4+LTS)

3 . Adolescence Education Programme

(<http://www.aeparc.org/>)

Role : Team Member(One Developer),Complete developed website & reply of student using it's admin panel

Description: India is home to 253 million adolescents; young people in the age group of 10-19 years who comprise 21% of the country's population (*Census, 2011*). Not only does this cohort represent India's future in the economic realm, but its experience, attitudes and behaviours will largely determine whether India is able to realize the vision of an equitable civil society envisaged in its constitution. National Council of Educational Research and Training (NCERT) and the United Nations Population Fund (UNFPA) is organising the Youth Festival (NYF) -2016 under the Adolescence Education Programme(AEP) on the theme, 'My School – My Space..!'. from 5 to 8 December, 2016 at NCERT, New Delhi. This document contains important instructions and guidelines regarding the event.

Environment: , Drupal 7, PHP, HTML, CSS, jquery MySQL, phptemplate, AJAX, Javascript.

Responsibilities:

-Developing the code as per the requirements.

-Handled different types of issues.

Tools: Eclipse.

OS: Linux using open source software Ubuntu 13.4+ LTS

Developed & Launched Several Drupal Sites including:

Role : Team Member(Three Developers)

Description:-IAS100.IN IAS100.in is online portal for Civil Services exam preparation, ias test series for prelims and mains in hindi and english, updated ias study material, upsc Data Analytic Solution (WWW.IAS100.IN) is developed for basic and advanced Statistical Computing and Graphics mechanism. Like mean, mode, median etc. All based on R (programming language).

Environment: PHP5, SQL engine, JQuery, Ajax, JS, MySQL, Apache, Tomcat etc.

Responsibilities:

-Developing the code as per the requirements.

Tools: Eclipse

OS: Linux using open source software Ubuntu 13.4+ LTS

4. NIMS(nims-icmr.nic.in/NIMS/index.jsp)

Role : Team Head (Two Developer)

Description: The project is being developed for **National Institute of Medical Statistics** is one of the permanent institutes of Institutes of Indian Council of Medical Research, (ICMR), New Delhi & known as National Institutes of Medical Statistics(NIMS).The Institute came into existence in the year 1977 with the mandate of the Institute to provide technical expertise on research methodology, programme evaluation, mathematical modeling, data analysis etc. It also provides technical assistance to institutes throughout India. Institute for Research in Medical Statistics (IRMS), an institute of Indian Council of Medical Research, established in 1977 will be known as National Institute of Medical Statistics (NIMS) from 9th November 2005.

Environment: JSP,HTML,CSS,JQUERY,.JS,Flash

Responsibilities:

Developing the Code as per the requirements of client.

- Involved in developing the library for company

- issues resolving

- integration of chart in project module as per requirements

Tools: Eclipse

OS: Linux using open source software Ubuntu 13.4+ LTS

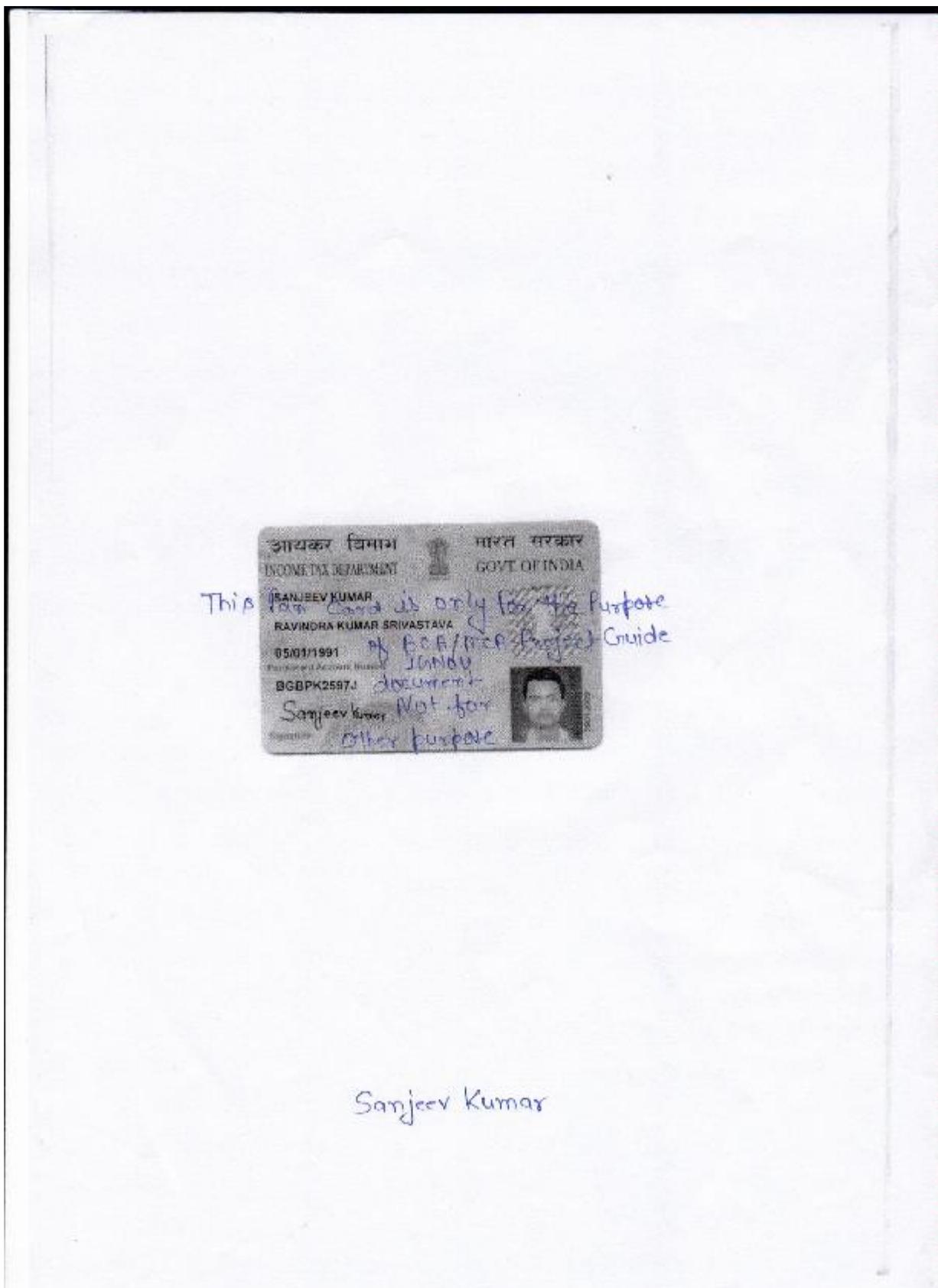
Contd... *[Handwritten]*

Hobbies	Computer Programming, Reading Computer Books along with PMA Books, Ready for Learn something New Technology or others, Travelling, Developed Creativity in software development, Problem Solving skill in Project flow issues, Good in analytical skills and quick problem solver.												
Achievements	<ul style="list-style-type: none"> ✓ Best Year of the employee. ✓ Best Month of the employee. <p><i>This resume Project Guide documents is only for BCA/MCA JUNIOR purpose not for Other</i></p>												
Personal Information	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>Father's Name</td><td>Shri Ravindra Kumar Shrivastava</td></tr> <tr> <td>Date of Birth</td><td>5th Jan 1991</td></tr> <tr> <td>Local Address</td><td>Sanjeev Kumar C/o Kundan Kumar, House.Number:-285, Ground Floor, Type-3, Sector-3, Sadique Nagar Delhi-110049.</td></tr> <tr> <td>Language</td><td>English, Hindi</td></tr> <tr> <td>Marital Status</td><td>Single</td></tr> <tr> <td>Contact No</td><td>+91-9718143197, +91-8178103225</td></tr> </table>	Father's Name	Shri Ravindra Kumar Shrivastava	Date of Birth	5 th Jan 1991	Local Address	Sanjeev Kumar C/o Kundan Kumar, House.Number:-285, Ground Floor, Type-3, Sector-3, Sadique Nagar Delhi-110049.	Language	English, Hindi	Marital Status	Single	Contact No	+91-9718143197, +91-8178103225
Father's Name	Shri Ravindra Kumar Shrivastava												
Date of Birth	5 th Jan 1991												
Local Address	Sanjeev Kumar C/o Kundan Kumar, House.Number:-285, Ground Floor, Type-3, Sector-3, Sadique Nagar Delhi-110049.												
Language	English, Hindi												
Marital Status	Single												
Contact No	+91-9718143197, +91-8178103225												


Date:- 27-09-2020

Sanjeev Kumar
Signature of the Guide

	इन्दिरा गांधी राष्ट्रीय मुक्त विश्वविद्यालय INDIRA GANDHI NATIONAL OPEN UNIVERSITY STUDENT EVALUATION DIVISION Maidan Garhi, New Delhi - 110 068 Master of Computer Applications STATEMENT OF MARKS										8952			
MCA 073806076 SANJEEV KUMAR PRAGATI MARKETING BHOLA BATH MARKET COMPLEX TEEN LALTAH RD WEST CHAMPARAN BIHAR 845438											CERTIFICATE NO.: AD 0004186			
<i>This certificate is only for IGNOU MCA Project Submission Grade Document no. 1 for MCA project</i>												<small>Marksheet</small> DATE: 23/06/2012 JUNE 2012		
TERM-END EXAM : Sanjeev Kumar														
COURSE CODE	CONTINUOUS EVALUATION			TERM-END EXAMINATION			WEIGHTAGE		M.M.	OVERALL MARKS		COURSE STATUS		
	MKS OUT	L.G.	MKS DBE	L.G.	25% of C.E.	75% of TEE	MKS	L.G.						
CREDIT TRANSFER FROM BCA														
MCS031	85	A	40	D	21	30	100	51	C+	SC				
MCS032	82	A	42	D	21	32	100	53	C	SC				
MCS033	85	A	40	D	21	30	100	56	C	SC				
MCS034	80	A	40	D	20	30	100	50	C	SC				
MCS035	87	A	40	D	22	30	100	52	C	SC				
MCS041	86	A	40	D	22	30	100	52	C	SC				
MCS042	84	A	40	D	21	30	100	51	C	SC				
MCS043	83	A	40	D	21	30	100	51	C	SC				
MCS051	85	A	40	D	22	30	100	52	C	SC				
MCS052	89	A	58	S	22	31	50	37	B	SC				
MCS053	87	A	44	D	22	33	100	55	C	SC				
MCS003	90	A	40	D	23	30	100	53	C	SC				
MCS004	89	A	40	D	22	30	100	52	C	SC				
MCS011	87	A	43	D	22	32	100	54	C	SC				
* STUDENT IS FROM BCA-MCA INTEGRATED SCHEME.														
COURSES OF 2nd AND 3RD SEMESTER WHICH IS 16 CREDITS ARE EXEMPTED FOR STUDENTS OF BCA ADMITTED DIRECTLY TO 3RD SEMESTER OF MCA UNDER THE SCHEME OF INTEGRATED PROGRAMME. MARKS FOR THESE SEMESTERS ARE TRANSFERRED INTO THIS PROGRAMME AS "CREDIT TRANSFER FROM" BCA.														
LAB COURSES														
COURSE CODE	CONTINUOUS EVALUATION			TERM-END PRACTICAL EXAMINATION			WEIGHTAGE		M.M.	OVERALL MARKS		COURSE STATUS		
	MKS OUT	L.G.	MKS SEC-1	MKS SEC-2	MKS SEC-3	MKS SEC-4	T.O.T. OUT OF 100	25% of C.E.		75% of TEE	MKS		L.G.	
MCS036	86	A	20	25	20	8	65	B	22	49	100	71	B	SC
MCS045	75	A	14	18	16	8	64	B	17	42	50	34	B	SC
MCS054	89	A	22	13	16	8	70	B	22	53	50	36	A	SC
PROJECT WORK														
COURSE CODE	CONTINUOUS EVALUATION			REPORT MARKS			VIVA MARKS			M.M.	OVERALL MARKS		COURSE STATUS	
	MKS	L.G.									MKS	L.G.		
MCS044	84	A		36			76		100	68	B	SC		
MCS060	8	A		36			36		200	122	B	SC		
MCA SUCCESSFULLY COMPLETED IN JUNE 2012 !														
MARKS = 1609/2800 WITH 57.45%														
<i>Sanjeev Kumar</i> <i>P. Bhattacharya</i> For details visit www.ignou.ac.in REGISTRATION NO.: 16092800														



SYNOPSIS

1. TITLE OF PROJECT

ONLINE BANKING SYSTEM



2. INTRODUCTION AND OBJECTIVES OF PROJECT

The “Online Banking System” project is a model Internet Banking Site. This site enables the customers to perform the basic banking transactions by sitting at their office or at homes through PC or laptop. The system provides the access to the customer to create an account, deposit/withdraw the cash from his account, also to view reports of all accounts present. The customers can access the banks website for viewing their Account details and perform the transactions on account as per their requirements. With Internet Banking, the brick and mortar structure of the traditional banking gets converted into a click and portal model, thereby giving a concept of virtual banking a real shape. Thus today's banking is no longer confined to branches. E-banking facilitates banking transactions by customers round the clock globally.

The primary aim of this “Online Banking System” is to provide an improved design methodology, which envisages the future expansion, and modification, which is necessary for a core sector like banking. This necessitates the design to be expandable and modifiable and so a modular approach is used in developing the application software. Anybody who is an Account holder in this bank can become a member of Bank Account Management System. He has to fill a form with his personal details and Account Number.

Bank is the place where customers feel the sense of safety for their property. In the bank, customers deposit and withdraw their money. Transaction of money also is a part where customer takes shelter of the bank. Now to keep the belief and trust of customers, there is the positive need for management of the bank, which can handle all this with comfort and ease. Smooth and efficient management affects the satisfaction of the customers and staff members,

indirectly. And of course, it encourages management committee in taking some needed decision for future enhancement of the bank.

Now a day's, managing a bank is tedious job up to certain limit. So software that reduces the work is essential. Also today's world is a genuine computer world and is getting faster and faster day-by-day. Thus, considering above necessities, the software for bank management has become necessary which would be useful in managing the bank more efficiently.

All transactions are carried out online by transferring from accounts in the same Bank or international bank. The software is meant to overcome the drawbacks of the manual system.

The OBJECTIVE is to prepare a software or application, which could maintain data & provide a user friendly interface for retrieving customer related details just in few seconds, with 100% accuracy. Software is completely computerized, so it is not time consuming process.

No paper work required & can be implemented further.

The application should also facilitate the addition of new Customer A/c, deletion of A/c and modification of existing customer A/C. Block transactions for any A/c by Freeze/Unfreeze facility. Show all or required transaction.

Any account can be opened with zero(0) balance also.

3. PROJECT CATEGORY

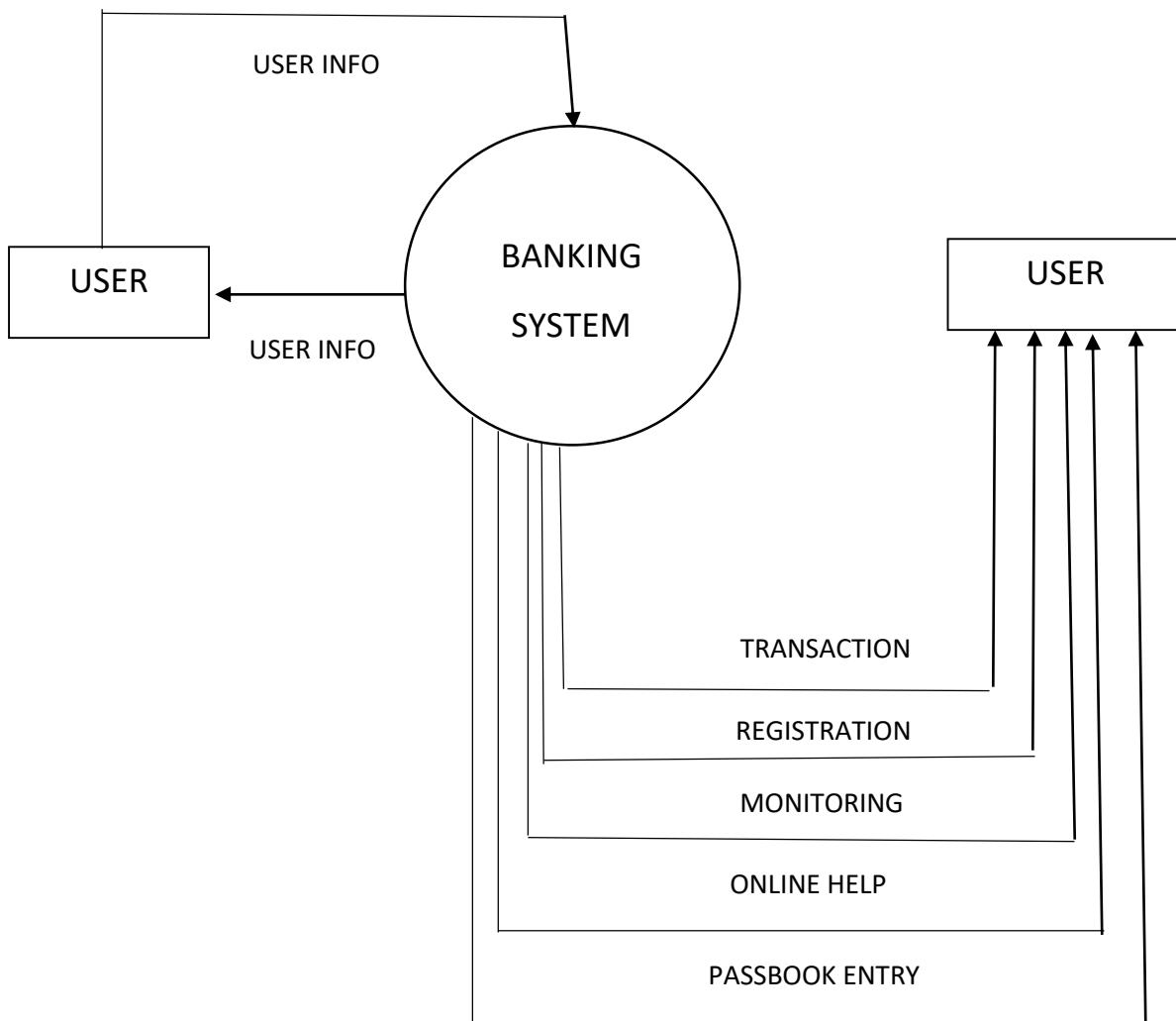
This project is under category of “**ONLINE BANKING SYSTEM**”. This project aims to provide improved functionality and better user interactive environment. With ‘HTML’ as a front end and JSP, Servlet and MySQL as a back end, this project falls in the category of Relational Database Management System (RDBMS) project.

HTML provides a fully user interactive windows based environment that helps user to learn the system in a better way. While MySQL on the other hand, provides better functionality and effectiveness to the system while maintaining the database records. And JSP and Servlet act as a bridge between the front-end and the database All the tables are being used in this project are inter-related and fully normalized, so this project “**ONLINE BANKING SYSTEM**” is a “Web Based Application Project” using Relational Database Management System (RDBMS).

Hibernate is also used, which is an ORM tool which maps the object containing the data to the relational database. It’s basically a tool for JAVA programming language and has many advantages over JDBC.

4. ANALYSIS

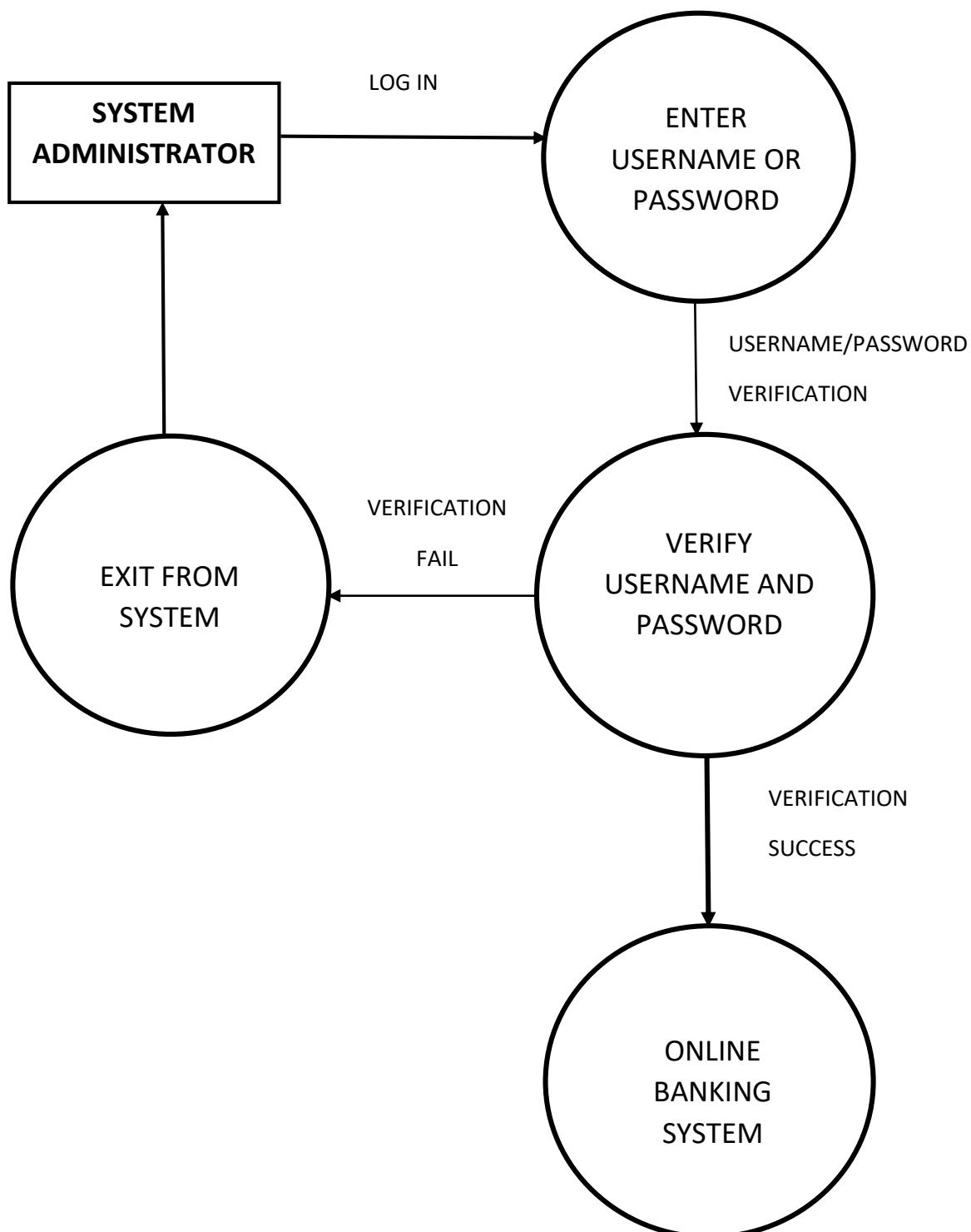
4.1) DATA FLOW DIAGRAM (DFD):



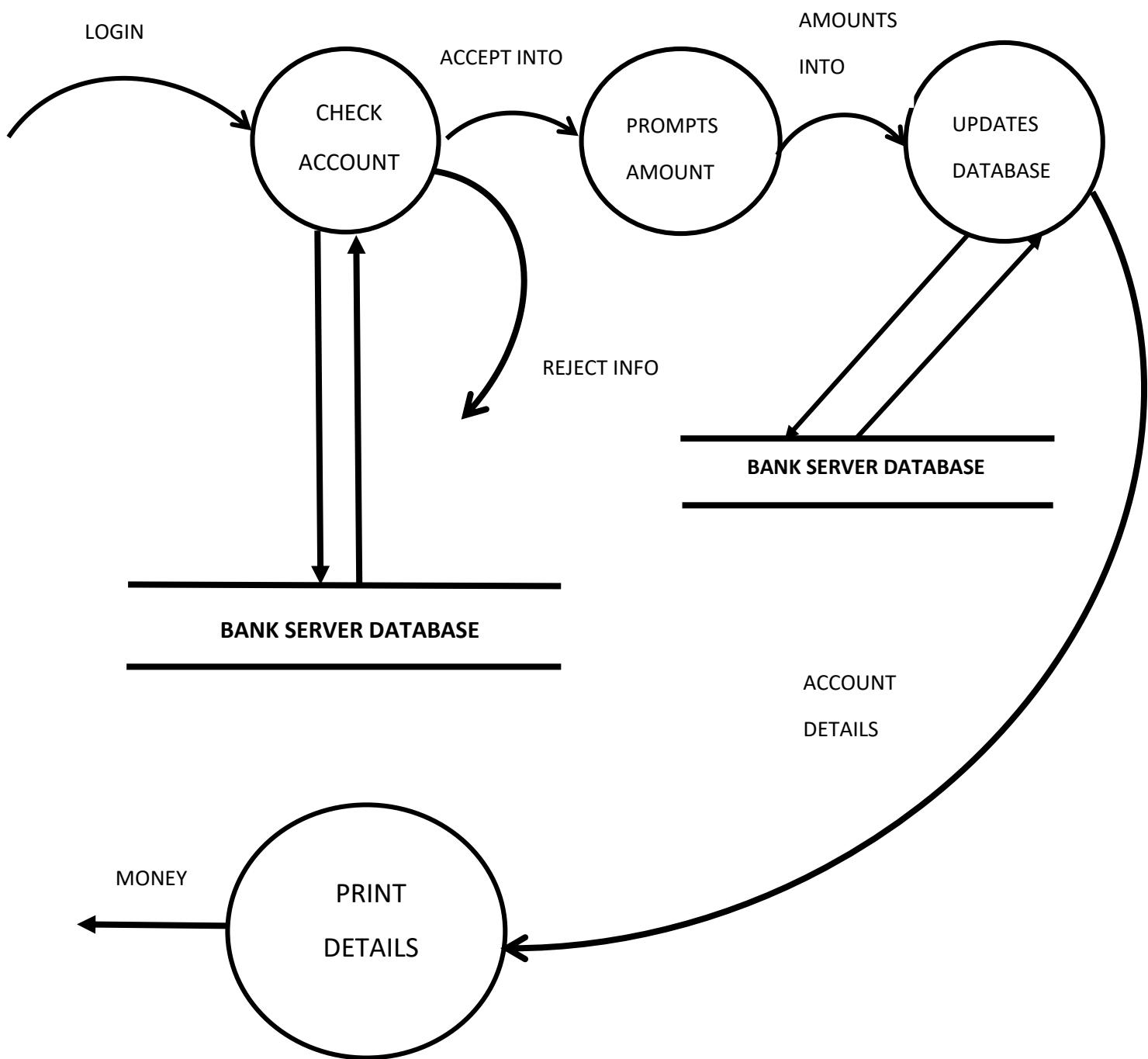
0-(Zero) Level DFD (Context Level DFD)

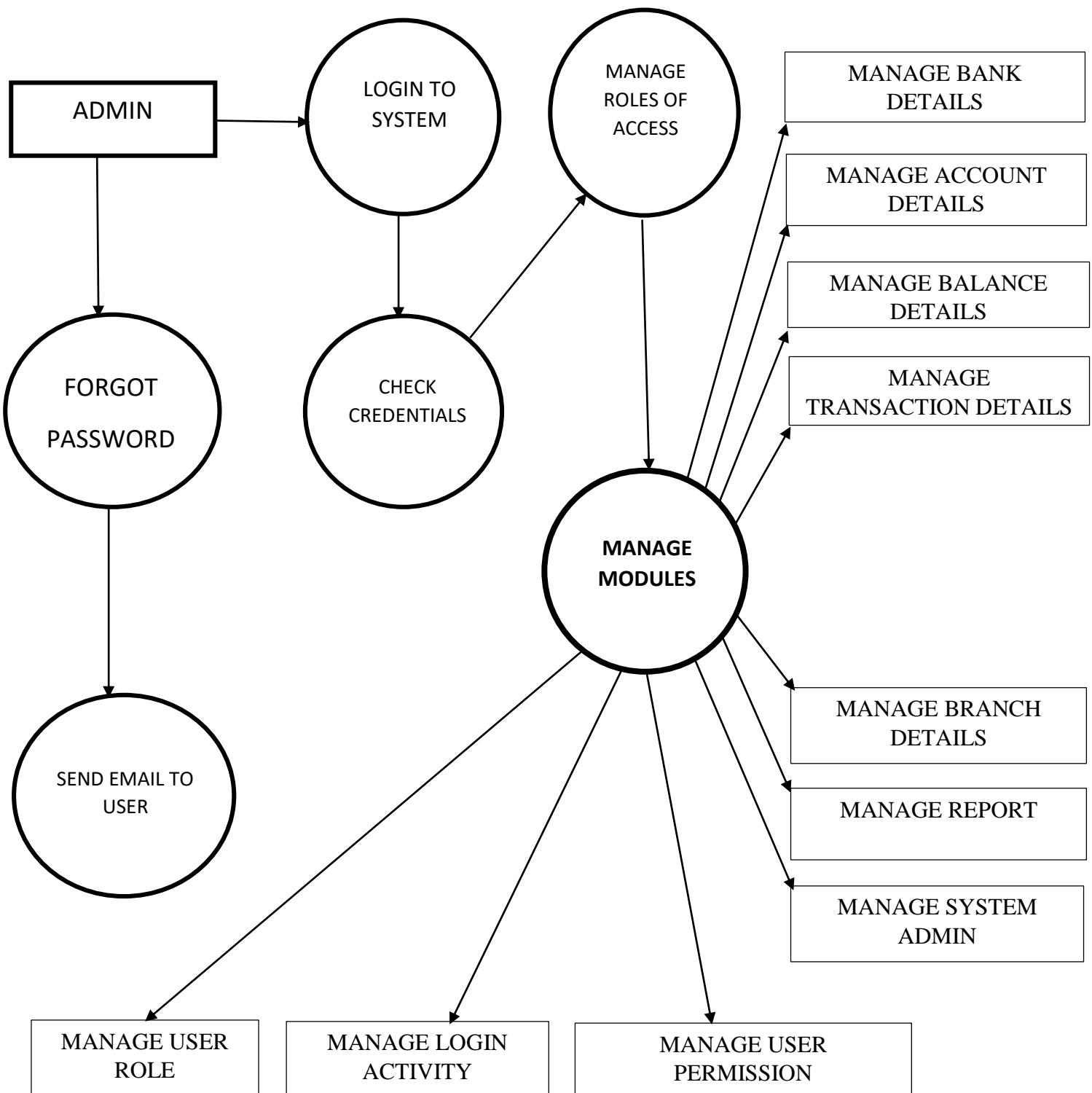
for ONLINE BANKING SYSTEM

First Level of Data Flow Diagram for
System Login in ONLINE BANKING SYSTEM of Admin Login Details

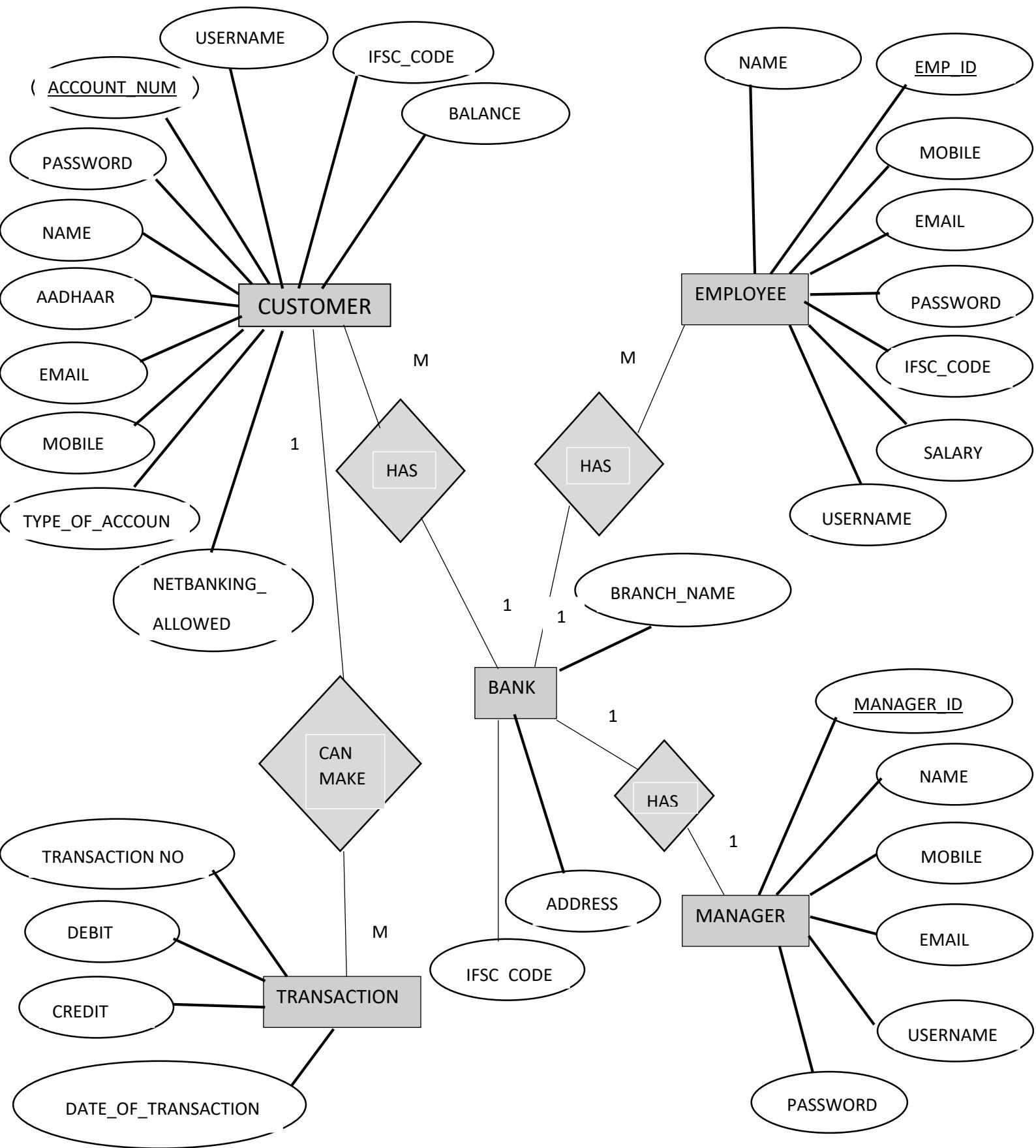


First Level of Data Flow Diagram for
CUSTOMER TRANSACTION in ONLINE BANKING SYSTEM

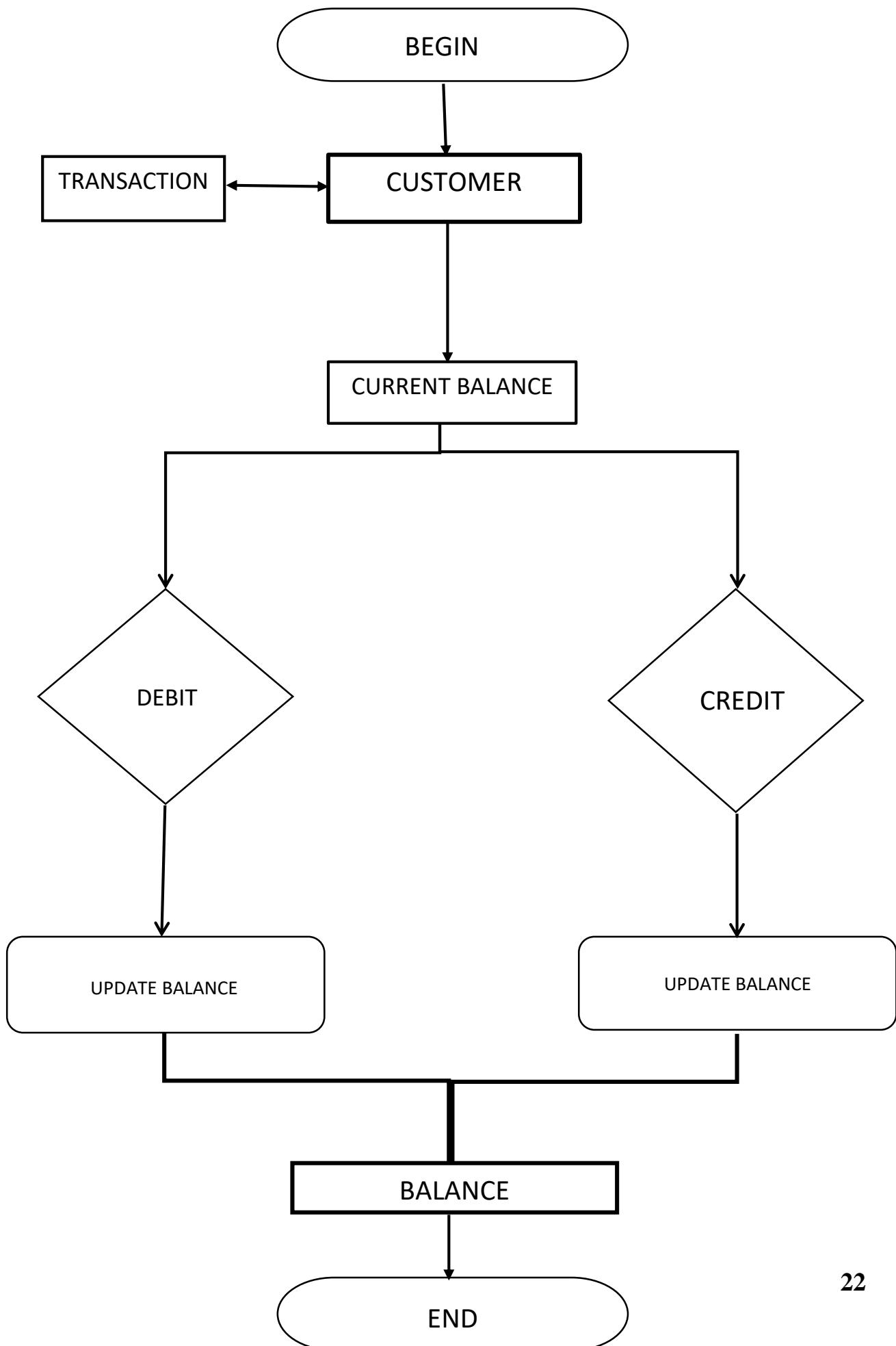


SECOND LEVEL DATA-FLOW-DIAGRAM**OF ONLINE BANKING SYSTEM**

4.2) ER-DIAGRAM



4.3) FLOWCHART



5. COMPLETE STRUCTURE OF PROJECT

5.1 NUMBER OF MODULES AND THEIR DESCRIPTION

The Online Banking System Project consists of 4 functional elements: *Customer transaction module, Employee Transaction Module, Manager Transaction Module and Developer/Administration Module.*

1. CUSTOMER TRANSACTION MODULE

An enhanced atomized system is developed to maintain customer transaction. Features include

- Creation of new Bank Customer
- Type of Customer: Savings, Current
- Account Opening Form/New Customer Sign Up Form
- Existing Customer Login Form
- Each Customer Uniquely identified by A/c no.
- Each customer having a unique User-id and Password
- Main Menu options are like:
 1. Account Summary
 2. Funds Transfer to Another A/c
 3. Customer Profile
 4. Logout
 5. Transaction Report

2. EMPLOYEE TRANSACTION MODULE

An enhanced atomized system is developed to maintain employee role.
Features include

- Employee Details
- Update Customer Details
- Remove Customer
- View All Transaction made by customer
- Freeze/Un-freeze Account
- Deposit into Customer Account

3. MANAGER TRANSACTION MODULE

An enhanced atomized system is developed to maintain manager role. Features include

- Manager details
- Update Employee Details
- Remove Employee
- List all Employee's

4. DEVELOPER/ADMINISTRATOR MODULE

An enhanced module which is used to maintain the entire banking software application. Features include:

- Add Manager
- Remove Manager
- View all bank system tables

5.2 DATA STRUCTURES OF PROJECT:

Table: CUSTOMER

Field	Type	Null	Key	Default
Account_number	Bigint	No	Primary	Null
IFSCCode	varchar	Yes		null
aadhaar	Varchar	yes		null
Balance	Double	yes		null
Date_of_creation	Date	yes		null
Email	Varchar	Yes		null
Mobile	Varchar	yes		null
Name	Varchar	yes		null
netBankingAllowed	varchar	yes		null
Pan	Varchar	yes		null
Password	Varchar	yes		null
Type_of_account	Varchar	yes		null
Username	Varchar	yes		null
IFSCcode_id	Int	yes	foreign	null

Table: BANK:

Field	Type	Null	Key	Default
Id	Int	No	Primary	Null
Address	Varchar	Yes		Null
Branch_name	Varchar	Yes		Null
IFSC	Varchar	Yes		Null
Employee_emp_id	Bigint	Yes	Foreign	Null
Customer_account_number	Bigint	Yes	Foreign	Null

Table: EMPLOYEE:

Field	Type	Null	Key	Default
Employee_id	Bigint	No	Primary	Null
IFSCcode	Varchar	Yes		Null
Aadhaar	Varchar	Yes		Null
Email	Varchar	Yes		Null
Mobile	Bigint	Yes		Null
Name	Varchar	Yes		Null
Salary	Double	Yes		Null
username	Varchar	Yes		Null
Password	Varchar	Yes		Null
IFSCcode_id	Int	Yes	Foreign	null

Table: MANAGER:

Field	Type	Null	Key	Default
Manager_id	Bigint	No	Primary	Null
Email	Varchar	Yes		Null
Mobile	Varchar	Yes		Null
Name	Varchar	Yes		Null
Username	Varchar	Yes		Null
Password	Varchar	Yes		Null
Id	Int	Yes	Foreign	null

Table: TRANSACTIONS:

Field	Type	Null	Key	Default
Transaction_no	Bigint	No	Primary	Null
Debit	Varchar	Yes		Null
Credit	Varchar	Yes		Null
Date_of_transaction	Varchar	Yes		Null

5.3 PROCESS LOGIC FOR EACH MODULE

Here in the project there will be a number of modules and each module is based on respective process logic. The process logic will be both, batch process and on line processing for the respective modules and related tables. For different query purpose the logic will be online type. But different updating processes will be batched type.

On-line process is the key-press-events of controls when a user(customer) try to write anything in the Bank Account Management System. If the user tries to make a transaction to another account after filling all the details in the respective form, then batch processing will occur. The manager, employee and admin module are also on-line process but in those modules, batch processing isn't that much frequent.

If a user exit from the system, then batch processing will save all the data in the table permanently and commits the tables of the database.

PLANNING AND SCHEDULING:

PROJECT CONTROL SYSTEMS

The purpose of controlling a project is to monitor the progress of the activities against the plans, to ensure that the goals are being approached and eventually achieved. Other aspects of control are to detect, as soon as possible, when deviations from the plan are occurring so that corrective action may be taken. Most project control techniques are based on breaking down the goal of the project into several Intermediate goals. Each Intermediate goal can turn be broken further. This process can be repeated until each goal can turn be broken further. This process can be repeated until each goal is small enough to be understood. We can plan for each goal individually – its resource requirements, assignments of responsibility, scheduling, etc.

Two general scheduling techniques are GANTT charts and PERT Charts as discussed below.

GANTT CHART:

A bar chart is perhaps the simplest form of formal project management. The bar chart also known as GANTT CHART is used almost exclusively for scheduling purpose and therefore controls only the time dimension of projects. Gantt chart is a project control technique that can be used for several purposes, including scheduling, budgeting and resource planning. A Gantt chart is a bar chart, with each bar representing an activity. The bars are drawn against a time line. The length of each bar is proportional to the length of time planned for the activity. Gantt chart can take different forms depending on their Intended use. They are best for resource scheduling. Gantt charts are useful for resource planning and scheduling. Gantt chart they show the tasks and their duration clearly. However they do not show Inter task dependencies plainly.

PERT CHART:

Unlike the bar chart, PERT can be both cost and a time management system PERT is organized by events and activities or tasks. PERT has several advantages over bar charts and is likely to be used with more complex projects. One advantage of PERT is that it is a scheduling device that also shows graphically which tasks must be completed before others are begun. PERT enable the calculation of a Critical path. Each path and cost associated with each task along a path is calculated, and the path that requires the greatest amount of elapsed time is the Critical path. Calculation of the critical path enables project manager to monitor this series of tasks more closely. PERT controls time and cost during the project the project and also facilities finding the right balance between completing a project on time and completing it within budget. PERT recognizes that projects are complex that some task must be completed before other can be started and that the appropriate way to manage a project is to be defined and control each task. Because projects often fall behind schedule, PERT is designed to facilitate getting back schedule. PERT is based in part on the premise that subjective estimates of the total completion time for a project are usually greatly inferior to the sum of subjective estimates for each task. The PERT chart gives a graphical representation of this information.

Advantages of PERT

- It forces the manager to plan.
- It shows an Interrelationship among the tasks in the project, in particular, clearly identifies the critical path of the project, thus helping to focus on it.
- It exposes all possible parallelism in the activities and thus helps in allocating resources.
- It allows scheduling and simulation of alternative schedule.
- It enables the manager to monitor and control the project.

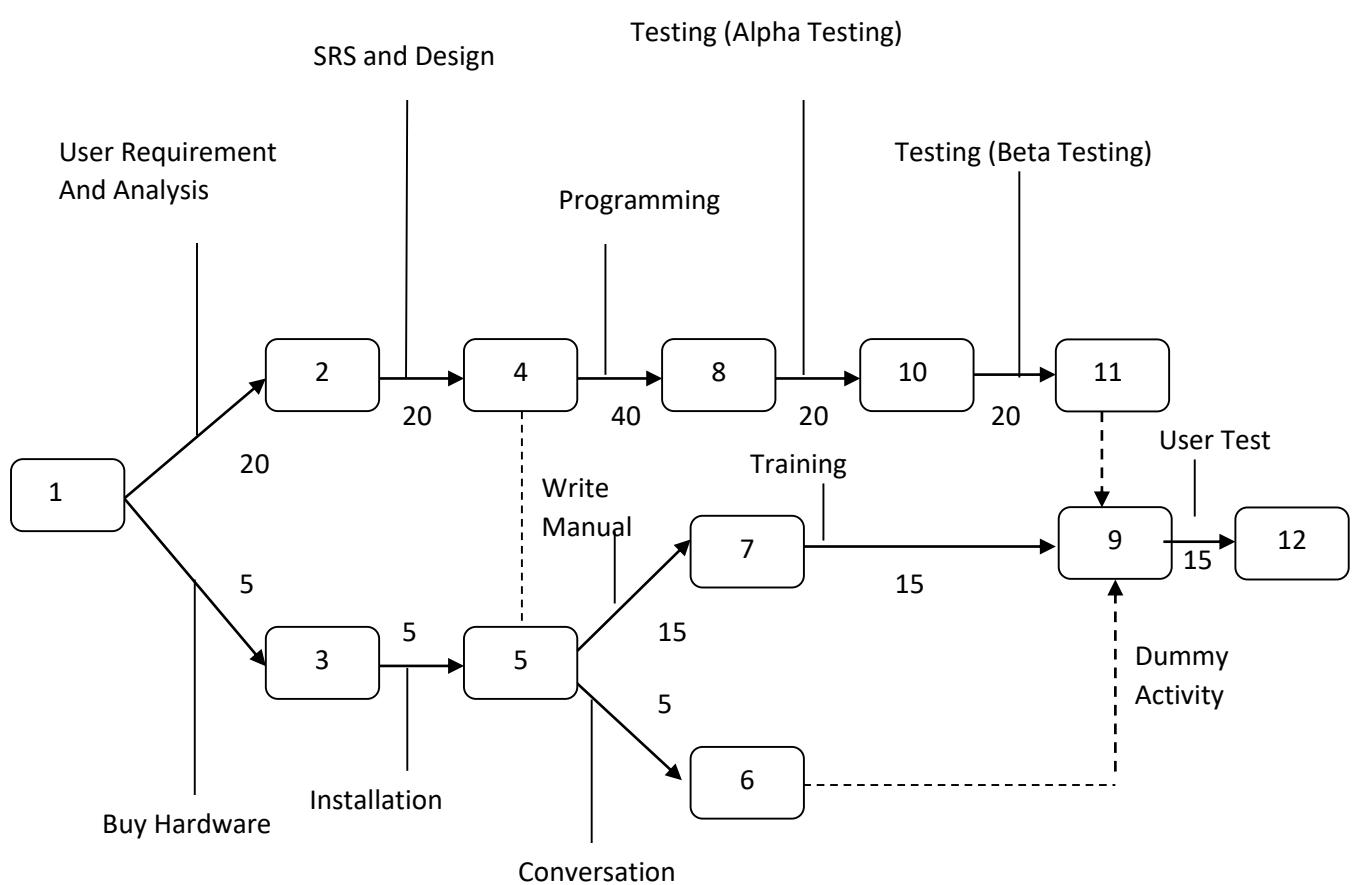
Despite these advantages, PERT is just a tool, and its use does not automatically guarantee the success of the project. Gantt chart can be derived automatically from PERT charts.

The charts are shown in figure A (Gantt chart) and B (PERT Chart).

GANTT CHART

<u>Work Tasks</u>	Month 1	Month 2	Month 3	Month 4	Month 5	Month 6
1. Gathering Information and Requirement Analysis						
2. SRS And Design						
2.1 Creating DFD						
2.2 ER-Diagram						
2.3 Data Dictionary Design						
3 Coding						
4 Testing						
5 Security						
6 Documentation						
7 Training						
8 Tests by User						

PERT CHART



SOFTWAR ENGINEERING APPROACH:

The field of software engineering is related to the development software in systematic manner unlike simple programs which can be developed in isolation and there may not be any systematic approach being followed. As there is large difference between programming and software engineering. As it provides models that lead to the production of well documented software in a manner that is predictable. For a mature process, it should be possible to determine in advance how much time and effort will be required to produce the final product. To develop successful software, I have to follow some models, which act as guidelines.

The model I have used is **Waterfall Model or Classic Life Cycle**. In this model first of all the existed system is observed. Then customer requirements are taken in consideration then planning, modelling, construction and finally deployment.

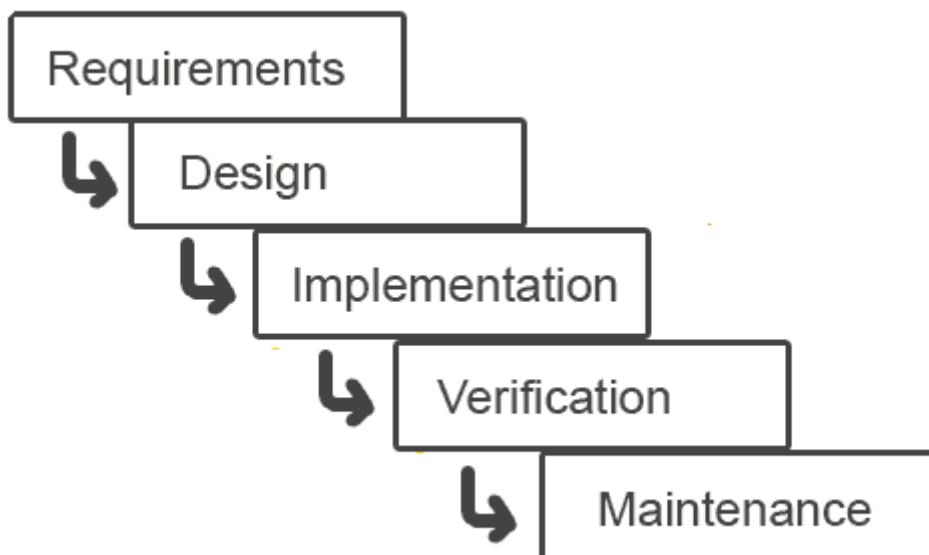


Fig.1. Waterfall Model

SYSTEM DESIGN:

INTRODUCTION:

System design is the specification of a detailed computer-based solution. (Bentley, L, D., & Whitten, J, L (2008)) Also known as physical design.

There were many techniques or approaches that are concerning to the aspect of the machine design and can be categorized as follow:

Model-Driven Approaches

Rapid Application Development (RAD)

Joint Application Development (JAD)

Internet is among the most method for conducting more and more transactions between suppliers and large business because of the speed, flexibility and efficiency that it includes. In this way, new market has been opened to the planet and diffusion of knowledge has been accelerated to the internet. Internet markets or online commercial business has been widely used. Since a special way to design the system must have been by the web-based system and implement it.

Nowadays, internet bank operating system widely used to increase demand of online banking transactions. Internet banking system is looking to provide the best value with highly available, fast, secure and safe to work with. System analysis can be used to analyze and design any system. In such a report, the framework of system analysis and design, system design and system architecture for internet banking system are discussed. Furthermore, about the system architecture is so important so it is among the most foundation of the system analysis and design is also discussed.

Many organizations consider information systems in order to produce useful information by capturing and managing data to employees, customers, suppliers and partners. It's important for his or her ability to compute or gain competitive advantage. Information systems can be classified by the functions such as

Transaction processing systems

Management information systems

Decision support systems

Executive information systems

Expert systems

Communication and collaborative systems

Office automation systems.

Various perspectives can view in information systems such as the players, business drivers, technology drivers and process. From the point of view of system stakeholder, the system analysts bridge the communication gap that can develop between system owners and users and also between designers and builders. System owners are usually executive managers for large systems and may be supervisors for small systems. Unlike system owners, costs and benefits of the system have a tendency to less concern by system users. You will discover two types of system users such as internal system users and external users.

For information systems, system designers are technology specialists such as Database administrators, Network architects, Web architects, Graphic artists, Security experts and Technology specialists. System users' business requirements and constraints are translated by system designer into technical solution. System builder is to create the system in line with the system designer's specifications. System designers and system builders will be the same in small system nonetheless they are often different jobs in large system. Application programmers, system programmer, database programmer, network administrators and security administrators are technical specialties. System owners, users, designers, and builders frequently have different perspective for building and using on any systems.

Steps of System design in Internet banking are as follow:

Firstly, the customer must request the URL.

Customer login the system, then the system checks User ID and Pin No.

After the system check User ID and Pin No. , then your system check that this customer is valid or not.

If it is valid, then that customer need to key in their OTP can access the machine so the customer can see the Main Menu page of the web banking website.

Then the customer can choose from many menus such as viewing USERNAME AND PASSWORDS, Funds Transfer, Payment, Trading and Investment Services, Opening New Account, Remittance, and Update Customer Profile and so forth.

For example, if the customer chooses the Funds transfer menu, then the customer need to choose Funds Transfer type such as Funds Transfer to My very own A/C, Funds Transfer to Other A/C and Funds Transfer to Other Bank.

Then customer needs to choose From Account, To Account and Amount.

After that, submit these details to the system.

And then ensure the detailed information and click Confirm button to accomplish the transaction.

Key in his/her iB Secure PIN(for OTP) to complete this transaction

After logout, customer needs to clear cache for security reason.

5.4 TESTING AND DEBUGGING:

Testing is the process of executing the program with the intent of finding errors and it establishes confidence that the program does what it is supposed to do. It can be done in many ways:

Unit Testing: It is testing of individual module. Before initiating unit testing, it must be ensured that the code is peer reviewed.

Integration Testing: It is performed after all the software units are combined together. The objective here is to test the software interfaces. Project team conducts the integration testing. Before entering integration testing, it may be ensured that code review and unit testing have been performed on the individual software modules.

System Testing: The software is compiled as product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- **Functionality testing** - Tests all functionalities of the software against the requirement.
- **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.
- **Security & Portability** - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

Regression Testing: Whenever a software product is updated with new code, feature or functionality, it is tested thoroughly to detect if there is any negative impact of the added code. This is known as regression testing.

5.5 **REPORT GENERATION:**

Any project or program is required what input it is giving. It is the input, which matters the most and any managements, which is decided for computerization of pay bills for their organization. Spending some money on it does the project designing of the organization. So a cost analysis is also involved to see what benefits the organization can get out of the project.

The Input of the project “**ONLINE BANKING SYSTEM**” has a Main Form containing the list of Different Forms i.e. admin/developer form, manager form, employee form, login form, new user form, personal information form, reporting form, review form, transaction form and many other forms. The Project also includes the salary description of the employees given to their details and the balance description, transaction report(s) of individual customer. The Current system has been made so versatile that any Organization can implement it.

Any project or program is required on what output it is giving. Output is compulsory for any organization for management to take the decision for computerization of their organization. Spending money on it, the organization needs in what respect the project can be benefited, which is possible by viewing the output. So a cost analysis is also involved to see what benefits the organization to give the output of the project.

The System has the facility to view different reports. It also contains the pages, which display the list of employees working in the Organization, their performances etc. which is the output of the project.

Any project or program is required on what output it is giving. Output is compulsory for any organization for management to take the decision for computerization of their organization. Spending money on it, the organization needs in what respect the project can be benefited, which is possible by viewing the output. So a cost analysis is also involved to see what benefits the organization to give the output of the project.

The System has the facility to view different reports. It also contains the pages, which display the list of employees working in the Organization, their performances etc. which is the output of the project.

6. TOOLS, SOFTWARE & HARDWARE REQUIREMENTS

We have a wide range of options of languages. From these options we can choose appropriate platform/ tools and languages for development of the project. Some of these are as follows :-

Project Category: Web-Based Application

SOFTWARE REQUIREMENTS:

IDE: NetBeans 8.0(or 11.2)

Front End: HTML, CSS, JavaScript, AJAX, Bootstrap

Programming Language: JAVA

Back End: JSP, Servlet, Hibernate

RDBMS: MySQL 8.10

Server: Apache Tomcat 8.0

Browser: Chrome, Firefox etc.(latest version)

Operating System: Windows 7 and above

HARDWARE REQUIREMENTS:

Processor : Intel Pentium, Core duo or more

Ram : 2GB or more

Cache : 512 KB

Hard-disk : 50 GB hard disk recommended

Monitor, Keyboard & mouse.

7. Are you doing this project for any Industry/Client? Mention Yes/No.

NO.

8. FUTURE SCOPE AND ENHANCEMENT OF PROJECT

This project was developed to fulfill user requirement, however there are lots of scope to improve the performance of the Banking System in the area of user interface, database performance, and query processing time. Etc.

So there are many things for future enhancement of this project. The future enhancements that are possible in the project are as follows.

- Interest calculation system is not implemented yet. It can be enhanced later.
- Linking and integration of any legacy system for accounting.
- Integration with other bank and government agencies through Web Services
- Connection to third-party OLAP applications
- Electronic Data Interchange (EDI) system for ATM machine
- Web Interface for net banking.
- In the area of data security and system security.
- Loan Account creation and management.

PROJECT REPORT

TABLE OF CONTENT

S. No.	CONTENT	PAGE NO.
1	TITLE OF THE PROJECT	44
2	INTRODUCTION OF THE PROJECT	45 - 46
3	OBJECTIVES OF THE PROJECT	47 - 48
4	TOOLS/ENVIRONMENT USED	49 - 50
5	ANALYSIS DOCUMENT	51 - 79
6	DESIGN DOCUMENT	80 - 87
7	PROGRAM CODE	88 - 301
8	TESTING	302 - 308
9	INPUT AND OUTPUT SCREENS	309 - 330

10	IMPLEMENTATION OF SECURITY	331 - 334
11	LIMITATIONS OF THE PROJECT	335
12	FUTURE APPLICATION OF THE PROJECT	336
13	BIBLIOGRAPHY	337

TITLE OF PROJECT

ONLINE BANKING SYSTEM



INTRODUCTION OF THE PROJECT

Online banking allows a user to conduct financial transactions via the Internet. Online banking is also known as Internet banking or web banking.

Online banking offers customers almost every service traditionally available through a local branch including deposits, transfers, and online bill payments. Virtually every banking institution has some form of online banking, available both on desktop versions and through mobile apps.

With online banking, consumers aren't required to visit a bank branch to complete most of their basic banking transactions. They can do all of this at their own convenience, wherever they want—at home, at work, or on the go.

Online banking requires a computer or other device, an Internet connection, and a bank or debit card. In order to access the service, clients need to register for their bank's online banking service. In order to register, they need to create a password. Once that's done, they can use the service to do all their banking.

Banking transactions offered online vary by the institution. Most banks generally offer basic services such as transfers and bill payments. Some banks also allow customers to open up new accounts and apply for credit cards through online banking portals. Other functions may include ordering checks, putting stop payments on checks, or reporting a change of address.

Online Banking System is a multifunctional software and hardware that enables bank customers to complete and submit to the Bank for execution documents for payment and other documents, monitor the status of their accounts, and receive a wide range of relevant financial information without directing the bank.

There is no need to bring to the Bank payment and other documents on paper – documents in electronic form have a similar force, and instead of the usual signature on them digital signature is used.

Using the Online Banking System allows managing the financial flows of the customer by accessing the account from home, office or any other workplace and significantly reduces the cost of working time associated with a visit to the Bank.

Using the services offered by online banking is simple and easy. Many find transacting online a lot easier than visiting the branch for the same.

Using Online Banking System, we can

1. Transfer funds in a matter of seconds via secure channel.
2. View all the previous transactions acknowledgements.
3. Add, Remove or Update customer, employee or manager respectively.
4. Track multiple transactions or login time for customer.
5. Register a branch for allowing net banking services for customers.

OBJECTIVES OF THE PROJECT

The main aim of designing and developing this Online Banking System, JSP primarily based Engineering project is to provide secure and efficient net banking facilities to the banking customers over the internet. Apache Server Pages, MYSQL database used to develop this bank application where all banking customers can login through the secured web page by their account login id and password. Users will have all options and features in that application like get money from other account, money transfer to others, and send cash or money to inter banking as well as other banking customers by simply adding them as payees.

It basically consists of 4 modules: customer, employee, manager and bank admin.

The other 3 modules other than customer module are used to manage the banking system via internet. These modules prove some useful services like:

- Add, Remove or update customers, employee or manager.
- Track customer transactions, Freeze/Un-freeze customer account (in employee module).
- Register a branch for Online Banking to provide services to its customers (in bank admin module).

The Traditional way of maintaining details of a user in a bank was to enter the details and record them. Every time the user needs to perform some transactions he has to go to bank and perform the necessary actions, which may not be so feasible all the time. It may be a hard-hitting task for the users and the bankers too. The project gives real life understanding of Online Banking System and activities performed by various roles in the supply chain. Here, we provide automation for banking system through Internet. Online Banking System project captures activities performed by different roles in real life banking which provides enhanced techniques for maintaining the required information up-to-date, which results in efficiency. The project gives real life understanding of Online Banking System and activities performed by various roles in the supply chain.

The primary aim of this “ONLINE BANKING SYSTEM” is to provide an improved design methodology, which envisages the future expansion, and modification, which is necessary for a core sector like banking. This necessitates the design to be expandable and modifiable and so a modular approach is used in developing the application software. Anybody who is an Account holder in this bank can become a member of ONLINE BANKING SYSTEM. He has to fill a form with his personal details and Account Number.

TOOLS / ENVIRONMENT USED

We have a wide range of options of languages. From these options we can choose appropriate platform/ tools and languages for development of the project. Some of these are as follows:-

Project Category:

Web-Based Application Available Technologies:

Languages: HTML, JSP and Servlet, JavaScript, AJAX

RDBMS: MySQL

Web Server: Apache Tomcat 8.5

Development Platform: Apache NetBeans 11.0

Tools Used:

Editor Used: NetBeans for JSP and Servlet, WAMP server for MySQL

Framework: Hibernate (ORM tool)

Operating System: Windows XP, Vista, 7, 8, 10

Hardware Used:

Processor: AMD A10 Quad-core

RAM: 8 GB

Hard Disk: 1 TB

Programming Languages: JAVA

Relational Database: MySQL

Others Technology: HTML, CSS, JavaScript, Ajax, JQuery, Bootstrap.

SOFTWARE REQUIREMENTS:

Operating System : Windows 7, 8, 10
Front End : JSP
Back End : MySQL
Markup Languages : HTML
Other Technologies : CSS, JQuery, Ajax, JavaScript,
Bootstrap, Servlet, Hibernate

HARDWARE REQUIREMENTS:

Processor : Intel or AMD processor with min. 2GHz
Ram : 1GB or more
Cache : 512 KB
Hard disk : 50 GB hard disk recommended
External Devices : Monitor, Keyboard & mouse

ANALYSIS DOCUMENT

1. ANALYSYS STUDY:

Systems analysis the process of observing systems for troubleshooting or development purposes. It is applied to information technology, where computer-based systems require defined analysis according to their makeup and design.

An Online Banking System is almost a necessity in day to day services if all parts of operations are consolidated within one software system so that:

- Manual files are not needed
- Communication between different banks and their branches is facilitated through an integrated database
- Functionality is improved
- Control is facilitated
- Security is enhanced

Analysis study is presented in the form of Software Requirement Specification. Review of SRS is conducted to determine the suitability and the adequacy of the software requirement. The review addresses the following questions/issues:

- Are the requirements appropriate to the user needs or project objectives?
- Are the requirements complete?
- Are the requirements defined unambiguously?
- Are the requirements self-consistent?
- Is every requirement testable?

2. USER REQUIREMENTS:

One must know what the problem is before it can be solved. General approaches for determining user requirements are:

- Preliminary investigation – asking general questions
- Analysis of existing System – getting information from existing System

2.1. PRELIMINARY INVESTIGATION:

For this, the need arises to understand the viewpoint of two important entities:

- Top management
- Users

In order to gather pertinent information, I interviewed the Top Management (employees and manager) and asked the following questions:

- How the present Banking System works and what are its drawbacks?
- What is their vision about the new System and what new facilities they want from the new System?
- How will data flow in the System?
- Who will be authenticated to access data and his/her access rights?
- Will users be accustomed to the new system?

To find more about present System's working mechanism such as the ways of getting inputs and providing outputs, I interviewed the Current Users of the System or customers by asking following question:

- Are they comfortable with the present System and flaws exist in it?
- Do they feel the necessity of new System?
- What will be their requirements from new System?
- Are they satisfied with their services in present System?
- Will they be comfortable to use the new System?

After carrying out these interviews, I drew conclusion about the Top Management's requirements and Users are in support of the new System.

2.2. ANALYSIS OF THE EXSISTING SYSTEM:

The existing version of the software was created in JSP, Servlet and MySQL.

This version will also have the same software and hardware specification as the existing one but certain new features will be introduced such as:

- Transaction Management
- Customer A/c management
- Block and/or resume transaction services
- Manage Employees

2.3. SYSTEM REQUIREMENT:

The techniques which were used to collect data in order to determine the System requirements:

- Reviewing organization documents
- Onsite observations
- Conducting interviews

2.3.1. REVIEWING ORGANIZATION DOCUMENTS:

I first learnt about how the bank's involved in the project. I then, got to know how the department works and the employees were directly involved with the application. Customer records and employee records and the documentation of their system helped me to understand the working of the system.

2.3.2. ONSITE OBSERVATIONS:

It is a process of recognizing and observing people, objects and their occurrence to obtain the information. The major objective of the Onsite Observation is to get as close as possible to real System being studied.

Here, I observed the activities of the System directly. I saw the office environment, workload on the System and on the users. The physical layout of the current System along with the location and movement of staff was analysed. In this way, the information about the present workflow, objects and people was gathered.

This helped me to understand various procedures & processes, which were to be developed in the new System.

2.3.3. CONDUCTING INTERVIEWES:

Written documents and onsite observation just tell that how the System should operate. They do not include enough details to allow a decision to be made about the merits of System proposal and do not present the user views about the current system.

I conducted interviews of the staff, which were directly involved with the application. Also the regular users of the application were

interviewed. Based on their viewpoints, crystal clear System requirements were jolted down.

2.4. HARDWARE REQUIREMENTS:

Processor : Intel or AMD processor with min. 2GHz

Ram : 1GB or more

Cache : 512 KB

Hard disk : 50 GB hard disk recommended

External Devices : Monitor, Keyboard, mouse

132 columns High Speed Dot Matrix Printer with local language support.

2.5. COMMUNICATION INTERFACE:

The software may either be installed on a client/server-based setup with a Local Area Network (using the Ethernet interface, one to one connection & TCP/IP protocols) or on a stand-alone machine whereby client and server components reside on the same machine.

A printer shall be used frequently. For this purpose, Dot Matrix printer is the minimum requirement. A line printer should prove to be more efficient.

Authenticated Reports can be generated using a Laser Printer. The software shall be independent of printer type. However dot matrix printer shall provide reports.

SOFTWARE REQUIREMENTS:

Operating System	:	Windows 7, 8, 10
Front End	:	JSP
Back End	:	MySQL
Markup Languages	:	HTML
Other Technologies	:	CSS, JQuery, Ajax, JavaScript, Bootstrap, Servlet, Hibernate

3. FEASIBILITY STUDY:

Feasibility Study is the test of the System proposal according to its workability, impact on the current System, ability to meet the needs of the current users and effective use of the resources.

Its main objective is not to solve the problem, but to acquire its scope. It focuses on following:

- Meet user requirements
- Best utilization of available resources
- Develop a cost effective System
- Develop a technically feasible System

There are three aspects in the feasibility study:

- **Technical Feasibility**
- **Economical Feasibility**
- **Operational Feasibility**

3.1. TECHNICAL FEASIBILITY

Issues to be studied are, whether the work for the project will be done with current equipment, existing S/W technology and available personnel? If the new technology is required, then what is the likelihood that it can be developed?

This software is technically feasible. The primary technical requirement includes the availability of Windows 7 or higher version of operating Systems installed in the network. MySQL is also required which was already installed. To develop programs NetBeans 8.1 or higher was required which was also available. Reliability, access power and data security was also available. Thus, through all the ends technical feasibility was met.

3.2. ECONOMICAL FEASIBILITY

Issues to be studied are, whether the new System is cost effective or not? The benefits in the form of reduced cost?

This software is economically feasible. As the hardware was installed from quite beginning, the cost on project of hardware is low. Similarly, the software loaded for this project was used for many other applications. The software cost was under budget. As student trainees were developing the application, there were no major personnel costs associated. Moreover, the technical requirements were already available so there was no further expenditure for buying software packages.

3.3. OPERATIONAL FEASIBILITY

Issues to be studied are, is there sufficient support for management and users? Is the current method acceptable to users? Will the proposed system cause any harm?

This software is operationally feasible. This application provides the necessary information to the user such as how to enter the information regarding different operations performed on the database. The application was planned in such a way that no prior knowledge was required to go through the various operations. The user just needed to have the basic knowledge of computers.

4. SOFTWARE REQUIREMENT SPECIFICATION:

Among all the documents produced during a software development life cycle, writing the SRS document is probably the toughest. One reason behind this difficulty is that the SRS document is expected to cater to the needs of a wide variety of audience. Different people need the SRS document for very different purposes.

Characteristics of a Good SRS Document

Some of the identified desirable qualities of the SRS documents are following:-

Concise: The SRS document should be concise and at the same time unambiguous.

Structured: The SRS document should be well structured.

Black-box view: It should only specify what the System should do and refrain from stating how to do.

Conceptual integrity: The SRS document should exhibit conceptual integrity so that the reader can easily understand the contents.

Response to undesired events: The document should characterize acceptable responses to undesired events.

Verifiable: All requirements of the System as documented in SRS document should be verifiable. This means that it should be possible to determine whether or not requirements have been met in an implementation.

The SRS of Online Banking System is as follows:

4.1. INTRODUCTION:

4.1.1. PURPOSE:

The purpose of this document is to present a detailed description of the Online Banking System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be liable for the approval or disapproval of the project by the community of the Bank.

4.1.2. SCOPE:

An online banking system will be applicable everywhere, where banking exists. It will be more efficient and easier way to have a record on systems through which everyone can easily access it according to his rights as compared to the traditional banking system. Every bank will prefer the online banking system instead of the traditional banking system as it contains many useful features and fastest methods for the transactions.

4.1.3. DEFINITIONS AND ABBREVIATIONS:

Following are the definitions for the jargoned words.

TERMS	DEFINITIONS
MySQL	Structure query language for the database purposes. Used to define procedures to store and retrieve data.
Customer	A lay person who needs the system to do his task efficiently and effectively. An account holder or a bank's website visitor.
Database	Collection of all the information monitored by this system.
JSP	JSP Hypertext Pre-processor, A server side scripting language, is used to connect the html with the databases.
Hibernate	An ORM (Object Relational mapping Tool) used to map objects of a class in Java to the relational table.
Employee / Account teller	Bank staff that provides information about an account to the user who visits the bank branch physically.
Computer Systems	Computers, which will be used as clients to access the server database according to its right.
Visitors	Anyone visiting the site.
Bank Features	All the benefits and characteristics that bank provide. These features will be explained to the new comer visiting the website without an account.
Administrator / Manager	A person that will be responsible for the addition and deletion of the staff members from the general database of the system.
SRS	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.
Stake Holder	Any person with an interest in the project who is not a developer.

4.1.4. REFERENCES:

This web application has been prepared on the basis of discussion with Team members, faculty members and also taken information from following books & website.

- Websites:
 - www.google.co.in
 - www.wikipedia.org
 - www.youtube.com
- Books:
 - Fundamental of Software Engineering by Rajiv Mall.
 - Software Engineering: A practitioner's approach Ed. by Pressman, Roger.
 - Software Engineering Seventh Edition Ian Somerville.
 - Software Engineering Ed.2 by Jalota & Pankaj.
 - Schaum's Series, "Software Engineering".

4.2. GENERAL DESCRIPTION:

4.2.1. PRODUCT PERSPECTIVE:

In traditional system, customer should have to visit the Bank branch physically for the transactions or some other task. It wastes time.

After implementing the online banking system customer will be able to connect to his account through the internet connection. Time usage will be minimized, task will be done fast instead of waiting someone other to complete his task.

4.2.2. FUNCTIONALITIES:

a) Online balance check and transaction information:

Customer will be able to check his balance online while sitting at home by accessing the database of the bank using his/her password and account no. allotted him by the bank.

b) Save or view past history of transactions:

It will be easy for the customer to view or save his history of all transactions. It will provide him/her the opportunity to maintain his bank balance and needs.

c) Balance transfer:

This system will provide a path to the customer of the bank to transfer his balance to other account in easy steps. A small transfer fee will be applicable for this transaction.

d) Online record Entry:

Bank staff will input and maintain their record online. It will be easy and efficient for them to serve more and more people in less time.

e) Online record search:

Bank staff will easily search a record and update it if needed. Transactions will be faster even physically from the branch because it will be very easy for the bank staff to check the balance of a specific person and update its record if necessary.

f) Online Billing Option:

Customers will be able to shop online and pay the bills from their account. A secure way will be provided for the billing. Online shopping will provide them the easiest way to buy and sell their items.

4.2.3. USER CHARACTERISTICS:

There are various kinds of users for the product. Usually web products are visited by various users for different reasons.

The users include:

- a) Chancellor who will be acting as the controller and he will have all the privileges of administrator.
- b) Manager who will handle the operations of the bank of that particular branch.

- c) Employee who will handle the request's and queries of the customer.
- d) Customer are those who will perform banking.

4.2.4. GENERAL CONSTRAINTS:

Some general constraints should be defined which will have a great part in the overall succession of the online banking project.

a) HARDWARE REQUIREMENTS:

As this system is an online Web-based application so a client server will be the most suitable Organizational style for this system. Computer systems will be needed by each of the actor as well as that user must be connected to the internet. So, concisely following hardware will be needed.

- 1) Computer systems
- 2) Internet availability

b) SAFETY AND SECURITY:

This Project must be safe and secure because customers will directly contact their account through the internet. Software will have to identify the valid customer according to his/her bank details and password. So it is a difficult task to prevent the system by major disasters by preventing the unauthorized access to the system.

4.2.5. ASSUMPTIONS AND DEPENDENCIES:

Following are the assumptions and dependencies which are related to this online banking project.

- This project is a stand-alone project so it will not affect the system where it will be embedded.
- This project is a web-based project while the staff was addicted of using traditional methods of data storage and retrieval so they will be trained a bit to jump to it.

- This system will not depend on any other module. It will be a web-based so everyone will independently contact it.
- It is will not affect the environment at all.
- Banks will feel free to adopt it because it will not be so much expensive.
- As this project contains valuable and new features so it will probably remove the previous online banking systems embedded in some banks.

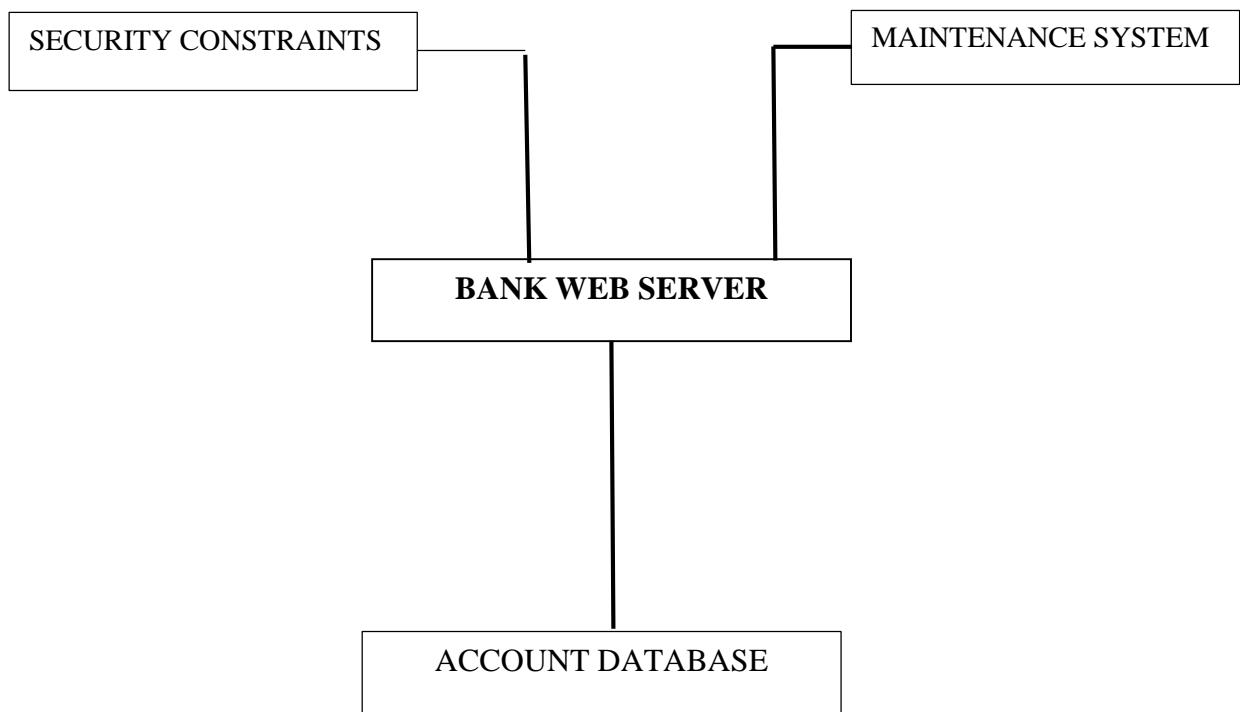
4.3. SPECIFIC REQUIREMENTS:

How the online banking will interact with the environment, what will be the functional and non-functional requirement. These all the steps should be defined here for providing a powerful base to the design phase. The design of the project will completely depend on the functional and non-functional requirements. So these should be defined clearly and accurately for the effectiveness.

4.3.1. FUNCTIONAL REQUIREMENTS:

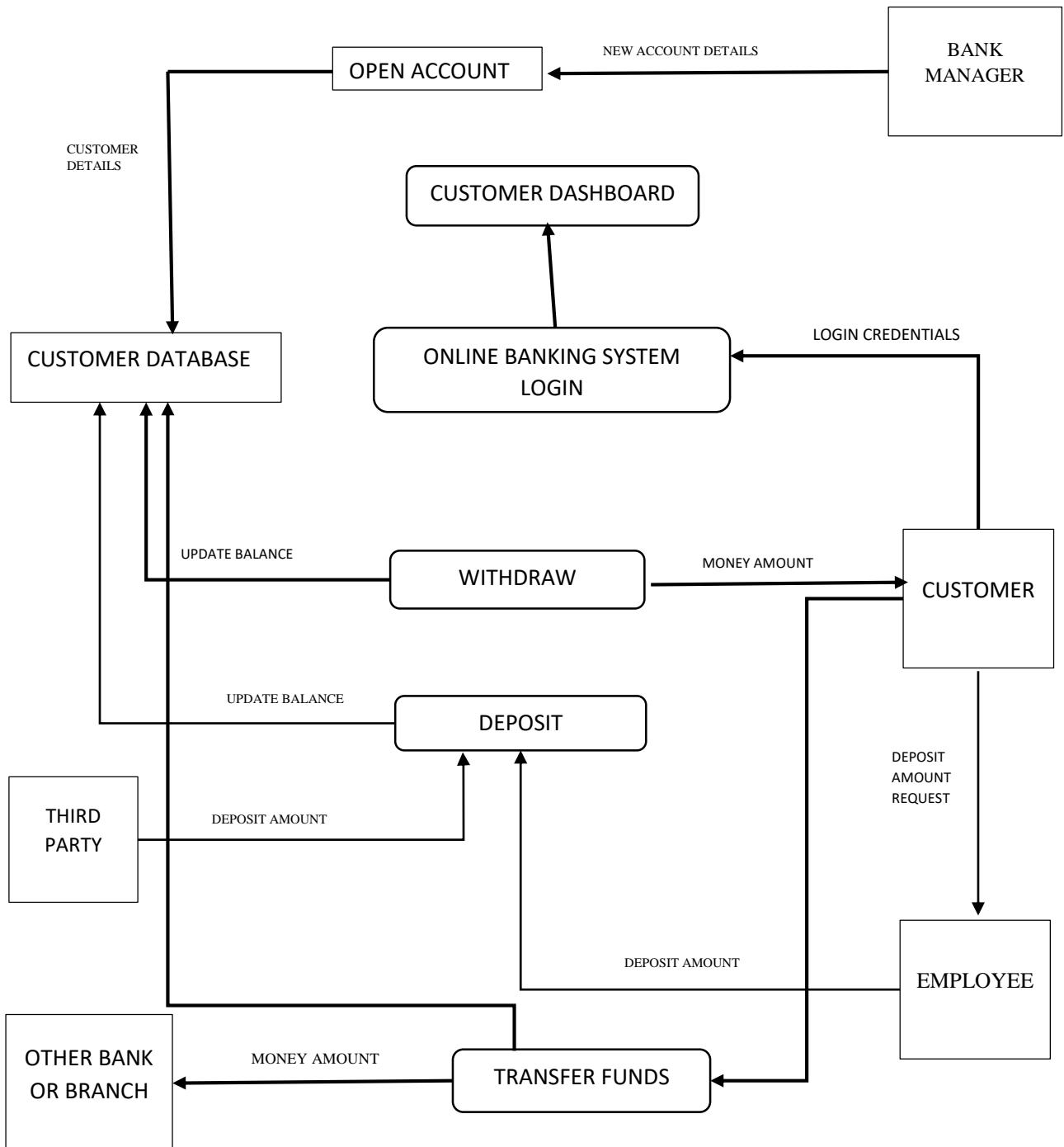
Following are the services which this system will provide. These are the facilities and functions required by the customer.

- i. Online balance check.
- ii. Online shopping opportunity.
- iii. Online data entry by the staff.
- iv. Updating the data.
- v. Balance transfer.

4.3.1.1. CONTEXT DIAGRAM AND EXTENT LIST:

4.3.1.2. DATAFLOW DIAGRAM:

Following is the data flow diagram for the online banking system:



4.3.1.3. PROCESS SPECIFICATION:

All the process mentioned in the DFD are described as below.

- ◆ **ONLINE BANKING SYSTEM LOGIN:** Each Customer will have its account Id and password. This page will require both of these attributes for them to access their account.
- ◆ **OPEN ACCOUNT:** A customer can open an account by filling a form in the bank and providing the necessary details. The bank manager will verify the details and allow the customer to open a bank account. The employee will fill the necessary details into the customer database. The customer will only have to register for online banking to avail the services via internet.
- ◆ **CUSTOMER DASHBOARD:** On successful login, the customer will be redirected to the welcome page (or the customer dashboard) which allows customer to use the said services.
- ◆ **WITHDRAW:** Customer can withdraw money from their account from the bank which will automatically get reflected in the database.
- ◆ **DEPOSIT:** Customer can deposit money to their account by depositing the money to the depositor (employee), who will deposit the amount in the respective account.
- ◆ **TRANSFER:** Customer can transfer money to another account via Online Banking.
- ◆ **STAFF LOGIN:** On the Website main page, a staff login link will also be provided. Bank staff will use to input their ID's and passwords to access their account. Here the

type of staff will also be recognized, if he will be of administration block, he will be sent to the administration module else he will be sent to the record management module.

4.3.2. EXTERNAL INTERFACE REQUIREMENTS:

These requirements are discussed under the following categorization:

4.3.2.1. USER INTERFACE:

Application will be accessed through a Browser Interface. The interface would be viewed best using 1024 x 768 and 800 x 600 pixels resolution setting. The software would be fully compatible with Microsoft Internet Explorer for version 6 and above.

No user would be able to access any part of the application without logging on to the system.

4.3.2.2. HARDWARE INTERFACE:

a) SERVER SIDE:

- ◆ **Operating System:** Windows 7 or higher.
- ◆ **Processor:** Pentium 3.0 GHz or higher.
- ◆ **RAM:** 256 Mb or more.
- ◆ **Hard Drive:** 10 GB or more.

b) CLIENT SIDE:

- ◆ **Operating System:** Windows 7 or above, MAC or UNIX.
- ◆ **Processor:** Pentium III or 2.0 GHz or higher.
- ◆ **RAM:** 256 Mb or more.

4.3.2.3. SOFTWARE INTERFACE:**a) CLIENT SIDE:**

HTML, JavaScript, Web Browser, Flash Player, MS Office, Windows XP or higher.

b) WEB SERVER:

HTML, MS Office, Windows XP or above.

4.3.2.4. COMMUNICATION INTERFACE:

The Customer must connect to the Internet to access the Website:

- Dialup Modem of 52 kbps.
- Broadband Internet.
- Dialup or Broadband Connection with an Internet Provider.

4.3.3. NON-FUNCTIONAL REQUIREMENTS:

Those requirements which are not the functionalities of a system but are the characteristics of a system are called the non-functionalities. Every software system has some non-functionalities. Just fulfilling the requirements of the user is not a good task, keeping the system accurate, easy to maintain, reliable and secure is also a basic part of software engineering. Online Banking System must have the following non-functional requirements so that I could be said as a complete system.

a) PERFORMANCE CONSTRAINTS:

This system must be fit according to the performance wise. It should use less memory and will be easily accessible by

the user. Memory management should be done wisely so that none of the memory part goes wasted.

b) HARDWARE LIMITATIONS:

It should be designed in such a way that cheap hardware must be installed to access and use it effectively. It should be platform independent. There should be no hardware limitations. It should be designed to work with the low specification hardware so that it could easily work with the high specification hardware.

c) MAINTAINABLE:

Each of the modules should be designed in such a way that a new module can easily be integrated with it.

d) RELIABLE:

It should be consistent in its performance and provide secure transactions.

e) TESTABLE:

Each module should be made in such a way to make it easy for all kinds of testing and debugging.

5. DATA FLOW DIAGRAMS (DFD):

Data Flow Diagramming is a means of representing a System at any level of detail with a graphic network of symbols showing data flows, data stores, data processes, and data sources/destinations.

The purpose of data flow diagrams is to provide a semantic bridge between users and System developers.

The diagrams are:

- Graphical, eliminating thousands of words;
- Logical representations, modelling ‘WHAT’ a System does, rather than physical models showing ‘HOW’ it does it;
- Hierarchical, showing System s at any level of detail; and Jargon less, allowing user understanding and reviewing.

The goal of data flow diagramming is to have a commonly understood model of a System. The diagrams are the basis of structured System’s analysis. Data flow diagrams are supported by other techniques of structured System s analysis such as data structure diagrams, data dictionaries, and procedure-representing techniques such as decision tables, decision trees, and structured English.

The objective of Data flow diagrams is avoiding the cost of user/developer misunderstanding of a System, resulting in a need to redo System’s or in not using the System.

Having to start documentation from scratch when the physical System changes since the logical System, ‘WHAT’ gets done, often remains the same when technology changes.

It helps in removing inefficiencies of System because a System gets "computerized" before it gets "Systematized". Also helps enabling to evaluate System project boundaries or degree of automation, resulting in a project of inappropriate scope.

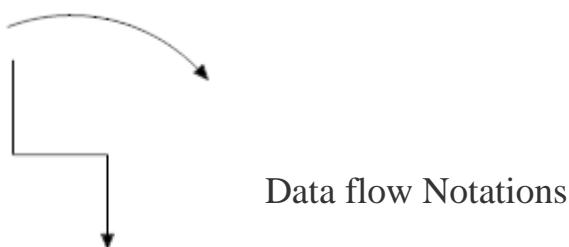
DFD SYMBOLS:

In the DFD, there are four symbols shown in figure below:

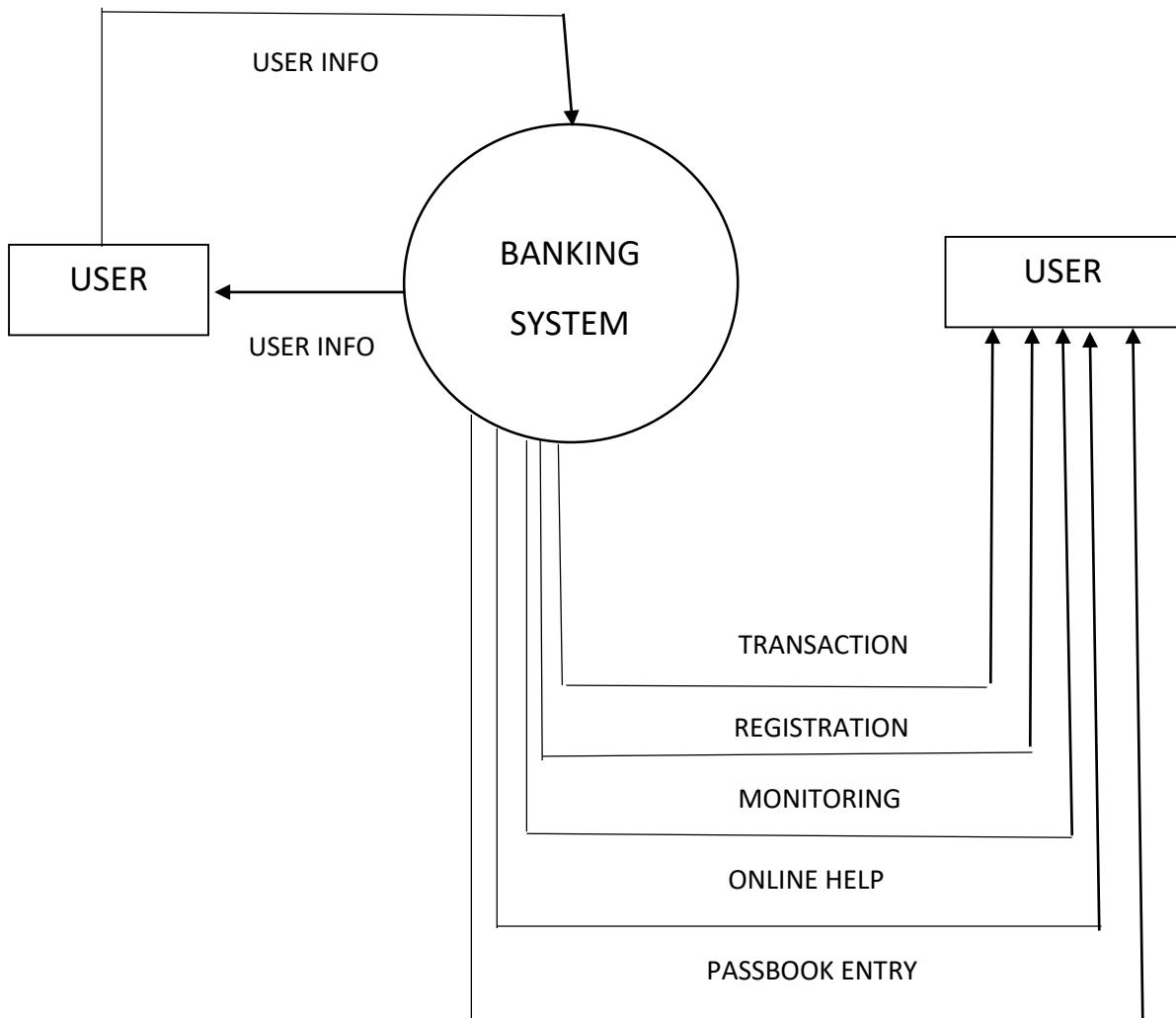
- A **Square** defines a source (originator) or destination of data pipeline through which information flows.
- An **Arrow** identifies data flow data in motion. It is a pipeline through which information flows.

- A **Circle** or a "bubble" (some people use an oval bubble) represents a process that transforms incoming data flow into outgoing data flow.
- An **Open rectangle** is a data store - data at rest, or a temporary repository of data.

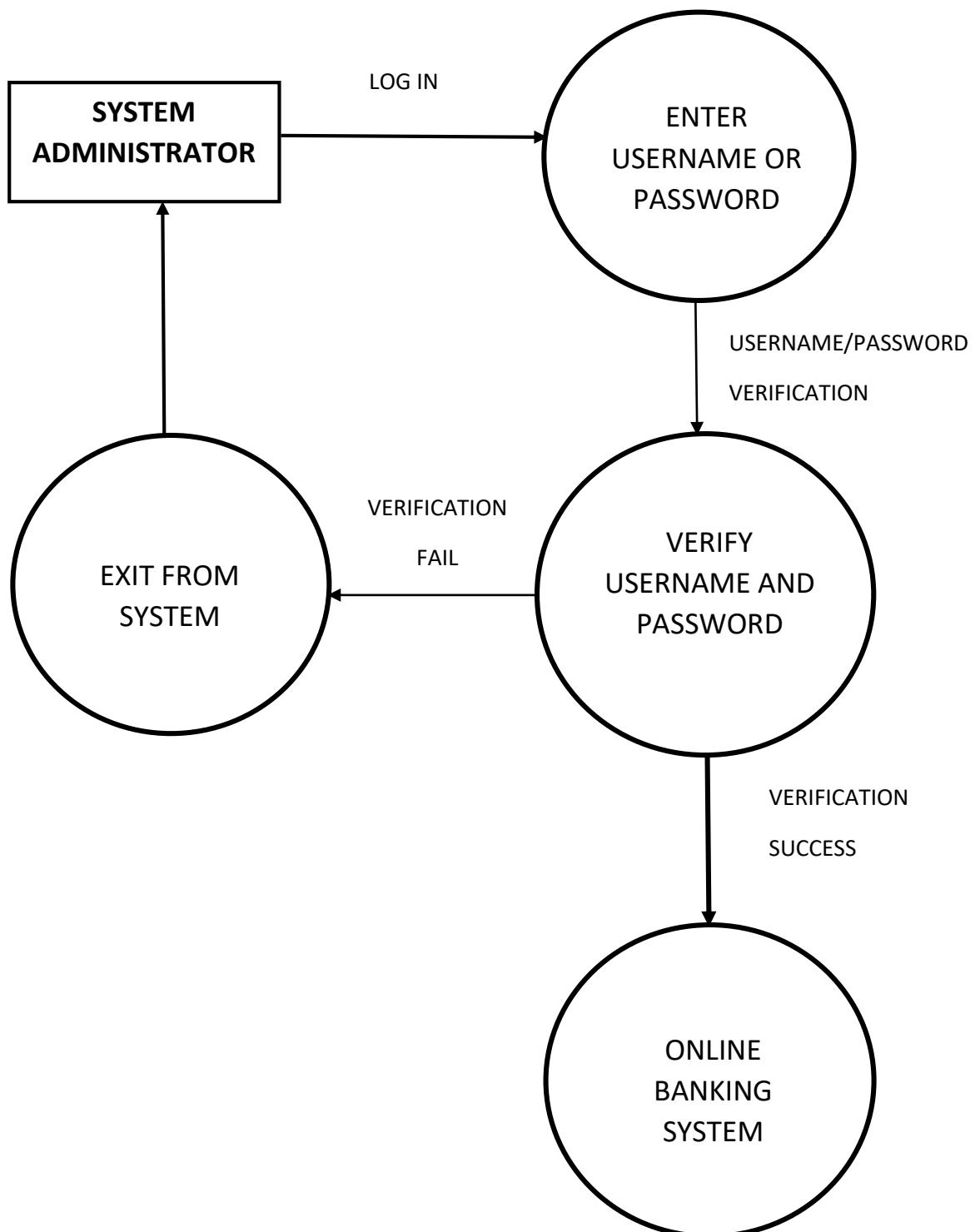
Basic Symbols and their Meaning



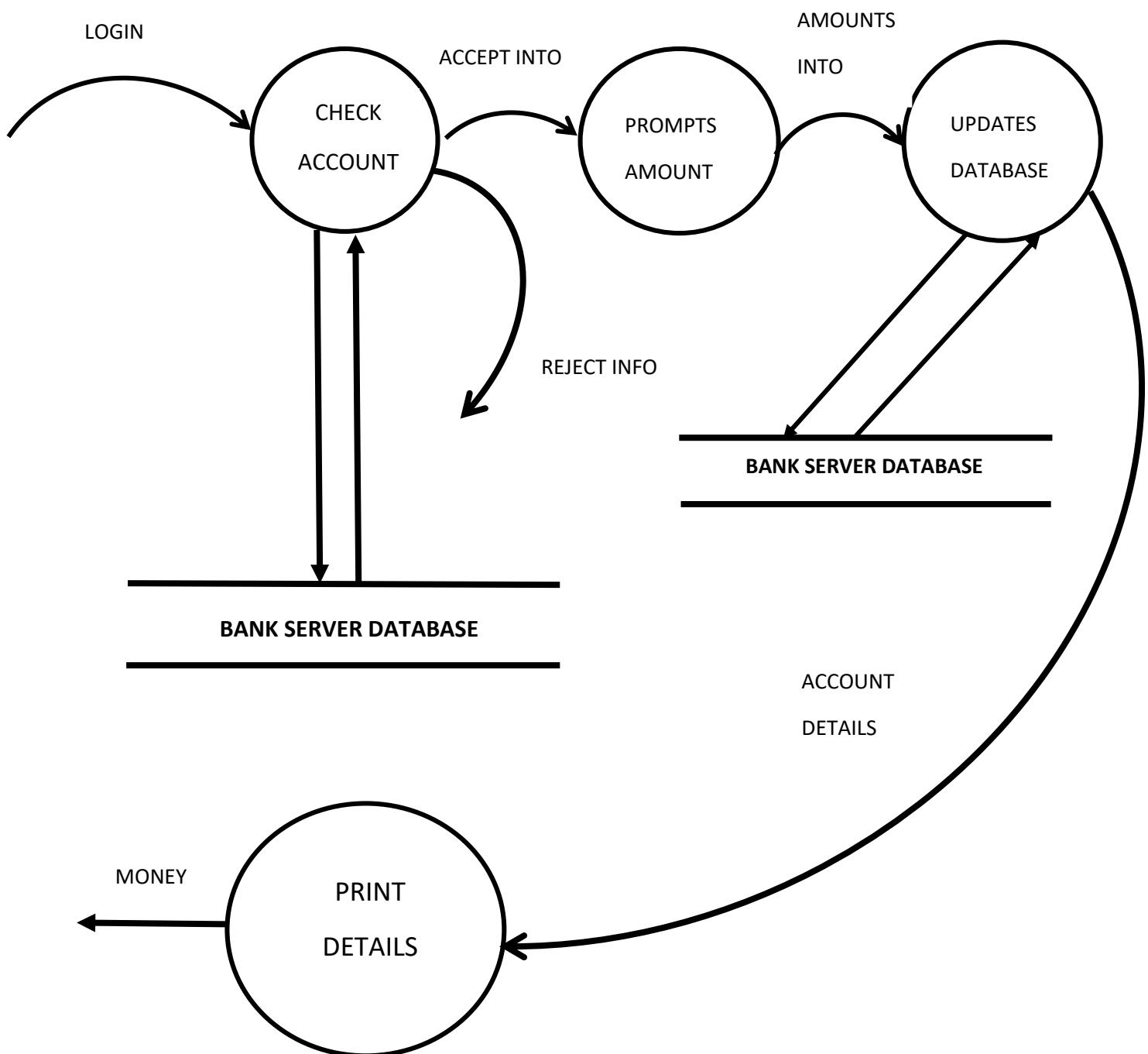
**0-(ZERO) LEVEL DFD (CONTEXT LEVEL DFD) FOR
ONLINE BANKING SYSTEM**



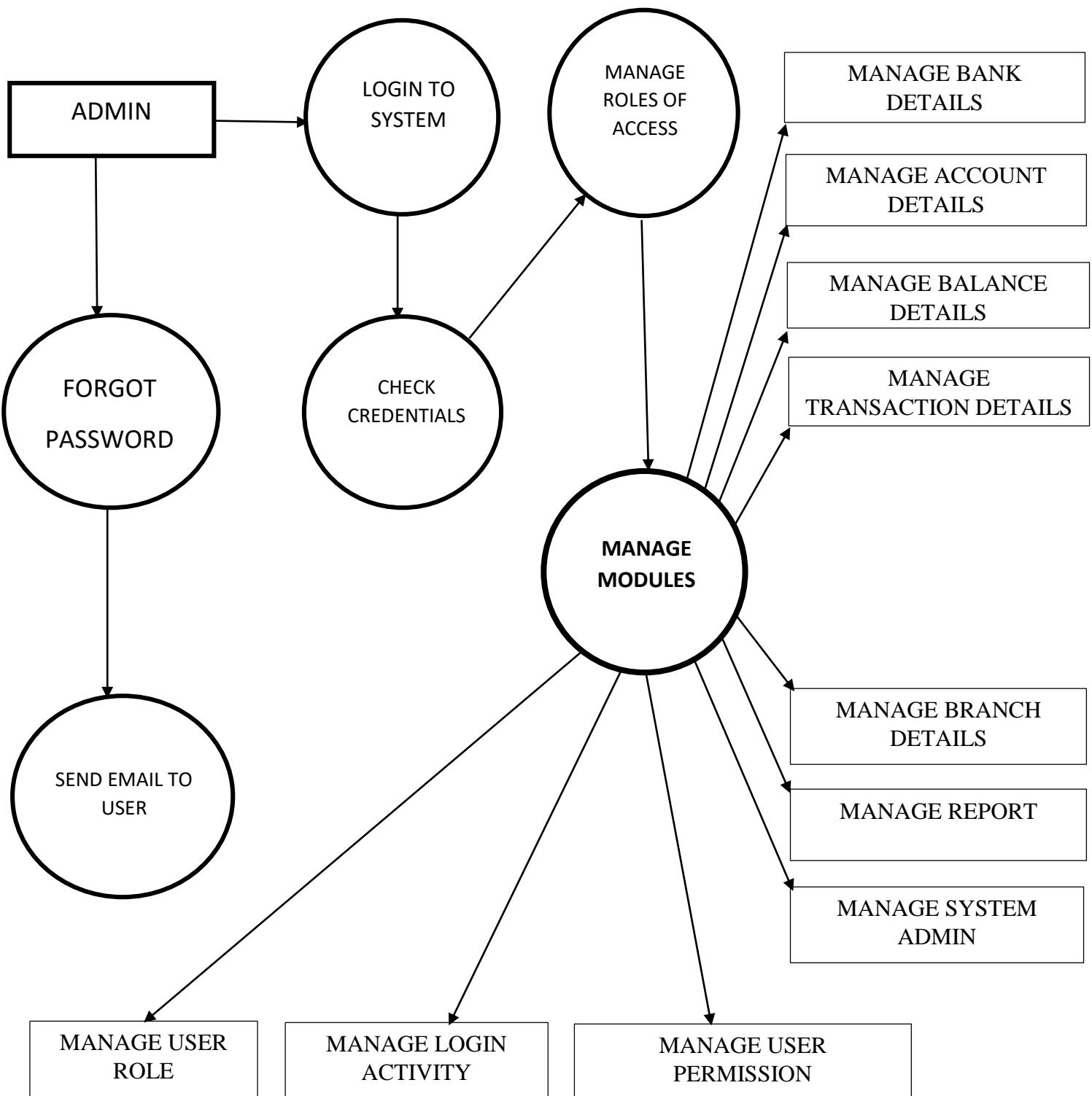
FIRST LEVEL OF DATA-FLOW DIAGRAM FOR SYSTEM LOGIN IN
ONLINE BANKING SYSTEM OF ADMIN LOGIN DETAILS



FIRST LEVEL OF DATA-FLOW DIAGRAM FOR
CUSTOMER TRANSACTION IN ONLINE BANKING SYSTEM



SECOND LEVEL DATA-FLOW-DIAGRAM
OF ONLINE BANKING SYSTEM



6. ER-DIAGRAM:

E-R (Entity – Relationship) Model Diagram.

In software engineering an entity relationship model diagram (E-R model diagram) is a data model for describing a database in an abstract way. An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. It is widely used to develop an initial design of a database. It describes data as a collection of entities, relationship and attributes.

E-R Diagram are composed of:

a) ENTITY:

An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.



ENTITY

b) ATTRIBUTE:

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

- Key attribute
- Composite attribute
- Multivalued attribute
- Derived attribute

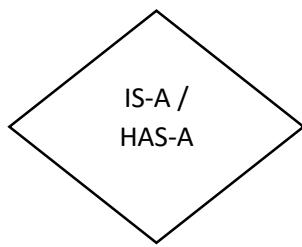


ATTRIBUTE

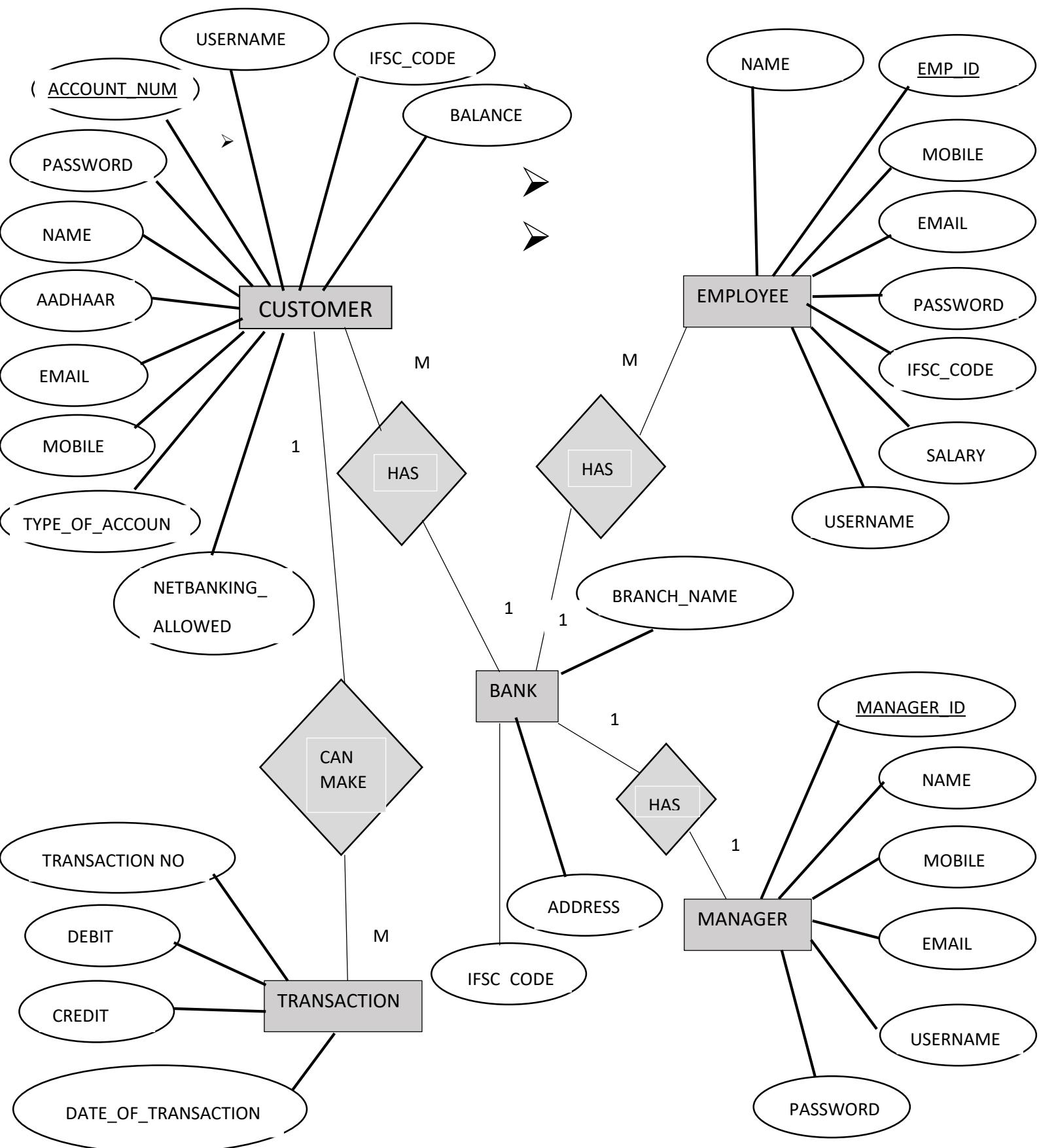
c) RELATIONSHIP:

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:

- One to One
- One to Many
- Many to One
- Many to Many



IS-A /
HAS-A



7. DATA DICTIONARY:

A Data Dictionary is a collection of names, definitions, and attributes about data elements that are being used or captured in a database, information system, or part of a research project. It describes the meanings and purposes of data elements within the context of a project, and provides guidance on interpretation, accepted meanings and representation. A Data Dictionary also provides metadata about data elements. The metadata included in a Data Dictionary can assist in defining the scope and characteristics of data elements, as well as the rules for their usage and application.

Data Dictionaries are useful for a number of reasons. In short, they:

- Assist in avoiding data inconsistencies across a project.
- Help define conventions that are to be used across a project.
- Provide consistency in the collection and use of data across multiple members of a research team.
- Make data easier to analyse.
- Enforce the use of Data Standards.

DESIGN DOCUMENTS

1. MODULARIZATION DETAILS:

The Online Banking System Project consists of 4 functional elements: *Customer transaction module, Employee Transaction Module, Manager Transaction Module* and *Developer/Administration Module*.

1.1. CUSTOMER MODULE:

This module handles all the customer related functionalities. This is a very important module in the Online Banking System. This module performs the following functions:

- Customer Login using username and password.
- Customer online banking registration, username and password generation.
- Shows a customized user interface defined as ‘customer dashboard’.
- Customers can view their details in the dashboard.
- It provides the functionality of fund transfer which can either be in the same bank or a different bank.
- View all the past transactions made by the customer.
- View customer current account balance.

1.2. EMPLOYEE/STAFF MODULE:

This module handles all the customer related queries and requests such as:

- Add a new customer
- Update an existing customer details
- Remove a customer from the system
- Block any and all transaction services in case of suspicious activity. Un-blocking functionality is also available.
- Deposit money to a customer’s account.
- View all transactions of a customer of their own branch.

1.3. MANAGER MODULE:

This module handles all the staff data. It performs the following functionalities:

- Add a new employee
- Update details of an employee
- Remove an employee
- View all the details of each employee

1.4. ADMINISTRATOR MODULE:

This module handles the entire banking system. It performs the following functionalities:

- Register a branch for online banking by adding a new manager to the system.
- Update a property of an existing manager
- View all branches and their managers registered for online banking.

2. DATABASE DESIGN:

Table: CUSTOMER

Field	Type	Null	Key	Default
Account_number	Bigint	No	Primary	Null
IFSCCode	varchar	Yes		null
aadhaar	Varchar	yes		null
Balance	Double	yes		null
Date_of_creation	Date	yes		null
Email	Varchar	Yes		null
Mobile	Varchar	yes		null
Name	Varchar	yes		null
netBankingAllowed	varchar	yes		null
Pan	Varchar	yes		null

Password	Varchar	yes		null
Type_of_account	Varchar	yes		null
Username	Varchar	yes		null
IFSCcode_id	Int	yes	foreign	null

Table: BANK

Field	Type	Null	Key	Default
Id	Int	No	Primary	Null
Address	Varchar	Yes		Null
Branch_name	Varchar	Yes		Null
IFSC	Varchar	Yes		Null
Employee_emp_id	Bigint	Yes	Foreign	Null
Customer_account_number	Bigint	Yes	Foreign	Null

Table: EMPLOYEE

Field	Type	Null	Key	Default
Employee_id	Bigint	No	Primary	Null
IFSCcode	Varchar	Yes		Null
Aadhaar	Varchar	Yes		Null
Email	Varchar	Yes		Null
Mobile	Bigint	Yes		Null
Name	Varchar	Yes		Null
Salary	Double	Yes		Null
username	Varchar	Yes		Null
Password	Varchar	Yes		Null
IFSCcode_id	Int	Yes	Foreign	null

Table: MANAGER

Field	Type	Null	Key	Default
Manager_id	Bigint	No	Primary	Null
Email	Varchar	Yes		Null
Mobile	Varchar	Yes		Null
Name	Varchar	Yes		Null
Username	Varchar	Yes		Null
Password	Varchar	Yes		Null
Id	Int	Yes	Foreign	null

Table: TRANSACTIONS

Field	Type	Null	Key	Default
Transaction_no	Bigint	No	Primary	Null
Debit	Varchar	Yes		Null
Credit	Varchar	Yes		Null
Date_of_transaction	Varchar	Yes		Null

3. PROCEDURAL DESIGN:

Procedural design is often classified as a computational approach relying upon a set of instructions that, when used in a particular sequence, are the generators of form. While within this frame-work certain methods may be iterative and cyclical, procedural design often denotes the construction, conceptually, of a linear solver. The work documented in this section, though, shows a significant evolution of this approach. Intelligent systems are formed in which computation is given the freedom to absorb, interpret, and respond within the sequential set of procedures, thus shifting from linear logics to networked ones. This is addressed through

papers that discuss the language through which such processes are enacted and explore the emergence of a built architecture through dynamic logics of design computation.

Traditionally, procedural design has offered means of testing the relationships of parameters, but the work shown in this section demonstrates an evolution of this approach. Procedural processes become an active agent for resolving the relationships of systems.

This design starts right after the data and its structure has been established.

Procedural details can be represented in different ways:

- **Graphical Design Notation:**

The most widely used notation is the flowchart. Some notation used in flowcharts are:

- i. Boxes to indicate processing steps;
- ii. Diamond to indicate logical conditions;
- iii. Arrows to indicate flow of control;
- iv. Two boxes connected by a line of control will indicate a Sequence.

- **Tabular Design Notation:**

Decision tables provide a notation that translates actions and conditions (described in a processing narrative) into a tabular form.

- i. The upper left-hand section contains a list of all conditions.
- ii. The lower left-hand section lists all actions that are possible based on the conditions.
- iii. The right-hand sections form a matrix that indicates condition combinations and the corresponding actions that will occur for a specific combination.

- **Program Design Language (PDL):**

Program Design Language (PDL) is also called structured English, or Pseudo code. The main difference between PDL and its nearest neighbour, 4th Generation Languages, is that in PDL, the use of narrative text (e.g. English) is embedded directly within PDL statements. PDL have the following characteristics:

- i. A fixed syntax of keywords that provide for all structured constructs, data declaration, and modularity characteristics.
- ii. A free syntax of natural language that describes processing features.
- iii. Data declaration facilities that should include both simple (scalar, array) and complex (linked list or tree) data structures.

4. USER INTERFACE DESIGN:

With time and creativity, developments have been part of technology. With creativity comes advancement and with advancement comes complexity but we still prefer things to be simple yet advanced. This is when we need to think how to simplify things. We need simplicity in product development and we need simplicity in product use. We need something to fulfil this need. User interface design just fits in there. It helps developer/designer to create end product in logical and sequential way. User Interface (UI) design has four main elements:

- Usability
- Visualization
- Functionality
- Accessibility

For making user experience enjoyable there are some things that should be considered before jumping into the development process. This will save developer's time and also designer's work will go smoothly. The end result will be usable.

While creating the UI of the Online Banking System, the points were kept in mind:

- **CONSISTENCY:**

Users do not like inconsistent pages. Inconsistency makes things complex while consistency provides clarity. Some basic elements of an application user interface that designer should be consistent with are colour scheme, style, borders, type and fonts, size, background images and effects.

It helps users remember one's design. It adds the right feel for users to be there. Throw users different typography in different pages, different sizes and colours without any meaning or reason behind, they are going to get bored and never return to that site again. Consistency can be achieved by a thoughtful design, potential end users research, using master pages and CSS while developing.

- **RESPONSIVENESS:**

The basic idea of the site being responsive is the site responding to users actions. It also means giving the users the feeling that we are listening to them.

Nobody likes to talk to a tree. If the page is taking time to load, one can provide some visual graphical representation or any plain text that suggests that the page is loading or telling them the progress status.

Use of heavy graphics can add to the response time of a web page. Therefore, minimizing sizes and use of them as little as possible can help minimize the problem.

Using alternatives also help, for example, if an image has not loaded for some reason, alternative text stating what image it is, helps the user to at least understand the idea behind the problem.

- **FAMILY METAPHORS:**

Use of terms already familiar to users from other existing internet sites helps them to familiarize with the website faster. For example, in general sites, words such as share, signup and login are very common, so using them to the

newly designed pages will help users to skip understanding these parts of the site which will help in minimizing the learning process.

One should always be sure to use metaphors taking cultural boundaries into account.

It is easy to get people offended when cultural differences are not well under-stood. Background research and consult with respective representative should be made to avoid possible tension and misunderstandings.

- **STREAMLINING THE EXPERIENCE:**

Streamlining the experience is about improving the navigation of a site, making navigation more consistent and enjoyable. Use of unnecessary pages or navigation paths should be avoided. The site should be kept simple and navigation clear.

Top level navigation flow should be predefined. Every page could for instance have a banner that on clicking takes the user to the first page or commonly known as home page. Also, all pages should contain link to organization's contact information, for example, bank's customer care information.

PROGRAM CODE

Index.html

```

<!DOCTYPE html>

<html lang="en">

<head>

<title>ONLINE BANKING</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" href="image/logo.png" type="image/x-icon"/>

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<script src="https://kit.fontawesome.com/1a88ea4f70.js" crossorigin="anonymous"></script>

<style>

.navbar-inverse .navbar-nav > li > a,

.navbar-inverse .navbar-nav > li a:hover,

.navbar-inverse .navbar-nav > li a:focus {

color: black;

background-color: #4CAF50;

font-size: 20px;

}


```

```

.navbar-inverse .navbar-nav > .active > a,

.navbar-inverse .navbar-nav > .active > a:hover,

.navbar-inverse .navbar-nav > .active > a:focus {

```

```
<body style="background-color: rgba(183, 189, 163, 0.411); ">
```

```
<nav class="navbar navbar-inverse navbar-fixed-top shadow-lg p-3 mb-5 bg-white rounded" style="background-color: #1E90FF; height: 80px; padding-top: 15px;">  
  <div class="container-fluid">  
    <div class="navbar-header">  
      <a class="navbar-brand" href="#" style="color: #0000CD; font-weight: bold; font-size: 20px;">&nbsp;ONLINE BANKING</a>  
    </div>
```

```

<ul class="nav navbar-nav navbar-right" style="font-family: Georgia, 'Times New Roman', Times, serif; font-size: 20px; ">

    <li><a href="#" data-toggle="modal" data-target="#loginPage"><span class="glyphicon glyphicon-log-in"></span> Login</a></li>

    <li><a href="#" data-toggle="modal" data-target="#signupPage"><span class="glyphicon glyphicon-user"></span> Sign Up</a></li>

</ul>

<ul class="nav navbar-nav navbar-right " style="font-family: Georgia, 'Times New Roman', Times, serif; font-size: 20px; text-align: center;">

    <li class="active"><a href="#">Home</a></li>

    <li class="dropdown"><a class="dropdown-toggle" data-toggle="dropdown" href="#">Admin Login<span class="caret"></span></a>

        <ul class="dropdown-menu" style="text-align: center;padding: 10px;">

            <li><a href="employee/employee_login.html">Employee</a></li>

            <div class="divider"></div>

            <li><a href="manager/manager_login.html">Manager</a></li>

        </ul>

    </li>

</ul>

</div>

</nav>

<div class="container-fluid" style="padding-top: 100px;">

    <div id="myCarousel" class="carousel slide" data-ride="carousel" style="width: 70%; float: left; margin-bottom: 20px;">

        <!-- Indicators -->

        <ol class="carousel-indicators">

            <li data-target="#myCarousel" data-slide-to="0" class="active"></li>

```

```
<li data-target="#myCarousel" data-slide-to="1"></li>
<li data-target="#myCarousel" data-slide-to="2"></li>
<li data-target="#myCarousel" data-slide-to="3"></li>
</ol>

<div class="carousel-inner">
  <div class="item active">
    
  </div>
  <div class="item">
    
  </div>
  <div class="item">
    
  </div>
  <div class="item">
    
  </div>
</div>

<div class="container-fluid col-md-3" style="float: right; text-align: center; padding: 30px; margin-right: 28px;">
  <ul class="nav nav-pills nav-stacked">
```

```
<li class="active"><a href="#">SERVICES</a></li>
<li class="dropdown">
  <a class="dropdown-toggle" data-toggle="dropdown" href="#">ACCOUNTS
  <span class="caret"></span></a>
  <ul class="dropdown-menu" style="width: 100%;text-align: center;">
    <li><a href="#">SAVINGS</a></li>
    <div class="divider"></div>
    <li><a href="#">CURRENT</a></li>
  </ul>
</li>

<li class="dropdown">
  <a class="dropdown-toggle" data-toggle="dropdown" href="#">ATM-CARDS
  <span class="caret"></span></a>
  <ul class="dropdown-menu" style="width: 100%;text-align: center;">
    <li><a href="#">DEBIT</a></li>
    <div class="divider"></div>
    <li><a href="#">CREDIT</a></li>
  </ul>
</li>

<li class="dropdown">
  <a class="dropdown-toggle" data-toggle="dropdown" href="#">FUNDS TRANSACTION
  <span class="caret"></span></a>
  <ul class="dropdown-menu" style="width: 100%;text-align: center;">
    <li><a href="#">WITHIN SAME BANK</a></li>
    <div class="divider"></div>
    <li><a href="#">OTHER BANK TRANSFER</a></li>
```

```
</ul>
</li>
</ul>
</div>
</div>

<div class="container-fluid">
<div class="row" style="text-align: center;background-color: rgba(173, 235, 196, 0.603);">
<div class="col-md-3" style=" padding-left: 60px;padding-top: 20px;">
<div class="card" style="width: 18rem;padding: 20px;">

<div class="card-body" style="padding: 8px; text-align: center;">
<p class="card-text" style=" font-size: 20px;font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;">Access your account online from anywhere</p>
</div>
</div>
</div>

<div class="col-md-3" style=" padding-left: 60px;padding-top: 20px;">
<div class="card" style="width: 18rem;padding: 20px;">

<div class="card-body" style="padding: 8px; text-align: center;">
<p class="card-text" style=" font-size: 20px;font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;">Transfer funds to anyone from anywhere</p>
</div>
</div>
</div>
```

```
<div class="col-md-3" style=" padding-left: 60px;padding-top: 20px;">  
    <div class="card" style="width: 18rem;padding: 20px;">  
          
        <div class="card-body" style="padding: 8px; text-align: center;">  
            <p class="card-text" style=" font-size: 20px;font-family: 'Franklin Gothic Medium', 'Arial  
Narrow', Arial, sans-serif;">Vast variety of services offered by the bank</p>  
        </div>  
    </div>  
</div>  
  
<div class="col-md-3" style=" padding-left: 60px;padding-top: 20px;">  
    <div class="card" style="width: 18rem;padding: 20px;">  
          
        <div class="card-body" style="padding: 8px; text-align: center;">  
            <p class="card-text" style=" font-size: 20px;font-family: 'Franklin Gothic Medium', 'Arial  
Narrow', Arial, sans-serif;">High end encryption to provide safe online banking</p>  
        </div>  
    </div>  
</div>  
  
</div>
```

```
<div class="modal" id="loginPage" style="padding: 10px;margin-top:50px;  
background:transparent;">  
    <div class="modal-dialog">  
        <div class="modal-content" style="text-align: center;">
```

```
<button type="button" class="close" data-dismiss="modal" style="margin-right: 20px; margin-top: 10px; ">&times;</button>

<!-- Modal Header -->

<div class="modal-header" >

  <h4 class="modal-title" >LOGIN</h4>

</div>

<!-- Modal body -->

<div class="modal-body">

  <form action="login_process" method="post">

    <div class="form-group">

      <i class="fas fa-user-tie"></i> <label for="username">USERNAME:</label>

      <input type="text" class="form-control" id="username" placeholder="Enter username" name="username" style="padding: 20px;text-align: center;" required>

    </div>

    <div class="form-group">

      <i class="fas fa-key"></i> <label for="pwd">PASSWORD:</label>

      <input type="password" class="form-control" id="pwd" placeholder="Enter password" name="pswd" style="padding: 20px;text-align: center;" required>

    </div>

    <input type="submit" class="btn btn-primary" name="login" value="LOGIN"/>

    <input type="reset" class="btn btn-primary" value="RESET"/>

  </form>

</div>

</div>

</div>
```

```
<div class="modal" id="signupPage" style="padding: 10px; margin-top: 50px; background: transparent;">

<div class="modal-dialog">

<div class="modal-content" style="text-align: center;">

    <button type="button" class="close" data-dismiss="modal" style="margin-right: 20px; margin-top: 10px; ">&times;</button>

    <div class="modal-header" >

        <h4 class="modal-title" >NET-BANKING REGISTRATION</h4>

    </div>

    <div class="modal-body">

        <form action="Register" method="post">

            <div class="form-group">

                <label for="name">FULL NAME:</label>

                <input type="text" class="form-control" id="name" placeholder="Enter your name" name="fullname" style="padding: 20px; text-align: center;" required>

            </div>

            <div class="form-group">

                <label for="acc_no">ACCOUNT NO.:</label>

                <input type="text" class="form-control" id="acc_no" placeholder="Enter Account Number" name="account_num" maxlength="8" minlength="8" style="padding: 20px; text-align: center;" required>

            </div>

            <div class="form-group">

                <label for="email">EMAIL ID:</label>

                <input type="email" class="form-control" id="email" placeholder="Enter email" name="email" style="padding: 20px; text-align: center;" required>

            </div>

            <div class="form-group">

                <label for="username">USERNAME:</label>

                <input type="text" class="form-control" id="username" placeholder="Enter username" name="user" maxlength="15" minlength="8" style="padding: 20px; text-align: center;" required>

            </div>

        </form>

    </div>

</div>
```

```
<div class="form-group">
    <label for="pwd">PASSWORD:</label>
    <input type="password" class="form-control" id="pwd" placeholder="Enter password" name="pswd" maxlength="15" minlength="8" style="padding: 20px;text-align: center;" data-trigger="focus" data-toggle="popover" minlength="8" maxlength="15" data-content="Password must be 8 to 15 characters long and must include a capital alphabet,small alphabet , numbers from 0-9 and special characters such as #,@,$,&" required>
</div>

<input type="submit" class="btn btn-primary" name="login" value="SIGN UP"/>
<input type="reset" class="btn btn-primary" value="RESET"/>
</form>
</div>
</div>
</div>
</div>

<div class="container-fluid">
    <div class="row" style="text-align: center; background-color: rgb(201, 198, 197); font-size: 30px; height: 250px; padding-top: 5px;">
        Announcements
        <hr>
        <div class="col-lg-12" style="font-style: italic; font-size: 18px;">
            No recent announcements.....
        </div>
    </div>
    <div class="row" style="text-align: center; background-color: coral; font-size: 30px; height: 50px; padding-top: 5px;">
        <div class="col-lg-12">
            <div class="shadow-lg p-3 mb-5 bg-white rounded">
                Follow us on.....
            </div>
        </div>
    </div>
</div>
```

```
</div>

</div>

</div>

<div class="row" style="height: 300px; background-color: rgba(18, 231, 82, 0.445);">

<div class="col-lg-3" style="text-align: center; align-items: center; padding-top: 100px;">
    <i class="fab fa-facebook fa-7x" style="color: rgb(9, 89, 168);"></i>
</div>

<div class="col-lg-3" style="text-align: center; align-items: center; padding-top: 100px;">
    <i class="fab fa-instagram fa-7x" style="color: rgba(241, 103, 11, 0.705);"></i>
</div>

<div class="col-lg-3" style="text-align: center; align-items: center; padding-top: 100px;">
    <i class="fab fa-twitter fa-7x" style="color: rgb(37, 177, 233); padding: 3px;"></i>
</div>

<div class="col-lg-3" style="text-align: center; align-items: center; padding-top: 100px;">
    <i class="fab fa-linkedin fa-7x" style="color: rgb(27, 192, 192);"></i>
</div>

</div>

<div class="row" style="text-align: center; background-color: rgb(80, 150, 255); font-size: 30px; height: 250px; padding-top: 5px;">

<div class="col-lg-6" style="padding-top: 10px;">
    Contact-Us: <hr style="border: 2px dashed grey;">
    <p style="font-size: 17px; padding-top: 10px; font-family: Arial, Helvetica, sans-serif;">
        Kaustav Bhattacharya<br/>
        +91-9582009715<br/>
        kaustavb79@gmail.com
    </p>
</div>

<div class="col-lg-6">
```

```
<ul id="footer_links">
<li><a href="About.html">About Us</a></li>
<li><a href="MobileBank.html">Mobile Banking</a></li>
<li><a href="terms_and_condt.html">Terms and Conditions</a></li>
<li><a href="admin/developer_login.html">Developer Login</a></li>
</ul>

</div>
</div>
</div>

<footer style="padding-top: 10px; padding-bottom: 10px; background-color: dimgray; text-align: center;">
<i class="far fa-copyright fa-2x" style="color: cornsilk;">opyright: Kaustav Bhattacharya</i>
</footer>

<script>
$(document).ready(function(){
    $('[data-toggle="popover"]').popover();
});
</script>
</body>
</html>
```

Terms_and_conditions.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<title>ONLINE BANKING</title>

<style>

#wrap {

width: 100%;

height: 40px;

margin: 0;

z-index: 99;

position: relative;

background-color: skyblue;

}

.navbar {

height: 40px;

padding: 0;

margin: 0;

position: absolute;

}

.navbar li {

height: auto;

}

.navbar > li > p{

margin-top: 7px;

margin-left: 25px;

font-size: 25px;

color: rgb(95, 93, 93);

}
```

```
}

.footer {
    width: 100%;
    background-color: slategrey;
    color: white;
    text-align: center;
    font-size: 20px;
    font-family: 'Times New Roman', Times, serif;
    padding-top: 3px;
    padding-bottom: 3px;
}

.navbar-header > img {
    height: 70px; float: left;
    position: relative;
    top: -10px;
}

#head {
    position: relative;
    top: 15px;
    font-size: 30px;
    font-family: 'Times New Roman', Times, serif;
}

#body li {
    font-size: 20px;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
}

</style>

</head>
```

<body>

<div class="navbar-header">

<h3 id="head">ONLINE BANKING</h3>

</div>

<div id="wrap">

<ul class="navbar">

<p> Terms and Conditions </p>

</div>

<p>

<ul id="body">

I. Scope of Application

Terms and Conditions of Online Banking Services (as may be varied and amended from time to time, hereinafter referred to as these “Terms”) shall apply to the Online Banking Services provided by Standard Chartered Bank (China) Limited (hereinafter referred to as “We” or “Us”) and shall be legally binding on customers and us.

Both customers and we shall comply with the Terms.

Before a customer applies to us for activation of Online Banking Services or uses Online Banking Services for the first time, the customer shall carefully read the Terms and fully understand relevant provisions hereof, and the customer shall have the right to require us to provide sufficient explanation about the Terms.

BY APPLYING TO US FOR THE ACTIVATION OF ONLINE BANKING SERVICES OR USING ONLINE BANKING SERVICES FOR THE FIRST TIME, CUSTOMERS SHALL BE DEEMED TO HAVE CAREFULLY READ, UNDERSTOOD AND ACCEPTED THE TERMS AND AGREED TO BE BOUND HEREBY.

II. Provision of Services

Customers may apply to us for the activation of Online Banking Services, and we may also activate all or part of the functions of Online Banking Services for qualified customers. Customers may use part of Online Banking Services without applications or requiring our additional operations. Customers may apply to us for the termination of certain Online Banking Services, and we have the right to accept or reject such application of the customers.

Customers may use computers, fixed-line phones, mobile phones, ATMs and other self-service bank facilities, or other electronic devices to use or operate all or part of financial services via the Internet, telephone communication networks, wireless networks, other open public networks or private networks provided by us (hereinafter referred to as “Online Banking Services” or these “Services”).

III. Content of Services

Online Banking Services

(a) Customers may enjoy Online Banking Services via our website at sc.com/cn.

(b) Online Banking Services include the Mobile Banking Services.

Online Account Services for Credit Cards and Credit Card Mall Services

Customers may log on our websites at <http://ccibanking.standardchartered.com.cn> and <http://ccreward.standardchartered.com.cn/> to enjoy online account services for credit cards and credit card mall services, respectively.

WeChat Banking Services

Customers can enjoy WeChat Banking Services for credit cards via WeChat public platform developed by Shenzhen Tencent Computer System Company Limited (“Tencent”), and “Standard Chartered China” is our only WeChat public account. For the avoidance of doubt, the Terms shall not apply to the Internet Technology Service relationships between customers and Tencent, as well as between us and Tencent.

Phone Banking Services

(a) Customers may enjoy Phone Banking Services via the customer service hotline or contact number for Phone Banking published by us on our official website www.sc.com/cn.

(b) The customer service hotline for credit cards is 400-820-6663; for customers located in Hong Kong, Macao, Taiwan and overseas regions, the credit card service hotline is (86-755) 82686080.

(c) We/our branches and sub-branches may also dial the telephone numbers registered by customers with us to provide financial services to them.

Video Banking Services

Video banking service is available at some pages of our official website www.sc.com.cn, and customers can click on the “Start Chat” button on the pages and choose from video call, audio call or text dialog to enjoy relevant banking services provided by our remote customer advisors.

SMS Banking Services

(a) SMS Banking Services refer to those services with which we will handle financial business for customers according to SMS instructions sent by customers and automatically notify customers of relevant results via SMS.

(b) Customers may enjoy SMS Banking Services via our registered SMS platform number, and we will publish the registered SMS platform number then in effect on our official website www.sc.com/cn.

We may provide other Online Banking Services to customers from time to time.

We may change the website address, telephone number, SMS number and domain name for Online Banking Services from time to time and make announcement or provide notice within a reasonable period of time. Thereafter, the changed website address, telephone number, SMS number, and domain name for Online Banking Services as announced or notified to customers by us from time to time shall apply.

CUSTOMERS SHALL NOT USE WEBSITE ADDRESS, TELEPHONE NUMBER, SMS NUMBER, OR DOMAIN NAME FOR ONLINE BANKING SERVICES NOT ANNOUNCED OR NOTIFIED BY US.

```
</ul>

</p>

</p>

<div class="footer">

    Footer

</div>

</body>

</html>
```

MobileBanking.html

```
<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

    <title>ONLINE BANKING</title>

    <style>

        #wrap {

            width: 100%;

            height: 40px;

            margin: 0;

            z-index: 99;

        }

    </style>
```

```
position: relative;  
background-color: skyblue;  
}  
  
.navbar {  
height: 40px;  
padding: 0;  
margin: 0;  
position: absolute;  
}  
  
.navbar li {  
height: auto;  
}  
  
.navbar > li > p{  
margin-top: 7px;  
margin-left: 25px;  
font-size: 25px;  
color: rgb(95, 93, 93);  
}  
  
.footer {  
width: 100%;  
background-color:slategrey;  
color:white;  
text-align: center;  
font-size: 20px;  
font-family: 'Times New Roman', Times, serif;  
padding-top: 3px;  
padding-bottom:3px;  
}
```

```
.navbar-header > img{  
    height:70px; float: left;  
    position: relative;  
    top:-10px;  
}  
  
#head{  
    position: relative;  
    top: 15px;  
    font-size: 30px;  
    font-family: 'Times New Roman', Times, serif;  
}  
  
.content,ul li h4,ul ul li{  
    font-size: 20px;  
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;  
}  
  
</style>  
  
</head>  
  
<body>  
  
<div class="navbar-header">  
    <h3 id="head">ONLINE BANKING</h3> </img>  
</div>  
  
<br>  
  
<div id="wrap">  
    <ul class="navbar">  
        <li><p> Mobile Banking </p> </li>  
    </ul>  
</div>  
  
<p>
```

<p class="content">

Mobile banking is a service provided by a bank or other financial institution that allows its customers to conduct financial transactions remotely using a mobile device such as a smartphone or tablet. Unlike the related internet banking it uses software, usually called an app, provided by the financial institution for the purpose.

Mobile banking is usually available on a 24-hour basis. Some financial institutions have restrictions on which accounts may be accessed through mobile banking, as well as a limit on the amount that can be transacted. Mobile banking is dependent on the availability of an internet or data connection to the mobile device.

Transactions through mobile banking depend on the features of the mobile banking app provided and typically includes obtaining account balances and lists of latest transactions, electronic bill payments, remote check deposits, P2P payments, and funds transfers between a customer's or another's accounts.

Some apps also enable copies of statements to be downloaded and sometimes printed at the customer's premises.

Using a mobile banking app increases ease of use, speed, flexibility and also improves security because it integrates with the user built-in mobile device security mechanisms.

From the bank's point of view, mobile banking reduces the cost of handling transactions by reducing the need for customers to visit a bank branch for non-cash withdrawal and deposit transactions. Mobile banking does not handle transactions involving cash, and a customer needs to visit an ATM or bank branch for cash withdrawals or deposits.

Many apps now have a remote deposit option; using the device's camera to digitally transmit cheques to their financial institution.

Mobile banking differs from mobile payments, which involves the use of a mobile device to pay for goods or services either at the point of sale or remotely, analogously to the use of a debit or credit card to effect an EFTPOS payment.

</p>

<h4>In one academic model, mobile banking is defined as:</h4>

<p class="content">Mobile Banking refers to provision and availment of banking- and financial services with the help of mobile telecommunication devices. The scope of offered services may include facilities to conduct bank and stock market transactions, to administer accounts and to access customised information.</p>

<h4>According to this model mobile banking can be said to consist of two inter-related concepts:</h4>

<li class="list1">Mobile accounting

<li class="list1">Mobile financial information services

<p class="content">Most services in the categories designated accounting and brokerage are transaction-based. The non-transaction-based services of an informational nature are however essential for conducting transactions – for instance, balance inquiries might be needed before committing a money remittance.

The accounting and brokerage services are therefore offered invariably in combination with information services. Information services, on the other hand, may be offered as an independent module.</p>

<h4>Mobile banking may also be used to help in business situations as well as for financial situation.</h4>

Account information

Mini-statements and checking of account history

Alerts on account activity or passing of set thresholds

Monitoring of term deposits

Access to loan statements

Access to card statements

Mutual funds / equity statements

Insurance policy management

Transaction

</p>

<div class="footer">

Footer

</div>

</body>

</html>

AboutUs.html

```
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<title>ONLINE BANKING</title>
<style>
    #wrap {
        width: 100%;
        height: 40px;
        margin: 0;
        z-index: 99;
        position: relative;
        background-color: skyblue;
    }
    .navbar {
        height: 40px;
        padding: 0;
        margin: 0;
        position: absolute;
    }
    .navbar li {
        height: auto;
    }
    .navbar > li > p{
        margin-top: 7px;
    }
</style>
```

```
margin-left: 25px;  
font-size: 25px;  
color: rgb(95, 93, 93);  
}  
  
.footer {  
width: 100%;  
background-color:slategrey;  
color:white;  
text-align: center;  
font-size: 20px;  
font-family: 'Times New Roman', Times, serif;  
padding-top: 3px;  
padding-bottom:3px;  
}  
.navbar-header > img{  
height:70px; float: left;  
position: relative;  
top:-10px;  
}  
#head{  
position: relative;  
top: 15px;  
font-size: 30px;  
font-family: 'Times New Roman', Times, serif;  
}  
.content,ul li h4,ul h3{  
font-size: 20px;
```

```
font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;  
}  
</style>  
</head>  
<body>  
  <div class="navbar-header">  
    <h3 id="head">ONLINE BANKING</h3> </img>  
  </div>  
  <br>  
  <div id="wrap">  
    <ul class="navbar">  
      <li><p> About Us </p> </li>  
    </ul>  
  </div>
```

<p>

<p class="content">Online banking, also known as internet banking or web banking, is an electronic payment system that enables customers of a bank or other financial institution to conduct a range of financial transactions through the financial institution's website. The online banking system will typically connect to or be part of the core banking system operated by a bank and is in contrast to branch banking which was the traditional way customers accessed banking services.

Some banks operate as a "direct bank" (or "virtual bank"), where they rely completely on internet banking.

Internet banking software provides personal and corporate banking services offering features such as viewing account balances, obtaining statements, checking recent transactions, transferring money between accounts, and making payments.</p>

<h3>First online banking services by region:</h3>

<h4>UNITED KINGDOM: </h4>

<p class="content">Online banking started in the United Kingdom with the launch of Nottingham Building Society (NBS)'s Homelink service in September 1982, initially on a restricted basis, before it was expanded nationally in 1983. Homelink was delivered through a partnership with the Bank of Scotland and British Telecom's Prestel service. The system used Prestel viewlink system and a computer, such as the BBC Micro, or keyboard (Tandata Td1400) connected to the telephone

system and television set. The system allowed users to "transfer money between accounts, pay bills and arrange loans... compare prices and order goods from a few major retailers, check local restaurant menus or real estate listings, arrange vacations. enter bids in Homelink's regular auctions and send electronic mail to other Homelink users.

In order to make bank transfers and bill payments, a written instruction giving details of the intended recipient had to be sent to the NBS who set the details up on the Homelink system. Typical recipients were gas, electricity and telephone companies and accounts with other banks. Details of payments to be made were input into the NBS system by the account holder via Prestel. A cheque was then sent by NBS to the payee and an advice giving details of the payment was sent to the account holder. BACS was later used to transfer the payment directly.</p>

<h4>UNITED STATES:</h4>

<p class="content">In the United States in-home banking was "is still in its infancy" with banks "cautiously testing consumer interest" in 1984, a year after online banking went national in the UK. At the time Chemical Bank in New York was "still working out the bugs from its service, which offers somewhat limited features".The service from Chemical, called Pronto, was launched in 1983 and was aimed at individuals and small businesses. It enabled them to maintain electronic checkbook registers, see account balances, and transfer funds between checking and savings accounts. The other three major banks — Citibank, Chase Bank and Manufacturers Hanover — started to offer home banking services soon after. Chemical's Pronto failed to attract enough customers to break even and was abandoned in 1989. Other banks had a similar experience.

Since it first appeared in the United States, online banking has been federally governed by the Electronic Funds Transfer Act of 1978.</p>

<h4>FRANCE:</h4>

<p class="content">After a test period with 2,500 users starting in 1984, online banking services were launched in 1988,using Minitel terminals that were distributed freely to the population by the government.

By 1990, 6.5 million Minitels were installed in households. Online banking was one of the most popular services.

Online banking services later migrated to Internet.</p>

<h4>JAPAN:</h4>

<p class="content">In January 1997, the first online banking service was launched by Sumitomo Bank.By 2010, most major banks implemented online banking services, however, the types of services offered varied.According to a poll conducted by Japanese Bankers Association (JBA) in 2012, 65.2% were the users of personal internet banking.</p>

<h4>CHINA:</h4>

<p class="content"> January 2015, WeBank, the online bank created by Tencent, started 4-month-long online banking trial operation.</p>

<h4>AUSTRALIA:</h4>

<p class="content">In December 1995, Advance Bank acquired by St.George Bank, started to provide customers with online banking with the rollout of the C++ Internet banking program.</p>

<h4>INDIA:</h4>

<p class="content">In 1998, ICICI Bank introduced internet banking to its customers.</p>

</p>

<div class="footer">

Footer

</div>

</body>

</html>

1. CUSTOMER MODULE:

Account statement.jsp

```
<%@page import="bank.model.Customer"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="java.util.List"%>
<%@page import="bank.connect_dao.DBOperation"%>
<%@page import="bank.model.Transactions"%>
<%@page contentType="text/html" pageEncoding="UTF-8" session="false"%>
<!DOCTYPE html>
<html lang="en">
<head>
    <title>CUSTOMER DASHBOARD</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<style>
.header {
    padding: 9px 10px;
    background: rgba(85, 218, 200, 0.911);
    color: #f1f1f1;
    font-size: medium;
    text-align: center;
}

.footer {
    padding-top: 5px;
    padding-bottom: 3px;
    background: rgba(85, 218, 200, 0.911);
    color: black;
    font-size: 18px;
    text-align: center;
    font-family: 'Times New Roman', Times, serif;
    position: relative;
    bottom: 0;
    width: 100%;
}
.header h2{
    font-weight: bolder;
    color: black;
    font-family: 'Times New Roman', Times, serif;
}
.navbar{
    background-color: rgb(158, 230, 240);
    font-size: 18px;
    border-color: transparent;
}
</style>
</head>
<body>
<%
if(request.getSession() != null){
HttpSession sess = request.getSession();
if(sess.getAttribute("account_num") != null){
    long anum = (Long)sess.getAttribute("account_num");
    Customer cus = DBOperation.returnCustomerByAccountNumber(anum);
    List<Transactions> list = DBOperation.fetchTransactions(anum);
    SimpleDateFormat format = new SimpleDateFormat("E, dd MMM yyyy HH:mm:ss
z");
    String gender;
    if(cus.getGender().equals("Male")){
        gender = "Mr.";
    }
    else{
        gender = "Mrs.";
    }
    String path = application.getContextPath() + "/customer/";
%>
<div class="header">

```

```

<h2>ONLINE BANKING SYSTEM</h2><h4 style=" color:black;">Welcome <%
gender.concat(" "+cus.getName().toUpperCase()) %> </h4>
</div>
<div class="container">
<nav class="navbar navbar-inverse">
<div class="container-fluid">
<ul class="nav navbar-nav">
    <li class="list"><a id="active" href="<% path %>account_summary.jsp"
style="color: black; font-family: 'Times New Roman', Times, serif;">Account
Summary</a></li>
    <li class="list"><a href="<% path %>transfer.jsp" style="color: black; font-
family: 'Times New Roman', Times, serif;">Fund Transfer</a></li>
    <li class="list"><a href="<% path %>account_statement.jsp" style="color:
black; font-family: 'Times New Roman', Times, serif;">Account Statement</a></li>
    <li class="list"><a href="<% path %>profile.jsp" style="color: black; font-
family: 'Times New Roman', Times, serif;">Profile</a></li>
</ul>
<ul class="nav navbar-nav navbar-right">
    <li><a href="<% path %>logout.jsp" style="color: black; font-family: 'Times New
Roman', Times, serif;"><span class="glyphicon glyphicon-log-in" style="color: black; font-
size: 20px;"></span>&nbsp; Logout</a></li>
</ul>
</div>
</nav>
<div id="frame" style="width: 100%;height: 100%;padding:25px;">
    <div style="text-align:center"><br>
        <h3><u>ACCOUNT STATEMENT</u></h3><br>
        <h4><b>Name:</b>&nbsp; <%= cus.getName() %></h4><br>
        <h4><b>Account Number:</b>&nbsp; <%= cus.getAccount_number()
%></h4><br>
        <h4><b>Balance:</b>&nbsp; Rs. <%= cus.getBalance() %></h4><br>
    </div><br>
    <table class="table text-center">
        <thead class="thead-dark">
            <tr>
                <th class="text-center">TRANSACTION NO.</th>
                <th class="text-center">COMMENT</th>
                <th class="text-center">DEBIT</th>
                <th class="text-center">CREDIT</th>
                <th class="text-center">TRANSACTION DATE</th>
            </tr>
        </thead>
        <tbody>
            <% for(Transactions trx:list){ %>
            <tr>
                <td><%= trx.getTransaction_no() %></td>
                <td><%= trx.getComment() %></td>
                <td><%= trx.getDebit() %></td>
                <td><%= trx.getCredit() %></td>
                <td><%= format.format(trx.getDateOfTransaction()) %></td>
            </tr>
            <% } %>
        </tbody>
    </table>
</div>

```

```

<br>
<footer>
    Online Banking. All Rights Reserved.
</footer>
</div>
<%
    }else{
        %>
        <h4>Session Expired</h4>
    <%
        response.setHeader("Refresh", "3;url=../index.html");
    }
    %
%>
</body>
</html>

```

Account summary.jsp

```

<%@page import="bank.model.Customer"%>
<%@page import="bank.connect_dao.DBOperation"%>
<%@page contentType="text/html" pageEncoding="UTF-8" session="false"%>
<!DOCTYPE html>
<html lang="en">
<head>
    <title>CUSTOMER DASHBOARD</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <style>
.header {
    padding: 9px 10px;
    background: #f1f1f1;
    color: #f1f1f1;
    font-size: medium;
    text-align: center;
}

.footer {
    padding-top: 5px;
    padding-bottom: 3px;
    background: #f1f1f1;
    color: black;
    font-size: 18px;
    text-align: center;
    font-family: 'Times New Roman', Times, serif;
    position: fixed;
    bottom: 0;
    width: 88%;
}

```

```

.header h2{
    font-weight: bolder;
    color: black;
    font-family: 'Times New Roman', Times, serif;
}
.navbar{
    background-color: rgb(158, 230, 240);
    font-size: 18px;
    border-color: transparent;
}
</style>
</head>
<body>
<%
if(request.getSession()!=null){
HttpSession sess=request.getSession();
if(sess.getAttribute("account_num")!=null){
    long anum=(Long)sess.getAttribute("account_num");
    Customer cus=DBOperation.returnCustomerByAccountNumber(anum);
    String gender;
    if(cus.getGender().equals("Male")){
        gender="Mr.";
    }
    else{
        gender="Mrs.";
    }
    String path=application.getContextPath() + "/customer/";
}
%>
<div class="header">
    <h2>ONLINE BANKING SYSTEM</h2><h4 style=" color:black;">Welcome <%=gender.concat(" "+cus.getName().toUpperCase()) %> </h4>
</div>
<div class="container">
    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <ul class="nav navbar-nav">
                <li class="list"><a id="active" href="<%= path %>/account_summary.jsp" style="color: black; font-family: 'Times New Roman', Times, serif;">Account Summary</a></li>
                <li class="list"><a href="<%= path %>/transfer.jsp" style="color: black; font-family: 'Times New Roman', Times, serif;">Fund Transfer</a></li>
                <li class="list"><a href="<%= path %>/account_statement.jsp" style="color: black; font-family: 'Times New Roman', Times, serif;">Account Statement</a></li>
                <li class="list"><a href="<%= path %>/profile.jsp" style="color: black; font-family: 'Times New Roman', Times, serif;">Profile</a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
                <li><a href="<%= path %>/logout.jsp" style="color: black; font-family: 'Times New Roman', Times, serif;"><span class="glyphicon glyphicon-log-in" style="color: black; font-size: 20px;"></span>&ampnbsp Logout</a></li>
            </ul>
        </div>
    </nav>
    <div id="frame" style="width: 100%;height: 100%;padding:25px;">

```

```

<table class="table text-center">
<thead class="thead-dark">
<tr>
<th class="text-center">ACCOUNT NUMBER</th>
<th class="text-center">TYPE OF ACCOUNT</th>
<th class="text-center">EMAIL ID</th>
<th class="text-center">MOBILE NO.</th>
<th class="text-center">BALANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td><%= cus.getAccount_number() %></td>
<td><%= cus.getType_of_account() %></td>
<td><%= cus.getEmail() %></td>
<td><%= cus.getMobile() %></td>
<td>Rs. <%= cus.getBalance() %></td>
</tr>
</tbody>
</table>
</div>
<br>
<footer>
    Online Banking. All Rights Reserved.
</footer>
</div>
<%
    }else{
        %
        <h4>Session Expired</h4>
    %
    response.setHeader("Refresh", "3;url=../index.html");
}
%
</body>
</html>

```

Customer dashboard.jsp

```

<%@page import="bank.model.Customer"%>
<%@page import="bank.connect_dao.DBOperation"%>
<%@page contentType="text/html" pageEncoding="UTF-8" session="false"%>
<!DOCTYPE html>
<html lang="en">
<head>
    <title>CUSTOMER DASHBOARD</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

```

```

<style>
.header {
    padding:9px 10px;
    background:rgba(85, 218, 200, 0.911);
    color: #f1f1f1;
    font-size: medium;
    text-align: center;
}

footer {
    padding-top: 5px;
    padding-bottom: 3px;
    background:rgba(85, 218, 200, 0.911);
    color: black;
    font-size:18px;
    text-align: center;
    font-family: 'Times New Roman', Times, serif;
    position: fixed;
    bottom: 0;
    width: 88%;
}
.header h2{
    font-weight: bolder;
    color: black;
    font-family: 'Times New Roman', Times, serif;
}
.navbar{
    background-color: rgb(158, 230, 240);
    font-size: 18px;
    border-color: transparent;
}
</style>
</head>
<body>
<%
if(request.getSession() != null){
HttpSession sess = request.getSession();
if(sess.getAttribute("account_num") != null){
    long anum = (Long)sess.getAttribute("account_num");
    Customer cus = DBOperation.returnCustomerByAccountNumber(anum);
    String gender;
    if(cus.getGender().equals("Male")){
        gender = "Mr.";
    }
    else{
        gender = "Mrs.";
    }
    String path = application.getContextPath() + "/customer/";
%>
<div class="header">
    <h2>ONLINE BANKING SYSTEM</h2><h4 style=" color:black;">Welcome <%
    gender.concat(" "+cus.getName().toUpperCase()) %> </h4>
</div>
<div class="container">
    <nav class="navbar navbar-inverse">

```

```

<div class="container-fluid">
  <ul class="nav navbar-nav">
    <li class="list"><a id="active" href="<%=" path %>account_summary.jsp"
      style="color: black; font-family: 'Times New Roman', Times, serif;">Account
      Summary</a></li>
    <li class="list"><a href="<%=" path %>transfer.jsp" style="color: black; font-
      family: 'Times New Roman', Times, serif;">Fund Transfer</a></li>
    <li class="list"><a href="<%=" path %>account_statement.jsp" style="color:
      black; font-family: 'Times New Roman', Times, serif;">Account Statement</a></li>
    <li class="list"><a href="<%=" path %>profile.jsp" style="color: black; font-
      family: 'Times New Roman', Times, serif;">Profile</a></li>
  </ul>
  <ul class="nav navbar-nav navbar-right">
    <li><a href="<%=" path %>logout.jsp" style="color: black; font-family: 'Times New
      Roman', Times, serif;"><span class="glyphicon glyphicon-log-in" style="color: black; font-
      size: 20px;"></span>&ampnbsp Logout</a></li>
  </ul>
</div>
</nav>
<div id="frame" style="width: 100%;height: 100%;padding:25px;">

<table class="table text-center">
<thead class="thead-dark">
  <tr>
    <th class="text-center">ACCOUNT NUMBER</th>
    <th class="text-center">TYPE OF ACCOUNT</th>
    <th class="text-center">EMAIL ID</th>
    <th class="text-center">MOBILE NO.</th>
    <th class="text-center">BALANCE</th>
  </tr>
</thead>
<tbody>
  <tr>
    <td><%= cus.getAccount_number() %></td>
    <td><%= cus.getType_of_account() %></td>
    <td><%= cus.getEmail() %></td>
    <td><%= cus.getMobile() %></td>
    <td>Rs. <%= cus.getBalance() %></td>
  </tr>
</tbody>
</table>
</div>
<br>
<footer>
  Online Banking. All Rights Reserved.
</footer>
</div>
<%
  }else{
    %>
    <h4>Session Expired</h4>
<%
  response.setHeader("Refresh", "3;url=../index.html");
  %
}
}

```

```
%>
</body>
</html>
```

Logout.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>CUSTOMER DASHBOARD</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
            <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
            awesome/4.7.0/css/font-awesome.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
        <script src="https://kit.fontawesome.com/1a88ea4f70.js"
crossorigin="anonymous"></script>
    </head>
    <body style="background-color: #f0f0f0;">
        <div class="container-fluid text-center" style="padding-top: 20%; color: #333399; font-size: 18px; font-weight: bold;">
            Redirecting.....</p>
        </div>
    </body>

<%
    session.invalidate();
    System.out.println("Session Closed");
    response.setHeader("Refresh", "5;url=../index.html");
%>
</html>
```

Profile.jsp

```
<%@page import="bank.model.Customer"%>
<%@page import="bank.connect_dao.DBOperation"%>
<%@page contentType="text/html" pageEncoding="UTF-8" session="false"%>
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>CUSTOMER DASHBOARD</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
            <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
<script  
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>  
<style>  
.header {  
padding: 9px 10px;  
background:rgba(85, 218, 200, 0.911);  
color: #f1f1f1;  
font-size: medium;  
text-align: center;  
}  
  
footer {  
padding-top: 5px;  
padding-bottom: 3px;  
background:rgba(85, 218, 200, 0.911);  
color: black;  
font-size: 18px;  
text-align: center;  
font-family: 'Times New Roman', Times, serif;  
position: relative;  
bottom: 0;  
width: 100%;  
}  
.header h2{  
font-weight: bolder;  
color: black;  
font-family: 'Times New Roman', Times, serif;  
}  
.navbar{  
background-color: rgb(158, 230, 240);  
font-size: 18px;  
border-color: transparent;  
}  
body  
{  
text-align: center;  
font-size: 18px;  
font-family: 'Times New Roman', Times, serif;  
}  
#btn{  
display: inline-block;  
text-align: left;  
margin-left: 150px;  
}  
#addr{  
display: inline;  
}  
#addr > p{  
float: right;  
margin-top: auto;  
font-weight: normal;  
}  
label{  
font-weight: bold;  
}
```



```

        }
    %>
</body>
</html>
```

Fund transfer.jsp

```

<%@page import="bank.model.Customer"%>
<%@page import="bank.connect_dao.DBOperation"%>
<%@page contentType="text/html" pageEncoding="UTF-8" session="false"%>
<!DOCTYPE html>
<html lang="en">
<head>
    <title>CUSTOMER DASHBOARD</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <style>
.header {
    padding: 9px 10px;
    background:rgba(85, 218, 200, 0.911);
    color: #f1f1f1;
    font-size: medium;
    text-align: center;
}
footer {
    padding-top: 5px;
    padding-bottom: 3px;
    background:rgba(85, 218, 200, 0.911);
    color: black;
    font-size: 18px;
    text-align: center;
    font-family: 'Times New Roman', Times, serif;
    position: relative;
    bottom: 0;
    width: 100%;
}
.header h2{
    font-weight: bolder;
    color: black;
    font-family: 'Times New Roman', Times, serif;
}
.navbar{
    background-color: rgb(158, 230, 240);
    font-size: 18px;
    border-color: transparent;
}
h3{
    text-align: center;
```

```

        padding:35px;
        font-weight: bolder;
    }
    input[type=submit]{
        margin-left: 40%;
        margin-top: 20px;
    }
    input[type=reset]{
        margin-top: 20px;
    }
    input[type=text]{
        text-align: center;
        font-size: 18px;
    }

```

</style>

</head>

<body>

<%

```

if(request.getSession()!=null){
    HttpSession sess=request.getSession();
    if(sess.getAttribute("account_num")!=null){
        long anum=(Long)sess.getAttribute("account_num");
        Customer cus=DBOperation.returnCustomerByAccountNumber(anum);
        String gender;
        if(cus.getGender().equals("Male")){
            gender="Mr.";
        }
        else{
            gender="Mrs.";
        }
        String path=application.getContextPath()+"/customer/";
        sess.setAttribute("customer", cus);
    }
}

```

%>

<div class="header">

<h2>ONLINE BANKING SYSTEM</h2><h4 style=" color:black;">Welcome <%=

```

gender.concat(" "+cus.getName().toUpperCase()) %> </h4>

```

</div>

<div class="container">

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<ul class="nav navbar-nav">

<li class="list"><a id="active" href="<%= path %>account_summary.jsp"

>Account

Summary

<li class="list"><a href="<%= path %>transfer.jsp" style="color: black; font-

family: 'Times New Roman', Times, serif;">Fund Transfer

<li class="list"><a href="<%= path %>account_statement.jsp" style="color:

black; font-family: 'Times New Roman', Times, serif;">Account Statement

<li class="list"><a href="<%= path %>profile.jsp" style="color: black; font-

family: 'Times New Roman', Times, serif;">Profile

<ul class="nav navbar-nav navbar-right">

<a href="<%= path %>logout.jsp" style="color: black; font-family: 'Times New

Roman', Times, serif;"><span class="glyphicon glyphicon-log-in" style="color: black; font-

size: 20px;">&nbsp Logout

```

        </ul>
    </div>
</nav>
<div id="frame" style="width: 100%;height: 100%;padding:25px;">
    <% if(cus.getNetbankingAllowed().equalsIgnoreCase("yes")){ %>
        <h3>FUND TRANSFER PAGE</h3>
    <div class="container">
        <form id="form" class="form-horizontal" action="Transfer" method="post">
            <div class="form-group">
                <label class="control-label col-sm-5" for="name">Beneficiary Name:</label>
                <div class="col-sm-3">
                    <input type="text" class="form-control" id="name" name="name" required/><br>
                    </div>
                </div>
                <div class="form-group">
                    <label class="control-label col-sm-5" for="anum">Account Number:</label>
                    <div class="col-sm-3">
                        <input type="text" class="form-control" id="anum" name="account_num" minlength="8" maxlength="8" onkeyup="accountNum()" required/><br><span id="s1"></span>
                    </div>
                </div>
                <div class="form-group">
                    <label class="control-label col-sm-5" for="canum">Confirm Account Number:</label>
                    <div class="col-sm-3">
                        <input type="text" class="form-control" id="canum" minlength="8" maxlength="8" onmouseout="checkAccount()" required/><br><span id="s2"></span>
                    </div>
                </div>
                <div class="form-group">
                    <label class="control-label col-sm-5" for="amt">Amount:</label>
                    <div class="col-sm-3">
                        <input type="text" class="form-control" id="amt" name="amount" maxlength="7" onmouseout="balance()" required/><br><span id="s3"></span>
                    </div>
                </div>
                <div id="radio" style="text-align: center;">
                    <label class="control-label" style="margin-right: 50px;">Branch:</label>
                    <label class="radio-inline"><input type="radio" id="sbranch" name="branch" value="<%=" cus.getIFSCcode().getIFSC() %>" checked>Same Branch</label>
                    <label class="radio-inline"><input type="radio" id="obranch" name="branch" value="">Other Branch</label>
                </div><br>
                <input type="submit" id="submit" value="SUBMIT" class="btn btn-default"/>
                <input type="reset" value="RESET" class="btn btn-default"/>
            </form>
        </div>
        <%
    }else{
    %>
        <h4>Net- Banking Disabled!!!<br> Contact your respective branch...</h4>
        <%
    
```

```

        }
        %>
    </div>
<br>
<footer>
    Online Banking. All Rights Reserved.
</footer>
</div>
<script>
    function accountNum(){
        var anum=document.getElementById("anum").value;
        var s1=document.getElementById("s1");
        if(isNaN(anum)){
            s1.innerHTML="Invalid Account Number!!";
            s1.color="RED";
        }
        else{
            s1.innerHTML="";
        }
    };

    function checkAccount(){
        var anum=document.getElementById("anum").value;
        var canum=document.getElementById("canum").value;
        var s2=document.getElementById("s2");
        if(anum==canum){
            s2.innerHTML="";
        }
        else{
            s2.innerHTML="Account Number Doesn't Match!!";
        }
    };

    function balance(){
        var amt=document.getElementById("amt").value;
        var s3=document.getElementById("s3");
        var amount=parseInt(amt);
        console.log(amount);
        if(isNaN(amt)){
            s3.innerHTML="Invalid Amount!!";
            s3.innerHTML.color="RED";
        }else if(amount<0 || amount>=10000001){
            s3.innerHTML="Amount Should Be greater than Rs.0 and Less than equal to Rs.
10,00,000(10 Lacks)!!";
            s3.innerHTML.color="RED";
        }
        else{
            s3.innerHTML="";
        }
    };

    document.getElementById("obranch").addEventListener("click",function(){
        document.getElementById("radio").insertAdjacentHTML("afterend","<br><div
id='demo' class='form-group'>"+
        "<label class='control-label col-sm-5' for='ifsc'>IFSC Code:</label>"+

```

```

    "<div class='col-sm-3'>"+
    "<input type='text' class='form-control' id='ifsc' name='ifsc' required/>"+
    "</div></div>");
});

document.getElementById("sbranch").addEventListener("click",function(){
    document.getElementById("demo").remove();
});

document.getElementById("submit").addEventListener("click",function(){
    document.getElementById("submit").disabled=true;
    document.getElementById("reset").disabled=true;
});
</script>
<%
}else{
%>
<h4>Session Expired</h4>
<%
    response.setHeader("Refresh", "3;url=../index.html");
}
%>
</body>
</html>

```

CustomerLogin.java

```

package bank.customer;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import bank.connect_dao.DBOperation;
import bank.model.Customer;
import javax.servlet.http.HttpSession;

@WebServlet(name = "CustomerLogin", urlPatterns = { "/login_process" })
public class CustomerLogin extends HttpServlet {

    private void process(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        boolean check=false;
        String username=request.getParameter("username");
        String password=request.getParameter("pswd");
        Customer c=DBOperation.loginCustomer(username,password);
        HttpSession session=request.getSession(true);
        //Session Timeout after 20 Mins.
        session.setMaxInactiveInterval(1200);
        if(c!=null){
            session.setAttribute("account_num", c.getAccount_number());
            response.sendRedirect("customer/customer_dashboard.jsp");
        }
    }
}

```

```

        }
    else{
        response.getWriter().print("Invalid Username Or Password!!!");
        response.setHeader("Refresh","2;url=index.html");
    }
}

@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    process(req, resp);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    process(request, response);
}
}

```

Register.java

```

package bank.customer;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import bank.connect_dao.DBOperation;
import bank.connect_dao.SendEmail;
import bank.model.Customer;

@WebServlet(name = "Register", urlPatterns = {"/Register"})
public class Register extends HttpServlet {

    String host,port,receiver_address,receiver_pass;
    @Override
    public void init() throws ServletException {
        ServletContext context=getServletContext();
        host=context.getInitParameter("host");
        port=context.getInitParameter("port");
        receiver_address=context.getInitParameter("receiver_email");
        receiver_pass=context.getInitParameter("password");
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out=response.getWriter();

```

```

String username=request.getParameter("user");
String password=request.getParameter("pswd");
String email=request.getParameter("email");
String acc_num=request.getParameter("account_num");
String name=request.getParameter("fullname");
Customer
cus=DBOperation.returnCustomerByAccountNumber(Long.parseLong(acc_num));
String update="update Customer set username='"+username+"',
password='"+password+"' where account_number='"+cus.getAccount_number();
String subject="ONLINE BANKING REGISTRATION";
String message;
String gen;
if(cus.getGender().equals("Male")){
    gen="Mr.";
}
else{
    gen="Mrs.";
}

message=<body style="text-align: center;">  

"      <div style="width:100%;">  

"          <h3 style="font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande',
'Lucida Sans', Arial, sans-serif;background-color: rgb(114, 194, 124);font-size: 30px;padding:
10px;color: white;">  

"              Online Banking Registration Details</h3>  

"              <div style="background-color: #C4D3DF;padding: 15px;">  

"                  <p style="font-size: 23px;font-family: 'Times New Roman', Times,
serif;">  

"                      Welcome "+gen+"<b>"+name.toUpperCase()+"</b></p>  

"                  <p style="font-family: 'Times New Roman', Times, serif;font-size: 20px;">  

"                      Thank you for registering for Online Banking .</p><br>  

"                  <p style="font-family: 'Times New Roman', Times, serif;font-size: 20px;">  

"                      Your Login Details are</p><br>  

"                  <p style="font-family: 'Times New Roman', Times, serif;font-size:
20px;">  

"                      <u><i>Username: </i></u></p>&nbsp;&nbsp;" +username+"<br>  

"                  <p style="font-family: 'Times New Roman', Times, serif;font-size:
20px;">  

"                      <u><i>Password: </i></u></p>&nbsp;&nbsp;" +password+"<br>  

"                      <br><br>  

"                  <br> +  

"                  <span style="font-family: 'Times New Roman', Times, serif;font-size: 20px;">  

"                      ><a href="http://onlinebank-env-1.eba-qej9spxf.us-east-2.elasticbeanstalk.com/" style="text-
decoration: none;font-size: 22px;font-style: italic;">  

"                          Click Here</a><br> to login into Online Banking..</span>  

"                  </div>  

"              </div>  

"          </body>";
if(cus!=null){
    if(!cus.getEmail().equalsIgnoreCase(email)){
        out.println("\t\t --> Invalid Email Id!!!\n");
    }
    else if(!cus.getName().equalsIgnoreCase(name)){
        out.println("\t\t --> Invalid Name!!!\n");
    }
    else{
        DBOperation.registerCustomer(update);
        try{

```

```
SendEmail.sendMail(host, port, sender_address, sender_pass, cus.getEmail(),
subject, message);
    }catch(Exception e){
        e.printStackTrace();
    }
    out.println("\n\n\n\n\t\t\tSuccessfully Registered for Online Banking!!!\n\n\t\t\tRedirecting....");
    response.setHeader("Refresh", "3;url=index.html");
}
response.setHeader("Refresh", "3;url=index.html");
}
else{
    out.println("Invalid Account Number!!!!");
    response.setHeader("Refresh", "2;url=index.html");
}
}
```

Transfer.java

```
package bank.customer;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import bank.connect_dao.DBOperation;
import bank.model.Customer;
import bank.model.Transactions;

@WebServlet(name = "Transfer", urlPatterns = {"/customer/Transfer"})
public class Transfer extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        HttpSession s=request.getSession();
        Customer cust_debit=(Customer)s.getAttribute("customer");
        Date dateOfTransaction=new Date();
        String name=request.getParameter("name");
        long account_number=Long.parseLong(request.getParameter("account_num"));
        double amount=Double.parseDouble(request.getParameter("amount"));
        String branch=request.getParameter("branch");
        String ifsc=null;
        if(branch==null){
            ifsc=request.getParameter("ifsc");
        }
    }
}
```

```

String comment_credit="Fund Transfer by "+cust_debit.getName().toUpperCase()+" .";
String comment_debit="Transfer to "+account_number+"\n Benificiary Name: "+name;

Customer
cust_credit=DBOperation.returnCustomerByAccountNumber(account_number);
if(cust_credit==null){
    out.println("Invalid Account Number!!!\nRedirecting");
    response.setHeader("Refresh","2,url=transfer.jsp");
}
if(ifsc!=null){
    if(!(cust_credit.getIFSCcode().getIFSC().equals(ifsc))){
        out.println("Invalid IFSC Code!!!\nRedirecting");
        response.setHeader("Refresh","2,url=transfer.jsp");
    }
}
double cust_debit_balance=cust_debit.getBalance()-amount;
double cust_credit_balance=cust_credit.getBalance()+amount;

Transactions debit=new Transactions();
debit.setComment(comment_debit);
debit.setCredit(" ");
debit.setDebit("Rs. "+String.valueOf(amount));
debit.setAccount_number(cust_debit.getAccount_number());
debit.setDateOfTransaction(dateOfTransaction);

Transactions credit=new Transactions();
credit.setComment(comment_credit);
credit.setCredit("Rs. "+String.valueOf(amount));
credit.setDebit("");
credit.setAccount_number(account_number);
credit.setDateOfTransaction(dateOfTransaction);

if(cust_debit.getBalance()>amount && cust_debit_balance>0){
    if(DBOperation.updateBalance(cust_credit_balance,
account_number)&&DBOperation.updateBalance(cust_debit_balance,
cust_debit.getAccount_number())){
}

if(DBOperation.commitTransaction(debit)&&DBOperation.commitTransaction(credit)){
    out.println("Transaction Successfull.....Redirecting");
    response.setHeader("Refresh","2,url=account_statement.jsp");
}
else{
    out.println("Transaction Failed.....Redirecting");
    response.setHeader("Refresh","2,url=transfer.jsp");
}
}
else{
    out.println("Amount specified is more than your balance / your balance after debit is
less than 0...Redirecting");
    response.setHeader("Refresh","2,url=transfer.jsp");
}
}

```

}

2. EMPLOYEE MODULE:

Employee login.html

```
<html>

<head>

<title>EMPLOYEE LOGIN PAGE</title>

<link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">

<script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<script src="https://kit.fontawesome.com/1a88ea4f70.js"
crossorigin="anonymous"></script>

<style>

html {

background-color: #56baed;

}

body {

font-family: "Poppins", sans-serif;

height: 100vh;

}

a {

color: #92badd;

display:inline-block;

text-decoration: none;

font-weight: 400;

}
```

```
}
```

```
h2 {  
    text-align: center;  
    font-size: 16px;  
    font-weight: 600;  
    text-transform: uppercase;  
    display:inline-block;  
    margin: 40px 8px 10px 8px;  
    color: #cccccc;  
}
```

```
/* STRUCTURE */
```

```
.wrapper {  
    display: flex;  
    align-items: center;  
    flex-direction: column;  
    justify-content: center;  
    width: 100%;  
    min-height: 100%;  
    padding: 20px;  
}
```

```
#formContent {  
    -webkit-border-radius: 10px 10px 10px 10px;
```

```
border-radius: 10px 10px 10px 10px;  
background: #fff;  
padding: 30px;  
width: 90%;  
max-width: 450px;  
position: relative;  
padding: 0px;  
-webkit-box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);  
box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);  
text-align: center;  
}
```

```
#formFooter {  
background-color: #f6f6f6;  
border-top: 1px solid #dce8f1;  
padding: 25px;  
text-align: center;  
-webkit-border-radius: 0 0 10px 10px;  
border-radius: 0 0 10px 10px;  
}
```

```
/* TABS */
```

```
h2.inactive {  
color: #cccccc;  
}
```

```
h2.active {  
    color: #0d0d0d;  
    border-bottom: 2px solid #5fbae9;  
}  
  
/* FORM TYPOGRAPHY */  
  
input[type=button], input[type=submit], input[type=reset] {  
    background-color: #56baed;  
    border: none;  
    color: white;  
    padding: 15px 80px;  
    text-align: center;  
    text-decoration: none;  
    display: inline-block;  
    text-transform: uppercase;  
    font-size: 13px;  
    -webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);  
    box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);  
    -webkit-border-radius: 5px 5px 5px 5px;  
    border-radius: 5px 5px 5px 5px;  
    margin: 5px 20px 40px 20px;  
    -webkit-transition: all 0.3s ease-in-out;  
    -moz-transition: all 0.3s ease-in-out;  
    -ms-transition: all 0.3s ease-in-out;
```

```
-o-transition: all 0.3s ease-in-out;  
transition: all 0.3s ease-in-out;  
}
```

```
input[type=button]:hover, input[type=submit]:hover, input[type=reset]:hover {  
background-color: #39ace7;  
}
```

```
input[type=button]:active, input[type=submit]:active, input[type=reset]:active {  
-moz-transform: scale(0.95);  
-webkit-transform: scale(0.95);  
-o-transform: scale(0.95);  
-ms-transform: scale(0.95);  
transform: scale(0.95);  
}
```

```
input[type=text],input[type=password] {  
background-color: #f6f6f6;  
border: none;  
color: #0d0d0d;  
padding: 15px 32px;  
text-align: center;  
text-decoration: none;  
display: inline-block;  
font-size: 16px;  
margin: 5px;  
width: 85%;  
border: 2px solid #f6f6f6;
```

```
-webkit-transition: all 0.5s ease-in-out;  
-moz-transition: all 0.5s ease-in-out;  
-ms-transition: all 0.5s ease-in-out;  
-o-transition: all 0.5s ease-in-out;  
transition: all 0.5s ease-in-out;  
  
-webkit-border-radius: 5px 5px 5px 5px;  
border-radius: 5px 5px 5px 5px;  
}
```

```
input[type=text]:focus,input[type=password]:focus {  
background-color: #fff;  
border-bottom: 2px solid #5fbae9;  
}
```

```
input[type=text]:placeholder,input[type=password]:placeholder {  
color: #cccccc;  
}
```

```
/* ANIMATIONS */
```

```
/* Simple CSS3 Fade-in-down Animation */  
.fadeInDown {  
-webkit-animation-name: fadeInDown;  
animation-name: fadeInDown;  
-webkit-animation-duration: 1s;  
animation-duration: 1s;
```

```
-webkit-animation-fill-mode: both;  
animation-fill-mode: both;  
}  
  
  
@-webkit-keyframes fadeInDown {  
0% {  
    opacity: 0;  
    -webkit-transform: translate3d(0, -100%, 0);  
    transform: translate3d(0, -100%, 0);  
}  
  
100% {  
    opacity: 1;  
    -webkit-transform: none;  
    transform: none;  
}  
  
}  
  
  
@keyframes fadeInDown {  
0% {  
    opacity: 0;  
    -webkit-transform: translate3d(0, -100%, 0);  
    transform: translate3d(0, -100%, 0);  
}  
  
100% {  
    opacity: 1;  
    -webkit-transform: none;  
    transform: none;  
}
```

}

```
/* Simple CSS3 Fade-in Animation */

@-webkit-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@-moz-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@keyframes fadeIn { from { opacity:0; } to { opacity:1; } }
```

```
.fadeIn {

    opacity:0;

    -webkit-animation:fadeIn ease-in 1;

    -moz-animation:fadeIn ease-in 1;

    animation:fadeIn ease-in 1;

    -webkit-animation-fill-mode:forwards;

    -moz-animation-fill-mode:forwards;

    animation-fill-mode:forwards;

    -webkit-animation-duration:1s;

    -moz-animation-duration:1s;

    animation-duration:1s;

}
```

```
.fadeIn.first {

    -webkit-animation-delay: 0.4s;

    -moz-animation-delay: 0.4s;

    animation-delay: 0.4s;

}
```

```
.fadeIn.second {  
    -webkit-animation-delay: 0.6s;  
    -moz-animation-delay: 0.6s;  
    animation-delay: 0.6s;  
}
```

```
.fadeIn.third {  
    -webkit-animation-delay: 0.8s;  
    -moz-animation-delay: 0.8s;  
    animation-delay: 0.8s;  
}
```

```
.fadeIn.fourth {  
    -webkit-animation-delay: 1s;  
    -moz-animation-delay: 1s;  
    animation-delay: 1s;  
}
```

```
/* Simple CSS3 Fade-in Animation */
```

```
.underlineHover:after {  
    display: block;  
    left: 0;  
    bottom: -10px;  
    width: 0;  
    height: 2px;  
    background-color: #56baed;  
    content: "";  
    transition: width 0.2s;
```

```
}
```

```
.underlineHover:hover {  
    color: #0d0d0d;  
}  
  
}
```

```
.underlineHover:hover:after{  
    width: 100%;  
}  
  
}
```

```
/* OTHERS */
```

```
:focus {  
    outline: none;  
}  
  
}
```

```
#icon {  
    width:60%;  
}  
  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="wrapper fadeInDown">  
    <div id="formContent">
```

```
<div class="fadeIn first">
    <i class="fas fa-user-circle fa-5x" style="color: cornflowerblue;"></i><br/><br>EMPLOYEE LOGIN
</div>

<form action="emplogin.do" method="post">
    <input type="text" id="login" class="fadeIn second" name="username" placeholder="login">
    <input type="password" id="password" class="fadeIn third" name="password" placeholder="password">
    <input type="submit" class="fadeIn fourth" value="Log In">
</form>

<div id="formFooter">
    <a href="../index.html">Back to Home</a>
</div>

</div>
</div>
</body>
</html>
```

Employee dashboard.jsp

```
<%@page import="bank.model.Employee"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
```

```
<title>BANK EMPLOYEE DASHBOARD</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" href="image/..../logo.png" type="image/x-icon"/>

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<script src="https://kit.fontawesome.com/1a88ea4f70.js"
crossorigin="anonymous"></script>

<style>

.navbar-inverse .navbar-nav > li > a,
.navbar-inverse .navbar-nav > li a:hover,
.navbar-inverse .navbar-nav > li a:focus {
color: black;
background-color: transparent;
font-size: 20px;

}

.navbar-fixed-left {
width: 250px;
position: fixed;
border-radius: 0;
height: 100%;
margin-top: 80px;
text-align: center;
}
```

```
.navbar-fixed-left .navbar-nav > li {  
    float: none; /* Cancel default li float: left */  
    width: 100%;  
}  
  
.navbar-fixed-left + .container {  
    padding-left: 160px;  
}  
  
/* On using dropdown menu (To right shift popuded) */  
.navbar-fixed-left .navbar-nav > li > .dropdown-menu {  
    margin-top: -50px;  
    margin-left: 250px;  
    padding: 20px;  
    font-size: 18px;  
}  
  
.navbar-fixed-left .dropdown > a:hover ,  
.navbar .dropdown > a:active ,  
.navbar .dropdown > a:visited,  
.navbar-fixed-left .navbar-nav:hover {  
    background-color: transparent;  
    list-style: none;  
}  
  
.navbar-fixed-top .dropdown > a:hover,.navbar-fixed-top .dropdown > a:active{  
    background-color: transparent;
```

```
}

.navbar-fixed-left .navbar-nav li a:hover{
    background-color: transparent;
}

</style>

<script type="text/javascript">

    function preventBack() {
        window.history.forward();

    }

    setTimeout("preventBack()", 0);

    window.onunload = function () { null; };

</script>

</head>

<body>

<%
    Employee m=(Employee)request.getAttribute("employee");

    Cookie ck=new Cookie("employee_id", String.valueOf(m.getEmployee_id()));

    Cookie ck1=new Cookie("ifsc_id",String.valueOf(m.getIFSCcode()));

    ck.setMaxAge(60*30);

    ck1.setMaxAge(60*30);

    response.addCookie(ck);

    response.addCookie(ck1);

%>

<nav class="navbar navbar-inverse navbar-fixed-top shadow-lg p-3 mb-5 bg-white rounded" style="background-color: #a3a199; height: 80px;padding-top: 15px;">

    <div class="container-fluid">
```

```

<div class="navbar-header">

    <a class="navbar-brand" href="#" style="color: rgb(10, 7, 7); font-weight: bold; font-size: 20px;">&nbsp;EMPLOYEE DASHBOARD</a>

</div>

<ul class="nav navbar-nav navbar-right " style="font-family: Georgia, 'Times New Roman', Times, serif; font-size: 20px; text-align: center; padding-right: 50px;">

    <li><a href="emp_logout.jsp"><i class="fas fa-power-off"></i>&nbsp;Log Out</a></li>

</ul>

</div>

</nav>

<nav class="navbar navbar-fixed-left" style="background-color: #f2e4aa;">

<ul class="nav navbar-nav" style="width: 100%; list-style: none; ">

    <li><a href="employee_details.jsp" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">Employee Details</a></li>

    <li><a href="add_customer.html" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">Add Customer</a></li>

    <li><a href="remove_customer.jsp" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">Remove Customer</a></li>

    <li><a href="listcustomer.jsp" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">List All Customer's</a></li>

    <li><a href="update_customer.jsp" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">Update Customer Details</a></li>

    <li><a href="transactions.jsp" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">Customer Account Statement</a></li>

    <li><a href="freeze_account.jsp" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">Freeze/Un-freeze Customer Account</a></li>

    <li><a href="deposit.html" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">Deposit Money</a></li>

</ul>

</nav>

<iframe name="main_frame" src="employee_details.jsp" style="margin-left: 260px; margin-top: 100px; width: 80%; height: 500px; border: none; "></iframe>

```

```
</body>
```

```
</html>
```

Add customer.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="utf-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
```

```
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
```

```
<style>
```

```
.form-group .form-control{
```

```
    padding: 20px;
```

```
    font-size: 20px;
```

```
}
```

```
.form-group .form-control::placeholder{
```

```
    text-align: center;
```

```
    font-size: 18px;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body >
```

```
<div class="container" style="text-align: center; align-content: center;">
```

```
    <h2> New Customer </h2>
```

```
    <form action="AddCustomer" method="post" style="width: 50%; margin-left: 24%; ">
```

```
        <div class="form-group">
```

```
            <label for="name">Customer Name:</label>
```

```
<input type="text" class="form-control text-center" id="name" maxlength="100"
placeholder="Enter Customer name" name="name" required>

</div>

<div class="form-group">

    <label for="email">Customer Email:</label>

    <input type="email" class="form-control text-center" id="email" maxlength="70"
placeholder="Enter Customer email" name="email" required>

</div>

<div class="form-group">

    <label for="mobile">Customer Mobile:</label>

    <input type="text" class="form-control text-center" id="mobile" placeholder="Enter
Customer mobile" maxlength="10" minlength="10" name="mobile" required>

</div>

<label>Gender:&nbsp;&nbsp;</label>

<label class="radio-inline">

    <input type="radio" name="gender" value="Male" checked>Male

</label>

<label class="radio-inline">

    <input type="radio" name="gender" value="Female">Female

</label><br><br>

<div class="form-group">

    <label for="address">Customer Address:</label>

    <textarea class="form-control" rows="6" id="address" name="address"
required></textarea>

</div>

<div class="form-group">

    <label for="aadhaar">Customer Aadhaar No:</label>

    <input type="text" class="form-control text-center" id="aadhaar" placeholder="Enter
Customer Aadhaar" maxlength="12" minlength="12" name="aadhaar" required>

</div>

<div class="form-group">
```

```

<label for="aadhaar">Customer Pan Card No:</label>

<input type="text" class="form-control text-center" id="pan" placeholder="Enter
Customer Pan No." maxlength="10" name="pan" required>

</div>

<div class="form-group">

<label for="type_of_account">Type Of Account:</label>

<select name="type" id="type_of_account">

<option value="Savings">SAVINGS</option>

<option value="Current">CURRENT</option>

</select>

</div>

<input type="submit" class="btn bg-primary" name="submit" value="ADD
CUSTOMER"/>

<input type="reset" class="btn bg-primary" value="RESET FORM"/>

</form>

</div>

</body>

</html>

```

Update customer.jsp

```

<% @page import="java.util.List"%>

<% @page import="bank.model.Customer"%>

<% @page import="bank.connect_dao.DBOperation"%>

<% @page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

```

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<style>

select,option{

text-align: center;

font-size: 20px;

padding: 5px;

}

form{

padding-top: 20px;

text-align: center;

font-size: 24px;

font-family: 'Courier New', Courier, monospace;

}

input[type=submit]{

padding:7px;

width:150px;

border-radius: 4px;

}

</style></head>

<body>

<%
Cookie ck=null;

Cookie[] c=request.getCookies();

for(int i=0;i<c.length;i++){

if(c[i].getName().equals("employee_id")){

ck=c[i];


```

```
        break;  
    }  
  
    String value=ck.getValue();  
  
    long emp_id=Integer.parseInt(value);  
  
    List<Customer> list=DBOperation.returnAllCustomerByIFSC(emp_id);  
  
%>  
  
<form action="update_customer_next.jsp" method="POST">  
  
    <div class="form-group">  
  
        <label for="sel1">Select Customer Account Number: </label>  
  
        <select class="form-control" id="sel1" name="account_num">  
  
            <% for(Customer cus:list){ %>  
  
                <option value="<%= cus.getAccount_number() %>"><%=  
                    cus.getAccount_number()+"----"+cus.getName() %></option>  
  
            <% } %>  
  
        </select>  
  
        <br>  
  
        <label for="sel2">Field(s) to be Updated:</label>  
  
        <select multiple class="form-control" id="sel2" name="fields">  
  
            <option value="email">Email Id</option>  
  
            <option value="mobile">Mobile</option>  
  
            <option value="aadhaar">Aadhaar</option>  
  
            <option value="pan">Pan</option>  
  
            <option value="address">Address</option>  
  
        </select>  
  
    </div>  
  
<br>  
  
<input type="submit" value="Next"/>
```

```
</form>  
</body>  
</html>
```

Update customer next.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
  
<html lang="en">  
  
<head>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <style>  
        form{  
            text-align: center;  
            padding: 50px;  
        }  
        label{  
            font-size: 18px;  
            font-style: italic;  
            font-family: 'Courier New', Courier, monospace;  
        }  
        input[type=text],[type=email]{  
            padding: 10px;  
            text-align: center;  
            font-size: 18px;  
        }  
        input[type=submit]{
```

```
padding: 10px;  
text-align: center;  
font-size: 20px;  
text-transform: uppercase;  
border-radius: 10px;  
background-color: aquamarine;  
  
}  
  
textarea{  
width: 400px;  
height:100px;  
font-family: sans-serif;  
font-size: 18px;  
top:25px;  
position: relative;  
}  
  
</style>  
  
</head>  
  
<body>  
<%  
String acc_num=request.getParameter("account_num");  
String[] fields=request.getParameterValues("fields");  
session.setAttribute("fields", fields);  
%>  
<form action="UpdateCustomer" method="POST">  
<label for="anum">Account Number: &nbsp;</label>  
<input type="text" name="acc_num" id="anum" value="<%=" acc_num %>" readonly/>
```

```
<br><br>

<% for(String field:fields) { %>

<label for="usr"><%= field.toUpperCase() %>: &nbsp;</label>

<%
    if(field.equalsIgnoreCase("address")){
        %>
        <textarea name="<%= field %>" id="usr" required></textarea><br><br>
    <%
    }

    if(field.equalsIgnoreCase("mobile") || field.equalsIgnoreCase("pan")){
        %>
        <input type="text" name="<%= field %>" id="usr" maxlength="10" minlength="10"
required/>
        <br><br>
    <%
    }

    if(field.equalsIgnoreCase("aadhaar")){
        %>
        <input type="text" name="<%= field %>" id="usr" maxlength="12" minlength="12"
required/>
        <br><br>
    <%
    }

    if(field.equalsIgnoreCase("email")){
        %>
        <input type="email" name="<%= field %>" id="usr" required/>
        <br><br>
    <%
    }
}
```

```

        }

%>

<br><br><br>

<input type="submit" value="UPDATE"/>

</form>

</body>

</html>

```

Employee_details.jsp

```

<%@page import="bank.connect_dao.DBOperation"%>

<%@page import="bank.model.Employee"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

</head>

<style>

.table th{

    text-align: center;

}

</style>

<body>

<%
Cookie ck=null;

```

```
Cookie[] c=request.getCookies();
for(int i=0;i<c.length;i++){
    if(c[i].getName().equals("employee_id")){
        ck=c[i];
        break;
    }
}
String value=ck.getValue();
long emp_id=Integer.parseInt(value);
Employee emp=DBOperation.fetchEmployeeById(emp_id);
%>
<div class="container">
<table class="table table-striped" style="text-align: center;">
<thead style="padding-left: 50px;">
<tr>
<th>ID</th>
<th>NAME</th>
<th>MOBILE</th>
<th>EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td><%= emp.getEmployee_id() %></td>
<td><%= emp.getName() %></td>
<td><%= emp.getMobile() %></td>
<td><%= emp.getEmail() %></td>
</tr>
```

```

</tbody>
</table>
</div>
</body>
</html>

```

Remove_customer.jsp

```

<%@page import="bank.model.Customer"%>
<%@page import="java.util.List"%>
<%@page import="bank.connect_dao.DBOperation"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<script>
$(document).ready(function(){
    $("#sel1").change(function(){
        var account_num=$(this).val();
        $.ajax({
            method:"get",
            url: "FetchCustomer",
            data: {acc_num:account_num},
            success: function (data, textStatus, jqXHR) {

```

```
        $("#" + cus_id).html(data);
    },
    error: function (jqXHR, textStatus, errorThrown) {
        console.log(errorThrown);
    }
});

var link = $(this).parents().find("#link");
window.console.log("'" + $(this).val());
link.attr("href", "RemoveCustomer?id=" + $(this).val());
});

});

});


```

```
</script>

</head>

<body>

<%
Cookie ck=null;
Cookie[] c=request.getCookies();
for(int i=0;i<c.length;i++){
if(c[i].getName().equals("employee_id")){
ck=c[i];
break;
}
String value=ck.getValue();
long emp_id=Integer.parseInt(value);
```

```

List<Customer> list=DBOperation.returnAllCustomerByIFSC(emp_id);

%>

<div class="container-fluid" style="text-align: center;padding-top: 30px;">

<form id="#form" name="customer">

<div class="form-group" style="font-size: 20px;">

<label for="sel1">Select Customer:</label>

<select class="form-control text-center" id="sel1" >

<option value="">Select a Customer</option>

<%

for(Customer cus:list){

%>

<option value="<%= cus.getAccount_number() %>"><%= cus.getAccount_number()+"---"+cus.getName() %></option>

<%

}

%>

</select>

</div>

</form>

<div class="row" id="cus_detatils" style="height: 200px; font-family: cursive; font-size:25px; ">

</div>

</div>

<a href="" id="link"><button id="button" class="btn btn-primary text-center" style="margin-top: 50px; margin-left: 43%;">Remove Customer</button></a>

</body>

</html>

```

Transactions.jsp

```

<% @page import="bank.model.Customer"%>

<% @page import="java.util.List"%>

```

```
<%@page import="bank.connect_dao.DBOperation"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<script>
$(document).ready(function(){
    $("#anum").change(function(){
        var account_num=$(this).val();
        $.ajax({
            method:"post",
            url: "CustomerTransaction",
            data: {acc_num:account_num},
            success: function (data, textStatus, jqXHR) {
                $("#record").html(data);
            },
            error: function (jqXHR, textStatus, errorThrown) {
                console.log(errorThrown);
            }
        });
    });
});
</script>
```

```
</head>

<body>

<%

Cookie ck=null;

Cookie[] c=request.getCookies();

for(int i=0;i<c.length;i++){

if(c[i].getName().equals("employee_id")){

ck=c[i];

break;

}

}

String value=ck.getValue();

long emp_id=Integer.parseInt(value);

List<Customer> list=DBOperation.returnAllCustomerByIFSC(emp_id);

%>

<div class="container" style="padding-top: 50px;">

<form id="#form" name="customer_transactions">

<div class="form-group">

<label for="anum" style="font-size: 24px; margin-left: 35%;">Select Customer Account Number </label>

<select class="form-control text-center" id="anum" name="acc_num" >

<option value="">Select an account number</option>

<% for(Customer cus:list){ %>

<option value="<%=" cus.getAccount_number() %>" style="text-align: center;"><%=" cus.getAccount_number() %>&nbsp;/&nbsp;<%=" cus.getName() %></option>

<% } %>

</select>

</div>

</form>
```

```
<br><br><br>

<table class="table text-center">

  <thead class="thead-dark">

    <tr>

      <th class="text-center">Transaction No.</th>
      <th class="text-center">Comment</th>
      <th class="text-center">Debit</th>
      <th class="text-center">Credit</th>
      <th class="text-center">Transaction Date</th>

    </tr>
  </thead>

  <tbody id="record">

  </tbody>
</table>

</div>

</body>

</html>
```

Deposit.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>

      form{

        text-align: center;

        padding: 50px;
```

```
}

label{
    font-size: 18px;
    font-style: italic;
    font-family: 'Courier New', Courier, monospace;
}

input[type=text]{
    padding: 10px;
    text-align: center;
    font-size: 18px;
}

input[type=submit]{
    padding: 10px;
    text-align: center;
    font-size: 20px;
    text-transform: uppercase;
    border-radius: 10px;
    background-color: aquamarine;
}

</style>

</head>

<body>

<form action="Deposit" method="POST">

<label for="account_num">Customer Account Number: &nbsp;&nbsp;&nbsp;</label>

<input type="text" name="account_num" id="account_num" minlength="8" maxlength="8" onfocusout="account_number()" required/>

<br><br><br>

<label for="amount">Amount: &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</label>

<input type="text" name="amount" id="amount" onfocusout="deposit()" required/>
```

```

<br><br><br>

<input type="submit" name="deposit" value="deposit">

</form>

<script>

function account_number() {

    var x;

    x = document.getElementById("account_num").value;

    if (isNaN(x)) {

        alert("Please enter a 8 digit account number");

    }

}

function deposit() {

    var x,y;

    x = document.getElementById("amount").value;

    y=parseInt(x);

    if (isNaN(x)) {

        alert("Invalid input in amount field!!!!");

    }

    else if(y<=0 || y>1000000){

        alert("Max deposit limit---- Rs. 10,00,000 \- ");

    }

}

</script>

</body>

</html>

```

Listcustomers.jsp

```

<% @page import="java.util.List"%>

<% @page import="bank.connect_dao.DBOperation"%>

```

```
<%@page import="bank.model.Customer"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<style>

.table th{

    text-align: center;

}

</style>

</head>

<body>

<%

Cookie ck=null;

Cookie[] c=request.getCookies();

for(int i=0;i<c.length;i++){

    if(c[i].getName().equals("employee_id")){

        ck=c[i];

        break;

    }

}

String value=ck.getValue();
```

```
long emp_id=Integer.parseInt(value);

List<Customer> list=DBOperation.returnAllCustomerByIFSC(emp_id);

%>

<div class="container">

<table class="table table-striped" style="text-align: center;">

<thead style="padding-left: 50px;">

<tr>

<th>ACCOUNT NUMBER</th>

<th>NAME</th>

<th>MOBILE</th>

<th>EMAIL</th>

<th>BALANCE</th>

<th>AADHAAR NO.</th>

</tr>

</thead>

<tbody>

<%

    for(Customer emp:list){

%>

<tr>

<td><%= emp.getAccount_number() %></td>

<td><%= emp.getName() %></td>

<td><%= emp.getMobile() %></td>

<td><%= emp.getEmail() %></td>

<td><%= emp.getBalance() %></td>

<td><%= emp.getAadhaar() %></td>

</tr>

<%
```

```
        }  
    %>  
  </tbody>  
</table>  
</div>  
</body>  
</html>
```

Freeze account.jsp

```
<%@page import="java.util.List"%>  
<%@page import="bank.connect_dao.DBOperation"%>  
<%@page import="bank.model.Customer"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
<head>  
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1">  
  <link rel="stylesheet"  
  href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">  
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>  
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>  
  <style>  
    .table th{  
      text-align: center;  
    }  
  </style>  
</head>  
<body>  
<%
```

```
Cookie ck=null;  
Cookie[] c=request.getCookies();  
  
for(int i=0;i<c.length;i++){  
    if(c[i].getName().equals("employee_id")){  
        ck=c[i];  
        break;  
    }  
}  
  
String value=ck.getValue();  
long emp_id=Integer.parseInt(value);  
List<Customer> list=DBOperation.returnAllCustomerByIFSC(emp_id);  
  
%>  
<div class="container">  
  
<table class="table table-striped" style="text-align: center;">  
    <thead style="padding-left: 50px;">  
        <tr>  
            <th>ACCOUNT NUMBER</th>  
            <th>NAME</th>  
            <th>EMAIL</th>  
            <th>BALANCE</th>  
            <th>NET-BANKING ALLOWED</th>  
            <th>ACTION</th>  
        </tr>  
    </thead>  
    <tbody>
```

```

<%
for(Customer emp:list){

%>

<tr>

<td><%= emp.getAccount_number() %></td>

<td><%= emp.getName() %></td>

<td><%= emp.getEmail() %></td>

<td><%= emp.getBalance() %></td>

<td><%= emp.getNetbankingAllowed().toUpperCase() %></td>

<td><button><a href="freeze_acc?acc_num=<%= emp.getAccount_number()%>">Enable/Disable</a></button></td>

</tr>

<%
}

%>

</tbody>

</table>

</div>

</body>

</html>

```

emp_logout.jsp

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<html>

<head>

<title>BANK EMPLOYEE DASHBOARD</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@4.7.0/css/font-awesome.min.css">

```

```

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<script src="https://kit.fontawesome.com/1a88ea4f70.js" crossorigin="anonymous"></script>

</head>

<body style="background-color: #e6f2ff;">
  <div class="container-fluid text-center" style="padding-top: 20%; color: #5c7d9a;">
    <i class="fas fa-spinner fa-spin fa-7x"></i>
    <br>
    <br>
    <p style="font-size: 35px; font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif; font-weight: bold;">Redirecting.....</p>
  </div>
</body>

<%
Cookie[] ck=request.getCookies();
for(int i=ck.length-1;i>0;i--){
  ck[i].setMaxAge(0);
  ck[i].setValue(null);
}
session.invalidate();
System.out.println("Session Closed");
response.setHeader("Refresh","5;url=../index.html");
%>
</html>

```

AddCustomer.java

```
package bank.employee;
```

```
import java.io.IOException;
import java.util.Date;
```

```
import java.util.Random;  
import javax.servlet.ServletContext;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.Cookie;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import bank.connect_dao.DBOperation;  
import bank.connect_dao.SendEmail;  
import bank.model.Customer;  
import bank.model.Employee;  
  
@WebServlet(name = "AddCustomer", urlPatterns = {" /employee/AddCustomer"})  
public class AddCustomer extends HttpServlet {  
  
    String host,port, sender_address, sender_pass;  
  
    @Override  
    public void init() throws ServletException {  
        ServletContext context=getServletContext();  
        host=context.getInitParameter("host");  
        port=context.getInitParameter("port");  
        sender_address=context.getInitParameter("sender_email");  
        sender_pass=context.getInitParameter("password");  
    }  
  
    @Override  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {  
  
    Cookie ck=null;  
  
    Cookie[] cks=request.getCookies();  
  
  
    for(int i=0;i<cks.length;i++){  
  
        if(cks[i].getName().equals("employee_id")){  
  
            ck=cks[i];  
  
            break;  
  
        }  
  
    }  
  
    String value=ck.getValue();  
  
    long emp_id=Long.parseLong(value);  
  
    Employee emp=DBOperation.fetchEmployeeById(emp_id);  
  
    Random ran=new Random();  
  
    long acc_num=DBOperation.returnAccountNumber();  
  
  
    if(acc_num==0){  
  
        acc_num+=10000000;  
  
    }  
  
    else{  
  
        acc_num+=1;  
  
    }  
  
    Customer c=new Customer();  
  
    String name=request.getParameter("name");  
  
    String mobile=request.getParameter("mobile");  
  
    String email=request.getParameter("email");  
  
    String aadhaar=request.getParameter("aadhaar");
```

```
String type_of_account=request.getParameter("type");
String address=request.getParameter("address");
Date date_of_creation=new Date();
String pan=request.getParameter("pan");
String gender=request.getParameter("gender");

c.setAccount_number(acc_num);
c.setName(name);
c.setMobile(mobile);
c.setEmail(email);
c.setType_of_account(type_of_account);
c.setAadhaar(aadhaar);
c.setPan(pan);
c.setBalance(0.00);
c.setDate_of_creation(date_of_creation);
c.setNetbankingAllowed("yes");
c.setAddress(address);
c.setGender(gender);

String subject="NEW ACCOUNT OPENING";
String message;
String gen;
if(gender.equals("Male")){
    gen="Mr.";
}
else{
    gen="Mrs.";
}
```

```

message = "<body style='text-align: center;'>\n" +
"    <div style='width:100%;'>\n" +
"        <h3 style='font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif; background-color: #C4D3DF; font-size: 30px; padding: 10px; color: white;'>Account Creation</h3>\n" +
"        <div style='background-color: #C4D3DF; padding: 15px;'>\n" +
"            <p style='font-size: 23px; font-family: 'Times New Roman', Times, serif;'>Welcome " + gen + " " + name.toUpperCase() + "</p>\n" +
"            <p style='font-family: 'Times New Roman', Times, serif; font-size: 20px;'>Thank you for opening a " + type_of_account + " account in our bank.</p><br>\n" +
"            <p style='font-family: 'Times New Roman', Times, serif; font-size: 20px;'>Your Account Details are</p><br>\n" +
"            <p style='font-family: 'Times New Roman', Times, serif; font-size: 20px;'><u><i>Account Number: </i></u></p>&nbsp;&nbsp;" + acc_num + "<br>\n" +
"            <p style='font-family: 'Times New Roman', Times, serif; font-size: 20px;'><u><i>Email Id: </i></u></p>&nbsp;&nbsp;" + email + "<br>\n" +
"            <p style='font-family: 'Times New Roman', Times, serif; font-size: 20px;'><u><i>Mobile: </i></u></p>&nbsp;&nbsp;" + mobile + "<br>\n" +
"            <p style='font-family: 'Times New Roman', Times, serif; font-size: 20px;'><u><i>Address: </i></u></p>&nbsp;&nbsp;" + address + "<br>\n" +
"\n" +
"        <br><br>\n" +
"\n" +
"        <span style='font-family: 'Times New Roman', Times, serif; font-size: 20px;'>To register for Online Banking, <a href='http://onlinebank-env-1.eba-qej9spxf.us-east-2.elasticbeanstalk.com/' style='text-decoration: none; font-size: 22px; font-style: italic;'>Click Here</a><br>Click on Sign Up and fill in the Details.</span>\n" +
"    </div>\n" +
"    </div>\n" +
"  </body>";

```

```

if(DBOperation.addCustomer(c,emp.getIFSCCode())){
    try{
        SendEmail.sendMail(host, port, sender_address, sender_pass, c.getEmail(), subject, message);
    }
}

```

```
        }catch(Exception e){  
            e.printStackTrace(response.getWriter());  
        }  
  
        response.getWriter().println("Customer Added");  
        response.setHeader("Refresh","2;url=listcustomer.jsp");  
    }  
  
    else{  
        response.getWriter().println("Error Occurred!!");  
        response.setHeader("Refresh","3;url=employee_details.jsp");  
    }  
}  
}
```

CustomerTransaction.java

```
package bank.employee;  
  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.text.SimpleDateFormat;  
import java.util.List;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import bank.connect_dao.DBOperation;  
import bank.model.Transactions;
```

```
@WebServlet(name = "CustomerTransaction", urlPatterns = { "/employee/CustomerTransaction" })
```

```

public class CustomerTransaction extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        SimpleDateFormat format=new SimpleDateFormat("E, dd MMM yyyy HH:mm:ss z");
        List<Transactions>
        records=DBOperation.fetchTransactions(Long.parseLong(request.getParameter("acc_num")));
        System.out.println("\n\n"+records.size()+"\n"+request.getParameter("acc_num"));
        try(PrintWriter out=response.getWriter()){
            for(Transactions tr:records){
                out.println("<tr>");
                out.println("<td>"+tr.getTransaction_no()+"</td>" +
                        "<td>"+tr.getComment()+"</td>" +
                        "<td>"+tr.getDebit()+"</td>" +
                        "<td>"+tr.getCredit()+"</td>" +
                        "<td>" +format.format(tr.getDateOfTransaction())+"</td>");
                out.println("</tr>");
            }
        }
    }
}

```

Deposit.java

```

package bank.employee;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;

```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import bank.connect_dao.DBOperation;
import bank.model.Customer;
import bank.model.Transactions;

@WebServlet(name = "Deposit", urlPatterns = { "/employee/Deposit" })
public class Deposit extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String acc_num=request.getParameter("account_num");
        String amount=request.getParameter("amount");
        double amt=Double.parseDouble(amount);

        Customer cus=DBOperation.returnCustomerByAccountNumber(Long.parseLong(acc_num));
        Transactions tr=new Transactions();
        tr.setComment("Deposit");
        tr.setDebit(" Rs. "+amount);
        tr.setCredit(" ");
        tr.setDateOfTransaction(new Date());
        tr.setAccount_number(cus.getAccount_number());
    }
}
```

```
if(cus!=null){  
    double bal=cus.getBalance();  
    bal+=amt;  
  
    if(DBOperation.updateBalance(bal, Long.parseLong(acc_num)) &&  
        DBOperation.commitTransaction(tr )){  
        response.getWriter().println("\n\n\tRs. "+amount+" deposited to account number:  
        "+acc_num+" successfully....\n\n");  
        response.setHeader("Refresh","2;url=listcustomer.jsp");  
    }  
}  
else{  
    response.getWriter().println("\n\n\tInvalid account number!!\n\n");  
    response.setHeader("Refresh","2;url=deposit.html");  
}  
}  
}
```

EmployeeLogin.java

```
package bank.employee;  
  
import java.io.IOException;  
import java.io.PrintWriter;  
import javax.servlet.RequestDispatcher;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;
```

```
import bank.connect_dao.DBOperation;  
import bank.model.Employee;  
  
@WebServlet(name = "EmployeeLogin", urlPatterns = {" /employee/emplogin.do"})  
public class EmployeeLogin extends HttpServlet {  
  
    @Override  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        response.setContentType("text/html; charset=UTF-8");  
        PrintWriter out = response.getWriter();  
        String username = request.getParameter("username");  
        String password = request.getParameter("password");  
        Employee e = DBOperation.loginEmployee(username, password);  
  
        if (e != null) {  
            request.setAttribute("employee", e);  
            RequestDispatcher rd = request.getRequestDispatcher("employee_dashboard.jsp");  
            rd.forward(request, response);  
        } else {  
            out.println("Invalid username & password!!");  
            response.setHeader("Refresh", "5;url=../index.html");  
        }  
    }  
}
```

FetchCustomer.java

```
package bank.employee;
```

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import bank.connect_dao.DBOperation;
import bank.model.Customer;

@WebServlet(name = "FetchCustomer", urlPatterns = { "/employee/FetchCustomer" })
public class FetchCustomer extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            String acc_num=request.getParameter("acc_num");
            Customer emp=DBOperation.returnCustomerByAccountNumber(Long.parseLong(acc_num));

            out.println("<br> <u>Name:</u>&nbsp; &nbsp;&nbsp; "+emp.getName());
            out.println("<br> <u>Email:</u> &nbsp; &nbsp;&nbsp; "+emp.getEmail());
            out.println("<br> <u>Mobile:</u> &nbsp; &nbsp;&nbsp; "+emp.getMobile());
            out.println("<br> <u>Aadhaar No.:</u> &nbsp; "+emp.getAadhaar());
            out.println("<br> <u>Balance:</u> &nbsp;&nbsp;&nbsp; "+emp.getBalance());
        }
    }
}
```

```
    }  
}  
}
```

FreezeAccount.java

```
package bank.employee;  
  
import bank.connect_dao.DBOperation;  
  
import bank.model.Customer;  
  
import java.io.IOException;  
  
import java.io.PrintWriter;  
  
import javax.servlet.ServletException;  
  
import javax.servlet.annotation.WebServlet;  
  
import javax.servlet.http.HttpServlet;  
  
import javax.servlet.http.HttpServletRequest;  
  
import javax.servlet.http.HttpServletResponse;  
  
@WebServlet(name = "FreezeAccount", urlPatterns = { "/employee/freeze_acc" })  
public class FreezeAccount extends HttpServlet {  
  
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        response.setContentType("text/html;charset=UTF-8");  
        String account_num=request.getParameter("acc_num");  
        Customer c=DBOperation.returnCustomerByAccountNumber(Long.parseLong(account_num));  
        String netBankingAllowed=c.getNetbankingAllowed();  
        if(DBOperation.freezeUnfreezeAccount(netBankingAllowed, c.getAccount_number())){  
  
            response.getWriter().println("Action Performed");  
        }  
    }  
}
```

```
        }

    else{
        response.getWriter().println("Error occurred");
    }

}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

}
```

UpdateCustomer.java

```
package bank.employee;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import bank.connect_dao.DBOperation;
```

```
@WebServlet(name = "UpdateCustomer", urlPatterns = { "/employee/UpdateCustomer" })  
  
public class UpdateCustomer extends HttpServlet {  
  
    @Override  
  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        String anum=request.getParameter("acc_num");  
  
        HttpSession session=request.getSession();  
        String[] str=(String[])session.getAttribute("fields");  
  
        //Generating the required Update Query  
        String string="update Customer set";  
        for(int i=0;i<str.length;i++){  
            if(str[i].equalsIgnoreCase("email")){  
                string+=" "+str[i]+"="+request.getParameter(str[i])+"";  
            }  
            else if(str[i].equalsIgnoreCase("mobile")){  
                string+=" "+str[i]+"="+request.getParameter(str[i]);  
            }  
            else if(str[i].equalsIgnoreCase("aadhaar")){  
                string+=" "+str[i]+"="+request.getParameter(str[i]);  
            }  
            else if(str[i].equalsIgnoreCase("pan")){  
                string+=" "+str[i]+"="+request.getParameter(str[i]);  
            }  
        }  
    }  
}
```

```

else if(str[i].equalsIgnoreCase("address")){
    string+=" "+str[i]+"="+request.getParameter(str[i]);
}

if(i>=0&&i<str.length-1){
    string+=" , ";
}

string+=" where account_number="+anum;

if(DBOperation.updateCustomerDetails(string,anum)){
    response.getWriter().println("Record Updated!!!!");
    response.setHeader("Refresh", "2;url=listcustomer.jsp");
}
else{
    response.getWriter().println("Error Occured!!! Please Try Later");
    response.setHeader("Refresh", "2;url=update_customer.jsp");
}
}
}

```

RemoveCustomer.java

```

package bank.employee;

import java.io.IOException;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;

```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import bank.connect_dao.DBOperation;
import bank.connect_dao.SendEmail;
import bank.model.Customer;

@WebServlet(name = "RemoveCustomer", urlPatterns = {"/employee/RemoveCustomer"})
public class RemoveCustomer extends HttpServlet {

    String host,port,sender_address,sender_pass;

    @Override
    public void init() throws ServletException {
        ServletContext context=getServletContext();
        host=context.getInitParameter("host");
        port=context.getInitParameter("port");
        sender_address=context.getInitParameter("sender_email");
        sender_pass=context.getInitParameter("password");
    }

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");

        String acc_num=request.getParameter("id");

        Customer cus=DBOperation.returnCustomerByAccountNumber(Long.parseLong(acc_num));
    }
}
```

```

String subject="Account Closing in Branch "+cus.getIFSCcode().getIFSC();

String message=<body style="text-align: center;">
"      <div style="width:100%;">
"          <h3 style="font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;background-color: #b00707;font-size: 30px;padding: 10px;color: white;">Account Removed</h3>
"          <div style="background-color: #C4D3DF;padding: 15px;">
"              <p style="font-size: 23px;font-family: 'Times New Roman', Times, serif;">Thank you
Mr./Mrs. <b>"+cus.getName().toUpperCase()+"</b></p>
"              <p style="font-family: 'Times New Roman', Times, serif;font-size: 20px;">Thank you for
closing your "+cus.getType_of_account()+" account in our bank.</p></div></div></body>";

if(DBOperation.removeCustomer(Long.parseLong(acc_num))){
    try{
        SendEmail.sendMail(host, port, sender_address, sender_pass, cus.getEmail(), subject,
message);
    }catch(Exception e){
        e.printStackTrace();
    }
    response.getWriter().print("Customer Removed!!!");
    response.setHeader("Refresh", "1;url=remove_customer.jsp");
}
else{
    response.getWriter().println("Error Ocurred");
}
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

```

```

processRequest(request, response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

}

```

3. MANAGER MODULE:

Manager login.html

```

<!DOCTYPE html>

<html>

<head>

<title>MANAGER LOGIN PAGE</title>

<link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">

<script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js"
crossorigin="anonymous"></script>

<script type="text/javascript">

function preventBack() {

    window.history.forward();

}

setTimeout("preventBack()", 0);

window.onunload = function () { null; };

</script>

<style>

html {

```

```
background-color: #56baed;  
}  
  
body {  
font-family: "Poppins", sans-serif;  
height: 100vh;  
}  
  
a {  
color: #92badd;  
display:inline-block;  
text-decoration: none;  
font-weight: 400;  
}  
  
h2 {  
text-align: center;  
font-size: 16px;  
font-weight: 600;  
text-transform: uppercase;  
display:inline-block;  
margin: 40px 8px 10px 8px;  
color: #cccccc;  
}  
/* STRUCTURE */  
  
.wrapper {  
display: flex;
```

```
align-items: center;  
flex-direction: column;  
justify-content: center;  
width: 100%;  
min-height: 100%;  
padding: 20px;  
}  
  
  
#formContent {  
-webkit-border-radius: 10px 10px 10px 10px;  
border-radius: 10px 10px 10px 10px;  
background: #fff;  
padding: 30px;  
width: 90%;  
max-width: 450px;  
position: relative;  
padding: 0px;  
-webkit-box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);  
box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);  
text-align: center;  
}  
  
  
#formFooter {  
background-color: #f6f6f6;  
border-top: 1px solid #dce8f1;  
padding: 25px;  
text-align: center;  
-webkit-border-radius: 0 0 10px 10px;
```

```
border-radius: 0 0 10px 10px;  
}  
  
/* TABS */  
  
h2.inactive {  
color: #cccccc;  
}  
  
h2.active {  
color: #0d0d0d;  
border-bottom: 2px solid #5fbae9;  
}  
  
/* FORM TYPOGRAPHY */  
  
input[type=button], input[type=submit], input[type=reset]  
background-color: #56baed;  
border: none;  
color: white;  
padding: 15px 80px;  
text-align: center;  
text-decoration: none;  
display: inline-block;  
text-transform: uppercase;  
font-size: 13px;  
-webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);  
box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);  
-webkit-border-radius: 5px 5px 5px 5px;
```

```
border-radius: 5px 5px 5px 5px;  
margin: 5px 20px 40px 20px;  
-webkit-transition: all 0.3s ease-in-out;  
-moz-transition: all 0.3s ease-in-out;  
-ms-transition: all 0.3s ease-in-out;  
-o-transition: all 0.3s ease-in-out;  
transition: all 0.3s ease-in-out;  
}
```

```
input[type=button]:hover, input[type=submit]:hover, input[type=reset]:hover {  
background-color: #39ace7;  
}
```

```
input[type=button]:active, input[type=submit]:active, input[type=reset]:active {  
-moz-transform: scale(0.95);  
-webkit-transform: scale(0.95);  
-o-transform: scale(0.95);  
-ms-transform: scale(0.95);  
transform: scale(0.95);  
}
```

```
input[type=text],input[type=password] {  
background-color: #f6f6f6;  
border: none;  
color: #0d0d0d;  
padding: 15px 32px;  
text-align: center;  
text-decoration: none;
```

```
display: inline-block;  
font-size: 16px;  
margin: 5px;  
width: 85%;  
  
border: 2px solid #f6f6f6;  
  
-webkit-transition: all 0.5s ease-in-out;  
-moz-transition: all 0.5s ease-in-out;  
-ms-transition: all 0.5s ease-in-out;  
-o-transition: all 0.5s ease-in-out;  
  
transition: all 0.5s ease-in-out;  
  
-webkit-border-radius: 5px 5px 5px 5px;  
border-radius: 5px 5px 5px 5px;  
}
```

```
input[type=text]:focus,input[type=password]:focus {  
  
background-color: #fff;  
  
border-bottom: 2px solid #5fbae9;  
}
```

```
input[type=text]:placeholder,input[type=password]:placeholder {  
  
color: #cccccc;  
}
```

```
/* ANIMATIONS */
```

```
/* Simple CSS3 Fade-in-down Animation */
```

```
.fadeInDown {  
    -webkit-animation-name: fadeInDown;  
    animation-name: fadeInDown;  
    -webkit-animation-duration: 1s;  
    animation-duration: 1s;  
    -webkit-animation-fill-mode: both;  
    animation-fill-mode: both;  
}  
  
@-webkit-keyframes fadeInDown {  
    0% {  
        opacity: 0;  
        -webkit-transform: translate3d(0, -100%, 0);  
        transform: translate3d(0, -100%, 0);  
    }  
    100% {  
        opacity: 1;  
        -webkit-transform: none;  
        transform: none;  
    }  
}  
  
@keyframes fadeInDown {  
    0% {  
        opacity: 0;  
        -webkit-transform: translate3d(0, -100%, 0);  
        transform: translate3d(0, -100%, 0);  
    }  
}
```

```
100% {  
    opacity: 1;  
    -webkit-transform: none;  
    transform: none;  
}  
  
}  
  
/* Simple CSS3 Fade-in Animation */  
  
@-webkit-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }  
  
@-moz-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }  
  
@keyframes fadeIn { from { opacity:0; } to { opacity:1; } }  
  
.fadeIn {  
    opacity:0;  
    -webkit-animation:fadeIn ease-in 1;  
    -moz-animation:fadeIn ease-in 1;  
    animation:fadeIn ease-in 1;  
    -webkit-animation-fill-mode:forwards;  
    -moz-animation-fill-mode:forwards;  
    animation-fill-mode:forwards;  
    -webkit-animation-duration:1s;  
    -moz-animation-duration:1s;  
    animation-duration:1s;  
}  
  
.fadeIn.first {  
    -webkit-animation-delay: 0.4s;  
    -moz-animation-delay: 0.4s;  
    animation-delay: 0.4s;
```

```
}

.fadeIn.second {
    -webkit-animation-delay: 0.6s;
    -moz-animation-delay: 0.6s;
    animation-delay: 0.6s;
}

.fadeIn.third {
    -webkit-animation-delay: 0.8s;
    -moz-animation-delay: 0.8s;
    animation-delay: 0.8s;
}

.fadeIn.fourth {
    -webkit-animation-delay: 1s;
    -moz-animation-delay: 1s;
    animation-delay: 1s;
}

/* Simple CSS3 Fade-in Animation */

.underlineHover:after {
    display: block;
    left: 0;
    bottom: -10px;
    width: 0;
    height: 2px;
    background-color: #56baed;
    content: "";
    transition: width 0.2s;
}
```

```
}

.underlineHover:hover {
    color: #0d0d0d;
}

.underlineHover:hover:after{
    width: 100%;
}

/* OTHERS */

*:focus {
    outline: none;
}

#icon {
    width:60%;
}

</style>

</head>

<body>

<div class="wrapper fadeInDown">

    <div id="formContent">

        <div class="fadeIn first">

            <i class="fas fa-user-circle fa-5x" style="color: cornflowerblue;"></i><br/><br>MANAGER LOGIN

        </div>

        <form action="manager_login.do" method="post">
```

```

    <input type="text" id="login" class="fadeIn second" name="username"
placeholder="login">

    <input type="password" id="password" class="fadeIn third" name="password"
placeholder="password">

    <input type="submit" class="fadeIn fourth" value="Log In">

</form>

<div id="formFooter">
    <a href=".//index.html">Back to Home</a>
</div>
</div>
</body>
</html>

```

Manager_dashboard.jsp

```

<%@page import="bank.model.Manager"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>BANK MANAGER DASHBOARD</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="icon" href="image//logo.png" type="image/x-icon"/>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

```

```
<script src="https://kit.fontawesome.com/1a88ea4f70.js" crossorigin="anonymous"></script>
```

```
<style>
```

```
.navbar-inverse .navbar-nav > li > a,  
.navbar-inverse .navbar-nav > li a:hover,  
.navbar-inverse .navbar-nav > li a:focus {  
color: black;  
background-color: transparent;  
font-size: 20px;  
}  
  
}
```

```
.navbar-fixed-left {  
width: 250px;  
position: fixed;  
border-radius: 0;  
height: 100%;  
margin-top: 80px;  
text-align: center;  
}
```

```
.navbar-fixed-left .navbar-nav > li {  
float: none; /* Cancel default li float: left */  
width: 100%;  
}
```

```
.navbar-fixed-left + .container {  
padding-left: 160px;  
}
```

```
/* On using dropdown menu (To right shift popup) */

.navbar-fixed-left .navbar-nav > li > .dropdown-menu {
    margin-top: -50px;
    margin-left: 250px;
    padding: 20px;
    font-size: 18px;

}

.navbar-fixed-left .dropdown > a:hover ,
.navbar .dropdown > a:active ,
.navbar .dropdown > a:visited,
.navbar-fixed-left .navbar-nav:hover {
    background-color: transparent;
    list-style: none;
}

.navbar-fixed-top .dropdown > a:hover,.navbar-fixed-top .dropdown > a:active{
    background-color: transparent;
}

.navbar-fixed-left .navbar-nav li a:hover{
    background-color: transparent;
}

</style>

<script type="text/javascript">

function preventBack() {

    window.history.forward();
}
```

```
}
```

```
setTimeout("preventBack()", 0);
```

```
window.onunload = function () { null; };
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<%
```

```
session.setMaxInactiveInterval(5*60);
```

```
Manager m=(Manager)request.getAttribute("manager");
```

```
Cookie ck=new Cookie("manager_id", String.valueOf(m.getManager_id()));
```

```
Cookie ck1=new Cookie("ifsc_id",String.valueOf(m.getIFSCcode()));
```

```
ck.setMaxAge(60*30);
```

```
ck1.setMaxAge(60*30);
```

```
response.addCookie(ck);
```

```
response.addCookie(ck1);
```

```
%>
```

```
<nav class="navbar navbar-inverse navbar-fixed-top shadow-lg p-3 mb-5 bg-white rounded" style="background-color: #a3a199; height: 80px;padding-top: 15px;">
```

```
<div class="container-fluid">
```

```
<div class="navbar-header">
```

```
  <a class="navbar-brand" href="#" style="color: rgb(10, 7, 7); font-weight: bold; font-size: 20px;">&nbsp;MANAGER DASHBOARD</a>
```

```
</div>
```

```
  <ul class="nav navbar-nav navbar-right " style="font-family: Georgia, 'Times New Roman', Times, serif; font-size: 20px; text-align: center;padding-right: 50px;">
```

```

<li><a href="manager_logout.jsp"><i class="fas fa-power-off"></i>&nbsp;Log
Out</a></li>

</li>

</ul>

</div>

</nav>

<nav class="navbar navbar-fixed-left" style="background-color: #f2e4aa;">

<ul class="nav navbar-nav" style="width: 100%; list-style: none; ">

<li><a href="manager_details.jsp" target="main_frame" style="font-size: 20px; text-
decoration: none; color: black;">Manager Details</a></li>

<li><a href="add_employee.html" target="main_frame" style="font-size: 20px; text-
decoration: none; color: black;">Add Employee</a></li>

<li><a href="remove_employee.jsp" target="main_frame" style="font-size: 20px; text-
decoration: none; color: black;">Remove Employee</a></li>

<li><a href="update_employee.jsp" target="main_frame" style="font-size: 20px; text-
decoration: none; color: black;">Update Employee</a></li>

<li><a href="list_employee.jsp" target="main_frame" style="font-size: 20px; text-
decoration: none; color: black;">List All Employee</a></li>

</ul>

</nav>

<iframe name="main_frame" src="manager_details.jsp" style="margin-left: 260px;
margin-top: 100px; width: 80%; height: 500px; border: none; "></iframe>

</body>

</html>

```

Add employee.html

```

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

```

```
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<style>

.form-group .form-control{

    padding: 20px;

    font-size: 20px;

}

.form-group .form-control::placeholder{

    text-align: center;

    font-size: 18px;

}

</style>

</head>

<body >

<div class="container" style="text-align: center; align-content: center;">

<h2> New Employee </h2>

<form action="AddEmployee" method="post" style="width: 50%; margin-left: 24%;">

<div class="form-group">

    <label for="name">Employee Name:</label>

    <input type="text" class="form-control text-center" id="name" maxlength="100"
placeholder="Enter Employee name" name="name" required >

</div>

<div class="form-group">

    <label for="email">Employee Email:</label>

    <input type="email" class="form-control text-center" id="email" maxlength="70"
placeholder="Enter Employee email" name="email" required>

</div>
```

```
<div class="form-group">
    <label for="mobile">Employee Mobile:</label>
    <input type="text" class="form-control text-center" id="mobile" placeholder="Enter Employee mobile" maxlength="10" minlength="10" name="mobile" required>
</div>

<label>Gender:&nbsp;&nbsp;</label>
<label class="radio-inline">
    <input type="radio" name="gender" value="Male" checked>Male
</label>
<label class="radio-inline">
    <input type="radio" name="gender" value="Female">Female
</label><br><br>
<div class="form-group">
    <label for="aadhaar">Employee Aadhaar No:</label>
    <input type="text" class="form-control text-center" id="aadhaar" placeholder="Enter Employee Aadhaar" maxlength="12" minlength="12" name="aadhaar" required>
</div>

<div class="form-group">
    <label for="sal">Employee Salary:</label>
    <input type="text" class="form-control text-center" id="pwd" placeholder="Enter Employee Salary in INR" name="salary" required>
</div>

<div class="form-group">
    <label for="address">Employee Address:</label>
    <textarea class="form-control" rows="6" id="address" name="address" required></textarea>
</div>

<input type="submit" class="btn bg-primary" name="submit" value="ADD EMPLOYEE"/>
<input type="reset" class="btn bg-primary" value="RESET FORM"/>
</form>
```

```
</div>  
</body>  
</html>
```

Manager_details.jsp

```
<%@page import="bank.connect_dao.DBOperation"%>  
<%@page import="bank.model.Manager"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <title>Bootstrap Example</title>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <link rel="stylesheet"  
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">  
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>  
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>  
</head>  
<style>  
    .table th{  
        text-align: center;  
    }  
</style>  
<body>  
<%  
    Cookie ck=null;  
    Cookie[] c=request.getCookies();  
    for(int i=0;i<c.length;i++){  
        if(c[i].getName().equals("manager_id")){
```

```
ck=c[i];  
break;  
}  
}  
  
String value=ck.getValue();  
long man_id=Integer.parseInt(value);  
  
Manager m=DBOperation.fetchManagerDetailById(man_id);  
  
%>  
  
<div class="container">  
  
<table class="table table-striped" style="text-align: center;">  
  
<thead style="padding-left: 50px;">  
  
<tr>  
  
<th>ID</th>  
  
<th>NAME</th>  
  
<th>MOBILE</th>  
  
<th>EMAIL</th>  
  
</tr>  
  
</thead>  
  
<tbody>  
  
<tr>  
  
<td><%= m.getManager_id() %></td>  
  
<td><%= m.getName() %></td>  
  
<td><%= m.getMobile() %></td>  
  
<td><%= m.getEmail() %></td>  
  
</tr>  
  
</tbody>  
  
</table>  
  
</div>
```

```
</body>
```

```
</html>
```

List_employee.jsp

```
<%@page import="bank.model.IFSCCode"%>
<%@page import="java.util.List"%>
<%@page import="bank.model.Employee"%>
<%@page import="bank.connect_dao.DBOperation"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<style>
.table th{
    text-align: center;
}
</style>
<body>
<%
Cookie ck=null;
Cookie[] c=request.getCookies();
for(int i=0;i<c.length;i++){
    if(c[i].getName().equals("manager_id")){

```

```
        ck=c[i];  
        break;  
    }  
}  
  
String value=ck.getValue();  
  
long man_id=Integer.parseInt(value);  
  
List<Employee> list=DBOperation.returnAllEmployeeByIFSC(man_id);  
  
%>  
  
<div class="container">  
  
<table class="table table-striped" style="text-align: center;">  
  
<thead style="padding-left: 50px;">  
  
<tr>  
  
    <th>EMPLOYEE ID</th>  
  
    <th>NAME</th>  
  
    <th>MOBILE</th>  
  
    <th>EMAIL</th>  
  
    <th>SALARY</th>  
  
    <th>AADHAAR NO.</th>  
  
</tr>  
  
</thead>  
  
<tbody>  
  
<%  
  
    for(Employee emp:list){  
  
        %>  
  
<tr>  
  
    <td><%= emp.getEmployee_id() %></td>  
  
    <td><%= emp.getName() %></td>  
  
    <td><%= emp.getMobile() %></td>
```

```

<td><%= emp.getEmail() %></td>
<td><%= emp.getSalary() %></td>
<td><%= emp.getAadhaar() %></td>
</tr>
<%
}
%>
</tbody>
</table>
</div>
</body>
</html>

```

update_employee.jsp

```

<%@page import="java.util.List"%>
<%@page import="bank.connect_dao.DBOperation"%>
<%@page import="bank.model.Employee"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<style>
select,option{
    text-align: center;

```

```
font-size: 20px;  
padding: 5px;  
}  
  
form{  
    padding-top: 20px;  
    text-align: center;  
    font-size: 24px;  
    font-family: 'Courier New', Courier, monospace;  
}  
  
input[type=submit]{  
    padding: 7px;  
    width: 150px;  
    border-radius: 4px;  
}  
</style>  
</head>  
<body>  
<%  
Cookie ck=null;  
Cookie[] c=request.getCookies();  
  
for(int i=0;i<c.length;i++){  
    if(c[i].getName().equals("manager_id")){  
        ck=c[i];  
        break;  
    }
}
```

```
}

String value=ck.getValue();

long m_id=Integer.parseInt(value);

List<Employee> list=DBOperation.returnAllEmployeeByIFSC(m_id);

%>

<form action="update_employee_next.jsp" method="POST">

<div class="form-group">

<label for="sel1">Select Employee ID: </label>

<select class="form-control" id="sel1" name="emp_id">

<% for(Employee emp:list){ %>

<option value="<%=" emp.getEmployee_id() %>"><%=" emp.getEmployee_id()+"----"+emp.getName() %></option>

<% } %>

</select>

<br>

<label for="sel2">Field(s) to be Updated:</label>

<select multiple class="form-control" id="sel2" name="fields">

<option value="email">Email Id</option>

<option value="mobile">Mobile</option>

<option value="address">Address</option>

</select>

</div>

<br>

<input type="submit" value="Next"/>

</form>

</body>

</html>
```

Update_employee_next.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

form{

    text-align: center;

    padding: 50px;

}

label{

    font-size: 18px;

    font-style: italic;

    font-family: 'Courier New', Courier, monospace;

}

input[type=text],[type=email]{

    padding: 10px;

    text-align: center;

    font-size: 18px;

}

input[type=submit]{

    padding: 10px;

    text-align: center;

    font-size: 20px;

    text-transform: uppercase;

    border-radius: 10px;

}
```

```
background-color: aquamarine;  
}  
  
textarea{  
width: 400px;  
height:100px;  
font-family: sans-serif;  
font-size: 18px;  
top:25px;  
position: relative;  
}  
  
</style>  
  
</head>  
  
<body>  
  
<%  
  
String emp_id=request.getParameter("emp_id");  
  
String[] fields=request.getParameterValues("fields");  
  
session.setAttribute("fields", fields);  
  
%>  
  
<form action="UpdateEmployee" method="POST">  
  
    <label for="eid">Employee ID: &nbsp;</label>  
  
    <input type="text" name="emp_id" id="eid" value="<% = emp_id %>" readonly/>  
  
    <br><br>  
  
    <% for(String field:fields) { %>  
  
        <label for="usr"><% = field.toUpperCase() %>: &nbsp;</label>  
  
        <%  
  
            if(field.equalsIgnoreCase("address")){  
  
                %>  
  
                <textarea name="<% = field %>" id="usr" required></textarea><br><br>
```

```

<%
}

if(field.equalsIgnoreCase("mobile")){
%>

<input type="text" name=<%= field %> id="usr" maxlength="10" minlength="10"
required/>

<br><br>

<%
}

if(field.equalsIgnoreCase("email")){
%>

<input type="email" name=<%= field %> id="usr" required/>

<br><br>

<%
}

%>

<br><br><br>

<input type="submit" value="UPDATE"/>

</form>

</body>

</html>

```

Remove_employee.jsp

```

<%@page import="java.util.List"%>

<%@page import="bank.connect_dao.DBOperation"%>

<%@page import="bank.model.Employee"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<html>

<head>

```

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<script>
$(document).ready(function(){
    $("#sel1").change(function(){
        var emp=$(this).val();
        $.ajax({
            method:"get",
            url: "employee.do",
            data: {emp_id:emp},
            success: function (data, textStatus, jqXHR) {
                $("#emp_detatils").html(data);
            },
            error: function (jqXHR, textStatus, errorThrown) {
                console.log(errorThrown);
            }
        });
        var link = $(this).parents().find("#link");
        window.console.log(""+$(this).val());
        link.attr("href", "RemoveEmployee?id=" + $(this).val());
    });
});
```

```
</script>

</head>

<body>

<%
Cookie ck=null;

Cookie[] c=request.getCookies();

for(int i=0;i<c.length;i++){

if(c[i].getName().equals("manager_id")){

ck=c[i];

break;

}

}

String value=ck.getValue();

long man_id=Integer.parseInt(value);

List<Employee> list=DBOperation.returnAllEmployeeByIFSC(man_id);

%>

<div class="container-fluid" style="text-align: center;padding-top: 30px;">

<form id="#form" name="employee">

<div class="form-group" style="font-size: 20px;">

<label for="sel1">Select Employee:</label>

<select class="form-control text-center" id="sel1" >

<option value="">Select an Employee</option>

<%
for(Employee e:list){

%>

<option value="<%=" e.getEmployee_id() %>"><%=" e.getEmployee_id()+"-----%>e.getName() %></option>

<%
}
```

```

%>

</select>

</div>

</form>

<div class="row" id="emp_detatils" style="height: 200px; font-family: cursive; font-size:25px; ">

</div>

</div>

<a href="" id="link"><button id="button" class="btn btn-primary text-center" style="margin-top: 50px; margin-left: 43%;">Remove Employee</button></a>

</body>

</html>

```

Manager_logout.jsp

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<title>MANAGER LOGOUT</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<script src="https://kit.fontawesome.com/1a88ea4f70.js"
crossorigin="anonymous"></script>

</head>

<body style="background-color: #e0e0e0;">

<div class="container-fluid text-center" style="padding-top: 20%; color: #337ab7;">

<i class="fas fa-spinner fa-spin fa-7x"></i>

```

```

<br>
<br>

<p style="font-size: 35px;font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif; font-weight: bold;">Redirecting.....</p>

</div>
</body>

<%
Cookie[] ck=request.getCookies();
for(int i=ck.length-1;i>0;i--){
    ck[i].setMaxAge(0);
    ck[i].setValue(null);
}
session.invalidate();
System.out.println("Session Closed");
response.setHeader("Refresh","5;url=../index.html");
%>
</html>

```

AddEmployee.java

```

package bank.manager;

import java.io.IOException;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import bank.connect_dao.DBOperation;

```

```
import bank.connect_dao.SendEmail;  
import bank.model.Employee;  
import bank.model.Manager;  
  
@WebServlet(name = "AddEmployee", urlPatterns = {"/manager/AddEmployee"})  
public class AddEmployee extends HttpServlet {  
  
    String host,port,sender_address, sender_pass;  
  
    @Override  
    public void init() throws ServletException {  
        ServletContext context=getServletContext();  
        host=context.getInitParameter("host");  
        port=context.getInitParameter("port");  
        sender_address=context.getInitParameter("sender_email");  
        sender_pass=context.getInitParameter("password");  
    }  
  
    @Override  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        Cookie ck=null;  
        Cookie[] c=request.getCookies();  
  
        for(int i=0;i<c.length;i++){  
            if(c[i].getName().equals("manager_id")){  
                ck=c[i];  
                break;  
            }  
        }  
    }  
}
```

```
}

String value=ck.getValue();

long man_id=Long.parseLong(value);

Manager m=DBOperation.fetchManagerDetailById(man_id);

String[] name=request.getParameter("name").split(" ");

Employee emp=new Employee();

emp.setName(request.getParameter("name"));

emp.setEmail(request.getParameter("email"));

emp.setMobile(request.getParameter("mobile"));

emp.setAadhaar(request.getParameter("aadhaar"));

emp.setGender(request.getParameter("gender"));

emp.setUsername(name[0].toLowerCase().concat(request.getParameter("aadhaar").substring(7)));

emp.setPassword("#"+name[0].toUpperCase().concat(request.getParameter("mobile").substring(6)));

emp.setSalary(Double.parseDouble(request.getParameter("salary")));

String subject="EMPLOYEE LOGIN DETAILS";

String message;

String gen;

if(emp.getGender().equals("Male")){

    gen="Mr./";

}

else{

    gen="Mrs./";

}
```

```

message="<body style="text-align: center;">\n" +
"      <div style="width:100%;\n" +
"          <h3 style="font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;background-color: #116280;font-size: 30px;padding: 10px;color: white;">Employee Login Details</h3>\n" +
"          <div style="background-color: #94d9f2;padding: 15px;\n" +
"              <p style="font-size: 23px;font-family: 'Times New Roman', Times, serif;">Welcome
"              "+gen+" <b>"+emp.getName().toUpperCase()+"</b></p>\n" +
"              <p style="font-family: 'Times New Roman', Times, serif;font-size: 20px;">Your
Login Details are</p><br>\n" +
"              <p style="font-family: 'Times New Roman', Times, serif;font-size:
20px;"><u><i>Username: </i></u></p>&nbsp;&nbsp;" +emp.getUsername()+"<br>\n" +
"              <p style="font-family: 'Times New Roman', Times, serif;font-size:
20px;"><u><i>Password: </i></u></p>&nbsp;&nbsp;" +emp.getPassword()+"<br>\n" +
"              <br><br>\n" +
"\n" +
"          <span style="font-family: 'Times New Roman', Times, serif;font-size: 20px;"><a
href="http://onlinebank-env-1.eba-qej9spxf.us-east-
2.elasticbeanstalk.com/employee/employee_login.html" style="text-decoration: none;font-size:
22px;font-style: italic;">Click Here</a><br> to login into Online Banking System..</span>\n" +
"      </div>\n" +
"  </div>\n" +
"  </body>";
}

if(DBOperation.addEmployee(emp,m.getIFSCcode())){
    try{
        SendEmail.sendMail(host, port, sender_address, sender_pass, emp.getEmail(),
subject, message);
    }catch(Exception e){
        e.printStackTrace(response.getWriter());
    }
    response.getWriter().println("Employee Added...\\n Mail sent to "+emp.getEmail());
    response.setHeader("Refresh", "2;url=list_employee.jsp");
}

```

```
    }

    else{

        response.getWriter().println("Error Occurred!! Employee Not added");

        response.setHeader("Refresh","3;url=manager_details.jsp");

    }

}

}
```

ManagerLogin.java

```
package bank.manager;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import bank.connect_dao.DBOperation;
import bank.model.Manager;

@WebServlet(name = "ManagerLogin", urlPatterns = { "/manager/manager_login.do" })
public class ManagerLogin extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

```

```

response.setContentType("text/html;charset=UTF-8");

PrintWriter out=response.getWriter();

String username=request.getParameter("username");

String password=request.getParameter("password");

Manager m=DBOperation.loginManager(username,password);

if(m!=null){

    request.setAttribute("manager", m);

    RequestDispatcher rd=request.getRequestDispatcher("manager_dashboard.jsp");

    rd.forward(request, response);

}

else{

    out.println("Invalid username & password!!");

    response.setHeader("Refresh", "5;url=../index.html");

}

}
}

```

FetchEmployee.java

```

package bank.manager;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import bank.connect_dao.DBOperation;

```

```
import bank.model.Employee;

@WebServlet(name = "FetchEmployee", urlPatterns = {" /manager/employee.do"})

public class FetchEmployee extends HttpServlet {

    @Override

    protected void doGet(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

        response.setContentType("text/html; charset=UTF-8");

        try (PrintWriter out = response.getWriter()) {

            String emp_id=request.getParameter("emp_id");

            Employee emp=DBOperation.fetchEmployeeById(Long.parseLong(emp_id));



            out.println("<br> <u>Name:</u>&nbsp; &nbsp;&nbsp; "+emp.getName());

            out.println("<br> <u>Email:</u> &nbsp; &nbsp;&nbsp; "+emp.getEmail());

            out.println("<br> <u>Mobile:</u> &nbsp; &nbsp;&nbsp; "+emp.getMobile());

            out.println("<br> <u>Aadhaar No.:</u> &nbsp; "+emp.getAadhaar());

            out.println("<br> <u>Salary:</u> &nbsp;&nbsp;&nbsp; "+emp.getSalary());

        }

    }

}
```

UpdateEmployee.java

```
package bank.manager;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import bank.connect_dao.DBOperation;

@WebServlet(name = "UpdateEmployee", urlPatterns = { "/manager/UpdateEmployee" })
public class UpdateEmployee extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String emp_id=request.getParameter("emp_id");

        HttpSession session=request.getSession();
        String[] str=(String[])session.getAttribute("fields");

        //Generating the required Update Query
        String string="update Employee set";
        for(int i=0;i<str.length;i++){
            if(str[i].equalsIgnoreCase("email")){
                string+=" "+str[i]+"='"+request.getParameter(str[i])+"'";
            }
            else if(str[i].equalsIgnoreCase("mobile")){
                string+=" "+str[i]+"='"+request.getParameter(str[i])+"'";
            }
            else if(str[i].equalsIgnoreCase("address")){
                string+=" "+str[i]+"='"+request.getParameter(str[i])+"'";
            }
        }
    }
}
```

```
if(i>=0&&i<str.length-1){  
    string+=" , ";  
}  
  
string+=" where employee_id="+emp_id;  
  
if(DBOperation.updateEmployee(string,emp_id)){  
    response.getWriter().println("Employee Record Updated!!!");  
}  
else{  
    response.getWriter().println("Error Occured!!! Please Try Later");  
    response.setHeader("Refresh", "2;url=update_employee.jsp");  
}  
}  
}
```

RemoveEmployee.java

```
package bank.manager;  
  
import java.io.IOException;  
import javax.servlet.ServletContext;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import bank.connect_dao.DBOperation;
```

```
import bank.connect_dao.SendEmail;  
import bank.model.Employee;  
  
@WebServlet(name = "RemoveEmployee", urlPatterns = { "/manager/RemoveEmployee"})  
public class RemoveEmployee extends HttpServlet {  
  
    String host,port,sender_address,sender_pass;  
  
    @Override  
    public void init() throws ServletException {  
  
        ServletContext context=getServletContext();  
  
        host=context.getInitParameter("host");  
  
        port=context.getInitParameter("port");  
  
        sender_address=context.getInitParameter("sender_email");  
  
        sender_pass=context.getInitParameter("password");  
    }  
  
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        response.setContentType("text/html;charset=UTF-8");  
  
        String emp_id=request.getParameter("id");  
  
        Employee emp=DBOperation.fetchEmployeeById(Long.parseLong(emp_id));  
  
        String subject="EMPLOYEE REMOVAL";  
        String gen;  
        if(emp.getGender().equals("Male")){  
            gen="Mr. ";  
        }  
    }
```

```

else{
    gen="Mrs.";
}

String message=<body style="text-align: center;">\n +
"    <div style="width:100%;">\n +
"        <h3 style="font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;background-color: rgb(114, 194, 124);font-size: 30px;padding: 10px;color: white;">THANK YOU</h3>\n +
"        <div style="background-color: #C4D3DF;padding: 15px;">\n +
"            <p style="font-size: 23px;font-family: 'Times New Roman', Times, serif;font-style:italic;">"+gen+" <b>" +emp.getName().toUpperCase()+"</b></p>\n +
"            <p style="font-family: 'Times New Roman', Times, serif;font-size: 20px;font-style:italic;">On behalf of '<b>Online Banking</b>,<br> sincerely thank you for your dedication and commitment for your services.</p><br>\n +
"            <p style="font-family: 'Times New Roman', Times, serif;font-size: 20px;font-style:italic;">Your work is truly commendable and we appreciate your contribution.</p><br>\n +
"            <p style="font-family: 'Times New Roman', Times, serif;font-size: 20px;font-style:italic;">Thank you for your services to our bank.</p><br>\n +
"            <br><br>\n +
"            <p style="font-family: 'Times New Roman', Times, serif;font-size: 20px;font-style:italic;">Regards,<br>Online Banking<br>Branch:
"            "+emp.getIFSCcode().getBranchName()+"<br>" +emp.getIFSCcode().getIFSC()+"</p><br>\n +
"\n +
"        </div>\n +
"    </div>\n +
"    </body>";
}

if(DBOperation.removeEmployee(Long.parseLong(emp_id))){
    try{
        SendEmail.sendMail(host, port, sender_address, sender_pass, emp.getEmail(),
subject, message);
    }catch(Exception e){
}
}

```

```
        e.printStackTrace();

    }

    response.getWriter().print("Employee Removed!!!!");

    response.setHeader("Refresh", "2;url=remove_employee.jsp");

}

else{

    response.getWriter().println("Error Ocurred");

}

}

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

    throws ServletException, IOException {

    processRequest(request, response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

    throws ServletException, IOException {

    processRequest(request, response);

}

}
```

4. ADMINISTRATOR MODULE:

Developer login.html

```
<!DOCTYPE html>

<html>

<head>

    <title>DEVELOPER LOGIN PAGE</title>

    <link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
```

```
<script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script src="https://kit.fontawesome.com/1a88ea4f70.js"
crossorigin="anonymous"></script>

<style>
html {
background-color: #56baed;
}

body {
font-family: "Poppins", sans-serif;
height: 100vh;
}

a {
color: #92badd;
display:inline-block;
text-decoration: none;
font-weight: 400;
}

h2 {
text-align: center;
font-size: 16px;
font-weight: 600;
text-transform: uppercase;
display:inline-block;
margin: 40px 8px 10px 8px;
color: #cccccc;
```

```
}
```

```
/* STRUCTURE */
```

```
.wrapper {  
    display: flex;  
    align-items: center;  
    flex-direction: column;  
    justify-content: center;  
    width: 100%;  
    min-height: 100%;  
    padding: 20px;  
}
```

```
#formContent {  
    -webkit-border-radius: 10px 10px 10px 10px;  
    border-radius: 10px 10px 10px 10px;  
    background: #fff;  
    padding: 30px;  
    width: 90%;  
    max-width: 450px;  
    position: relative;  
    padding: 0px;  
    -webkit-box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);  
    box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);  
    text-align: center;  
}
```

```
#formFooter {
```

```
background-color: #f6f6f6;  
border-top: 1px solid #dce8f1;  
padding: 25px;  
text-align: center;  
-webkit-border-radius: 0 0 10px 10px;  
border-radius: 0 0 10px 10px;  
}  
/* TABS */  
  
  
h2.inactive {  
color: #cccccc;  
}  
  
  
h2.active {  
color: #0d0d0d;  
border-bottom: 2px solid #5fbae9;  
}  
/* FORM TYPOGRAPHY */  
  
  
input[type=button], input[type=submit], input[type=reset] {  
background-color: #56baed;  
border: none;  
color: white;  
padding: 15px 80px;  
text-align: center;  
text-decoration: none;  
display: inline-block;  
text-transform: uppercase;
```

```
font-size: 13px;  
  
-webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);  
  
box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);  
  
-webkit-border-radius: 5px 5px 5px 5px;  
  
border-radius: 5px 5px 5px 5px;  
  
margin: 5px 20px 40px 20px;  
  
-webkit-transition: all 0.3s ease-in-out;  
  
-moz-transition: all 0.3s ease-in-out;  
  
-ms-transition: all 0.3s ease-in-out;  
  
-o-transition: all 0.3s ease-in-out;  
  
transition: all 0.3s ease-in-out;  
  
}
```

```
input[type=button]:hover, input[type=submit]:hover, input[type=reset]:hover {  
  
background-color: #39ace7;  
  
}
```

```
input[type=button]:active, input[type=submit]:active, input[type=reset]:active {  
  
-moz-transform: scale(0.95);  
  
-webkit-transform: scale(0.95);  
  
-o-transform: scale(0.95);  
  
-ms-transform: scale(0.95);  
  
transform: scale(0.95);  
  
}
```

```
input[type=text],input[type=password] {  
  
background-color: #f6f6f6;  
  
border: none;
```

```
color: #0d0d0d;  
padding: 15px 32px;  
text-align: center;  
text-decoration: none;  
display: inline-block;  
font-size: 16px;  
margin: 5px;  
width: 85%;  
  
border: 2px solid #f6f6f6;  
-webkit-transition: all 0.5s ease-in-out;  
-moz-transition: all 0.5s ease-in-out;  
-ms-transition: all 0.5s ease-in-out;  
-o-transition: all 0.5s ease-in-out;  
transition: all 0.5s ease-in-out;  
  
-webkit-border-radius: 5px 5px 5px 5px;  
border-radius: 5px 5px 5px 5px;  
}
```

```
input[type=text]:focus,input[type=password]:focus {  
  
background-color: #fff;  
  
border-bottom: 2px solid #5fbae9;  
}  
  
}
```

```
input[type=text]:placeholder,input[type=password]:placeholder {  
  
color: #cccccc;  
}  
  
/* ANIMATIONS */
```

```
/* Simple CSS3 Fade-in-down Animation */
```

```
.fadeInDown {  
    -webkit-animation-name: fadeInDown;  
    animation-name: fadeInDown;  
    -webkit-animation-duration: 1s;  
    animation-duration: 1s;  
    -webkit-animation-fill-mode: both;  
    animation-fill-mode: both;  
}
```

```
@-webkit-keyframes fadeInDown {
```

```
    0% {  
        opacity: 0;  
        -webkit-transform: translate3d(0, -100%, 0);  
        transform: translate3d(0, -100%, 0);  
    }  
    100% {
```

```
        opacity: 1;  
        -webkit-transform: none;  
        transform: none;  
    }  
}
```

```
@keyframes fadeInDown {
```

```
    0% {  
        opacity: 0;  
        -webkit-transform: translate3d(0, -100%, 0);  
        transform: translate3d(0, -100%, 0);  
    }
```

```
}

100% {

    opacity: 1;

    -webkit-transform: none;

    transform: none;

}

}

/* Simple CSS3 Fade-in Animation */

@-webkit-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@-moz-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

.fadeIn {

    opacity:0;

    -webkit-animation:fadeIn ease-in 1;

    -moz-animation:fadeIn ease-in 1;

    animation:fadeIn ease-in 1;

    -webkit-animation-fill-mode:forwards;

    -moz-animation-fill-mode:forwards;

    animation-fill-mode:forwards;

    -webkit-animation-duration:1s;

    -moz-animation-duration:1s;

    animation-duration:1s;

}
```

```
.fadeIn.first {  
    -webkit-animation-delay: 0.4s;  
    -moz-animation-delay: 0.4s;  
    animation-delay: 0.4s;  
}  
  
/* Simple CSS3 Fade-in Animation */
```

```
.fadeIn.second {  
    -webkit-animation-delay: 0.6s;  
    -moz-animation-delay: 0.6s;  
    animation-delay: 0.6s;  
}  
  
/* Simple CSS3 Fade-in Animation */
```

```
.fadeIn.third {  
    -webkit-animation-delay: 0.8s;  
    -moz-animation-delay: 0.8s;  
    animation-delay: 0.8s;  
}  
  
/* Simple CSS3 Fade-in Animation */
```

```
.fadeIn.fourth {  
    -webkit-animation-delay: 1s;  
    -moz-animation-delay: 1s;  
    animation-delay: 1s;  
}  
  
/* Simple CSS3 Fade-in Animation */
```

```
.underlineHover:after {
```

```
    display: block;  
    left: 0;
```

```
bottom: -10px;  
width: 0;  
height: 2px;  
background-color: #56baed;  
content: "";  
transition: width 0.2s;  
}  
  
.underlineHover:hover {  
color: #0d0d0d;  
}  
  
.underlineHover:hover:after{  
width: 100%;  
}  
  
/* OTHERS */  
  
*:focus {  
outline: none;  
}  
  
#icon {  
width:60%;  
}  
  
</style>  
</head>  
<body>  
<div class="wrapper fadeInDown">
```

```

<div id="formContent">

    <div class="fadeIn first">
        <i class="fas fa-user-circle fa-5x" style="color: cornflowerblue;"></i><br/><br>DEVELOPER LOGIN

    </div>

    <form action="admin.login" method="post">

        <input type="text" id="login" class="fadeIn second" name="username" placeholder="login">

        <input type="password" id="password" class="fadeIn third" name="password" placeholder="password">

        <input type="submit" class="fadeIn fourth" value="Log In">

    </form>

    <div id="formFooter">
        <a href="../index.html">Back to Home</a>
    </div>

    </div>

</div>

</body>

</html>

```

Admin_page.jsp

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

    </head>

```

```
<body>

<div class="container" style="text-align: center; align-content: center;">

    <h2> Add New Manager </h2>

    <form action="AddManager" method="post" style="width: 50%;margin-left: 24%;">

        <div class="form-group">

            <label for="name">Manager Name:</label>

            <input type="text" class="form-control text-center" id="name" maxlength="100" placeholder="Enter Manager name" name="name" required >

        </div>

        <div class="form-group">

            <label for="email">Manager Email:</label>

            <input type="email" class="form-control text-center" id="email" maxlength="70" placeholder="Enter Manager email" name="email" required>

        </div>

        <div class="form-group">

            <label for="mobile">Manager Mobile:</label>

            <input type="text" class="form-control text-center" id="mobile" placeholder="Enter Manager mobile" maxlength="10" minlength="10" name="mobile" required>

        </div>

        <label>Gender:&nbsp;&nbsp;</label>

        <label class="radio-inline">

            <input type="radio" name="gender" value="Male" checked>Male

        </label>

        <label class="radio-inline">

            <input type="radio" name="gender" value="Female">Female

        </label><br><br>

        <div class="form-group">

            <label for="m_address">Manager Address:</label>

            <textarea class="form-control" rows="6" id="m_address" name="manager_address" required></textarea>

        </div>

    </form>

</div>
```

```
</div>

<div class="form-group">
    <label for="ifsc">IFSC Code:</label>
    <input type="text" class="form-control text-center" id="ifsc" placeholder="Enter Bank IFSC Code" maxlength="10" minlength="10" name="ifsc" required>
</div>

<div class="form-group">
    <label for="branch">Branch:</label>
    <input type="text" class="form-control text-center" id="branch" placeholder="Enter Branch Name" name="branch" required>
</div>

<div class="form-group">
    <label for="b_address">Branch Address:</label>
    <textarea class="form-control" rows="6" id="b_address" name="address" required></textarea>
</div>

<input type="submit" class="btn bg-primary" name="submit" value="ADD MANAGER"/>
<input type="reset" class="btn bg-primary" value="RESET FORM"/>
</form>
</div>
</body>
</html>
```

Main_page.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>BANK ADMINISTRATOR DASHBOARD</title>
        <meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" href="image/../logo.png" type="image/x-icon"/>

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<script src="https://kit.fontawesome.com/1a88ea4f70.js" crossorigin="anonymous"></script>

<style>

.navbar-inverse .navbar-nav > li > a,
.navbar-inverse .navbar-nav > li a:hover,
.navbar-inverse .navbar-nav > li a:focus {
    color: black;
    background-color: transparent;
    font-size: 20px;
}

.navbar-fixed-left {
    width: 250px;
    position: fixed;
    border-radius: 0;
    height: 100%;
    margin-top: 80px;
    text-align: center;
}

.navbar-fixed-left .navbar-nav > li {
```

```
float: none; /* Cancel default li float: left */

width: 100%;

}

.navbar-fixed-left + .container {

padding-left: 160px;

}

/* On using dropdown menu (To right shift popued) */

.navbar-fixed-left .navbar-nav > li > .dropdown-menu {

margin-top: -50px;

margin-left: 250px;

padding: 20px;

font-size: 18px;

}

.navbar-fixed-left .dropdown > a:hover , .navbar .dropdown > a:active , .navbar .dropdown > a:visited, .navbar-fixed-left .navbar-nav:hover {

background-color: transparent;

list-style: none;

}

.navbar-fixed-top .dropdown > a:hover,.navbar-fixed-top .dropdown > a:active{

background-color: transparent;

}
```

```
.navbar-fixed-left .navbar-nav li a:hover{  
background-color: transparent;  
}  
</style>  

```

```

<nav class="navbar navbar-fixed-left" style="background-color: #f2e4aa;">
    <ul class="nav navbar-nav" style="width: 100%; list-style: none;">
        <li><a href="admin_page.jsp" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">Add manager</a></li>
        <li><a href="update_manager.jsp" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">Update Manager</a></li>
        <li><a href="list_managers.jsp" target="main_frame" style="font-size: 20px; text-decoration: none; color: black;">View All Managers</a></li>
    </ul>
</nav>

<iframe name="main_frame" style="margin-left: 260px; margin-top: 100px; width: 80%; height: 500px; border: none; ">Select An Option from The Menu !!!</iframe>

</body>
</html>

```

List managers.jsp

```

<%@page import="bank.connect_dao.DBOperation"%>
<%@page import="bank.model.Manager"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <style>
        .table th{
            text-align: center;

```

```
        }

    </style>

</head>

<body>

<%

    List<Manager> list=DBOperation.returnManager();

%>

<div class="container">

<table class="table table-striped" style="text-align: center;">

<thead style="padding-left: 50px;">

<tr>

<th>MANAGER ID</th>

<th>NAME</th>

<th>MOBILE</th>

<th>EMAIL</th>

<th>IFSC CODE</th>

<th>BRANCH</th>

</tr>

</thead>

<tbody>

<% for(Manager man:list){ %>

<tr>

<td><%= man.getManager_id() %></td>

<td><%= man.getName() %></td>

<td><%= man.getMobile() %></td>

<td><%= man.getEmail() %></td>

<td><%= man.getIFSCcode().getIFSC() %></td>

<td><%= man.getIFSCcode().getBranchName() %></td>
```

```
</tr>

<% } %>

</tbody>

</table>

</div>

</body>

</html>
```

Update_manager.jsp

```
<%@page import="bank.connect_dao.DBOperation"%>

<%@page import="bank.model.Manager"%>

<%@page import="java.util.List"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<style>

select,option{

text-align: center;

font-size: 20px;

padding: 5px;

}
```

```
form{  
    padding-top: 20px;  
    text-align: center;  
    font-size: 24px;  
    font-family: 'Courier New', Courier, monospace;  
}  
  
input[type=submit]{  
    padding: 7px;  
    width: 150px;  
    border-radius: 4px;  
}  
</style>  
  
</head>  
  
<body>  
<%  
    List<Manager> list=DBOperation.returnManager();  
%>  
<form action="update_manager_next.jsp" method="POST">  
    <div class="form-group">  
        <label for="sel1">Select Manager Id: </label>  
  
        <select class="form-control" id="sel1" name="manager_id">  
            <% for(Manager m:list){ %>  
                <option value=<%= m.getManager_id() %>><%= m.getManager_id() + "----"  
                    "+m.getIFSCcode().getIFSC() %></option>  
            <% } %>  
        </select>
```

```
<br>
<label for="sel2">Field(s) to be Updated:</label>
<select multiple class="form-control" id="sel2" name="fields">
    <option value="email">Email Id</option>
    <option value="mobile">Mobile</option>
    <option value="address">Address</option>
</select>
</div>
<br>
<input type="submit" value="Next"/>
</form>
</body>
</html>
```

Update manager next.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <style>
            form{
                text-align: center;
                padding: 50px;
            }
            label{
                font-size: 18px;
                font-style: italic;
            }
        </style>
    </head>
    <body>
        <h1 style="text-align: center; margin-bottom: 20px;">Welcome to Update Manager
```

```
font-family: 'Courier New', Courier, monospace;

}

input[type=text],[type=email]{
    padding: 10px;
    text-align: center;
    font-size: 18px;
}

input[type=submit]{
    padding: 10px;
    text-align: center;
    font-size: 20px;
    text-transform: uppercase;
    border-radius: 10px;
    background-color: aquamarine;
}

}

textarea{
    width: 400px;
    height:100px;
    font-family: sans-serif;
    font-size: 18px;
    top:25px;
    position: relative;
}

</style>

</head>
```

```
<body>

<%
    String manager_id=request.getParameter("manager_id");

    String[] fields=request.getParameterValues("fields");

    session.setAttribute("fields", fields);

%>

<form action="UpdateManager" method="POST">

    <label for="anum">Manager ID: &nbsp;</label>

    <input type="text" name="manager_id" id="anum" value="<%="manager_id %>" readonly/>

    <br><br>

    <% for(String field:fields) { %>

        <label for="usr"><%=field.toUpperCase()%>: &nbsp;</label>

        <%
            if(field.equalsIgnoreCase("address")){
                %>

                <textarea name="<%="field %>" id="usr" required></textarea><br><br>

            <%
            }

            if(field.equalsIgnoreCase("mobile")){
                %>

                <input type="text" name="<%="field %>" id="usr" maxlength="10" minlength="10" required/>

                <br><br>

            <%
            }

            if(field.equalsIgnoreCase("email")){
                %>

                <input type="email" name="<%="field %>" id="usr" required/>

```

```

<br><br>

<%
    }
}

%>

<br><br><br>

<input type="submit" value="UPDATE"/>

</form>

</body>

</html>

```

Admin_logout.jsp

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

    <title>ONLINE BANKING</title>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

    <script src="https://kit.fontawesome.com/1a88ea4f70.js" crossorigin="anonymous"></script>

</head>

<body style="background-color: #f0f0f0;">

    <div class="container-fluid text-center" style="padding-top: 20%; color: #337ab7;">

        <i class="fas fa-spinner fa-spin fa-7x"></i>

        <br>

```

```
<br>
<p style="font-size: 35px;font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif; font-weight: bold;">Redirecting.....</p>
</div>
</body>
<%
    session.invalidate();
    System.out.println("Session Closed");
    response.setHeader("Refresh", "5;url=../index.html");
%>
</html>
```

AdminServlet.java

```
package admin;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "AdminServlet", urlPatterns = {"/admin/admin.login"})
public class AdminServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
```

```

ServletContext context=getServletContext();

String username=request.getParameter("username");

String password=request.getParameter("password");

if(username.equals(context.getInitParameter("username"))&&password.equals(context.getInitParameter("pwd"))){

    response.sendRedirect("main_page.jsp");

}

else{

    response.getWriter().print("Invalid username && Password!!!");

    response.setHeader("Refresh", "2;url=../index.html");

}

}

}

```

AddManager.java

```

package admin;

import java.io.IOException;

import javax.mail.MessagingException;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import bank.connect_dao.DBOperation;

import bank.connect_dao.SendEmail;

import bank.model.IFSCCode;

import bank.model.Manager;

```

```
@WebServlet(name = "AddManager", urlPatterns = { "/admin/AddManager" })  
public class AddManager extends HttpServlet {  
  
    String host,port,sender_address,sender_pass;  
  
    @Override  
    public void init() throws ServletException {  
  
        ServletContext context=getServletContext();  
  
        host=context.getInitParameter("host");  
  
        port=context.getInitParameter("port");  
  
        sender_address=context.getInitParameter("sender_email");  
  
        sender_pass=context.getInitParameter("password");  
  
    }  
  
    @Override  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        response.setContentType("text/html;charset=UTF-8");  
  
        String address=request.getParameter("manager_address");  
  
        String name=request.getParameter("name");  
  
        String[] val=name.split(" ");  
  
        String mobile=request.getParameter("mobile");  
  
        String email=request.getParameter("email");  
  
        String gender=request.getParameter("gender");  
  
        String username=val[0].toLowerCase().concat(mobile.substring(3));  
  
        String password="@"+val[0].toUpperCase().concat(mobile.substring(5));  
  
        Manager m=new Manager();  
  
        m.setName(name);
```

```
m.setMobile(mobile);
m.setEmail(email);
m.setUsername(username);
m.setPassword(password);
m.setAddress(address);
m.setGender(gender);

IFSCCode ifsc=new IFSCCode();
ifsc.setIFSC(request.getParameter("ifsc"));
ifsc.setBranchName(request.getParameter("branch"));
ifsc.setAddress(request.getParameter("address"));

String subject="MANAGER LOGIN DETAILS";
String message;
String gen;
if(gender.equals("Male")){
    gen="Mr.";
}
else{
    gen="Mrs.";
}

message=<body style="text-align: center;">  

"<div style="width:100%;">  

"    <h3 style="font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif; background-color: #116280; font-size: 30px; padding: 10px; color: white;">Manager Login Details  

"    <div style="background-color: #94d9f2; padding: 15px;">  

"        <p style="font-size: 23px; font-family: 'Times New Roman', Times, serif;">Welcome  

"+gen+" <b>" + m.getName().toUpperCase() + "</b></p>  

"
```

```

"      <p style="font-family: 'Times New Roman', Times, serif;font-size: 20px;">Your
Login Details are</p><br>\n" +
"      <p style="font-family: 'Times New Roman', Times, serif;font-size:
20px;"><u><i>Username: </i></u></p>&nbsp;&nbsp;" +m.getUsername() + "<br>\n" +
"      <p style="font-family: 'Times New Roman', Times, serif;font-size:
20px;"><u><i>Password: </i></u></p>&nbsp;&nbsp;" +m.getPassword() + "<br>\n" +
"      <p style="font-family: 'Times New Roman', Times, serif;font-size:
20px;"><u><i>BRANCH:
</i></u></p>&nbsp;&nbsp;" +m.getIFSCcode().getIFSC() + ">>>" +m.getIFSCcode().getBranch
Name() + "<br>\n" +
"      <br><br>\n" +
"\n" +
"      <span style="font-family: 'Times New Roman', Times, serif;font-size: 20px;"><a
href="http://onlinebank-env-1.eba-qej9spxf.us-east-
2.elasticbeanstalk.com/manager/manager_login.html" style="text-decoration: none;font-size:
22px;font-style: italic;">Click Here</a><br> to login into Online Banking System..</span>\n" +
"      </div>\n" +
"      </div>\n" +
"    </body>";
}

if(DBOperation.addManager(m, ifsc)){
    try{
        SendEmail.sendMail(host, port, sender_address, sender_pass, m.getEmail(), subject,
message);
    }catch(MessagingException e){
        e.printStackTrace(response.getWriter());
    }
    response.getWriter().println("Manager added!!!");
    response.setHeader("Refresh", "2;url=admin_page.jsp");
}
else{
    response.getWriter().println("Manager not added!!!");
    response.setHeader("Refresh", "2;url=admin_page.jsp");
}

```

```
    }  
}  
}
```

UpdateManager.java

```
package admin;
```

```
import java.io.IOException;  
  
import javax.servlet.ServletException;  
  
import javax.servlet.annotation.WebServlet;  
  
import javax.servlet.http.HttpServlet;  
  
import javax.servlet.http.HttpServletRequest;  
  
import javax.servlet.http.HttpServletResponse;  
  
import javax.servlet.http.HttpSession;  
  
import bank.connect_dao.DBOperation;
```

```
@WebServlet(name = "UpdateManager", urlPatterns = { "/admin/UpdateManager"})
```

```
public class UpdateManager extends HttpServlet {
```

```
    @Override
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
```

```
        throws ServletException, IOException {
```

```
        String man_id=request.getParameter("manager_id");
```

```
        HttpSession session=request.getSession();
```

```
        String[] str=(String[])session.getAttribute("fields");
```

```
        //Generating the required Update Query
```

```
String string="update Manager set";
for(int i=0;i<str.length;i++){
    if(str[i].equalsIgnoreCase("email")){
        string+=" "+str[i]+"='"+request.getParameter(str[i])+"'";
    }
    else if(str[i].equalsIgnoreCase("mobile")){
        string+=" "+str[i]+"='"+request.getParameter(str[i]);
    }
    else if(str[i].equalsIgnoreCase("address")){
        string+=" "+str[i]+"='"+request.getParameter(str[i]);
    }

    if(i>=0&&i<str.length-1){
        string+=" , ";
    }
}

string+=" where manager_id="+man_id;

if(DBOperation.updateManager(string,man_id)){
    response.getWriter().println("Record Updated!!!\"");
}
else{
    response.getWriter().println("Error Occured!!! Please Try Later");
    response.setHeader("Refresh", "2;url=update_manager.jsp");
}
}
```

5. Plain Old Java Objects (POJO):

Customer.java

```
package bank.model;

import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Temporal;

@Entity
public class Customer {

    @Id
    private long account_number;
    @Column(nullable = true)
    private String name;
    @ManyToOne(cascade = CascadeType.ALL)
    private IFSCCode IFSCcode;
    @Column(nullable = true)
    private String email;
    @Column(nullable = true)
    private String mobile;
    @Column(nullable = true)
    private String username;
    @Column(nullable = true)
    private String password;
```

```
@Column(nullable = true)
private double balance;

@Column(nullable = true)
private String pan;

@Column(nullable = true)
private String aadhaar;

@Column(nullable = true)
@Temporal(javax.persistence.TemporalType.DATE)
private Date date_of_creation;

@Column(nullable = true)
private String type_of_account;

@Column(nullable = true)
private String netbankingAllowed;

@Column(nullable = true)
private String address;

private String gender;

public long getAccount_number() {
    return account_number;
}

public void setAccount_number(long account_number) {
    this.account_number = account_number;
}

public String getName() {
    return name;
}
```

```
public void setName(String name) {  
    this.name = name;  
}  
  
public IFSCCode getIFSCcode() {  
    return IFSCcode;  
}  
  
public void setIFSCcode(IFSCCode IFSCcode) {  
    this.IFSCcode = IFSCcode;  
}  
  
public String getGender() {  
    return gender;  
}  
  
public void setGender(String gender) {  
    this.gender = gender;  
}  
  
public String getEmail() {  
    return email;  
}  
  
public void setEmail(String email) {  
    this.email = email;  
}
```

```
public String getMobile() {  
    return mobile;  
}  
  
public void setMobile(String mobile) {  
    this.mobile = mobile;  
}  
  
public String getUsername() {  
    return username;  
}  
  
public void setUsername(String username) {  
    this.username = username;  
}  
  
public String getPassword() {  
    return password;  
}  
  
public void setPassword(String password) {  
    this.password = password;  
}  
  
public double getBalance() {  
    return balance;  
}
```

```
public void setBalance(double balance) {  
    this.balance = balance;  
}  
  
public String getPan() {  
    return pan;  
}  
  
public void setPan(String pan) {  
    this.pan = pan;  
}  
  
public String getAadhaar() {  
    return aadhaar;  
}  
  
public void setAadhaar(String aadhaar) {  
    this.aadhaar = aadhaar;  
}  
  
public Date getDate_of_creation() {  
    return date_of_creation;  
}  
  
public void setDate_of_creation(Date date_of_creation) {  
    this.date_of_creation = date_of_creation;  
}
```

```
public String getType_of_account() {  
    return type_of_account;  
}  
  
public void setType_of_account(String type_of_account) {  
    this.type_of_account = type_of_account;  
}  
  
public String getNetbankingAllowed() {  
    return netbankingAllowed;  
}  
  
public void setNetbankingAllowed(String netbankingAllowed) {  
    this.netbankingAllowed = netbankingAllowed;  
}  
  
public String getAddress() {  
    return address;  
}  
  
public void setAddress(String address) {  
    this.address = address;  
}  
}
```

Employee.java

```
package bank.model;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long employee_id;

    private String name;

    private String mobile;

    private String email;

    @ManyToOne(cascade = CascadeType.ALL)
    private IFSCCode IFSCcode;

    private double salary;

    private String aadhaar;

    private String username;

    private String password;

    private String address;

    private String gender;
```

```
public String getGender() {  
    return gender;  
}  
  
public void setGender(String gender) {  
    this.gender = gender;  
}  
  
public String getUsername() {  
    return username;  
}  
  
public void setUsername(String username) {  
    this.username = username;  
}  
  
public String getPassword() {  
    return password;  
}  
  
public void setPassword(String password) {  
    this.password = password;  
}  
  
public String getName() {  
    return name;  
}
```

```
public void setName(String name) {  
    this.name = name;  
}  
  
public String getMobile() {  
    return mobile;  
}  
  
public void setMobile(String mobile) {  
    this.mobile = mobile;  
}  
  
public String getEmail() {  
    return email;  
}  
  
public void setEmail(String email) {  
    this.email = email;  
}  
  
public long getEmployee_id() {  
    return employee_id;  
}  
  
public void setEmployee_id(long employee_id) {  
    this.employee_id = employee_id;  
}
```

```
public IFSCCode getIFSCcode() {  
    return IFSCcode;  
}  
  
public void setIFSCcode(IFSCCode IFSCcode) {  
    this.IFSCcode = IFSCcode;  
}  
  
public double getSalary() {  
    return salary;  
}  
  
public void setSalary(double salary) {  
    this.salary = salary;  
}  
  
public String getAadhaar() {  
    return aadhaar;  
}  
  
public void setAadhaar(String aadhaar) {  
    this.aadhaar = aadhaar;  
}  
  
public String getAddress() {  
    return address;  
}
```

```
public void setAddress(String address) {  
    this.address = address;  
}  
}
```

Manager.java

```
package bank.model;  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.JoinColumn;  
import javax.persistence.OneToOne;
```

```
@Entity
```

```
public class Manager{  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private long manager_id;  
    private String name;  
    private String mobile;  
    private String address;  
    private String email;  
    private String username;  
    private String password;  
    private String gender;  
    @OneToOne  
    @JoinColumn(name="id")
```

```
private IFSCCode IFSCcode;

public long getManager_id() {
    return manager_id;
}

public void setManager_id(long manager_id) {
    this.manager_id = manager_id;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getGender() {
    return gender;
}
```

```
}
```

```
public void setGender(String gender) {
```

```
    this.gender = gender;
```

```
}
```

```
public String getName() {
```

```
    return name;
```

```
}
```

```
public void setName(String name) {
```

```
    this.name = name;
```

```
}
```

```
public String getMobile() {
```

```
    return mobile;
```

```
}
```

```
public void setMobile(String mobile) {
```

```
    this.mobile = mobile;
```

```
}
```

```
public String getEmail() {
```

```
    return email;
```

```
}
```

```
public void setEmail(String email) {
```

```
    this.email = email;
```

```
}

public IFSCCode getIFSCcode() {
    return IFSCcode;
}

public void setIFSCcode(IFSCCode IFSCcode) {
    this.IFSCcode = IFSCcode;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}
```

IFSCCode.java

```
package bank.model;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;
```

```
@Entity  
@Table(name="BANK")  
public class IFSCCode {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(name = "id")  
    private int id;  
    private String IFSC;  
    private String BranchName;  
    private String Address;  
    @OneToOne(cascade = CascadeType.ALL)  
    private Employee employee;  
    @OneToOne(cascade = CascadeType.ALL)  
    private Customer customer;  
  
    public Customer getCustomer() {  
        return customer;  
    }  
    public void setCustomer(Customer customer) {  
        this.customer = customer;  
    }  
  
    public int getId() {  
        return id;  
    }  
    public void setId(int id) {
```

```
        this.id = id;  
    }  
  
    public Employee getEmployee() {  
        return employee;  
    }  
  
    public void setEmployee(Employee employee) {  
        this.employee = employee;  
    }  
  
    public String getIFSC() {  
        return IFSC;  
    }  
    public void setIFSC(String IFSC) {  
        this.IFSC = IFSC;  
    }  
  
    public String getBranchName() {  
        return BranchName;  
    }  
  
    public void setBranchName(String BranchName) {  
        this.BranchName = BranchName;  
    }  
  
    public String getAddress() {  
        return Address;
```

```
}
```

```
public void setAddress(String Address) {  
    this.Address = Address;  
}  
}
```

Transactions.java

```
package bank.model;  
  
import java.util.Date;  
  
import javax.annotation.Generated;  
  
import javax.persistence.Entity;  
  
import javax.persistence.GeneratedValue;  
  
import javax.persistence.GenerationType;  
  
import javax.persistence.Id;  
  
import javax.persistence.OneToMany;  
  
import javax.persistence.Temporal;  
  
import javax.persistence.TemporalType;
```

```
@Entity
```

```
public class Transactions {  
  
    @Id  
  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private long transaction_no;  
  
    private String comment;  
  
    private String debit;  
  
    private String credit;  
  
    @Temporal(TemporalType.TIMESTAMP)
```

```
private Date dateOfTransaction;  
  
private long account_number;  
  
  
public long getAccount_number() {  
    return account_number;  
}  
  
  
public void setAccount_number(long account_number) {  
    this.account_number = account_number;  
}  
  
  
public String getComment() {  
    return comment;  
}  
  
  
public void setComment(String comment) {  
    this.comment = comment;  
}  
  
  
public long getTransaction_no() {  
    return transaction_no;  
}  
  
  
public void setTransaction_no(long transaction_no) {  
    this.transaction_no = transaction_no;  
}  
  
  
public String getDebit() {  
    return debit;
```

```
}
```

```
public void setDebit(String debit) {  
    this.debit = debit;  
}  
  
public String getCredit() {  
    return credit;  
}  
  
public void setCredit(String credit) {  
    this.credit = credit;  
}  
  
public Date getDateOfTransaction() {  
    return dateOfTransaction;  
}  
  
public void setDateOfTransaction(Date dateOfTransaction) {  
    this.dateOfTransaction = dateOfTransaction;  
}
```

6. CONNECTION PARAMETERS AND OPERATIONS:

HibernateConnection.java

```
package bank.connect_dao;  
  
import java.util.Properties;  
import bank.model.*;  
import org.hibernate.SessionFactory;
```

```
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.cfg.Environment;
import org.hibernate.service.ServiceRegistry;

public class HibernateConnection {
    private static SessionFactory factory;
    private static final String DIALECT="org.hibernate.dialect.MySQL5Dialect";
    private static final String USERNAME="root";
    private static final String URL="jdbc:mysql://localhost:3306/ignou?useSSL=false";
    private static final String DRIVER="com.mysql.jdbc.Driver";
    private static final String PASSWORD="17121999";
    private static final String HBM2DDL="update";
    private static final String SHOWSQL="true";

    static {
        Configuration config=new Configuration();
        Properties settings = new Properties();

        settings.put(Environment.DIALECT, DIALECT);
        settings.put(Environment.DRIVER, DRIVER);
        settings.put(Environment.URL, URL);
        settings.put(Environment.USER, USERNAME);
        settings.put(Environment.PASS, PASSWORD);
        settings.put(Environment.HBM2DDL_AUTO, HBM2DDL);
        settings.put(Environment.SHOW_SQL, SHOWSQL);
        settings.put(Environment.CURRENT_SESSION_CONTEXT_CLASS, "thread");
        config.addProperties(settings);
```

```
config.addAnnotatedClass(Customer.class);
config.addAnnotatedClass(Manager.class);
config.addAnnotatedClass(Employee.class);
config.addAnnotatedClass(IFSCCode.class);
config.addAnnotatedClass(Tra
```



```
ServiceRegistry sr=new
StandardServiceRegistryBuilder().applySettings(config.getProperties()).build();

factory=config.buildSessionFactory(sr);

}

public static SessionFactory createFactory(){

    return factory;

}

}
```

SendEmail.java

```
package bank.connect_dao;

import java.io.UnsupportedEncodingException;
import java.util.Date;
import java.util.Properties;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
```

```
import javax.mail.internet.MimeMessage;

public class SendEmail {

    //call method to send mail

    public static void sendMail(String host, String port, String sender_mail, String
sender_mail_pass, String reciever_mail, String subject, String message) throws
AddressException, MessagingException, UnsupportedEncodingException {

        Properties properties = new Properties();

        properties.put("mail.smtp.host", host);

        properties.put("mail.smtp.port", port);

        properties.put("mail.smtp.auth", "true");

        properties.put("mail.smtp.starttls.enable", "true");

        Authenticator auth = new Authenticator() {

            @Override

            public PasswordAuthentication getPasswordAuthentication() {

                return new PasswordAuthentication(sender_mail, sender_mail_pass);

            }

        };

        Session session = Session.getInstance(properties, auth);

        Message msg = new MimeMessage(session);

        msg.setFrom(new InternetAddress(sender_mail, "bank_online"));

        InternetAddress[] toAddresses = { new InternetAddress(reciever_mail) };

        msg.setRecipients(Message.RecipientType.TO, toAddresses);

        msg.setSubject(subject);

        msg.setSentDate(new Date());

        msg.setContent(message, "text/html");

        Transport.send(msg);

        System.out.println("Mail Sent to : "+reciever_mail);

    }

}
```

```
}
```

DBOperation.java

```
package bank.connect_dao;
```

```
import java.util.List;
```

```
import bank.model.Customer;
```

```
import bank.model.Employee;
```

```
import bank.model.IFSCCode;
```

```
import bank.model.Manager;
```

```
import org.hibernate.Query;
```

```
import org.hibernate.Session;
```

```
import org.hibernate.Transaction;
```

```
import bank.model.Transactions;
```

```
import org.hibernate.Criteria;
```

```
import org.hibernate.criterion.Restrictions;
```

```
//Contains methods to perform all the operations
```

```
//in the Software.
```

```
//All the methods called in each and every module's jsp or servlet page
```

```
//is defined here.
```

```
//I have create multiple sessions for each transaction with the database.
```

```
public class DBOperation {
```

```
    //Session Creation
```

```
    private Session openSession(){
```

```
        Session session=HibernateConnection.createFactory().openSession();
```

```
        return session;
```

```
}
```

```
//Bank Module
```

```
//-----
```

```
public static boolean updateManager(String query, String manager_id){
```

```
    boolean flag;
```

```
    Session s=new DBOperation().openSession();
```

```
    Transaction t=s.beginTransaction();
```

```
    Query q=s.createQuery(query);
```

```
    int stat=q.executeUpdate();
```

```
    System.out.println("Record Updated for manager : "+manager_id);
```

```
    try{
```

```
        t.commit();
```

```
        flag=true;
```

```
    }catch(Exception e){
```

```
        System.err.print(e);
```

```
        if(t!=null){
```

```
            t.rollback();
```

```
        }
```

```
        flag=false;
```

```
}
```

```
s.close();
```

```
return flag;
```

```
}
```

```
public static List<IFSCCode> returnIfsc(){  
    Session session=new DBOperation().openSession();  
    Query q=session.createQuery("from IFSCcode");  
    List<IFSCCode> list=q.list();  
    session.close();  
    return list;  
}
```

```
public static List<Manager> returnManager(){  
    Session session=new DBOperation().openSession();  
    Query q=session.createQuery("from Manager");  
    List<Manager> list=q.list();  
    session.close();  
    return list;  
}
```

```
public static Manager fetchManagerDetailById(long id){  
    Session session=new DBOperation().openSession();  
    Query<Manager> query=session.createQuery("from Manager where  
manager_id='"+id);  
    Manager m=query.getSingleResult();  
    session.close();  
    return m;  
}
```

```
public static boolean addManager(Manager manager,IFSCCode ifsc){
```

```
Session session=new DBOperation().openSession();

boolean flag;

session.save(manager);

manager.setIFSCCode(ifsc);

session.save(ifsc);

Transaction t=session.beginTransaction();

try{

    t.commit();

    flag=true;

}catch(Exception e){

    e.printStackTrace();

    if(t!=null){

        t.rollback();

    }

    flag=false;

}

session.close();

return flag;

}

//-----
//Manager Module

//-----



public static Manager loginManager(String username, String password) {

    Session s=new DBOperation().openSession();

    Criteria cr=s.createCriteria(Manager.class);
```

```
cr.add(Restrictions.eq("username", username));
cr.add(Restrictions.eq("password", password));
List<Manager> list=cr.list();
Manager m=null;
for(Manager man:list){
    m=man;
}
return m;
}

public static boolean addEmployee(Employee emp,IFSCCode ifsc){
    Session session=new DBOperation().openSession();
    boolean flag;
    emp.setIFSCcode(ifsc);
    session.save(emp);
    Transaction t=session.beginTransaction();
    try{
        t.commit();
        flag=true;
    }
    catch(Exception e){
        e.printStackTrace();
        if(t!=null){
            t.rollback();
        }
        flag=false;
    }
}
```

```
}

session.close();

return flag;

}

public static Employee fetchEmployeeDetailById(long id){

Session session=new DBOperation().openSession();

Query<Employee> query=session.createQuery("from Employee where
employee_id='"+id);

Employee e=query.getSingleResult();

session.close();

return e;

}

public static List<Employee> returnAllEmployeeByIFSC(long id){

Session session=new DBOperation().openSession();

Manager m=fetchManagerDetailById(id);

Query<Employee> query=session.createQuery("from Employee where
IFSCcode_id='"+m.getIFSCcode().getId());

List<Employee> list=query.list();

session.close();

return list;

}

public static List<Employee> returnEmployee(){

Session session=new DBOperation().openSession();

List<Employee> list=session.createQuery("from Employee").list();
```

```
session.close();

return list;

}

public static Employee fetchEmployeeById(long id){

Session session=new DBOperation().openSession();

Employee emp=(Employee)session.createQuery("from Employee where
employee_id="+id).getSingleResult();

session.close();

return emp;

}

public static boolean removeEmployee(long id){

Session session=new DBOperation().openSession();

boolean flag;

Transaction t=session.beginTransaction();

Query<Employee> query=session.createQuery("delete from Employee where
employee_id=:id");

query.setLong("id", id);

int result=query.executeUpdate();

try{

t.commit();

flag=true;

}catch(Exception e){

System.err.print(e);

}
```

```
if(t!=null){  
    t.rollback();  
}  
  
flag=false;  
  
}  
  
  
session.close();  
  
return flag;  
}  
  
  
public static boolean updateEmployee(String query,String emp_id){  
    boolean flag;  
    Session s=new DBOperation().openSession();  
    Transaction t=s.beginTransaction();  
    Query q=s.createQuery(query);  
    int stat=q.executeUpdate();  
    System.out.println("Record Updated for employee : "+emp_id);  
    try{  
        t.commit();  
        flag=true;  
    }catch(Exception e){  
        System.err.print(e);  
        if(t!=null){  
            t.rollback();  
        }  
        flag=false;  
    }  
}
```

```
s.close();

return flag;

}

//-----



//Employee Module

//-----



public static Employee loginEmployee(String username,String password){

Session s=new DBOperation().openSession();

Criteria cr=s.createCriteria(Employee.class);

cr.add(Restrictions.eq("username", username));

cr.add(Restrictions.eq("password", password));

List<Employee> list=cr.list();

Employee e=null;

for(Employee emp:list){

e=emp;

}

return e;

}







public static List<Customer> returnAllCustomerByIFSC(long id){

Session session=new DBOperation().openSession();

Employee e=fetchEmployeeDetailById(id);
```

```
Query<Customer> query=session.createQuery("from Customer where  
IFSCcode_id='"+e.getIFSCcode().getId());
```

```
    List<Customer> list=query.list();  
  
    session.close();  
  
    return list;  
}
```

```
public static boolean addCustomer(Customer c,IFSCCode ifsc){
```

```
    boolean flag;  
  
    Session s=new DBOperation().openSession();  
  
    c.setIFSCcode(ifsc);  
  
    s.save(c);  
  
    Transaction t=s.beginTransaction();
```

```
    try{  
        t.commit();  
        flag=true;  
    }  
  
    catch(Exception e){  
        System.err.print(e);  
  
        if(t!=null){  
            t.rollback();  
        }  
  
        flag=false;  
    }  
  
    s.close();  
  
    return flag;
```

```
}
```

```
public static long returnAccountNumber(){

    Session s=new DBOperation().openSession();

    Customer c=(Customer)s.createQuery("from Customer order by account_number
DESC").setMaxResults(1).uniqueResult();

    long x;

    if(c==null){

        x=0;

    }

    else{

        x=c.getAccount_number();

    }

    s.close();

    return x;

}
```

```
public static Customer returnCustomerByAccountNumber(long acc_num){

    Session s=new DBOperation().openSession();

    Criteria cr=s.createCriteria(Customer.class);

    cr.add(Restrictions.eq("account_number", acc_num));

    Customer c=null;

    List<Customer> list=cr.list();

    for(Customer cus:list){

        c=cus;

    }

    s.close();

}
```

```
return c;  
}  
  
public static boolean removeCustomer(long acc_num) {  
    Session session=new DBOperation().openSession();  
    boolean flag;  
    Transaction t=session.beginTransaction();  
    Query<Customer> query=session.createQuery("delete from Customer where  
account_number=:acc_num");  
    Query<Transactions> query1=session.createQuery("delete from Transactions where  
account_number=:anum");  
    query.setLong("acc_num", acc_num);  
    query1.setLong("anum", acc_num);  
  
    int result=query.executeUpdate();  
    int result1=query1.executeUpdate();  
    try{  
        t.commit();  
        flag=true;  
    }catch(Exception e){  
        System.err.print(e);  
        if(t!=null){  
            t.rollback();  
        }  
        flag=false;  
    }  
    session.close();
```

```
        return flag;  
  
    }  
  
    public static boolean updateBalance(double balance,long acc_num){  
        boolean flag;  
        Session s=new DBOperation().openSession();  
        Transaction t=s.beginTransaction();  
        Query q=s.createQuery("update Customer set balance=:bal where  
account_number=:anum");  
        q.setParameter("bal", balance);  
        q.setParameter("anum", acc_num);  
        int stat=q.executeUpdate();  
        System.out.println("Balance Updated!!!\n"+ stat+" Records updated");  
        try{  
            t.commit();  
            flag=true;  
        }catch(Exception e){  
            System.err.print("Invalid Account Number!!!");  
            if(t!=null){  
                t.rollback();  
            }  
            flag=false;  
        }  
        s.close();  
        return flag;  
    }
```

```
public static boolean updateCustomerDetails(String query, String anum){  
    boolean flag;  
    Session s=new DBOperation().openSession();  
    Transaction tr=s.beginTransaction();  
    Query q=s.createQuery(query);  
    int res=q.executeUpdate();  
    System.out.println("Record Updated Form Account Number : "+anum);  
    try{  
        tr.commit();  
        flag=true;  
    }catch(Exception e){  
        System.err.print(e);  
        if(tr!=null){  
            tr.rollback();  
        }  
        flag=false;  
    }  
  
    s.close();  
    return flag;  
}
```

```
public static boolean freezeUnfreezeAccount(String value, long account_num){  
    boolean flag;  
    Session s=new DBOperation().openSession();  
    Transaction tr=s.beginTransaction();  
    Query q=s.createQuery("update customer set status=:value where account_number=:num");  
    q.setParameter("value", value);  
    q.setParameter("num", account_num);  
    int res=q.executeUpdate();  
    System.out.println("Record Updated Form Account Number : "+account_num);  
    if(res>0){  
        tr.commit();  
        flag=true;  
    }else{  
        flag=false;  
    }  
    s.close();  
    return flag;  
}
```

```
Query q=s.createQuery("update Customer set netBankingAllowed=:value where
account_number=:anum");

if("yes".equals(value)){
    q.setParameter("value", "no");
}

else{
    q.setParameter("value", "yes");
}

q.setParameter("anum", account_num);

int res=q.executeUpdate();

System.out.println("Column NetBankingAllowed Updated for Account Number :
"+account_num);

try{
    tr.commit();
    flag=true;
}catch(Exception e){
    System.err.print(e);
    if(tr!=null){
        tr.rollback();
    }
    flag=false;
}

s.close();

return flag;
}

//-----
//CUSTOMER module
```

//-----

```
public static Customer loginCustomer(String username, String password){  
    Session s=new DBOperation().openSession();  
    Criteria cr=s.createCriteria(Customer.class);  
    cr.add(Restrictions.eq("username", username));  
    cr.add(Restrictions.eq("password", password));  
    List<Customer> list=cr.list();  
    Customer c=null;  
    for(Customer cus:list){  
        c=cus;  
    }  
    return c;  
}
```

```
public static void registerCustomer(String query){  
    Session s=new DBOperation().openSession();  
    Transaction tr=s.beginTransaction();  
    int rs=s.createQuery(query).executeUpdate();  
    tr.commit();  
    s.close();  
}
```

//-----

//TRANSACTION

//-----

```
public static boolean commitTransaction(Transactions t){  
    Session s=new DBOperation().openSession();  
    boolean flag;  
    Transaction tr=s.beginTransaction();  
    s.save(t);  
    try{  
        tr.commit();  
        flag=true;  
    }catch(Exception e){  
        System.err.print(e);  
        if(tr!=null){  
            tr.rollback();  
        }  
        flag=false;  
    }  
    s.close();  
    return flag;  
}
```

```
public static List<Transactions> fetchTransactions(long account_number){  
    Session s=new DBOperation().openSession();  
    Query<Transactions> query=s.createQuery("from Transactions where  
    account_number='"+account_number+"');  
    List<Transactions> list=query.list();  
    s.close();  
    return list;
```

}

//-----

}

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">

    <context-param>
        <param-name>host</param-name>
        <param-value>smtp.gmail.com</param-value>
    </context-param>

    <context-param>
        <param-name>port</param-name>
        <param-value>587</param-value>
    </context-param>

    <context-param>
        <param-name>sender_email</param-name>
        <param-value>kb1037583@gmail.com</param-value>
    </context-param>

    <context-param>
        <param-name>password</param-name>
        <param-value>@admin17121999</param-value>
    </context-param>
```

```
<context-param>
    <param-name>username</param-name>
    <param-value>admin</param-value>
</context-param>
<context-param>
    <param-name>pwd</param-name>
    <param-value>@admin12345</param-value>
</context-param>
<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
</web-app>
```

TESTING

INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified?.

Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Validation : Are we doing the right job?

Verification : Are we doing the job right?

Software testing should not be confused with debugging. Debugging is the process of analyzing and localizing bugs when software does not behave as expected. Although the identification of some bugs will be obvious from playing with the software, a methodical approach to software testing is a much more thorough means for identifying bugs. Debugging is therefore an activity which supports testing, but cannot replace testing.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis

looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted System atically. Testing begins at the module level and work towards the integration of entire computers based System . Nothing is complete without testing, as it vital success of the System testing objectives, there are several rules that can serve as testing objectives. They are

- Testing is a process of executing a program with the intend of finding an error.
- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncovered errors in the software also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

1. TEST PLAN:

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

- ◆ Unit Testing
- ◆ Integration Testing
- ◆ System Testing

- ◆ Data Validation Testing
- ◆ Output Testing

1.1. UNIT TESTING:

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. Unit testing, also known as component testing , refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working

as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

1.2. INTEGRATION TESTING:

Integration testing is Systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop.

After unit testing in Sell-Soft System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

1.3. SYSTEM TESTING:

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.

System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

1.4. DATA VALIDATION TESTING:

This is the final step in testing. In this the entire System was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

1.5. OUTPUT TESTING:

The System considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective System ; user at the time of developing and making changes whenever required.

This done with respect to the following points:

- ◆ Input Screen Designs,
- ◆ Output Screen Designs,
- ◆ Online message to guide the user and the like.

The above testing is done taking various kinds of test data.

Preparation of test data plays a vital role in the System testing. After preparing the test data, the System under study is tested using that test data. While testing the System by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

2. IMPLEMENTATION PROCEDURES:

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the System . In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that

- The active user must be aware of the benefits of using the new System .
- Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the System , the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

2.1. USER TRAINING:

User training is designed to prepare the user for testing and converting the System . To achieve the objective and benefits expected from computer based System , it is essential for the people who will be involved to be confident of their role in the new System . As System becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

2.2. TRAINING ON THE APPLICATION SOFTWARE:

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new System such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the System or part of the System while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

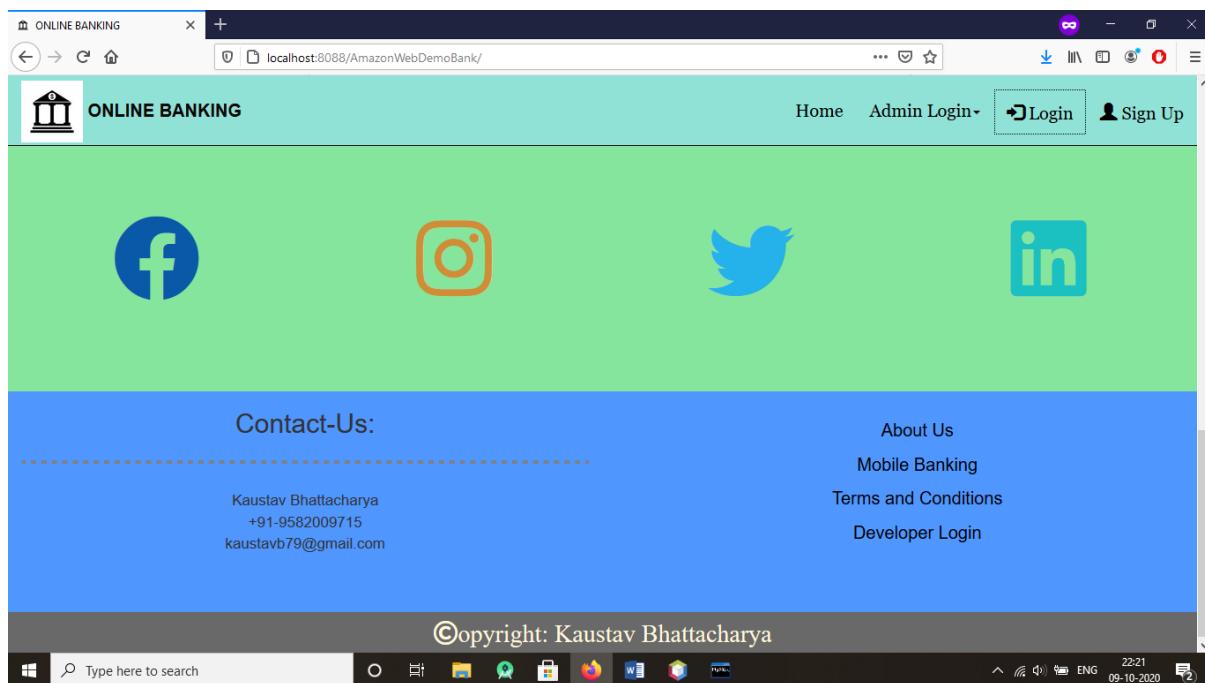
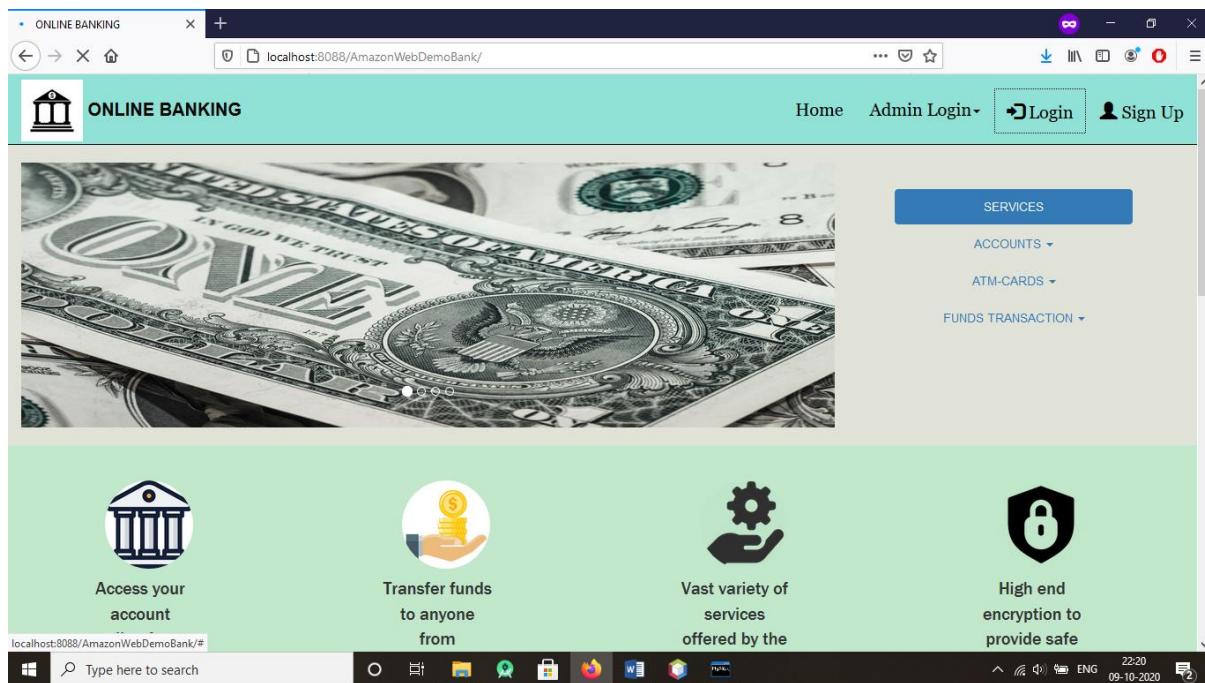
2.3. OPERATIONAL DOCUMENT:

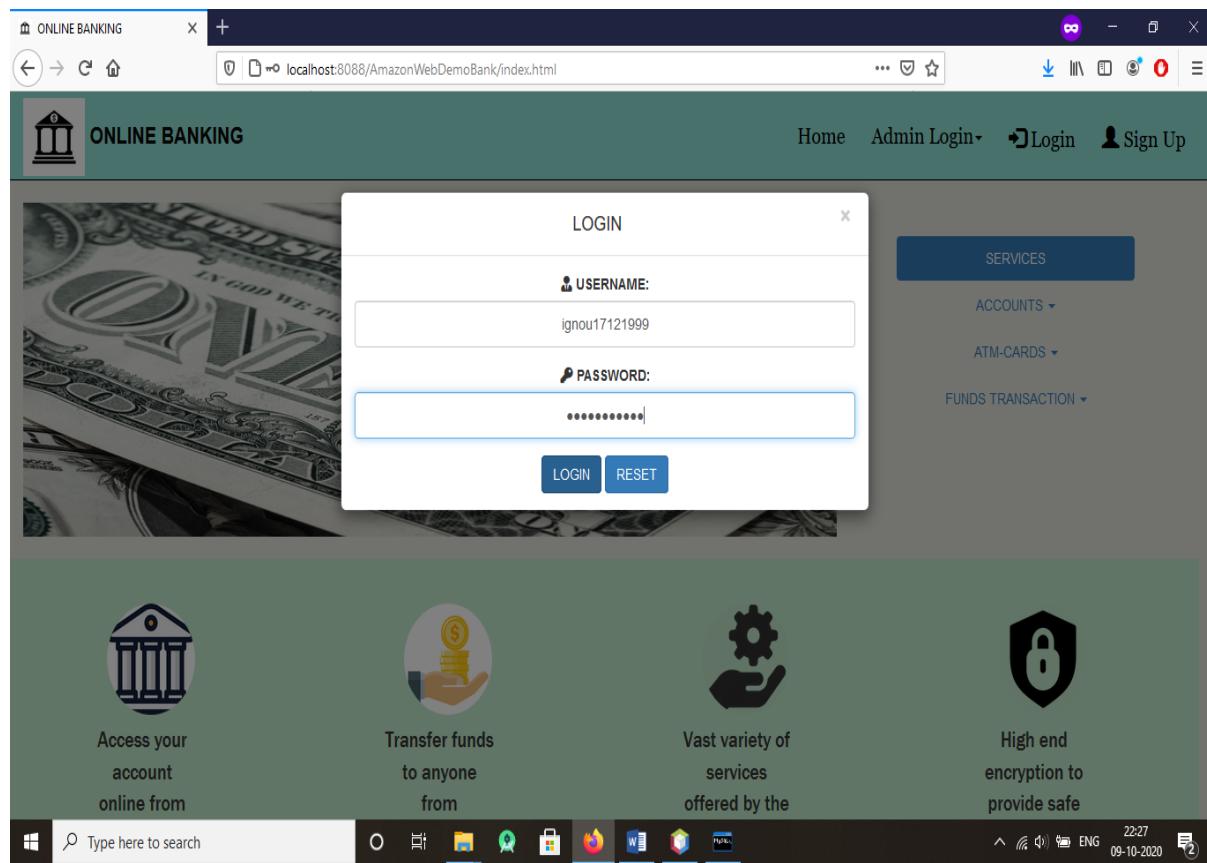
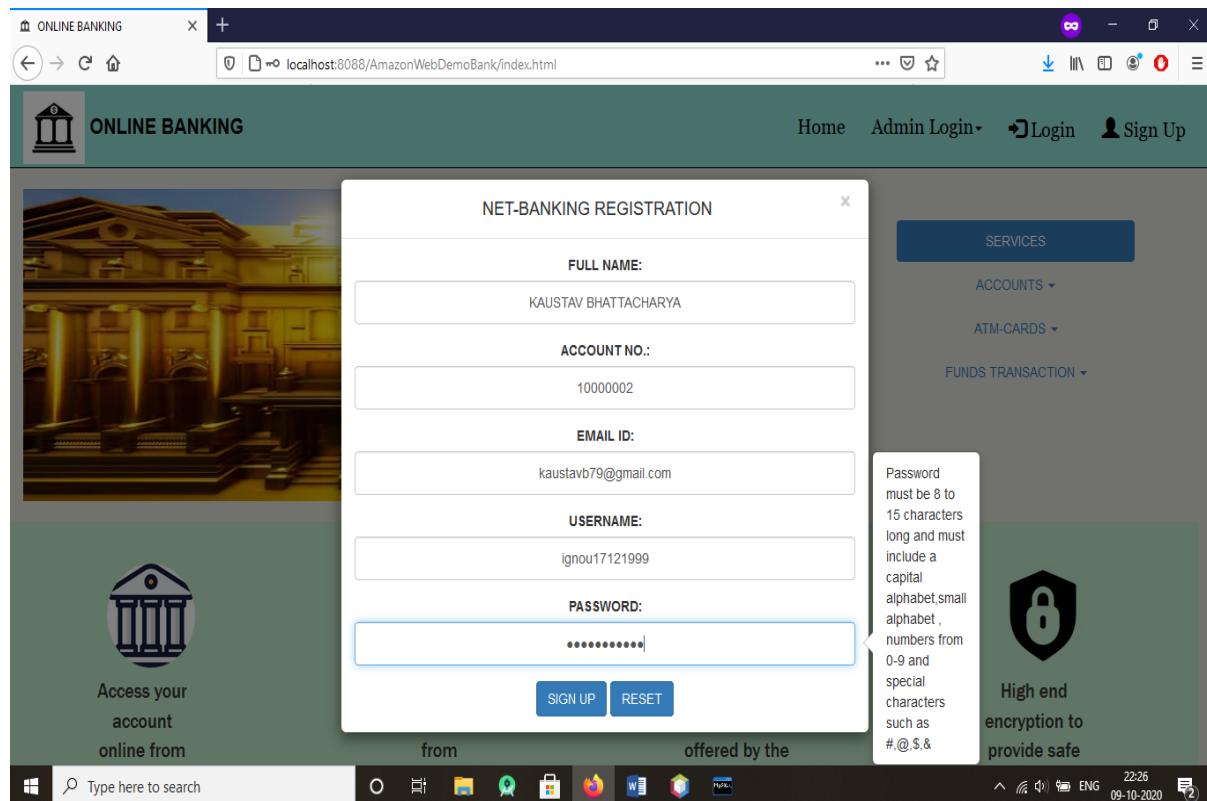
Once the implementation plan is decided, it is essential that the user of the System is made familiar and comfortable with the environment. Education involves right atmosphere and motivating the user. A documentation providing the whole operations of the System is being developed in such a way that the user can work with it in well consistent way. The System is developed user friendly so that the user can work the System from the tips given in the application itself. Useful tip and guidance is given inside the application itself to help the user. Users have to be made aware that what can be achieved with the new System and how it increases the performance of the System . The user of the System should be given a general idea of the System before he uses the System .

2.4. SYSTEM MAINTENANCE:

Maintenance is the enigma of System development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a System is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for System maintenance is for it to make adaptable to the changes in the System environment. Software maintenance is of course, far more than "Finding Mistakes". Maintenance may be defined by describing four activities that are undertaken after a program is released for use.

INPUT AND OUTPUT SCREENS





The screenshot shows the customer dashboard with a teal header bar. The title 'ONLINE BANKING SYSTEM' and a welcome message 'Welcome Mr. KAUSTAV BHATTACHARYA' are displayed. Below the header, there are four navigation links: 'Account Summary', 'Fund Transfer', 'Account Statement', and 'Profile'. A 'Logout' button is located on the right side of the header. The main content area displays a table with account details:

ACCOUNT NUMBER	TYPE OF ACCOUNT	EMAIL ID	MOBILE NO.	BALANCE
10000002	Savings	kaustavb79@gmail.com	9582009715	Rs. 10000.0

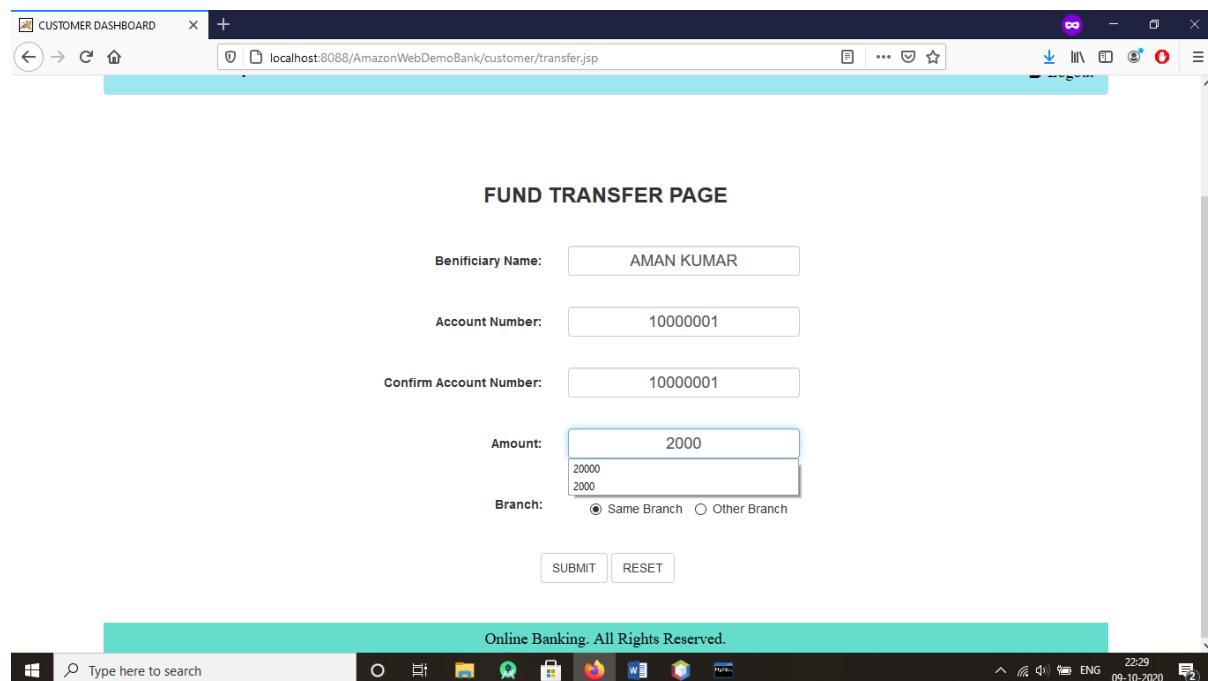
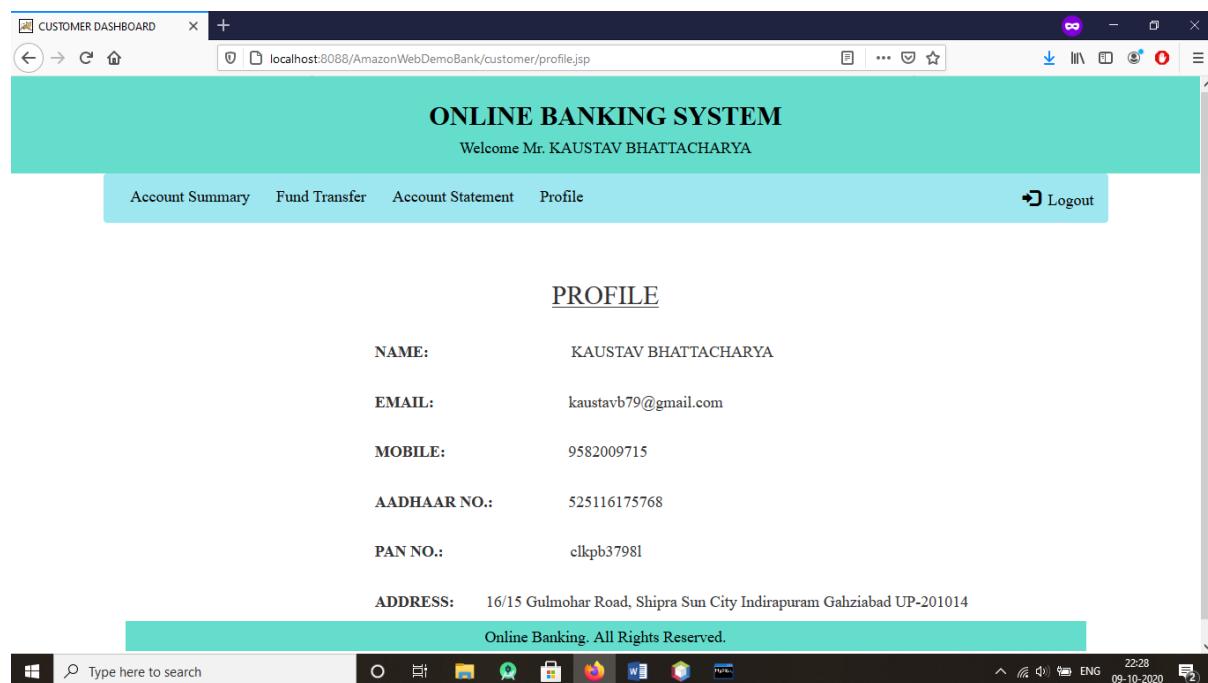
At the bottom of the window, there is a status bar with the text 'Online Banking. All Rights Reserved.' and system information including the date and time.

The screenshot shows the account statement page with a teal header bar. The title 'ONLINE BANKING SYSTEM' and a welcome message 'Welcome Mr. KAUSTAV BHATTACHARYA' are displayed. Below the header, there are four navigation links: 'Account Summary', 'Fund Transfer', 'Account Statement', and 'Profile'. A 'Logout' button is located on the right side of the header. The main content area displays a section titled 'ACCOUNT STATEMENT' followed by account details and a transaction history table:

Name: KAUSTAV BHATTACHARYA
Account Number: 10000002
Balance: Rs. 10000.0

TRANSACTION NO.	COMMENT	DEBIT	CREDIT	TRANSACTION DATE
13	Deposit	Rs. 10000		Fri, 09 Oct 2020 22:25:32 IST

At the bottom of the window, there is a status bar with the text 'Online Banking. All Rights Reserved.' and system information including the date and time.



Welcome Mr. KAUSTAV BHATTACHARYA

Account Summary Fund Transfer Account Statement Profile Logout

ACCOUNT STATEMENT

Name: KAUSTAV BHATTACHARYA

Account Number: 10000002

Balance: Rs. 8000.0

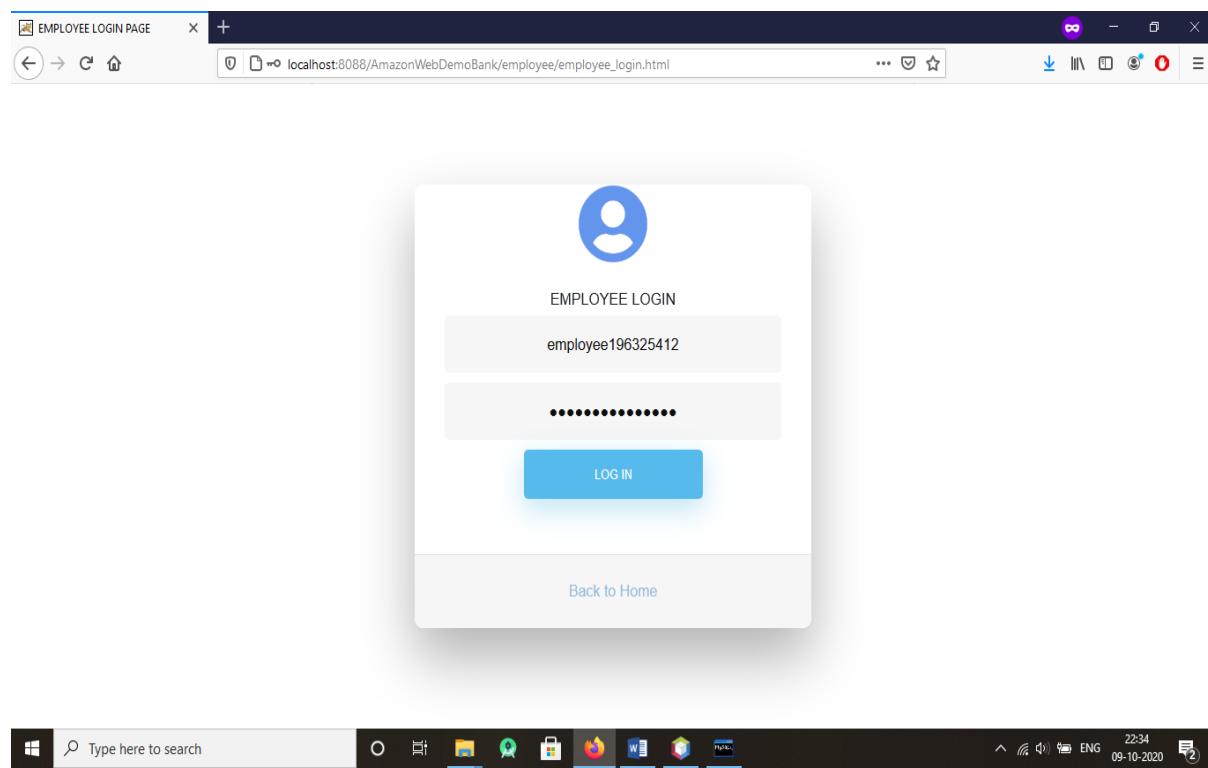
TRANSACTION NO.	COMMENT	DEBIT	CREDIT	TRANSACTION DATE
14	Transfer to 10000001 Beneficiary Name: AMAN KUMAR	Rs. 2000.0		Fri, 09 Oct 2020 22:29:34 IST
13	Deposit	Rs. 10000		Fri, 09 Oct 2020 22:25:32 IST

Online Banking. All Rights Reserved.

(Transaction No.14)

Redirecting.....

(Logout page)

A screenshot of a web browser window titled "BANK EMPLOYEE DASHBOARD". The URL is "localhost:8088/AmazonWebDemoBank/employee/emplogin.do". The dashboard has a header with a user icon and the text "EMPLOYEE DASHBOARD" on the left and a "Log Out" button on the right. On the left is a sidebar with a yellow background containing a list of menu items: "Employee Details", "Add Customer", "Remove Customer", "List All Customer's", "Update Customer Details", "Customer Account Statement", "Freeze/Un-freeze Customer Account", and "Deposit Money". The main content area shows a table with one row of data:

ID	NAME	MOBILE	EMAIL
1	EMPLOYEE1 TEST	7458623145	outlook0123bca@gmail.com

BANK EMPLOYEE DASHBOARD

EMPLOYEE DASHBOARD

- Employee Details
- Add Customer
- Remove Customer
- List All Customer's
- Update Customer Details
- Customer Account Statement
- Freeze/Un-freeze Customer Account
- Deposit Money

New Customer

Customer Name: Vaibhav Kumar

Customer Email: vaibhavk@gmail.com

Customer Mobile: 7879536412

Gender: Male Female

Customer Address: 123-R , Nriyta Apartments, Niti Khand, Indirapuram , Ghaziabad, UP-201010

Type here to search

22:36 09-10-2020

BANK EMPLOYEE DASHBOARD

EMPLOYEE DASHBOARD

- Employee Details
- Add Customer
- Remove Customer
- List All Customer's
- Update Customer Details
- Customer Account Statement
- Freeze/Un-freeze Customer Account
- Deposit Money

CUSTOMER ADDRESS.

123-R , Nriyta Apartments, Niti Khand, Indirapuram , Ghaziabad, UP-201010

Customer Aadhaar No: 758582424242

Customer Pan Card No: GHIU5589P|

Type Of Account: SAVINGS

ADD CUSTOMER RESET FORM

Type here to search

22:36 09-10-2020

The screenshot shows the 'EMPLOYEE DASHBOARD' window. On the left, a sidebar contains links: 'Employee Details', 'Add Customer', 'Remove Customer', 'List All Customer's' (which is selected and highlighted in yellow), 'Update Customer Details', 'Customer Account Statement', 'Freeze/Un-freeze Customer Account', and 'Deposit Money'. The main area displays a table of customer information:

ACCOUNT NUMBER	NAME	MOBILE	EMAIL	BALANCE	AADHAAR NO.
10000000	KAUSTAV BHATTACHARYA	9582009715	kaustavb79@gmail.com	41890.0	525116175768
10000001	AMAN KUMAR	7856896325	outlook0123bca@gmail.com	46110.0	785694125896123
10000002	KAUSTAV BHATTACHARYA	9582009715	kaustavb79@gmail.com	8000.0	525116175768
10000003	Vaibhav Kumar	7879536412	vaibhavk@gmail.com	0.0	758582424242

The screenshot shows the 'EMPLOYEE DASHBOARD' window. The 'List All Customer's' link is still selected in the sidebar. A dropdown menu titled 'Select Customer:' is open, listing four customer entries:

- 10000001----AMAN KUMAR
- 10000000----KAUSTAV BHATTACHARYA
- 10000002----KAUSTAV BHATTACHARYA
- 10000003----Vaibhav Kumar

Below the dropdown, the customer details are displayed:

Mobile: 7856896325
Aadhaar No.: 785694125896123
Balance: 46110.0

A blue 'Remove Customer' button is visible at the bottom right of the dropdown area.

EMPLOYEE DASHBOARD

Select Customer:
10000003-----Vaibhav Kumar

Name: Vaibhav Kumar
Email: vaibhavk@gmail.com
Mobile: 7879536412
Aadhaar No.: 758582424242
Balance: 0.0

[Remove Customer](#)

Employee Details

- [Add Customer](#)
- [Remove Customer](#)
- [List All Customer's](#)
- [Update Customer Details](#)
- [Customer Account Statement](#)
- [Freeze/Un-freeze Customer Account](#)
- [Deposit Money](#)

EMPLOYEE DASHBOARD

ACCOUNT NUMBER	NAME	MOBILE	EMAIL	BALANCE	AADHAAR NO.
10000000	KAUSTAV BHATTACHARYA	9582009715	kaustavb79@gmail.com	41890.0	525116175768
10000001	AMAN KUMAR	7856896325	outlook0123bca@gmail.com	46110.0	785694125896123
10000002	KAUSTAV BHATTACHARYA	9582009715	kaustavb79@gmail.com	8000.0	525116175768

Employee Details

- [Add Customer](#)
- [Remove Customer](#)
- [List All Customer's](#)
- [Update Customer Details](#)
- [Customer Account Statement](#)
- [Freeze/Un-freeze Customer Account](#)
- [Deposit Money](#)

Customer 'Vaibhav Kumar' Removed Successfully

BANK EMPLOYEE DASHBOARD

localhost:8088/AmazonWebDemoBank/employee/emplogin.do

EMPLOYEE DASHBOARD

Employee Details

- Add Customer
- Remove Customer
- List All Customer's
- Update Customer Details
- Customer Account Statement
- Freeze/Un-freeze Customer Account
- Deposit Money

Select Customer Account Number:

Field(s) to be Updated:

Email Id
Mobile
Aadhaar
Pan

Next

Type here to search

22:38 09-10-2020

BANK EMPLOYEE DASHBOARD

localhost:8088/AmazonWebDemoBank/employee/emplogin.do

EMPLOYEE DASHBOARD

Employee Details

- Add Customer
- Remove Customer
- List All Customer's
- Update Customer Details
- Customer Account Statement
- Freeze/Un-freeze Customer Account
- Deposit Money

Account Number:

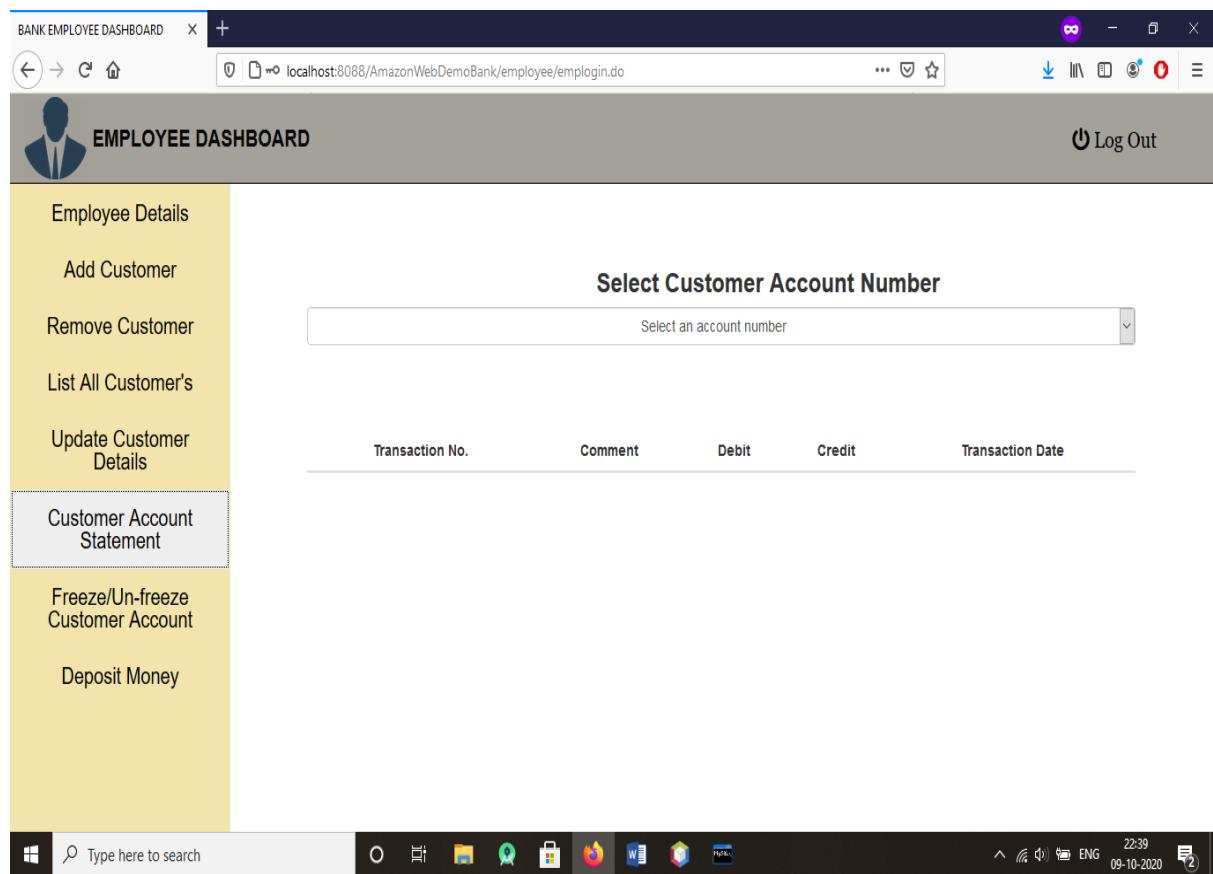
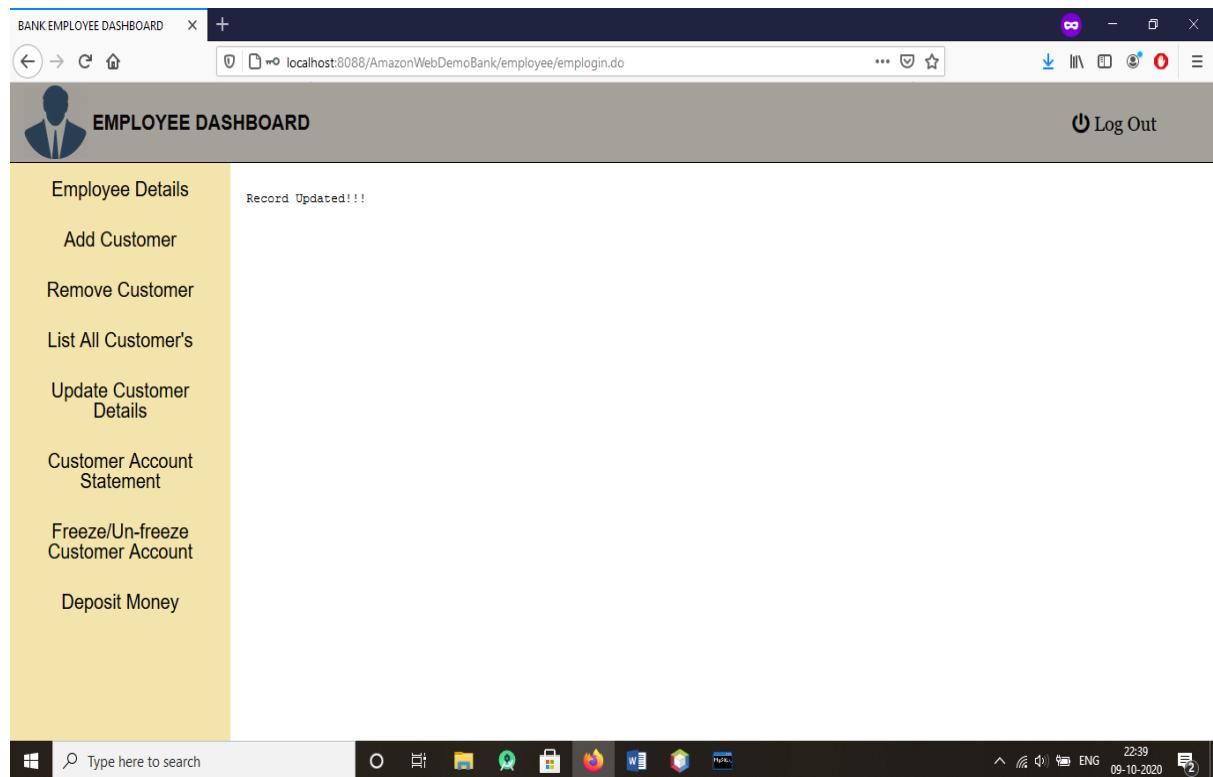
MOBILE:

AADHAAR:

UPDATE

Type here to search

22:39 09-10-2020



EMPLOYEE DASHBOARD

Select Customer Account Number
10000002 / KAUSTAV BHATTACHARYA

Transaction No.	Comment	Debit	Credit	Transaction Date
14	Transfer to 10000001 Beneficiary Name: AMAN KUMAR	Rs. 2000.0		Fri, 09 Oct 2020 22:29:34 IST
13	Deposit	Rs. 10000		Fri, 09 Oct 2020 22:25:32 IST

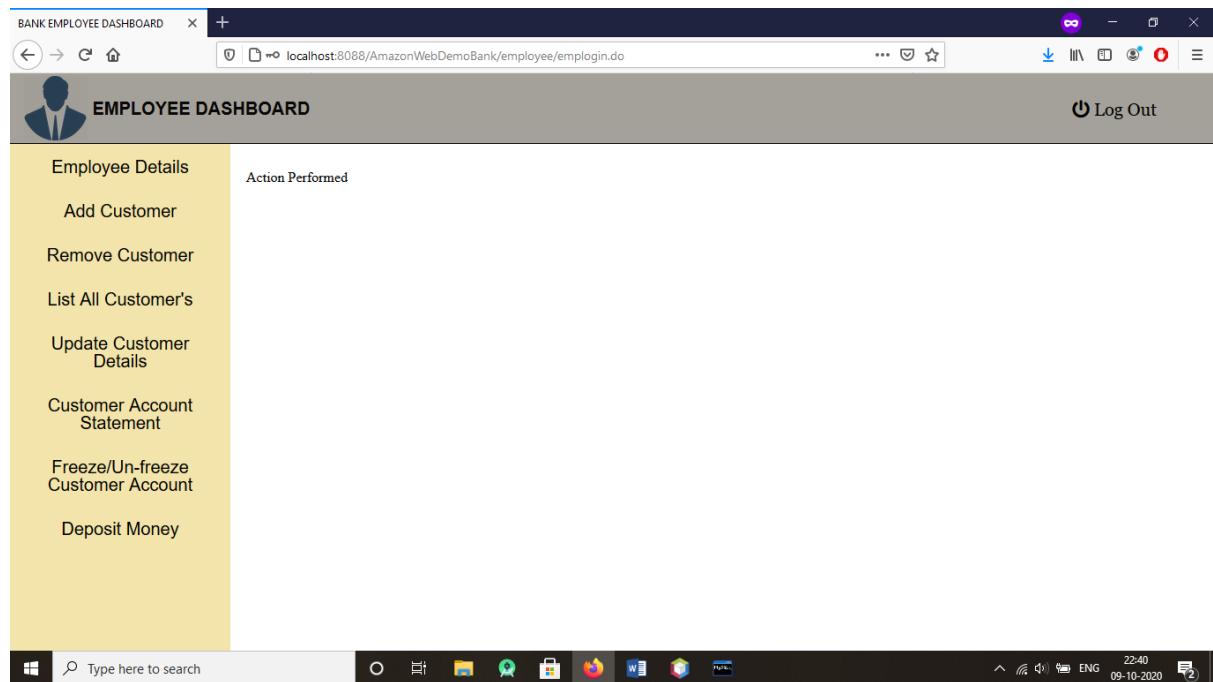
Type here to search

(Records of all past transactions for a/c – 100000002)

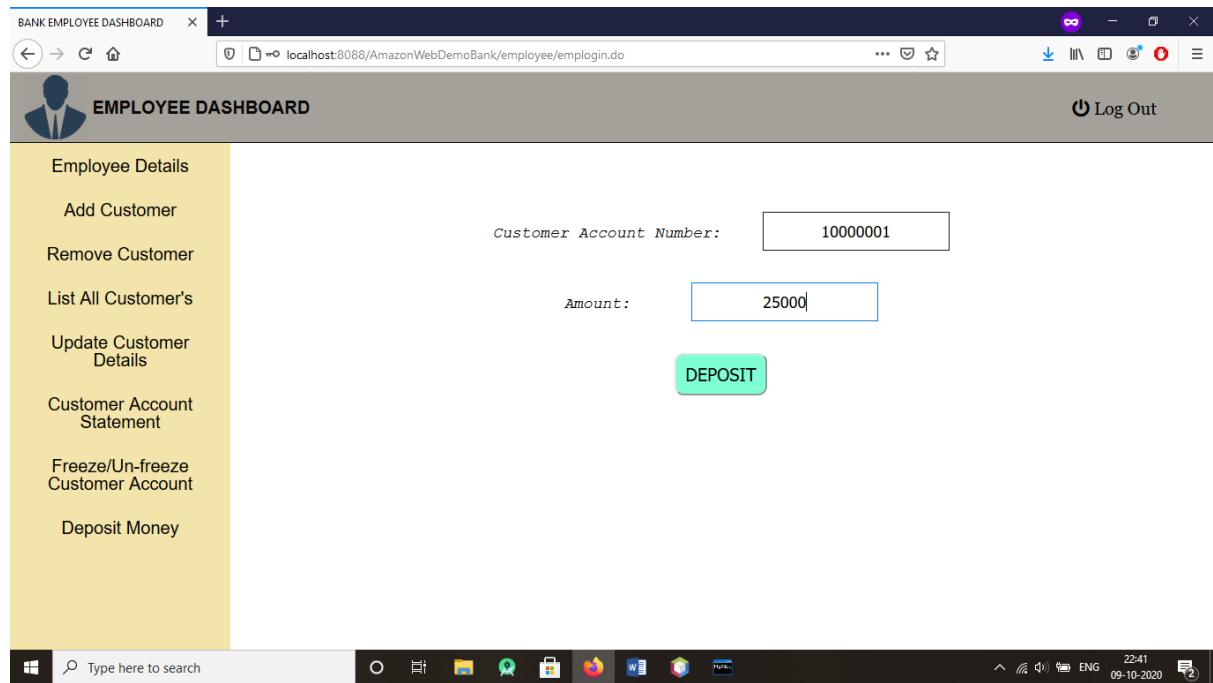
EMPLOYEE DASHBOARD

ACCOUNT NUMBER	NAME	EMAIL	BALANCE	NET-BANKING ALLOWED	ACTION
10000000	KAUSTAV BHATTACHARYA	kaustavb79@gmail.com	41890.0	NO	Enable/Disable
10000001	AMAN KUMAR	outlook0123bca@gmail.com	46110.0	YES	Enable/Disable
10000002	KAUSTAV BHATTACHARYA	kaustavb79@gmail.com	8000.0	YES	Enable/Disable

Type here to search



Transaction Services Disabled for account number – 10000002



EMPLOYEE DASHBOARD

Employee Details

- Add Customer
- Remove Customer
- List All Customer's
- Update Customer Details
- Customer Account Statement
- Freeze/Un-freeze Customer Account
- Deposit Money

Rs. 25000 deposited to account number: 10000001 successfully....

Log Out

(Amount- Rs. 25000 deposited to account 100000001)

ONLINE BANKING

Home Admin Login ➔ Login Sign Up

Manager

ACCOUNTS ATM-CARDS FUNDS TRANSACTION

Access your account

Transfer funds to anyone from

Vast variety of services offered by the

High end encryption to provide safe

localhost:8088/AmazonWebDemoBank/manager/manager_login.html

The screenshot shows a browser window titled "MANAGER LOGIN PAGE". The URL in the address bar is "localhost:8088/AmazonWebDemoBank/manager/manager_login.html". The page displays a "MANAGER LOGIN" form with a blue user icon at the top. Below it, a text input field contains "manager2009715". A link "View Saved Logins" is visible below the input field, followed by a series of black dots. A large blue "LOG IN" button is centered at the bottom of the form. At the very bottom, there is a link "Back to Home".



The screenshot shows a browser window titled "BANK MANAGER DASHBOARD". The URL in the address bar is "localhost:8088/AmazonWebDemoBank/manager/manager_login.do". The page has a header with a user icon and the text "MANAGER DASHBOARD" on the left, and a "Log Out" link on the right. A sidebar on the left contains links: "Manager Details", "Add Employee", "Remove Employee", "Update Employee", and "List All Employee". The main content area displays a table with one row of data:

ID	NAME	MOBILE	EMAIL
1	MANAGER DEMO	9582009715	outlook0123bca@gmail.com

At the bottom of the screen is a Windows taskbar with various pinned application icons and system status icons on the right.

BANK MANAGER DASHBOARD X +

localhost:8088/AmazonWebDemoBank/manager/manager_login.do

MANAGER DASHBOARD

Log Out

Manager Details

Add Employee

Remove Employee

Update Employee

List All Employee

New Employee

Employee Name: Enter Employee name

Employee Email: Enter Employee email

Employee Mobile: Enter Employee mobile

Gender: Male Female

Employee Aadhaar No: Enter Employee Aadhaar

Employee Salary: Enter Employee Salary in INR

Type here to search

22:43 09-10-2020

BANK MANAGER DASHBOARD X +

localhost:8088/AmazonWebDemoBank/manager/manager_login.do

MANAGER DASHBOARD

Log Out

Manager Details

Add Employee

Remove Employee

Update Employee

List All Employee

Employee Aadhaar No: Enter Employee Aadhaar

Employee Salary: Enter Employee Salary in INR

Employee Address:

ADD EMPLOYEE RESET FORM

Type here to search

22:44 09-10-2020

The screenshot shows the 'BANK MANAGER DASHBOARD' window. On the left, a sidebar titled 'Manager Details' lists five options: 'Add Employee', 'Remove Employee', 'Update Employee', and 'List All Employee'. The main area is titled 'Select Employee:' and displays the following information for 'EMPLOYEE2 TEST':
Name: EMPLOYEE2 TEST
Email: outlook0123bca@gmail.com
Mobile: 8588250102
Aadhaar No.: 236984758211
Salary: 45000.0

[Remove Employee](#)

The screenshot shows the 'BANK MANAGER DASHBOARD' window. The sidebar remains the same. The main area now displays the message 'Employee Removed!!!'.

(Employee with id -2 Removed)

The screenshot shows a Windows desktop environment with a web browser window titled "BANK MANAGER DASHBOARD". The URL in the address bar is "localhost:8088/AmazonWebDemoBank/manager/manager_login.do". The main content area is titled "MANAGER DASHBOARD" and contains a sidebar with the following menu items: Manager Details, Add Employee, Remove Employee, Update Employee, and List All Employee. The "List All Employee" option is currently selected and highlighted with a gray border. To the right of the sidebar, there is a table with the following data:

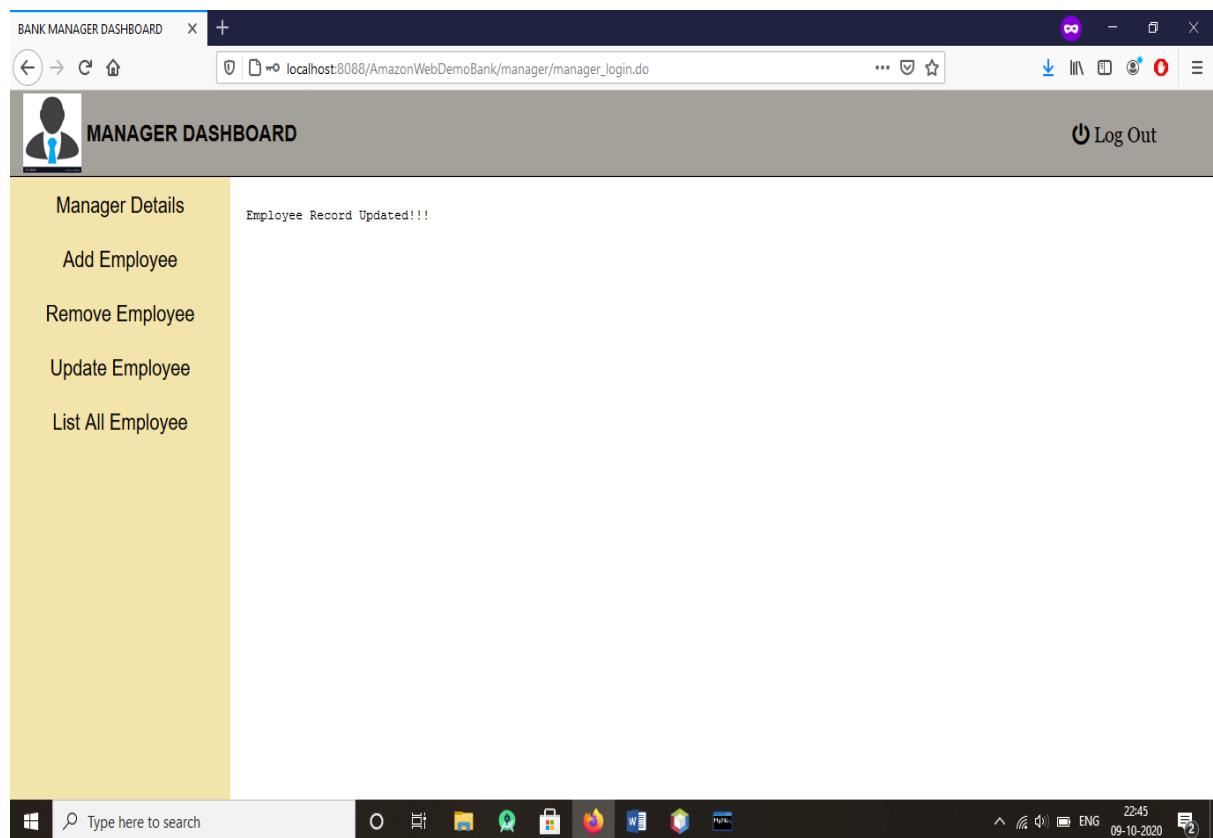
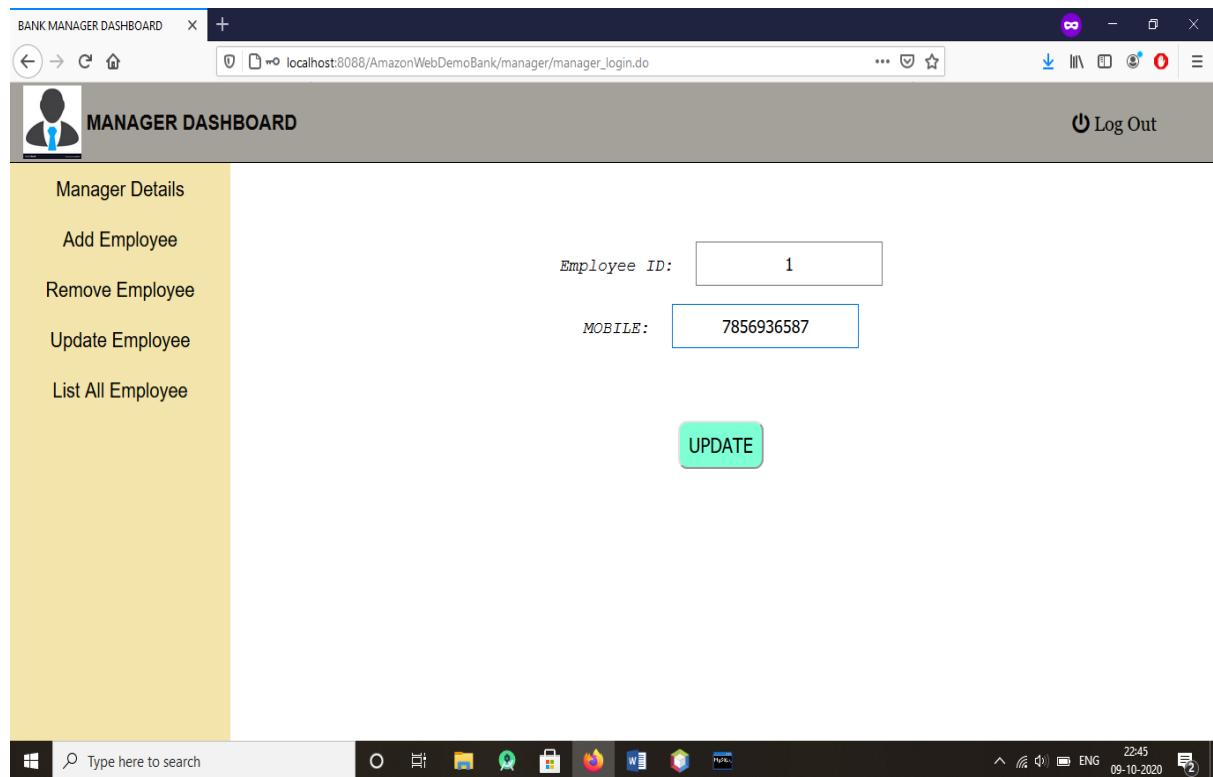
EMPLOYEE ID	NAME	MOBILE	EMAIL	SALARY	AADHAAR NO.
1	EMPLOYEE1 TEST	7458623145	outlook0123bca@gmail.com	50000.0	745896325412

The status bar at the bottom of the screen shows system information: 22:44, ENG, 09-10-2020, and a battery icon.

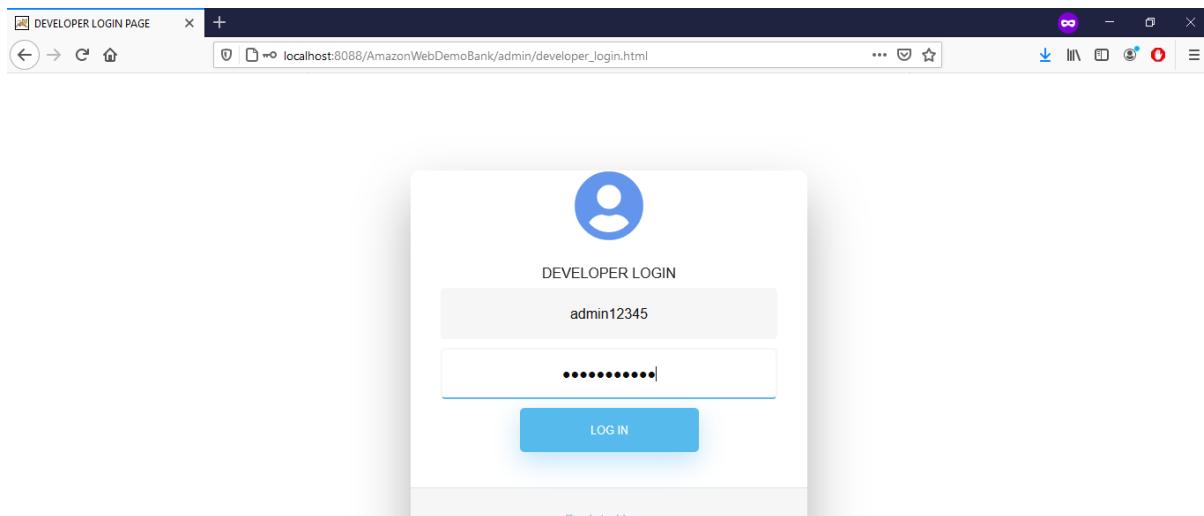
The screenshot shows a Windows desktop environment with a web browser window titled "BANK MANAGER DASHBOARD". The URL in the address bar is "localhost:8088/AmazonWebDemoBank/manager/manager_login.do". The main content area is titled "MANAGER DASHBOARD" and contains a sidebar with the following menu items: Manager Details, Add Employee, Remove Employee, Update Employee, and List All Employee. The "Update Employee" option is currently selected and highlighted with a gray border. The main panel displays two input fields:

- A dropdown menu labeled "Select Employee ID:" containing the option "1-----EMPLOYEE1 TEST".
- A scrollable list box labeled "Field(s) to be Updated:" containing the options "Email Id", "Mobile", and "Address".

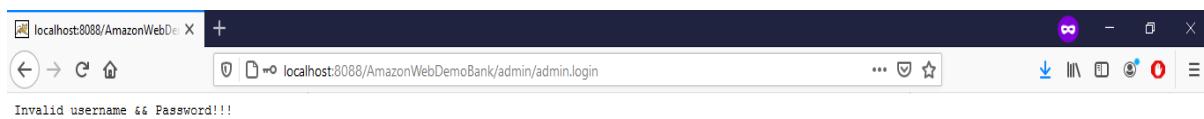
A "Next" button is located at the bottom of the main panel. The status bar at the bottom of the screen shows system information: 22:45, ENG, 09-10-2020, and a battery icon.



(Mobile no. of employee with id- 1 updated)



Incorrect Details...



ADMINISTRATOR DASHBOARD

- Add manager
- Update Manager
- View All Managers

Add New Manager

Manager Name:
IGNOU NOIDA

Manager Email:
ignou@gmail.com

Manager Mobile:
0120456789

Gender: Male Female

Manager Address:
C-53, Institutional Area, Rajat Vihar, Industrial Area, Sector 62, Noida, Uttar Pradesh 201305

ADD MANAGER **RESET FORM**

ADMINISTRATOR DASHBOARD

- Add manager
- Update Manager
- View All Managers

IFSC Code:
GBIK025600

Branch:
NOIDA

Branch Address:
C- 25, BHA Millennium Rd, C Block, Phase 2, Industrial Area, Sector 62, Noida, Uttar Pradesh 201305

ADD MANAGER **RESET FORM**

The screenshot shows a web-based administrator dashboard titled "BANK ADMINISTRATOR DASHBOARD". On the left, there is a sidebar with three options: "Add manager", "Update Manager", and "View All Managers". The main area displays a table of managers with the following data:

	MANAGER ID	NAME	MOBILE	EMAIL	IFSC CODE	BRANCH
	1	MANAGER DEMO	9582009715	outlook0123bca@gmail.com	OBJ0000245	Indirapuram
	3	IGNOU NOIDA	7879536412	ignou@gmail.com	GBIK025600	NOIDA

The browser address bar shows "localhost:8088/AmazonWebDemoBank/admin/main_page.jsp". The system status bar at the bottom indicates "Type here to search", "Windows", "22:56", "ENG", "09-10-2020", and a battery icon.

(manager – ‘ Ignou Noida ‘ Added successfully)

IMPLEMENTATION OF SECURITY

As computers and other digital devices have become essential to business and commerce, they have also increasingly become a target for attacks. In order for a company or an individual to use a computing device with confidence, they must first be assured that the device is not compromised in any way and that all communications will be secure. In this chapter, we will review the fundamental concepts of information System s security and discuss some of the measures that can be taken to mitigate security threats. We will begin with an overview focusing on how organizations can stay secure. Several different measures that a company can take to improve security will be discussed. We will then follow up by reviewing security precautions that individuals can take in order to secure their personal computing environment.

The Information Security Triad: Confidentiality, Integrity, Availability (CIA)

- **Confidentiality**

When protecting information, we want to be able to restrict access to those who are allowed to see it; everyone else should be disallowed from learning anything about its contents. This is the essence of confidentiality. For example, federal law requires that universities restrict access to private student information. The university must be sure that only those who are authorized have access to view the grade records.

- **Integrity**

Integrity is the assurance that the information being accessed has not been altered and truly represents what is intended. Just as a person with integrity means what he or she says and can be trusted to consistently represent the truth, information integrity means information truly represents its intended meaning. Information can lose its integrity through malicious intent, such as when someone who is not authorized makes a change to intentionally misrepresent something. An example of this would be when a hacker is hired to go into the university's System and change a grade.

Integrity can also be lost unintentionally, such as when a computer power surge corrupts a file or someone authorized to make a change accidentally deletes a file or enters incorrect information.

- **Availability**

Information availability is the third part of the CIA triad. Availability means that information can be accessed and modified by anyone authorized to do so in an appropriate timeframe. Depending on the type of information, appropriate timeframe can mean different things. For example, a stock trader needs information to be available immediately, while a sales person may be happy to get sales numbers for the day in a report the next morning.

Companies such as Amazon.com will require their servers to be available twenty-four hours a day, seven days a week. Other companies may not suffer if their web servers are down for a few minutes once in a while.

Tools for Information Security

In order to ensure the confidentiality, integrity, and availability of information, organizations can choose from a variety of tools. Each of these tools can be utilized as part of an overall information-security policy, which will be discussed in the next section.

- **Authentication**

The most common way to identify someone is through their physical appearance, but how do we identify someone sitting behind a computer screen or at the ATM? Tools for authentication are used to ensure that the person accessing the information is, indeed, who they present themselves to be.

Authentication can be accomplished by identifying someone through one or more of three factors: something they know, something they have, or something they are. For example, the most common form of authentication today is the user ID and password. In this case, the authentication is done by confirming something that the user knows (their ID and password). But this form of authentication is easy to compromise (see sidebar) and stronger forms of authentication are sometimes needed. Identifying someone only by something they have, such as a key or a card, can also be problematic. When that identifying token is lost or stolen, the identity can be easily stolen. The final factor, something you are, is much harder to compromise. This factor identifies a user through the use of a physical characteristic, such as an eye-

scan or fingerprint. Identifying someone through their physical characteristics is called biometrics.

A more secure way to authenticate a user is to do multi-factor authentication. By combining two or more of the factors listed above, it becomes much more difficult for someone to misrepresent themselves.

- **Access Control**

Once a user has been authenticated, the next step is to ensure that they can only access the information resources that are appropriate. This is done through the use of access control. Access control determines which users are authorized to read, modify, add, and/or delete information. Several different access control models exist. Here we will discuss two: the access control list (ACL) and role-based access control (RBAC).

For each information resource that an organization wishes to manage, a list of users who have the ability to take specific actions can be created. This is an access control list, or ACL. For each user, specific capabilities are assigned, such as read, write, delete, or add. Only users with those capabilities are allowed to perform those functions. If a user is not on the list, they have no ability to even know that the information resource exists.

ACLs are simple to understand and maintain. However, they have several drawbacks. The primary drawback is that each information resource is managed separately, so if a security administrator wanted to add or remove a user to a large set of information resources, it would be quite difficult. And as the number of users and resources increase, ACLs become harder to maintain. This has led to an improved method of access control, called role-based access control, or RBAC. With RBAC, instead of giving specific users access rights to an information resource, users are assigned to roles and then those roles are assigned the access. This allows the administrators to manage users and roles separately, simplifying administration and, by extension, improving security.

- **Encryption**

Many times, an organization needs to transmit information over the Internet or transfer it on external media such as a CD or flash drive. In these cases, even with proper authentication and access control, it is possible for an unauthorized person to get access to the data. Encryption is a process of encoding data upon its transmission or storage so that only authorized individuals can read it. This encoding is accomplished by a computer program, which encodes the plain text that needs to be transmitted; then the recipient receives the cipher text and decodes it (decryption). In order for this to work, the sender and receiver need to agree on the method of encoding so that both parties can communicate properly. Both parties share the encryption key, enabling them to encode and decode each other's messages. This is called symmetric key encryption. This type of encryption is problematic because the key is available in two different places.

An alternative to symmetric key encryption is public key encryption. In public key encryption, two keys are used: a public key and a private key. To send an encrypted message, you obtain the public key, encode the message, and send it. The recipient then uses the private key to decode it. The public key can be given to anyone who wishes to send the recipient a message. Each user simply needs one private key and one public key in order to secure messages. The private key is necessary in order to decrypt something sent with the public key.

LIMITATIONS OF PROJECT

- **Documentation:**

It is time consuming and requires expertise in creating good documentation from view point of top administrators users.

- **Manuals:**

Various manuals are to be prepared such as user manuals, System manuals etc. It needs time, human labor and are subject to change drastically as the technology changes.

- **Online Help:**

One needs to provide online help to various users so that all the features of the software can be properly understood by the user.

- **Complexity:**

It is time consuming to manage and debug all the modules one by one.

FUTURE APPLICATION OF THE PROJECT

- Linking and integration of any legacy system for accounting.
- Integration with other bank and government agencies through Web Services
- Connection to third-party OLAP applications
- Electronic Data Interchange (EDI) system for ATM machine
- Addition of other financial services like loan management, Fixed Deposit.
- In the area of data security and system security.
- Provide more online tips and help.
- Check Book Issueing service.

BIBLIOGRAPHY

Websites

- <http://www.google.com>
- <http://www.codeproject.com>
- <http://www.w3schools.com>
- <http://www.sqltuner.com>
- <https://getbootstrap.com/>
- <https://www.javatpoint.com/>
- https://en.wikipedia.org/wiki/Systems_design

Books

- Head First (JAVA)
- HTML & CSS: Design and Build Web Sites
- Head First SQL: Your Brain on SQ
- SQL Bible, 2nd Edition (Paperback)
- System Analysis, Design, and Development: Concepts, Principles, and Practices, Textbook by Charles S. Wasson
- Java Servlet & JSP Cookbook: Practical Solutions to Real World Problems, Book by Bruce W. Perry