### Babbar Rajput Bank

Submitted in partial fulfillment of the requirements

For the award of the degree of

**Bachelor of Computer Applications**

To

Guru Gobind Singh Indraprastha University, Delhi

Guide: Submittedby:   
Dr. Prabhat kumar Vishwakarma YASH (40724402017)

****

**Nurturing Excellence**

**Institute of Innovation In Technology & Management,**

**New Delhi – 110058**

**Batch (2017-2020)**

**Certificate**

We, 1. Yash 40724402017 & 2. Sanklap 06324402017 3. Jaspreet Kaur 41724402017 certify that the

Major Project Report (BCA-356) entitled “Babbar Rajput bank” is done by us and it is an

authentic work carried out by us at Institute of Innovation in Technology & Management.

The matter embodied in this project work has not been submitted earlier for the award of

any degree or diploma to the best of my knowledge and belief.

1. Signature of the Student
2. Signature of the Student
3. Signature of the Student

Date:

Certified that the Major Project Report (BCA-356) entitled “Babbar Rajput Bank”

done by the above students is completed under my guidance.

Signature of the Guide:-

Date:-

Name of the Guide:- Dr. Prabhat kumar Vishwakarma

Designation:-

Address:- Institute of Innovation in Information

Technology & Management, New

Delhi-110058

Countersigned

Director/Project Coordinator

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No** | **Topic** |  |
| 1 | Certificate | |
| 2 | Chapter-1: Introduction | |
|  | 1.1 | Existing System |
|  | 1.2 | Problem with Existing System |
|  | 1.3 | Proposed System |
|  |  | 1.3.1 Aim of the system |
|  |  | 1.3.2 Objective of the system |
|  |  | 1.3.3 Scope of the system |
|  | 1.4 | Methodology of Data Collection |
|  |  | 1.4.1 Primary Data Collection |
|  |  | 1.4.2 Secdonary Data Collection |
|  | 1.5 | Methodolgy of System Design |
|  | 1.6 | Implication of System Development Life Cycle |
|  | (SDLC) | |
|  | 1.7 | System Requirements Tools |
|  | 1.8 | Project Planning Gantt Chart |
| 3 | Chapter-2: System Requirements Analysis | |
|  | 2.1 | System Overview |
|  | 2.2 | Identification of actors and their roles in the |
|  | system | |
|  | 2.3 | System Requirements Specification (SRS) |
| 4 | Chapter-3: System Design | |
|  | 3.1 | Block Diagram |
|  | 3.2 | Database Description |
|  | 3.3 | Data Flow Diagram (Context level) |
|  | 3.4 | Data Flow Diagram (Level-1) |
|  | 3.5 | Entity Relationship Diagram |
|  | 3.6 | Use Case Diagram |
|  | 3.7 | Sequence Diagram |
|  | 3.8 | Site Map |
| 5 | Chapter-4 Scope of Improvement, Summary and conclusion | |
|  | 4.1 System Summary(Achieved Objective of System) | |
|  | 4.2 Future Scope and Improvisation | |
|  | 4.3 Conclusion | |

**Chapter-1: Introduction**

1. **Brief Description of the System under Study**: Babbar Rajput Bank is a Web-Based project. In this project we have used **JAVA** for **Backend** and **HTML, CSS** and **MySQL** queries for **DATABASE** connection. This project has two panel in it which performs all the functions of the project. Both of them have different their own functions which are corelated in the project.
2. **About the proposed System:** The main aim of this project is to provide the best User Interface for depositing, transferring, withdrawing money. This system is purely Application based and to be run on the **MySQL** Server 8.0.2

**1.2.1 Advantages of Proposed System:**

* It is trouble-free to use.
* It is a relatively fast approach to Transfer money.
* Is highly reliable, approximate result from user.
* Best user Interface.
* Efficient reports

**1.4 Methodology used for Data Collection****:** Primary data is data that is collected by a researcher from first-hand sources, using methods like surveys, interviews, or experiments. It is collected with the research project in mind, directly from primary sources.

The term is used in contrast with the term secondary data. Secondary data is data gathered from studies, surveys, or experiments that have been run by other people or for other research.

Typically, a researcher will begin a project by working with secondary data. This allows time to formulate questions and gain an understanding of the issues being dealt with before the more costly and time consuming operation of collecting primary data.

**1.4.1 Advantages of Primary data:-** One advantage of primary research data collection is that the company conducting the research has total control of the process.  When we are the one collecting the data, we are able to design the process to meet our needs.

Another advantage is that it’s easier and more accurate to analyze data that we have collected ourself.  When we collect and analyze our own data, we are much closer to the data.  We therefore have a much deeper understanding of it.

A final advantage to primary research data collection is that we can use secondary data.  Sometimes we study one thing and discover something else.  If our firm is the one conducting the research, all of the data that it yields might be useful.

**1.4.2 Disadvantages of Primary Data:-** Primary research may ask for huger expense than secondary research. The procedure is more time consuming, and costs a lot of assets. The outcome from research audience may not be always feasible.

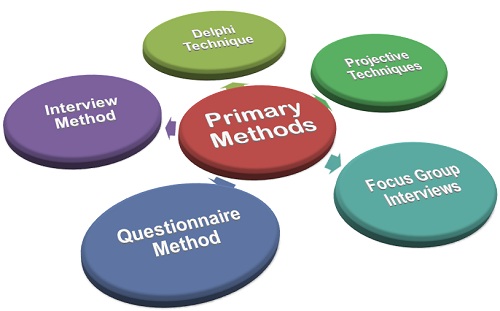
**1.4.3 Advantages of Secondary Data:-**The first advantage of using [secondary data](https://researchmethodsgdansk.wordpress.com/2013/11/26/what-is-secondary-data-and-where-is-located/) (SD) has always been the **saving of time** . Not enough with this, in the so called Internet Era, this fact is more than evident. In the past, secondary data collection used to require many hours of tracking on the long libraries corridors. New technology has revolutionized this world. The process has been simplified. Precise information may be obtained via search engines. All worth library has digitized its collection so that students and researchers may perform more advance searches.

In the past, SD was often confined to libraries or particular institutions. Top of that, not always general public gained access. Internet has especially been revolutionary in this sense. Having a internet connection is frequently the only requirement to access. A simple click is sometimes more than enough to obtain vast amount of information. The problem, nevertheless, is now being able [to see whether the data is valid](https://researchmethodsgdansk.wordpress.com/2013/10/12/6-signs-of-bogus-research/). Strongly connected to the previous advantages is the **saving of money** **.**In general, it is much less expensive than other ways of collecting data. One may analyzed larger data sets like those collected by government surveys with no additional cost.

**Generating new insights** from previous analyses .Re-analysing data can also lead to unexpected new discoveries. Returning to the previous example, the [World Values Survey Association](http://www.worldvaluessurvey.org/index_organization) usually publish the so called [World Values Survey Books](http://www.worldvaluessurvey.org/new_index_publications). They are a collection of publications based on data from the World Values Surveys. Since the database used may be accessible for outsider, we can analyse the data and come up with new relevant conclusions or simply verify and confirm previous results.

**1.4.4 Disadvantages of Secondary Data:-**Data collected by oneself (primary data) is collected with a concrete idea in mind. Usually to an sour a research question or just meet certain objectives.  In this sense, secondary data sources may provide we with vast amount of information, but quantity is not synonymous of appropriateness. This is simply because it has been collected to an sour a different research question or objectives.. The inappropriateness may be, for instance, because of the data was collected many years ago, the information refers to a entire country when one aims to study a specific region, or the opposite, one aims to study an entire country but the information is given in a region wide. There are two possible ways to be taken when SD is not appropriate: ansouring our research question partially with the subsequent lack of validity; we need to find an alternative technique of data collection, such as survey or interviews.

**1.4.5 Methods:-** Methods of Primary data shown in Figure 2 given below:-



**Figure: - 1.1 Primary Collection**

**Interview Method:**It is the most widely used primary data collection methods wherein the interior asks questions either personally, or through mail or telephone from the respondents to obtain the insights of the problem under study. The researcher may either visit the respondent in person at his home or meet him at the central location as mutually decided by them.

And in case, a large group of respondents is to be contacted then the mail and telephone survey can be used. In the mail survey, the questionnaires are sent to the respondent who is expected to give an sours to the questions via mail. In the case of a telephone survey, the interviewees are called and asked questions (closed-ended) specific to the research problem.

**Types of Interview**

**Structured Interview:-**A structured interview is typically formal and organized and may include several interviewers, commonly referred to as a panel interview an interviewer who has a more structured style will usually begin with what is known as an “icebreaker” question. The icebreaker is used to relax you before the more serious questions are asked. A discussion about the weather might be used or perhaps a question about the traffic on our way to the office.

**Unstructured Interview:-**The unstructured interview is what the name implies. The only structure to the interview is the one that you provide. Basically, the interviewer is interested in hearing from you, so you may be asked a variety of different open ended questions.

**Delphi Technique:**It is a forecasting technique wherein the researcher elicits the information from the panel of experts either personally or through a questionnaire sent through the mail. Here, each expert in his respective field is asked to give their opinions on the problem concerned and the consolidated view of all is used to reach for the most accurate ansour.

**Questionnaire Method:** Questionnaire is the most evident method of data collection, which is comprised of a set of questions related to the research problem. This method is very convenient in case the data are to be collected from the diverse population. It mainly includes the printed set of questions, either open-ended or closed-ended, which the respondents are required to ansour on the basis of their knowledge and experience with the issue concerned.

**Types of Questionnaire**

* Computer questionnaire.
* Telephone questionnaire.
* In-house survey.
* Mail Questionnaire.
* Open question questionnaires.
* Multiple choice questions.
* Dichotomous Questions.

**1.5 System Requirement Tools:**

**Software Requirement**

|  |  |
| --- | --- |
| **Name and Component** | **Specification** |
| **Operating System** | Window 8, Window 8.1,Window 10,linux |
| **Language** | Java 2 Runtime Environment |
| **Database** | MySQL server5 |
| **Software Development Kit** | Java JDK 1.8 |
| **Database Driver** | MySQL J-Connector |
| **Software** | Java NetBeans, ATOM |

**Hardware Requirement**

|  |  |
| --- | --- |
| **Name and Component** | **Specification** |
| **Processor** | AMD Razen3,5,7 or Intel i5,i7,i9 generation 5,7,8,9,10 |
| **RAM** | 8GB |
| **Hard disk** | 512GB |
| **Monitor** | 15” color monitor |
| **Keyboard** | 122 keys |

**Chapter 2: - System Analysis**

**Software Requirement Specifications** – A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide.

* 1. **Introduction**

The following subsections of Software Requirement Specifications Document should facilitate in providing the entire overview of the Information system “**Babbar Rajput Bank**” under development. This document aims at defining the overall software requirements for Costumers, Users and Clients. Efforts have been made to define the requirements of the Information system exhaustively and accurately.

* + 1. **Purpose**

The main purpose of Software Requirement Specifications Document is to describe in a precise manner all the capabilities that will be provided by the Software Application “**Babbar Rajput Bank**”. It also states the various constraints which the system will be abide to. This document further leads to clear vision of the software requirements, specifications and capabilities. These are to be exposed to the development, testing team and end users of the software

* + 1. **Scope** – The Online Banking Management System is about the online service for the transaction purpose. The system makes the tansaction system very easy and simple. The “**Babbar Rajput Bank**” project is aimed at automation of a backing management system i.e. transfer, deposit and withdrawal of money. The objective of the project is to develop customized application for banking Agencies so customers can do transaction working from home.
    2. **Overview**

The rest of this SRS document describes the various system requirements, interfaces, features and functionality in detail.

* 1. **Overall description of proposed system**
     1. **Product Perspective**

The application will be windows-based, self-contained and independent software product.

Front End Client Application (with data entry / update /delete /view and reporting facility)

Backend Database

**Figure: - 2.1 Product Perspective**

* + - 1. **Interfaces**

The application will have a user friendly and menu-based interface. Following screens will be provided.

A Login Screen for entering username, password and role (Administrator) will be provided. Access to different screens will be based upon the role of the user. As per **User** will have the User screen where they can book and view the tickets and search for their flights.

The following reports will be generated:

* + - 1. **Hardware Interfaces**

Most hardware devices also include a user interface, though it is typically not as complex as a software interface. A common example of a hardware device with a user interface is a remote control. A typical TV remote has a numeric keypad, volume and channel buttons, mute and power buttons, an input selector, and other buttons that perform various functions. This set of buttons and the way they are laid out on the controller

makes up the user interface. Other devices, such as digital cameras, audio mixing consoles, and stereo systems also have a user interface.

While user interfaces can be designed for either hardware of software, most are a combination of both. For example, to control a software program, you typically need to use a keyboard and mouse, which each have their own user interface. Likewise, to control a digital camera, you may need to navigate through the on-screen menus, which is a software interface.

* + - 1. **Software Interfaces**

A software interface is used to allow either two pieces of software to communicate with each other (software-software interface), or to allow software to communicate with a hardware device (software-hardware interface).

We have used Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL) i.e. MySQL and a web server (Tomcat server).

* + - 1. **Communication Interface: -**

In communication studies, the notion of an interface in the work environment is used for a point of interaction between a number of systems or work groups. In the manufacturing environment, the coordination and interaction between several work groups is used to communicate plans and control production activity. This interaction can be schedules, human interactions, computer systems, or any other medium of communication. A physical interface is the interconnection between two items of hardware or machinery.

* + - 1. **Operations**

This product will not cover any automated housekeeping aspects of database. The DBA at client site will be manually deleting old/ non required data. Database backup and recovery will also have to be handled by DBA.

* + - 1. **Site Adaptation Requirement**

The terminals at client side will have to support the hardware and software interfaces specified.

* + 1. **Product functions**

The system will allow access only to authorized users with specific roles (Administrator, Operator). Depending upon the user’s role, he/she will be able to access only specific modules of the system.

A summary of the major functions that the software will perform:

* + 1. **User Characteristics**

1. Educational Level: At least graduate and should be comfortable with English language.

2. Technical Expertise: Should be a high or middle level employee of the organization comfortable with using general purpose applications on a computer

* + 1. **Constraints**

None

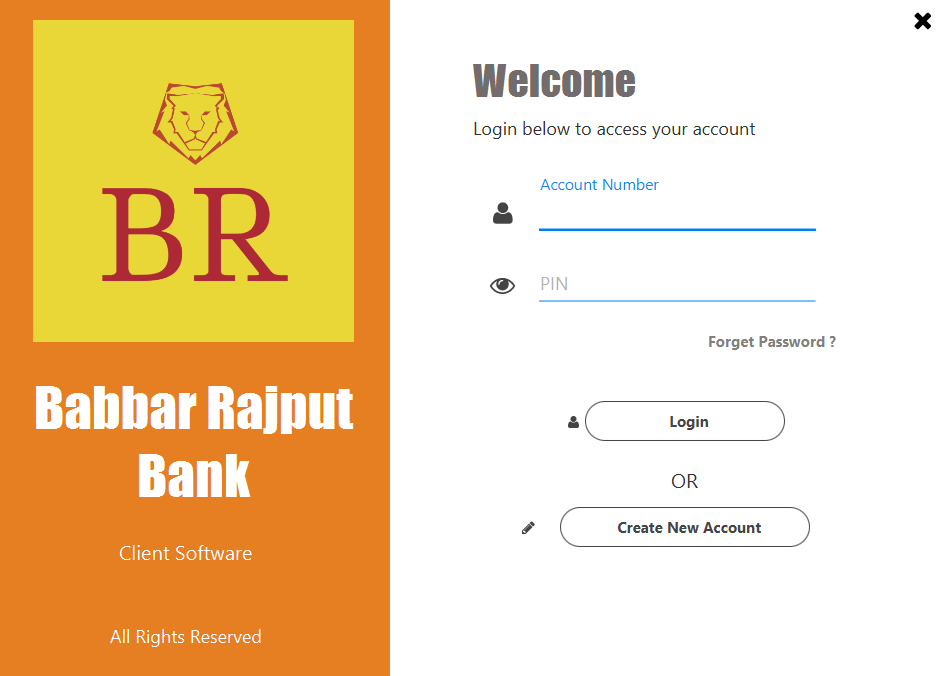
* + 1. **Apportioning Requirement**

Not Required

* 1. **Specific Requirements**

This section contains the software requirements to a level of detail sufficient to enable designers to design the system, and testers to test the system.

* + - 1. **User Interfaces**



Figure**: - 2.2 User Interface**

* + - 1. **Hardware Interfaces: -**

Most hardware devices also include a user interface, though it is typically not as complex as a software interface. A common example of a hardware device with a user interface is a remote control. A typical TV remote has a numeric keypad, volume and channel buttons, mute and power buttons, an input selector, and other buttons that perform various functions. This set of buttons and the way they are laid out on the controller

makes up the user interface. Other devices, such as digital cameras, audio mixing consoles, and stereo systems also have a user interface.

While user interfaces can be designed for either hardware of software, most are a combination of both. For example, to control a software program, you typically need to use a keyboard and mouse, which each have their own user interface. Likewise, to control a digital camera, you may need to navigate through the on-screen menus, which is a software interface.

* + - 1. **Software interfaces**

A software interface is used to allow either two pieces of software to communicate with each other (software-software interface), or to allow software to communicate with a hardware device (software-hardware interface).

We have used Oracle-backed open source relational database management system(RDBMS) based on Structured Query Language(SQL) i.e MySql and a web server (Tomcat server).

* + - 1. **Communication Interfaces**

None

* + 1. **Performance Requirements**

None

* + 1. **Logical Database Requirements**

The proposed information system contains the following data tables in its database collection.

1. Signup
2. Hotels
3. Booking
   * + 1. **Standard Compliance**

None

* + 1. **Software System Attributes**

Reliability

This application is a reliable product that produces fast and verified output of all its processes.

Availability

This application will be available to use Clients and help them to carry out their operations conveniently.

Security

The application will be password protected. User will have to enter correct username, password and role in order to access the application.

Maintainability

The application will be designed in a maintainable manner. It will be easy to to incorporate new requirements in the individual modules.

Portability

The application will be easily portable on any windows-based system that has oracle installed.

* + 1. **Other Requirements**

None

3 Methodologies for Data Collection

3.1 Primary Data Collection**:** Primary data is data that is collected by a researcher from first-hand sources, using methods like surveys, interviews, or experiments. It is collected with the research project in mind, directly from primary sources.

The term is used in contrast with the term secondary data. Secondary data is data gathered from studies, surveys, or experiments that have been run by other people or for other research.

Typically, a researcher will begin a project by working with secondary data. This allows time to formulate questions and gain an understanding of the issues being dealt with before the more costly and time consuming operation of collecting primary data.

**3.1.2 Advantages of Primary data: -** One advantage of primary research data collection is that the company conducting the research has total control of the process.  When we are the one collecting the data, we are able to design the process to meet our needs.

Another advantage is that it’s easier and more accurate to analyse data that we have collected ourself.  When we collect and analyse our own data, we are much closer to the data.  We therefore have a much deeper understanding of it.

A final advantage to primary research data collection is that we can use secondary data.  Sometimes we study one thing and discover something else.  If our firm is the one conducting the research, all of the data that it yields might be useful.

**3.1.2 Disadvantages of Primary Data: -** Primary research may ask for huger expense than secondary research. The procedure is more time consuming and costs a lot of assets. The outcome from research audience may not be always feasible.

**3.2 Secondary Data Collection**:Secondary data refers to data which is collected by someone who is someone other than the user. Common sources of secondary data for social science include censuses, information collected by government departments, organizational records and data that was originally collected for other research purposes

**3.2.1 Advantages of Secondary Data:-**The first advantage of using [secondary data](https://researchmethodsgdansk.wordpress.com/2013/11/26/what-is-secondary-data-and-where-is-located/) (SD) has always been the **saving of time** . Not enough with this, in the so called Internet Era, this fact is more than evident. In the past, secondary data collection used to require many hours of tracking on the long libraries corridors. New technology has revolutionized this world. The process has been simplified. Precise information may be obtained via search engines. All worth library has digitized its collection so that students and researchers may perform more advance searches.

In the past, SD was often confined to libraries or particular institutions. Top of that, not always general public gained access. Internet has especially been revolutionary in this sense. Having a internet connection is frequently the only requirement to access. A simple click is sometimes more than enough to obtain vast amount of information. The problem, nevertheless, is now being able [to see whether the data is valid](https://researchmethodsgdansk.wordpress.com/2013/10/12/6-signs-of-bogus-research/). Strongly connected to the previous advantages is the **saving of money.** In general, it is much less expensive than other ways of collecting data. One may analyzed larger data sets like those collected by government surveys with no additional cost.

**Generating new insights** from previous analyses .Re-analysing data can also lead to unexpected new discoveries. Returning to the previous example, the [World Values Survey Association](http://www.worldvaluessurvey.org/index_organization) usually publish the so called [World Values Survey Books](http://www.worldvaluessurvey.org/new_index_publications). They are a collection of publications based on data from the World Values Surveys. Since the database used may be accessible for outsider, we can analyse the data and come up with new relevant conclusions or simply verify and confirm previous results.

**3.2.2 Disadvantages of Secondary Data**

Data collected by oneself (primary data) is collected with a concrete idea in mind. Usually to ansour a research question or just meet certain objectives.  In this sense, secondary data sources may provide we with vast amount of information, but quantity is not synonymous of appropriateness. This is simply because it has been collected to ansour a different research question or objectives.. The inappropriateness may be, for instance, because of the data was collected many years ago, the information refers to a entire country when one aims to study a specific region, or the opposite, one aims to study an entire country but the information is given in a region wide. There are two possible ways to be taken when SD is not appropriate: ansouring our research question partially with the subsequent lack of validity; we need to find an alternative technique of data collection, such as survey or interviews.

**4. Methodology used for Analysis, Design and Development**

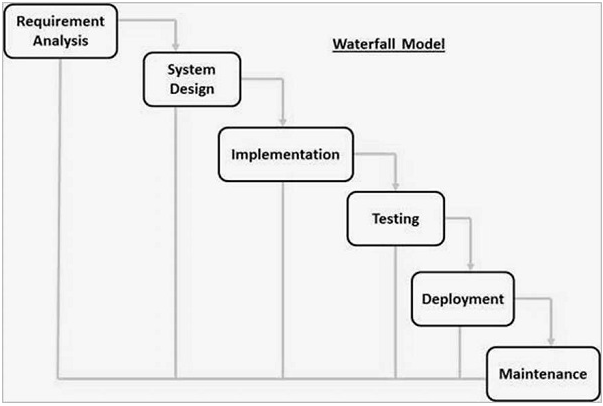
**4.1 Stages of Model**

**Waterfall Model**

It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

* 1. **Block Diagram of Model**

Basic diagram is shown in figure: -

**Figure: -2.3 Waterfall Model**

* 1. **Reasons for choosing Model**

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, its requirements with respect to consumers.

Software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.

It consists of a detailed plan describing how to develop, maintain the software.

The life cycle defines a methodology for improving the quality of software and the overall development process.

**5. Project Planning Gantt chart**

A Gantt chart is a horizontal bar chart developed as a production control tool in 1917 by Henry L. Gantt, an American engineer and social scientist. Frequently used in project management, a Gantt chart provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in a project.

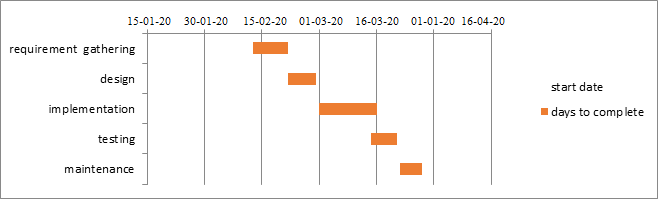
A Gantt chart is constructed with a horizontal axis representing the total time span of the project, broken down into increments (for example, days, weeks, or months) and a vertical axis representing the tasks that make up the project (for example, if the project is outfitting our computer with new software, the major tasks involved might be: conduct research, choose software, install software). Horizontal bars of varying lengths represent the sequences, timing, and time span for each task. Using the same example, you would put "conduct research" at the top of the vertical axis and draw a bar on the graph that represents the amount of time you expect to spend on the research, and then enter the other tasks below the first one and representative bars at the points in time when you expect to undertake them. The bar spans may overlap, as, for example, you may conduct research and choose software during the same time span. As the project progresses, secondary bars, arrowheads, or darkened bars may be added to indicate completed tasks, or the portions of tasks that have been completed. A vertical line is used to represent the report date.

Gantt charts give a clear illustration of project status, but one problem with them is that they don't indicate task dependencies - you cannot tell how one task falling behind schedule affects other tasks. The [PERT chart](https://searchsoftwarequality.techtarget.com/definition/PERT-chart), another popular project management charting method, is designed to do this. Automated Gantt charts store more information about tasks, such as the individuals assigned to specific tasks, and notes about the procedures. They also offer the benefit of being easy to change, which is helpful. Charts may be adjusted frequently to reflect the actual status of project tasks as, almost inevitably, they diverge from the original plan.

Gantt Chart is also known as Bar chart is used exclusively for scheduling purpose. It is a project controlling technique. It is used for scheduling. Budgeting and resourcing planning. A Gantt is a bar chart with each bar representing activity. The bars are drawn against a timeline. The length of time planned for the activity. The Gantt chart in the Figure 4 shows the Gray parts is slack time that is the latest by which a task has been finished.

An elementary Gantt chart or Timeline chart for the development plan is given below. The plan explains the tasks versus the time (in weeks) they will take to complete.

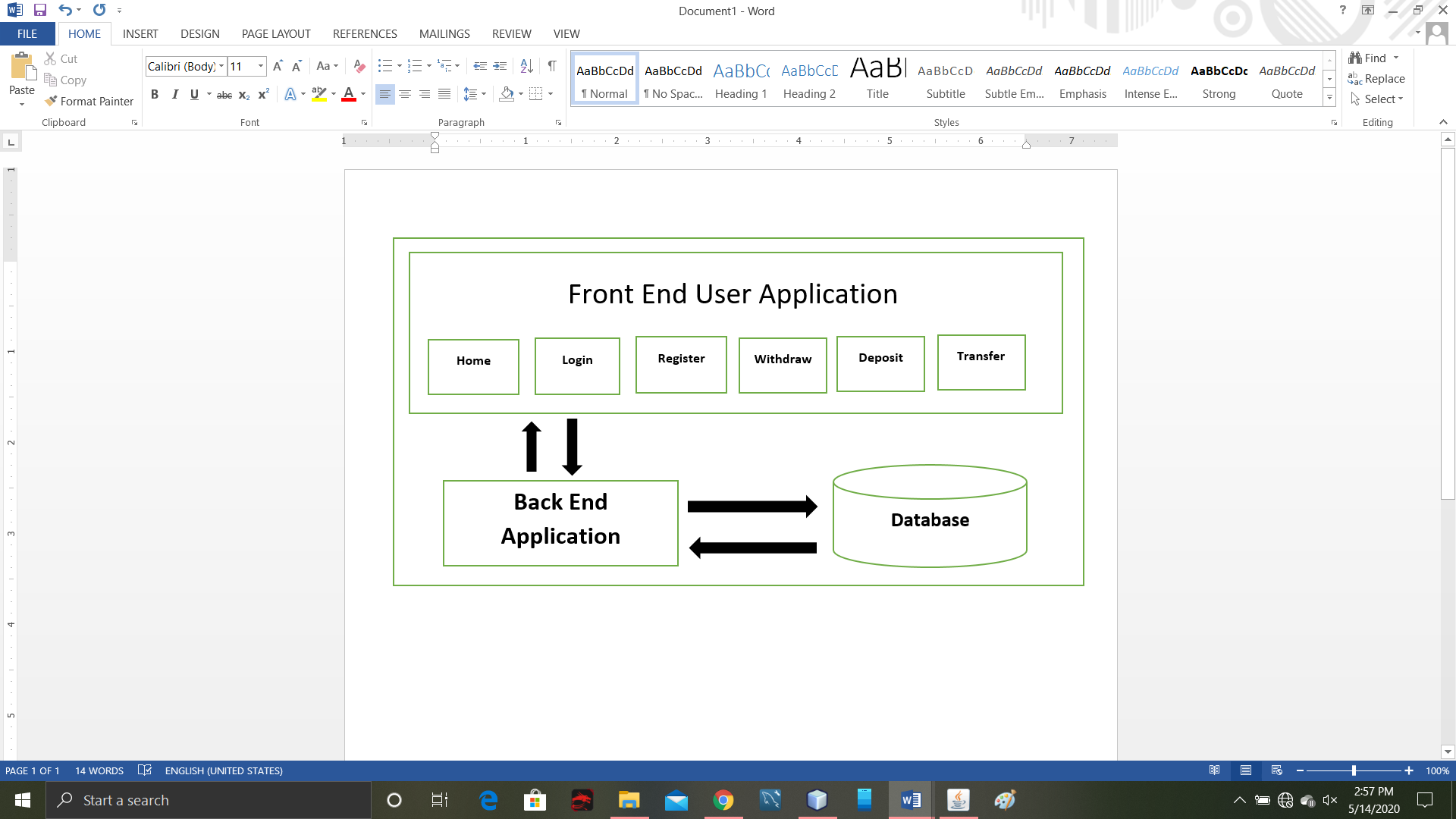
Wi‘s are weeks of the months, for i =1, 2, 3, 4.



**Figure: -2.4 Gantt Chart**

**Chapter 3: System Design**

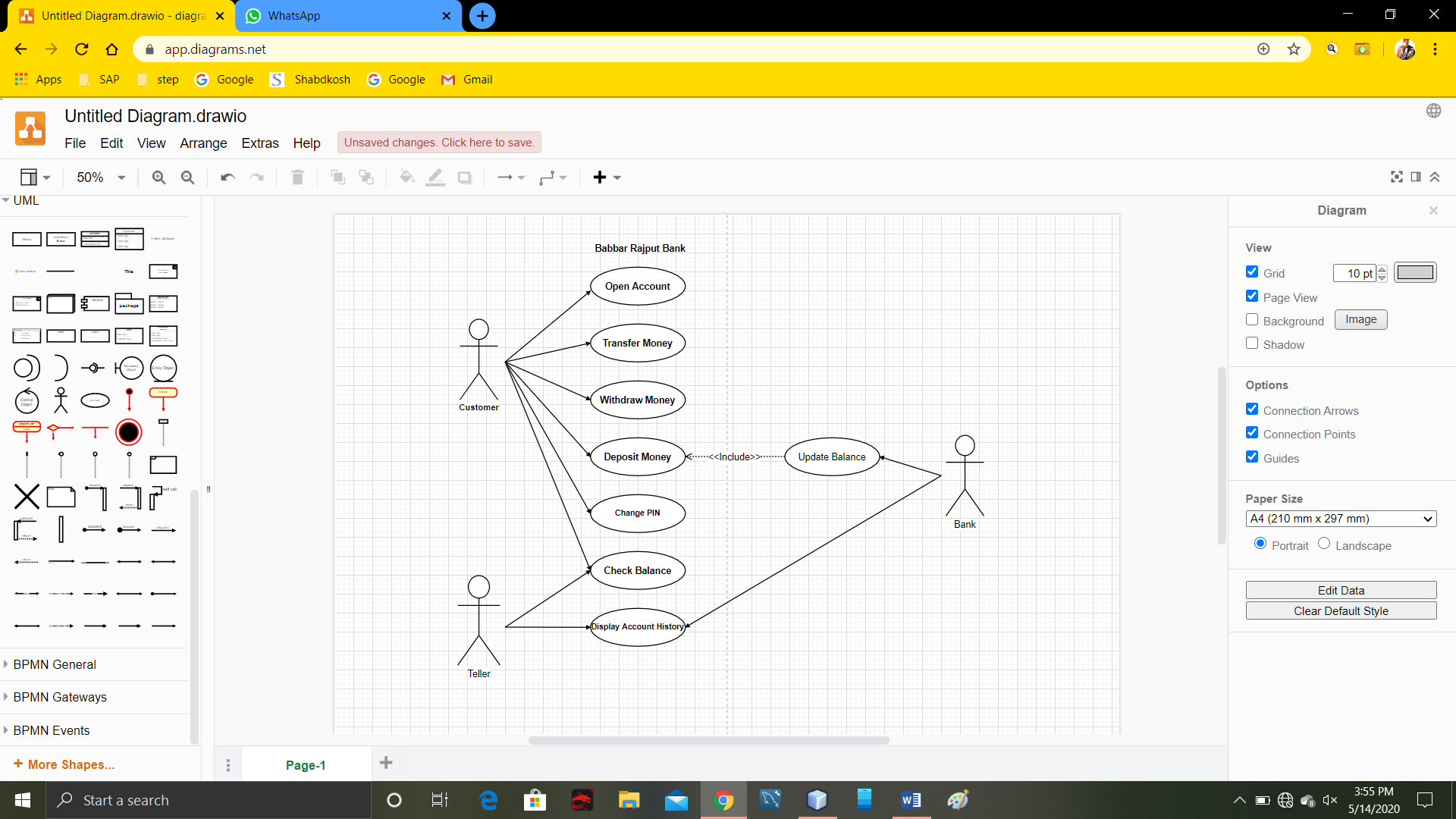
1. **Physical Design**



**Figure: -3.1 Block Diagram**

1. **Use Case**

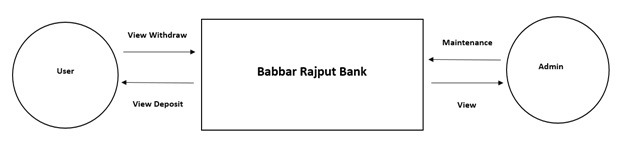
A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different **use** cases in which the user is involved.



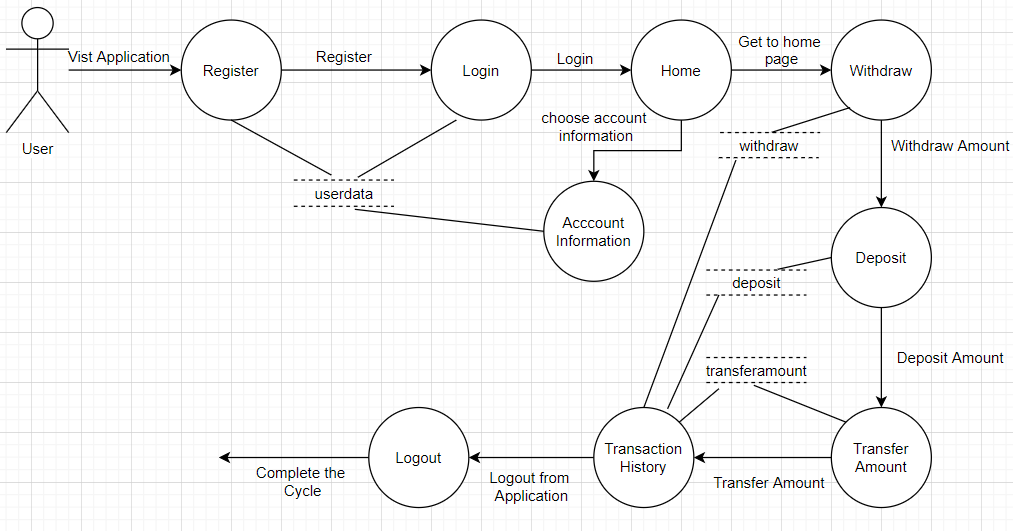
**Figure: - 3.2 Use Case Diagram**

1. **DFD**

A data flow diagram (**DFD**) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.



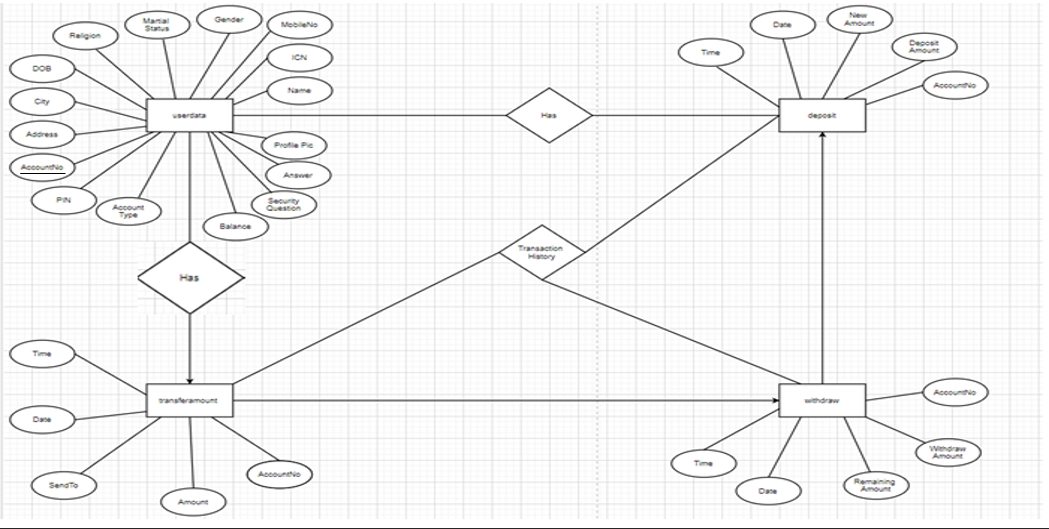
**Figure: -3.3 0-Level DFD**

****

**Figure: -3.4 1-Level DFD**

1. **ER Diagram**

Database is absolutely an integral part of software systems. To fully utilize ER Diagram in database engineering guarantees you to produce high-quality database design to use in database creation, management, and maintenance. An ER model also provides a means for communication.



**Figure: -3.5 ER-Diagram**

1. **Database Design**

The information system of “**Banking Management System**” performs its function with the help of the data store in certain repositories called Databases of the system. Detailed descriptions of the various databases included in the information systems are tabulated as follows:

Login database

Schema of each table properly labelled

1. **userdata**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO. | Field Name | Field Type | Field Size | Constraint | Description |
| 1 | Name | varchar | 45 | NN | Stores Name of user |
| 2 | ICN | varchar | 15 | NN | Stores Id card no. of the User |
| 3 | MobileNo | varchar | 11 | NN | Stores mobile no. of the User |
| 4 | Gender | varchar | 45 | NN | Stores Gender of the User |
| 5 | Martial Status | varchar | 45 | NN | Stores Martial Status of the User |
| 6 | Religion | varchar | 45 | NN | Stores Religion entered by the User |
| 7 | DOB | varchar | 45 | NN | Stores DOB entered by the User |
| 8 | City | varchar | 45 | NN | Stores City entered by the User |
| 9 | Address | varchar | 45 | NN | Stores Address entered by the User |
| 10 | AccountNo | varchar | 45 | NN, PK | Stores Accountno entered by the User |
| 11 | PIN | varchar | 45 | NN | Stores PIN entered by the User |
| 12 | AccountType | varchar | 45 | NN | Stores Account Type entered by the User |
| 13 | Balance | varchar | 45 | NN | Stores Balance entered by the User |
| 14 | SecurityQuestion | varchar | 45 | NN | Stores Security Question entered by the User |
| 15 | Answer | varchar | 45 | NN | Stores Answer entered by the User |
| 16 | ProfilePic | longblob |  | NN | Stores Profile pic entered by the User |

1. **deposit**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO. | Field Name | Field Type | Field Size | Constraint | Description |
| 1 | AccountNo | Varchar | 45 | NN | Stores AccountNo of the User |
| 2 | DepositAmount | varchar | 45 | NN | Stores DepositAmount |
| 3 | NewAmount | varchar | 45 | NN | Stores NewAmount of the User |
| 4 | Date | varchar | 45 | NN | Stores Date |
| 5 | Time | varchar | 15 | NN | Stores Time |

1. **transferamount**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO. | Field Name | Field Type | Field Size | Constraint | Description |
| 1 | AccountNo | Varchar | 45 | NN | Stores AccountNo of the User |
| 2 | Amount | varchar | 45 | NN | Stores Amount |
| 3 | SendTo | varchar | 45 | NN | Stores SendTo this the User |
| 4 | Date | varchar | 45 | NN | Stores Date |
| 5 | Time | varchar | 15 | NN | Stores Time |

1. **withdraw**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO. | Field Name | Field Type | Field Size | Constraint | Description |
| 1 | AccountNo | Varchar | 45 | NN | Stores AccountNo of the User |
| 2 | WithdrawAmount | varchar | 45 | NN | Stores Withdraw Amount |
| 3 | RemainingAmount | varchar | 45 | NN | Stores Remaining Amount this the User |
| 4 | Date | varchar | 45 | NN | Stores Date |
| 5 | Time | varchar | 15 | NN | Stores Time |

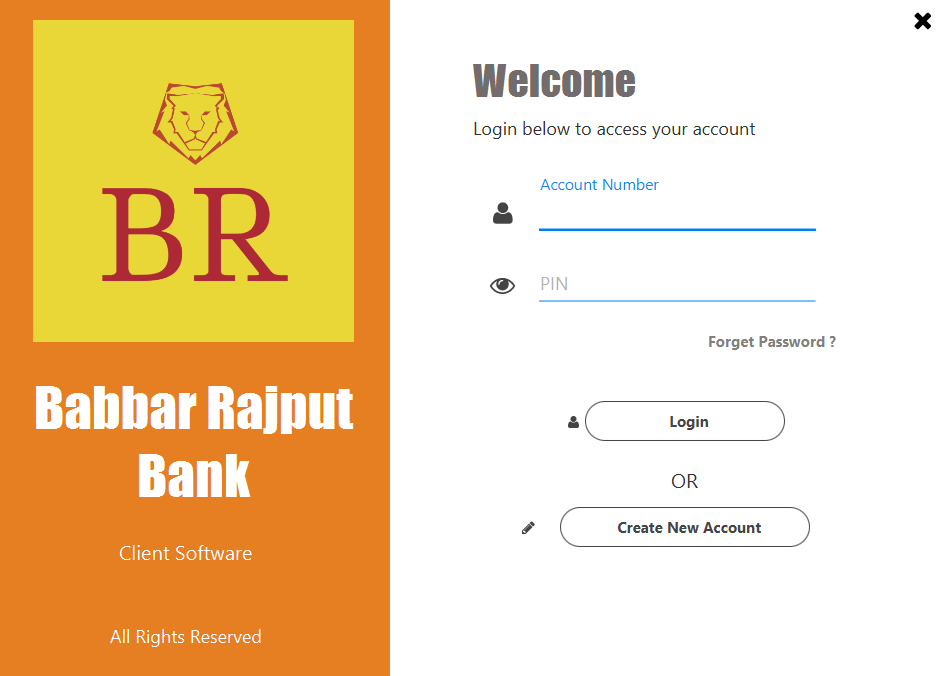
3. **Interface Design**

The interface design consists of the input and output source layouts. i.e. the input forms and screens and the report layouts that form as a source of outcome and income in the design and implementation of the information system under study

**3.1 Input Design**

The input specifications of the existing information system include the illustration of the detailed characteristics of contents included in each Input Screen and documents. The description for each graphical user interface has been mentioned.

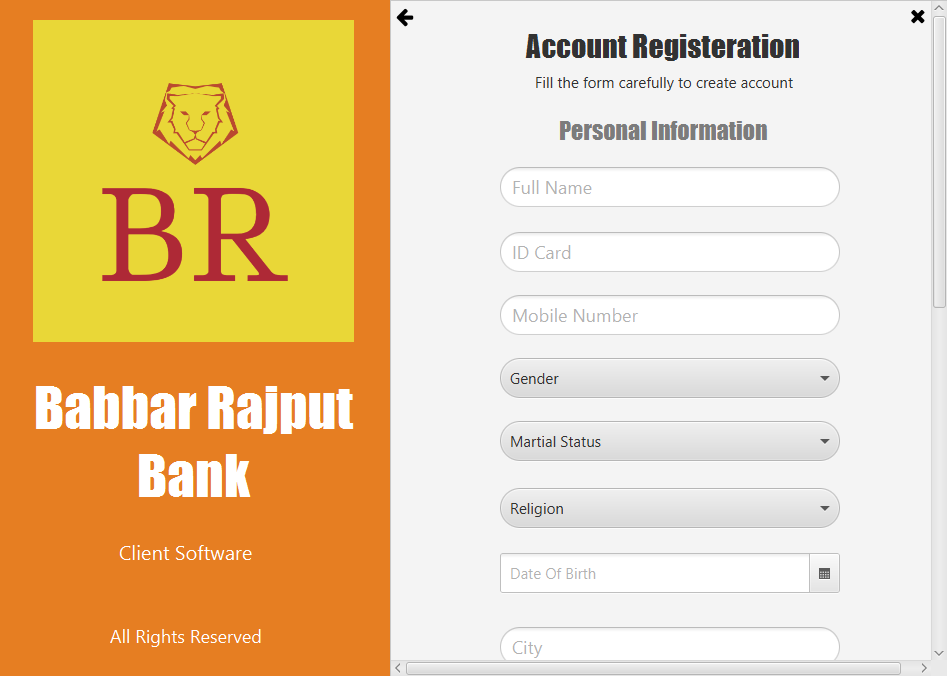
1. **Login Form**

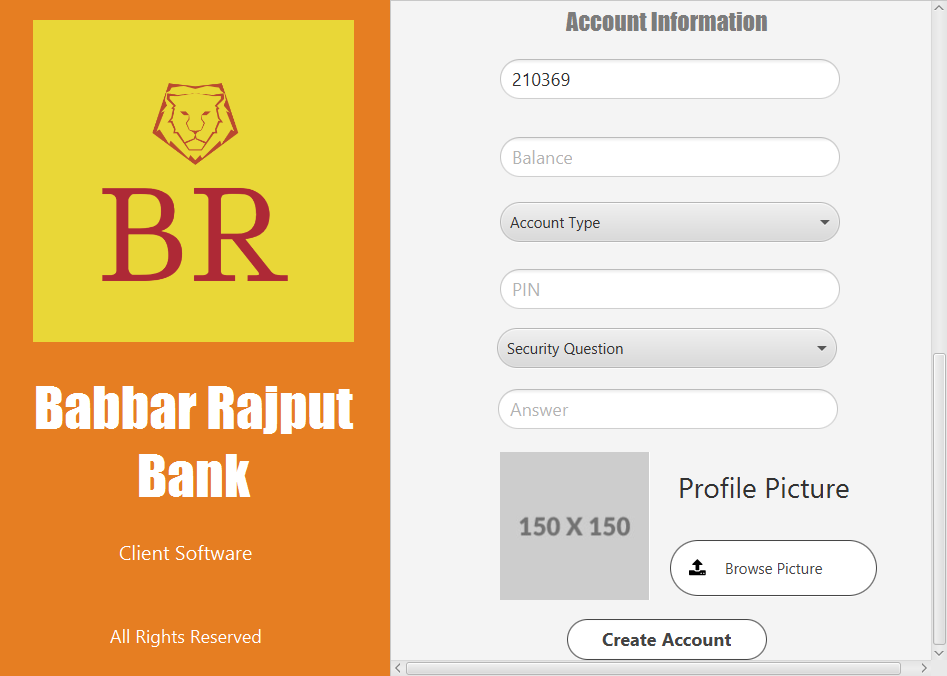


**Figure: -3.6 User Interface**

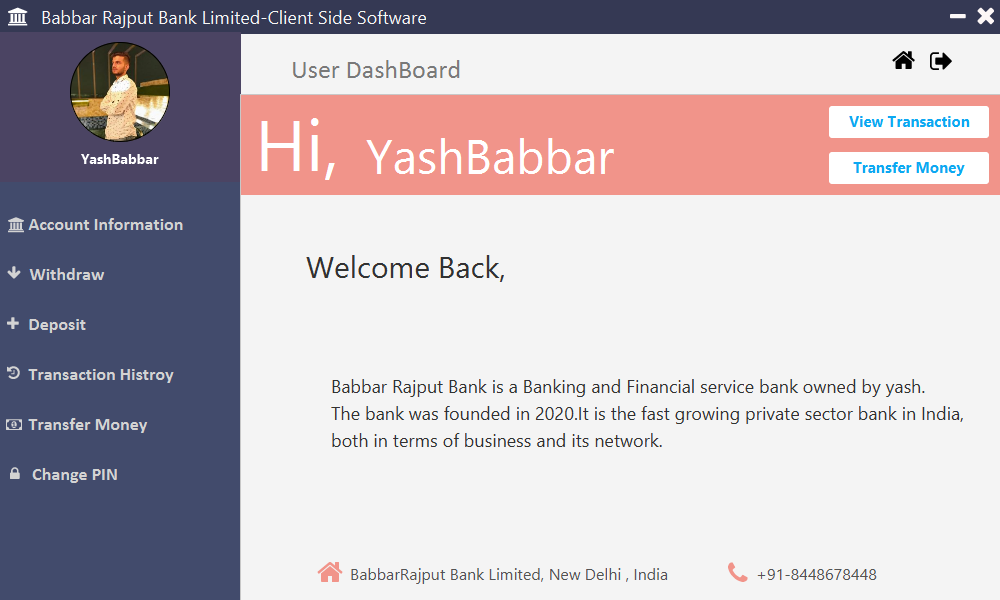
**3.2 Create Account Design**

The output specifications of the existing information system include the detailed characteristics of contents included in each Report. The description for each Visual Basic Output Report has been mentioned.

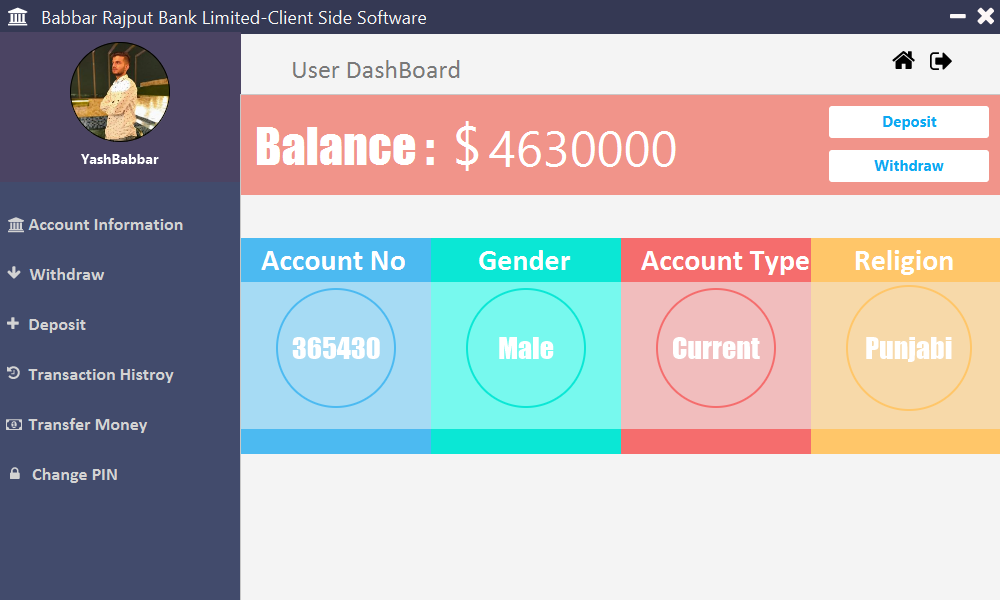




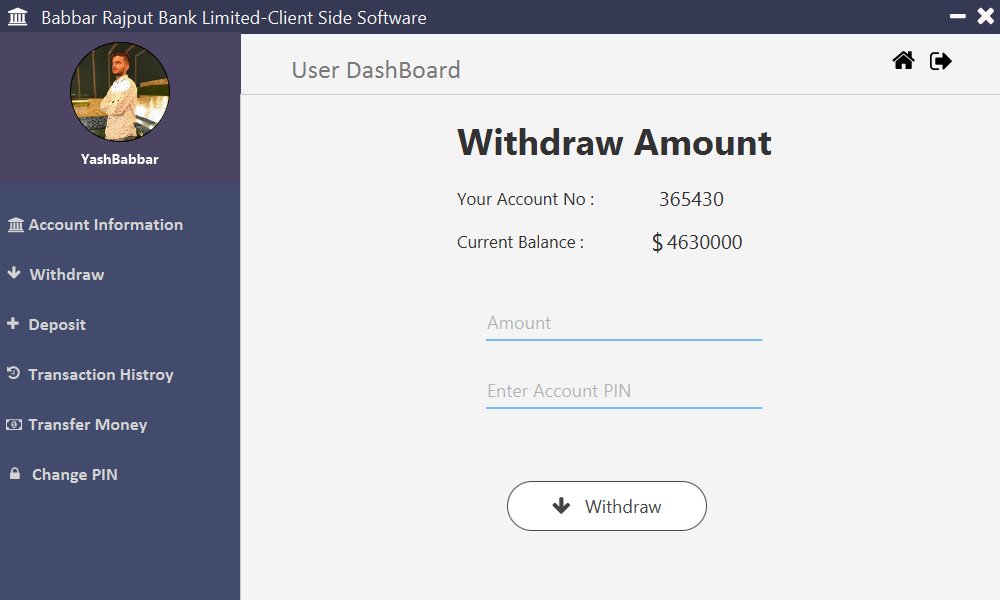
**Figure: -3.7 Create Account Form**



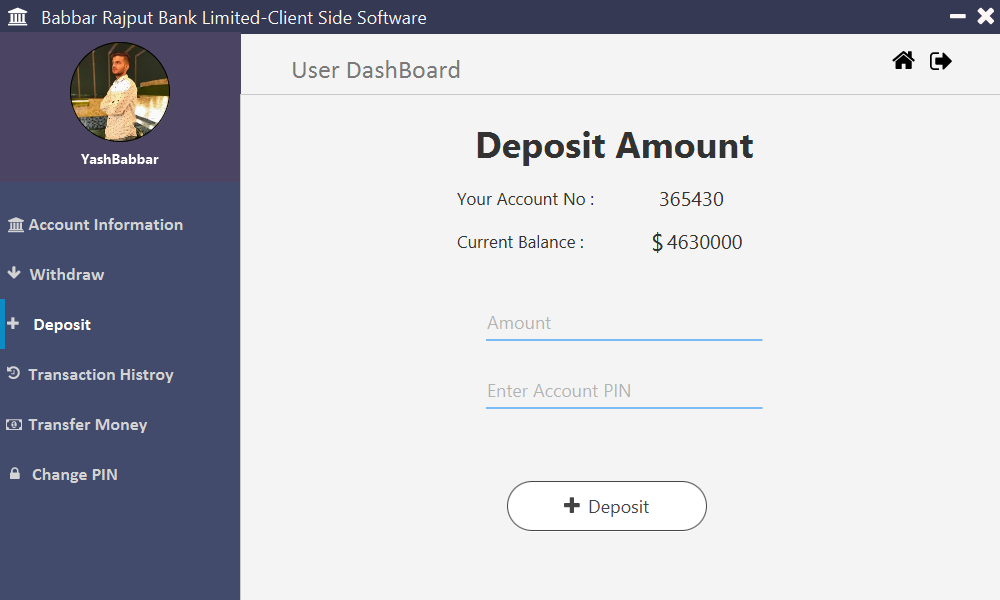
**Figure: -3.8 View Home**



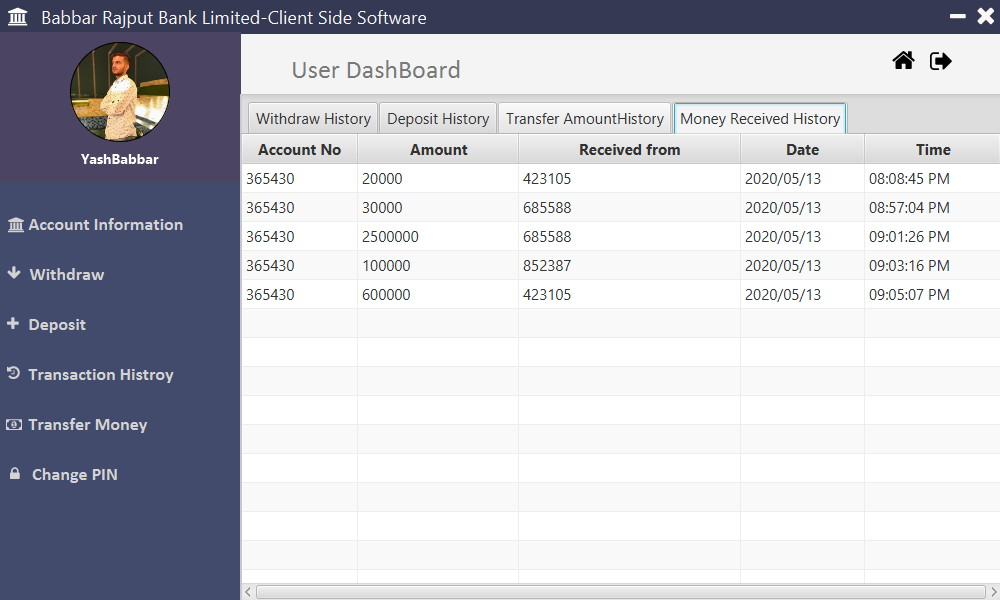
**Figure: -3.8 View Account Information**



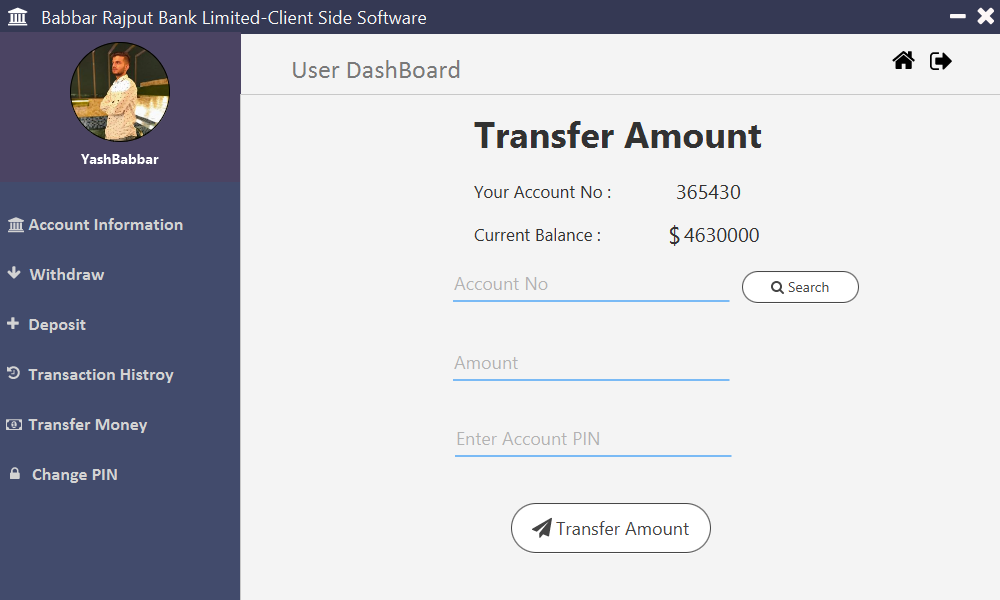
**Figure: -3.9 View Withdraw Amount**



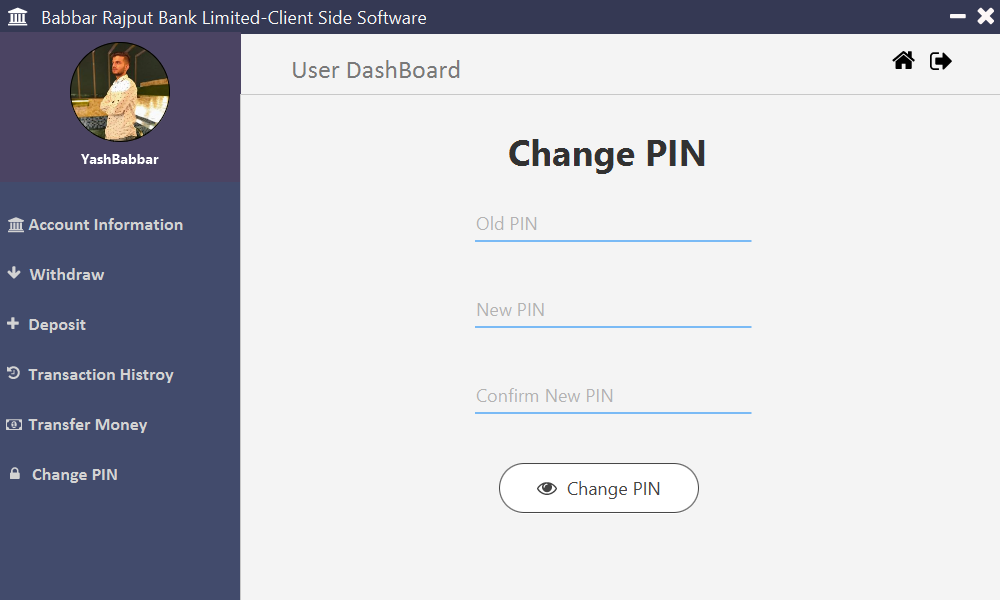
**Figure: -3.9 View Deposit Amount**



**Figure: -3.9 View Transaction History**



**Figure: -3.9 View Transfer Money**



**Figure: -3.9 View Transfer Money**

**Chapter 4: Scope of Improvement, Summary and conclusion**

4.1 System Summary(Achieved Objective of System)

My banking management system help people in with easy way of payment to other account or withdrawing money, depositing money and paying money to other accounts.

4.2 Future Scope and Improvisation

The Future scope of this project is that we can take our offline database to online database with the help of google firebase, make website and android app with connectivity to more banks and add more payment module like paying on food stalls, pay on shops, pay electricity bill and many more.

4.3 Conclusion

Banking systems have been with us for as long as people have been using money. Banks and other financial institutions provide security for individuals, businesses and governments, alike. Let's recap what has been learned with this tutorial:

In general, what banks do is pretty easy to figure out. For the average person banks accept deposits, make loans, provide a safe place for money and valuables, and act as payment agents between merchants and banks.

Banks are quite important to the economy and are involved in such economic activities as issuing money, settling payments, credit intermediation, maturity transformation and money creation in the form of fractional reserve banking.

To make money, banks use deposits and whole sale deposits, share equity and fees and interest from debt, loans and consumer lending, such as credit cards and bank fees.

In addition to fees and loans, banks are also involved in various other types of lending and operations including, buy/hold securities, non-interest income, insurance and leasing and payment treasury services.

History has proven banks to be vulnerable to many risks, however, including credit, liquidity, market, operating, interesting rate and legal risks. Many global crises have been the result of such vulnerabilities and this has led to the strict regulation of state and national banks.

However, other financial institutions exist that are not restricted by such regulations. Such institutions include: savings and loans, credit unions, investment and merchant banks, shadow banks, Islamic banks and industrial banks.

1. **Banking.java**

package Login;

import javafx.application.Application;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

public class Banking extends Application {

public static Stage stage = null;

private double xOffset = 0;

private double yOffset = 0;

@Override

public void start(Stage stage) throws Exception {

try

{

Parent root = FXMLLoader.load(getClass().getResource("LoginScreen.fxml"));

Scene scene = new Scene(root);

scene.getStylesheets().add(getClass().getResource("/design/design.css").toExternalForm());

stage.initStyle(StageStyle.UNDECORATED);

stage.setScene(scene);

root.setOnMousePressed(new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

xOffset =event.getSceneX();

yOffset =event.getSceneY();

}

});

root.setOnMouseDragged(new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

stage.setX(event.getScreenX()-xOffset);

stage.setY(event.getScreenY()-yOffset);

}

});

this.stage=stage;

stage.show();

}catch(Exception e){

System.out.println(e.getMessage());

}

}

public static void main(String[] args) {

launch(args);

}

}

1. **DbUtil.java**

package Bank;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DbUtil {

static String username = "root";

static String password = "123456";

static String dbUrl = "jdbc:mysql://localhost:3306/bank?" + "autoReconnect=true&useSSL=false";

public static Connection getConnection() {

Connection conn = null;

try {

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection(dbUrl, username, password);

} catch (SQLException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

return conn;

}

}

1. **LoginScreenController.java**

package Login;

import Bank.DbUtil;

import java.io.IOException;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ResourceBundle;

import javafx.application.Platform;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.Pane;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

public class LoginScreenController implements Initializable {

public static Stage stage;

public static String acc;

@FXML

private Pane main\_area;

@FXML

private TextField accountno;

@FXML

private PasswordField pin;

@FXML

private void closeApp(MouseEvent event) {

Platform.exit();

System.exit(0);

}

@FXML

private void createAccount(MouseEvent event) throws IOException {

Parent fxml = FXMLLoader.load(getClass().getResource("/CreateAccount/CreateAccount.fxml"));

main\_area.getChildren().removeAll();

main\_area.getChildren().addAll(fxml);

}

@FXML

private void forgotPassword(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/ForgotPassword/ForgotPass.fxml"));

main\_area.getChildren().removeAll();

main\_area.getChildren().addAll(fxml);

}

public void loginAccount(MouseEvent event){

Connection con= null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=? and PIN=?";

ps=con.prepareStatement(sql);

ps.setString(1, accountno.getText());

ps.setString(2, pin.getText());

acc=accountno.getText();

rs = ps.executeQuery();

if(rs.next()){

((Node)event.getSource()).getScene().getWindow().hide();

Parent root=FXMLLoader.load(getClass().getResource("/DashBoard/Dashboard.fxml"));

Scene scene=new Scene(root);

scene.getStylesheets().add(getClass().getResource("/design/design.css" ).toExternalForm());

Stage stage=new Stage();

stage.initStyle(StageStyle.UNDECORATED);

stage.setScene(scene);

stage.show();

this.stage=stage;

}

else

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error is Login Account");

a.setContentText("Your account number or password is wrong. Enter it again!!!");

a.showAndWait();

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

@Override

public void initialize(URL location, ResourceBundle resources) {

}

}

1. **CreateAccountController.java**

package CreateAccount;

import Bank.DbUtil;

import java.io.IOException;

import java.net.URL;

import java.util.ResourceBundle;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.input.MouseEvent;

import Login.Banking;

import java.io.File;

import java.io.FileInputStream;

import java.io.InputStream;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.util.Random;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javafx.application.Platform;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.ComboBox;

import javafx.scene.control.DatePicker;

import javafx.scene.control.TextField;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.stage.FileChooser;

import javafx.stage.FileChooser.ExtensionFilter;

public class CreateAccountController implements Initializable {

private FileChooser filechooser= new FileChooser();

private File file;

private FileInputStream fis;

@FXML

private ImageView pic;

@FXML

private TextField name;

@FXML

private TextField idcardno;

@FXML

private TextField mobileno;

@FXML

private TextField city;

@FXML

private TextField address;

@FXML

private TextField accountno;

@FXML

private TextField pin;

@FXML

private TextField balance;

@FXML

private TextField answer;

@FXML

private DatePicker dob;

@FXML

private ComboBox gender;

@FXML

private ComboBox martialstatus;

@FXML

private ComboBox religion;

@FXML

private ComboBox accountype;

@FXML

private ComboBox questions;

ObservableList<String> list=FXCollections.observableArrayList("Male","Female","Other");

ObservableList<String> list1=FXCollections.observableArrayList("Single","Maried");

ObservableList<String> list2=FXCollections.observableArrayList("Hindu","Christian","Muslim","Punjabi","Other");

ObservableList<String> list3=FXCollections.observableArrayList("Saving","Current");

ObservableList<String> list4=FXCollections.observableArrayList("What is your Pet Name?","What is your Childhood Town?","What is your Nickname?");

public void backtologin(MouseEvent event) throws IOException{

Banking.stage.getScene().setRoot(FXMLLoader.load(getClass().getResource("/Login/LoginScreen.fxml")));

}

public void closeApp(MouseEvent event){

Platform.exit();

System.exit(0);

}

public void setUpPic(MouseEvent event){

filechooser.getExtensionFilters().add(new ExtensionFilter("Image Files","\*png","\*jpg"));

file=filechooser.showOpenDialog(null);

if(file!=null){

Image img=new Image(file.toURI().toString(), 150, 150, true, true);

pic.setImage(img);

pic.setPreserveRatio(true);

}

}

public boolean validateName(){

Pattern p = Pattern.compile("[a-zA-Z]+");

Matcher m = p.matcher(name.getText());

if(m.find() && m.group().equals(name.getText())){

return true;

}else{

Alert a = new Alert(AlertType.ERROR);

a.setTitle("Wrong Name");

a.setHeaderText("Your Name is Wrong");

a.setContentText("Please enter character only in name. Try Again");

a.showAndWait();

return false;

}

}

public boolean validateMobileno(){

Pattern p = Pattern.compile("[0-9]+");

Matcher m = p.matcher(mobileno.getText());

if(m.find() && m.group().equals(mobileno.getText())){

return true;

}else{

Alert a = new Alert(AlertType.ERROR);

a.setTitle("Wrong Mobileno");

a.setHeaderText("Your Mobileno is Wrong");

a.setContentText("Please enter number only in name. Try Again");

a.showAndWait();

return false;

}

}

public boolean validateIdcardno(){

Pattern p = Pattern.compile("[0-9]+");

Matcher m = p.matcher(idcardno.getText());

if(m.find() && m.group().equals(idcardno.getText())){

return true;

}else{

Alert a = new Alert(AlertType.ERROR);

a.setTitle("Wrong Idcardno");

a.setHeaderText("Your Idcardno is Wrong");

a.setContentText("Please enter number only in name. Try Again");

a.showAndWait();

return false;

}

}

public boolean validateBalance(){

Pattern p = Pattern.compile("[0-9]+");

Matcher m = p.matcher(balance.getText());

if(m.find() && m.group().equals(balance.getText())){

return true;

}else{

Alert a = new Alert(AlertType.ERROR);

a.setTitle("Invalod Balance");

a.setHeaderText("Your Balance is Wrong");

a.setContentText("Please enter Cureent Balance. Try Again");

a.showAndWait();

return false;

}

}

public void clearAllFields(){

name.clear();

idcardno.clear();

mobileno.clear();

gender.getSelectionModel().clearSelection();

martialstatus.getSelectionModel().clearSelection();

religion.getSelectionModel().clearSelection();

dob.getEditor().clear();

city.clear();

address.clear();

pin.clear();

accountype.getSelectionModel().clearSelection();

balance.clear();

questions.getSelectionModel().clearSelection();

answer.clear();

Image img = new Image("/Images/150.png");

pic.setImage(img);

accountno.setText(String.valueOf(generateAccountno()));

}

public int generateAccountno(){

Random rand = new Random();

int num = rand.nextInt(899999) + 100000;

return num;

}

public void newAccount(MouseEvent event){

Connection con=null;

PreparedStatement ps = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

if(validateName() && validateMobileno() && validateIdcardno() && validateBalance()){

String sql= "INSERT INTO userdata(Name, ICN, MobileNo, Gender, MartialStatus, Religion, DOB, City, Address, AccountNo, PIN, AccountType, Balance, SecurityQuestion, Answer, ProfilePic) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";

ps=con.prepareStatement(sql);

ps.setString(1, name.getText());

ps.setString(2, idcardno.getText());

ps.setString(3, mobileno.getText());

ps.setString(4, (String) gender.getValue());

ps.setString(5, (String) martialstatus.getValue());

ps.setString(6, (String) religion.getValue());

ps.setString(7, ((TextField)dob.getEditor()).getText());

ps.setString(8, city.getText());

ps.setString(9, address.getText());

ps.setString(10, accountno.getText());

ps.setString(11, pin.getText());

ps.setString(12, (String) accountype.getValue());

ps.setString(13, balance.getText());

ps.setString(14, (String) questions.getValue());

ps.setString(15, answer.getText());

fis=new FileInputStream(file);

ps.setBinaryStream(16, (InputStream)fis, (int)file.length());

int i=ps.executeUpdate();

if(i>0){

Alert a = new Alert(AlertType.INFORMATION);

a.setTitle("Account Created");

a.setHeaderText("Account Created");

a.setContentText("Your account has been created sucessfully. You can now login with your login");

a.showAndWait();

clearAllFields();

}

else

{

Alert a = new Alert(AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Account not Creating");

a.setContentText("Your account is not created. there is some error. Try Again");

a.showAndWait();

}

}

}catch(Exception e)

{

Alert a = new Alert(AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error is Creating Account");

a.setContentText("Your account is not Connected. there is some technical issue"+ e.getMessage());

a.showAndWait();

}

}

@Override

public void initialize(URL url, ResourceBundle rb) {

gender.setItems(list);

martialstatus.setItems(list1);

religion.setItems(list2);

accountype.setItems(list3);

questions.setItems(list4);

accountno.setText(String.valueOf(generateAccountno()));

accountno.setEditable(false);

}

}

1. **ForgotPassword.java**

package ForgotPassword;

import Bank.DbUtil;

import Login.Banking;

import java.io.IOException;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ResourceBundle;

import javafx.application.Platform;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.control.Alert;

import javafx.scene.control.ComboBox;

import javafx.scene.control.TextField;

import javafx.scene.input.MouseEvent;

public class ForgotPassController implements Initializable {

@FXML

private TextField accountno;

@FXML

private ComboBox<String> sq;

@FXML

private TextField ans;

ObservableList<String> list =FXCollections.observableArrayList("What is your Pet Name?","What is your Childhood Town?","What is your Nickname?");

public void backtologin(MouseEvent event) throws IOException{

Banking.stage.getScene().setRoot(FXMLLoader.load(getClass().getResource("/Login/LoginScreen.fxml")));

}

public void closeApp(MouseEvent event){

Platform.exit();

System.exit(0);

}

public void recoverPassword(MouseEvent event){

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=? and SecurityQuestion=? and Answer=?";

ps=con.prepareStatement(sql);

ps.setString(1, accountno.getText());

ps.setString(2, sq.getValue());

ps.setString(3, ans.getText());

rs = ps.executeQuery();

if(rs.next()){

Alert a = new Alert(Alert.AlertType.INFORMATION);

a.setTitle("Password Recovery");

a.setHeaderText("Below is your password");

a.setContentText("Account Number"+ rs.getString("AccountNo")+"\nPIN: "+rs.getString("PIN"));

a.showAndWait();

}

else

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Wrong data");

a.setContentText("Please Try Again!!!");

a.showAndWait();

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in recovery");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

@Override

public void initialize(URL url, ResourceBundle rb) {

sq.setItems(list);

}

}

1. **MainScreenController.java**

package DashBoard;

import Bank.DbUtil;

import Login.LoginScreenController;

import java.io.IOException;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ResourceBundle;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Parent;

import javafx.scene.control.Alert;

import javafx.scene.control.Label;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.Pane;

public class MainScreenController implements Initializable {

@FXML

private Label name;

@FXML

private Label body;

@FXML

private Pane dashboard\_main;

public void setInfo(){

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

if(rs.next()){

name.setText(rs.getString("Name"));

}

else

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error is Login Account");

a.setContentText("Your account number or password is wrong. Enter it again!!!");

a.showAndWait();

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

@FXML

public void history(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/TransactionHistory/TransactionHistory.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@FXML

public void amount(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/TransferAmount/TransferAmount.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@Override

public void initialize(URL url, ResourceBundle rb) {

body.setText("Babbar Rajput Bank is a Banking and Financial service bank owned by yash.\nThe bank was founded in 2020.It is the fast growing private sector bank in India,\nboth in terms of business and its network.");

setInfo();

}

}

1. **DashboardController.java**

package DashBoard;

import Bank.DbUtil;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ResourceBundle;

import javafx.application.Platform;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Alert;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import Login.LoginScreenController;

import de.jensd.fx.glyphs.fontawesome.FontAwesomeIconView;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.scene.layout.Pane;

import javafx.scene.paint.ImagePattern;

import javafx.scene.shape.Circle;

import javafx.scene.text.Text;

import javafx.stage.StageStyle;

public class DashboardController implements Initializable {

private double xOffset = 0;

private double yOffset = 0;

@FXML

private Pane dashboard\_main;

@FXML

private Text name;

@FXML

private FontAwesomeIconView icon;

@FXML

private Circle profilepic;

@FXML

private void closeApp(MouseEvent event) {

Platform.exit();

System.exit(0);

}

@FXML

private void ninimizeApp(MouseEvent event) {

Stage stage=(Stage) icon.getScene().getWindow();

stage.setIconified(true);

}

public void setData(){

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

if(rs.next()){

name.setText(rs.getString("Name"));

InputStream is = rs.getBinaryStream("ProfilePic");

OutputStream os = new FileOutputStream(new File("yash.png"));

byte[] content = new byte[1024];

int size = 0;

while((size = is.read(content)) != -1){

os.write(content, 0, size);

}

os.close();

is.close();

Image img=new Image("file:yash.png", false);

profilepic.setFill(new ImagePattern(img));

}

else

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error is Login Account");

a.setContentText("Your account number or password is wrong. Enter it again!!!");

a.showAndWait();

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

@FXML

public void click(MouseEvent event){

xOffset =event.getSceneX();

yOffset =event.getSceneY();

}

@FXML

public void drag(MouseEvent event){

LoginScreenController.stage.setX(event.getScreenX()-xOffset);

LoginScreenController.stage.setY(event.getScreenY()-yOffset);

}

@FXML

public void accountInformation(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/AccountInfomation/Accountinfo.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@FXML

public void withdraw(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/WithdrawAmount/Withdrawamount.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@FXML

public void transactionHistory(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/TransactionHistory/TransactionHistory.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@FXML

public void deposit(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/Deposit/Deposit.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@FXML

public void changepin(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/Changepin/Changepin.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@FXML

public void transferAmount(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/TransferAmount/TransferAmount.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@FXML

public void mainScreen() throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("MainScreen.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@FXML

public void logout(MouseEvent event) throws IOException{

((Node)event.getSource()).getScene().getWindow().hide();

Parent root=FXMLLoader.load(getClass().getResource("/Login/LoginScreen.fxml"));

Scene scene=new Scene(root);

scene.getStylesheets().add(getClass().getResource("/design/design.css" ).toExternalForm());

Stage stage=new Stage();

stage.initStyle(StageStyle.UNDECORATED);

stage.setScene(scene);

stage.show();

root.setOnMousePressed(new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

xOffset =event.getSceneX();

yOffset =event.getSceneY();

}

});

root.setOnMouseDragged(new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

stage.setX(event.getScreenX()-xOffset);

stage.setY(event.getScreenY()-yOffset);

}

});

}

@Override

public void initialize(URL url, ResourceBundle rb) {

setData();

try

{

mainScreen();

} catch (IOException ex)

{

Logger.getLogger(DashboardController.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

1. **AccountinfoController.java**

package AccountInfomation;

import Bank.DbUtil;

import Login.LoginScreenController;

import java.io.IOException;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ResourceBundle;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Parent;

import javafx.scene.control.Alert;

import javafx.scene.control.Label;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.Pane;

import javafx.scene.text.Text;

public class AccountinfoController implements Initializable {

@FXML

private Text account\_no;

@FXML

private Text sex;

@FXML

private Text account\_type;

@FXML

private Text religion;

@FXML

private Label balance;

@FXML

private Pane dashboard\_main;

public void setInfo(){

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

if(rs.next()){

balance.setText(rs.getString("Balance"));

account\_no.setText(rs.getString("AccountNo"));

sex.setText(rs.getString("Gender"));

account\_type.setText(rs.getString("AccountType"));

religion.setText(rs.getString("Religion"));

}

else

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error is Login Account");

a.setContentText("Your account number or password is wrong. Enter it again!!!");

a.showAndWait();

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

@FXML

public void withdraw(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/WithdrawAmount/Withdrawamount.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@FXML

public void deposit(MouseEvent event) throws IOException{

Parent fxml = FXMLLoader.load(getClass().getResource("/Deposit/Deposit.fxml"));

dashboard\_main.getChildren().removeAll();

dashboard\_main.getChildren().addAll(fxml);

}

@Override

public void initialize(URL url, ResourceBundle rb) {

setInfo();

}

}

1. **WithdrawamountController.java**

package WithdrawAmount;

import Bank.DbUtil;

import Login.LoginScreenController;

import java.net.URL;

import java.sql.Connection;

import java.util.Date;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.time.LocalTime;

import java.time.format.DateTimeFormatter;

import java.util.Calendar;

import java.util.ResourceBundle;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Alert;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

public class WithdrawamountController implements Initializable {

@FXML

private Label account\_no;

@FXML

private Label balance;

@FXML

private TextField amt\_field;

@FXML

private TextField pin\_field;

Calendar cal = Calendar.getInstance();

int year = cal.get(Calendar.YEAR);

int month = cal.get(Calendar.MONTH);

int day = cal.get(Calendar.DAY\_OF\_MONTH);

int hour = cal.get(Calendar.HOUR);

int minute = cal.get(Calendar.MINUTE);

int second = cal.get(Calendar.SECOND);

int daynight = cal.get(Calendar.AM\_PM);

DateFormat dateformat = new SimpleDateFormat("yyyy/mm/dd");

Date d = new Date();

String date =dateformat.format(d);

LocalTime localTime=LocalTime.now();

DateTimeFormatter dt = DateTimeFormatter.ofPattern("hh:mm:ss a");

String time =localTime.format(dt);

public void setInfo(){

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

if(rs.next()){

account\_no.setText(rs.getString("AccountNo"));

balance.setText(rs.getString("Balance"));

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

public void withdrawButton(){

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=? and PIN=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

ps.setString(2, pin\_field.getText());

rs = ps.executeQuery();

if(rs.next()){

int wda=Integer.parseInt(amt\_field.getText());

int ta=Integer.parseInt(balance.getText());

if(wda>ta){

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error is withdraw");

a.setContentText("Your don't have enough balance. Enter it again!!!");

a.showAndWait();

}

else

{

int total=ta-wda;

String sql1="UPDATE userdata SET Balance='"+total+"'WHERE AccountNo='"+LoginScreenController.acc+"'";

ps=con.prepareStatement(sql1);

ps.execute();

String sql2="INSERT INTO withdraw(AccountNo, WithdrawAmount, RemainingAmount, Date, Time) VALUES (?,?,?,?,?)";

ps=con.prepareStatement(sql2);

ps.setString(1, LoginScreenController.acc);

ps.setString(2, String.valueOf(wda));

ps.setString(3, String.valueOf(total));

ps.setString(4, date);

ps.setString(5, time);

int i = ps.executeUpdate();

if(i>0)

{

Alert a = new Alert(Alert.AlertType.INFORMATION);

a.setTitle("Amount Withdraw");

a.setHeaderText("Amount Withdraw Sucessfully");

a.setContentText("Amount"+wda+"has been successfully withdrawn\n"+"Current Balance"+total);

a.showAndWait();

amt\_field.setText("");

pin\_field.setText("");

balance.setText(String.valueOf(total));

}

}

}

else

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error is Login Account");

a.setContentText("Your account number or password is wrong. Enter it again!!!");

a.showAndWait();

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

@Override

public void initialize(URL url, ResourceBundle rb) {

setInfo();

}

}

1. **DepositController.java**

package Deposit;

import Bank.DbUtil;

import Login.LoginScreenController;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.time.LocalTime;

import java.time.format.DateTimeFormatter;

import java.util.Calendar;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Alert;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

public class DepositController implements Initializable {

@FXML

private Label account\_no;

@FXML

private Label balance;

@FXML

private TextField amt\_field;

@FXML

private TextField pin\_field;

Calendar cal = Calendar.getInstance();

int year = cal.get(Calendar.YEAR);

int month = cal.get(Calendar.MONTH);

int day = cal.get(Calendar.DAY\_OF\_MONTH);

int hour = cal.get(Calendar.HOUR);

int minute = cal.get(Calendar.MINUTE);

int second = cal.get(Calendar.SECOND);

int daynight = cal.get(Calendar.AM\_PM);

DateFormat dateformat = new SimpleDateFormat("yyyy/mm/dd");

Date d = new Date();

String date =dateformat.format(d);

LocalTime localTime=LocalTime.now();

DateTimeFormatter dt = DateTimeFormatter.ofPattern("hh:mm:ss a");

String time =localTime.format(dt);

public void setInfo(){

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

if(rs.next()){

account\_no.setText(rs.getString("AccountNo"));

balance.setText(rs.getString("Balance"));

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

public void depositButton(){

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=? and PIN=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

ps.setString(2, pin\_field.getText());

rs = ps.executeQuery();

if(rs.next()){

int da=Integer.parseInt(amt\_field.getText());

int ta=Integer.parseInt(balance.getText());

int total=ta+da;

String sql1="UPDATE userdata SET Balance='"+total+"'WHERE AccountNo='"+LoginScreenController.acc+"'";

ps=con.prepareStatement(sql1);

ps.execute();

String sql2="INSERT INTO deposit(AccountNo, DepositAmount, NewAmount, Date, Time) VALUES (?,?,?,?,?)";

ps=con.prepareStatement(sql2);

ps.setString(1, LoginScreenController.acc);

ps.setString(2, String.valueOf(da));

ps.setString(3, String.valueOf(total));

ps.setString(4, date);

ps.setString(5, time);

int i = ps.executeUpdate();

if(i>0)

{

Alert a = new Alert(Alert.AlertType.INFORMATION);

a.setTitle("Amount Deposit");

a.setHeaderText("Amount Deposit Sucessfully");

a.setContentText("Amount"+da+"has been successfully Deposited\n"+"Current Balance"+total);

a.showAndWait();

amt\_field.setText("");

pin\_field.setText("");

balance.setText(String.valueOf(total));

}

}

else

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in depositing amount");

a.setContentText("You enter wrong pin. Enter it again!!!");

a.showAndWait();

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in depositing");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

@Override

public void initialize(URL url, ResourceBundle rb) {

setInfo();

}

}

1. **TransactionHistoryController.java**

package TransactionHistory;

import Bank.DbUtil;

import Login.LoginScreenController;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ResourceBundle;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Alert;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

public class TransactionHistoryController implements Initializable {

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

@FXML

private TableView<History> withdraw\_table;

@FXML

private TableColumn<History, String> withdraw\_accountno;

@FXML

private TableColumn<History, String> withdraw\_amount;

@FXML

private TableColumn<History, String> withdraw\_remainingamount;

@FXML

private TableColumn<History, String> withdraw\_date;

@FXML

private TableColumn<History, String> withdraw\_time;

@FXML

private TableView<History> deposit\_table;

@FXML

private TableColumn<History, String> deposit\_accountno;

@FXML

private TableColumn<History, String> deposit\_amount;

@FXML

private TableColumn<History, String> deposit\_remainingamount;

@FXML

private TableColumn<History, String> deposit\_date;

@FXML

private TableColumn<History, String> deposit\_time;

@FXML

private TableView<History> transfer\_table;

@FXML

private TableColumn<History, String> transfer\_accountno;

@FXML

private TableColumn<History, String> transfer\_amount;

@FXML

private TableColumn<History, String> transfer\_remainingamount;

@FXML

private TableColumn<History, String> transfer\_date;

@FXML

private TableColumn<History, String> transfer\_time;

@FXML

private TableView<History> receive\_table;

@FXML

private TableColumn<History, String> receive\_accountno;

@FXML

private TableColumn<History, String> receive\_amount;

@FXML

private TableColumn<History, String> receive\_remainingamount;

@FXML

private TableColumn<History, String> receive\_date;

@FXML

private TableColumn<History, String> receive\_time;

public void withdraw(){

withdraw\_accountno.setCellValueFactory(new PropertyValueFactory<History, String>("name"));

withdraw\_amount.setCellValueFactory(new PropertyValueFactory<History, String>("amount"));

withdraw\_remainingamount.setCellValueFactory(new PropertyValueFactory<History, String>("generic"));

withdraw\_date.setCellValueFactory(new PropertyValueFactory<History, String>("date"));

withdraw\_time.setCellValueFactory(new PropertyValueFactory<History, String>("time"));

ObservableList<History> list = FXCollections.observableArrayList();

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM withdraw WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

while(rs.next()){

list.add(new History(rs.getString("AccountNo"), rs.getString("WithdrawAmount"), rs.getString("RemainingAmount"), rs.getString("Date"), rs.getString("Time")));

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Fetching Data");

a.setContentText("There is Error in Fetching Data. Try Again");

a.showAndWait();

}

withdraw\_table.setItems(list);

}

public void deposit(){

deposit\_accountno.setCellValueFactory(new PropertyValueFactory<History, String>("name"));

deposit\_amount.setCellValueFactory(new PropertyValueFactory<History, String>("amount"));

deposit\_remainingamount.setCellValueFactory(new PropertyValueFactory<History, String>("generic"));

deposit\_date.setCellValueFactory(new PropertyValueFactory<History, String>("date"));

deposit\_time.setCellValueFactory(new PropertyValueFactory<History, String>("time"));

ObservableList<History> list = FXCollections.observableArrayList();

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM deposit WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

while(rs.next()){

list.add(new History(rs.getString("AccountNo"), rs.getString("DepositAmount"), rs.getString("NewAmount"), rs.getString("Date"), rs.getString("Time")));

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Fetching Data");

a.setContentText("There is Error in Fetching Data. Try Again");

a.showAndWait();

}

deposit\_table.setItems(list);

}

public void transfer(){

transfer\_accountno.setCellValueFactory(new PropertyValueFactory<History, String>("name"));

transfer\_amount.setCellValueFactory(new PropertyValueFactory<History, String>("amount"));

transfer\_remainingamount.setCellValueFactory(new PropertyValueFactory<History, String>("generic"));

transfer\_date.setCellValueFactory(new PropertyValueFactory<History, String>("date"));

transfer\_time.setCellValueFactory(new PropertyValueFactory<History, String>("time"));

ObservableList<History> list = FXCollections.observableArrayList();

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM transferamount WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

while(rs.next()){

list.add(new History(rs.getString("AccountNo"), rs.getString("Amount"), rs.getString("SendTo"), rs.getString("Date"), rs.getString("Time")));

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Fetching Data");

a.setContentText("There is Error in Fetching Data. Try Again");

a.showAndWait();

}

transfer\_table.setItems(list);

}

public void receive(){

receive\_accountno.setCellValueFactory(new PropertyValueFactory<History, String>("name"));

receive\_amount.setCellValueFactory(new PropertyValueFactory<History, String>("amount"));

receive\_remainingamount.setCellValueFactory(new PropertyValueFactory<History, String>("generic"));

receive\_date.setCellValueFactory(new PropertyValueFactory<History, String>("date"));

receive\_time.setCellValueFactory(new PropertyValueFactory<History, String>("time"));

ObservableList<History> list = FXCollections.observableArrayList();

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM transferamount WHERE SendTo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

while(rs.next()){

list.add(new History(rs.getString("SendTo"), rs.getString("Amount"), rs.getString("AccountNo"), rs.getString("Date"), rs.getString("Time")));

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Fetching Data");

a.setContentText("There is Error in Fetching Data. Try Again");

a.showAndWait();

}

receive\_table.setItems(list);

}

@Override

public void initialize(URL url, ResourceBundle rb) {

withdraw();

deposit();

transfer();

receive();

}

}

1. **TransferAmountController.java**

package TransferAmount;

import Bank.DbUtil;

import Login.LoginScreenController;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.time.LocalTime;

import java.time.format.DateTimeFormatter;

import java.util.Calendar;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Alert;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

public class TransferAmountController implements Initializable {

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

@FXML

private Label account\_no;

@FXML

private Label balance;

@FXML

private TextField account\_no\_field;

@FXML

private TextField amt\_field;

@FXML

private TextField pin\_field;

Calendar cal = Calendar.getInstance();

int year = cal.get(Calendar.YEAR);

int month = cal.get(Calendar.MONTH);

int day = cal.get(Calendar.DAY\_OF\_MONTH);

int hour = cal.get(Calendar.HOUR);

int minute = cal.get(Calendar.MINUTE);

int second = cal.get(Calendar.SECOND);

int daynight = cal.get(Calendar.AM\_PM);

DateFormat dateformat = new SimpleDateFormat("yyyy/MM/dd");

Date d = new Date();

String date =dateformat.format(d);

LocalTime localTime=LocalTime.now();

DateTimeFormatter dt = DateTimeFormatter.ofPattern("hh:mm:ss a");

String time =localTime.format(dt);

public void setInfo(){

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

rs = ps.executeQuery();

if(rs.next()){

account\_no.setText(rs.getString("AccountNo"));

balance.setText(rs.getString("Balance"));

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

public void checkButton(){

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=?";

ps=con.prepareStatement(sql);

ps.setString(1, account\_no\_field.getText());

rs = ps.executeQuery();

if(rs.next()){

Alert a = new Alert(Alert.AlertType.INFORMATION);

a.setTitle("Account Information");

a.setHeaderText("Below is the Information of account.");

a.setContentText("Account No ="+account\_no\_field.getText()+"\n Name="+rs.getString("Name")+"\n Mobile Number ="+rs.getString("MobileNo"));

a.showAndWait();

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

public void transferAmountButton() throws SQLException, ClassNotFoundException{

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=? and PIN=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

ps.setString(2, pin\_field.getText());

rs = ps.executeQuery();

if(rs.next()){

int transfer\_amt = Integer.parseInt(amt\_field.getText());

int ta = Integer.parseInt(balance.getText());

if(transfer\_amt>ta){

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error is Transfer Money");

a.setContentText("Your don't have enough balance. Enter it again!!!");

a.showAndWait();

}

else

{

int total=ta-transfer\_amt;

String sql1="UPDATE userdata SET Balance='"+total+"'WHERE AccountNo='"+LoginScreenController.acc+"'";

ps=con.prepareStatement(sql1);

ps.execute();

String sql2= "SELECT \* FROM userdata WHERE AccountNo=?";

ps=con.prepareStatement(sql2);

ps.setString(1, account\_no\_field.getText());

rs = ps.executeQuery();

if(rs.next()){

int cur=Integer.parseInt(amt\_field.getText());

int prev=Integer.parseInt(rs.getString("Balance"));

int total1=cur+prev;

String sql4="UPDATE userdata SET Balance='"+total1+"'WHERE AccountNo='"+account\_no\_field.getText()+"'";

ps=con.prepareStatement(sql4);

ps.execute();

String sql5="INSERT INTO transferamount(AccountNo, Amount, SendTo, Date, Time) VALUES (?,?,?,?,?)";

ps=con.prepareStatement(sql5);

ps.setString(1, LoginScreenController.acc);

ps.setString(2, String.valueOf(amt\_field.getText()));

ps.setString(3, String.valueOf(account\_no\_field.getText()));

ps.setString(4, date);

ps.setString(5, time);

int i = ps.executeUpdate();

if(i>0)

{

Alert a = new Alert(Alert.AlertType.INFORMATION);

a.setTitle("Amount Transfer");

a.setHeaderText("Amount Transfered Sucessfully");

a.setContentText("Amount"+cur+"has been successfully transfered\n"+"To Account No ="+account\_no\_field.getText());

a.showAndWait();

account\_no\_field.setText("");

amt\_field.setText("");

pin\_field.setText("");

balance.setText(String.valueOf(total));

}

}

else

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in depositing amount");

a.setContentText("You enter wrong pin. Enter it again!!!");

a.showAndWait();

}

}

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in Login");

a.setContentText("There is some error. Try Again"+e.getMessage());

a.showAndWait();

}

}

@Override

public void initialize(URL url, ResourceBundle rb) {

setInfo();

}

}

1. **ChangepinController.java**

package Changepin;

import Bank.DbUtil;

import DashBoard.DashboardController;

import Login.LoginScreenController;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ResourceBundle;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Alert;

import javafx.scene.control.TextField;

import javafx.scene.input.MouseEvent;

public class ChangepinController implements Initializable {

@FXML

private TextField oldpin;

@FXML

private TextField newpin;

@FXML

private TextField confirmpin;

DashboardController d = new DashboardController();

public void changepinButton(MouseEvent event){

Connection con=null;

PreparedStatement ps = null;

ResultSet rs = null;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DbUtil.getConnection();

String sql= "SELECT \* FROM userdata WHERE AccountNo=? and PIN=?";

ps=con.prepareStatement(sql);

ps.setString(1, LoginScreenController.acc);

ps.setString(2, oldpin.getText());

rs = ps.executeQuery();

if(rs.next()){

if(newpin.getText().equals(confirmpin.getText())){

String sql1="UPDATE userdata SET PIN='"+newpin.getText()+"'WHERE AccountNo='"+LoginScreenController.acc+"'";

ps=con.prepareStatement(sql1);

ps.execute();

Alert a = new Alert(Alert.AlertType.INFORMATION);

a.setTitle("PIN Change");

a.setHeaderText("PIN Change Sucessfully");

a.setContentText("your pin has been changed now you have to login again with new PIN.");

a.showAndWait();

oldpin.setText("");

newpin.setText("");

confirmpin.setText("");

d.logout(event);

}

}

else

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in depositing amount");

a.setContentText("You enter wrong pin. Enter it again!!!");

a.showAndWait();

}

}catch(Exception e)

{

Alert a = new Alert(Alert.AlertType.ERROR);

a.setTitle("Error");

a.setHeaderText("Error in depositing");

a.setContentText("There is some error. Try Again");

a.showAndWait();

}

}

@Override

public void initialize(URL url, ResourceBundle rb) {

// TODO

}

}