Chapter 5

Analysis and Design

5.1 Introduction

The previous chapter was about what technology/methodology was selected to solve the problem. In this chapter describes proposed system's analysis and design work carried out. Using OOAD there support diagrams included, namely use case, activity, sequence and class diagram. In later part database design and user interface design included.

5.2 Software Requirements

Software requirements are divided in to two as Functional and Non-Functional. Again the Functional requirements are divided in two as Mandatory and Desirable.

5.2.1 Functional Requirements

Functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Each use case can be said to implement one or more functional requirement.

5.2.1.1 Mandatory

1. System shall facilitate to upload Inward Data Images CD to the system in multiple branch format or single branch format.
2. System shall facilitate to allocate cheques among users in branch wise or regional wise.
3. System shall facilitate to reallocate unprocessed cheques of particular user to another user at any time.
4. System shall facilitate to get the Core Banking "Invalid File".
5. System shall facilitate mark the cheque as invalid (Incorrect Acc No/Cheque No or both) by referring the said file.
6. System shall facilitate to correct the account numbers and cheque numbers in semi automate way.
7. System shall facilitate to authorize the change of both account number and cheque number by the particular users' Section Manager.

8. System shall facilitate to send the Repaired file to the Core banking server.
9. System shall facilitate to scrutinize cheques to mark them honour or dishonour due to technical reasons.
10. System shall facilitate to mark the reason(s) if a cheque dishonour.
11. System shall facilitate to display relevant signatures in authorized amount levels
12. System shall facilitate to automate Signature Verification System.
13. System shall facilitate to users to skip marking honour or dishonour a particular cheque and pass to his Section Manager.
14. System shall facilitate to the Section Manager to honour or dishonour or send cheque details along with image to the relevant Branch Manager for his reference.
15. System shall facilitate to the Branch Manager to honour or dishonour the said cheque.
16. System shall facilitate to receive the account base return file which was prepared by the core banking system.
17. System shall facilitate to mark cheques on account base reason using above said file.
18. System shall facilitate to send technical return data file to the core banking system.
19. System shall facilitate to burn return data file to a CD in the format was given by the LCPL.
20. System shall facilitate to re burn said file if necessary.
21. System shall facilitate to customers to view cheque; those are presented for clearing in the day.
22. System shall facilitate to grant additional day(s) for those branches failed to complete the cheque processing in their own or technical failure.

5.2.1.2 Desirable

22. System should facilitate to set the clearing date when upload the CD.
23. System should facilitate to allocate cheques among users fully or partially.
24. System should facilitate to view said data in meaningful format.
25. System should facilitate to archive cheque data with image and history of its clearing process.

26. System should facilitate to observe user activity in user wise or branch wise by the Section Managers.
27. System should facilitate to add branches to the system as centralize clearing operation expansion.

5.2.2 Non Functional Requirements

Non-functional requirements define how a system is supposed to be. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals" and "quality of service requirements". Followings are the non functional requirements concern project.

- System shall not facilitate to upload an already uploaded cheque.
- System shall not facilitate to set the clearing date to a non working day.
- System shall not facilitate to set the clearing date to a future day.
- System shall not facilitate allocate a cheque already processed by a user.
- System shall not facilitate send the repaired file to the core banking system unless the entire invalid items are been corrected.
- System shall not facilitate send the repaired file unless if there any items with both account number and cheque number has changed and has not authorized by the relevant manager.
- System shall not facilitate burn outward return CD unless entire cheques are processed for a particular branch.
- System shall not facilitate users to use the same password more than 30 days.
- System shall not facilitate users to more than 3 error attempts to log in to the system.

5.3 Top Level Architectural Design

To get a top level overview of the proposed system it was hope to discuss the top level architecture. The proposed system has to interface with the existing Core Banking System and the Signature Verification System. As the standard the software vendor of the said systems they allow to access their system through an interface.

Then the components of the top level architecture are User Interface of the CICPS, the common interface, the Core Banking Database and SVS. The Figure 5.1 shows the interaction of each module.

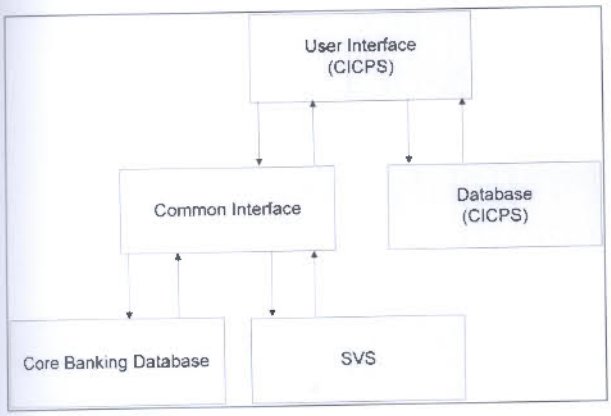


Figure 5.1 : Top Level Architectural Design

As per the Figure 5.1 the user interface pass a message (formats are pre defined) to the common interface. The common interface validates the message and sends to the Core Banking Database or SVS. Core Banking Database provide With respect to the message user interface receive the reply. The interaction with the CICPS database is direct using Open Database Connectivity – ODBC.

To see the graphical overview of the proposed system refer Appendix D.

5.4 System Architectural Design

The Figure 5.2 shows the detailed system architectural design. Individually manageable components were divided into modules.

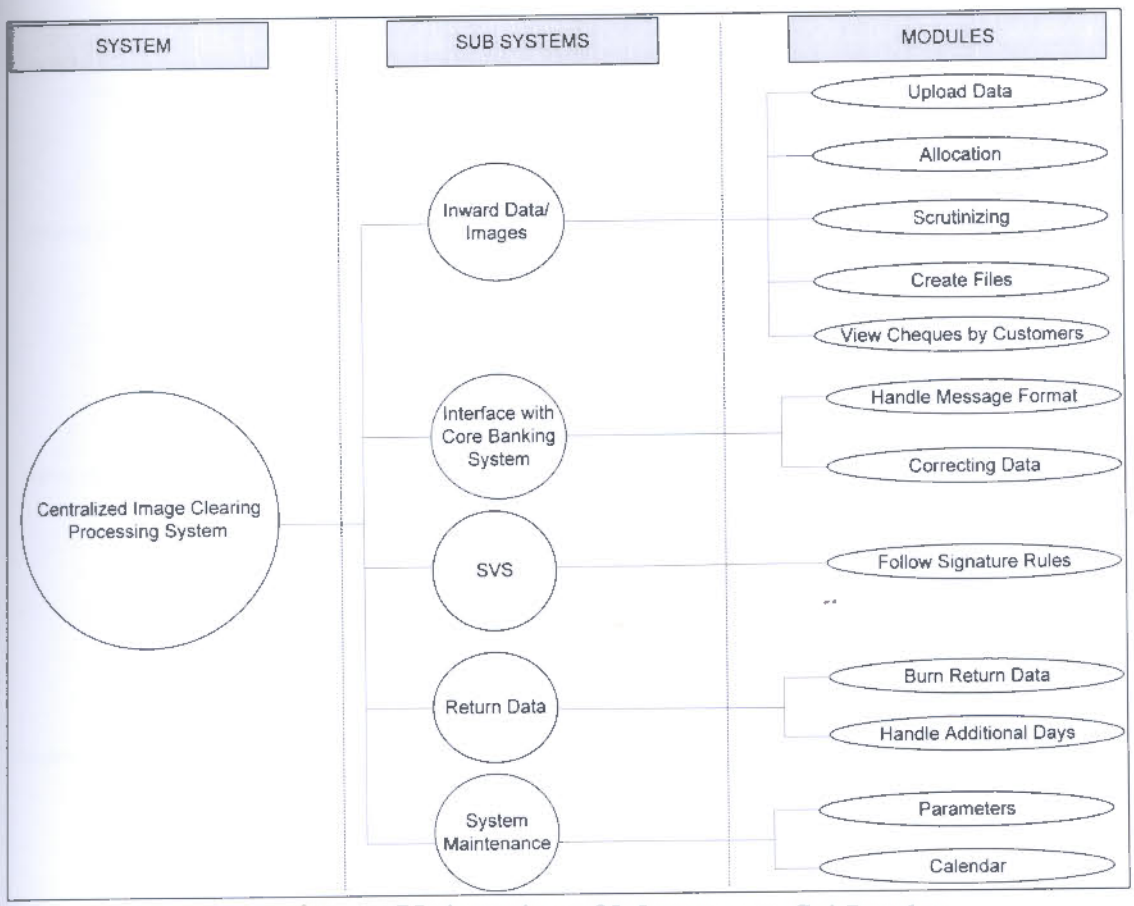


Figure 5.2 : System Architectural Design

For the sake of make easy to maintain and implement it has been divided the system into five sub systems.

In module wise there are interactions exists. As an example "Burn Return Data" in the Figure 5.2 depends on the "Create File" as the burning file need to be created first and in the agreed format with the LCPL.

5.5 Use Case Diagram – Static View of the System

By analyzing the functional requirements (section 5.1.1) it can be created the use case diagram for the proposed system. Figure 5.3 shows the use case diagram.

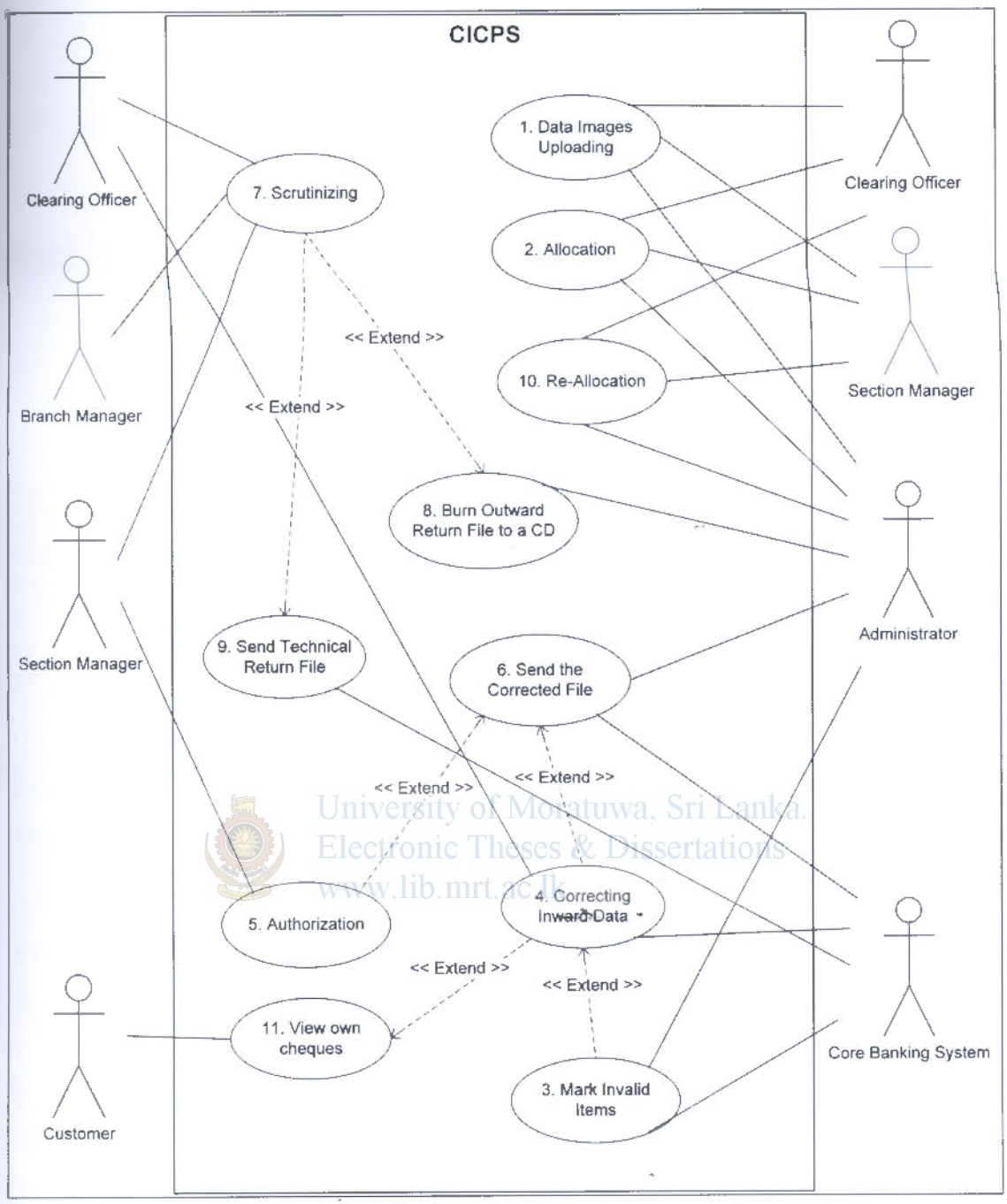


Figure 5.3 : Use case diagram of the proposed system

As per the Figure 5.3 use case No.4 can not be done without completing the use case No. 3. Because of that reason use case No. 3 extends the use case No.4.

5.6 Use Case Descriptions and Activity Diagrams

In order to minimize the complexity a check list is created considering functional requirements, use case descriptions and activity diagrams. Table 5.1 shows the check list.

Use Case	Software Requirements	Use Case Description	Activity Diagram
1. Data Images Uploading	1	1	✓
2. Allocation	2	2	✓
3. Mark Invalid Items	4,5	3	✓
4. Correcting Inward Data	6	4	✓
5. Authorization	7	5	✓
6. Send the Corrected File	8	6	✓
7. Scrutinizing	9,10,11,12,13,14,15	7	✓
8. Burn Outward Return CD	16,17,19,20,22	8	
9. Sending Technical Return File	18	9	✓
10. Re-Allocation	3	10	✓
11. View own cheques	21	11	

Table 5.1: Check list of software requirements, use case descriptions and activity diagrams

5.6.1 Use Case Descriptions

For an example a use case description for "Data Images Uploading" is stated below.

Use Case 1: Data Images Uploading
Actors : Administrator, Clearing Officer, Section Manager
Pre-conditions • Previous days' work must be completed • Current day must be marked as a working day • Branch data in the DVD must be activated branches in the system • Data should not be already uploaded for a particular branch and date
Post-conditions • There must be same number of inward clearing data and cheque images.
Basic Flow

1. Check for data and Images file available
2. Administrator checks the Inward Data for whether all the participating branches are included in the DVD.
3. Check current process date is match with the clearing date.
4. Check for each and every cheque image must have a data line in the data file.
5. Upload images & data lines.
6. Acknowledge the uploading to Section Managers and Clearing Officers for their readiness to the days' work.

Exceptions

1.1 Else make available the data and images file

2.1 Else acknowledge missing branches

3.1 If it is not match set to the correct date

4.1 If above check fails for a particular branch

4.1.1 Delete the branch's data.

4.1.2 Acknowledge the omission

Use case relationships

- "Allocation system" refers cheques uploaded by the Data/Images uploading system.

5.6.2 Activity Diagram – Dynamic View of the System

Following the use case descriptions it can design the activity diagrams for use cases. The activity diagram for "Data Images Uploading" is showed in the Figure 5.4.

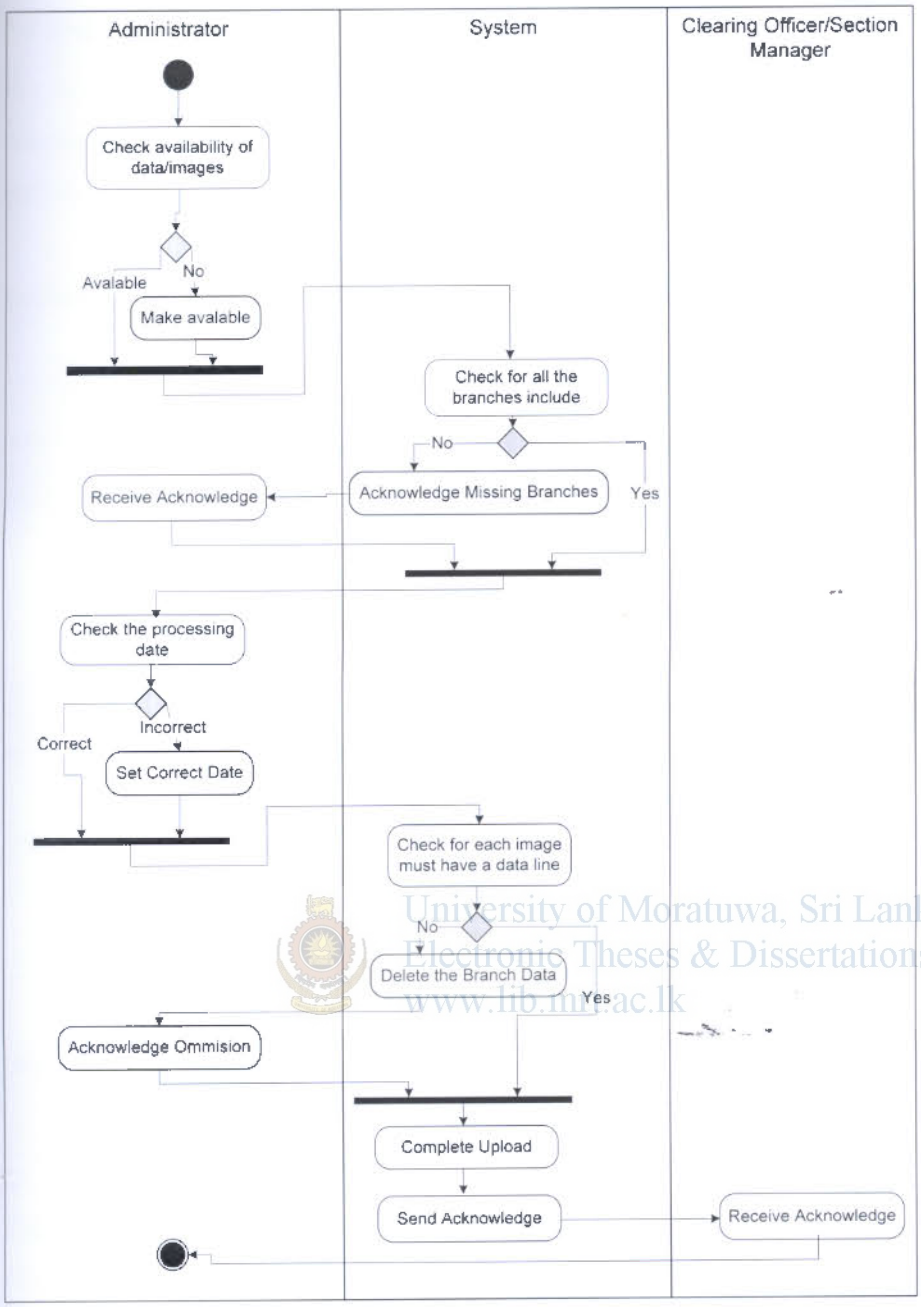


Figure 5.4 : Activity Diagram for Data Images Uploading

As per the Figure 5.4 activity is stated with initial node. The first use case is indicated that the "Check availability of data/images" use case is invoked as one of the activities. At the point of diamond there is decision to make. Fork a black bar with one flow going into it and several leaving it. This denotes the beginning of parallel activity.

For the other activity diagrams refer the Appendix F.

5.7 Sequence Diagrams

For each activity diagram it can be design activity diagrams to view the dynamic behaviour of the system. In order to minimize the complexity of the sequence diagrams only the happy path is considered.

For example Figure 5.5 shows sequence diagram for data images uploading.

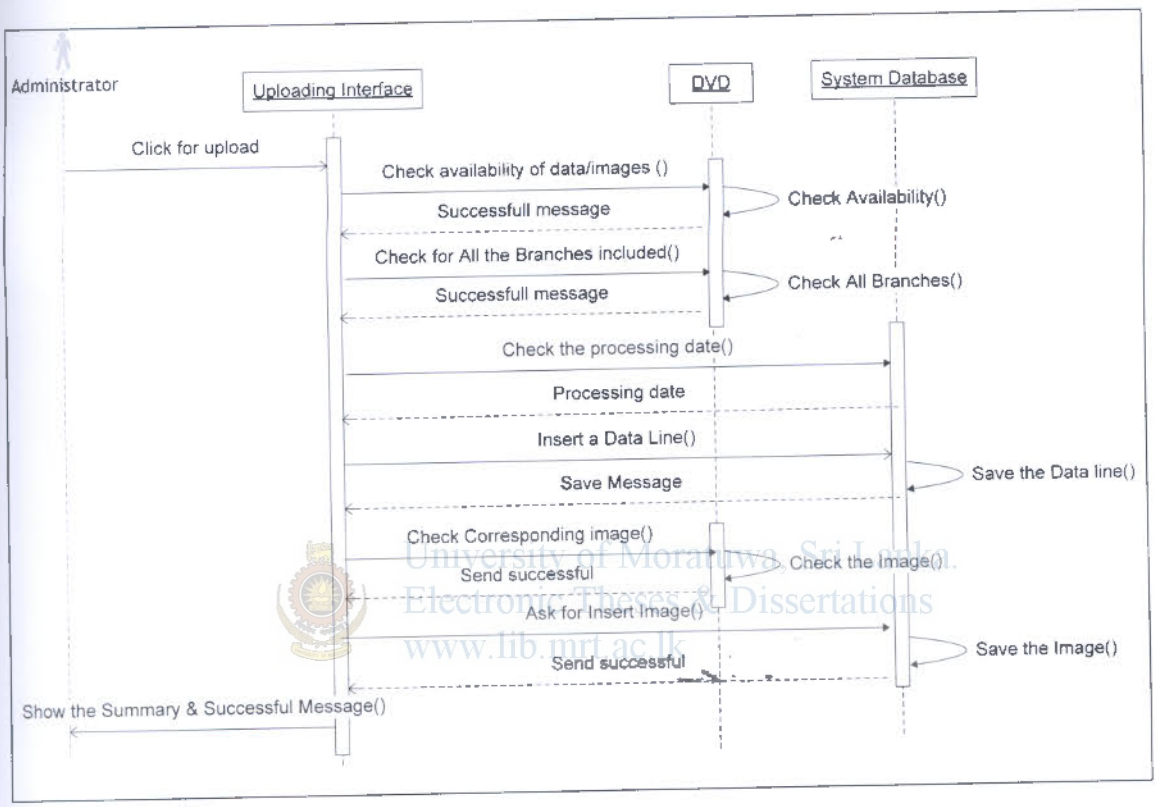


Figure 5.5 : Sequence diagram for Data Images Uploading

(To get the understanding on notations used in the Figure 5.5 refer the page no. 25)

For the other sequence diagrams refer the Appendix G.

5.8 Class Diagram

Now it is the time to visualize the class diagram. By referring sequence diagram's interface class, entity classes and their methods the class developed. Figure 5.6 shows the proposed system class diagram.

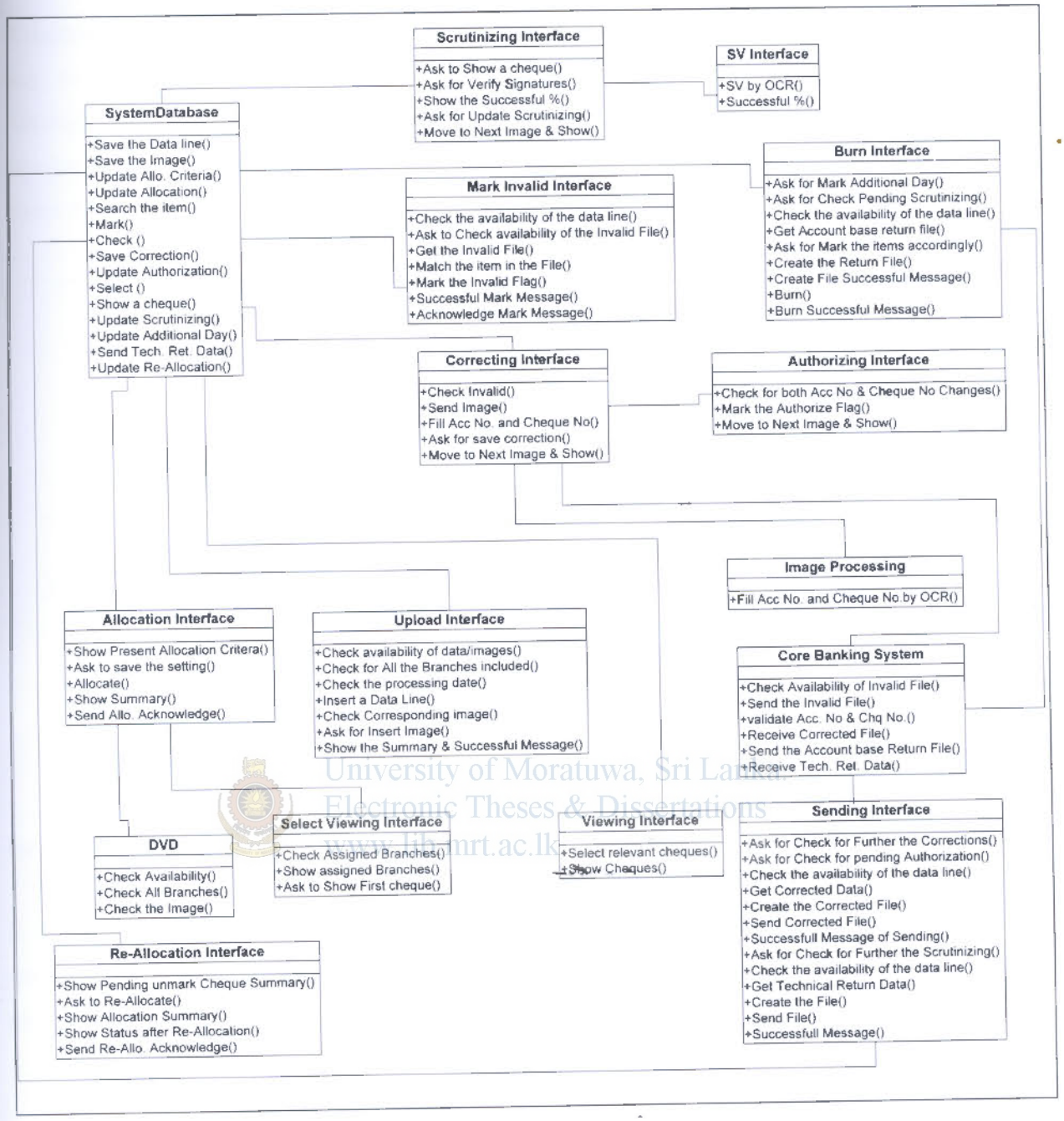


Figure 5.6 : Class Diagram

As per the Figure 5.6, all the classes are interface classes, apart from the "System Database" and "DVD". Associates among classes are denoted with a line. Methods belongs to each classes are indicated in side the box.

5.9 Database Design

A database is a collection of data which can be stored and manipulated in an efficient way. In a relational database data is structured in a form of related tables. In concept of relational database the real world objects are considered as entities and the connection between them are as relations. The diagrammatic presentation of said relation is shows in Figure 5.7 as Entity-Relation – ER diagram.

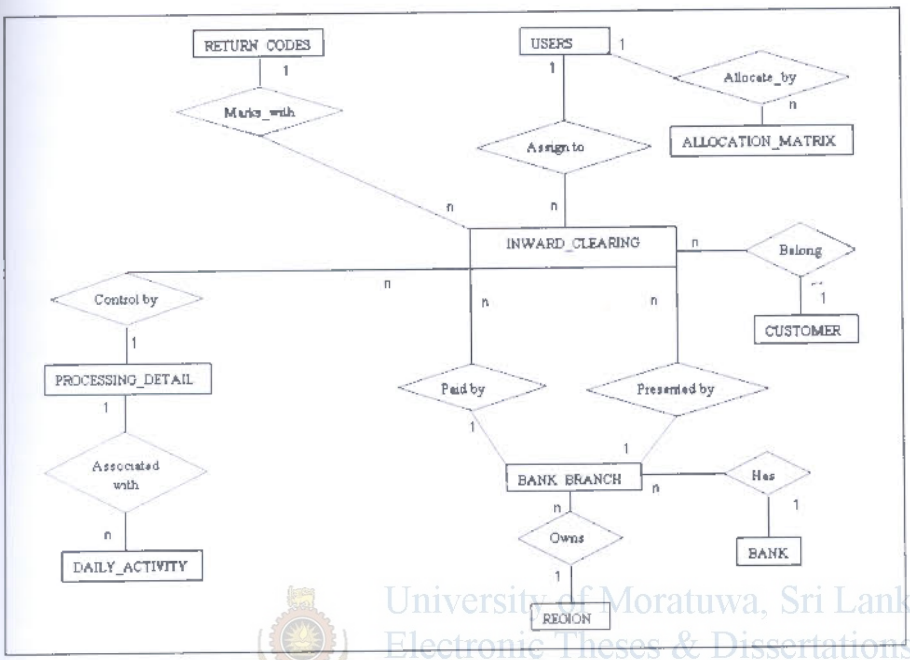


Figure 5.7 : ER Diagram

In the Figure 5.7 rectangles are representing Entities and diamonds are relations. The attributes of each entity are representing using ellipses. The Figure 5.8 shows the attributes of BANK_BRANCH entity.

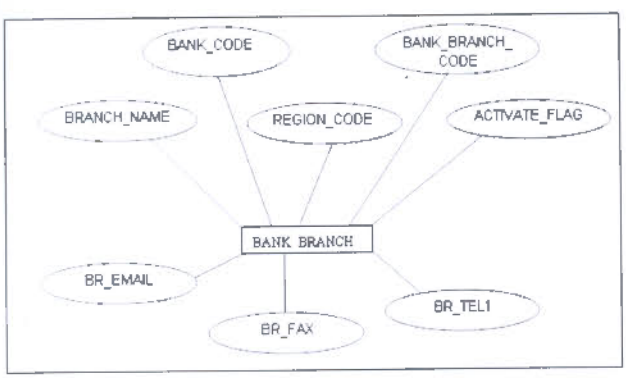


Figure 5.8 : Attributes of BANK_BRANCH Entity

5.9.1 Relational Database

A database can be understood as a collection of related files. How those files are related depends on the model used. Early models included the hierarchical model (where files are related in a parent/child manner, with each child file having at most one parent file), and the network model (where files are related as owners and members, similar to the network model except that each member file can have more than one owner) [16].

The relational database model was a huge step forward, as it allowed files to be related by means of a common field. In order to relate any two files, they simply need to have a common field, which makes the model extremely flexible.

The creation of table by using Structured Query Language – SQL is as follows.

Table Name: BANK

```
CREATE TABLE BANK
(BANK_CODE char(4) PRIMARY KEY,
BANK_NAME varchar(50));
```

Table Name: BANK_BRANCH


```
CREATE TABLE BANK_BRANCH
(BANK_CODE char(4) REFERENCES BANK(BANK_CODE),
BANK_BRANCH_CODE char(3),
BRANCH_NAME varchar(60),
BR_TEL1 varchar(15),
BR_FAX varchar(15),
BR_EMAIL varchar(60),
REGION_CODE char(3),
ACTIVATE_FLAG char(1),
PRIMARY KEY (BANK_CODE, BANK_BRANCH_CODE));
```

The Figure 5.8 shows all relations in proposed system.

[16] http://www.databasejournal.com/sqletc/article.php/1469521

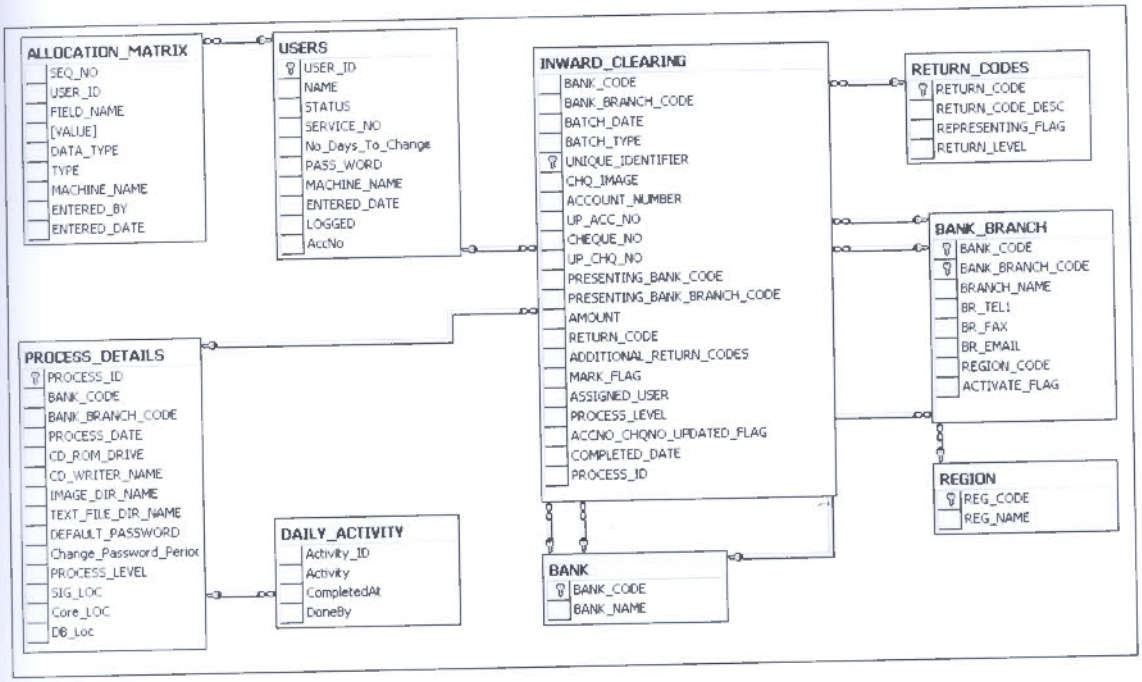


Figure 5.9 : Relations in proposed system

The Figure 5.9 shows relations among tables and the "key" denotes primary key fields of respective tables.

5.10 User Interface Design

After all the hard work now it is time to design how user interaction to the system. Interfaces are the users' interactions to the system in order to get the intended task done. As such it is paramount offer attractive and friendly interfaces.

The following principles [9] were adapted at the user interface design.

1. **The structure principle**. Your design should organize the user interface purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another. The structure principle is concerned with your overall user interface architecture.

[9] http://www.ambysoft.com/essays/userInterfaceDesign.html

2. **The simplicity principle**. Your design should make simple, common tasks simple to do, communicating clearly and simply in the user's own language, and providing good shortcuts that are meaningfully related to longer procedures.
3. **The visibility principle**. Your design should keep all needed options and materials for a given task visible without distracting the user with extraneous or redundant information. Good designs don't overwhelm users with too many alternatives or confuse them with unneeded information.
4. **The feedback principle**. Your design should keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.
5. **The tolerance principle**. Your design should be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions reasonable.
6. **The reuse principle**. Your design should reuse internal and external components and behaviours, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.

5.10.1 Specifications of design of user interface

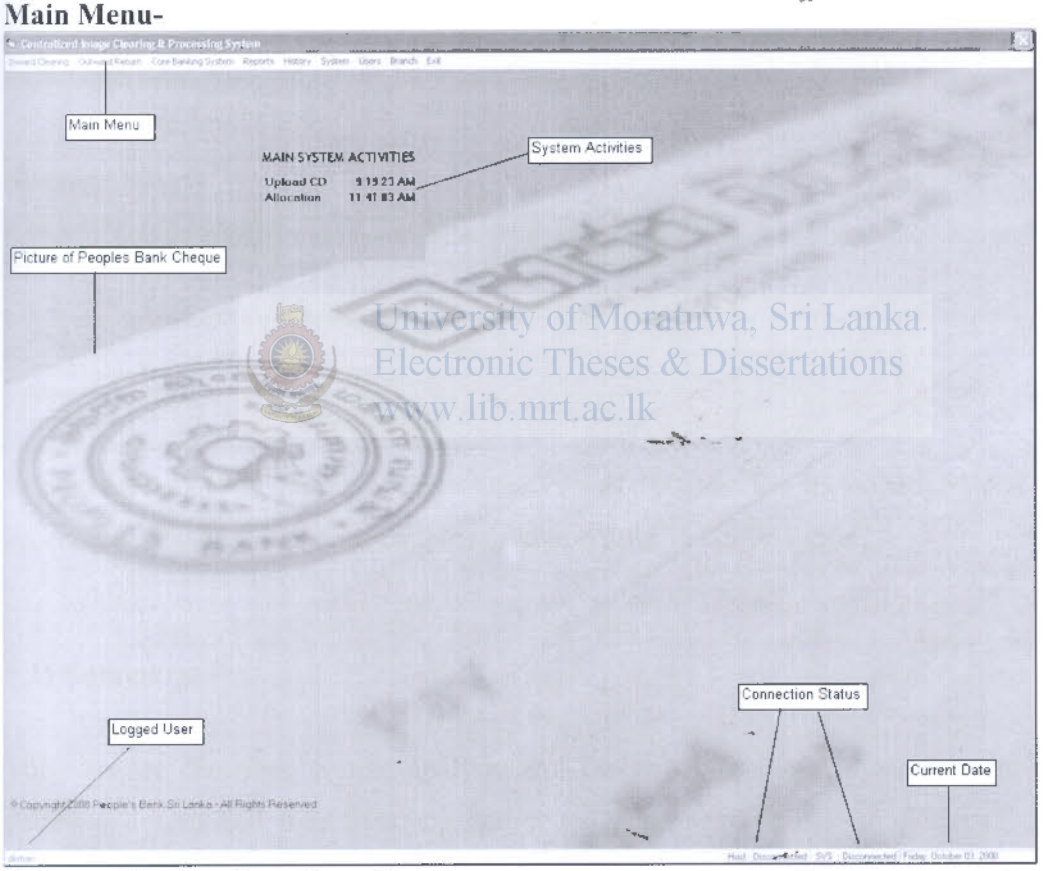
In the frame of said principles specifications for the user interface design is summarized in the Table 5.2.

Form Header	Font - MS Sans Serif Font size – 8 Font Colour - &H8000000D&
Form Back ground	Colour - &H80000013&
Labels	Font - MS Sans Serif Font Size - 8 Font Colour - &H80000017&
Text Boxes	Font - MS Sans Serif Font Size - 8 Font Colour Summarized Fields- &H80000001&

	Other Fields - &H80000008& Alignment: Numeric Fields – Right Justify Others – Left Justify Height - 285
Buttons	Height – 420 Colour Standard buttons - &H8000000F& Graphical buttons – N/A

Table 5.2: Specifications for UI Design

Main Menu-



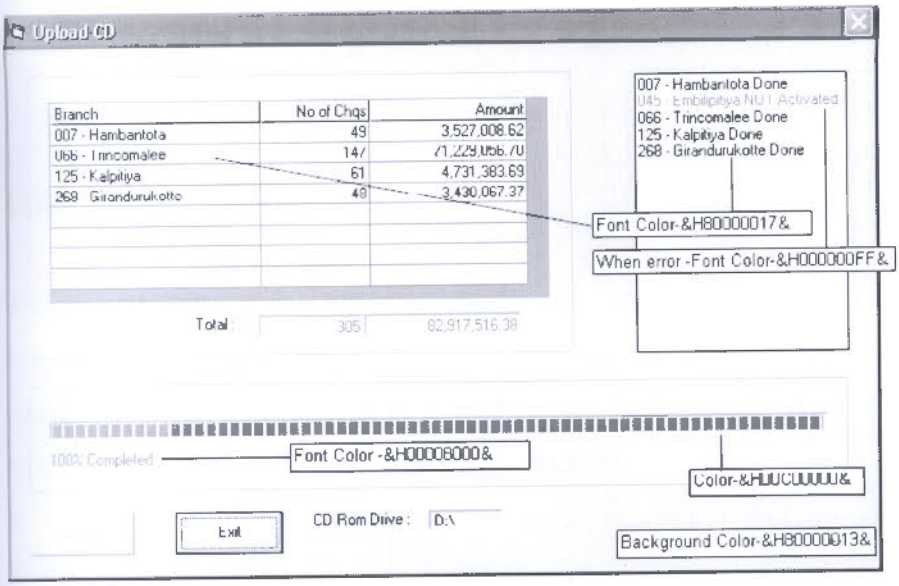
Picture of Cheque Size- Size of screen

Font -MS Sans Serif

Font Color - &H80000012&

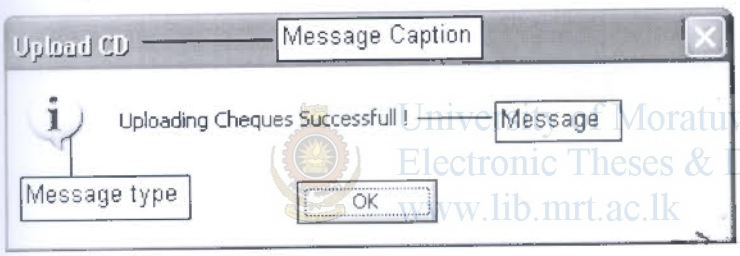
Font Size – 8

Other interfaces-



In the interfaces Red and Green have used in meaning they were inherited.

Message Boxes-



In message boxes the length of them were kept at 100mm. The message type was used appropriately as in the above example message type is an "Information".

5.11 Summary

This chapter describes system analysis and design of proposed system. It included database design and user interface design too. The next chapter is discussed about implementation of the system.