INDEX

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION OF THE SYSTEM:

Postmate is an application which is created for the use of an automated system all over the country. Post office administration provides depositors who do not have access to banks a safe and convenient method to save money. This project helps in maintaining the transactions that takes place in the postal.

An account can be opened at the post office as it is a secure investment that offers easy liquidation of funds, fully or partially an automated module for the Post Office will help in automating functions of the various departments. It helps in reducing the time spent in record keeping and the work can be carried out effectively. The searching of records in future will also become easy. Our project postal account commencement helps in ensuring smooth flow of transactions in the post thus helping the users as well as customers. Process from creating an account till different services provided by post office is recorded in this system.

1.1.1 PROJECT TITLE:

Postmate

1.1.2 PROJECT CATEGORY:

Web Application and RDBMS

1.1.3 OVERVIEW:

Postmate is an application which is created for the use of an automated system all over the country. Post office administration provides depositors who do not have access to banks a safe and convenient method to save money. This project helps in maintaining the transactions that takes place in the postal.

An account can be opened at the post office as it is a secure investment that offers easy liquidation of funds, fully or partially an automated module for the Post Office will help in automating functions of the various departments. It helps in reducing the time spent in record keeping and the work can be carried out effectively. The searching of records in future will also become easy. Our project postal account commencement helps in ensuring smooth flow of transactions in the post thus helping the users as well as customers. Process from creating an account till different services provided by post office is recorded in this system.

1.2 BACKGROUND:

1.2.1 INTRODUCTION OF THE COMPANY:

Accolade Tech Solutions is a well-known Website and Mobile App Development Company in Mangalore. We are well known for our innovative approach to delivering business values and its best results. Promising commitment to long-term sustainability. We specialise in creating dynamic, user-friendly interfaces and also secure web applications in React, Node, PHP, and WordPress. We have Android and iOS experts on staff who can create robust mobile applications. We also configure and manage cloud platforms such as AWS, Digital Ocean, and others. We are highly skilled in the development of ecommerce sites, both multivendor and single-vendor. In addition, we have developed OTT (over-the-top) applications for the News Media industry. We have R & D team who works in Machine Learning, AI, Deep Learning & Block chain Technology.

Company goals to thrive for over the years:

- Making products that is compatible with any device.
- Make us an empire of best solution in town and more.
- Facilitate the creation of proficient delegates.

1.2.2 BRIEF NOTE ON EXISTING SYSTEM:

Considering the present situation of the Post Department where the officers need to store the postal information about the applications, transactions and other information on one platform. So our team came up with the idea to handle all the postal activities like applications, transaction details, provident fund, insurance etc.… under one platform.

1.3 OBJECTIVES OF THE SYSTEM:

- Automation of administrative tasks.
- Centralized information management.
- Case and investigation management.
- Resource management.
- Integration and Scalability.

1.4 SCOPE OF THE SYSTEM:

Our system provides opportunities for public to register to the post office to enjoy the financial benefits. Here the public can register to the system and the admin will approve or reject the application. If the application is approved then the user can enjoy the features. It is developed for the ease of public. The public can register using their email and other details like name, phone number and password. After entering to the system they can update the details by selecting the city, branch and by entering the state, door number, pin-code, photo etc.…. and it will be stored in the MySQL Database. The Admin and the staff can manage the transactions. The admin and the staff can view the details of registered users.

1.5 STRUCTURE OF THE SYSTEM:

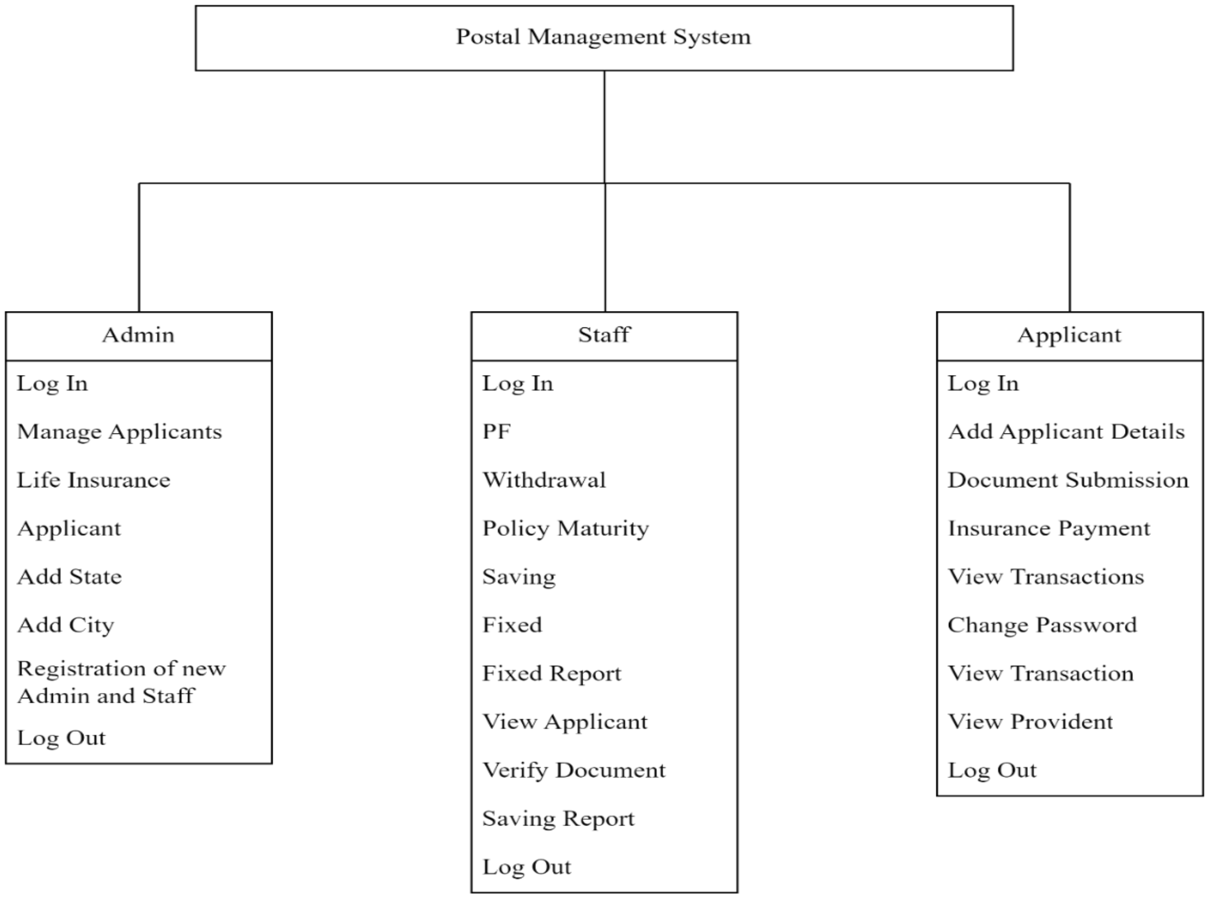


Figure 1.1 Structure of the System

1.6 SYSTEM ARCHITECTURE:

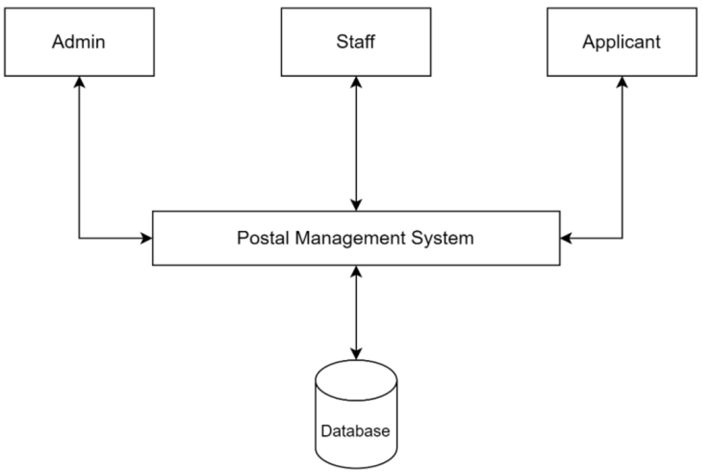


Figure 1.2 System Architecture

1.7 END USERS:

- Admin
- Staff
- Applicant

1.8 SOFTWARE / HARDWARE USED FOR THE DEVELOPMENT:

- **Language:** C# (C Sharp), ASP.Net, HTML, CSS
- **Web Component:** Microsoft Visual studio 2010
- **Web Server:** IIS 7.0
- **Data Base (Backend):** MySQL

1.9 SOFTWARE / HARDWARE REQUIRED FOR THE IMPLEMENTATION:

Hardware Requirements:

- RAM : 1 GB or above
- Hard Disk : 10 GB or above
- Processor : 2.4 GHZ or above

Software Requirements:

- Front end: Microsoft Visual studio 2010
- Back end: My SQL 5.0
- Languages: C#.Net, Asp.Net, HTML, CSS
- Frame work: .Net Frame work 4.0
- Server: IIS 7.0

CHAPTER-2

SOFTWARE REQUIREMENTS SPECIFICATION

2.1 INTRODUCTION:

A Software Requirements Specification (SRS) is a comprehensive document that serves as a foundation for software development projects. It defines the detailed requirements and expectations of the software system to be developed. The SRS document acts as a communication bridge between the stakeholders (clients, users, developers, testers, etc.) and the development team, ensuring a clear understanding of the software's functionalities, features, and constraints.

The purpose of an SRS is to provide a complete and unambiguous description of the software system's behaviour, interfaces, and performance requirements. It serves as a contract between the stakeholders and the development team, guiding the development process and ensuring that the resulting software meets the specified needs and objectives.

2.2 OVERALL DESCRIPTION:

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce its basic functionality of it. It will also describe what type of users will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

2.2.1 PRODUCT PERSPECTIVE:

Product perspective is the relationship of product of other product defining if the product is independent or is a part of large product.

The architecture which is used in the project is 3 tier architecture with presentation layer, application layer, product layer.

2.2.2 PRODUCT FUNCTIONS:

- Master
- Branch
- Applicant Master
- State

        - City
    - Document Master
    - Withdrawal form
    - Savings Bank
    - Fixed Deposit
    - Provident fund account
    - Transaction
        - Postal Life Insurance
        - Insurance Payment
    - Reports
        - Applicant Report
        - Documents
        - Linked Applications Report
    - User Management
        - Login
        - User Creation
        - Change Password
        - Forget Password

2.2.3 USER CHARACTERISTICS:

Postal Management System is designed with the intension to provide easy to use.

This system has 3 levels of users.

- **Admin:**
    - She/he manages the system.
    - She/he upload the branch details.
    - They can add state and city.
    - She/he provide application and document.
    - She/he can provide the required agent information.
- **Staff:**
    - She/he link a application.
    - She/he can manage minor application.

- ❖ She/he can approve the account.
- ❖ They can also reject the account.
- ❖ She/he add Standardized Agency System (SAS Agent) information.
- ❖ They can upload a Public Provident Fund (PPF Agent) information.
- ❖ She/he can upload Postal Life Insurance (PLI).
- ❖ They can manage Insurance payment.
- ❖ They can add maturity of the policy.

➢ **Applicant:**

- ❖ They can apply to create new account in post office.
- ❖ They can submit documents for verification.
- ❖ Insurance payments can also be done
- ❖ Transactions done by the applicant can be viewed.
- ❖ Viewing of Insurance and provident fund details can also be possible.

2.2.4 GENERAL CONSTRAINTS:

- Requires all the mandatory fields to filled with the proper information.

2.2.5 ASSUMPTIONS:

- User should be familiar with Basic computer knowledge.
- The system is completely dependent on internet connection.
- The Information provided by the user is assumed to be genuine.

2.3 SPECIAL REQUIREMENTS:

This section describes all the details that the system developer needs to know for designing and developing this system, these requirements can be organized by the modes of operation user class object, features, and functional hierarchies.

Software Requirements:

- Operating System: Windows
- Text Editor: VS code
- Language: ASP.Net
- Server: IIS 7.0
- User Interface: HTML, CSS
- Database: MySQL

• Browser: Chrome, Mozilla Firefox, or any other browsing applications

Hardware Requirements:

- Processor: AMD RYZEN 3 or above Or Intel I3 or above
- Processor Speed: 2GHz
- RAM: 4GB
- Hard Disk: Minimum 40 GB

2.4 FUNCTIONAL REQUIREMENTS:

2.4.1 LOGIN MODULE: This module is used to add users.

• Input: username, password.
• Output: A new user is created.

• Processing: If user already exist appropriate message is displayed.

2.4.2 REGISTRATION DETAILS: This module contains user can register.

• Input: user id, user name, password, change password, user type, phone, email.

• Output user can successfully registered will be displayed.

• Processing: if any field is not filled it throws an error.

2.4.3 CHANGE PASSWORD DETAILS: If the user wishes to change the password, they can change it using the change password option.

• Input: current password, new password, change password.

• Output: Change password details will be displayed

• Processing: Redirects to respective page.

2.4.4 FORGET PASSWORD: If user forgets his password he can recover his password in this module.

• Input: Username

• Output: Your new password

• Processing: If any filed is not filled then it throws an error.

2.4.5 BRANCH DETAILS: The post office will be situated in different branches. The details of different branches will be stored.

• Input: post office name, pin code, office type, telephone no.

• Output: details of the branch information will be displayed.

• Processing: If there is any pin code problem error message is displayed.

2.4.6 APPLICANT MASTER DETAILS: Initially if the person needs to open an account he/she has to fill the application form given by the post office where he will be asked to give his personal details.

• Input: applicant name, account no, pin code, phone no, email.

• Output: applicant details will be displayed

• Processing: If the entered information is correct then it is successfully added.

2.4.7 SAVING BANK: People can open the account for the savings account. It has certain criteria's like rate of interest is 4% per annum.

• Input: customer name, address

• Output: Account will be created.

• Processing: Savings account will be created.

2.4.8 Fixed deposit: People can deposit money for a higher rate of interest than savings account.

• Input: customer name, amount

• Output: Account will be created.

• Processing: Authorized page will be displayed.

2.4.9 PROVIDENT FUND ACCOUNT: An individual can open account with INR 100 but has to deposit minimum of INR 500 in a financial year and maximum INR 150000.

• Input: customer name, details, amount.

• Output: Amount will be displayed.

• Processing: Provident fund will be increasing in a financial year.

2.4.10 WITHDRAWAL FORM: The user needs to fill the form in order to withdraw the money deposited.

• Input: name, pan card details, Aadhaar number

• Output: Form will be submitted

• Processing: Once verified he/she can withdraw the amount.

2.4.11 STATE DETAILS: The name and detail s of the state.

• Input: state name, state id.

• Output: displays the contact details of the state.

2.4.12 CITY DETAILS: The name and details of the city

• Input: city name, city id, state id.

• Output: Display the details of the state.

2.4.12 DOCUMENT MASTER DETAILS: If the new account has to be opened then the person needs to submit few of his documents like Adhaar, PAN Etc.

• Input: Adhaar card, phone no, photo, email, pan.

• Output: Document information will be displayed.

• Processing: If the entered information is correct then it is successfully added.

2.4.13 POSTAL LIFE INSURANCE: Policyholder can choose to convert the plan into any other endowment plan. This insurance is a 50 lakh policy.

• Input: name, age, policy name, Adhaar, pan.

• Output: Policy will be opened.

2.4.14 INSURANCE PAYMENT: The amount money that an individual or business must pay for an insurance policy. Like a EMI.

• Input: policy no, name, address, account no.

• Output: Insurance document will be added.

2.4.15 MATURITY: An insurance policy is mature when it has full length of the policy period / years which usually spans 5 – 15 years.

• Input: name, address, policy no, account no, pin code.

• Output: Details of the maturity information will be displayed.

2.5 DESIGN CONSTRAINTS:

- Message field to be entered with text in Message format itself
- Requires to specify the information for all the mandatory fields The application shall have a relational database.
- The application shall be implemented using MY SQL 5.0, and Microsoft Visual studio 2010
- The application shall display error messages to the user when an error is detected.

2.6 SYSTEM ATTRIBUTES:

This section of the SRS describes the applications attributes and properties.

- ❖ **Availability:** Postal Management system shall be available in internet 24x7 and capable of supporting a multiple login.
- ❖ **Security:** Postal management System shall be managed by the Admin and Staff via predetermined roles.
- ❖ **Maintainability:** During maintenance stage, the SRS can be referred for the validation.
- ❖ **Portability:** Since it is a system, it is portable.
- ❖ **Timeliness:** The system carries out all the operations with consumptions of very less time.

CHAPTER 3

SYSTEM DESIGN

3.1 INTRODUCTION:

System design is the process of creating a blueprint or plan for building a complex software or hardware system. It involves defining the structure, components, interfaces, and behaviours of the system to meet specific requirements. System design encompasses both high-level architectural decisions and detailed low-level design choices.

The primary goal of system design is to ensure that the system is reliable, scalable, maintainable, and efficient. It involves breaking down the system into smaller components or modules and defining their relationships and interactions. Each module performs a specific function and has well-defined inputs, outputs, and interfaces.

During the system design phase, various considerations are taken into account, such as system requirements, performance objectives, security concerns, user experience, and integration with other systems. Designers may use various techniques, such as flowcharts, diagrams, data models, and algorithms, to visualize and document the system's structure and behaviour.

3.2 ASSUMPTIONS AND CONSTRAINTS:

3.2.1 ASSUMPTIONS:

- User should be familiar with Basic computer knowledge.
- The system is completely dependent on internet connection.
- The Information provided by the user is assumed to be genuine.

3.2.2 CONSTRAINTS:

- Requires all the mandatory fields to filled with the proper information.
- The tables of the database are designed as normalized table.
- Database is not shared.

3.3 FUNCTION DECOMPOSITION:

Function decomposition is a technique or method for breaking down a complex system, process, or problem into smaller and simpler parts. It involves defining the overall function and then dividing it into sub-functions based on the inputs and outputs of each part. The purpose of function decomposition is to gain insight, compress representation, align with goals, or design solutions.

3.3.1 System Software Architecture:

Software System Architecture is the architecture that explains the basic structure of software elements and their arrangement. The relations and the links between the elements are explained along with the structure.

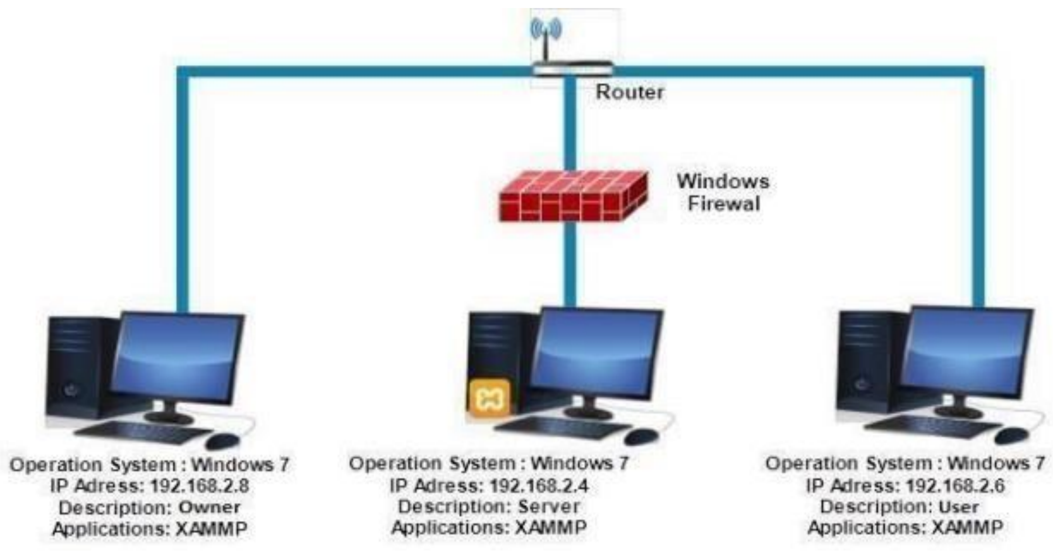


Figure 3.1 System Software Architecture

3.3.1 SYSTEM TECHNICAL ARCHITECTURE:

Technical architecture includes the major components of the system, their relationships, and the contracts that define the interactions between the components. The goal of technical architects is to achieve all the business needs with an application that is optimized for both performance and security.

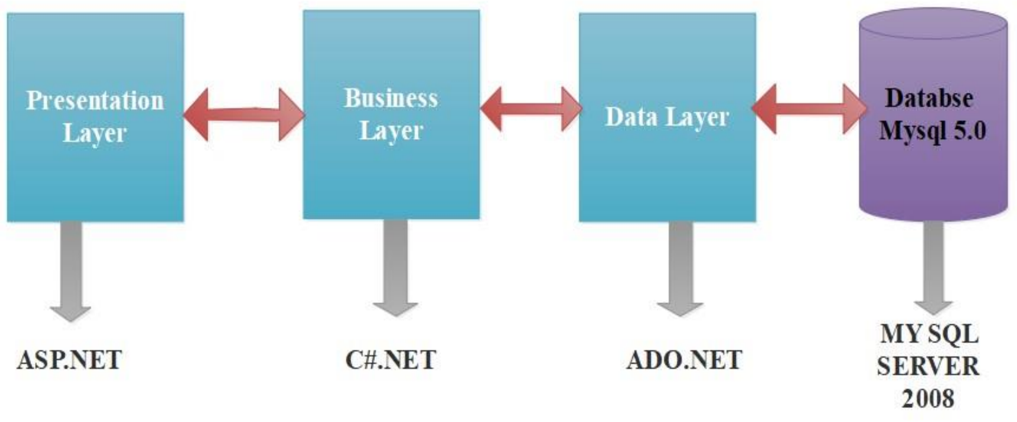


Figure 3.2 System Technical Architecture

3.3.2 SYSTEM HARDWARE ARCHITECTURE:

Hardware architecture refers to the design of the physical components of a computer system. It includes the selection of hardware components and their interconnections. The hardware architecture is designed to meet the requirements of the software that will run on it. The hardware architecture is also designed to be scalable and flexible so that it can be easily upgraded or expanded as needed.

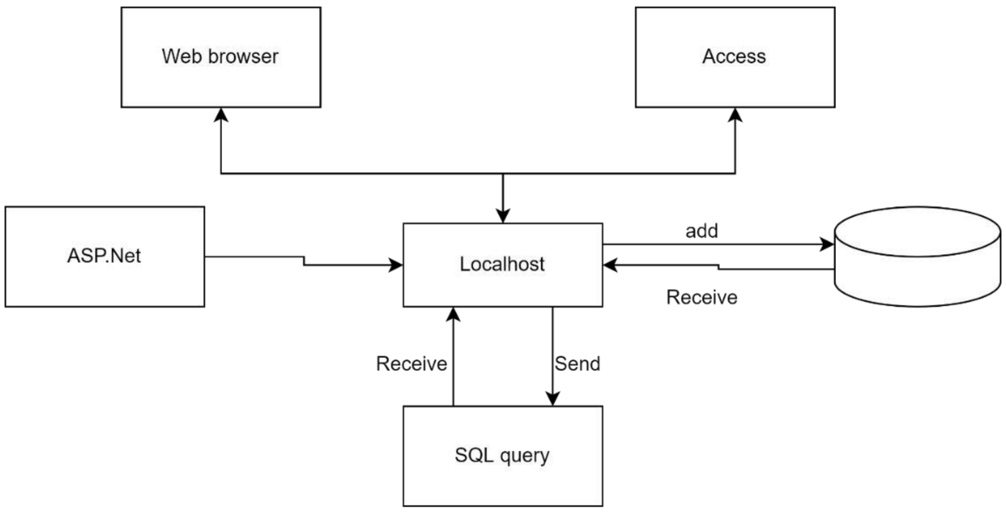


Figure 3.3 System Hardware Architecture

3.4 DESCRIPTION OF PROGRAMS:

3.4.1 CONTEXT FLOW DIAGRAM (CFD):

A Context Flow Diagram, also known as a Context Diagram or Level 0 Diagram, is a visual representation of the high-level interaction between a system and its external entities. It provides an overview of how data or information flows into and out of the system, without going into the details of internal processes or subsystems.

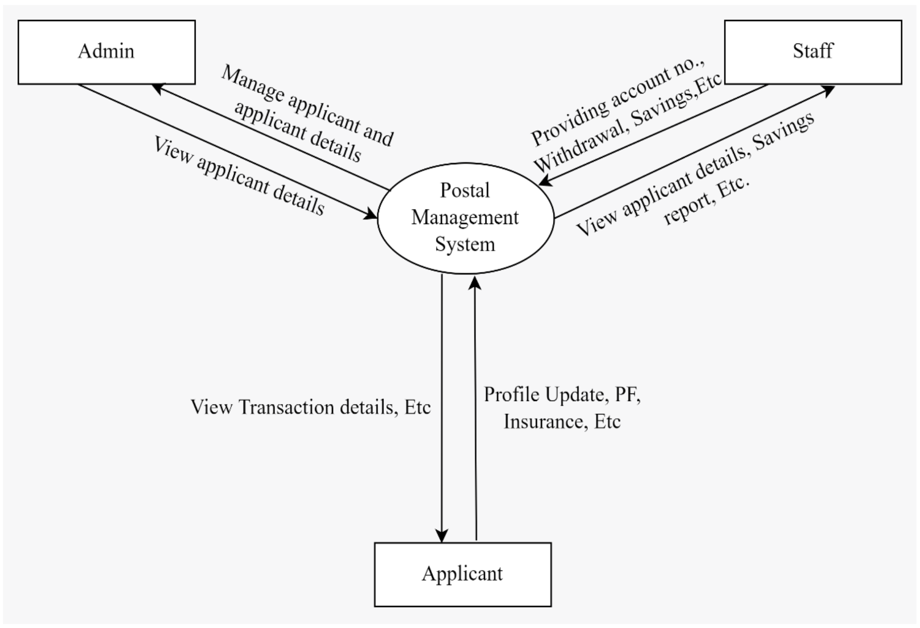

Admin
Staff
Manage applicant and applicant details
View applicant details
Providing account no., Withdrawal, Savings,Etc
View applicant details, Savings report, Etc.
Postal Management System
View Transaction details, Etc
Profile Update, PF, Insurance, Etc
Applicant

Figure 3.4 Context Flow Diagram

3.4.2 DATA FLOW DIAGRAMS (DFDs - LEVEL 1, LEVEL 2):

DFD stands for Data Flow Diagram. It is a graphical representation of the flow of data through an information system. It is used to describe the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

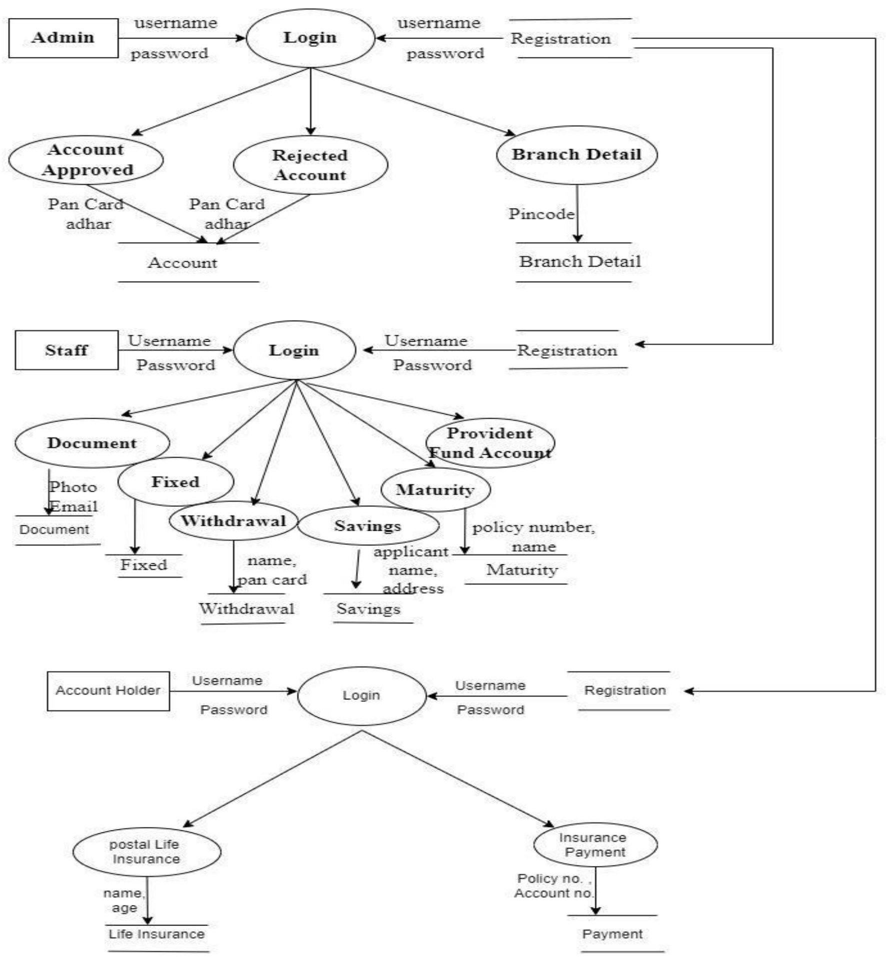
Rules Regarding DFD Construction:

- Data cannot flow between two entities.
- Data flow must be from entity to a process or a process to an entity.
- There can be multiple data flows between one entity and a process.
- A process can have multiple inputs and outputs.
- A data store can have multiple inputs and outputs.
- A data store cannot be connected directly to another data store.

DFD Symbols:

DIAGRAM	DESCRIPTION
▭	Represents Source or destination of data
⬭	Represents a process that transforms Incoming data into Outgoing flows
→	Represents data flow
═	Represents data stores

DFD LEVEL 1:



DFD LEVEL 2.1: ADMIN

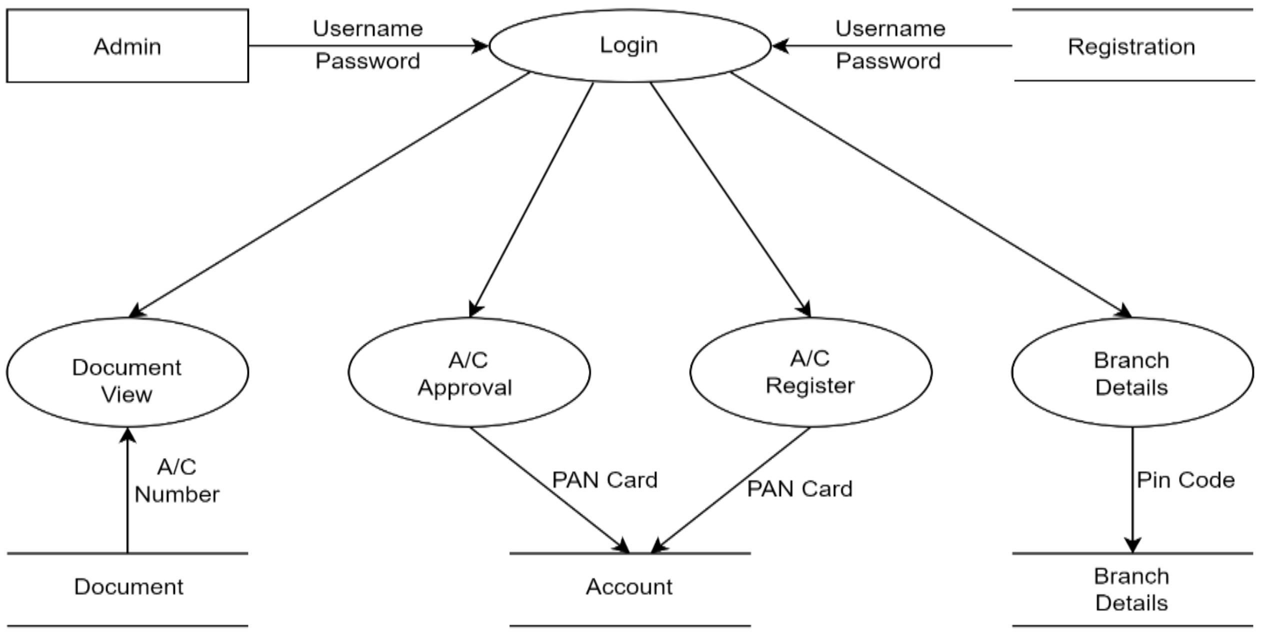


Figure 3.6 DFD Level 2.1

DFD LEVEL 2.2: STAFF

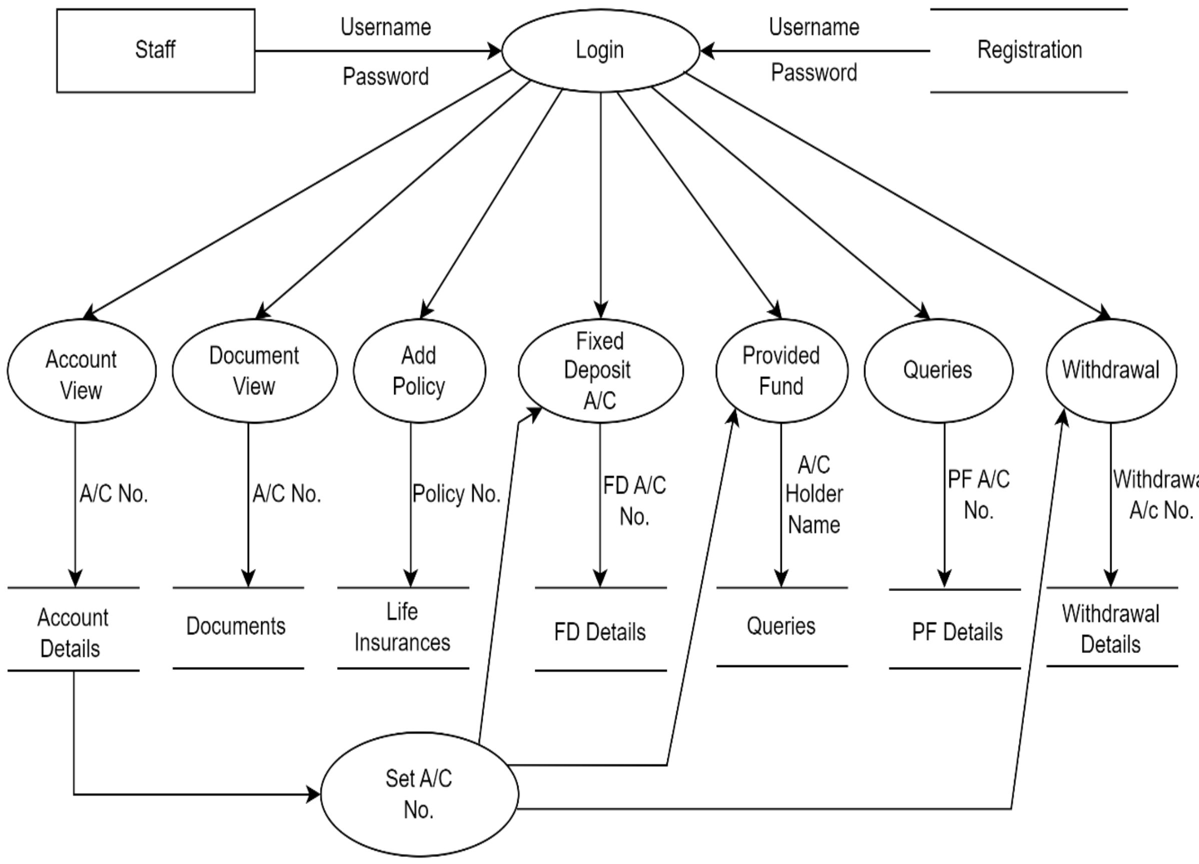


Figure 3.7 DFD Level 2.2

DFD LEVEL 2.3: APPLICANT

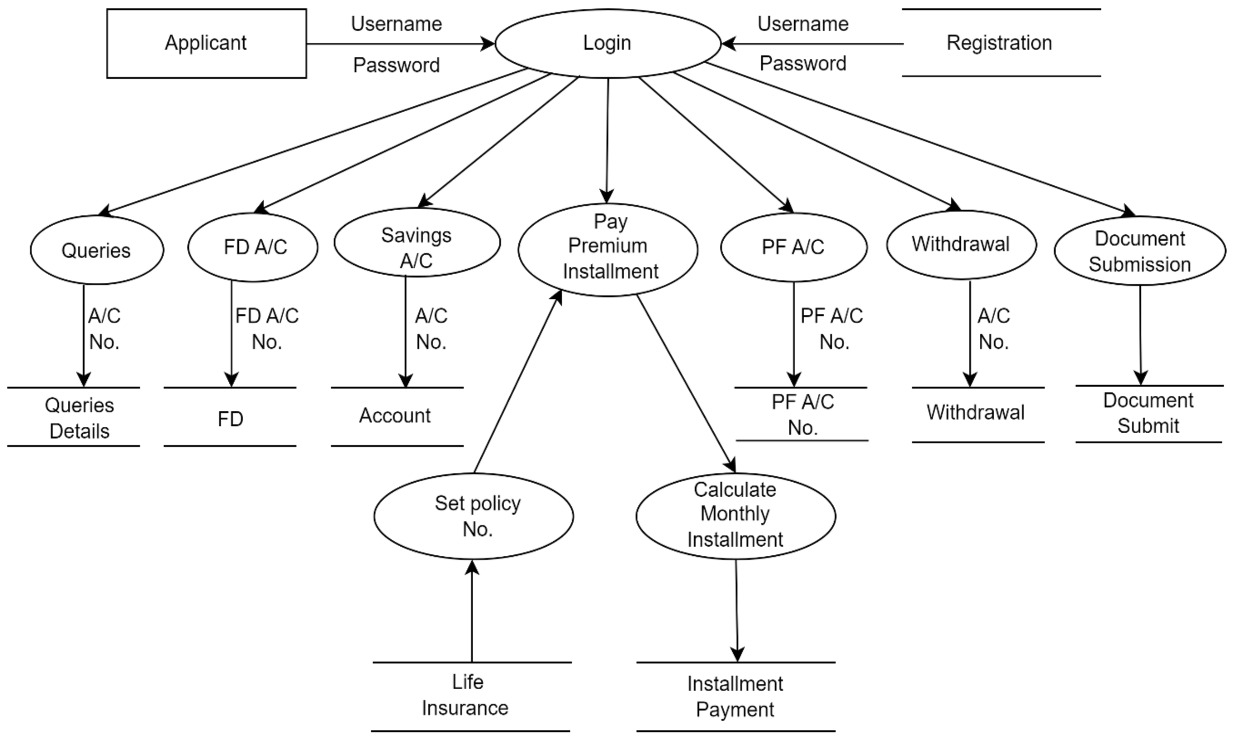


Figure 3.8 DFD Level 2.3

3.5 DESCRIPTION OF COMPONENTS:

3.5.1 FUNCTIONAL COMPONENT 1:

2.4.1 LOGIN MODULE:

This module is used to add users.

• Input: username, password.

• Output: A new user is created.

• Processing: If user already exist appropriate message is displayed.

2.4.2 REGISTRATION DETAILS:

This module contains user can register.

• Input: user id, user name, password, change password, user type, phone, email.

• Output user can successfully registered will be displayed.

• Processing: if any field is not filled it throws an error.

2.4.3 CHANGE PASSWORD DETAILS:

If the user wishes to change the password, they can change it using the change password option.

• Input: current password, new password, change password.

• Output: Change password details will be displayed

• Processing: Redirects to respective page.

2.4.4 FORGET PASSWORD:

If user forgets his password he can recover his password in this module.

• Input: Username

• Output: Your new password

• Processing: If any filed is not filled then it throws an error.

2.4.5 BRANCH DETAILS:

The post office will be situated in different branches. The details of different branches will be stored.

• Input: post office name, pin code, office type, telephone no.

• Output: details of the branch information will be displayed.

• Processing: If there is any pin code problem error message is displayed.

2.4.6 APPLICANT MASTER DETAILS:

Initially if the person needs to open an account he/she has to fill the application form given by the post office where he will be asked to give his personal details.

• Input: applicant name, account no, pin code, phone no, email.

• Output: applicant details will be displayed

• Processing: If the entered information is correct then it is successfully added.

2.4.7 SAVING BANK:

People can open the account for the savings account. It has certain criterias like rate of interest is 4% per annum.

• Input: customer name, address

• Output: Account will be created.

• Processing: Savings account will be created.

2.4.8 FIXED DEPOSIT:

People can deposit money for a higher rate of interest than savings account.

• Input: customer name, amount

• Output: Account will be created.

• Processing: Authorized page will be displayed.

2.4.9 PROVIDENT FUND ACCOUNT:

An individual can open account with INR 100 but has to deposit minimum of INR 500 in a financial year and maximum INR 150000.

• Input: customer name, details, amount.

• Output: Amount will be displayed.

• Processing: Provident fund will be increasing in a financial year.

2.4.10 WITHDRAWAL FORM:

The user needs to fill the form in order to withdraw the money deposited.

• Input: name, pan card details, Aadhaar number

• Output: Form will be submitted

• Processing: Once verified he/she can withdraw the amount.

2.4.11 STATE DETAILS:

The name and detail s of the state.

• Input: state name, state id.

• Output: displays the contact details of the state.

2.4.12 CITY DETAILS:

The name and details of the city

• Input: city name, city id, state id.

• Output: Display the details of the state.

2.4.12 DOCUMENT MASTER DETAILS:

If the new account has to be opened then the person needs to submit few of his documents like Adhaar, PAN Etc.

• Input: Adhaar card, phone no, photo, email, pan.

• Output: Document information will be displayed.

• Processing: If the entered information is correct then it is successfully added.

2.4.13 POSTAL LIFE INSURANCE:

Policyholder can choose to convert the plan into any other endowment plan. This insurance is a 50-lakh policy.

• Input: name, age, policy name, Adhaar, pan card.

• Output: Policy will be opened.

2.4.14 INSURANCE PAYMENT:

The amount money that an individual or business must pay for an insurance policy. Like a EMI.

• Input: policy no, name, address, account no.

• Output: Insurance document will be added.

2.4.15 MATURITY:

An insurance policy is mature when it has full length of the policy period / years which usually spans 5 – 15 years.

• Input: name, address, policy no, account no, pin code.

• Output: Details of the maturity information will be displayed.

• Processing: If user already exist appropriate message is displayed.

CHAPTER 4

DATABASE DESIGN

4.1 INTRODUCTION:

Database is an organized collection of data, generally stored and accessed electronically through computer systems. Database design is a collection of tasks or processes that enhance the designing, development, implementation, and maintenance data management system. The main objective of database design is to produce physical and logical design models of the proposed data-based system. The logical model is primarily concentrated on the requirement of the data. Physical data base design models include a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as DBMS.

4.2 PURPOSE AND SCOPE:

The ultimate purpose of a database management system is to store and transform data into information to support making decisions. The physical database the collection of files that contain the data. The database engine: the software that makes it possible to access and modify the contents of the database.

4.2.1 PURPOSE:

This database requirement specification describes the function and performance requirements by the Postal Management. This database stores

- Admin, Staff and User's login details.
- Admin, Staff and User's details.
- Branch, State and City details.
- Transaction details of users.
- User's Registration details.

4.2.2 SCOPE:

The scope of a database refers to the range of data that it contains and the types of operations that can be performed on that data. The scope of a database is determined by the needs of the organization that uses it. For example, a database used by a hospital might contain patient records, medical histories, and test results. The scope of this database would be limited to medical information and would not include financial or other types of data. The proposed system provides the following features:

- Multiple Admins and Staffs can log in to the system.
- Multiple users can register to the system.
- The Admin Approve or reject the registration.
- The approved user can enjoy the features.

4.3 DATABASE IDENTIFICATION:

- A Primary key is a special relational database table column designed to uniquely identify each table record.
- A FORIEGN KEY is a column that is used to establish and enforce a link between the two tables to control the data that can be stored in a FORIEGN KEY table.
- The database consists of tables, each of which has columns and rows. Each row is a dataset that applies to a single item, and each column contains characteristics that describe the rows. In the database, these columns are called as ‘Attribute’.
- Primary key and FORIEGN KEY are defined with same name.
- Make a separate table for each set of related attributes, and give each table a PRIMARY KEY.
- If an attribute depends on only part of a multi-valued key, remove it to a separate table.
- If attributes do not contribute to a description of the key, remove them to a separate table.

4.4 SCHEMA INFORMATION:

A database schema represents the logical configuration of all or part of a relational database. it can exist both as a visual representation and as asset of formulas known as integrity constraints that govern a database. These formulas are expressed in a data definition language such as SQL.
A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.
A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema Diagrams.
It’s the database designers who design the schema to help programmers understand the database and make it useful.

4.5 TABLE DEFINITION:

4.5.1 REGISTRATION:

Reg_id	User_name	Password	User_type	Email	Phone

4.5.2 APPLICANT:

Applicant _id	Account _id	Name	First name	Middle name	Last name	Gender	Flat no	Street	Post_office_name	State	City	Pincode	Phone no	Email

4.5.3 SAVING:

Saving_id	Applicant_d	Date	Amount	Cheque no

4.5.4 WITHDRAWAL:

Withdrawal_id	Applicant_id	Name	Date	Account_no	Amount	Deposited by

4.5.5 FIXED:

Fixed_id	Applicant_id	Name	Account no	Date	Amount	Cheque no	Deposited by

4.5.6 PROVIDENT FUND ACCOUNT:

Provident_id	Applicant_id	Name	Date	Amount	Account no	Deposited by

4.5.7 BRANCH:

Branch_id	Branch_name

4.5.8 DOCUMENT:

Document_id	Applicant_id	File 1	File 2	File 3

4.5.10 INSURANCE PAYMENT:

Payment_id	Applicant_id	Policy_no	Date	Amount	Post_office_name	Status

4.5.11 PROVIDENT FUND ACCOUNT:

Provident_id	Applicant_id	Name	Date	Amount	Account no	Deposited by

4.5.12 STATE:

State_id	State Name

4.5.13 CITY:

City_id	State_id	City_name

4.6 PHYSICAL DESIGN:

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is inputted into a system, how it is verified. Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer’s physical design involves input via keyboard, processing within the CPU, and output via monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc.

4.7 DATA DICTIONARY

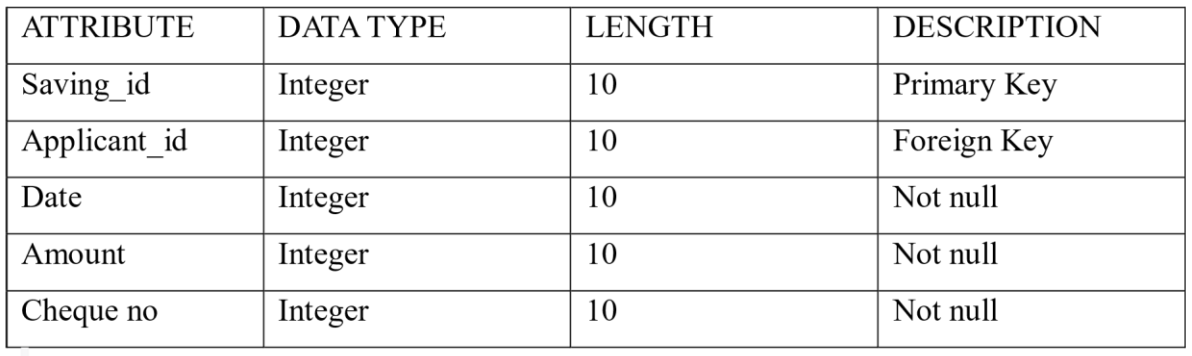
4.7.1 Registration:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Reg_id	Integer	10	Primary Key
User_name	Varchar	45	Foreign Key
Password	Varchar	45	Not null
User_type	Varchar	45	Not null
Email	Varchar	50	Not null
Phone	Integer	10	Not null

4.7.2 Applicant:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Applicant_id	Integer	10	Primary Key
Account_id	Integer	10	Primary Key
Name	Varchar	45	Not null
First name	Varchar	45	Not null
Middle name	Varchar	45	Not null
Last name	Varchar	45	Not null
Gender	Varchar	45	Not null
Flat no	Integer	10	Not null
Street	Varchar	45	Not null
Post_office_name	Varchar	45	Not null
State	Varchar	45	Not null
City	Varchar	45	Not null
Pincode	Integer	10	Not null
Phone no	Integer	10	Not null
Email	Varchar	45	Not null

4.7.3 Saving:



ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Saving_id	Integer	10	Primary Key
Applicant_id	Integer	10	Foreign Key
Date	Integer	10	Not null
Amount	Integer	10	Not null
Cheque no	Integer	10	Not null

4.7.4 Withdrawal:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Withdrawal_id	Integer	10	Primary Key
Applicant_id	Integer	10	Foreign Key
Name	Varchar	45	Not null
Date	Integer	10	Not null
Account_no	Integer	10	Not null
Amount	Integer	10	Not null
Deposited by	Varchar	45	Not null

4.7.5 Fixed:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Fixed_id	Integer	10	Primary Key
Applicant_id	Integer	10	Foreign Key
Name	Varchar	45	Not null
Account no	Integer	10	Not null
Date	Integer	10	Not null
Amount	Integer	10	Not null
Cheque no	Integer	10	Not null
Deposited by	Varchar	45	Not null

4.7.6 Provident fund account:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Provident_id	Integer	10	Primary Key
Applicant_id	Integer	10	Foreign Key
Name	Varchar	45	Not null
Date	Integer	10	Not null
Amount	Integer	10	Not null
Account no	Integer	10	Not null
Deposited by	Varchar	45	Not null

4.7.7 Branch:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Branch _id	Integer	10	Primary Key
Branch_name	Varchar	45	Not null

4.7.8 Document:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Document_id	Integer	10	Primary Key
Applicant_id	Integer	10	Foreign Key
File 1	Varchar	45	Not null
File 2	Varchar	45	Not null
File 3	Varchar	45	Not null

4.7.9 Postal Life Insurance:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Insurance_id	Integer	10	Primary Key
Applicant_id	Integer	10	Foreign Key
Proposal_No	Varchar	45	Not Null
Date_of_Birth	Varchar	30	Not Null
Amount_deposit	Varchar	20	Not Null
Post_office	Varchar	25	Not Null
Policy_no	Varchar	15	Not Null
Deposit	Varchar	10	Not Null
Interest	Varchar	10	Not Null

4.7.10 Insurance Payment:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Payment_id	Integer	10	Primary Key
Applicant_id	Integer	10	Foreign Key
Policy_no	Varchar	45	Not null
Date	Varchar	30	Not null
Amount	Varchar	20	Not null
Post_office_name	Varchar	25	Not null
Status	Varchar	15	Not null

4.7.11 Provident fund account:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
Provident_id	Integer	10	Primary Key
Applicant_id	Integer	10	Foreign Key
Name	Varchar	45	Not null
Date	Integer	10	Not null
Amount	Integer	10	Not null
Account no	Integer	10	Not null
Deposited by	Varchar	45	Not null

4.7.12 State:

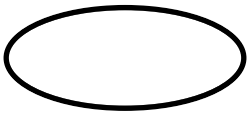
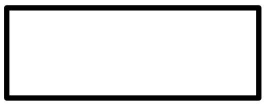
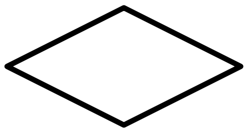
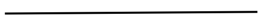
ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
State_id	Integer	10	Primary Key
State Name	Varchar	45	Not null

4.7.13 City:

ATTRIBUTE	DATA TYPE	LENGTH	DESCRIPTION
City_id	Integer	10	Primary Key
State_id	Integer	10	Foreign Key
City Name	Varchar	45	Not null

4.8 ER DIAGRAM:

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

Name	**Symbol**	**Description**
Oval		Shows different attributes.
Rectangle		Shows entity set.
Diamond		Shows relationship among entity set.
Line		Links entity set to attributes and entity set to relationship.

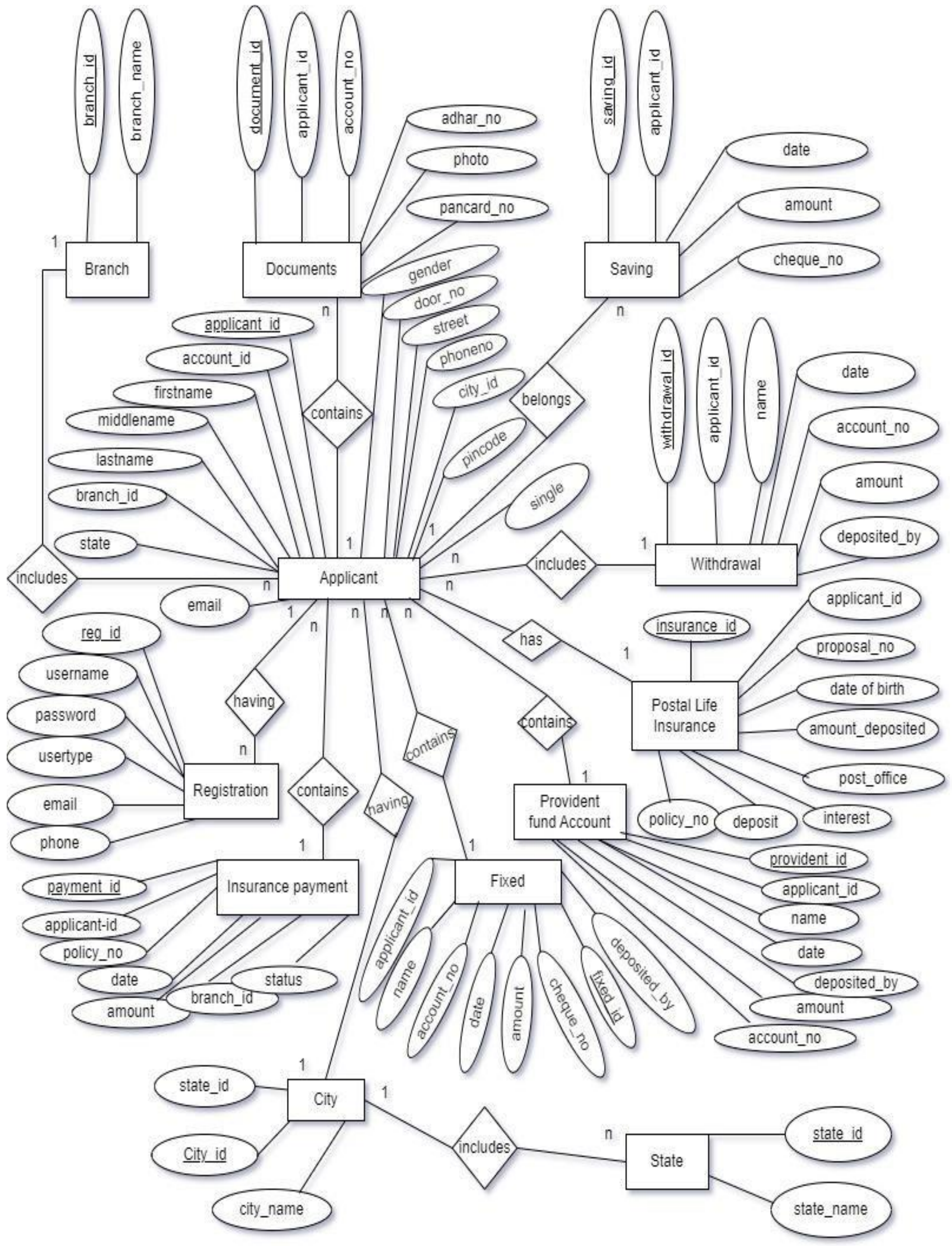


Figure 4.1 ER Diagram

4.9.2 DBMS CONFIGURATION:

Part of the DBMS installation process is the connection of the DBMS to other system software components that must interact with the DBMS. Typical infrastructure software is that may need to be configured to work with the DBMS includes networks, transaction processing monitors, message queues, other types of middleware, programming languages, Systems management software, Operations and job control software, Web servers, and application servers.

4.9.3 SUPPORT SOFTWARE REQUIRED:

MySQL Query, Visual Studio 2010

4.9.4 STORAGE REQUIREMENT:

- The storage engine represents the heart of a MySQL Server.
- Recovering the database from system failure.
- Management of files and database pages used to store data.
- Manage data buffers and system IO to the physical data pages.
- Manage locking and concurrency issues.

4.9.5 Backup and Recovery:

Database recovery is the process of restoring the databases to a correct state following a failure. The failure may be the result of a system crash due to hardware of software errors, a media failure, such as a head crash, or a software error in the application, such as a logical error in the program that is accessing the database. It is the responsibility of DBMS to ensure that the database is reliable and remains in a consistent state in the presence of failure. In general, backup and recovery refer to the various strategies and procedures involved in protecting the database against data loss and reconstructing the data such as that no data is lost after failure.

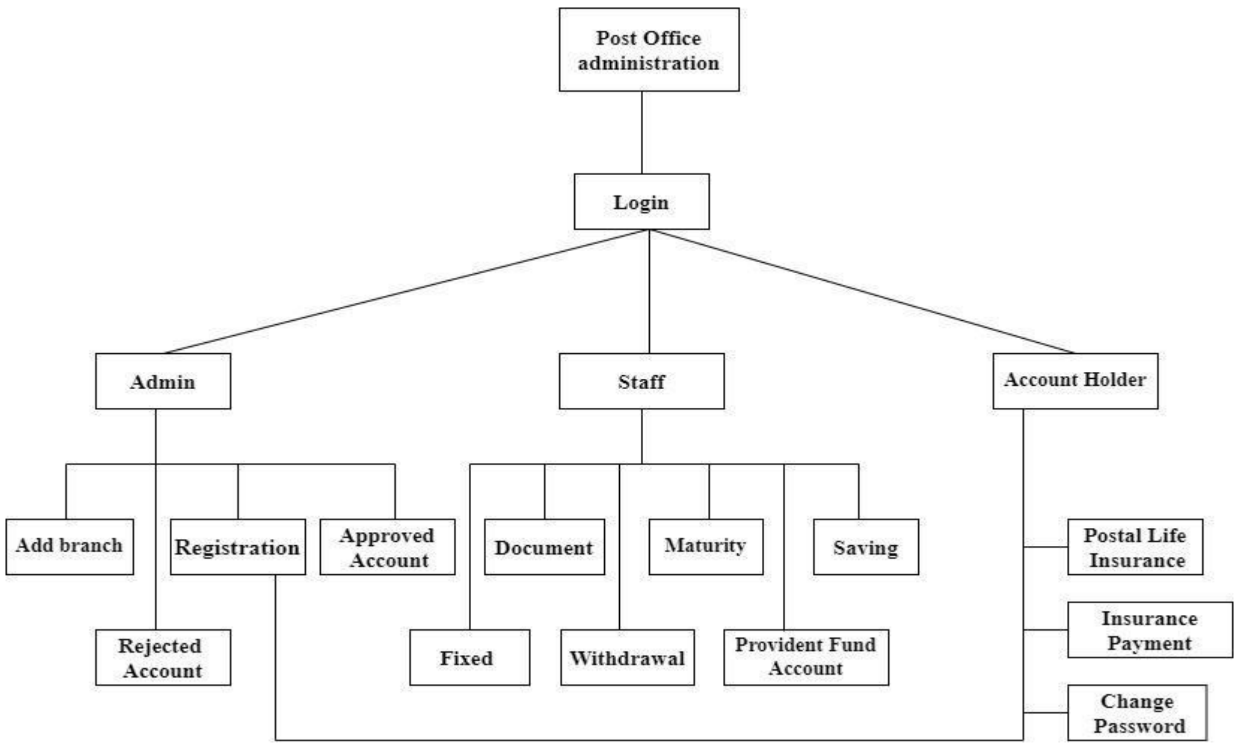
CHAPTER 5

DETAILED DESIGN

5.1 INTRODUCTION:

Detailed design is the second level of the design process. During detailed design, we specify how the module in the system interacts with each other and the internal logic of each of the modules specified during system design is decided, hence it is also called as logic design. Detailed design essentially expands the system design and database design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding. Detailed design is the phase where the design is refined and plans, specifications and estimates are created. Detailed design will include outputs such as 2D and 3D models, P & ID's, cost build up estimates, procurement plans etc. This phase is where the full cost of the project is identified.

5.2 STRUCTURE OF THE SOFTWARE PACKAGE:



5.3 MODULAR DECOMPOSITION OF THE SYSTEM:

5.3.1 Module 1: Login

Input: User ID, Password

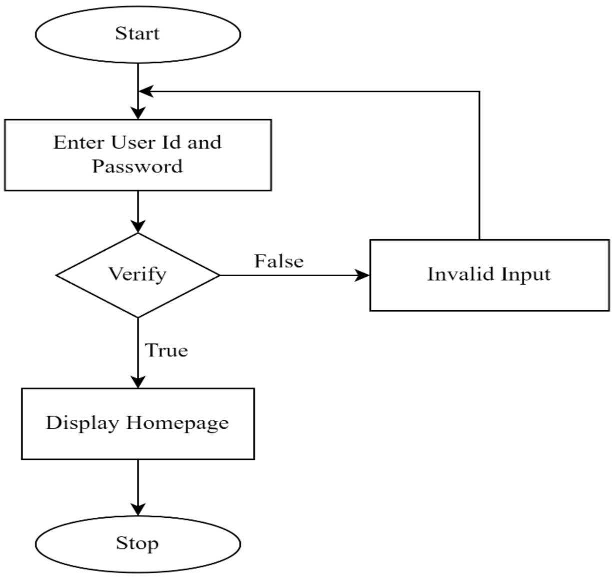


Figure 5.1 Login

- File I/O Interface: Registration Table.
- Output: Entered Username and password will be checked for validity, if it is valid admin will be directed to Homepage.

5.3.2 Module 2: Forget Password

Input: Username

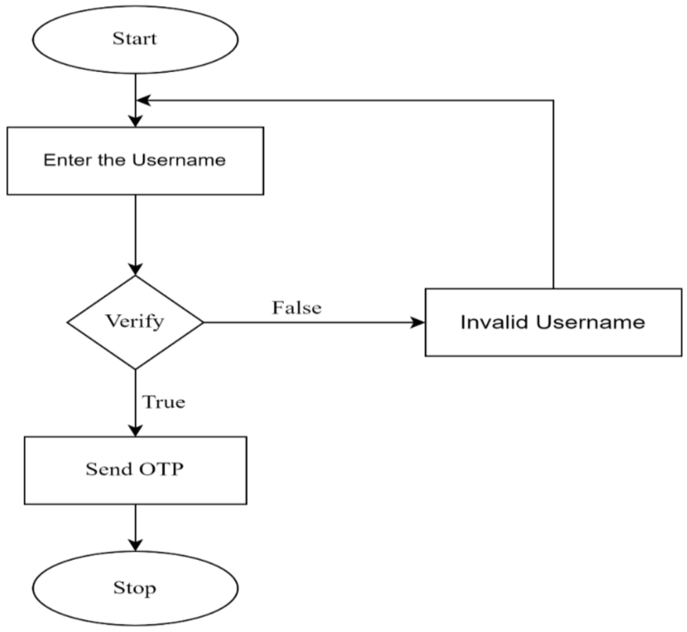


Figure 5.2 Forget Password

- File I/O Interface: Registration Table.
- Output: Entered username will be checked for validity, if it is valid the OTP will be sent to the User ID.

5.3.3 Module 3: Provident Fund

Input: A/C Number

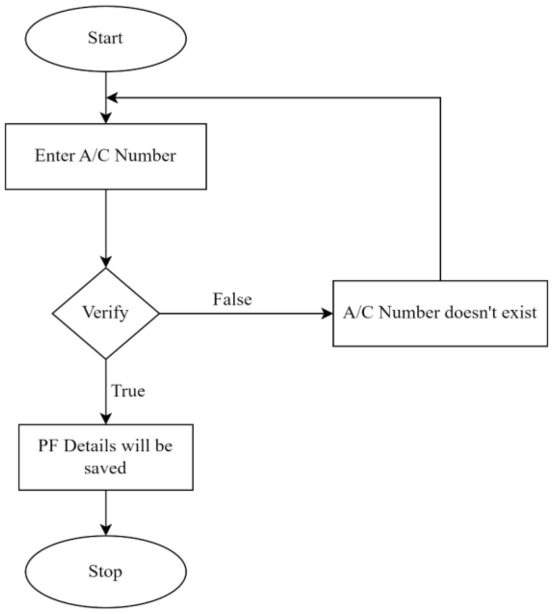


Figure 5.3 Provident Fund

- File I/O Interface: Provident Table.
- Output: Entered A/C Number will be checked for validity, if it is valid the PF details will be saved.

5.3.4 Module 4: Withdraw

Input: A/C Number.

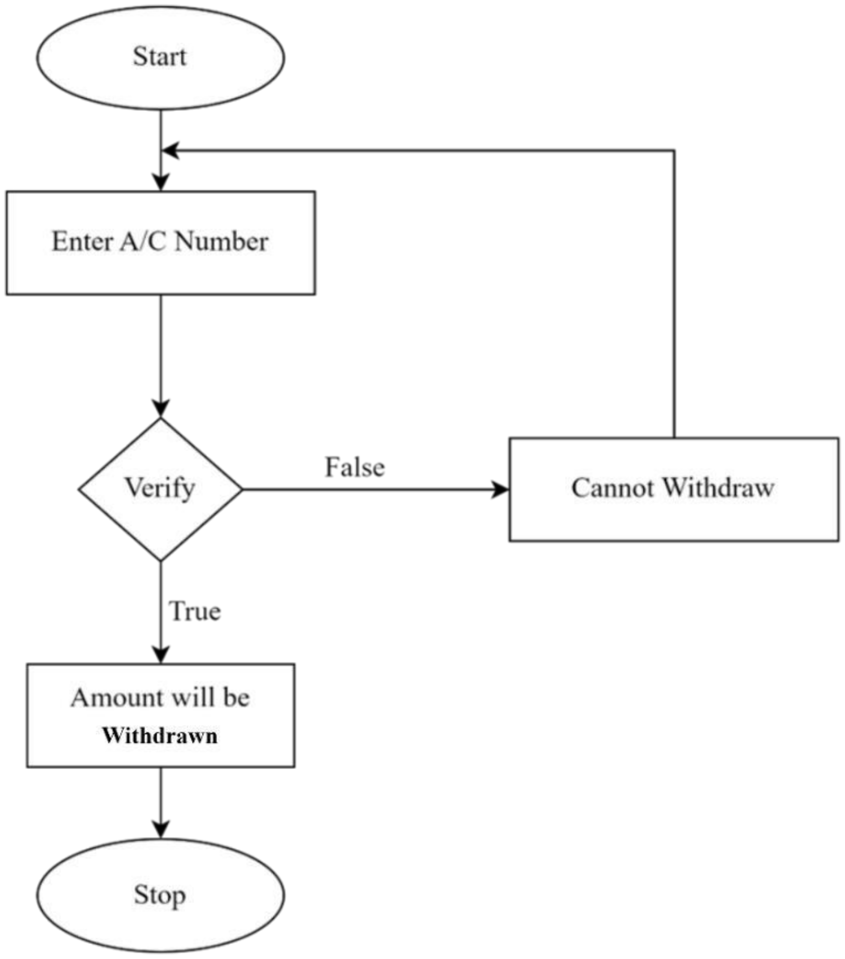


Figure 5.4 Withdraw

- File I/O Interface: Withdrawal Table.
- Output: Entered A/C Number will be checked for validity, if it is valid the amount will be withdrawn.

5.3.5 Module 5: Policy Maturity

Input: A/C Number and Policy Number.

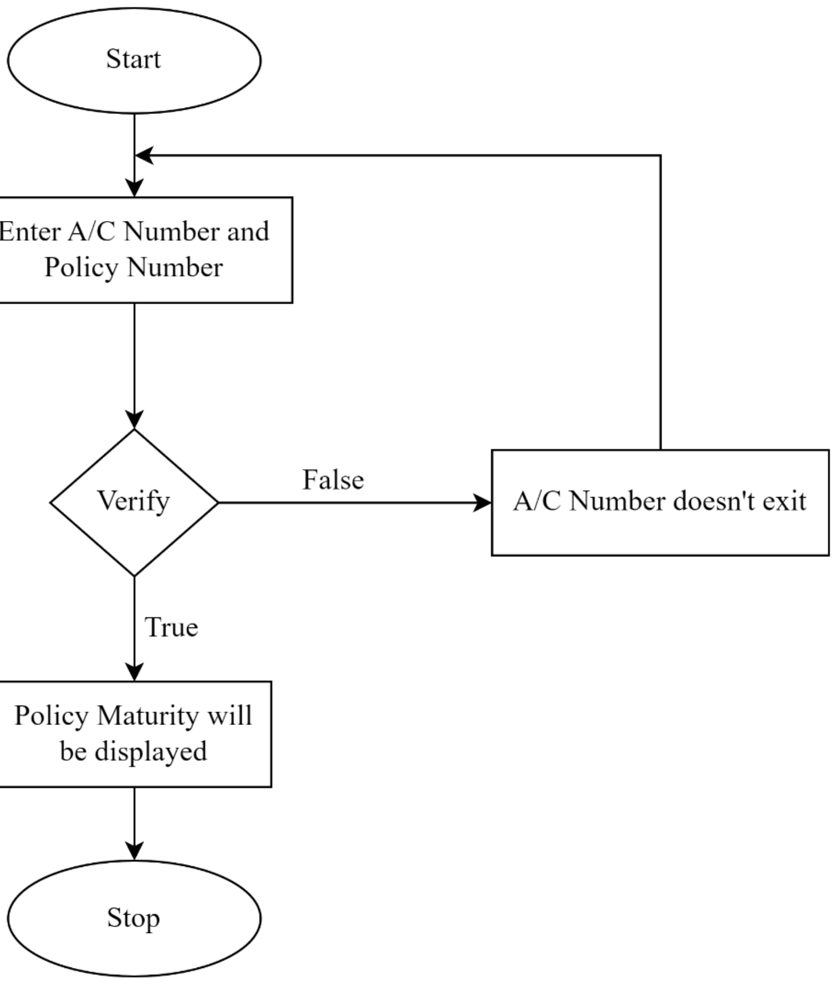

Start
Enter A/C Number and Policy Number
Verify
False
A/C Number doesn't exit
True
Policy Maturity will be displayed
Stop

Figure 5.5 Policy Maturity

- File I/O Interface: Policy Maturity Table.
- Output: Entered A/C Number and Policy number will be checked for validity, if it is valid the policy maturity will be displayed.

5.3.6 Module 6: Saving

Input: A/c Number.

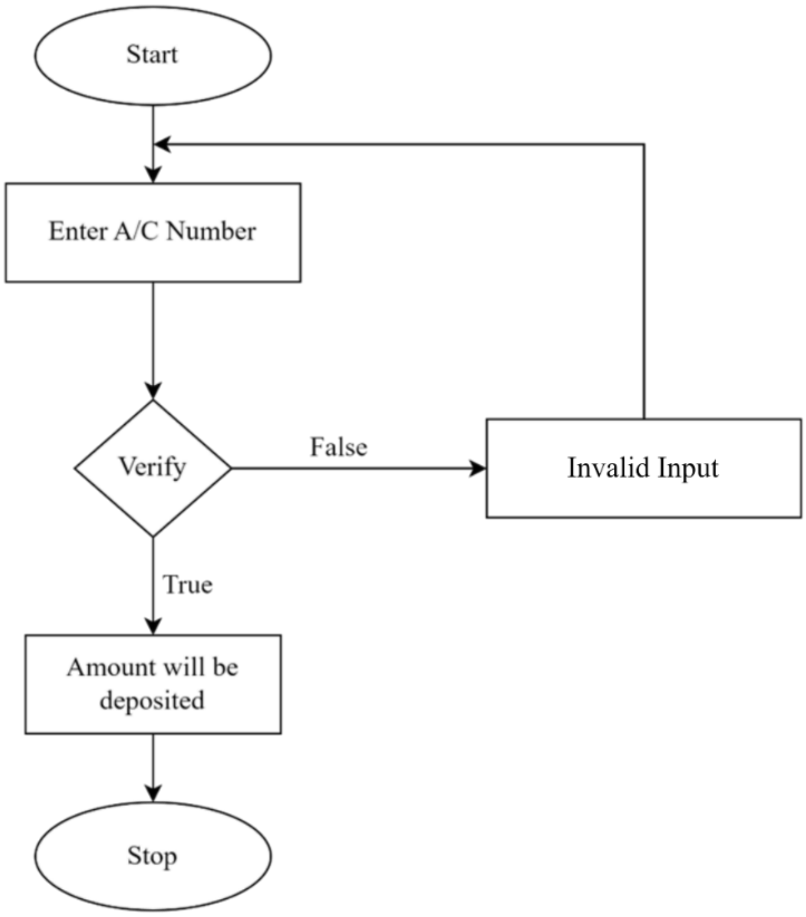


Figure 5.6 Saving

- File I/O Interface: Saving Table.
- Output: Entered A/C Number will be checked for validity, if it is valid the amount will be deposited.

5.3.7 Module 7: Fixed

Input: A/C Number.

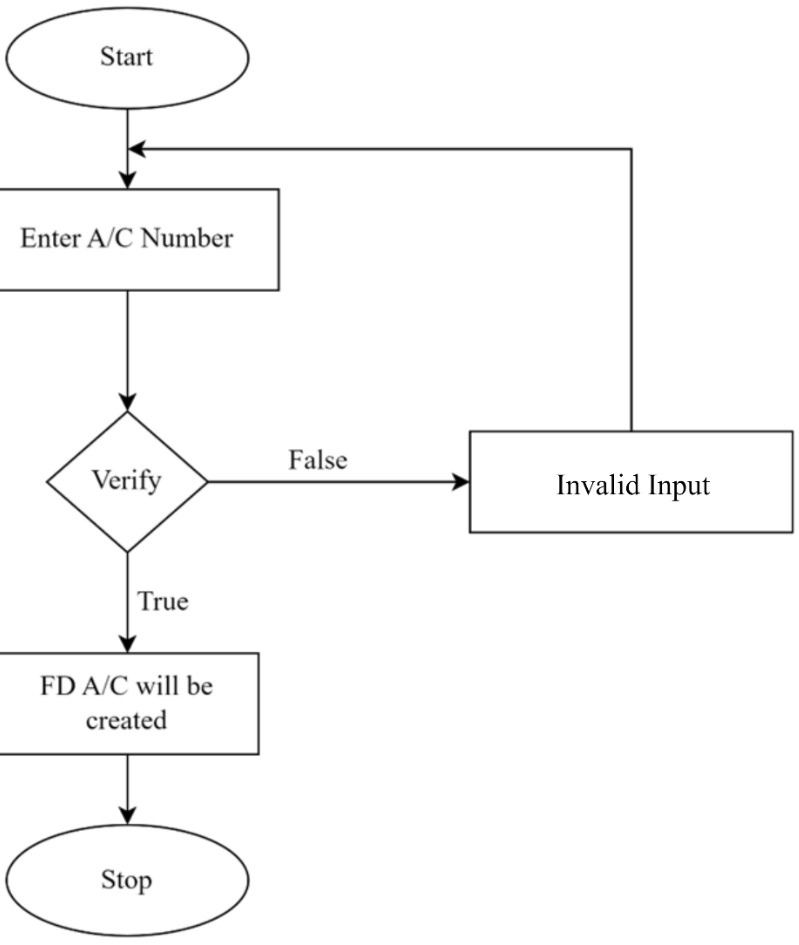


Figure 5.7 Fixed

- File I/O Interface: Fixed Table.
- Output: Entered A/C Number will be checked for validity, if it's valid the FD A/C will be created.

5.3.8 Module 8: Fixed Report

Input: A/C Number.

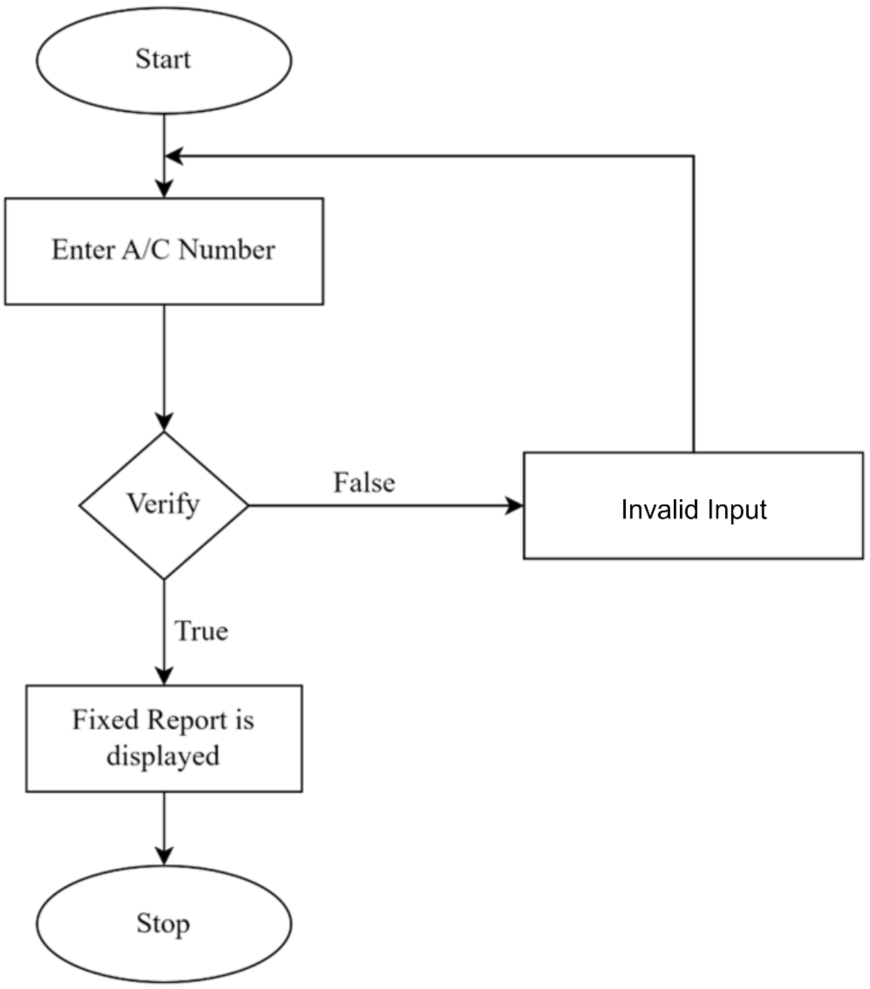


Figure 5.8 Fixed Report

- File I/O Interface: Fixed Table.
- Output: Entered A/C Number will be checked for validity, if it’s valid the fixed report is displayed.

5.3.9 Module 9: View Applicant

Input: A/C Number or Phone Number.

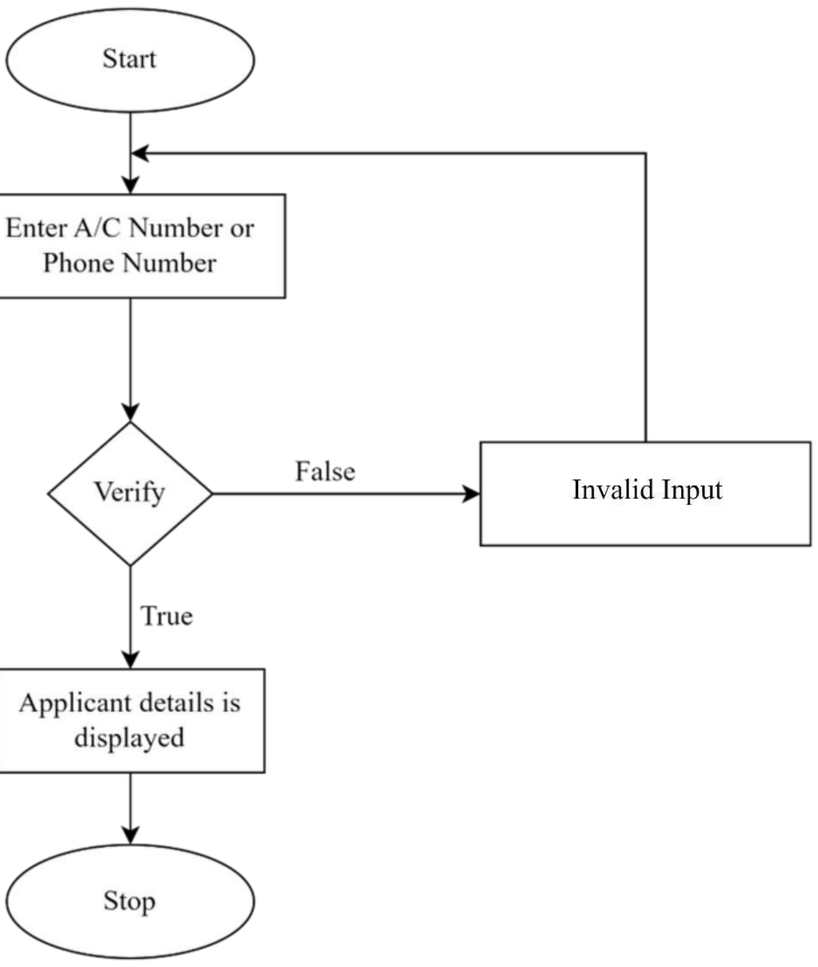

Start
Enter A/C Number or Phone Number
Verify
False
Invalid Input
True
Applicant details is displayed
Stop

Figure 5.9 View Application

5.3.10 Module 10: Applicant

Input: Applicant Credentials with Phone No. and Email.

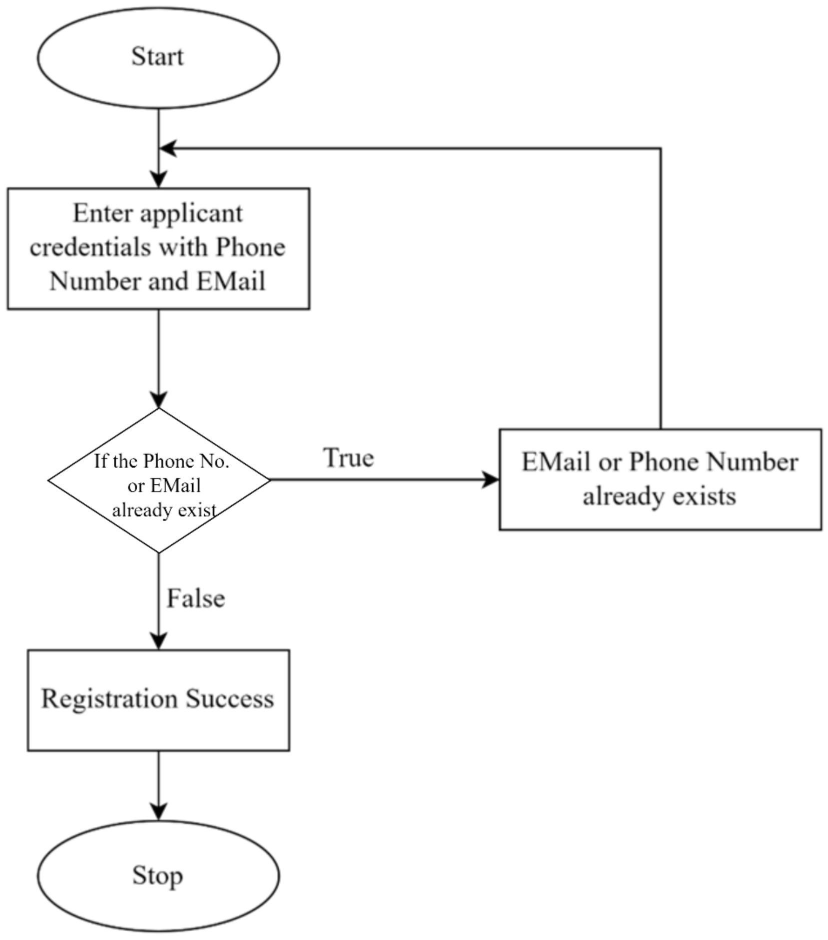

Start
Enter applicant credentials with Phone Number and EMail
If the Phone No. or EMail already exist
True
EMail or Phone Number already exists
False
Registration Success
Stop

Figure 5.10 Applicant

- File I/O Interface: Applicant Table.
- Output: Entered applicant details with the Phone No. and the Email will be checked for validity, if the Phone No. or the Email already exists then the system will display an error message.

5.3.11 Module 11: Change Password

Input: Current Password and New Password.

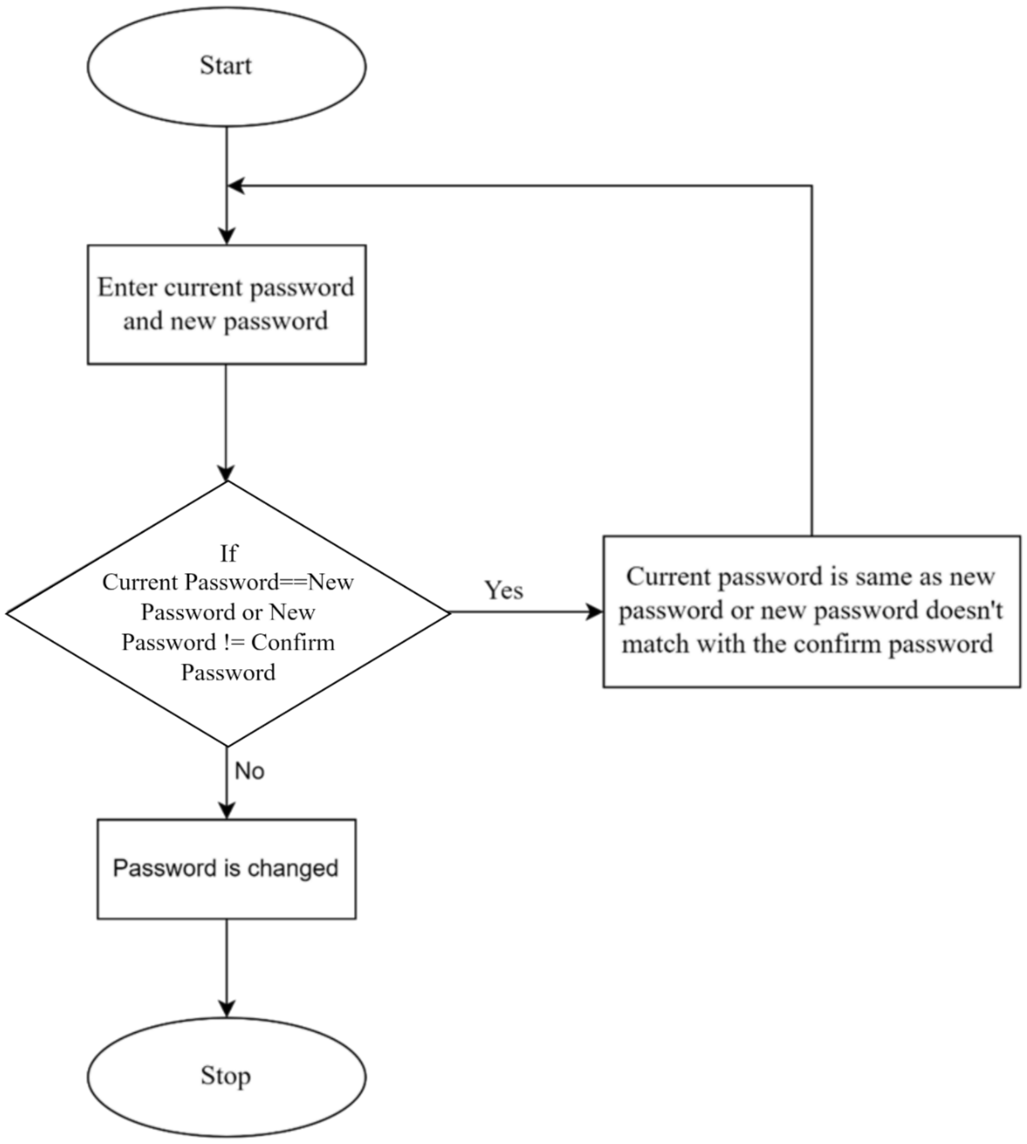

Start
Enter current password and new password
If Current Password==New Password or New Password != Confirm Password
Yes
Current password is same as new password or new password doesn't match with the confirm password
No
Password is changed
Stop

Figure 5.11 Change Password

- File I/O Interface: Registration Table.
- Output: Entered current password and new password will be checked for validity, if it is valid then the password will be updated.

CHAPTER 6
PROGRAM CODE LISTING

6.1 DATABASE CONNECTION:

A database connection is a SQL Developer object that specifies the necessary information for connecting to a specific database, as a specific user of that database.

```
DataLayer dl = new DataLayer();
String str = "insert into registration( name, email, phoneno, password,usertype) values('" +
txtname.Text + "','" + txtemail.Text + "','" + txtnum.Text + "','" + txtpass.Text + "','"+
ddlusertype.SelectedItem.Text  +"')"; dl.DmlCmd(str);
```

6.2 AUTHORIZATION / AUTHENTICATION:

Authentication is the process of verifying who you are. When you log on to a PC with a user name and password you are authenticating.

Authorization is the process of verifying that you access to something gaining access to a resource.

6.2.1 LOGIN (ADMIN, STAFF & APPLICANT):

```
using System; using
System.Collections.Generic; using
System.Linq; using System.Web;
using System.Web.UI; using
System.Web.UI.WebControls;
using System.Data;

public partial class Login : System.Web.UI.Page
{     protected void Page_Load(object sender, EventArgs
e)
    {


    }
    protected void btnlogin_Click(object sender, EventArgs e)
    {
        DataLayer dl = new DataLayer();
        String str = "select * from registration where email='" + txtemail.Text + "' and
password='" + txtpass.Text + "'";                    DataSet ds = new DataSet();
ds=dl.GetDataSet(str);        if (ds.Tables[0].Rows.Count>0)
        {
            Session["email"] = txtemail.Text;             if
(ds.Tables[0].Rows[0]["usertype"].ToString() == "Admin")
            {
                Response.Redirect("Approved.aspx");
            }
            else if (ds.Tables[0].Rows[0]["usertype"].ToString() == "Staff")
            {
                Response.Redirect("Provident.aspx");
            }
            else if (ds.Tables[0].Rows[0]["usertype"].ToString() == "Account Holder")
            {
                Response.Redirect("ApplicantNew.aspx");
            }

}
else
        {
            Response.Write("<script language=javascript> alert('Invalid Password') </script>");

        }
  }
}
```

6.2.2 Register (Applicant)

```
using System; using System.Collections.Generic;
using System.Linq; using System.Web; using
System.Web.UI; using System.Web.UI.WebControls;
public partial class Registration :
System.Web.UI.Page
{     protected void Page_Load(object sender, EventArgs
e)
    {

    }
    protected void btnsubmit_Click(object sender, EventArgs e)
    {

        String str = "insert into registration( name, email, phoneno, password,usertype) values('"
+ txtname.Text + "','" + txtemail.Text + "','" + txtnum.Text + "','" + txtpass.Text + "','Account
Holder')";
        DataLayer dl = new DataLayer();
dl.DmlCmd(str);
        Response.Write("<script language=javascript> alert(' You have Registered to Postal
Account Successfully!!,THANK YOU!!') </script>");
    }}
```

6.2.3 REGISTRATION (NEW ADMIN, STAFF OR USER):

```
using System;
using System.Collections.Generic;
using System.Linq; using
System.Web; using
System.Web.UI; using
System.Web.UI.WebControls;
public partial class Registration01 :
System.Web.UI.Page
{     protected void Page_Load(object sender, EventArgs
e)
```

```
    {

    }
    protected void btnsubmit_Click(object sender, EventArgs e)
    {
        DataLayer dl = new DataLayer();
        String str = "insert into registration( name, email, phoneno, password,usertype) values('"
+ txtname.Text + "','" + txtemail.Text + "','" + txtnum.Text + "','" + txtpass.Text + "','"+
ddlusertype.SelectedItem.Text +"')";        dl.DmlCmd(str);
        Response.Write("<script language=javascript> alert('You have Registered to Postal
Account Successfully!!,THANK YOU!!') </script>");
    }
}
```

6.3.2 Staff

6.3.2.1 Saving Account

```
using System; using
System.Collections.Generic; using
System.Linq; using System.Web;
using System.Web.UI; using
System.Web.UI.WebControls;
using System.Data;

public partial class Saving : System.Web.UI.Page
{     public static string id =
"";

    protected void Page_Load(object sender, EventArgs e)
    {         if
(!IsPostBack)
        {

        }
    }     public void
fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from applicant where account_id='" + txtno.Text + "'";
ds = dl.GetDataSet(str);         GridView1.DataSource = ds;
        GridView1.DataMember = "table";
        GridView1.DataBind();
    }

    protected void btnsave_Click(object sender, EventArgs e)
    {
        String str = "insert into saving( account_no, date, amount, cheque_no)values ('" +
txtno.Text + "','" + txtdate.Text + "','" + txtamt.Text + "','" + txtcheque.Text + "')";
DataLayer dl = new DataLayer();         dl.DmlCmd(str);
        Response.Write("<script language=javascript> alert('Record Inserted Successfully!!!
THANK YOU!!. ') </script>");
    }
    protected void btnclear_Click(object sender, EventArgs e)
    {         txtno.Text = "";
txtdate.Text       =       "";
txtamt.Text       =       "";
txtcheque.Text = "";
    }
    protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs
e)
    {
        String gender = null;
        String txtfname = null;
        String txtstate = null;
String txtemail = null;
        int rowIndex = Convert.ToInt32(e.CommandArgument);
        GridViewRow row = GridView1.Rows[rowIndex];
Label lblid = (Label)row.FindControl("lblID");        id =
lblid.Text;         txtfname =
GridView1.Rows[rowIndex].Cells[1].Text;         gender =
GridView1.Rows[rowIndex].Cells[2].Text;         txtstate =
GridView1.Rows[rowIndex].Cells[2].Text;         txtemail =
GridView1.Rows[rowIndex].Cells[2].Text;
    }
    protected void btnsearch_Click(object sender, EventArgs e)
    {
fillgrid();
    }
}
```

6.3 Data store/retrieval/update

6.3.1 Admin

6.3.1.1 Add Applicant

```
using System; using
System.Collections.Generic; using
System.Linq; using System.Web;
using System.Web.UI; using
System.Web.UI.WebControls;
using System.Data;

public partial class _Applicant : System.Web.UI.Page
{     public static string id = "";     protected void
Page_Load(object sender, EventArgs e)
    {         if
(!IsPostBack)
        {             fillgrid();
filldropdown();
filldropdown2();
```

```
        }     }     public
void fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String  str = "select
a.applicant_id,a.account_id,a.firstname,a.middlename,a.lastname,a.gender,a.doorno,a.street,a.
state,a.pincode,a.phoneno,a.email,a.status,a.deposit,a.photo,b.branch_name,c.cityname from
applicant a,branch b,city c where a.branch_id=b.branch_id and a.city_id=c.city_id";         ds
= dl.GetDataSet(str);         GridView1.DataSource = ds;
        GridView1.DataMember = "table";
        GridView1.DataBind();
    }
    public void filldropdown()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select branch_id ,branch_name from branch";
ds = dl.GetDataSet(str);         drpid.DataSource = ds;
drpid.DataTextField = "branch_name";
drpid.DataValueField = "branch_id";         drpid.DataBind();
drpid.Items.Insert(0, "-- Select Any --");
    }
    public void filldropdown2()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select city_id, cityname from city";
ds = dl.GetDataSet(str);         drpcity.DataSource =
ds;         drpcity.DataTextField = "cityname";
drpcity.DataValueField = "city_id";
drpcity.DataBind();         drpcity.Items.Insert(0, "--
Select Any --");
    }
    protected void btnsave_Click1(object sender, EventArgs e)
    {
        if (txtphone.Text.Length != 10)
        {
            Response.Write("<script language='javascript'>alert(' Phone Number should be 10
digits')</script>");
        }
else
        {
            DataLayer dl = new DataLayer();
            DataSet ds = new DataSet();
            String str = "select * from applicant where phoneno='" + txtphone.Text + "'";
ds = dl.GetDataSet(str);             if (ds.Tables[0].Rows.Count > 0)
            {
                Response.Write("<script language='javascript'>alert(' Phone Number already
exits')</script>");
            }

else
            {
                DataLayer d2 = new DataLayer();
                DataSet ds1 = new DataSet();
                String str1 = "select * from applicant where account_id='" + txtno.Text + "'";
ds1 = d2.GetDataSet(str1);                 if (ds1.Tables[0].Rows.Count > 0)
                {
                    Response.Write("<script language='javascript'>alert(' Account Number already
exits')</script>");
                }
else
                {
                    DataLayer d3 = new DataLayer();
                    DataSet ds3 = new DataSet();
                    String str4 = "select * from applicant where email='" + txtemail.Text + "'";
ds3 = dl.GetDataSet(str4);                    if (ds3.Tables[0].Rows.Count > 0)
                    {
                        Response.Write("<script language='javascript'>alert('This Email already
exits')</script>");
                    }
else
                    {
                        String gender = null;
if (rbmale.Checked == true)
                        {
                            gender = "male";
                        }                        else
{                              gender =
"female";
                        }

                        String str2 = "insert into
applicant(account_id,branch_id,city_id,firstname,middlename,lastname,gender,doorno,street,
state,pincode,phoneno,email,status,deposit,photo) values ('" + txtno.Text + "','" + drpid.Text
+ "','" + drpcity.Text + "','" + txtfname.Text + "','" + txtmname.Text + "','" + txtlname.Text +
"','" + gender + "','" + txtdoor.Text + "','" + txtstreet.Text + "','" + txtstate.Text + "','" +
txtpin.Text + "','" + txtphone.Text + "','" + txtemail.Text + "','Manual','" + txtdeposit.Text +
"','" + FileUpload1.FileName + "' )";
                        DataLayer d4 = new DataLayer();
d2.DmlCmd(str2);
                        Response.Write("<script language=javascript> alert('You have Registered to
Postal Account Successfully!!,THANK YOU!!') </script>");
                    }
                }
            }
        }
    }
    public void clear()
    {         txtno.Text = "";
drpid.SelectedIndex = 0;
drpcity.SelectedIndex = 0;
txtfname.Text = "";
txtmname.Text = "";
txtlname.Text = "";
rbmale.Checked = false;
rbfemale.Checked = false;
txtdoor.Text = "";
txtstreet.Text = "";
txtstate.Text = "";
txtpin.Text = "";
txtphone.Text = "";
txtemail.Text = "";
txtdeposit.Text = "";
    }

    protected void btnclear_Click1(object sender, EventArgs e)
    {
clear();
    }

    protected void btnupdate_Click(object sender, EventArgs e)
    {         if (id ==
null)
        {
            Response.Write("<script language='javascript'>alert('Please choose
record')</script>");
        }
else
        {
          String gender = null;
if (rbmale.Checked == true)
          {
             gender = "male";
          }
else
          {
             gender = "female";
          }
            DataLayer dl = new DataLayer();
          String str = "update applicant set account_id='" + txtno.Text + "', branch_id='" +
drpid.Text + "',city_id='" + drpcity.Text + "', firstname='" + txtfname.Text + "',
middlename='" + txtmname.Text + "', lastname='" + txtlname.Text + "', gender='" + gender +
"', doorno='" + txtdoor.Text + "', street='" + txtstreet.Text + "', state='" + txtstate.Text + "',
pincode='" + txtpin.Text + "', phoneno='" + txtphone.Text + "', email='" + txtemail.Text + "'
,deposit='" + txtdeposit.Text + "' where applicant_id='" + id + "'";
          //dl.DmlCmd(str);
          //fillgrid();
//string str = "update company_master set company_name='" + txtcname.Text + "', country='"
+ txtcountry.Text + "' where companyid='" + id + "'";
dl.DmlCmd(str);            fillgrid();
          Response.Write("<script language='javascript'>alert('Updated
successfully')</script>");
       }
    }
    protected void btndelete_Click(object sender, EventArgs e)
    {        if (id ==
null)
       {
          Response.Write("<script language='javascript'>alert('Please choose
record')</script>");
       }
else
```

```
        {
           String gender = null;
if (rbmale.Checked == true)
           {
              gender = "male";
           }
else
           {
              gender = "female";
           }

           DataLayer dl = new DataLayer();
           String str = "delete  from applicant where applicant_id='" + id + "'";
           //fillgrid();
           // dl.DmlCmd(str);
           //String str = "delete from company_master where companyid='" + id + "'";
dl.DmlCmd(str);            fillgrid();
        }
    }
    protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs
e)
    {
        String gender = null;         if
(rbmale.Checked == true)
        {
           gender = "male";
        }
else
        {
           gender = "female";
        }

        int rowIndex = Convert.ToInt32(e.CommandArgument);
        GridViewRow row = GridView1.Rows[rowIndex];
Label lblid = (Label)row.FindControl("lblID");         id =
lblid.Text;         txtno.Text =
GridView1.Rows[rowIndex].Cells[1].Text;       // drpid.Text =
GridView1.Rows[rowIndex].Cells[2].Text;       // drpcity.Text =
GridView1.Rows[rowIndex].Cells[2].Text;         txtfname.Text =
GridView1.Rows[rowIndex].Cells[4].Text;         txtmname.Text =
GridView1.Rows[rowIndex].Cells[5].Text;          txtlname.Text =
GridView1.Rows[rowIndex].Cells[6].Text;         gender=
GridView1.Rows[rowIndex].Cells[7].Text;         txtdoor.Text =
GridView1.Rows[rowIndex].Cells[8].Text;         txtstreet.Text =
GridView1.Rows[rowIndex].Cells[9].Text;         txtstate.Text =
GridView1.Rows[rowIndex].Cells[10].Text;         txtpin.Text =
GridView1.Rows[rowIndex].Cells[11].Text;         txtphone.Text =
GridView1.Rows[rowIndex].Cells[12].Text;         txtemail.Text =
GridView1.Rows[rowIndex].Cells[13].Text;         //single =
GridView1.Rows[rowIndex].Cells[14].Text;         //txtstatus.Text
= GridView1.Rows[rowIndex].Cells[15].Text;
txtdeposit.Text = GridView1.Rows[rowIndex].Cells[16].Text;
        Image1.Visible = true;
        Image1.ImageUrl = "~//photo//" + GridView1.Rows[rowIndex].Cells[15].Text;
    }
}
```

6.3.1.2 Add Life insurance

```
using System; using
System.Collections.Generic; using
System.Linq; using System.Web;
using System.Web.UI; using
System.Web.UI.WebControls;
using System.Data;

public partial class _Insurance : System.Web.UI.Page
```

```
{
   public static string id = "";
public static int flag = 0;
   protected void Page_Load(object sender, EventArgs e)
   {        if
(!IsPostBack)
       {

           filldropdown();
           DateTime dt = DateTime.Now ;
txtreceipt.Text = dt.ToString("yyyy-MM-dd");        }
}    public void fillgrid()
   {
       DataLayer dl = new DataLayer();
       DataSet ds = new DataSet();
       String str = "select * from applicant where account_id='" + txtno.Text + "' and
status='approved'";         ds = dl.GetDataSet(str);         if (ds.Tables[0].Rows.Count
> 0)
       {
flag = 1;
           GridView2.DataSource = ds;
           GridView2.DataMember = "table";
           GridView2.DataBind();
       }        else
{           flag =
0;
           Response.Write("<script language=javascript> alert('Applicant is either in pending
process or rejected') </script>");
       }
   }
   public void filldropdown()
   {
       DataLayer dl = new DataLayer();
       DataSet ds = new DataSet();
```

```
        String str = "select branch_id ,branch_name from branch";
    ds = dl.GetDataSet(str);         drpbranch.DataSource = ds;
    drpbranch.DataTextField = "branch_name";
    drpbranch.DataValueField = "branch_id";
    drpbranch.DataBind();        drpbranch.Items.Insert(0, "--
    Select Any --");
    }
    protected void btnsave_Click(object sender, EventArgs e)
    {         if (flag
== 1)
        {
            String str = "insert into policy_insurance( account_no, branch_id, policy_no,
date_of_receipt, maturity,deposit, interest)values ('" + txtno.Text + "','" + drpbranch.Text +
"','" + txtpolicy.Text + "','" + txtreceipt.Text + "','" + ddlmaturity.SelectedItem.Text + "','"
+ txtamt.Text + "','" + txtinterest.Text + "')";             DataLayer dl = new DataLayer();
dl.DmlCmd(str);
            Response.Write("<script language=javascript> alert('Record Inserted Successfully!!!
THANK YOU!!. ') </script>");
        }         else if
(flag == 0)
        {
            Response.Write("<script language=javascript> alert('Unable to save. Applicant is
either in pending process or rejected') </script>");
        }
    }
    protected void btnclear_Click(object sender, EventArgs e)
    {
        txtno.Text = "";
txtreceipt.Text = "";
txtamt.Text = "";
drpbranch.SelectedIndex = 0;
txtpolicy.Text = "";
```

```
txtinterest.Text = "";
ddlmaturity.SelectedIndex = 0;
    }

    protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs
e)
    {
        String gender = null;
        String txtfname = null;
        String txtstate = null;
String txtemail = null;
        int rowIndex = Convert.ToInt32(e.CommandArgument);
        GridViewRow row = GridView2.Rows[rowIndex];
Label lblid = (Label)row.FindControl("lblID");         id =
lblid.Text;         txtfname =
GridView2.Rows[rowIndex].Cells[1].Text;         gender =
GridView2.Rows[rowIndex].Cells[2].Text;         txtstate =
GridView2.Rows[rowIndex].Cells[2].Text;         txtemail =
GridView2.Rows[rowIndex].Cells[2].Text;
    }

    protected void btnsearch_Click(object sender, EventArgs e)
    {         if (id ==
null)
        {
            Response.Write("<script language='javascript'>alert('Please Enter Account
Number')</script>");
        }         else
{
fillgrid();
        }
    }
}
```

6.3.1.3 Add Branch

```
using System;
using System.Collections.Generic;
using System.Linq; using
System.Web;
using System.Web.UI; using
System.Web.UI.WebControls; using
System.Data; public partial class Branch :
System.Web.UI.Page
{     public static string id = "";      protected void
Page_Load(object sender, EventArgs e)
    {
if(!IsPostBack)
         {
fillgrid();
         }     }        public
void fillgrid()
{
    DataLayer dl=new DataLayer();
    DataSet ds= new DataSet();
String str="select * from branch";
ds=dl.GetDataSet(str);
GridView1.DataSource=ds;
    GridView1.DataMember="table";
    GridView1.DataBind();
} protected void  btnclear_Click(object sender, EventArgs
e)
{
clear(); }
protected void btnsave_Click(object sender, EventArgs e)
{
    DataLayer dl = new DataLayer();
    DataLayer d3 = new DataLayer();
    DataSet ds3 = new DataSet();
    String str4 = "select * from branch where branch_name='" + txtbranch.Text + "'"; ds3
    = dl.GetDataSet(str4);
    if (ds3.Tables[0].Rows.Count > 0)
    {
        Response.Write("<script language='javascript'>alert('This Branch already
exits')</script>");
    }
else
    {
        String str = "insert into branch(branch_name) values ('" + txtbranch.Text + "')";
DataLayer d2 = new DataLayer();         d2.DmlCmd(str);
        fillgrid();
        Response.Write("<script language='javascript'> alert('You have selected your Branch
Name!!! THANK YOU!!. ') </script>");
    }
} public void
clear()
{     txtbranch.Text =
"";
}


protected void btnupdate_Click(object sender, EventArgs e)
{
    DataLayer dl = new DataLayer();
    DataLayer d3 = new DataLayer();
    DataSet ds4 = new DataSet();
    String str4 = "select * from branch where branch_name='" + txtbranch.Text + "'";
ds4 = dl.GetDataSet(str4);     if (ds4.Tables[0].Rows.Count > 0)
    {
        Response.Write("<script language='javascript'>alert('This Branch already
exits')</script>");
```

```
    } else
    {
        DataLayer d2 = new DataLayer();
        String str = "update branch set branch_name='" + txtbranch.Text + "'where branch_id='"
+ id + "'";
        //dl.DmlCmd(str);
        //fillgrid();
       //string str = "update company_master set company_name='" + txtcname.Text + "',
country='" + txtcountry.Text + "' where companyid='" + id + "'";
d2.DmlCmd(str);
        fillgrid();
        Response.Write("<script language='javascript'>alert('Updated Successfully')</script>");
    }
}     protected void btndelete_Click(object sender, EventArgs
e)
    {
        DataLayer dl = new DataLayer();
        String str = "delete from branch where branch_id='" + id + "'";
        //dl.DmlCmd(str);
        //fillgrid();

         //String str = "delete from company_master where companyid='" + id + "'";
dl.DmlCmd(str);         fillgrid();
    }
protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        int rowIndex = Convert.ToInt32(e.CommandArgument);
        GridViewRow      row      =      GridView1.Rows[rowIndex];
Label  lblid  =  (Label)row.FindControl("lblID");                id  =
lblid.Text;                                        txtbranch.Text      =
GridView1.Rows[rowIndex].Cells[1].Text;
    }
    }
```

6.3.1.4 Manage Applicant

```
using System; using
System.Collections.Generic; using
System.Linq; using System.Web;
using System.Web.UI; using
System.Web.UI.WebControls; using
System.Data;

public partial class _Approved : System.Web.UI.Page
{     public static string id = null;     protected void
Page_Load(object sender, EventArgs e)
    {         if
(!IsPostBack)
        {
fillgrid();
        }
}
    public void fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
String str = "select * from applicant";
ds = dl.GetDataSet(str);
GridView1.DataSource = ds;
        GridView1.DataMember = "table";
        GridView1.DataBind();
    }
 protected void btnapprove_Click(object sender, EventArgs e)
    {         if (id ==
null)         {
```

```
            Response.Write("<script language='javascript'>alert('Please choose
record')</script>");
        }
else
        {
            String str = "update applicant set status='approved' where applicant_id='" + id + "'";
DataLayer dl = new DataLayer();             dl.DmlCmd(str);             fillgrid();
            Response.Write("<script language='javascript'>alert('Your ID has approved!!!!,
THANK YOU !!')</script>");
        }
    }
    protected void btnreject_Click(object sender, EventArgs e)
    {         if (id ==
null)
        {
            Response.Write("<script language='javascript'>alert('Please choose
record')</script>");
        }
else
        {
            String str = "update applicant set status='rejected' where applicant_id='" + id + "'";
DataLayer dl = new DataLayer();             dl.DmlCmd(str);             fillgrid();
            Response.Write("<script language=javascript>alert('Your ID has rejected!!, THANK
YOU !!')</script>");
        }
    }
   protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        String txtid = null;
        String txtfname = null;
        String txtstate = null;
        String txtphone = null;
        String txtmail = null;
String txtstatus = null;
```

```
        int rowIndex = Convert.ToInt32(e.CommandArgument);
        GridViewRow row = GridView1.Rows[rowIndex];
Label lblid = (Label)row.FindControl("lblID");         id =
lblid.Text;         txtid =
GridView1.Rows[rowIndex].Cells[1].Text;         txtfname =
GridView1.Rows[rowIndex].Cells[1].Text;         txtstate =
GridView1.Rows[rowIndex].Cells[1].Text;         txtphone =
GridView1.Rows[rowIndex].Cells[1].Text;         txtmail =
GridView1.Rows[rowIndex].Cells[1].Text;         txtstatus =
GridView1.Rows[rowIndex].Cells[1].Text;
    }
    protected void btnsearch_Click(object sender, EventArgs e)
    {
            DataLayer dl = new DataLayer();
            DataSet ds = new DataSet();
String str = null;             if
(rball.Checked == true)
            {
                str = "select * from applicant";
            }
            else if (rbapproved.Checked == true)
            {
                str = "select * from applicant where status='approved'";

            }
            else if (rbrejected.Checked == true)
            {
                str = "select * from applicant where status='rejected'";
            }
            else if (rbmanul.Checked == true)
            {
                str = "select * from applicant where status='Manual'";
            }
            else if (rbonline.Checked == true)
            {
                str = "select * from applicant where status='Online'";
            }
            ds = dl.GetDataSet(str);
GridView1.DataSource = ds;
            GridView1.DataMember = "table";
            GridView1.DataBind();
            }
}
```

6.3.1.5 Add State using System;

```
using System.Collections.Generic;
using System.Linq; using
System.Web; using
System.Web.UI; using
System.Web.UI.WebControls;
using System.Data;
Spublic partial class _State : System.Web.UI.Page
{
    public static string id = "";
    protected void Page_Load(object sender, EventArgs e)
    {        if
(!IsPostBack)
        {
fillgrid();
        }

    }    public void
fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
String str = "select * from state";
ds = dl.GetDataSet(str);
GridView1.DataSource = ds;
        GridView1.DataMember = "table";
        GridView1.DataBind();
    }
   protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        int rowIndex = Convert.ToInt32(e.CommandArgument);
        GridViewRow     row     =     GridView1.Rows[rowIndex];
Label  lblid  =  (Label)row.FindControl("lblID");               id  =
lblid.Text;                                        txtstate.Text      =
GridView1.Rows[rowIndex].Cells[1].Text;
     }
    protected void btnsave_Click(object sender, EventArgs e)
    {
        DataLayer dl = new DataLayer();
        DataLayer d3 = new DataLayer();
        DataSet ds3 = new DataSet();
        String str4 = "select * from state where state_name='" + txtstate.Text + "'";
ds3 = dl.GetDataSet(str4);         if (ds3.Tables[0].Rows.Count > 0)
        {
            Response.Write("<script language='javascript'>alert('This State already
exits')</script>");
        }
else
        {

            String str = "insert into state(state_name) values ('" + txtstate.Text + "')";
DataLayer d2 = new DataLayer();            d2.DmlCmd(str);            fillgrid();
            Response.Write("<script language=javascript> alert('Inserted into Successfully!!!
THANK YOU!!. ') </script>");
        }
    }     public void
clear()
```

```
    {        txtstate.Text
= "";
    }

    protected void btnclear_Click(object sender, EventArgs e)
    {
clear();
    }
    protected void btnupdate_Click(object sender, EventArgs e)
    {
        DataLayer dl = new DataLayer();
    DataLayer d3 = new DataLayer();
    DataSet ds3 = new DataSet();
    String str4 = "select * from state where state_name='" + txtstate.Text + "'";
ds3 = dl.GetDataSet(str4);     if (ds3.Tables[0].Rows.Count > 0)
    {
        Response.Write("<script language='javascript'>alert('This State already
exits')</script>");
    }
else
    {
        DataLayer d2 = new DataLayer();
        String str = "update state set state_name='" +txtstate.Text+ "' where state_id='" + id +
"'";
        d2.DmlCmd(str);
        fillgrid();
        Response.Write("<script language=javascript> alert('State has Updated Successfully!!!
THANK YOU!!. ') </script>");
    }
    }
    protected void btndelete_Click(object sender, EventArgs e)
    {
        DataLayer dl = new DataLayer();
        String str = "delete from state where state_id='" + id + "'";
dl.DmlCmd(str);        fillgrid();
    }
}
```

6.3.1.6 Add City

```
using System; using
System.Collections.Generic; using
System.Linq; using System.Web;
using System.Web.UI; using
System.Web.UI.WebControls; using
System.Data;
public partial class _city : System.Web.UI.Page
{        public static string id = "";        protected void
Page_Load(object sender, EventArgs e)
    {
if(!IsPostBack)
        {
fillgrid();
fillstate();        }
}    public void
fillgrid()
{
    DataLayer dl=new DataLayer();
    DataSet ds= new DataSet();
    String str = "select c.city_id, c.cityname,s.state_name from city c, state s where
c.state_id=s.state_id";    ds=dl.GetDataSet(str);    GridView1.DataSource=ds;
    GridView1.DataMember="table";
    GridView1.DataBind();
}    public void
fillstate()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select state_id,state_name from state";
ds = dl.GetDataSet(str);        drpstate.DataSource = ds;
drpstate.DataTextField = "state_name";
drpstate.DataValueField = "state_id";
drpstate.DataBind();        drpstate.Items.Insert(0, "--
Select Any --");
    }
protected void  btnsave_Click(object sender, EventArgs e)
{
     DataLayer dl = new DataLayer();
DataLayer d3 = new DataLayer();
    DataSet ds3 = new DataSet();
    String str4 = "select * from city where cityname='" + txtcity.Text + "'";
ds3 = dl.GetDataSet(str4);     if (ds3.Tables[0].Rows.Count > 0)
    {
        Response.Write("<script language='javascript'>alert('This City already
exits')</script>");
    }
else
    {
        DataLayer d2 = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from city where cityname='" + txtcity.Text + "'";
ds = d2.GetDataSet(str);        if (ds.Tables[0].Rows.Count > 0)
        {
            Response.Write("<script language='javascript'>alert(' City already exits')</script>");
        }
else
        {
  str = "insert into city(state_id,cityname) values ('" + drpstate.Text + "','" + txtcity.Text + "')";
            // DataLayer dl = new DataLayer();
d2.DmlCmd(str);            fillgrid();
```

```
            Response.Write("<script language=javascript> alert('You have inserted City!!!
THANK YOU!!. ') </script>");
        }
    }
}
public void clear()
{      drpstate.SelectedIndex =
0;     txtcity.Text = "";
} protected void btnclear_Click(object sender, EventArgs
e)
{     clear(); } protected void btnupdate_Click(object
sender, EventArgs e) {
    DataLayer d2 = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from city where cityname='" + txtcity.Text + "'";
ds = d2.GetDataSet(str);         if (ds.Tables[0].Rows.Count > 0)
        {
            Response.Write("<script language='javascript'>alert(' City already exits')</script>");
        }
else
        {
            DataLayer dl = new DataLayer();
            String str1 = "update city set state_id='" + drpstate.Text + "',cityname='" +
txtcity.Text + "'where city_id='" + id + "' ";             dl.DmlCmd(str1);
fillgrid();
        }
} protected void btndelete_Click(object sender, EventArgs
e)
{
    DataLayer dl = new DataLayer();
    String str = "delete from city where city_id='" + id + "'";
dl.DmlCmd(str);
    fillgrid();
```

```
}
protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
{
   int rowIndex = Convert.ToInt32(e.CommandArgument);
   GridViewRow row = GridView1.Rows[rowIndex];
Label lblid = (Label)row.FindControl("lblID");     id =
lblid.Text;
   //drpstate.Text = GridView1.Rows[rowIndex].Cells[1].Text;
txtcity.Text = GridView1.Rows[rowIndex].Cells[2].Text;
}}
```

6.3.2.2 Withdrawal

```
using System; using System.Collections.Generic;
using System.Linq; using System.Web; using
System.Web.UI; using System.Web.UI.WebControls;
using System.Data; using System.Net.Mail; using
System.Net; public partial class Withdrawal :
System.Web.UI.Page
{     public static string id = "";     public static int flag =
0;     protected void Page_Load(object sender,
EventArgs e)
   {
      lbldeposit .Text =Session ["email"].ToString();
if (!IsPostBack)
      {
         DateTime dt = DateTime.Now;
txtdate.Text = dt.ToString("yyyy-MM-dd");
      }
}
   public void fillgrid()
   {
      DataLayer dl = new DataLayer();
      DataSet ds = new DataSet();
```

```
        String str = "select * from applicant where account_id='" + txtno.Text + "' and
status='approved'";         ds = dl.GetDataSet(str);         if (ds.Tables[0].Rows.Count
> 0)
        {
flag = 1;
GridView2.Dat
aSource = ds;
            GridView2.DataMember = "table";
            GridView2.DataBind();
        }
else
        {
            Response.Write("<script language=javascript> alert('Cannot Withdraw') </script>");
        }
    }
    protected void btnsave_Click(object sender, EventArgs e)
    {        if (flag
== 1)
        {
            int balance = 0
String str2 = "select deposit from applicant where account_id='" + txtno.Text + "' and
status='approved'";
            DataSet   ds3   =   new   DataSet();
DataLayer    d3    =    new    DataLayer();
ds3  =  d3.GetDataSet(str2);                   if
(ds3.Tables[0].Rows.Count > 0)
            {
                balance = int.Parse(ds3.Tables[0].Rows[0]["deposit"].ToString());
if (int.Parse(txtamt.Text) < balance)
                {
                    String   str   =   "insert   into   withdrawal(   account_no,   date,   amount,
deposited_by)values ('" + txtno.Text + "','" + txtdate.Text + "','" + txtamt.Text + "','" +
lbldeposit.Text + "')";
```

```
                DataLayer dl = new DataLayer();
dl.DmlCmd(str);
                String str1 = "update applicant set deposit=deposit-'" + int.Parse(txtamt.Text) +
"' where account_id='" + txtno.Text + "'";
dl.DmlCmd(str1);                fillgrid();
                //Email code below
                //   String email = GridView1.Rows[0].Cells[13].Text;
using (var client = new SmtpClient())
                {
                   client.UseDefaultCredentials = false;                    var
NetworkCredentials = new NetworkCredential() { UserName =
"sapaligasriraksha@gmail.com", Password = "mhrd hmcl jjyu rqfw" };
client.Port = 587;                    client.EnableSsl = true;
client.Host = "smtp.gmail.com";                    client.Credentials =
NetworkCredentials;

                   MailMessage msg = new MailMessage();
msg.From = new MailAddress("sapaligasriraksha@gmail.com");
msg.To.Add(GridView2.Rows[0].Cells[3].Text);                    msg.Subject =
"Withdrawal Transaction";

                   msg.Body = "Your Withrawal Amount is" + txtamt.Text;
client.Send(msg);
                }
                Response.Write("<script language=javascript> alert('Record Inserted
Successfully!!! THANK YOU!!. ') </script>");
             }
else
             {
                Response.Write("<script language='javascript'> alert('Balance is less!!!! cant
withdraw. ')</script>");
             }
}
        else if (flag == 0)
```

```
            {
                Response.Write("<script language=javascript> alert('Cannot nwithdraw, Applicant
either in pending or rejected') </script>");
            }
        }
    }
    protected void  btnclear_Click(object sender, EventArgs e)
{           txtno.Text = "";
txtdate.Text      =      "";
txtamt.Text       =      "";
lbldeposit.Text = "";
}     protected void btnsearch_Click(object sender, EventArgs
e)
    {        if (id ==
null)
        {
            Response.Write("<script language='javascript'>alert('Please Enter Account
Number')</script>");
        }
else
{
            fillgrid();
            String str1 = "select deposit from applicant where account_id='" + txtno.Text + "'";
            DataSet   ds1   =   new   DataSet();
DataLayer    d2    =    new    DataLayer();
ds1  =  d2.GetDataSet(str1);                    if
(ds1.Tables[0].Rows.Count > 0)
            {
                lblbalance.Text = ds1.Tables[0].Rows[0]["deposit"].ToString();
            }
        }
    }
}
```

6.3.2.3 PF (Provident Fund)

```
using System; using System.Collections.Generic;
using System.Linq; using System.Web; using
System.Web.UI; using
System.Web.UI.WebControls; using System.Data;
public partial class Provident :
System.Web.UI.Page
{     public static string id = "";     public static int flag =
0;     protected void Page_Load(object sender,
EventArgs e)
    {
        lbldeposit.Text = Session["email"].ToString();
if (!IsPostBack)
        {
            DateTime dt = DateTime.Now;
txtdate.Text = dt.ToString("yyyy-MM-dd");
        }
}
    public void fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from applicant where account_id='" + txtno.Text + "' and
status='approved'";         ds = dl.GetDataSet(str);         if (ds.Tables[0].Rows.Count
> 0)
        {
flag = 1;
            GridView1.DataSource = ds;
            GridView1.DataMember = "table";
            GridView1.DataBind();
        }
else
{
```

```
            Response.Write("<script language=javascript> alert('Record Inserted Successfully!!!
THANK YOU!!. ') </script>");
        }
    }
    protected void btnsave_Click(object sender, EventArgs e)
    {
        if (flag==1){
    String str = "insert into provident( account_no, date, salary, amount, deposited_by)values
('" + txtno.Text + "','" + txtdate.Text + "','" + txtsalary.Text + "','" + txtamt.Text + "','" +
lbldeposit.Text + "')";
                DataLayer dl = new DataLayer();
dl.DmlCmd(str);                 fillgrid();
                Response.Write("<script language=javascript> alert('Record Inserted
Successfully!!! THANK YOU!!. ') </script>");
        }
        else if (flag == 0)
        {
Response.Write("<script language=javascript> alert('Please enter accont number ')
</script>");
        }
    }
    protected void btnclear_Click(object sender, EventArgs e)
    {        txtno.Text = "";
txtdate.Text       =       "";
txtamt.Text        =       "";
lbldeposit.Text = "";
    }
    protected void btnview_Click(object sender, EventArgs e)
    {        if (id ==
null)
        {
            Response.Write("<script language='javascript'>alert('Please Enter Account
Number')</script>");
```

```
        }         else
{
fillgrid();
        }
    }
    protected void txtsalary_TextChanged(object sender, EventArgs e)
    {         if (txtsalary.Text.Length
> 0)
        {
            int calc = (int.Parse(txtsalary.Text) * 12 )/ 100;
txtamt.Text = calc.ToString();
        }
    }
}
```

6.3.2.4 Fixed Deposit

```
using System; using System.Collections.Generic;
using System.Linq; using System.Web; using
System.Web.UI; using
System.Web.UI.WebControls; using
System.Data; public partial class _Fixed :
System.Web.UI.Page
{     public static string id =
""; public static int flag = 0;
    protected void Page_Load(object sender, EventArgs e)
    {               if
(!IsPostBack)
        {
filldropdown();
            lbldeposit.Text = Session["email"].ToString();
DateTime dt = DateTime.Now;             txtdate.Text =
dt.ToString("yyyy-MM-dd");
        }
        Panel1.Visible = false;
    }     public void
fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from applicant where account_id='" + txtno.Text + "' and
status='approved'";         ds = dl.GetDataSet(str);         if (ds.Tables[0].Rows.Count
> 0)
        {
flag = 1;
            GridView1.DataSource = ds;
            GridView1.DataMember = "table";
            GridView1.DataBind();
        }         else
{             flag =
0;
            Response.Write("<script language=javascript> alert('Applicant is either in pending
process or rejected ') </script>");
        }
    }
    public void filldropdown()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select branch_id ,branch_name from branch";
ds = dl.GetDataSet(str);         drpbranch.DataSource = ds;
drpbranch.DataTextField = "branch_name";
drpbranch.DataValueField = "branch_id";
drpbranch.DataBind();         drpbranch.Items.Insert(0, "--
Select Any --");
    }
    protected void btnsave_Click(object sender, EventArgs e)
```

```
   {        if (flag
== 1)
       {
           String mode = null;
if (rbcash.Checked == true)
           {
               mode = "Cash";
               String str2 = "insert into fixed ( account_no, payment_mode, date, amount,
transaction_year, deposited_by) values ('" + txtno.Text + "','" + mode + "','" + txtdate.Text +
"','" + txtamt.Text + "','" + drpyear.SelectedItem.Text + "','" + lbldeposit.Text + "')";
DataLayer d2 = new DataLayer();                d2.DmlCmd(str2);
           }           else               if
(rbcheque.Checked == true)
               {
                   mode = "Cheque";
                   String str2 = "insert into fixed ( account_no, payment_mode, cheque_no,
bank_name, branch_id, ifsc, date, amount, transaction_year, deposited_by) values ('" +
txtno.Text + "','" + mode + "','" + txtcheque.Text + "','" + txtbank.Text + "','" +
drpbranch.SelectedItem.Text + "','" + txtcode.Text + "','" + txtdate.Text + "','" + txtamt.Text +
"','" + drpyear.SelectedItem.Text + "','" + lbldeposit.Text + "')";
                   DataLayer d2 = new DataLayer();
d2.DmlCmd(str2);
               }
           Response.Write("<script language=javascript> alert('Record Inserted Successfully!!!
THANK YOU!!. ') </script>");
       }        else if
(flag == 0)
       {
           Response.Write("<script language=javascript> alert('Unable to save, Applicant is
either in pending or rejected ') </script>");
       }
   }
       protected void btnclear_Click(object sender, EventArgs e)
   {                    txtno.Text  =  "";
txtdate.Text = "";        txtamt.Text
=  "";               txtbank.Text  =  "";
drpbranch.SelectedIndex     =     0;
txtcode.Text             =             "";
txtcheque.Text           =           "";
lbldeposit.Text = "";
   }
  protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
   {
       String gender = null;
       String txtfname = null;
       String txtstate = null;
String txtemail = null;
       int rowIndex = Convert.ToInt32(e.CommandArgument);
       GridViewRow   row   =   GridView1.Rows[rowIndex];
Label lblid = (Label)row.FindControl("lblID");           id =
lblid.Text;                                       txtfname      =
GridView1.Rows[rowIndex].Cells[1].Text;           gender =
GridView1.Rows[rowIndex].Cells[2].Text;          txtstate =
GridView1.Rows[rowIndex].Cells[2].Text;         txtemail =
GridView1.Rows[rowIndex].Cells[2].Text;
   }
   protected void btnsearch_Click(object sender, EventArgs e)
   {        if (id ==
null)
       {
           Response.Write("<script language='javascript'>alert('Please Enter Account
Number')</script>");
       }        else
{
fillgrid();
       }
```

```
    }
    protected void rbcash_CheckedChanged(object sender, EventArgs e)
    {
        Panel1.Visible = false;
    }
    protected void rbcheque_CheckedChanged(object sender, EventArgs e)
    {
        Panel1.Visible = true;
    }
    public string mode { get; set; }
    protected void txtinterest_TextChanged(object sender, EventArgs e)
    {
        if (txtinterest.Text.Length > 0)
        {
            lblyear.Text      =      drpyear.SelectedItem.Text;
long val = long.Parse(txtinterest.Text);                long
amount = long.Parse(txtamt.Text);               int years =
int.Parse(drpyear.SelectedItem.Text);
if(drpyear.SelectedIndex==1)
        {
            years=years*12;
        }
        else if(drpyear.SelectedIndex==3)
        {
            years=years*36;
     }
        else if(drpyear.SelectedIndex==5)
        {
            years=years*65;
        }
        else if(drpyear.SelectedIndex==10)
        {
            years=years*120;
        }
           long result=(amount*val)/100;
result= result*years;             long
total=result+long.Parse(txtamt.Text);
lbltotal.Text=total.ToString();
            }
    }
}
```

6.3.2.5 Fixed Report

```
using
System; using
System.Collections.Generic; using
System.Linq; using System.Web;
using System.Web.UI; using
System.Web.UI.WebControls;
using System.Data ;

public partial class ViewFixed_report : System.Web.UI.Page
{     public static string id = "";     protected void
Page_Load(object sender, EventArgs e)
    {
    }
    protected void btnview_Click(object sender, EventArgs e)
    {     if (id ==
null)
        {
            Response.Write("<script language='javascript'>alert('Please Enter Account
Number')</script>");
        }
else
        {
            DataLayer dl = new DataLayer();
            DataSet ds = new DataSet();
            String str = "select * from fixed where account_no='" + txtno.Text + "'";
ds = dl.GetDataSet(str);              GridView1.DataSource = ds;
            GridView1.DataMember = "table";
            GridView1.DataBind();
        }
    }
    }
```

6.3.2.6 Verify Documents

```
using System; using
System.Collections.Generic; using
System.Linq; using System.Web;
using System.Web.UI; using
System.Web.UI.WebControls;
using System.Data;


public partial class View_Document : System.Web.UI.Page
{     public static string id = null;     protected void
Page_Load(object sender, EventArgs e)
    {         if
(!IsPostBack)
        {
fillgrid();
        }     }     public
void fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from document where status='pending'";
ds = dl.GetDataSet(str);         GridView1.DataSource = ds;
        GridView1.DataMember = "table";
        GridView1.DataBind();
    }
```

```
  protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
   {
      int rowIndex = Convert.ToInt32(e.CommandArgument);
       GridViewRow row = GridView1.Rows[rowIndex];
Label lblid = (Label)row.FindControl("lblID");        id =
lblid.Text;
      // txtcname.Text = GridView1.Rows[rowIndex].Cells[1].Text;
       //txtcountry.Text = GridView1.Rows[rowIndex].Cells[2].Text;
     }
   protected void Button1_Click(object sender, EventArgs e)
   {        if (id ==
null)
       {
           Response.Write("<script language='javascript'>alert('Please choose
record')</script>");
       }
else
       {
          String str = "update document set status='verified' where document_id='" + id + "'";
DataLayer dl = new DataLayer();            dl.DmlCmd(str);            fillgrid();
           Response.Write("<script language='javascript'>alert('Your Id has verified!!!!,
THANK YOU !!')</script>");
       }
     }
   protected void DownloadFile1(object sender, EventArgs e)
   {
       string filePath = (sender as LinkButton).CommandArgument;
       Response.ContentType = ContentType;
       Response.AppendHeader("Content-Disposition", "attachment; filename=" +
Server.MapPath("document1//"+filePath));
       Response.WriteFile(Server.MapPath("document1//" + filePath));
       Response.End();
   }
  protected void DownloadFile2(object sender, EventArgs e)
   {
       string filePath = (sender as LinkButton).CommandArgument;
       Response.ContentType = ContentType;
       Response.AppendHeader("Content-Disposition", "attachment; filename=" +
Server.MapPath("document2//" + filePath));
       Response.WriteFile(Server.MapPath("document2//" + filePath));
       Response.End();
   }
protected void DownloadFile3(object sender, EventArgs e)
   {
       string filePath = (sender as LinkButton).CommandArgument;
       Response.ContentType = ContentType;
       Response.AppendHeader("Content-Disposition", "attachment; filename=" +
Server.MapPath("document3//" + filePath));
       Response.WriteFile(Server.MapPath("document3//" + filePath));
       Response.End();
   }
}
```

6.3.2.7 Policy Maturity

```
using System; using System.Collections.Generic;
using System.Linq; using System.Web; using
System.Web.UI; using
System.Web.UI.WebControls; using System.Data;
public partial class _Maturity :
System.Web.UI.Page
{      public static string id = "";     protected void
Page_Load(object sender, EventArgs e)
    {            if
(!IsPostBack)
        {
        }
}
        public void fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from applicant where account_id='" + txtno.Text + "'";
ds = dl.GetDataSet(str);        GridView1.DataSource = ds;
        GridView1.DataMember = "table";
        GridView1.DataBind();
    }    public void
fillgrid2()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str1 = "select p.account_no, p.policy_no, p.deposit, p.interest, i.amount,
i.transaction_month from policy_insurance p, insurance_transaction i where
i.policy_no=p.policy_no and p.policy_no='" + txtpolicy.Text  + "'";
ds = dl.GetDataSet(str1);        GridView3.DataSource = ds;
        GridView3.DataMember = "table";
        GridView3.DataBind();
    }
   protected void btnsearch_Click(object sender, EventArgs e)
    {        if (id ==
null)
        {
            Response.Write("<script language='javascript'>alert('Please Enter Account
Number')</script>");
        }        else
{
fillgrid();
        }
    }
  protected void btnfind_Click(object sender, EventArgs e)
```

```
    {
        if (id == null)
        {
            Response.Write("<script language='javascript'>alert('Please Enter Account
Number')</script>");
        }
        else
        {
        fillgr
        id2()
        ;
        }
    }
   }
```

6.3.2.8 View Applicant

```
using System; using System.Collections.Generic; using
System.Linq; using System.Web; using System.Web.UI;
using System.Web.UI.WebControls; using System.Data;
public partial class ViewApplicant : System.Web.UI.Page
{     public static string id = "";     protected void
Page_Load(object sender, EventArgs e)
    {         if
(!IsPostBack)
        {
        }
    }

    public void fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from applicant where status='Online' ";
ds = dl.GetDataSet(str);         GridView1.DataSource = ds;
        GridView1.DataMember = "table";
        GridView1.DataBind();
    }
   protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        int rowIndex = Convert.ToInt32(e.CommandArgument);
        GridViewRow row = GridView1.Rows[rowIndex];
Label lblid = (Label)row.FindControl("lblID");         id
= lblid.Text;         String txtno = null;
        String txtfname=null;
         String txtmname=null;
         String txtlname=null;
         String gender = null;
String txtdoor=null;
         String txtstreet=null;
         String txtstate=null;
         String txtpin=null;
         String txtphone=null;
         String txtemail=null;
         String txtdeposit = null;        //  String single = null;
txtno = GridView1.Rows[rowIndex].Cells[2].Text;         //
drpcity.Text = GridView1.Rows[rowIndex].Cells[2].Text;
txtfname = GridView1.Rows[rowIndex].Cells[4].Text;
txtmname = GridView1.Rows[rowIndex].Cells[5].Text;
txtlname = GridView1.Rows[rowIndex].Cells[6].Text;
gender = GridView1.Rows[rowIndex].Cells[7].Text;
txtdoor = GridView1.Rows[rowIndex].Cells[8].Text;
txtstreet = GridView1.Rows[rowIndex].Cells[9].Text;
txtstate = GridView1.Rows[rowIndex].Cells[10].Text;
txtpin = GridView1.Rows[rowIndex].Cells[11].Text;
txtphone = GridView1.Rows[rowIndex].Cells[12].Text;
txtemail = GridView1.Rows[rowIndex].Cells[13].Text;
```

```
      // single = GridView1.Rows[rowIndex].Cells[14].Text; txtdeposit
       = GridView1.Rows[rowIndex].Cells[15].Text

    }
    protected void btnsearch_Click(object sender, EventArgs e)
    {        if (id ==
null)
        {
            Response.Write("<script language='javascript'>alert('Please Enter Account Number
or Phone Number')</script>");
        }
else
        {
            DataLayer dl = new DataLayer();
            DataSet ds = new DataSet();
String str = null;            if
(rbphone.Checked == true)
            {
                str = "select * from applicant where phoneno='" + txtsearch.Text + "'";
            }
            else if (rbano.Checked == true)
            {
                str = "select * from applicant where account_id='" + txtsearch.Text + "'";
            }
            ds = dl.GetDataSet(str);
GridView1.DataSource = ds;
            GridView1.DataMember = "table";
            GridView1.DataBind();
        }
    }

}
```

6.3.2.9 Saving report

```
using System; using System.Collections.Generic; using
System.Linq; using System.Web; using
System.Web.UI; using System.Web.UI.WebControls;
using System.Data; public partial class saving_report :
System.Web.UI.Page
{     public static string id = "";     protected void
Page_Load(object sender, EventArgs e)
    {         if
(!IsPostBack)
        {

        }     }     public
void fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from applicant where account_id='" + txtno.Text + "'";
ds = dl.GetDataSet(str);         GridView2.DataSource = ds;
        GridView2.DataMember = "table";
        GridView2.DataBind();
    }
    public void fillgrid1()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from new_saving where account_no='" + txtno.Text + "'";         ds =
dl.GetDataSet(str);         if (ds.Tables[0].Rows.Count > 0)
        {
            GridView1.DataSource = ds;
            GridView1.DataMember            =            "table";
GridView1.DataBind();           int amount = 0;           for
(int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {
                amount = amount + int.Parse(ds.Tables[0].Rows[i]["amount"].ToString());
            }
            lblamt.Text = amount.ToString();
        }
    }
    protected void btnview_Click(object sender, EventArgs e)
    {
fillgrid();
fillgrid1();
    }
}
```

6.3.3 Applicant

6.3.3.1 Add Applicant Details

```
using System; using System.Collections.Generic; using
System.Linq; using System.Web; using System.Web.UI;
using System.Web.UI.WebControls; using System.Data;
public partial class ApplicantNew : System.Web.UI.Page
{     protected void Page_Load(object sender, EventArgs
e)     {         if (!IsPostBack)
        {
filldropdown();
filldropdown2();
            txtemail.Text = Session["email"].ToString();
             DataLayer dl = new DataLayer();
            DataSet ds = new DataSet();
            String str = "select * from registration where email='" + txtemail.Text + "'";
ds = dl.GetDataSet(str);            if (ds.Tables[0].Rows.Count > 0)
            {
                txtfname.Text = ds.Tables[0].Rows[0]["name"].ToString();
txtphone.Text = ds.Tables[0].Rows[0]["phoneno"].ToString();
            }
        }
    }
    public void filldropdown()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select branch_id ,branch_name from branch";
ds = dl.GetDataSet(str);         drpid.DataSource = ds;
        drpid.DataTextField = "branch_name";
drpid.DataValueField = "branch_id";
drpid.DataBind();
        drpid.Items.Insert(0, "-- Select Any --");
    }
    public void filldropdown2()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select city_id, cityname from city";
        ds = dl.GetDataSet(str);
drpcity.DataSource = ds;
drpcity.DataTextField = "cityname";
drpcity.DataValueField = "city_id";
drpcity.DataBind();         drpcity.Items.Insert(0,
"-- Select Any --");
    }
    protected void btnreg_Click(object sender, EventArgs e)
    {
        if (txtphone.Text.Length != 10)
        {
            Response.Write("<script language='javascript'>alert(' Phone Number should be 10
digits')</script>");
```

```
        }
else
        {
            DataLayer dl = new DataLayer();
            DataSet ds = new DataSet();
            String str = "select * from applicant where phoneno='" + txtphone.Text + "'";
ds = dl.GetDataSet(str);             if (ds.Tables[0].Rows.Count > 0)
            {
                Response.Write("<script language='javascript'>alert(' Phone Number already
exits')</script>");
            }
else
            {
                 DataLayer d3 = new DataLayer();
                   DataSet ds3 = new DataSet();
                   String str4 = "select * from applicant where email='" + txtemail.Text + "'";
ds3 = dl.GetDataSet(str4);                    if (ds3.Tables[0].Rows.Count > 0)
                   {
                       Response.Write("<script language='javascript'>alert('This Email already
exits')</script>");
                   }
else
                   {
                String gender = null;
if (rbmale.Checked == true)
                {
gender = "male";
                }
else
                {                   gender
= "female";
                }
                if (FileUpload1.HasFile)
                {
```

```
                FileUpload1.SaveAs(Server.MapPath("photo//" + FileUpload1.FileName));
            }
            String str1 = "insert into
applicant(branch_id,city_id,firstname,middlename,lastname,gender,doorno,street,state,pincod
e,phoneno,email,status,photo) values ('" + drpid.Text + "','" + drpcity.Text + "','" +
txtfname.Text + "','" + txtmname.Text + "','" + txtlname.Text + "','" + gender + "','" +
txtdoor.Text + "','" + txtstreet.Text + "','" + txtstate.Text + "','" + txtpin.Text + "','" +
txtphone.Text + "','" + txtemail.Text + "','Online','" + FileUpload1.FileName + "')";
DataLayer d2 = new DataLayer();          dl.DmlCmd(str1);
            Response.Write("<script language=javascript> alert('You have Registered to Postal
Account Successfully!!,THANK YOU!!') </script>");
          }
          }
        }
    }
}
```

6.3.3.2 Document Submission

```
using System; using System.Collections.Generic;
using System.Linq; using System.Web; using
System.Web.UI; using
System.Web.UI.WebControls; using System.Data;
public partial class Document :
System.Web.UI.Page
{     public static string id = "";     public static string
accountno = "";        protected void Page_Load(object
sender, EventArgs e)
    {        if
(!IsPostBack)
        {
fillgrid();
        }
}
    public void fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from applicant where account_id IN(select account_id from
applicant where email='" + Session["email"].ToString() + "')";            ds =
dl.GetDataSet(str);        if (ds.Tables[0].Rows.Count > 0)
        {
            accountno = ds.Tables[0].Rows[0]["account_id"].ToString();
txtno.Text=accountno.ToString();
        }
    }


   protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        String gender = null;
        String txtfname = null;
        String txtstate = null;
String txtemail = null;
        int rowIndex = Convert.ToInt32(e.CommandArgument);
        GridViewRow row = GridView2.Rows[rowIndex];
Label lblid = (Label)row.FindControl("lblID");        id =
lblid.Text;        txtfname =
GridView2.Rows[rowIndex].Cells[1].Text;        gender =
GridView2.Rows[rowIndex].Cells[2].Text;        txtstate =
GridView2.Rows[rowIndex].Cells[2].Text;        txtemail =
GridView2.Rows[rowIndex].Cells[2].Text;
    }
    protected void btnupload_Click(object sender, EventArgs e)
    {
        if (FileUpload1.HasFile)
        {
            FileUpload1.SaveAs(Server.MapPath("document1//" + FileUpload1.FileName));
        }
        if (FileUpload2.HasFile)
        {
           FileUpload2.SaveAs(Server.MapPath("document2//" + FileUpload2.FileName));
        }
        if (FileUpload3.HasFile)
        {
           FileUpload3.SaveAs(Server.MapPath("document3//" + FileUpload3.FileName));
        }


        String str = "insert into document( account_id, adharcard, photo, pancard,status)values('"
+ accountno  + "','" + FileUpload1.FileName + "','" + FileUpload2.FileName + "','" +
FileUpload3.FileName + "','pending')";
        //lblmsg.Text = "One row inserted";
DataLayer dl = new DataLayer();
dl.DmlCmd(str);
        Response.Write("<script language='javascript'>alert('Document Uploaded')</script>");
           }
    public void clear()
    {
txtno.Text="";
    }
  protected void clear_Click(object sender, EventArgs e)
    {
clear();
     }
}
```

6.3.3.3 Insurance Payment

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Web; using System.Web.UI; using
```

```
System.Web.UI.WebControls; public partial class
_Paymentmethod : System.Web.UI.Page
{     protected void Page_Load(object sender, EventArgs
e)
    {
    }
    protected void btnconfirm_Click(object sender, EventArgs e)
    {
        Response.Write("<script language=javascript> alert('Your Insurance Payment is paid
successfully'); window.location='View_Transaction.aspx' </script>");
    }
}
```

6.3.3.4 View Transactions

```
using System; using System.Collections.Generic; using
System.Linq; using System.Web; using System.Web.UI;
using System.Web.UI.WebControls; using System.Data;
public partial class View_Transaction :
System.Web.UI.Page
{
    DataLayer dl = new DataLayer();
public static string id = "";     public
void fillgrid()
    {
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            GridView2.Visible = false;
            GridView1.Visible = true;
            String str = "select * from new_saving where account_no IN(select account_id from
applicant where email='" + Session["email"].ToString() + "')";            DataSet ds = new
DataSet();            ds = dl.GetDataSet(str);            GridView1.DataSource = ds;
            GridView1.DataMember = "table";
            GridView1.DataBind();
        }
    }
    protected void rbdeposit_CheckedChanged(object sender, EventArgs e)
    {
        GridView2.Visible = false;
        GridView1.Visible = true;
        String str = "select * from new_saving where account_no IN(select account_id from
applicant where email='" + Session["email"].ToString() + "')";        DataSet ds = new
DataSet();        ds = dl.GetDataSet(str);        GridView1.DataSource = ds;
        GridView1.DataMember = "table";
        GridView1.DataBind();
    }
    protected void rbwithdrawal_CheckedChanged(object sender, EventArgs e)
    {
        GridView1.Visible = false;
        GridView2.Visible = true;
        String str = "select * from withdrawal where account_no IN(select account_id from
applicant where email='" + Session["email"].ToString() + "')";        DataSet ds = new
DataSet();        ds = dl.GetDataSet(str);        GridView2.DataSource = ds;
        GridView2.DataMember = "table";
        GridView2.DataBind();
    }
}
```

6.3.3.5 View Insurance

```
using System;
using System.Collections.Generic;
using System.Linq; using
System.Web;
using System.Web.UI; using
System.Web.UI.WebControls; using System.Data ; public
partial class View_Insurance : System.Web.UI.Page
{     public static string id = "";     protected void
Page_Load(object sender, EventArgs e)
   {         if
(!IsPostBack)
        {
            DataLayer dl = new DataLayer();
            DataSet ds = new DataSet();
            String str = "select * from insurance_transaction where account_no IN(select
account_id from applicant where email='" + Session["email"].ToString() + "')";
ds = dl.GetDataSet(str);             GridView1.DataSource = ds;
            GridView1.DataMember = "table";
            GridView1.DataBind();
                DataLayer d2 = new DataLayer();
                DataSet ds1 = new DataSet();
                String str1 = "select * from policy_insurance where account_no IN(select
account_id from applicant where email='" + Session["email"].ToString() + "')";
ds1 = d2.GetDataSet(str1);                 if (ds1.Tables[0].Rows.Count > 0)
                {
                    lblmaturity.Text = ds1.Tables[0].Rows[0]["maturity"].ToString();
                }
            }
            DataLayer d3 = new DataLayer();
            DataSet ds2 = new DataSet();
            String str2 = "select sum(amount)as amt from insurance_transaction where
account_no IN(select account_id from applicant where email='" +
Session["email"].ToString() + "')";             ds2 = d3.GetDataSet(str2);
            if (ds2.Tables[0].Rows.Count > 0)
            {
               lblamount.Text = ds2.Tables[0].Rows[0]["amt"].ToString();
            }
            }
    public void fillgrid()
    {
    }
  protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
    {
    }
}
```

6.3.3.6 View Provident Fund

```
using System; using System.Collections.Generic; using
System.Linq; using System.Web; using System.Web.UI;
using System.Web.UI.WebControls; using System.Data;
public partial class View_Provident : System.Web.UI.Page
{
    public static string id = "";     protected void
Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            fillgrid();
            DataLayer d2 = new DataLayer();
            DataSet ds1 = new DataSet();
            String str1 = "select sum(amount)as amt from provident where account_no IN(select
account_id from applicant where email='" + Session["email"].ToString() + "')";
            ds1 = d2.GetDataSet(str1);
if (ds1.Tables[0].Rows.Count > 0)
            {
               lblamt.Text = ds1.Tables[0].Rows[0]["amt"].ToString();
            }
        }     }     public
void fillgrid()
    {
        DataLayer dl = new DataLayer();
        DataSet ds = new DataSet();
        String str = "select * from provident where account_no IN(select account_id from
applicant where email='" + Session["email"].ToString() + "')";         ds =
dl.GetDataSet(str);         GridView1.DataSource = ds;
        GridView1.DataMember = "table";
        GridView1.DataBind();
    }
    protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
    {
    }
}
```

6.3.4 Forget Password

```
using System;
using System.Collections.Generic;
using System.Linq; using
System.Web; using
System.Web.UI; using
System.Web.UI.WebControls;
using System.Net.Mail; using
System.Net; using System.Data;


public partial class Forgotpass : System.Web.UI.Page
{     protected void Page_Load(object sender, EventArgs
e)
    {
    }
    protected void btnpass_Click(object sender, EventArgs e)
    {
        using (var client = new SmtpClient())
        {
            client.UseDefaultCredentials = false;             var
NetworkCredentials = new NetworkCredential() { UserName =
"sapaligasriraksha@gmail.com", Password = "mhrd hmcl jjyu rqfw" };
client.Port = 587;             client.EnableSsl = true;             client.Host =
"smtp.gmail.com";             client.Credentials = NetworkCredentials;
MailMessage msg = new MailMessage();             msg.From = new
MailAddress("sapaligasriraksha@gmail.com");
msg.To.Add(txtemail.Text);
            msg.Subject = "Recovery Email from Postal Account";
            Random r = new Random();
            String s = r.NextDouble().ToString();
s = s.Substring(4, 4);
            DataLayer dl = new DataLayer();                 DataSet ds = new DataSet();
string str = "select * from registration where email='" + txtemail.Text + "'";           ds
= dl.GetDataSet(str);             if (ds.Tables[0].Rows.Count > 0)
            {
           str = "update registration set password='" + s + "'where email='" + txtemail.Text + "'";
dl.DmlCmd(str);                 msg.Body = "Your password is"+s;                 client.Send(msg);
                Response.Write("<script language=javascript> alert('Mail has sent') </script>");
            }
else
            {
                Response.Write("<script language=javascript> alert('The Email is Invalid')
</script>");
            }
        }


    }
}
```

6.3.5 Reset Password

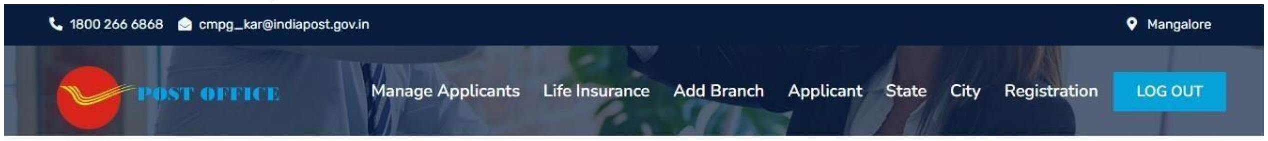
```
using System; using System.Collections.Generic;
using System.Linq; using System.Web; using
System.Web.UI; using
System.Web.UI.WebControls; public partial
class _Reset : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
}
```

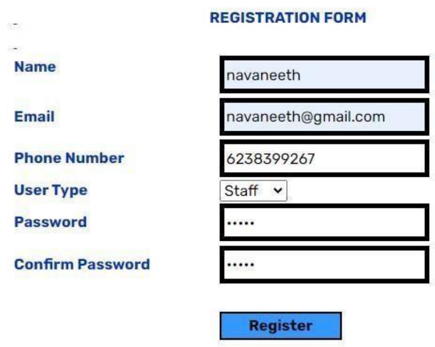
CHAPTER 7

USER INTERFACE

7.1 REGISTRATION AND LOGIN:

7.1.1 Admin/Staff Registration



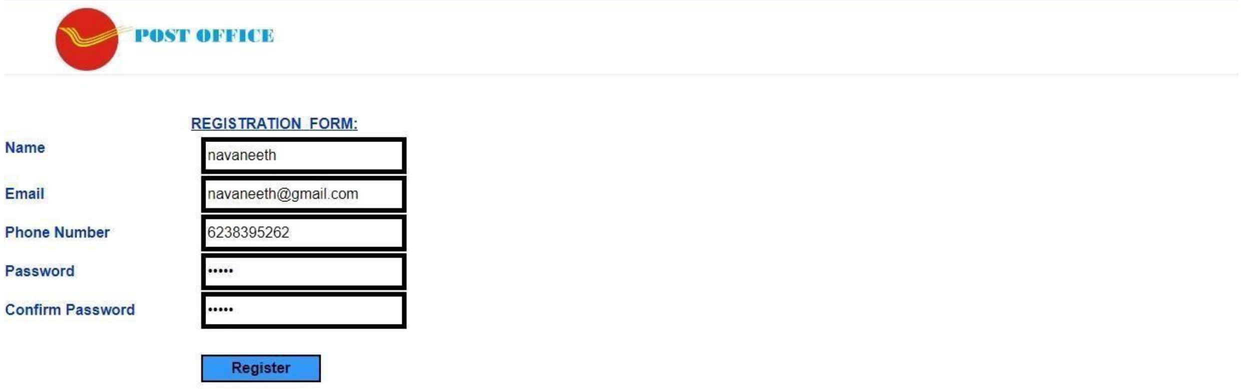


REGISTRATION FORM

Name	navaneeth
Email	navaneeth@gmail.com
Phone Number	6238399267
User Type	Staff
Password	•••••
Confirm Password	•••••

Fig 7.1 Admin and staff registration

7.1.2 User Registration:



REGISTRATION FORM:

Name	navaneeth
Email	navaneeth@gmail.com
Phone Number	6238395262
Password	•••••
Confirm Password	•••••

Fig 7.2 User registration

7.1.3 Applicant/Admin/Staff Login:

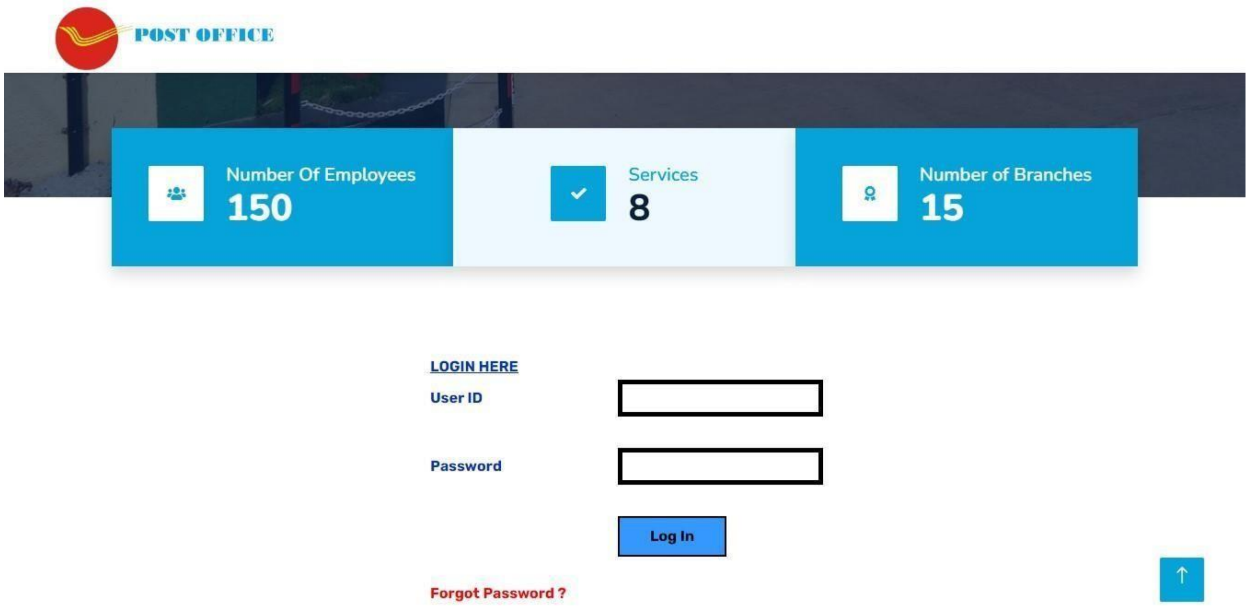
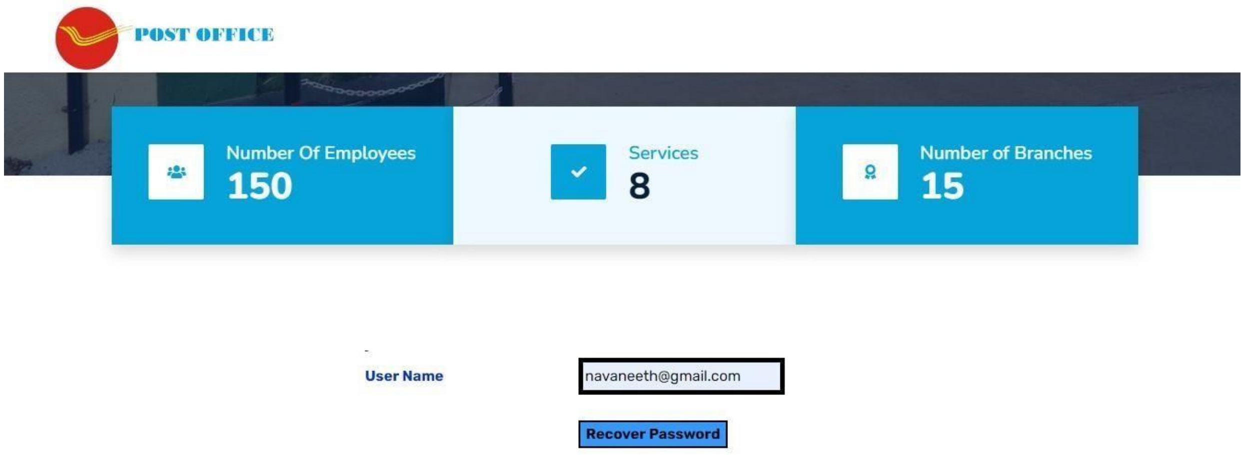


Fig 7.3 Applicant/Admin/Staff login

7.1.4 Forgot Password



navaneeth@gmail.com

Fig 7.4 Forgot Password

7.1.5 Change Password

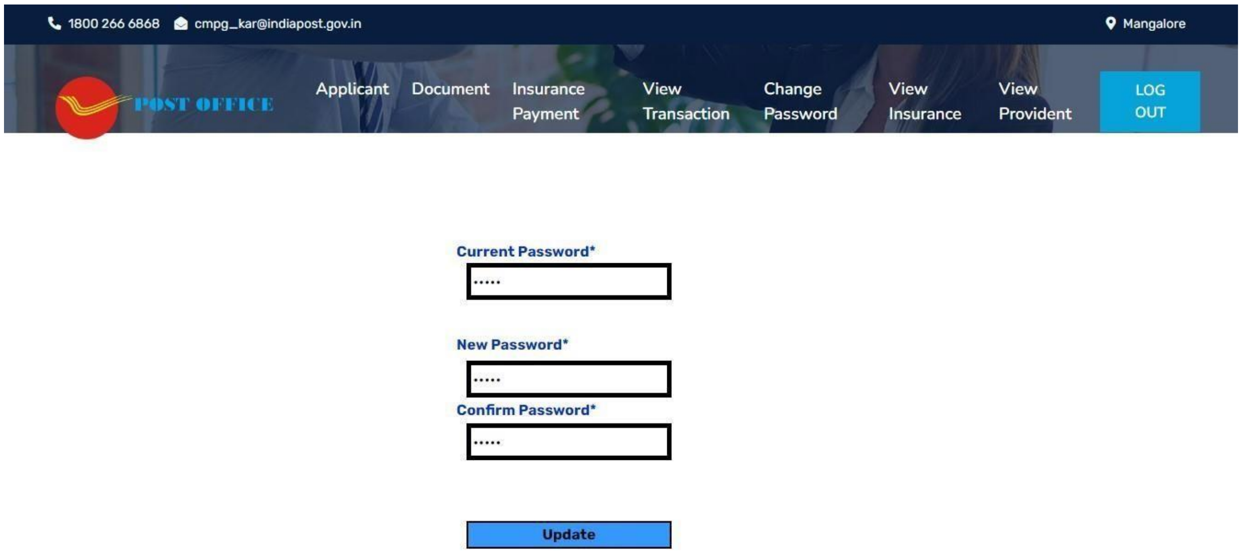


Fig 7.5 Change password

7.2 MAIN SCREEN:

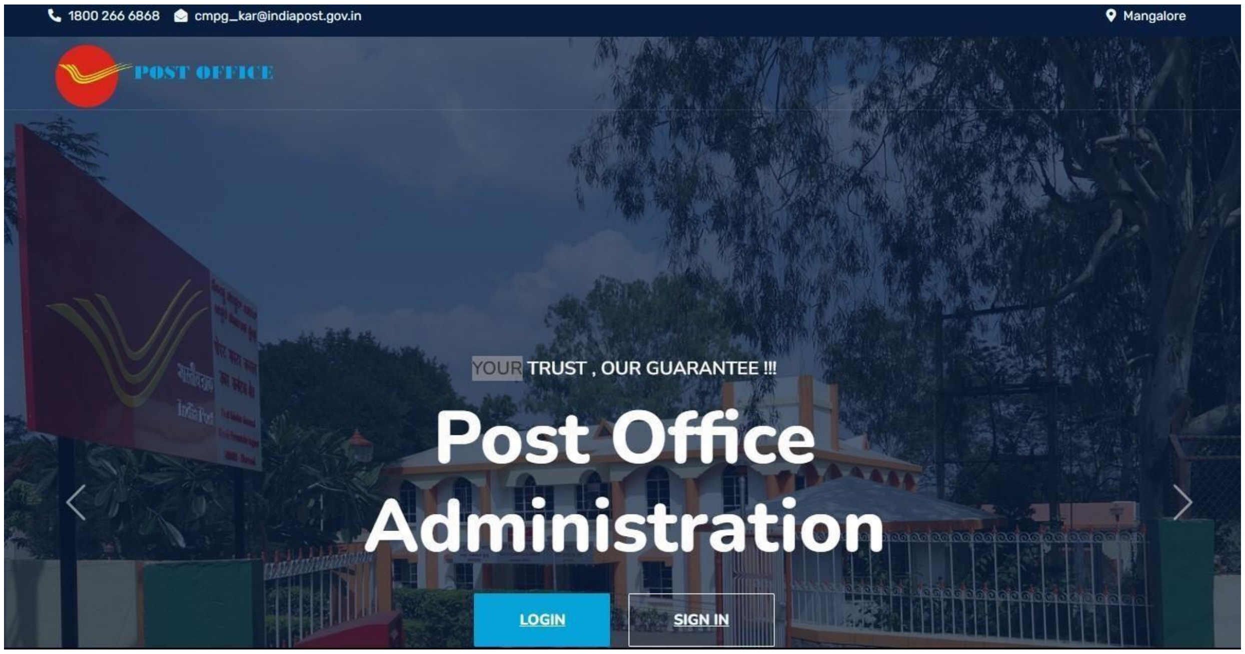
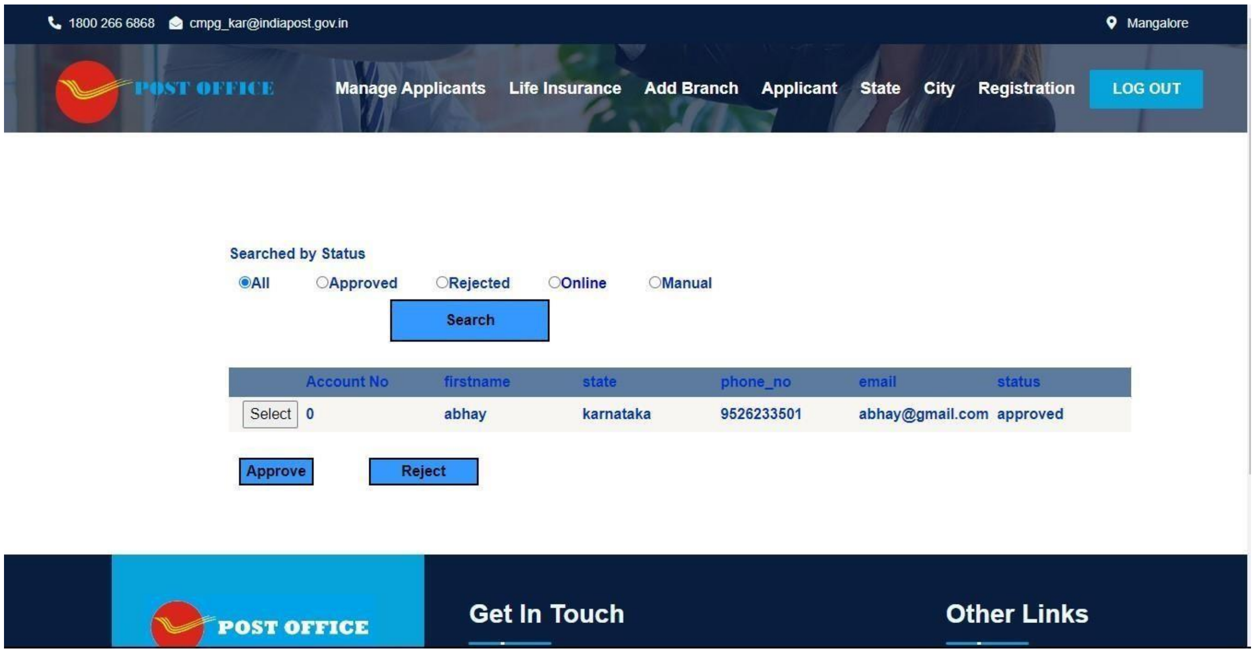

1800 266 6868
cmpg_kar@indiapost.gov.in
Mangalore
POST OFFICE
YOUR TRUST , OUR GUARANTEE !!!
Post Office Administration
LOGIN
SIGN IN

Fig 7.6 Main Screen

7.2 ADMIN MODULES:

7.2.1 Manage Applicants



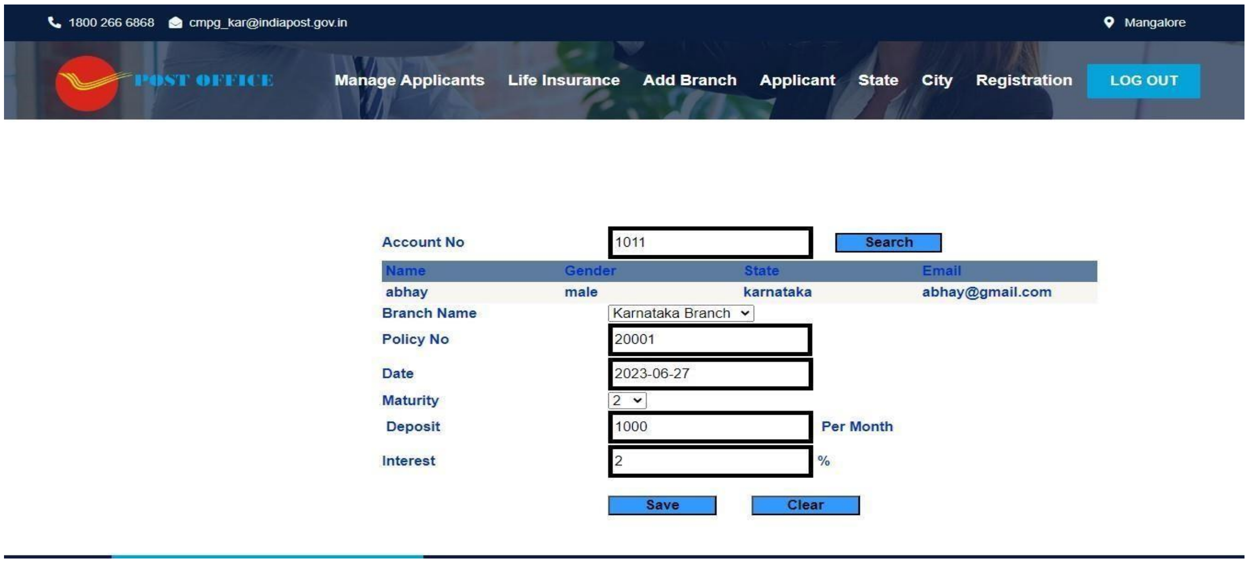
Searched by Status

[✓ All] Approved Rejected Online Manual

	Account No	firstname	state	phone_no	email	status
Select	0	abhay	karnataka	9526233501	abhay@gmail.com	approved

Fig 7.7 Manage applications

7.2.2 Life Insurance:



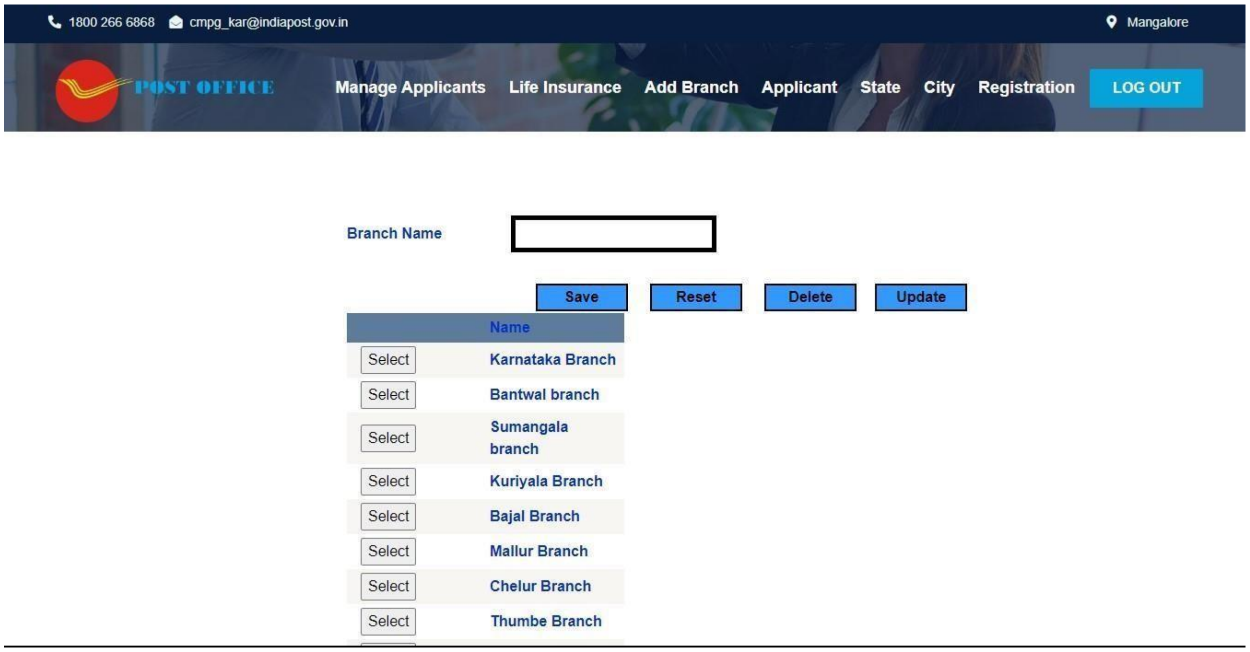
Account No 1011

Name	Gender	State	Email
abhay	male	karnataka	abhay@gmail.com

Branch Name	Karnataka Branch
Policy No	20001
Date	2023-06-27
Maturity	2
Deposit	1000 Per Month
Interest	2 %

Fig 7.8 Life insurance

7.2.2 Add Branch:

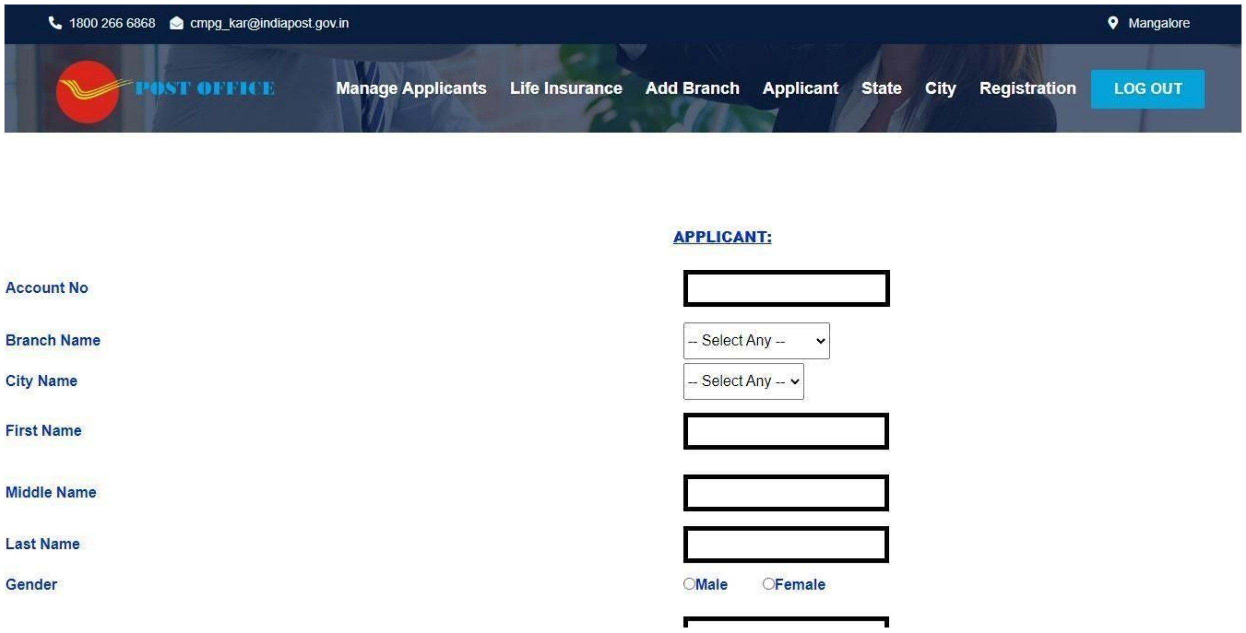


Branch Name

	Name
Select	Karnataka Branch
Select	Bantwal branch
Select	Sumangala branch
Select	Kuriyala Branch
Select	Bajal Branch
Select	Mallur Branch
Select	Chelur Branch
Select	Thumbe Branch

Fig 7.9 Add branch

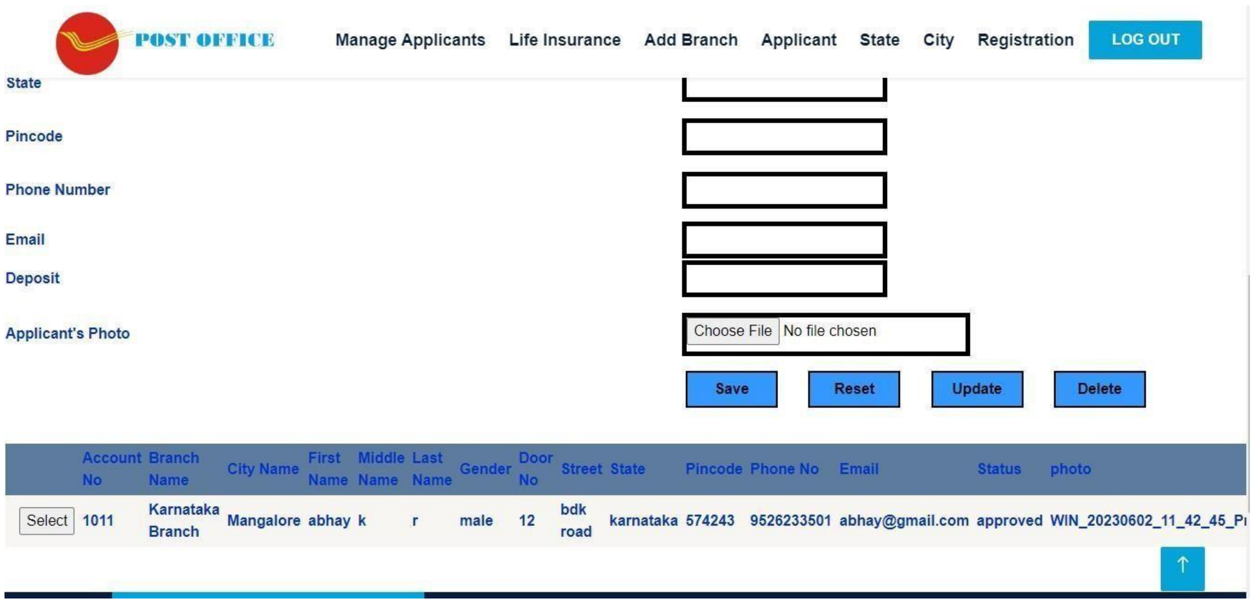
7.2.3 Applicant:



APPLICANT:

Account No	
Branch Name	-- Select Any --
City Name	-- Select Any --
First Name	
Middle Name	
Last Name	
Gender	○Male ○Female

Fig 7.10(1) Applicant



State

Pincode

Phone Number

Email

Deposit

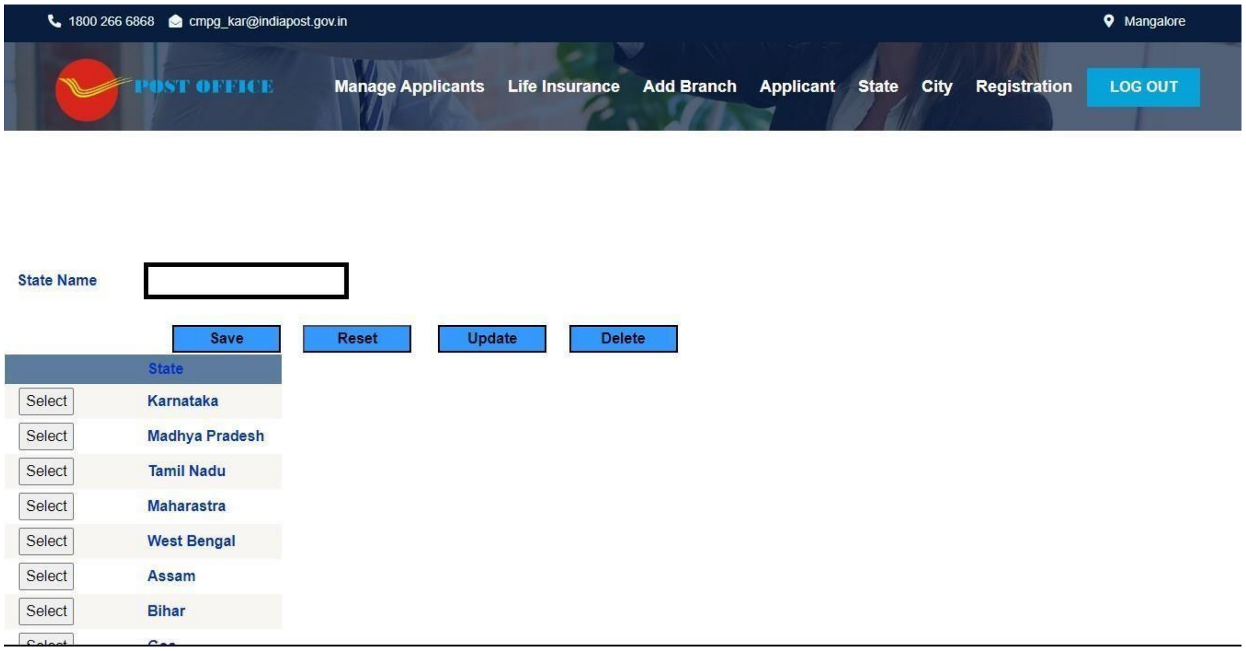
Applicant's Photo — Choose File No file chosen

Save Reset Update Delete

	Account No	Branch Name	City Name	First Name	Middle Name	Last Name	Gender	Door No	Street	State	Pincode	Phone No	Email	Status	photo
Select	1011	Karnataka Branch	Mangalore	abhay	k	r	male	12	bdk road	karnataka	574243	9526233501	abhay@gmail.com	approved	WIN_20230602_11_42_45_Pı

Fig 7.10(2) Applicant

7.2.4 State:



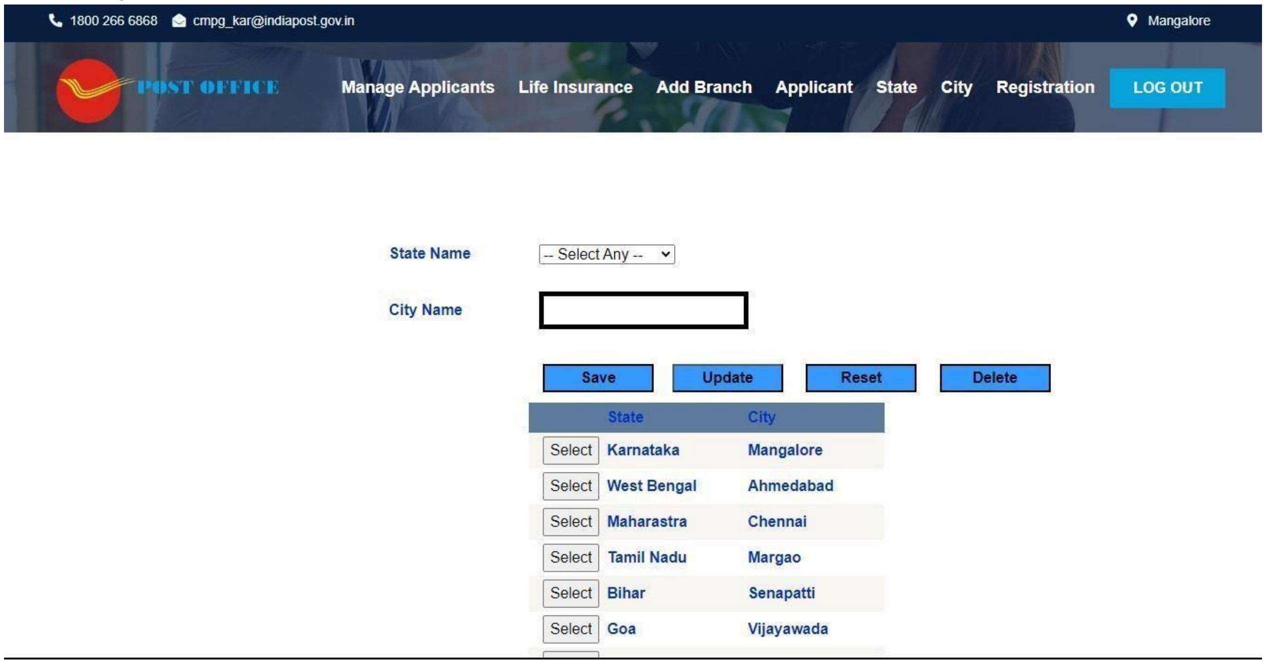
State Name

Save Reset Update Delete

	State
Select	Karnataka
Select	Madhya Pradesh
Select	Tamil Nadu
Select	Maharastra
Select	West Bengal
Select	Assam
Select	Bihar

Fig 7.11 State

7.2.5 City:



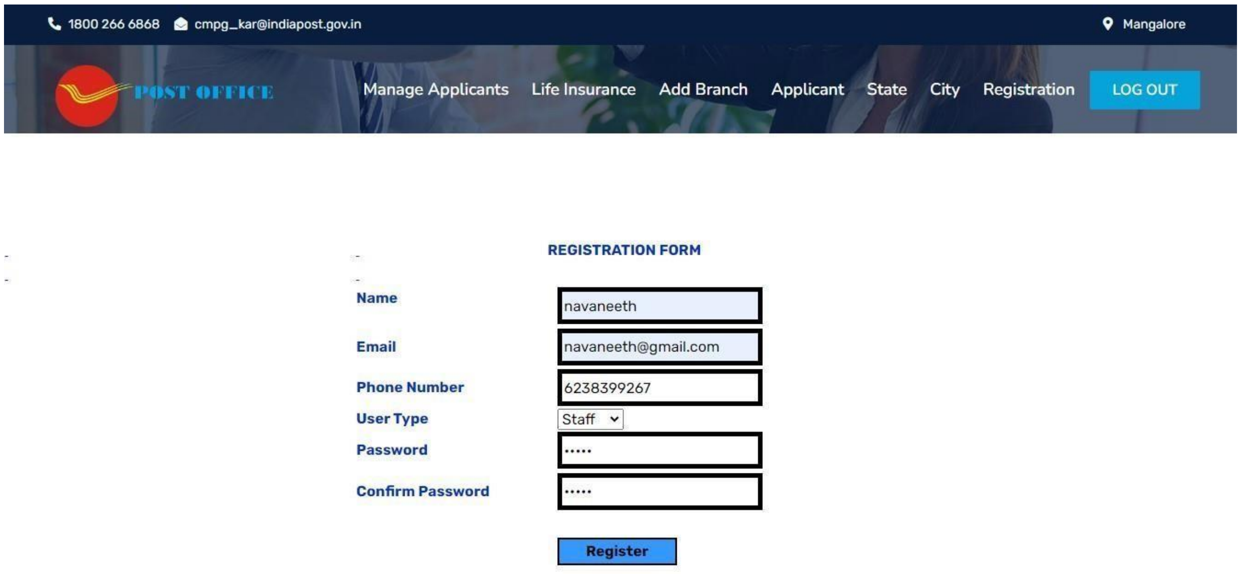
State Name: -- Select Any --

City Name:

	State	City
Select	Karnataka	Mangalore
Select	West Bengal	Ahmedabad
Select	Maharastra	Chennai
Select	Tamil Nadu	Margao
Select	Bihar	Senapatti
Select	Goa	Vijayawada

Fig 7.12 City

7.2.6 Registration:



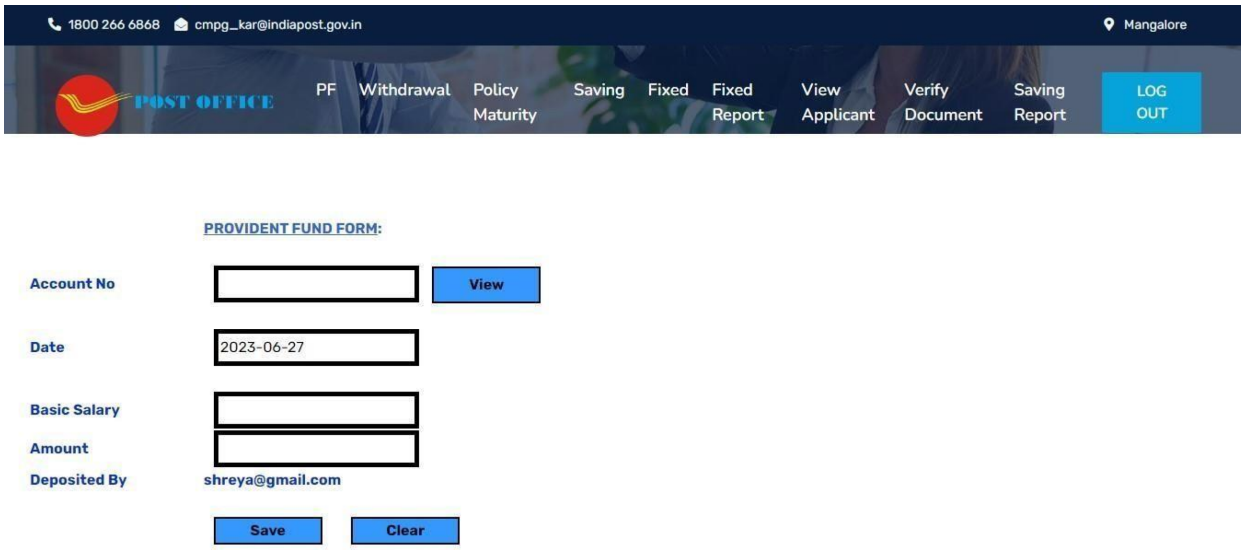
REGISTRATION FORM

Name	navaneeth
Email	navaneeth@gmail.com
Phone Number	6238399267
User Type	Staff
Password	•••••
Confirm Password	•••••

Fig 7.13 Registration

7.3 STAFF MODULES:

7.3.1 Provident Fund:

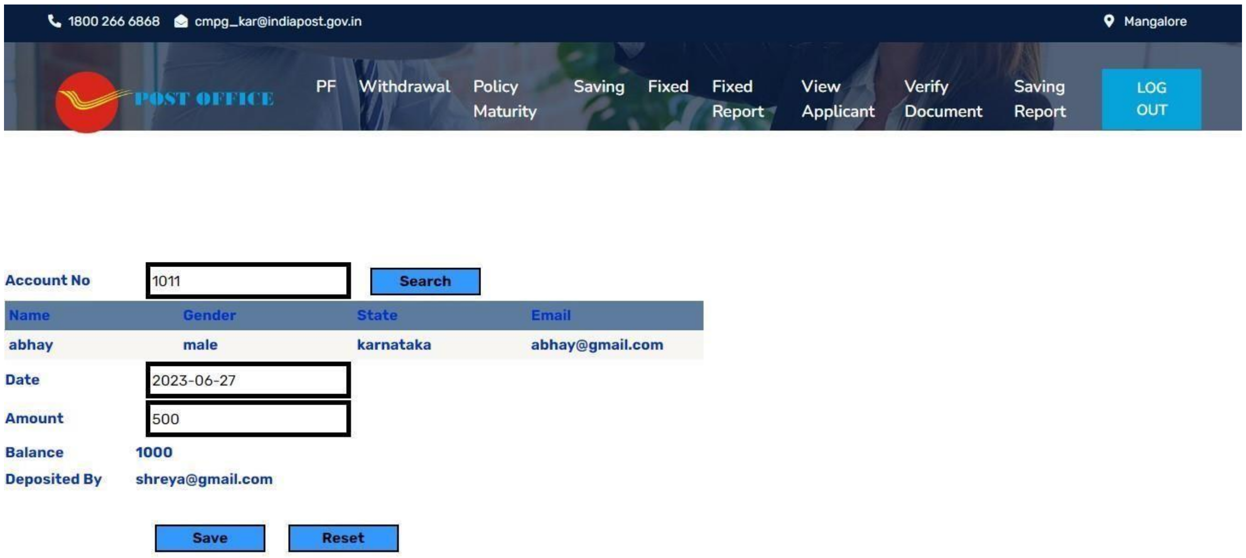


PROVIDENT FUND FORM:

Account No	
Date	2023-06-27
Basic Salary	
Amount	
Deposited By	shreya@gmail.com

Fig 7.14 Provident Fund

7.3.2 Withdrawal:



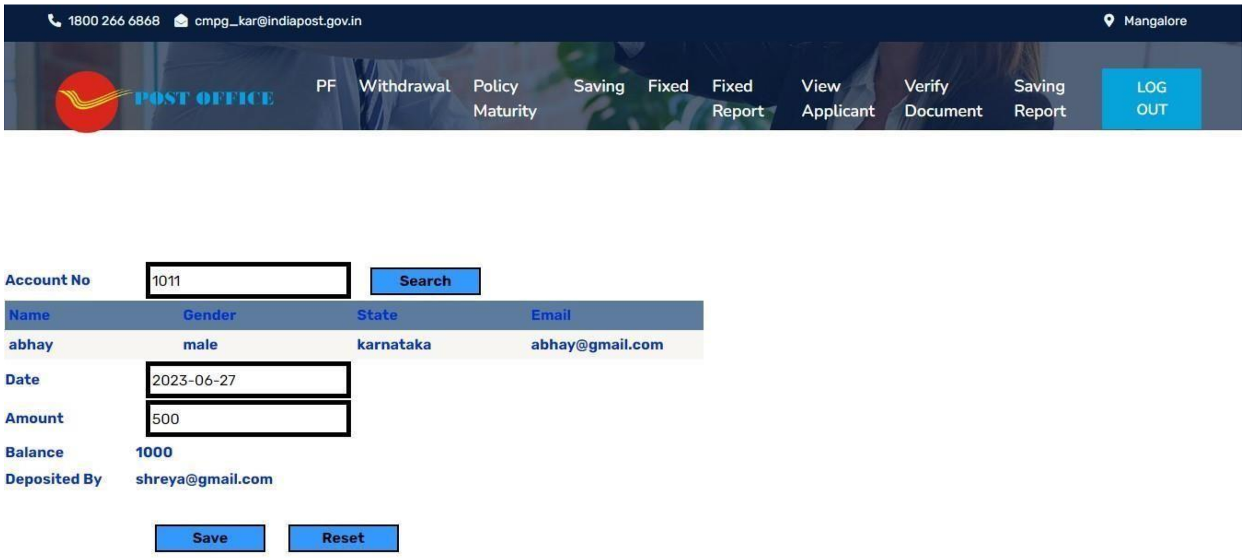
Account No 1011

Name	Gender	State	Email
abhay	male	karnataka	abhay@gmail.com

Date	2023-06-27
Amount	500
Balance	1000
Deposited By	shreya@gmail.com

Fig 7.15 Withdrawal

7.3.3 Policy Maturity:



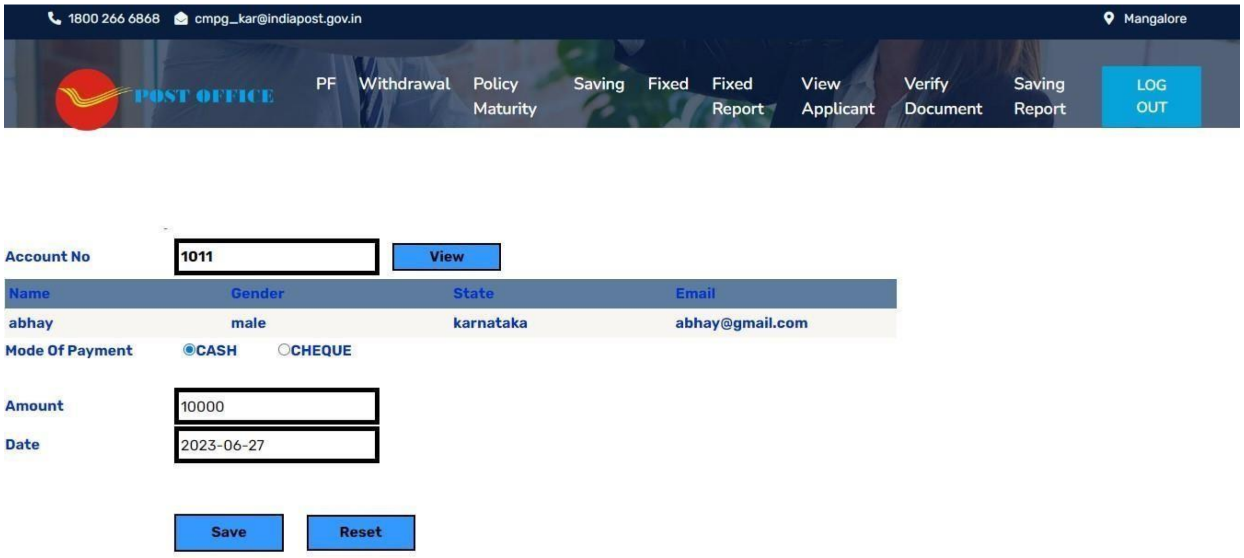
Account No	1011

Name	Gender	State	Email
abhay	male	karnataka	abhay@gmail.com

Date	2023-06-27
Amount	500
Balance	1000
Deposited By	shreya@gmail.com

Fig 7.16 Policy maturity

7.3.4 Saving:



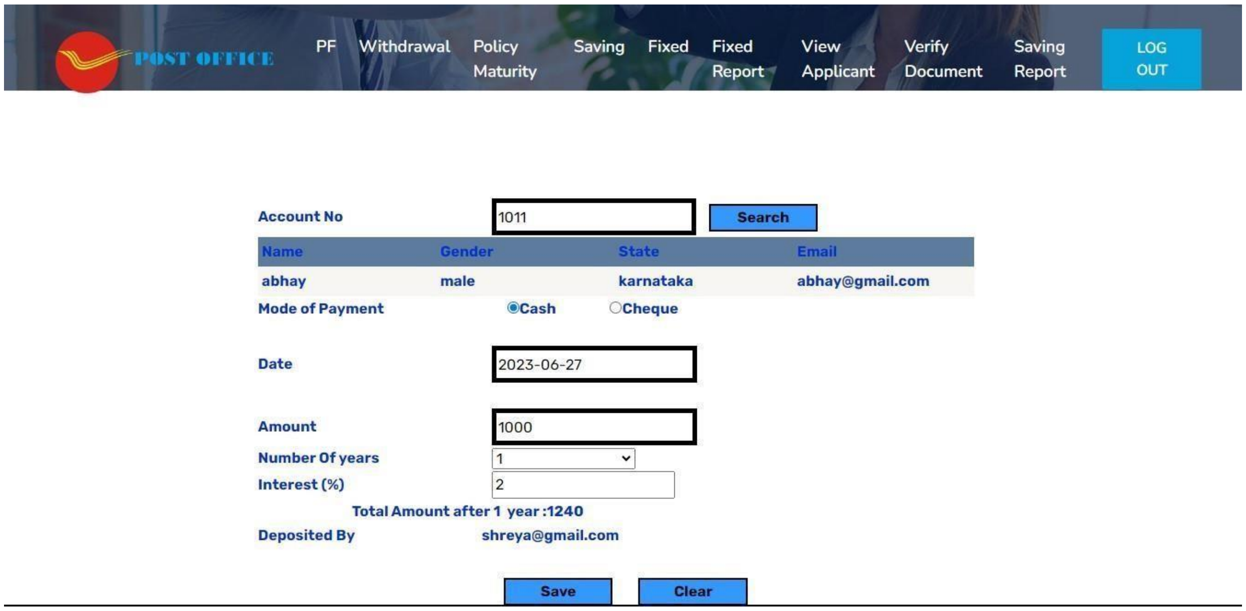
Account No	1011

Name	Gender	State	Email
abhay	male	karnataka	abhay@gmail.com

Mode Of Payment	[✓ CASH] CHEQUE
Amount	10000
Date	2023-06-27

Fig 7.17 Saving

7.3.5 Fixed:



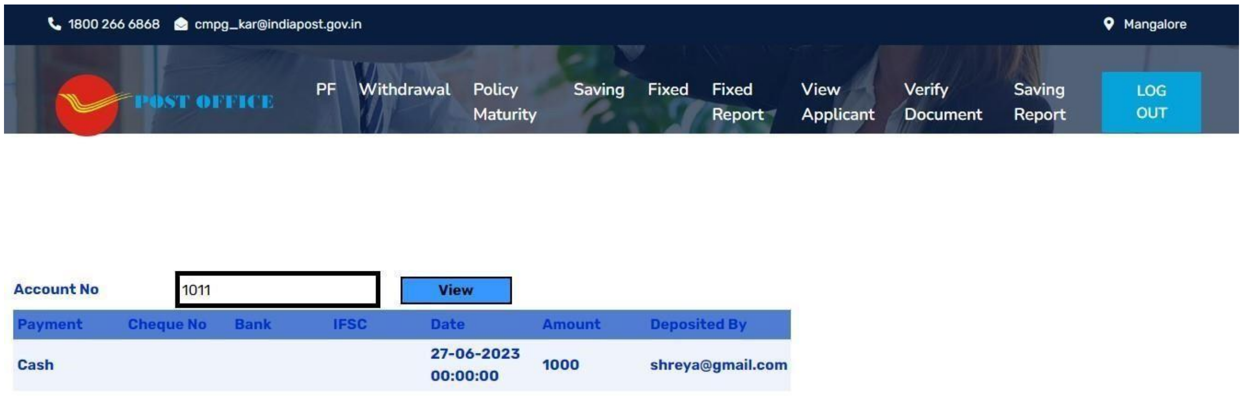
Account No	1011		
Name	Gender	State	Email
abhay	male	karnataka	abhay@gmail.com
Mode of Payment	[✓ Cash]	Cheque	
Date	2023-06-27		
Amount	1000		
Number Of years	1		
Interest (%)	2		

Total Amount after 1 year :1240

Deposited By shreya@gmail.com

Fig 7.18 Fixed

7.3.6 Fixed Report:

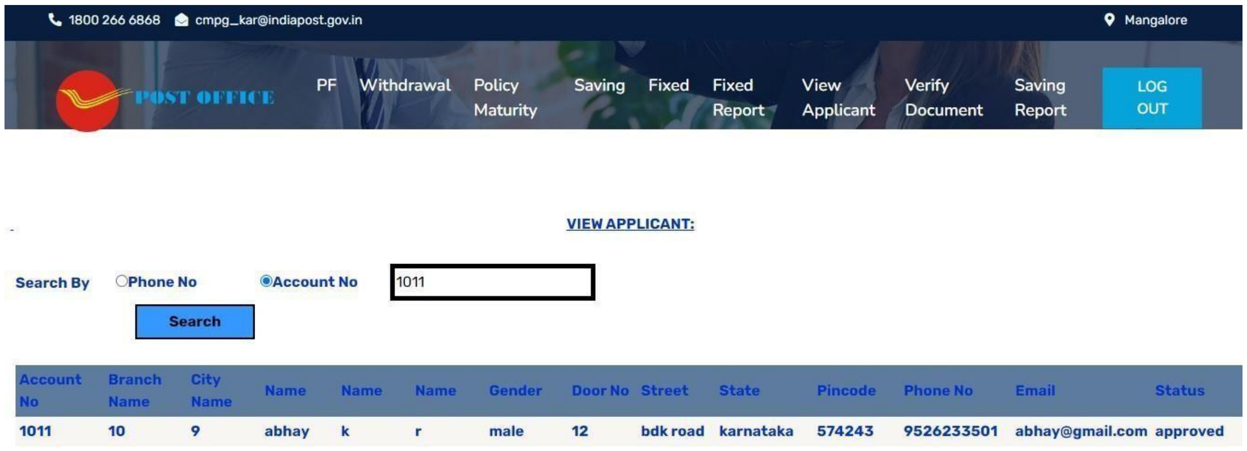


Account No 1011

Payment	Cheque No	Bank	IFSC	Date	Amount	Deposited By
Cash				27-06-2023 00:00:00	1000	shreya@gmail.com

Fig 7.19 Fixed report

7.3.7 View Applicant:



VIEW APPLICANT:

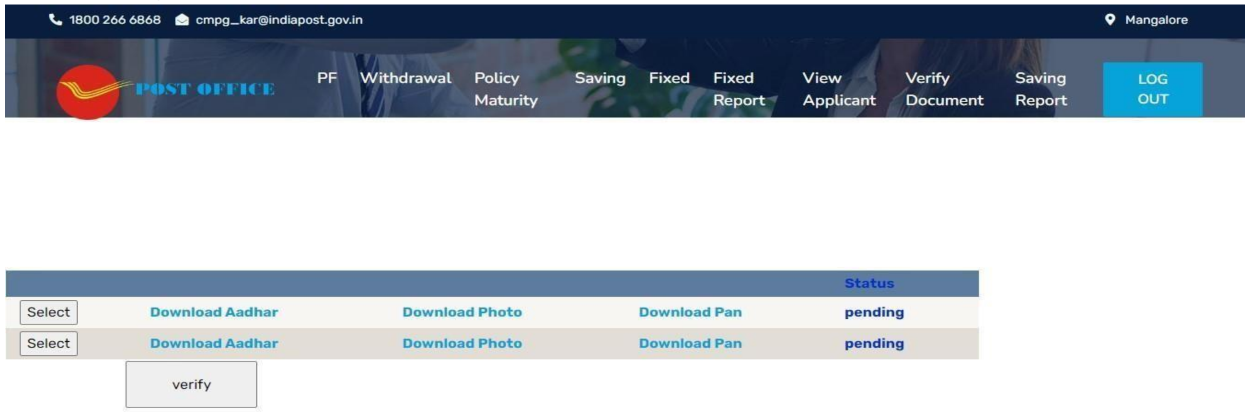
Search By ○Phone No ◉Account No 1011

Search

Account No	Branch Name	City Name	Name	Name	Name	Gender	Door No	Street	State	Pincode	Phone No	Email	Status
1011	10	9	abhay	k	r	male	12	bdk road	karnataka	574243	9526233501	abhay@gmail.com	approved

Fig 7.20 View Applicant

7.3.8 Verify Document:

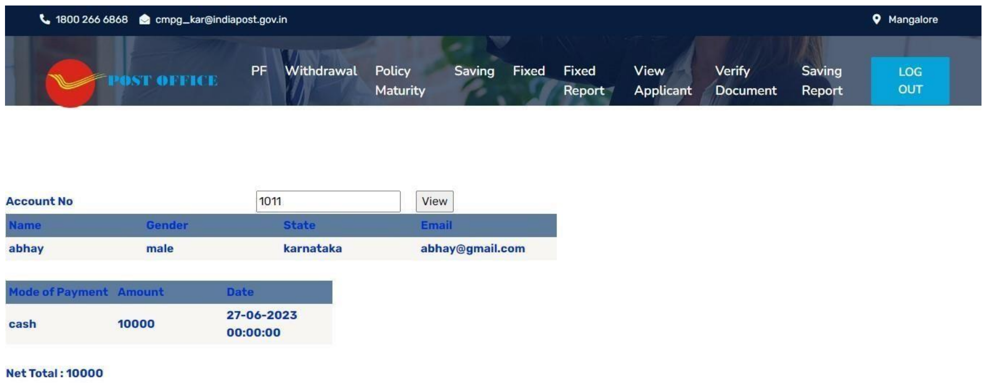


				Status
Select	Download Aadhar	Download Photo	Download Pan	pending
Select	Download Aadhar	Download Photo	Download Pan	pending

verify

Fig 7.21 Verify document

7.3.9 Saving Report:



Account No 1011 View

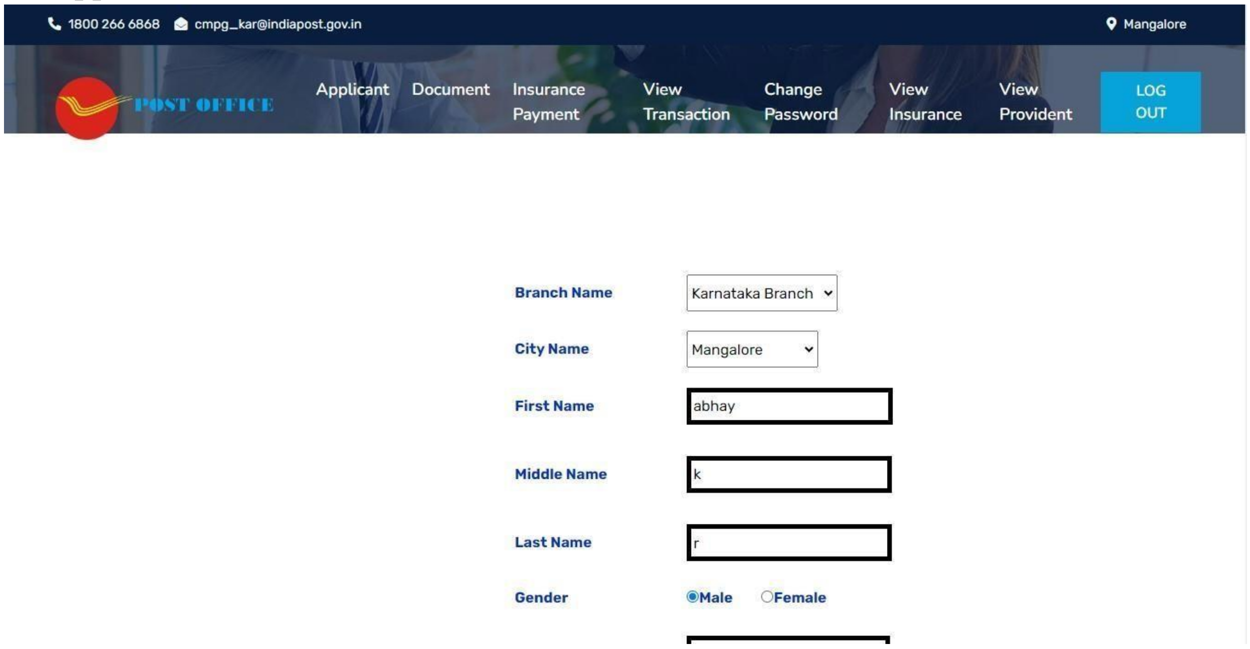
Name	Gender	State	Email
abhay	male	karnataka	abhay@gmail.com

Mode of Payment	Amount	Date
cash	10000	27-06-2023 00:00:00

Net Total : 10000

Fig 7.22 Saving Report

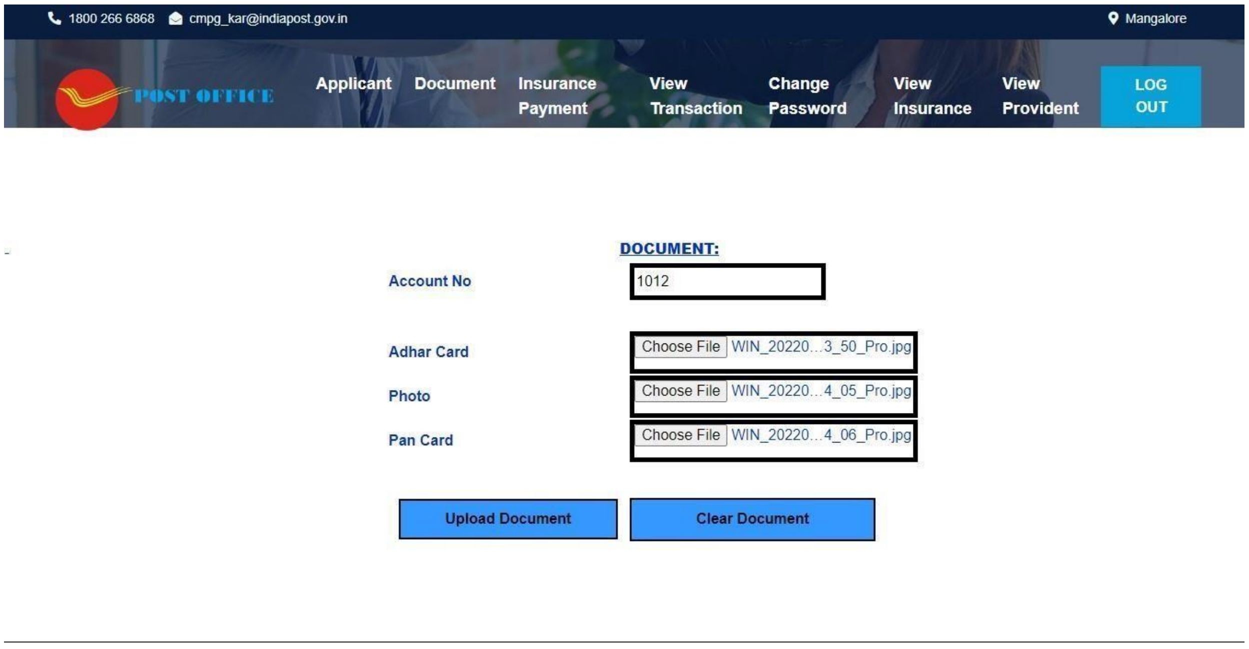
7.4 Applicant:



Branch Name	Karnataka Branch
City Name	Mangalore
First Name	abhay
Middle Name	k
Last Name	r
Gender	[✓ Male] Female

Fig 7.22 Applicant

7.4.2 Document:

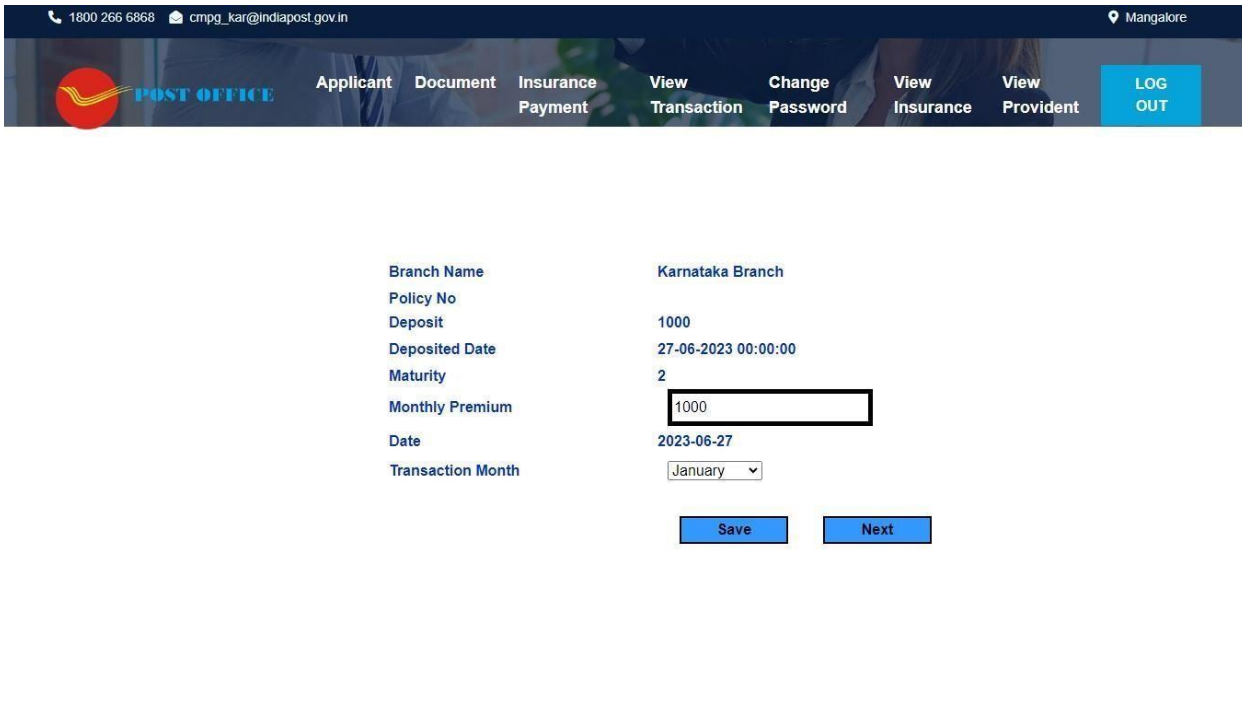


DOCUMENT:

Account No	1012
Adhar Card	Choose File WIN_20220...3_50_Pro.jpg
Photo	Choose File WIN_20220...4_05_Pro.jpg
Pan Card	Choose File WIN_20220...4_06_Pro.jpg

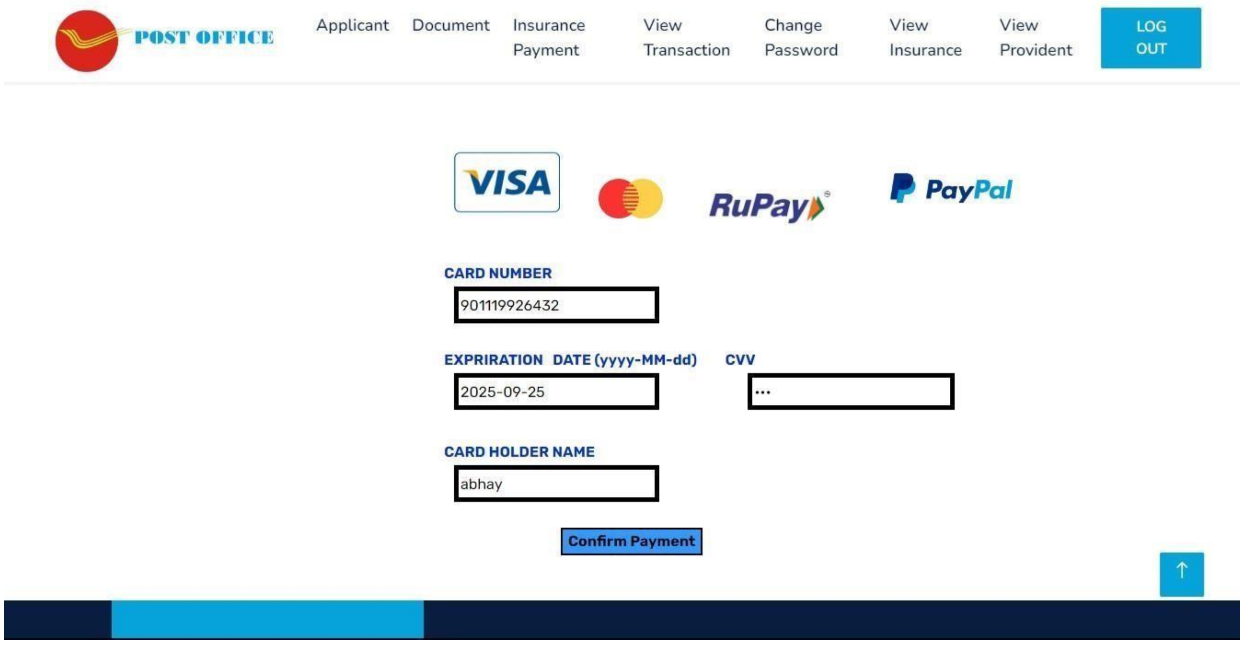
Fig 7.23 Document

7.4.3 Insurance Payment:



Branch Name	Karnataka Branch
Policy No	
Deposit	1000
Deposited Date	27-06-2023 00:00:00
Maturity	2
Monthly Premium	1000
Date	2023-06-27
Transaction Month	January

Fig 7.24(1) Insurance payment



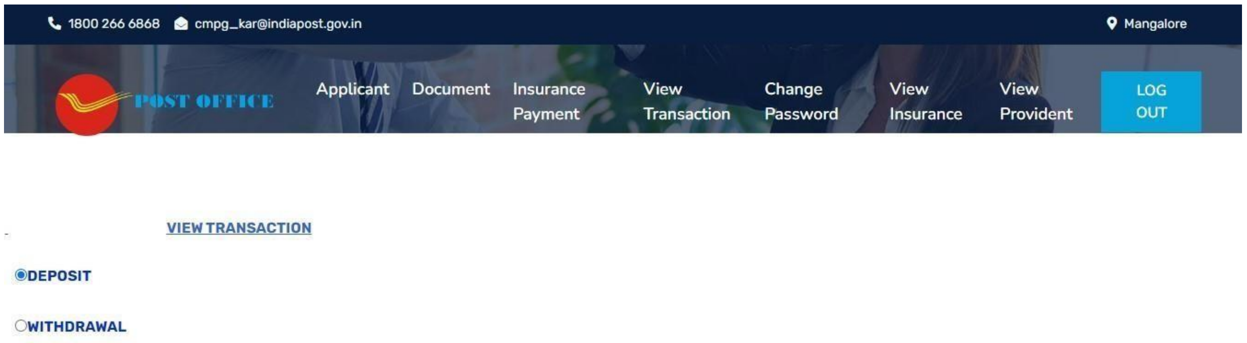
CARD NUMBER
901119926432

EXPRIRATION DATE (yyyy-MM-dd) 2025-09-25
CVV •••

CARD HOLDER NAME
abhay

Fig 7.24(1) Insurance payment

7.4.4 View Transaction:



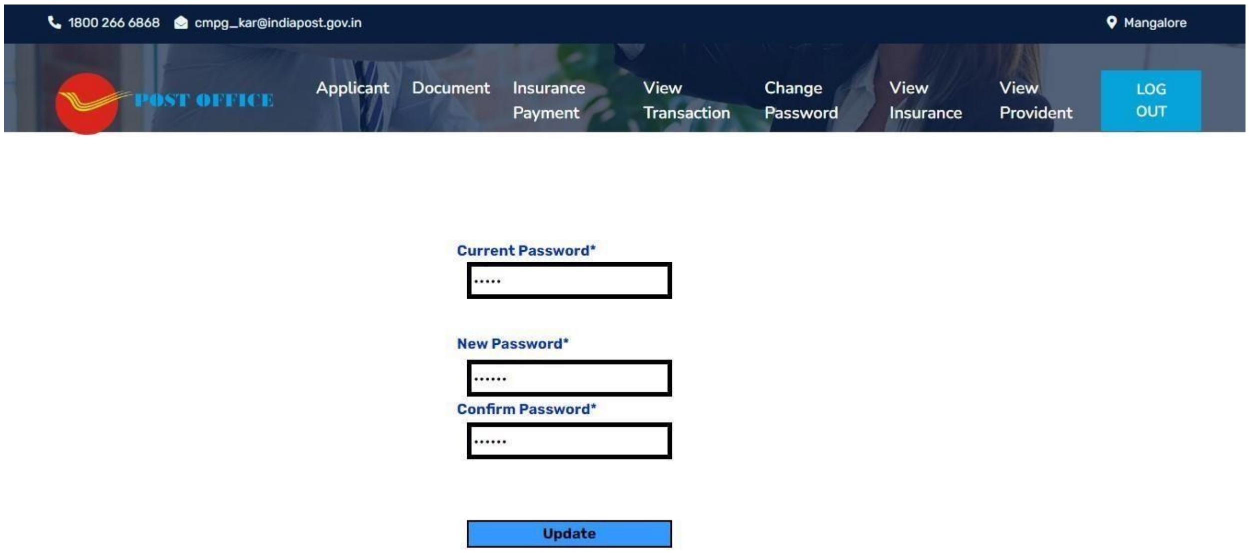
VIEW TRANSACTION

[✓ DEPOSIT]

WITHDRAWAL

Fig 7.25 View Transaction

7.4.5 Change Password:



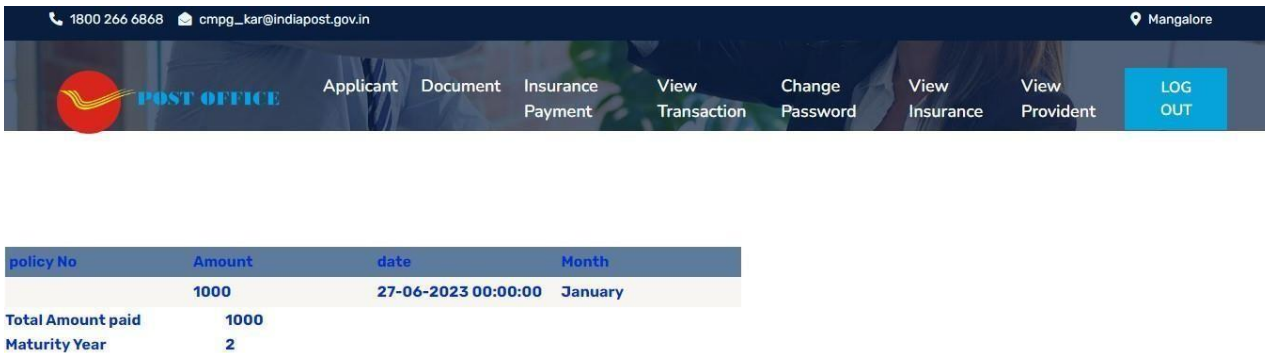
Current Password* ·····

New Password* ······

Confirm Password* ······

Fig 7.26 Change password

7.4.6 View Insurance:

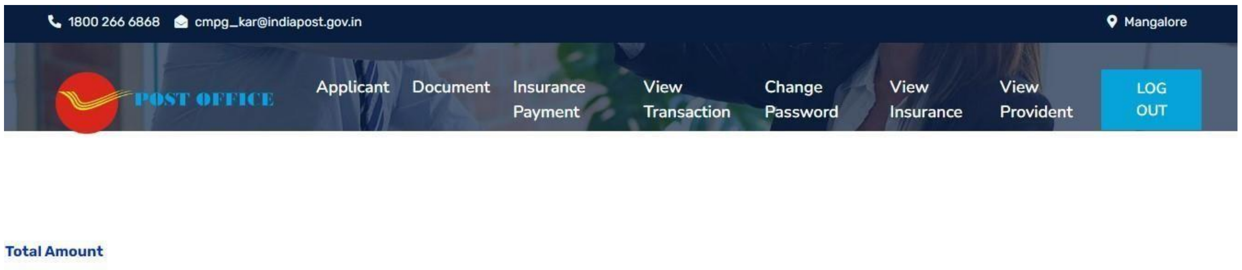


policy No	Amount	date	Month
	1000	27-06-2023 00:00:00	January

Total Amount paid	1000
Maturity Year	2

Fig 7.27 View Insurance

7.4.7 View Provident:

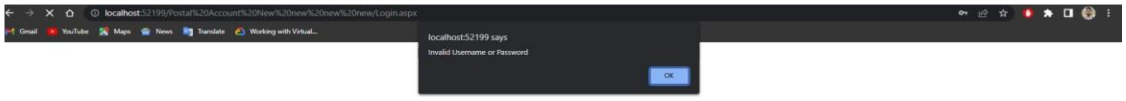


Total Amount

Fig 7.28 View provident

7.6 Validation:

7.6.1 Login:


localhost:52199/Postal%20Account%20New%20new%20new%20new/Login.aspx
Gmail
YouTube
Maps
News
Translate
Working with Virtual...

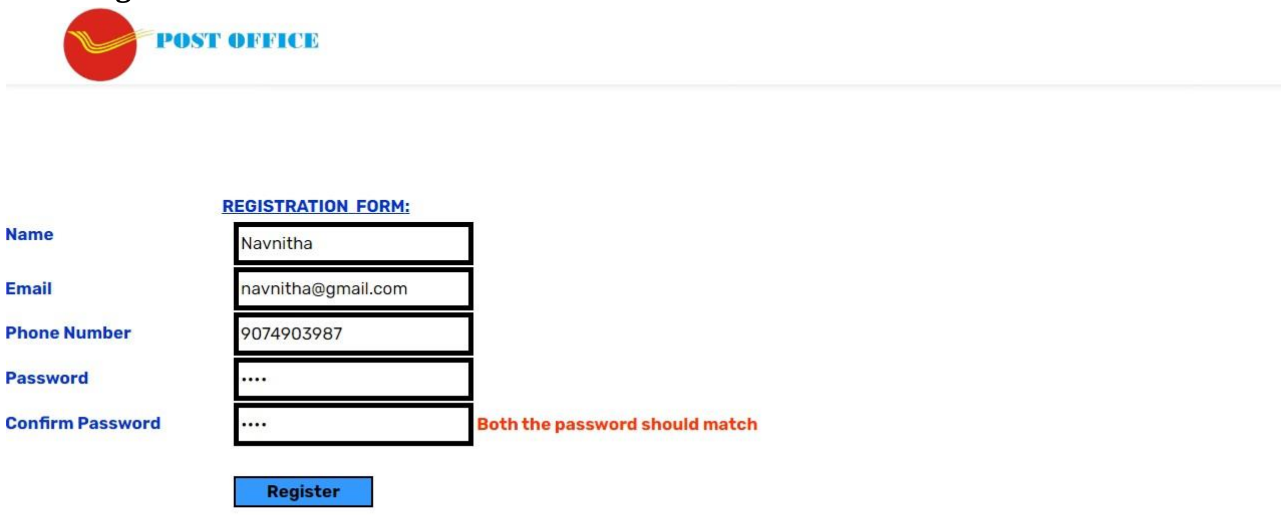
localhost:52199 says

Invalid Username or Password

OK

Fig 7.29 Login

7.6.2 Registration:



REGISTRATION FORM:

Name	Navnitha
Email	navnitha@gmail.com
Phone Number	9074903987
Password	••••
Confirm Password	•••• Both the password should match

Fig 7.30 Registration

CHAPTER 8

TESTING

8.1 Introduction:

Software Testing is the process used to help identify the correctness, completeness, security and quality of developed computer software. This includes the process of executing the program or application with the intent of finding errors. Quality is not an absolute; it is value to some person. With that in mind testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behavior of the product against a specification.

The testing phase consists of evaluating the software that has been developed in order to conform that it produces the output required in a safe and efficient manner. In this phase inherent errors that occur, have to be handled and the user should be informed so that he/she can follow the guidelines and instructions and get around the error and obtain the output. During testing, the program to be tested is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as expected. Due to its approach, dynamic testing can only ascertain the presence of errors in the program the exact nature of the errors is not usually decided by testing.

Testing forms the first step in determining the errors in a program. Clearly the success of testing in revealing errors in programs depends critically on the test cases. Because code is the only product that can be executed and whose actual behavior can be observed, testing is the phase where the errors remaining from all the previous phases must be detected.

The program to be tested is executed with a set of test cases and the output of the program for the test cases are evaluated to determine if the programming is performing as expected. Testing forms the first step in determining errors in a program. The success of testing in reveling errors in programs depends critically on the test cases.

8.2 Test Reports:

8.2.1 Unit Testing:

Unit testing focuses efforts on the smallest unit of software design this is known as module testing or white box testing, the modules are tested separately. The test is carried out during the programming stage itself. In this step, each module is found to be working satisfactorily as regards to the expected output from the module.

8.2.2 Integration Testing:

In integration testing the different units of the system are integrated together to form the complete system. This type of testing checks the system as a whole to ensure that it is doing what it's supposed to do the testing of an integrated system can be carried out top-down, bottom-up or Big-Bang. In this type of testing some parts are tested with white box testing and some with black box testing techniques. This type of testing plays a very important role in increasing systems productivity. We have checked the system by using integration testing techniques.

8.2.3 Acceptance Testing:

Apart from testing the system to validate the functionality of software against the requirements. It is also necessary to test the non-functional aspect of the system, some examples of nonfunctional tools include tests to check the performance data security usability volume load and stress that we have used in a project to test the various module. System testing consists of the following steps:

- ➢ Program(s) testing
- ➢ String testing
- ➢ System Testing
- ➢ System Documentation
- ➢ User Acceptance test

Testing Criteria:

- Testing for valid user name:

Test case	Input	Test description	Output
1	User name starts with number	User name cannot start with number	Must Enter Characters
2	User name is left blank	User name cannot be left blank	Must Enter username

- Testing for valid password:

Test case	Input	Test description	Output
1	Password is left blank	Password cannot be blank	Must Enter password
2	Invalid password entered	Valid password must be entered	Password mismatch

- Testing for valid Email address:

Test case	Input	Test description	Output
1	Email address is not in Correct format	Email address Should have Correct format	Invalid Expression
2	Email address with space	Email address cannot have space	Invalid Expression
3	Email is left blank	Email cannot be blank	Must Enter Email ID

- Testing for data insertion:

Test case	Input	Test description	Output
1	Mandatory fields left empty	Mandatory fields cannot be left empty	Must enter data
2	Duplicate entry	Duplicate entry not allowed	Appropriate error message
3	Input without above faults	Valid input	Record inserted Successfully

- Testing for deletion:

Test case	Input	Test description	Output
1	Deletion attempted when no entries selected	Entries, if not selected, cannot be deleted	Select any field
2	Valid deletion without above faults	Valid deletion	Record deleted Successfully

- Testing for phone number:

Test case	Input	Test description	Output
1	Phone number entered with alphabets	Phone number Cannot have alphabets	Enter only numbers
2	Phone number entered is more than 10 digits	Phone number with more than 10 digits cannot be entered	Invalid phone number

- Testing for change password:

Test case	Input	Test description	Output
1	Any field left blank	All fields are compulsory	Must enter Password
2	Invalid password	Valid password must be entered	Enter correct password
3	Retyped password does not match	Retyped password must match	Password mismatch
4	Valid input	Valid input	Password changed successfully

CHAPTER 9
CONCLUSION

Post office deposits are considered to be safe and secure mode of savings. The whole objective of the project is to provide the risk free postal financial features. Post office accounts offer a specified return on investment and preferred by senior citizens and individuals who want risk free investments. Hence the project has been developed where it helps in easy functioning of the work where it benefits the customers as well as the staff.

CHAPTER 10

LIMITATIONS

- Participants may not be truthful in their reporting so there may be questions over their validity.
- It doesn’t provide accurate analysis because of not enough resources
- Public can’t use this information and public can’t complaint to police and there no public involved in it
- It can’t be operated by other officers who are not given access and that sometimes is not helpful because some officers need case information but due to restrictions, they can’t access it.

CHAPTER 11

SCOPE FOR ENHANCEMENT

- In future we’ll convert it into a mobile application.
- In future this application can extend to all features provided by postal office.
- Tracker can be added to track the post.
- Payroll, Accounts application can be integrated.

CHAPTER 12

ABBREVIATIONS AND ACRONYMS

DBMS	Database Management System
RDBMS	Relational Database Management System
SRS	Software Requirement Specification
ADMIN	The Administrator.
SQL	Structured Query Language.
HTML	Hyper Text Markup Language.
CSS	Cascading Style Sheet.

CHAPTER 13

BIBLIOGRAY/ REFERENCES

13.1 Book References:

The Following are the books and the references that were referred during the development period of the project:

- PankajJalote An integrated approach to software engineering – 1991
  Edition- Narosa Publishing House, New Delhi
- Igor Hawryszkiewyez- Introduction to system analysis and design, 4th edition.
  Published by AsokeK.Ghosh, Prentice Hall of India Private Limited.
- Bill Evjen, Bill Hollis, Sheldon, Sharkey – Visual Basic 2008.
  Wiley Publishing Inc., New Delhi, Visual Basic Studio, working with Windows.
- Simon Robinson – Professional C#, 3rd Edition.
- C# Programming Language by E.Balaguruswamy

13.2 Website References:

- C# by Microsoft
- www.Google.com ➢ www.wikipedia.com
- www.msdn.microsoft.com/liberies
- www.codeproject.com
- www.stackoverflow.com