High Level Design & Low Level Design

The purpose of this document is to provide a template for documenting both HLD & LLD.

Document Control:

Project Revision History

Date	Version	Author	Brief Description of Changes	Approver Signature
19/08/2022	1.0	Group 1	System Requirement Study (SRS)	
22/08/2022	2.0	Group 1	Flowchart prepared	
23/08/2022	3.0	Group 1	ER Diagram	
24/08/2022	4.0	Group 1	Flow of Program	
26/08/2022	5.0	Group 1	Admin, Employee, Customer login logic	
27/08/2022	6.0	Group 1	All Structures are made	
29/08/2022	7.0	Group 1	Implementing linked list for reading and writing data into file	
02/09/2022	8.0	Group 1	Functions for all menus	
03/09/2022	9.0	Group 1	Validations were added for all the User Input data	
05/09/2022	10.0	Group 1	Display functions completed for all the Users	
06/09/2022	11.0	Group 1	Update functions completed for Employee	
07/09/2022	12.0	Group 1	Integration of all the modules	
08/09/2022	13.0	Group 1	tools for memory leak check were done	
09/09/2022	14.0	Group 1	Final System Check	

1. INTRODUCTION **6**

1.1. INTENDED AUDIENCE 6
1.2. ACRONYMS/ABBREVIATIONS 6
1.3. PROJECT PURPOSE 6
1.4. KEY PROJECT OBJECTIVES 7
1.5. PROJECT SCOPE AND LIMITATION 7
1.5.1. In Scope *7*
1.5.2. Out of scope *8*
1.6. FUNCTIONAL OVERVIEW 8
1.7. ASSUMPTIONS, DEPENDENCIES & CONSTRAINTS 8

2. DESIGN OVERVIEW **8**

2.1. DESIGN OBJECTIVES 9
2.1.1. Recommended Architecture *9*
2.2. ARCHITECTURAL STRATEGIES 9
2.2.1. Design Alternative *12*
2.2.2. Reuse of Existing Common Services/Utilities *12*
2.2.3. Creation of New Common Services/Utilities *12*
2.2.4. User Interface Paradigms *12*
2.2.5. System Interface Paradigms *11*
2.2.6. Error Detection / Exceptional Handling *12*
2.2.7. Memory Management *13*
2.2.8. Performance *13*
2.2.9. Security *13*
2.2.10. Concurrency and Synchronization *13*

3. DETAILED SYSTEM DESIGN **14**

3.1. KEY ENTITIES 12
3.2. DETAILED-LEVEL DATABASE DESIGN 15
3.2.1. Data Mapping Information *16*
3.2.2. Data Conversion *16*

4. ENVIRONMENT DESCRIPTION **17**

4.1. LANGUAGE SUPPORT 13
4.2. USER DESKTOP REQUIREMENTS 14
4.3. SERVER-SIDE REQUIREMENTS 14
4.3.1. Application Server Disk Space *14*
4.4. CONFIGURATION 14
4.4.1. Operating System *14*
4.4.2. Desktop *14*

5. APPENDIX **15**

1. Introduction

The banking system project is aimed to provide digital banking facilities and generate transactions in a very fast and efficient manner. This project will help the banks to maintain customer's record such as customer's personal details, customer's account details, customer's transaction history etc. in the database.

1.1. Intended Audience

ADMIN	Can access all Employee, Customer accounts and can manipulate the data.
EMPLOYEE	Can access Employee, Customer Accounts Management and bank functions
CUSTOMER	Can view customer menus, access many banking functions.

1.2. Acronyms/Abbreviations

Acronyms / Abbreviations	Full Form
un	User Name
Fp, fptr	File Pointer
log	login
pwd	Password
IFSC	Indian Financial System Code
trans_limit	Transaction Limit
cust	Customer
EOF	End Of File
fname	First Name
lname	Last Name
acc	Account

1.3. Project Purpose

The purpose of our project is to provide a banking system online, make customer's record easy to handle and reduce manual work of managing record files of customer's information.

1.4. Key Project Objectives

The main objective of the banking system project is to create a system where smooth banking functionality occurs such as a smooth transaction process between 2 accounts, automatically stores transaction history data and many more.

- Accuracy.
- Easy & fast retrieval of information.
- Decrease the load of the person involved in the existing manual system.
- Admins to control the access and maintain security
- Access any information.
- It satisfies the user requirement
- Have a good user interface
- Be easy to understand by the user and operator

1.5. Project Scope and Limitation

This system will be built to reduce the amount of manual labor in keeping track of all the banking records used, application of Customers and also the data of the Employee, admins and Customer.
It seeks to increase production and offer a better managed system. This system will be simple to use and comprehend. More specifically, this system is made to enable an administrator(admin) to control Employee and Customer and access all records.

1.5.1. In Scope

i) Creating New Customer Account: This project creates new account of customer (Customer's Personal Details and Saving Account for customer). It helps customer to update, change their personal details online.
ii) Balance Enquiry: Customers can check their balance online.
iii) Account Holder List: Admin can list the different accounts present in the database of Employee and Customer.
iv) Deleting: Customer can delete his/her account by choosing delete option.
v) Transaction History: Customers can view their previous transactions.
vi) Help Desk Facility: It provides contact details to resolve problems via phone call or text.

1.5.2. Out of scope

Since the project is based on C programming language and File Handling in C. It has limited graphic UI and functions. This project cannot provide net banking facilities. Database security is not available for this project. Multithreading is not applicable in this project.

1.6. Functional Overview

1.6.1 Structure: The project uses many different user defined data structures such as

i. Date Structure: date, month, year are the attributes of date structure.

ii. Login Structure: username and password are the attributes of Login Structure.

iii. Customer: customer Id, first name, last name, Aadhaar number, phone number, email, gender, date, address are the attributes of Customer Structure.

iv. Account: user Id, password, IFSC, account number, balance, transaction Limit are the attributes of account structure.

v. Transaction Structure: customer id, transaction id, type, amount, date remarks, receiver name, receiver account number are the attributes of Transaction structure.

vi. Help Desk: phone number and name are the attributes of help desk structures.

2.1.2 Functions: Functions are used to perform certain tasks to implement projects.

i) Validate Functions: This function is used to perform validation such as whether a user entered email id is correct email Id or not.

ii) Display Functions: These functions are used to display it to terminals. For example, display_account_menu, it displays a list of options to choose to perform operation in the account section.

iii) getData() Functions: These functions ask the user to provide data.

iv) Update Functions: These functions update the already present data in a database.

v) Delete functions: these functions delete the customer/employee's information from the database.

1.7. Assumptions, Dependencies & Constraints

i. Comma Separated files are created in the database.
ii. International transactions are restricted.
iii. Balance and transaction will be in INR Rupees.

1.8. Risks

Admin should be careful while handling databases as data leakage problems may occur because security to data is not possible in C File handling project, thus customer's crucial information may leak.

2. Design Architect

Design architect of banking system contains 3 sections:

i) Admin Section
ii) Employee Section
iii) Customer Section

Every section has login credentials, and it checks user-Id and password. Once the user has been logged in. It asks user input and every user input has been validated and then it performs operations accordingly.

2.1 Design Objectives

Design objective is that it helps to know the flow of the project, where to start and what to do next while writing code. It is to make a flowchart of a banking system representing flow of business logic and an ER diagram to represent attributes in every module.

- Memory leaks are not possible and well managed

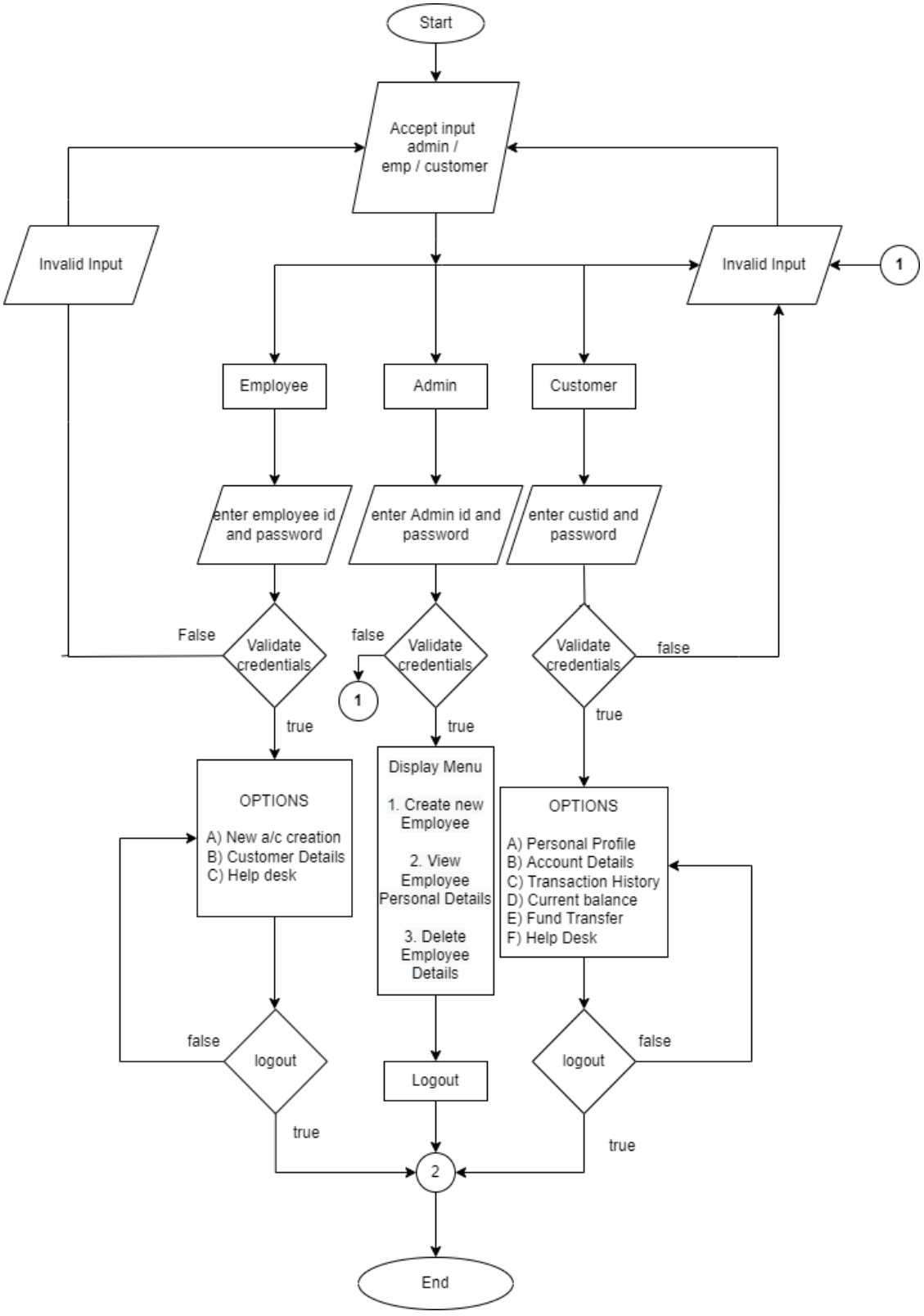
2.1.1 Recommended Architecture

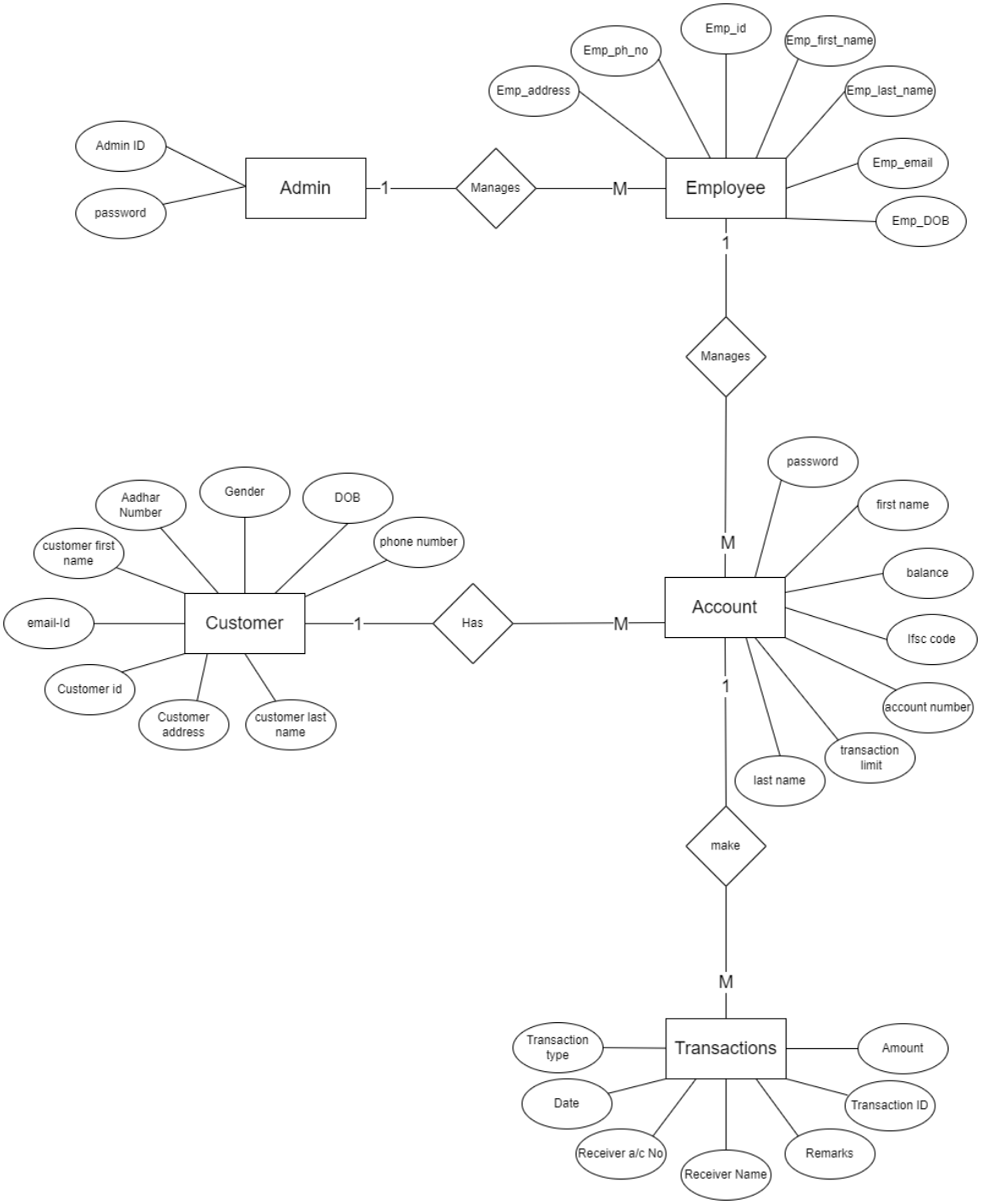
Flow chart with start and ask input users to choose from admin/employee/customer. In admin, employee and customer sections the user has to enter login credentials and then it is validated if the user is logged in otherwise it returns to ask input user.

2.2 Architectural Strategies

In Banking system applications we have used some strategies that affect the overall efficiency and memory management. Some are listed below:

- Functions for Updating and Deleting the existing data of Employee or customer is implemented using the Linked-List which helps to manage the data in an efficient way and again to rewrite the data in the file after performing the desired operation. For this first we create a structure for the Linked-List and after that create the nodes according to the structure variables.
  By this we are retrieving the data from the file one by one and storing it in a node of the Linked-List. We are using this Linked-List to update and delete operations. And finally, we again write the data in the file using this Linked-List and free the memory.
- Use of structures for easy access to the data from the file. Structures help to maintain the data in a systematic manner and at particular locations.





2.2.1 Design Alternative

There are many alternative strategies are available for a particular operation lets discuss one by one.

- For Updating and Deleting the data we have used the Linked List and the working of using Linked-List we have discussed above.
  The other method for Deleting or Updating is that we can access the data using the built-in function like **fscanf()** or **fwrite()** and store the data in temporary file after that we again open the original file in write mode and traverse the temporary file. We start to write the data in the file and skip the details that we want to delete. Or update the details that we want to update.
  The Main Reason to avoid this method is that it is time consuming and needs more memory as we have to create the temporary file for the particular operation.
- For storing the data, we have created the structures because structures provide the particular location of the data in file and easy maintenance.
  The other method for storing the data is just by creating the simple variable of each attribute of each entity. This would result in poor management and leads to consumption of high memory. By declaring the individual variables, we cannot provide a particular location to the particular variable in the file.
  The main reason to avoid this method is the poor management of variables and file data.

2.2.2 Reuse of Existing Common Services/Utilities

In this we have used the standard C Libraries to read, write the data and to store the data and more. In This System we are not using any type of existing services other than the C Libraries or Linked List definition.

2.2.3 Creation of New Common Services/Utilities

In this we have defined a simple way to manage the accounts in an efficient way. The most efficient way to manage the data is by using the Linked List. And we did all the operations like Delete, Update by using the Linked List.
Basically, the Linked List is not the new common service that we have created, it is the existing one and we are using the reference or its definition to manage the data.

2.2.4 User Interface Paradigms

We have used the console through which users can interact with the system. There are certain options with numbers on the screen. To select the particular option just press the number that represents the option. There is always a default option that prints a message if you enter the wrong choice. On other hand, an option for logout or to go back to the previous page is available. These things make the system more interactive and user friendly.

2.2.5 System Interface Paradigms

When it comes to System Interface, we have worked on the Linux platform with C language. Here we have used the GCC compiler to compile our code and run the code. Linux provides a flexible way to check about memory management and

memory leaks. Also, here we can optimize our code by estimating the running time for the particular process using the different tools.

2.2.6 Error Detection / Exceptional Handling

In Exceptional Handling, all the validations are introduced to avoid the Errors. Also, if there is any possibility of exceptions, we are printing the user-friendly message and returning the control of the program to the starting of that function from where the error occurred. With the help of these messages, users can find what is wrong with the inputs. Also, if the user selects the wrong choice, then there is always an option to back and select the right one.
Validations are introduced to minimize the error and manage the control flow of the system.

2.2.7 Memory Management

As we have discussed, we have worked on Linux, and it provides various tools for memory management and by using these tools we can check for memory leaks.
The chances of memory leaks are in the function of delete, search and update because we are using the dynamic memory allocation for the implementation of Linked List. During Implementation of Linked List, we have freed the memory of each node. For the confirmation we have used the Valgrind tool present in Linux. This tool helps to check how many blocks of memory are lost and how many memory leaks are possible.
As the final result we have found that there are no memory leaks possible, and all the dynamically allocated memory is freed.

2.2.8 Performance

In Banking System Application, we have used certain strategies to improve the performance. As we have already discussed about the Linked List implementation and Structures uses. Once again, let’s see the effect of using Linked List on performance:
Linked List provides an easy way to retrieve the data from the file and manage or modify it according to our needs. The Linked List helps us to find the solution of most time-consuming operations like the Delete and Update function. We would see the functionality of Delete and Update Later in this Document.

2.2.9 Security

Security is the main aspect of our application. Our application consists of the sensitive data of the Employee and customer and Accounts. When it comes to the access to that data, only the admin has the access to control the flow of the data for example only the admin can only manage Employee and Employee can only manage Customers.
Employees have the Functionalities to create and modify the details of the Customer and the accounts in a Bank. Customers have the functionalities to check their personal details, account details, transaction history, check balance, transfer funds, access Help Desk.

3. Detailed System Design

3.1. Key Entities

Banking system application has five main Key Entities associated with it:

- Admin
- Employee
- Customer
- Transactions
- Accounts

3.2. Detailed-Level Database Design

In our system, data for Employee, Customer and administrators was manually inserted into the database. Since each entity has unique attributes that are of different data types, structures have been used to store the data in the database.

Admins have the functions to manage the Employees. Each Employee is provided with a unique Employee ID whenever they are being added to system using which they are able to access the banking system and can perform their tasks efficiently.

The Employee has the functionalities to modify the Customers, and to see the records of Customers and has the functionalities to modify the account information.

The accounts have the description of the Customer Ids for that Account. Every customer will have a unique Customer ID, but they can also have multiple accounts.

Validations have been done for:

- **First Name:** First Name of an individual can only contain combination of Uppercase and Lowercase alphabets or '.'. Use of spaces, numeric characters, symbols, etc. are not permitted.
- **Last Name:** Last Name of an individual can only contain combination of Uppercase and Lowercase alphabets or '.'. Use of spaces, numeric characters, symbols, etc. are not permitted.
- **Password:** Password requirements includes a minimum of 1 to maximum of 15 characters.
- **Aadhaar:** Aadhaar number of an individual must be of 12 digits only.
- **Gender:** For Gender field, values accepted only are 'Male' or 'Female'.
- **Email Id:** Emails must suffix with "@school.com" or "@gmail.com,". For school login IDs, use "@school.com" suffix, and for personal email IDs, use "@gmail.com" suffix.
- **Phone number:** Phone number should include only 10 digits.

4. Environment Description

The Banking system Application is built on a Linux based operating system. Since it is a very basic program for Banking system application for business purpose, any high-end system requirements are not desired. Operating system, processor speed, system memory, free storage space, etc. are the basic parameters required to find out if the system used by the end consumer is compatible or not for usage of a Bank Application. The system doesn't require any audio hardware or GPU since it doesn't consist of any audio and video related functionality. The following sections observe discussion regarding minimum and recommended configuration requirements or system requirements for end user desktop compatibility check.

4.1. Language Support

The system is not built on multi-language, only one language has been used. C language has been used for the build of the system. Header file **stdio.h** has been used for standard input and output of the system. Header file **stdlib.h** has been included to utilize standard library files since the system makes use of heap memory allocation using malloc function. Another header file called **string.h** has been used since the system involves the application of string related functions like copying of string, etc.

4.2. User Desktop Requirements

Minimum configuration required for the system to run are 1 GHz processor, 2GB RAM and 100 GB of Hard disk space. The system has been developed in Ubuntu which is Linux based Operating System, also for end users, the executable file runs well on Ubuntu therefore it is the recommended and minimum required configuration for Operating System. The system requires data storage for further usage, which has been stored in .txt files which is being called a database for the system.

Minimum configuration required for Banking System Application is shown in the table below:

Category	Minimum Configuration
Processor	1 Gigahertz (GHz) processor
Storage	2 GB RAM & 100 GB Hard Drive
Operating System	Ubuntu \ Linux
Database	.txt file

4.3. Server-Side Requirements

4.3.1. Application Server Disk Space

The application overall takes up a few Megabytes of Disk Space. The directory containing all the files to be compiled consists of 2 header files having .h extension and other .c files containing client code, codes regarding basic operations in Linked List used for updating and deleting functionality of the program. Other files consist of individual programs having four major functionalities which are insertion of data, searching of particular data, deletion of data and for updating of data.

4.4. Configuration

All the required minimum configuration details and recommended configurations are discussed in further sections for smooth end-user experience.

4.4.1. Operating System

System was developed using the C language in Ubuntu which is a Linux based system. As per system requirements Linux OS is recommended for user-end as well. For compilation and execution of a system GCC compiler is required. Ubuntu can also be run on a windows OS using Hypervisor which creates and runs one or more virtual machines on the host machine. VMM Virtual Machine Manager, can also be used to run Linux concurrently over Windows OS.

4.4.2. Desktop

For the system to run smoothly and to find if the end user desktop is compatible, there are no high-end requirements. Linux based OS and Terminal are the only basic system software requirements. No driver software, or audio or video related task is intended. Table below shows system configurations for using VMM server Some of the basic hardware requirements are

- Display resolution should be minimum SVGA 800 x 600 or above.
- Any system manufactured in the past 10 years.

Hardware	VMM server	VMM console
Processor(minimum)	8 core Pentium 4, 2GHz(x64)	2 core Pentium 4, 1GHz CPU
Processor(recommended)	16-core, 2.66 GHz CPU	2 core 2 GHz CPU
RAM (minimum)	4 GB	2 GB
RAM (recommended)	16 GB	4GB
Hard drive (minimum)	50 GB	50 GB
Hard drive (recommended)	200 GB	200 GB
Display Resolution (minimum)	-	SVGA 800 x 600
Display Resolution (recommended)	-	FHD 1920 x 1080

5. Appendix

Group-1_Sprint 1_Banking system Application:

https://drive.google.com/drive/u/0/folders/1otRgOnGnE3P5luydmZ-xr5jTHYkzU-Hk

Change Log

QMS Template Version Control (Maintained by QA)

Date	Version	Author	Description
28-May-2015	1.0	QA Team	Initial Version