Comparison between .NET and Java EE
Implementation of Cash & Bank System

MASTER THESIS

Author
Hafiz Umer Waqar
hwr10001@student.mdh.se

Supervisor
Frank Lüders
frank.luders@mdh.se

Examiner
Kristina Lundqvist
kristina.lundqvist@mdh.se

November 12, 2013
ABSTRACT

The demands of software engineering increases with everyday passed and every organization shows interest to work on daily routine work in the computerized system to improve efficiency and accuracy of data. Most of organization demands different kinds of computerized software solutions that developed in modern technologies. There are different software development technologies that is getting popular with the passage of time and provide high quality product to their user. It is not an easy to decide which technology choose to develop computerized system where most software falls. The primary purpose of thesis is to compare two different modern software development technologies by develop an application with same functional requirements. This thesis report will help to user for selection of software development technology.

Cash and bank financial application is choose to develop using .NET and Java Enterprise Edition. Microsoft Visual Studio development environment is used for .NET development and NetBeans IDE is used for Java Enterprise Edition application. .NET framework contains many language that are interrelated with each other like C#, VB, J#, C++ and F#. Cash and bank application is develop in C# and Java programming languages and persistence storage of data is design in Microsoft SQL Server. Cash and bank application is develop with three tier architecture and layer approach. The syntax of .NET and Java Enterprise Edition is quite similar but procedure to establish connection, data retrieval and handling the data in different way. NET platform provide built in libraries, components and efficient user controls that helps for fast development in short period of time.

Keywords: Cash and Bank, book keeping, development process, .NET framework, layered architecture, 3-tier, Java EE, component, comparison, similarities, glassfish, entity class, differences.
ACKNOWLEDGEMENTS

I heartily thanks to my supervisor Frank Lüders for continue support, help and guidance throughout my degree program as well as during my thesis and obliged for his valuable time, suggestions and improvement in different stages of the thesis.

I also would like to thanks my father Muhammad Ashraf and my brother Muhammad Babar for their support, motivation and strength in every step of my life and would like to express my gratitude to my family for their encouragement during my thesis work.

Hafiz Umer Waqar
# TABLE OF CONTENTS

1. **INTRODUCTION** ...................................................................................................................................................... 8  
   1.1. **SCOPE OF THESIS** ........................................................................................................................................... 9  
   1.2. **PROBLEM STATEMENT** ...................................................................................................................................... 10  
   1.3. **GOALS** .............................................................................................................................................................. 10  
   1.4. **EXPECTED RESULTS** ........................................................................................................................................... 10  

2. **CASH & BANK SYSTEM** ................................................................................................................................................. 11  
   2.1. **RECEIPTS** ............................................................................................................................................................ 11  
   2.1.1. Direct Receipts by Cash Department ......................................................................................................................... 11  
   2.1.2. Direct Credits into Bank Accounts .............................................................................................................................. 12  
   2.2. **PAYMENTS** ............................................................................................................................................................ 12  
   2.2.1. Payments to Suppliers/ Service provider .................................................................................................................... 12  
   2.2.2. Process of Payment .................................................................................................................................................... 12  

3. **MICROSOFT .NET** ............................................................................................................................................................. 14  
   3.1. **MICROSOFT VISUAL STUDIO** .................................................................................................................................. 14  
   3.2. **.NET BUILDING BLOCK SERVICES** .............................................................................................................................. 14  
   3.3. **.NET ENTERPRISE SERVERS** ....................................................................................................................................... 14  
   3.4. **.NET FRAMEWORK** .................................................................................................................................................... 15  
   3.4.1. .NET Framework Components ........................................................................................................................................ 15  
   3.4.1.1. Common Language Runtime (CLR) ............................................................................................................................. 15  
   3.4.1.2. Base Class Library (BCL) ......................................................................................................................................... 16  
   3.4.1.3. .NET Framework Language ......................................................................................................................................... 17  
   3.4.1.4. Common Language Specification (CLS) ........................................................................................................................ 17  
   3.4.1.5. Common Type Specification (CTS) ............................................................................................................................ 18  
   3.4.1.6. Microsoft Intermediate Language (MSIL) ................................................................................................................... 18  

4. **JAVA ENTERPRISE EDITION** ............................................................................................................................................ 19  
   4.1. **ENTERPRISE JAVA BEANS (EJB)** ................................................................................................................................ 20  
   4.2. **GLASSFISH SERVER 3.0** ................................................................................................................................................. 21  
   4.3. **JDBC API** ...................................................................................................................................................................... 22  

5. **APPLICATION IMPLEMENTATION** .................................................................................................................................. 23  
   5.1. **APPLICATION DESIGN** .............................................................................................................................................. 23  
   5.2. **.NET IMPLEMENTATION** ............................................................................................................................................ 25  
   5.3. **JAVA EE IMPLEMENTATION** ......................................................................................................................................... 32  
   5.4. **DATABASE DESIGN** ...................................................................................................................................................... 40  

6. **COMPARISON BETWEEN PLATFORMS/ DEVELOPMENT ENVIRONMENTS** ..................................................................... 41  
   6.1. **COMPARISON SCENARIO** ........................................................................................................................................... 41  
   6.2. **MICROSOFT .NET / VISUAL STUDIO** ................................................................................................................................ 41  

---

**Table Listing**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>8</td>
</tr>
<tr>
<td>1.1</td>
<td>SCOPE OF THESIS</td>
<td>9</td>
</tr>
<tr>
<td>1.2</td>
<td>PROBLEM STATEMENT</td>
<td>10</td>
</tr>
<tr>
<td>1.3</td>
<td>GOALS</td>
<td>10</td>
</tr>
<tr>
<td>1.4</td>
<td>EXPECTED RESULTS</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>CASH &amp; BANK SYSTEM</td>
<td>11</td>
</tr>
<tr>
<td>2.1</td>
<td>RECEIPTS</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Direct Receipts by Cash Department</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Direct Credits into Bank Accounts</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>PAYMENTS</td>
<td>12</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Payments to Suppliers/ Service provider</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Process of Payment</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>MICROSOFT .NET</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>MICROSOFT VISUAL STUDIO</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>.NET BUILDING BLOCK SERVICES</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>.NET ENTERPRISE SERVERS</td>
<td>14</td>
</tr>
<tr>
<td>3.4</td>
<td>.NET FRAMEWORK</td>
<td>15</td>
</tr>
<tr>
<td>3.4.1</td>
<td>.NET Framework Components</td>
<td>15</td>
</tr>
<tr>
<td>3.4.1.1</td>
<td>Common Language Runtime (CLR)</td>
<td>15</td>
</tr>
<tr>
<td>3.4.1.2</td>
<td>Base Class Library (BCL)</td>
<td>16</td>
</tr>
<tr>
<td>3.4.1.3</td>
<td>.NET Framework Language</td>
<td>17</td>
</tr>
<tr>
<td>3.4.1.4</td>
<td>Common Language Specification (CLS)</td>
<td>17</td>
</tr>
<tr>
<td>3.4.1.5</td>
<td>Common Type Specification (CTS)</td>
<td>18</td>
</tr>
<tr>
<td>3.4.1.6</td>
<td>Microsoft Intermediate Language (MSIL)</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>JAVA ENTERPRISE EDITION</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>ENTERPRISE JAVA BEANS (EJB)</td>
<td>20</td>
</tr>
<tr>
<td>4.2</td>
<td>GLASSFISH SERVER 3.0</td>
<td>21</td>
</tr>
<tr>
<td>4.3</td>
<td>JDBC API</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>APPLICATION IMPLEMENTATION</td>
<td>23</td>
</tr>
<tr>
<td>5.1</td>
<td>APPLICATION DESIGN</td>
<td>23</td>
</tr>
<tr>
<td>5.2</td>
<td>.NET IMPLEMENTATION</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>JAVA EE IMPLEMENTATION</td>
<td>32</td>
</tr>
<tr>
<td>5.4</td>
<td>DATABASE DESIGN</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>COMPARISON BETWEEN PLATFORMS/ DEVELOPMENT ENVIRONMENTS</td>
<td>41</td>
</tr>
<tr>
<td>6.1</td>
<td>COMPARISON SCENARIO</td>
<td>41</td>
</tr>
<tr>
<td>6.2</td>
<td>MICROSOFT .NET / VISUAL STUDIO</td>
<td>41</td>
</tr>
</tbody>
</table>
6.2.1. Design................................................................................................................................. 41
6.2.2. Development ...................................................................................................................... 41
6.2.3. Object Orientation ............................................................................................................. 42
6.2.4. Data Types ......................................................................................................................... 42
6.2.5. Methods and Properties ...................................................................................................... 42
6.2.6. Reference Types ................................................................................................................. 42
6.2.7. Generics .............................................................................................................................. 43
6.2.8. Fields and Initialization ...................................................................................................... 43
6.2.9. Functional Programming ................................................................................................... 43
6.2.10. Runtime Type Information and Manipulation ................................................................... 43
6.2.11. Statements ......................................................................................................................... 43
6.2.12. Expressions and Operators ............................................................................................... 43
6.2.13. Exception ......................................................................................................................... 43
6.2.14. Arrays and Collections .................................................................................................... 44
6.2.15. Metadata Annotations ..................................................................................................... 44
6.2.16. Platform Support ............................................................................................................. 44

6.3. JAVA EE/NETBEANS IDE .................................................................................................... 44
6.3.1. Design................................................................................................................................. 44
6.3.2. Development ...................................................................................................................... 44
6.3.3. Object Orientation ............................................................................................................. 45
6.3.4. Data Types ......................................................................................................................... 45
6.3.5. Methods and Properties ...................................................................................................... 45
6.3.6. Reference Types ................................................................................................................. 45
6.3.7. Generics .............................................................................................................................. 45
6.3.8. Fields and Initialization ...................................................................................................... 45
6.3.9. Functional Programming ................................................................................................. 46
6.3.10. Runtime Type Information and Manipulation ................................................................. 46
6.3.11. Statements ......................................................................................................................... 46
6.3.12. Expressions and Operators ............................................................................................... 46
6.3.13. Exception ......................................................................................................................... 46
6.3.14. Arrays and Collections .................................................................................................... 46
6.3.15. Metadata Annotations ..................................................................................................... 47
6.3.16. Platform Support ............................................................................................................. 47

6.4. SIMILARITIES ......................................................................................................................... 47
6.4.1. Object Orientation ............................................................................................................. 47
6.4.2. Data Types ......................................................................................................................... 47
6.4.3. Methods and Properties ...................................................................................................... 48
6.4.4. Reference Types ................................................................................................................. 48
6.4.5. Generics .............................................................................................................................. 48
6.4.6. Fields and Initialization ...................................................................................................... 48
6.4.7. Functional Programming ................................................................................................. 48
6.4.8. Runtime Type Information and Manipulation ................................................................. 48
6.4.9. Statements ......................................................................................................................... 48
6.4.10. Expressions and Operators .............................................................................................. 49
6.4.11. Exceptions ......................................................................................................................... 49
6.4.12. Arrays and Collections .................................................................................................... 49
6.4.13. Metadata Annotations ..................................................................................................... 49
6.4.14. Platform Support ............................................................................................................. 49

6.5. DIFFERENCE....................................................................................................................... 49
6.5.1. Installation .......................................................................................................................... 50
6.5.2. Graphics ............................................................................................................................. 50
6.5.3. Components and Libraries ............................................................................................... 50
6.5.4. Third Party Components Support .................................................................................... 50
6.5.5. Debugging .......................................................................................................................... 51
APPENDIX B. NETBEANS IDE 6.0 IMPLEMENTATION ................................................................. 78

B.1. DBCONNECTION.JAVA ........................................................................................................ 78
B.2. ACCOUNTCODESDAL.JAVA .................................................................................................. 78
B.3. ACCOUNTSEARCH.JAVA ..................................................................................................... 79
B.4. CNBCODESDAL.JAVA ......................................................................................................... 79
B.5. GENERIC_FUNCTIONS.JAVA ................................................................................................. 80
B.6. BRVDAL.JAVA .................................................................................................................... 83
B.7. BPVDAL.JAVA .................................................................................................................... 85

APPENDIX A. VISUAL STUDIO 2008 IMPLEMENTATION ................................................................ 59

A.1. DBCONNECTION.CS .......................................................................................................... 59
A.2. FRMLOGIN.CS ...................................................................................................................... 59
A.3. LOGINDAL.CS ...................................................................................................................... 60
A.4. FRMACCOUNTCODES.CS .................................................................................................... 60
A.5. ACCOUNTCODESDAL.CS ..................................................................................................... 62
A.6. FRM CASHBANKCODES.CS .............................................................................................. 63
A.7. CNB CODES DAL.CS .......................................................................................................... 66
A.8. BANK RECEIPT VOUCHER.CS ............................................................................................ 67

7. CONCLUSIONS .................................................................................................................. 54

BIBLIOGRAPHY ...................................................................................................................... 56

APPENDIX A. VISUAL STUDIO 2008 IMPLEMENTATION ................................................................. 59

A.1. DBCONNECTION.CS .......................................................................................................... 59
A.2. FRMLOGIN.CS ...................................................................................................................... 59
A.3. LOGINDAL.CS ...................................................................................................................... 60
A.4. FRMACCOUNTCODES.CS .................................................................................................... 60
A.5. ACCOUNTCODESDAL.CS ..................................................................................................... 62
A.6. FRM CASHBANKCODES.CS .............................................................................................. 63
A.7. CNB CODES DAL.CS .......................................................................................................... 66
A.8. BANK RECEIPT VOUCHER.CS ............................................................................................ 67

APPENDIX B. NETBEANS IDE 6.0 IMPLEMENTATION ................................................................. 78

B.1. DBCONNECTION.JAVA ........................................................................................................ 78
B.2. ACCOUNTCODESDAL.JAVA .................................................................................................. 78
B.3. ACCOUNTSEARCH.JAVA ..................................................................................................... 79
B.4. CNBCODESDAL.JAVA ......................................................................................................... 79
B.5. GENERIC_FUNCTIONS.JAVA ................................................................................................. 80
B.6. BRVDAL.JAVA .................................................................................................................... 83
B.7. BPVDAL.JAVA .................................................................................................................... 85
List of Figures

Figure 4.1  Common Language Runtime
Figure 4.2  Base Class Library
Figure 4.3  .NET Framework Languages
Figure 4.4  Common Type Specification
Figure 5.1  NetBeans IDE 6.0, Editor, GUI, Other Visual Tools
Figure 5.2  GlassFish Server 3 Libraries
Figure 5.3  JDBC Database Connectivity
Figure 6.1  Login Screen
Figure 6.2  Main Screen
Figure 6.3  Accounts Head
Figure 6.4  Cash and Bank Codes
Figure 6.5  Bank Receipts Voucher
Figure 6.6  Search Screen
Figure 6.7  Bank Payment Voucher
Figure 6.8  Cash Receipt Voucher
Figure 6.9  Cash Payment Voucher
Figure 6.10 Cash & Bank Codes Window
Figure 6.11 Bank Receipt Voucher Window
Figure 6.12 Class Diagram of Cash & Bank
Figure 6.13 User Interface Diagram of Cash & Bank
Figure 7.1  Visual Studio GUI Interface
Figure 7.2  NetBeans New Project Window
Figure 7.3  NetBeans Enterprise Application Client Window
Figure 7.4  NetBeans JFrame Form Window
Figure 7.5  GUI of NetBeans IDE
Figure 7.6  New Entity Class Window
Figure 7.7  Entity Class Window
Figure 7.8  Cash & Bank Codes Window
Figure 7.9  Bank Receipt Voucher Window
Figure 7.10 Class Diagram of Cash & Bank
Figure 7.11 Component Diagram of Cash & Bank
Figure 7.12 User Interface Diagram of Cash & Bank
Figure 7.13 Relational Database Diagram of Cash & Bank
1. Introduction

This thesis describes comparison between two software development technologies .NET and Java Enterprise Edition. A financial application cash and bank is developed in both the technologies with same functional requirements. .NET and Java Enterprise Edition version 6 were opted to compare because these technologies are becoming popular nowadays in software development industry and Microsoft SQL Server is used for persistence storage of data.

Microsoft Visual Studio provides high level development environment for building their applications in .NET Platform. It provides key technologies to simplify the creation, deployment of secure applications and services. It also allows a new generation of windows based applications with many features available i.e. Windows Presentation Foundation (WPF) designer, windows forms integration, ClickOne improvement, multi-framework targeting, better Intellisence and code compilation support, organized “using” statements, refactoring enhancements, designer split view and efficient debugging in source code [1]. .NET framework consists of many languages i.e. C#, VB, J#, F# and C++ that interact with each other between different applications under the same solution. It can extract method by using dynamic link library that developed in other language of .NET framework. It contains several built-in libraries, components, functions, methods, XML and APIs. It also provides the efficient graphical user interface to design 2D and 3D applications i.e. Silver Light and Windows Presentation Foundation (WPF).

NetBeans IDE has broad Java framework platform in which user can developed large desktop web and mobile applications. NetBeans IDE contains hundreds of APIs itself that handles files, windows and actions. The NetBeans platform provide efficient support of customize APIs and add-in support to extend applications. It also provide distinct feature in application and each distinct feature provided by distinct NetBeans module. The main reusable features and components module system are lifecycle management, pluggability, service infrastructure, file system, windows system, standard user interface toolkit and advance data oriented components [2].
Microsoft SQL Server is a relational database management and analysis system whose primary functions is to store and retrieved data for e-commerce, line-of-business, and data warehousing solutions. There are different editions of SQL Server are launched for different audiences and for different workloads. SQL Server can easily handle disaster recovery solutions through AlwaysOn clusters, rapid data exploration and scalable business intelligence through tabular modeling, and new data management capability with Data Quality Services (DQS) [3]. SQL Server environment based on schema, Data Definition Language (DDL), Data Manipulation Language (DML) and Data Control Language (DCL) where schema contains description of objects i.e. base tables, views, constraints. DDL statement handles physical design and define a database including create, alter and drop tables but DML statement is used for implementation and query in the database like select, insert, update and delete. DCL maintain a database including administering privileges and committing data i.e. grant, add and revoke [4].

The project is based on solutions of problem of book keeping in respect of cash and bank transactions. This project is going to help to make book keeping improvement, their performance, efficiency and accuracy of transactions. Along with this it would be able to keep them updated with detail and summary transaction of an account. This project would be useful for any organization/ industry because it provides monitoring of the updated aspects like, voucher heads current account values, current value of head account, debit and credit transaction detail and summary, total sales, total purchase, total expenses, company balance, customer balance, accounts report like balance sheet, head ledger and other different types of reports.

Authors inspired by the existing Cash and Bank Management System that developed by EU to take external actions [5]. This application is design by keeping in view the existence of computerized environment to make user friendly.

1.1. Scope of Thesis
This thesis is a comparison of two different modern technologies that are getting more popular to design and develop the applications. Software engineering disciplines originate with numerous methods to increase the efficiency and quality of the software
development process including development models. Similarly, development platforms are invented that support high level features such as polymorphism, object orientated, generics, collections and inheritance to develop robust, efficient and correct application. The scope of the thesis is to evaluate both platform and compare the features based on practical and theoretical work after developed cash and bank application in .NET and Java Enterprise Edition.

1.2. Problem Statement
The purpose of the thesis is to compare both technologies and develop application in .NET and Java Enterprise Edition platform by using Visual Studio and NetBeans development environment respectively. Financial application need to develop by using built-in features and their available components and extract the results by comparing platform paradigms in both technologies based on study and practical experiments.

1.3. Goals
The Goals of the thesis is divided into sub goals. First, start to gathered requirement of cash & bank system and its workflow scenarios. Analyze the structure of both the technologies and develop cash and bank business application in .NET and Java Enterprise Edition.

The second main goal is to find differences and similarities. The comparison is based on theoretical background, key similarities and differences, practical aspects, typical usage during implementation of project with respect to design, development and maintenance. The thesis analyzes the comparison results of both platform and suggests which one is best.

1.4. Expected Results
The analysis results of both the platforms will help to decide which platform is best as per project nature, why respected technology is useful and what features are available. A cash and bank application will develop in .NET and Java Enterprise Edition to provide practical experiments results.
2. Cash & Bank System

Cash & Bank management system has been covered under the following main headings:

- Receipts
- Payments

 Receipts
The receipts at the organization are covered under this part of the report. Such receipts may be generated from different sources or may be in the form of collection of cash, cheques, pay orders etc from different locations of the company.

 Payments
The payment process is covered in the report to elaborate the mechanism, policies and procedures to be adopted by the organization. The payment process is divided in different stages on the basis of the payment process and procedures.

Cash & Bank management system incorporates all the transactions that covers cash transactions of the business processes and convert business process into the financial transactions.

2.1. Receipts
Money may be received by the organization in the following manner:

- Direct receipts by the cash department
- Direct credits into the bank accounts

2.1.1. Direct Receipts by Cash Department
The cash department receives cash from the companies on daily basis and deposited immediately in the bank accounts. The senior cash manager selects the name of the bank in which funds are to be deposited and forward to assistant cash manager to prepare a “Bank Deposit Statement” which consolidate all the cash / cheques to be deposited in bank account. Receipts from other sources are process in the same manner.
2.1.2. Direct Credits into Bank Accounts
Funds are directly credited by the branch manager into the bank accounts of the organization. A credit invoice is generally be raised by the bank in accordance with the normal banking practice and a copy is sent to the organization for intimation by the depositor. Sales can be made through credit cards are processed in the same manner. Funds are transferred by the bank within a day or two days of the occurrence of transaction. Direct credits by the parties are processed in the same manner.

2.2. Payments

2.2.1. Payments to Suppliers/Service provider
Payment is affected by crossed cheques / drafts marked as “Account Payee Only”. However, if payment made by cash or bearer cheque is specifically agreed with suppliers and such payment may be made within the limits as prescribed by the regulatory authorities. However for purposes of control, such cheques made for specific individuals. Cheque books must always be kept under lock and key. However it may be recommended that agreements may be linked with the suppliers/service providers before carrying out the assignment. This practice allows smooth and confident process.

2.2.2. Process of Payment

Initiation

All payments to supplier are duly supported by relevant documents, such as invoices and are written within the budget allocation. The concerned department initiate the payment process.

Processing

The following steps are involved in the processing of the payment.

The process for payment to the supplier/service provider is initiated after preparation of bill payment request form. The accounts and finance department acknowledge the
receipt of the bill payment request. The bill payment request form is received as per given number in the serial order.

The requesting department is able to follow up the progress of the payment with reference of bill payment request number. The accounts and finance department prepares a reconciliation of the gross amount invoice or advance to supplier with the net amount. It also ensure that the terms of payment agreed by the supplier whether the payment is to be made by the bank or through cash.
3. Microsoft .NET

.NET Platform is a set of tools and technologies that design to transform into full-scale able distributed computing platform. .NET platform provide a standard, countless opportunities for different services that allows programmers to develop powerful software solutions. It allows applications to communicate directly with other applications by using web technologies and supports the existing internet infrastructures like HTTP, XML, SOAP, REST and other web services [6].

3.1. Microsoft Visual Studio

Microsoft Visual Studio provides high level development environment to build applications in the .NET Platform. It provides key technologies to simplify the creation and deployment of secure web applications / web services. It also enables a new generation of windows, web and mobile based applications. It is complete development tools to design web applications, web services, desktop applications and mobile applications. It support many programming languages that use same integrated development environment i.e. VB, C++, C#, F#, J# and these languages facilitate in the construction of mixed-language solutions [7].

3.2. .NET Building Block Services

The .NET Building Block Services and distributed programmable services that are available both online and offline mode. A service can be invoked on a standalone computer that is not connected to the internet or accessed through internet. For example:

- .NET Calendar
- .NET Alerts
- .NET Passport
- .NET Documents

3.3. .NET Enterprise Servers

A family of server products of Microsoft provides back office services and these .NET Enterprise Servers are expansion of Microsoft's BackOffice suite with enhancements of
The .NET Framework. The .NET Enterprise Servers provide reliability, management and integration. [8].

- Microsoft SQL Server
- Microsoft BizTalk™ Server
- Microsoft Host Integration Server
- Microsoft Exchange Enterprise Server
- Microsoft Application Center
- Microsoft Commerce Server
- Microsoft ISA Server
- Microsoft Mobile Information Server

3.4. .NET Framework

.NET Framework is a new computing Microsoft platform that simplifies application development in highly distributed environment of the internet. It provides language interoperability across different programming languages. It is collection of elementary classes which designed to provide common services to run the applications [9]. The .NET Framework allows developers to build the following kinds of applications:

- XML Web Services
- Web Forms
- Windows Forms
- Win32 Console Applications etc.

3.4.1. .NET Framework Components

Following are the major components of .NET Framework:

3.4.1.1. Common Language Runtime (CLR)

Common Language Runtime (CLR) is the heart of .NET framework that manages execution of .NET code which is written in any supported languages. It allows to share common classes and provide services to make the development process easier [10]. The basic function of CLR is to take the code generated by the VB.NET compiler or any other
compiler and converts it to the native language of the current platform architecture.

![Common Language Runtime](image)

Common Language Runtime (CLR) is a virtual component that handles execution of application and known as Just in-time compiler. CLR performs just-in-time compiling of the code and makes the code more language and hardware independent. CLR also allows to ignore specific details that comes from different CPUs and manage the garbage collection, memory management, security issues and performance improvement [11].

### 3.4.1.2. Base Class Library (BCL)

Base Class Library (BCL) provides a collection of useful and reusable classes that are designed to integrate with CLR. The classes provided by .NET Framework are object-oriented and fully extendable. These classes are logically grouped into “Namespaces” that can access through using statement.
3.4.1.3. .NET Framework Language
The .NET Framework is language neutral – an application written in VB.NET can access in C# that is also can access in J# and so on.

Following Languages are currently available:

<table>
<thead>
<tr>
<th></th>
<th>APL</th>
<th>C++</th>
<th>C#</th>
<th>COBOL</th>
<th>Pascal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curriculum</td>
<td>Eiffel</td>
<td>Fortran</td>
<td>Haskell</td>
<td>J#</td>
<td></td>
</tr>
<tr>
<td>JScript</td>
<td>Mercury</td>
<td>Oz</td>
<td>Perl</td>
<td>Python</td>
<td></td>
</tr>
<tr>
<td>RPG</td>
<td>Scheme</td>
<td>Smalltalk</td>
<td>SML</td>
<td>VB</td>
<td></td>
</tr>
</tbody>
</table>

3.4.1.4. Common Language Specification (CLS)
The CLS defines the minimum standards that .NET languages must confirm.

Compliant Producer
The components in the language can be used by other language.

Consumer
The language can reuse classes produced in any other language and this feature is called “instantiate classes”.

Extender
Languages in this category can do more than just instantiate classes. It also extends those classes using the inheritance feature of .NET

3.4.1.5. Common Type Specification (CTS)

.NET Framework defines common type specification (CTS). Any data types are allowed to run inside the framework that belongs to Common language infrastructure (CLI) [12]. It is automatically converted to System.Int32 in VB.NET, C#, F#, J# and C++ according to Common Type System (CTS).

![Common Type Specification](image)

3.4.1.6. Microsoft Intermediate Language (MSIL)

When .NET application is compiled, it is converted from the language that was written in (VB.NET, C#, J#, F#, C++ etc) to a Managed Module and Managed Module is a low level set of instructions understood by Common Language Runtime. This Managed Module contains MSIL which is direct compiled form of source code and metadata.

Common Intermediate Language (CIL) previously called Microsoft Intermediate Language or MSIL is human-readable low level programming language which defined by the Common Language Infrastructure (CLI). CIL is assembled into an object code that kept as byte code-style format and it is totally stack based and object-oriented low level language. These byte codes are quickly translated into native code and executed. This well-defined universal syntax has major advantages i.e. performance improvement, platform independence as well as language interoperability [13].
4. Java Enterprise Edition

NetBeans IDE is an Integrated Development Environment to develop user friendly java based application manufactured by Sun Microsystems. NetBeans IDE provides wide range of components, libraries and controls to build large and small scale professional software applications. It also provides different types of project to develop desktop, web and mobile applications. There are lots of features like compilation, source code editor, debugging, tracing, solution explorer for viewing multiple projects, artifacts and also provide integration with other services. NetBeans IDE generate scripting for cleaning, build executable files, deploy projects, generating code from model and load different APIs that reduce work for developers. NetBeans IDE is developed to support different types of Java Enterprise Edition applications that can runs on Windows, Linux, Solaris, Mac and other platforms where Java Virtual Machine (JVM) is installed. Java EE is an industry standard for enterprise java computing and powerful platform for development of next-generation enterprise applications [14]. Java EE also describes the standard for developing multitier applications. There are many features such as JDBC API for database connectivity, "Write Once, Run Anywhere" portability, CORBA feature to interact with existing enterprise resources and provide security model to protect data of internet applications. It consists of compliance tests to ensure portability of other applications and provide full support for Java Servlets API, Enterprise JavaBeans (EJB) components, Java Server Pages (JSP) and XML technology [15].

Java Enterprise Edition (Java EE) is an oracle enterprise Java platform for developing large scale, robust, multi-tiered, reliable, scalable, portable and secure server side applications. It provides APIs and runtime environment for running enterprise software, including web services and networks. Java EE also provide functionality to deploy fault-tolerant application in distributed environment. Java EE decreases the complexity of multi-tier enterprise services and applications that can be speedily deployed on distributed environment [16].
4.1. Enterprise Java Beans (EJB)

Enterprise Java Beans (EJB) is an industry standard tool that is used to develop server side distributed components. Java programming language is used to develop EJB components but Enterprise Java Beans is different from Java Beans. Java Beans are classes in Java EE languages that contain core features, properties, methods and events. Java Beans are reusable components and libraries that are used to develop Java EE applications. Enterprise Java Beans are used for development of client and server side component based application.

Enterprise Java Beans (EJB) are useful to develop three tier architecture applications. Three tier architecture structure based on presentation layers, business logic layers and data access layer. Java beans handle client side operations which is fruitful for presentation layers but EJB works on server side operations that is not appropriate for presentation layers and cannot be used for graphical components. Java beans are useful for graphical components and development of EJB Components. Java Beans are effective for presentation and business logic layer but EJB are only used for business logic layers [17].
4.2. GlassFish Server 3.0

GlassFish Server is an open source application that introduce by Sun Microsystems in Java EE platform and it is reference implementation of Java EE that supports EJB, Java Persistence API, Java Message Service (JMS), Remote Method Invocation (RMI) Java Servlets and Java Server Pages (JSP). It provide different libraries to developed enterprise application which is scalable, portable and easy to integrate with legacy technologies.

GlassFish v3 is open, modular and extensible platform that is fast, easy and reliable application server. It support many dynamic languages i.e. Ruby, Groovy and available in Java EE 6 version. It also support Open Services Gateway initiative (OSGI), servlet and lots of other libraries that are available under GlassFish Server v3 as mentioned in figure 5.2. It is high performance application server for application deployment and also supports all major IDEs as well as other non-Java languages [21].

![Add Java EE Module](image)

*Figure 5.2: GlassFish Server 3 Libraries*
4.3. JDBC API

In the Java EE, JDBC API provides databases connectivity with all type of relational databases. By using the JDBC API, Java EE platform can execute SQL queries and write different kinds of scripts to retrieve expected results. JDBC API also provides interaction with multiple data sources in a distributed environment. [18].

In JDBC, the Driver.connect() method works with a connection object. Driver Manager calls getConnection() method and this method is overloaded to provide variety of ways to open the connections with database. getConnection() method has common string parameters that requires to establish the connection between front end and back end in different ways.[19]:-

- getConnection(String url)
- getConnection(String url, Properties prop)
- getConnection(String url, String user, String password)

![Figure 5.3: JDBC Database Connectivity](image-url)
5. Application Implementation

5.1. Application Design
Application design of both technologies is implemented in three tier architecture and layered approach. Presentation layer handles graphical user interface and business logic layer handles all types of calculation and apply business rules. Data access layer communicate with backend and execute database queries and statements. Project is implemented by using classes, method, properties and functions to fulfill the application requirements.

![Class Diagram of Cash & Bank](image-url)
Figure 7.11: Component Diagram of Cash & Bank

Figure 7.12: User Interface Diagram of Cash & Bank
5.2. .NET Implementation
To start the development of .NET application, a platform is required so first I installed Visual Studio .NET 2008 and configured environment. Installation of Visual Studio was easy as I followed wizard instructions. Visual Studio looks very good in design and have different stylish menus. I have good experiences to use Visual Studio .NET as I developed many projects. After all, I also took guidance from MSDN (Microsoft Development Network) and development methods. Graphical User Interface (GUI) of Visual Studio was professional and has many components on the left hand menus. Toolbox components are easy to use it is only drag and drop on the forms. There are different types of forms available in .NET library like MDI parent form, window form, about box form, splash screen form and help form but mostly window form is used as per requirements. After design all the forms in presentation layer, I created class library project for business logic and add classes to write programming instruction. Those classes are directly accessible that required frequently in other classes because these classes are created as static classes. Style class is created to load dynamic graphics in the other forms. Data Access Layer (DAL) class library project has different classes that communicate with front end and back end and execute statement of SQL Server.

Figure 7.1: Visual Studio GUI Interface
There are several ways to develop different type of projects. For this project, iterative development method is preferred because it solve many problems i.e. requirement specification, planning, design and continuously changing requirements. Many ambiguities in the requirements are solved in early stages that save time by reducing work and risk. Iterative approach provides several ways especially when requirements need to be change and also better for improvement of future development.

Project is mainly developed for any financial organization and its primary purpose is to kept different types of account heads and store book keeping transactions in detail level. Project handles different kinds of transaction via cash or in the shape of cheques. Project also handle different type of accounts head like expense, assets, liabilities and also handle chart of accounts in different levels. Project also distinguished to create cash and bank type definition separately as well as to keep bank account information.

The project also uses a login screen to login and verify the credential of user. On this screen user can login which is necessary for getting access to the application.

![Figure 6.1: Login Screen](image-url)
Figure 6.2. On the main screen of project, user is able to access every screen by select the transaction menu button. There are also other helping menus i.e. File, View and Users.

![Main Screen Image](image1)

Figure 6.2: Main screen

Figure 6.3 In the account codes screen of project user is able to add, edit, update, delete and search account head in chart of account.

![Account Codes Image](image2)
Figure 6.3: Accounts Head

Figure 6.4 In the cash and bank codes screen handles the bank detail against chart of account code and user is able to add, edit, update, delete and search the existing codes.

Figure 6.4: Cash and Bank Codes
Figure 6.5 In the bank receipt voucher screen handles all types of receipt transaction against chart of account code in the form of cheques where bank is debit and transactions detail is credit. User is also able to add, edit, update, delete, search and navigate the existing records.

Figure 6.6 User is able to search and navigate the existing records by enter document number.
Figure 6.7 In the bank payment voucher screen handles all types of payments that deal via cheques against chart of account code where bank is credit and transactions detail is debit.
Figure 6.8 In cash receipt voucher screen handles all types of receipt transaction against chart of account code in the form of cash where cash account is debit and transactions detail is credit.

Figure 6.8: Cash Receipt Voucher
5.3. Java EE Implementation

Before starting to implement, first I had to download and install NetBeans 6.9.1 and Java EE SDK Kit software. The full versions of Java EE are available for all windows version excluding Windows 3.1. In Windows 7, simply run the self-installing executable file and follows the wizard instructions. Installation wizard propose default installation directory which contains the Java SDK version number, i.e. jdk1.x.x and I preferred the same directory name for installation. After finish the installation process of Java SDK, there is one further step is to set the JDK bin directory to the execution path and this environment variable path is set under the system in control panel like this: c:\jdk\bin; [20].

This opening time of NetBeans IDE 6.9.1 was satisfied with the GUI editor and its functions. It contained all of those components that require for developing project. Now I start search and explore of NetBeans features and created new project after reading some tutorials.
There are different types of project listed under Java EE category and my project belongs to “Enterprise Application Client” that I opted. Next step is to choose the project name and project location in physical directory. After choosing project name and project location, the next step is to set server and settings. So select GlassFish Server 3 and Java EE version 6 from the dropdown list.
In the project tab window, project files are created. Under the source packages, all packages, forms, files and folders are created to start the design of project. First step is to create JFrame form to create container where all components are work under this JFrame form.
The GUI has drag and drop design that is easy to understand. The GUI use as seen in figure 7.5 is the development design interface and right window there is a palette of built-in components like labels, buttons and text boxes. In the left window, solution explorer windows showing all projects source code files. Developer can drag and drop any component onto the forms. A property window is available under the palette where developer can look and set any property of selected component. To the left side there is a window appears for project, file and services. Under the project, all files of the project are shown and developer can add new or existing files in the project. After compilation, an output window is visible that shows errors and warnings, if occurred.

As compared with Visual Studio, NetBeans GUI was not so good but all other programming paradigms are available that fulfill the project requirements. After making design, class library projects are created for business logic and data access.
After completion of design, new entity class is created by selecting the option entity classes from database under the persistence category as mentioned in figure 7.6. Entity class represents an object whose lifespan is longer than typical program execution. After selecting the file type, name and location of entity class is defined in the next step.
After the name and location, user enter data provider for database to establish JDBC connection that use the persistence library and finally add entity class where persistence unit interact with entity class.
After completion of design of all forms, project compile and run by pressing F5 button. Project builds successfully and ready to use. MDI parent form appears where all menus are available to select transaction form. Cash and bank code screen appears after click from menu where user define the codes for cash and bank.
Bank receipt voucher is main transaction form where user can records the receipt of banks under the selected head accounts. User need to enter document information, bank information, narration as well as account information.
5.4. Database Design

Database is collection of data and integrated database files that is required in every business application. There are lots of databases available in the market i.e. Oracle, SQL Server, SQLite, Microsoft Access etc but author choose SQL Server of persistence storage of data as it fulfill the requirement of the project and easy to implement. Data is stored in database tables as per defined schema. In the figure 7.6, relational database diagram of cash and bank shows relations with other tables. Cash and Bank application has many tables where three tables that store all the transactional records. Other tables are used for definitions, dimensions and user authentication. Database columns are created to store data in respective columns.

![Relational Database Diagram of Cash & Bank](image-url)

Figure 7.13: Relational Database Diagram of Cash & Bank
6. Comparison between Platforms/ Development Environments

6.1. Comparison Scenario
After developed cash and bank application in two different technologies, some specific comparison scenarios are formed. There are different areas that deeply checked during development of project. The primary purpose of the thesis is to compare .NET and Java EE platform features and its specialties but .NET and Java EE used Visual Studio and NetBeans development environment respectively. In this thesis, author also raised points of development environment comparisons as per study and experience.

6.2. Microsoft .NET / Visual Studio
Visual Studio.NET looks very professional with neat design of forms. It has rich toolbox library of components and every component has complete set of properties that helps to make good design i.e. control sizes, alignments, borders, shading, skins, etc. It has many development methods and features to interact with other libraries and integrate with modules.

6.2.1. Design
There are lots of built in forms available in controls library, i.e. MDI parent form, windows forms, about box form, inherited form and user control forms. Developer can also design own custom control form after choose from control library. To design the forms, developer selects the controls from the toolbox or can write code in designer file to get appearance as desired. Developer can drag and drop picture controls onto forms and also can change the background color and style to achieve good design. Developer has full access of changing colors, fonts and border styles of the controls. Data grid view and list view controls used to display the data in tabular formats and formatted by setting different properties. [Based on experience]

6.2.2. Development
After making the design, developer requires some events against the action that raise by end user. There are different events available for different controls that provided by .NET library. There are many language support by .NET framework
and developer can choose Visual Basic (VB), Csharp (C#), C++, J# and F# language to write programming instructions. It has many built in classes and functions to fulfill the requirement of the project. While writing the code in the code behind file, there is no problem encountered. It was easy to implement and retrieve the data from the database efficiently. .NET platform support different database connection drivers for different data sources and .NET platform provide all types of drivers for every database. [Based on experience]

6.2.3. Object Orientation
.NET supports partial classes with private access modifier, anonymous classes, operator overloading, indexer, implicit and explicit conversions but does not provides instance level inner classes, statement level anonymous classes and enums that can implement interfaces [24][26]. [Based on Study and Experience]

6.2.4. Data Types
.NET platform supports Single-root (unified) data type system as well as unsigned integers with 8, 16, 32 and 64 bits respectively. .NET provides high precision decimal number upto 128-bit (28 digits) decimal type and also supports complex numbers. Lifted (nullable), tuples and pointer data types are also available in .NET but does not provide arbitrary size decimals data type [22][28]. [Based on Study and Experience]

6.2.5. Methods and Properties
.NET platform provides explicit interface implementation, reference input/ output parameters, named arguments, generator methods, extension methods, conditional methods as well as partial methods but does not support statics import [28]. [Based on Study and Experience]

6.2.6. Reference Types
.NET platform provides proxy support by defining object contexts but does not support of soft and phantom references [29]. [Based on Study]
6.2.7. Generics
Runtime realization, value/ primitive type constraints, value/ primitive type support and constructor constraint are efficiently handles in .NET platform but does not support of migration compatibility [30][33]. [Based on Study]

6.2.8. Fields and Initialization
.NET platform provides object initializer as well as collection initializer but does not support instance initializer [22][25]. [Based on Study]

6.2.9. Functional Programming
.NET platform provides lots of procedure and feature in functional programming, i.e. methods references, closures, lambda expressions, expression trees, generic query language [27][22]. [Based on Study and Experience]

6.2.10. Runtime Type Information and Manipulation
.NET provides runtime generics realization and runtime type constructions as well as also provide late-bound (dynamic) type [25]. [Based on Study]

6.2.11. Statements
.NET support variable type inference that Java EE does not provide this functionality [22]. [Based on Study]

6.2.12. Expressions and Operators
.NET platform provides lifted operators, overflow controls as well as verbatim strings but do not provide feature of strict floating point evaluation [27]. [Based on Study and Experience]

6.2.13. Exception
.NET platform does not support checked exception features that is required to apply in the condition [28][31]. [Based on Study and Experience]
6.2.14. **Arrays and Collections**

.NET platform provide multidimensional arrays, rectangular (single array) and also limited support non-zero based arrays but does not provide feature of priority queue [22][23]. [Based on Study and Experience]

6.2.15. **Metadata Annotations**

.NET support positional arguments, specialization and conditional metadata, metadata annotation as class based, named arguments, default value as through initialization and nested types [22-24]. [Based on Study]

6.2.16. **Platform Support**

.NET platform provides partial support in AIX platform and full support for Windows Phone but does not provide facility for feature phones. In addition, for Symbian platform, .NET has been deprecated and for Blackberry it work with cross-compiler [23-25]. [Based on Study]

6.3. **Java EE / NetBeans IDE**

The design of NetBeans IDE looks like as old Linux operating system but beside that it seems professional with all the components and libraries in Java EE platform to fulfil the requirement of project.

6.3.1. **Design**

While implementing the design in NetBeans IDE, some problems was encountered. Sometimes the layout shows faults in components and sometime controls are overlap upon other controls. The alignments of the controls are not properly set and visible in another place onto the form. It is very difficult to know what to do if there is something wrong. This layout has many bugs and this reason it has minus point. [Based on Experience]

6.3.2. **Development**

NetBeans IDE is a functional palette that holds all the major components that required for developing the project. There are different categories of the palette which contains the different components. These are the components like text box,
lable, dropdown list and other components have a useful properties. Using the dropdown list, I found some problems by using static objects in the list. The components of Java EE platform missing some control like gridview, panel and documentation is also not well documented. NetBeans IDE becomes very slow while using longer amount of compilation and running time and sometimes it stops working. [Based on Experience]

6.3.3. Object Orientation
Java EE platform provides instance level inner classes and statement level anonymous classes as well as enum that can implement interfaces but does not provides partial classes, implicit anonymous classes, operator overloading, indexer, implicit and explicit conversions[24][26]. [Based on Study and Experience]

6.3.4. Data Types
Java EE platform provides reference type but does not support single-root (unified), complex numbers, tuples and pointers data types. High precision decimal number is treated as arbitrary size decimal that support below 128 bit [22][28]. [Based on Study and Experience]

6.3.5. Methods and Properties
Java EE platform provides static import functionality but does not support named arguments, generator methods, extension method, conditional methods as well as partial methods [28]. [Based on Study]

6.3.6. Reference Types
Java EE platform provides soft and phantom references and proxy support by defining proxy generation [29]. [Based on Study]

6.3.7. Generics
Java EE platform provides efficient migration compatibility facilities but does not provide runtime realization, value/ primitive type constraint/ support as well as constructor constraints [30][31]. [Based on Study]
6.3.8. **Fields and Initialization**
Java EE support instance initializer but does not provide object initializers and collection initializer that can be modeled [22][25]. [Based on Study]

6.3.9. **Functional Programming**
Java EE platform does not support expression trees, generic query language, method references, closures and lambda expression [27][22]. [Based on Study and Experience]

6.3.10. **Runtime Type Information and Manipulation**
Java EE platform provide runtime type information but with type erasure and do not support runtime generics realization, late bound dynamic type as well as runtime type construction but have some third party tools to use runtime type constructions [25]. [Based on Study]

6.3.11. **Statements**
There is no methods are available for variable type inference in Java EE [22]. [Based on Study]

6.3.12. **Expressions and Operators**
Java EE support strict floating point evaluation but does not provide lifted operators, overflow control and verbatim strings [27] [Based on Study and Experience].

6.3.13. **Exception**
Java EE platform provides checked exception feature which is helpful to develop quality product [28][31]. [Based on Study and Experience]

6.3.14. **Arrays and Collections**
Java EE fully support abstract data types but does not provide multidimensional array, rectangular (single array), non-zero based arrays, unified arrays and collections [23][22]. [Based on Study and Experience]
6.3.15. Metadata Annotations
Java EE does not provide positional arguments, specialization and conditional metadata but support metadata annotation as interface, user defined annotations and default value definition [22-24]. [Based on Study]

6.3.16. Platform Support
Java ME only support for Symbian platform and for Windows Phone, Java EE support if the code generated in C#. Java EE provides full support for feature phones and well as AIX platform [23-25]. [Based on Study]

6.4. Similarities
There are many features which are common in both the technologies that provide same functionality in different ways: -

6.4.1. Object Orientation
Classes, abstract classes, interfaces, deprecation are common in both the languages and access modifier define as public, package, protected and private in Java EE but in .NET it defines as public, internal, protected, private and protected internal. Only static inner classes are used in class level in Java EE and in .NET all inner classes are used in class level. Events are provided in standard libraries in Java EE and built-in languages features in .NET [26][27]. [Based on Study and Experience]

6.4.2. Data Types
Both Java EE and .NET supports signed integer 8, 16, 32 as well as 64 bits and also supports string as immutable reference type, unicode. Both the language provides reference types and arrays data types. Binary 32 and 64 floating point numbers are also available and also have character support data type. Date/Time treats in Java EE as reference type and .NET treats as value type. Enumerated types handle as reference type in Java EE and in .NET treats as scalar [28][22]. [Based on Study and Experience]
6.4.3. **Methods and Properties**
Both the language support abstract, sealed and value input parameter. By default virtual methods are defines as virtual and non-virtual in Java EE and .NET respectively [28]. [Based on Study]

6.4.4. **Reference Types**
Garbage collection and weak references are provides in both Java EE and .NET platform with difference syntax [28][29]. [Based on Study]

6.4.5. **Generics**
Both .NET and Java EE platform provides covariance, contra variance and relations constraints. Java EE provides only implicit reference type constraints and implantation deal as type erasure in Java EE and reification in .NET [30][33]. [Based on Study]

6.4.6. **Fields and Initialization**
Java EE and .NET provides fields, constants, static classes constructor, instance constructor, destructors as well as array initializers. Java EE provides object initialization of bottom up (field and constructor) approach and .NET support top-down (fields) as well as bottom up (constructor) approach [22][25]. [Based on Study]

6.4.7. **Functional Programming**
There is no functional programming features are similar in both platform but methods reference, closure, lambda expression can be solve by delegates in Java EE [27][22]. [Based on Study]

6.4.8. **Runtime Type Information and Manipulation**
Java EE and .NET support runtime type information but Java EE support with type erasure [25]. [Based on Study]

6.4.9. **Statements**
Both languages support loops, conditionals, flow control, assignment, exception control, variable declarations and also have features of deterministic disposal that
available in .NET from the beginning but added in latest versions of Java EE [22]. [Based on Study and Experience]

6.4.10. Expressions and Operators

There are different functionalities are common in Java EE and .NET i.e. arithmetic operator, logical operators, bitwise operator, conditional, string concatenation, casts, boxing and unboxing in implicit method in Java EE but unboxing in .NET should be explicit methods [27]. [Based on Study and Experience]

6.4.11. Exceptions

Both languages provide try, catch and finally block and have lots of built in exceptions classes that can be used as per requirements [28][31]. [Based on Study and Experience]

6.4.12. Arrays and Collections

Java EE and .NET provides one-dimensional, zero based index arrays, multidimensional arrays (arrays of arrays), maps/ dictionaries, sorted dictionaries, sets, sorted sets, lists/vectors, queues/ stack as well as multisets features [23][22]. [Based on Study and Experience]

6.4.13. Metadata Annotations

Both languages support named arguments and nested types in different way of carrying out a particular task and method [22-24]. [Based on Study]

6.4.14. Platform Support

.NET and Java EE provide support for Linux, Mac OS X, Solaris, FreeBSD, iOS, Windows, Windows Mobile and Android platform [23-25]. [Based on Study]

6.5. Difference

Java EE platform encountered a numerous bugs while developing the project and has limited number of components/ tools and its feature. Documentation of Java EE framework is poor as compared with .NET platform. Although, Java EE framework is open source platform but it has lots of annoying bugs in visual layout windows.
.NET platform has better documentation of their components than Java EE platform but some components of the Java EE platform are easier to understand than the .NET platform. It has much better debugger than Java EE platform because Java EE framework has a problem to point out the errors. When error occurs, it is possible to change the layout mode in .NET platform but that feature is not available in Java EE platform. .NET platform provides many different development features and complete set of components to develop quality application. It also has efficient support of debugging, executing and compilations. The detail differences of both the technologies are as follows: -

6.5.1. Installation

Visual Studio installation files are combined in whole package where the installation is easy and is not dependent on third party installation but NetBeans installation files are reliable on other installation packages and some of prerequisite need to download and install from internet. [Based on Experience]

6.5.2. Graphics

Visual Studio has many built in design, styles and graphical interface, icons, themes and skins but NetBeans IDE has limited built-in graphics library, icons, themes and skins. [Based on Experience]

6.5.3. Components and Libraries

Visual Studio contains complete sets of basic toolbox components and library but NetBeans does not have many basic components i.e. Group box and Data grid view controls. [Based on Experience]

6.5.4. Third Party Components Support

.NET platform provides excellent support of third party controls i.e. Telerik Controls, Syncfusion, Infragistics and Krypton. On the other hand, Java EE has limited support of third party controls but can support Add-in and Plugins more efficiently than .NET platform. [Based on Experience]
6.5.5. Debugging
.NET platform has excellent features of debugging and user can modify the programming instructions while application is in debugging mode but NetBeans does not provide facility to modify the programming instruction in debugging mode. [Based on Experience]

6.5.6. Documentation
.NET platform has excellent documentation that is available online and offline mode. Online documentation link is embedded in Visual Studio menu but Java EE does not provide documentation in visual designer. [Based on Experience]

6.5.7. Development Time
Projects in .NET platform are developed rapidly and reduce the development time but in Java EE project takes longer period to develop the application. [Based on Experience]

6.5.8. Security
There are many built in security classes and features are available in Java EE and have complete set of features to encrypt the data. .NET also has excellent built in security features but are not so powerful than Java EE [34]. [Based on Study]

6.5.9. Platform Support
.NET provides partial support in AIX platform and full support for Windows Phone but does not provide facility for feature phones. For Symbian platform, .NET has been deprecated and for Blackberry it work with cross-compiler. On the other hand, Java ME only support for Symbian platform and for Windows Phone, Java EE support if the code generated in C#. Java EE provides full support for feature phones and AIX platform. Overall, Java EE provide better facility for different platform [23-25][27]. [Based on Study]

6.5.10. Data Types
.NET support many different data types i.e. Single-root, unsigned integer, high precision decimal number, complex number, nullable, tuple and pointer data
types that are not available in Java EE. High precision decimal number data type is treated as arbitrary size decimal that support below 128 bit in Java EE [22][28]. [Based on Study and Experience]

6.5.11. Methods and Properties
.NET provides explicit interface implementation, reference input/ output parameters, named arguments, generator methods, extension methods, conditional methods as well as partial methods. On the other hand, Java EE provides static import functionality and does not support named arguments, generator methods, extension method, conditional methods as well as partial methods [28]. [Based on Study]

6.5.12. Reference Types
Java EE platform provides soft and phantom references and proxy support by defining proxy generation but .NET platform support proxy by defining object contexts but does not provide soft and phantom references [29]. [Based on Study]

6.5.13. Generics
Runtime realization, value/ primitive type constraints, value/ primitive type support and constructor constraint are efficiently handles in .NET platform but does not support migration compatibility. Java EE provides efficient migration compatibility facilities but does not provide runtime realization, value/ primitive type constraint/ support and constructor constraints [30][33]. [Based on Study]

6.5.14. Fields and Initialization
.NET platform provides object initializer as well as collection initializer and partial support for instance initializer but Java EE platform support instance initializer and does not provide object initializers and collection initializer [22][25]. [Based on Study]

6.5.15. Functional Programming
There are different functional programming features are available in .NET platform i.e. methods references, closures, lambda expressions, expression trees
and generic query language but Java EE platform does not provide expression trees, generic query language, closures and lambda expression [27][22]. [Based on Study]

6.5.16. Runtime Type Information and Manipulation

.NET platform provides runtime generics realization, runtime type construction and late-bound (dynamic) type. Java EE platform support runtime type information but with type erasure and do not support runtime generics realization, late bound dynamic type and runtime type construction [25]. [Based on Study]

6.5.17. Statements

.NET support variable type inference that handle in Java EE in different ways and used through alternate methods [22]. [Based on Study]

6.5.18. Exception

Checked exception functionality is available in Java EE that does no provides by .NET [28][31][Based on Study and Experience]

6.5.19. Arrays and Collections

.NET support multidimensional arrays, rectangular (single array) and also limited support non-zero based arrays but Java EE does not provide rectangular (single array), non-zero based arrays, unified arrays and collections [27][23]. [Based on Study and Experience]
7. Conclusions

This thesis work analysis of both technologies and its conclusion extract from theoretical background, key similarities and differences, practical aspects, typical usage and implementation of projects. The main purpose of this thesis is to develop cash and bank financial project in both the technologies with same functional requirements and compare the technical features in practical and theoretical aspects. .NET platform used Visual Studio development environment and Java EE platform used NetBeans IDE, eclipse and IntelliJ development environment. There is no doubt that eclipse development environment have excellent support of handling different types of plugin and their customization but author choose NetBeans because project requires different kinds of forms that NetBeans IDE support better than eclipse. Also NetBeans IDE is more intuitive and integrated than eclipse Rich Client Platform (RCP). The primary goal is to compare .NET and Java EE platform features but both platform used development environment to develop application so author compare the development environment features based on practical experiments.

Cash and bank project has been developed and each point is discussed in the report with respect to design, development and maintenance. Some of components are not available in Java EE like gridview and groupbox but practical work is implemented to use alternate components like listView and panel. There is different way of development while using different components. In comparison, the developed system is worked in both the technologies but it looks different in the style due to different layout and usage of alternate components. Java EE platform provides better security features over .NET. Java EE runs on many different operating systems without recompiling code, platform independent, better support of object oriented and exception handling, efficient migration compatibility and excellent support of reference type but its functional programming is weak, development process is more complex, more development cost, many missing features like expression trees, generic query language, method references, closures, generics realization, late bound dynamic type and runtime type construction, object initializers and collection initializer. .NET have many built in language features i.e. customization of user control, extend properties, parameterized constructor that inherited from
base class constructor, primitive types, complex and nullable datatype. .NET project can run on any operating system where installed Common Language Runtime (CLR) but it also has missing features like checked exception, soft and phantom references.

There will be more difficult analysis where most software falls. Author used both technologies professionally and know about strengths and weaknesses of both platform. If it is my choice to create a new application that would run on Windows operating system and limited development time then I would choose .NET platform over Java EE. From my experience, .NET platform provide very good object-oriented methods with many built in features, components and libraries. It is quite obvious that the architects of .NET platform spent lot of quality time to build this technology. It has few minor snags that author describes but the strengths are more than its minor weaknesses. Naturally, if project have long development period and will run on almost any operating system then Java EE platform seems to be the best choice but if application will run on Windows and have limited development time as well as no matter with the high security issues then .NET platform seems to a very excellent decision. From the author perspective after comparing both platforms and technologies, overall winner of this competition is .NET technology as it has better platform, performance, features and methods than Java EE platform. Although Java EE platform appear is fine development platform but it has many problems to use its platform, form, limited development features, components and methods. From the developer’s perspective, I would recommend Java EE platform for those projects that have longer period of time, required high level security as well as those kinds of projects which deals economical and financial applications. On the other hand, .NET platform can opted for those project that deals business, hardware, games development or any other type of projects. .NET platform is very good for middle level and small level project and performs better for fast or rapid development. So, both technologies in front of you, this is up to you what technology you choose depends on the project nature and its requirements.
Bibliography


[22] Allen Jones; Adam Freeman, C# for Java Developers, Microsoft Press, 2002-08-15


[31] Glenn W. Rowe, From Java to C#, Pearson Education, 2004,


[34] Denis Piliptchouk, Java vs. .NET Security, O'Reilly Media, May 2004
Appendix A. Visual Studio 2008 Implementation

A.1. DBConnection.cs

```csharp
public static class DBConnection
{
    public static SqlConnection GetConnection()
    {
        SqlConnection cn = null; string conStr;
        TextReader tr = new StreamReader("Config.ini");
        conStr = tr.ReadLine();
        tr.Close();
        return new SqlConnection(conStr);
    }
}
```

A.2. frmLogin.cs

```csharp
public partial class frmLogin : Form
{
    bool isValid = false;
    public frmLogin()
    {
        InitializeComponent();
    }

    private void btnClose_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void Login()
    {
        if (LoginDAL.ValidateUser(txtUsername.Text, txtPassword.Text) == true)
        {
            this.Close();
            isValid = true;
        }
        else
        {
            MessageBox.Show("Invalid Username and password");
        }
    }

    private void btnLogin_Click(object sender, EventArgs e)
    {
        Login();
    }

    private void frmLogin_FormClosed(object sender, FormClosedEventArgs e)
    {
        if (isValid == false)
        {
            Application.Exit();
        }
    }

    private void txtPassword_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.Enter)
        {
            Login();
        }
    }

    private void txtUsername_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.Enter)
        {
            Login();
        }
    }
}
```
A.3. LoginDAL.cs

```csharp
public static class LoginDAL
{
    private static string GlobalVar = "";
    public static bool isValidate;
    public static bool ValidateUser(string username, string password)
    {
        string SqlQuery = "";
        DataTable dt = new DataTable();
        SqlQuery = "Select * from USERS where Username = '" + username + '" and Password = '" + password + '";'';
        dt = Generics_Functions.LoadDataTableMemory(SqlQuery);
        if (dt.Rows.Count == 1)
        {
            return isValidate = true;
        }
        return isValidate = false;
    }
}
```

A.4. frmAccountCodes.cs

```csharp
public partial class frmAccountCodes : Form
{
    public DataTable dt;
    public frmAccountCodes()
    {
        InitializeComponent();
    }

    public void frmAccountCodes_Load(object sender, EventArgs e)
    {
        frmAccountCodes frm = new frmAccountCodes();
        Style.Mth_ControlStyle(frm, "Account Codes", lblHeading, pbLeft);
        LoadRecords();
        AccountInfoBLL.ShowRecord(txtAccountCode, txtAccountName, GobalDeclarations.currentRow, dt);
        GobalDeclarations.MenuItemsDefault(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
    }

    public void LoadRecords()
    {
        dt = AccountCodesDAL.Display();
    }

    public void btnNew_Click(object sender, EventArgs e)
    {
        AccountInfoBLL.UnLocksControls(txtAccountCode, txtAccountName);
        AccountInfoBLL.ClearControls(txtAccountCode, txtAccountName);
        GobalDeclarations.MenuItemsNew(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
    }

    public void btnSave_Click(object sender, EventArgs e)
    {
        if (AccountInfoBLL.FieldChecks(txtAccountCode.Text, txtAccountName.Text) == "true")
        {
            string isSuccess = AccountCodesDAL.Add(txtAccountCode.Text, txtAccountName.Text);
            if (isSuccess == "true")
            {
                MessageBox.Show("Record successfully Added");
                LoadRecords();
                AccountInfoBLL.LocksControls(txtAccountCode, txtAccountName);
                GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
            }
        }
    }
}
```
else
    MessageBox.Show(isSuccess.ToString());
}

public void btnEdit_Click(object sender, EventArgs e)
{
    AccountInfoBLL.UnLocksControls(txtAccountCode, txtAccountName);
    GobalDeclarations.MenuItemsEdit(btnNew, btnSave, btnEdit, btnUpdate, btnDelete,
    btnCancel, 1);
}

public void btnUpdate_Click(object sender, EventArgs e)
{
    if (AccountInfoBLL.FieldChecks(txtAccountCode.Text, txtAccountName.Text) == "true")
    {
        string isSuccess = AccountCodesDAL.Update(txtAccountCode.Text,
        txtAccountName.Text);
        if (isSuccess == "true")
        {
            MessageBox.Show("Record successfully Updated");
            LoadRecords();
            AccountInfoBLL.LockControls(txtAccountCode, txtAccountName);
            GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate,
            btnDelete, btnCancel, 1);
        }
        else
            MessageBox.Show(isSuccess.ToString());
    }
}

public void btnDelete_Click(object sender, EventArgs e)
{
    string isSuccess = AccountCodesDAL.Delete(txtAccountCode.Text);
    if (isSuccess == "true")
    {
        MessageBox.Show("Record successfully Deleted");
        LoadRecords();
        AccountInfoBLL.UnLocksControls(txtAccountCode, txtAccountName,
        dt);
        AccountInfoBLL.LockControls(txtAccountCode, txtAccountName);
        GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate,
        btnDelete, btnCancel, 1);
    }
    else
        MessageBox.Show(isSuccess.ToString());
}

public void btnFirst_Click(object sender, EventArgs e)
{
    AccountInfoBLL.First(txtAccountCode, txtAccountName, dt);
}

public void btnPrevious_Click(object sender, EventArgs e)
{
    AccountInfoBLL.Previous(txtAccountCode, txtAccountName, dt);
}

public void btnNext_Click(object sender, EventArgs e)
{
    AccountInfoBLL.Next(txtAccountCode, txtAccountName, dt);
}

public void btnLast_Click(object sender, EventArgs e)
{
    AccountInfoBLL.Last(txtAccountCode, txtAccountName, dt);
}

public void btnCancel_Click(object sender, EventArgs e)
{
A.5. AccountCodesDAL.cs
public static class AccountCodesDAL
{
  public static string Add(string AccCode, string AccName)
  {
    string query, isSuccess = "";
    DataTable dt;
    try
    {
      query = "SELECT * FROM Accounts WHERE AccCode='' + AccCode + '';";
      dt = Generics_Functions.LoadDataTableMemory(query);
      if (dt.Rows.Count == 0)
      {
        query = "INSERT INTO Accounts (AccCode, AccName) VALUES ('" + AccCode + ",
        " + AccName + ");"
        isSuccess = Generics_Functions.ExecuteNonQuery(query);
        isSuccess = "true";
      }
      else
      {
        isSuccess = "Record already exist, please enter another code";
      }
    } catch (Exception ex)
    {
      return isSuccess = ex.Message.ToString();
    }
    return isSuccess;
  }
  public static string Update(string AccCode, string AccName)
  {
    string isSuccess="";
    try
    {
      string query = "UPDATE Accounts SET AccName='" + AccName + "; WHERE AccCode='" + 
      AccCode + ";";"
      isSuccess = Generics_Functions.ExecuteNonQuery(query);
    } catch (Exception ex)
    {
      return isSuccess = ex.Message.ToString();
    }
    return isSuccess;
  }
  public static string Delete(string AccCode)
  {
    string isSuccess="";
    try
    {
      string query = "DELETE FROM Accounts WHERE AccCode='" + AccCode + ";"
      isSuccess = Generics_Functions.ExecuteNonQuery(query);
    } catch (Exception ex)
{ return isSuccess = ex.Message.ToString(); } return isSuccess;
}
public static DataTable Display()
{
    string query = "SELECT * FROM Accounts";
    DataTable dt = Generics_Functions.LoadDataTableMemory(query);
    return dt;
}
public static DataTable Search(string AccCode)
{
    string query = "SELECT * FROM Accounts WHERE AccCode LIKE '%" + AccCode + "%' OR AccName LIKE '%" + AccCode + "%';
    DataTable dt = Generics_Functions.LoadDataTableMemory(query);
    return dt;
}

A.6. frmCashBankCodes.css

class frmCashBankCodes : Form
{
    public partial class frmCashBankCodes : Form
    {
        public DataTable dt;
        public frmCashBankCodes()
        {
            InitializeComponent();
        }
        private void frmCashBankCodes_Load(object sender, EventArgs e)
        {
            frmCashBankCodes frm = new frmCashBankCodes();
            Style.Mch_ControlStyle(frm, "Cash && Bank Codes", lblHeading, pbLeft);
            CnBCodesDAL.FillAcctTypeCnB(cmbAccType);
            CnBCodesDAL.FillAcctTypeCS(cmbAcctType);
            LoadRecords();
            CnBCodesBLL.ShowRecord(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAccntNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity, GobalDeclarations.currentRow, dt);
            CnBCodesBLL.LocksControls(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAccntNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity);
            GobalDeclarations.MenuItemsDefault(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
        }
        private void btnNew_Click(object sender, EventArgs e)
        {
            CnBCodesBLL.UnLocksControls(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAccntNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity);
            CnBCodesBLL.ClearControls(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAccntNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity);
            GobalDeclarations.MenuItemsNew(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
        }
        private void btnSave_Click(object sender, EventArgs e)
        {
            if (CnBCodesBLL.FieldChecks(txtAcctCode.Text) == "true")
            {
                string isSuccess = CnBCodesDAL.Add(cmbAccType.SelectedValue.ToString(), txtCBCode.Text, txtAcctCode.Text, txtBankAcctTitle.Text, txtBankAccntNo.Text, cmbAcctType.SelectedValue.ToString(), txtBrCode.Text, txtBrName.Text, txtAddress.Text, txtCity.Text);
                if (isSuccess == "true")
                {
                }
            }
}
}
```csharp
MessageBox.Show("Record successfully Added");
LoadRecords();
CnBCodesBLL.LocksControls(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity);
GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
}
else
    MessageBox.Show(isSuccess.ToString());
}
public void LoadRecords()
{
    dt = CnBCodesDAL.Display();
}
private void txtAcctCode_Validated(object sender, EventArgs e)
{
    FetchAccountName();
}
public void FetchAccountName()
{
    txtAcctName.Text = CnBCodesDAL.LoadAcctInfo(txtAcctCode.Text);
}
private void btnEdit_Click(object sender, EventArgs e)
{
    CnBCodesBLL.UnlocksControls(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity);
    GobalDeclarations.MenuItemsEdit(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
}
private void btnUpdate_Click(object sender, EventArgs e)
{
    if (CnBCodesBLL.FieldChecks(txtAcctCode.Text) == "true")
    {
        string isSuccess = CnBCodesDAL.Update(cmbAccType.SelectedValue.ToString(), txtCBCode.Text, txtAcctCode.Text, txtBankAcctTitle.Text, txtBankAcctNo.Text, cmbAcctType.SelectedValue.ToString(), txtBrCode.Text, txtBrName.Text, txtAddress.Text, txtCity.Text);
        if (isSuccess == "true")
        {
            MessageBox.Show("Record successfully Updated");
            LoadRecords();
            CnBCodesBLL.LocksControls(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity);
            GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
        }
        else
            MessageBox.Show(isSuccess.ToString());
    }
}
private void btnDelete_Click(object sender, EventArgs e)
{
    string isSuccess = CnBCodesDAL.Delete(txtCBCode.Text);
    if (isSuccess == "true")
    {
        MessageBox.Show("Record successfully Deleted");
        LoadRecords();
        CnBCodesBLL.Previous(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity, dt);
    }
    else
        MessageBox.Show(isSuccess.ToString());
}
```
```csharp
CnBCodesBLL.LocksControls(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity);
GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
else
    MessageBox.Show(isSuccess.ToString());
}

private void btnCancel_Click(object sender, EventArgs e)
{
    CnBCodesBLL.Previous(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity, dt);
    CnBCodesBLL.LocksControls(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity);
    GobalDeclarations.MenuItemsDefault(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
}

private void btnFirst_Click(object sender, EventArgs e)
{
    CnBCodesBLL.First(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity, dt);
}

private void btnPrevious_Click(object sender, EventArgs e)
{
    CnBCodesBLL.Previous(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity, dt);
}

private void btnNext_Click(object sender, EventArgs e)
{
    CnBCodesBLL.Next(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity, dt);
}

private void btnLast_Click(object sender, EventArgs e)
{
    CnBCodesBLL.Last(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity, dt);
}

private void txtAcctCode_TextChanged(object sender, EventArgs e)
{
    FetchAccountName();
}

private void btnAcctSrch_Click(object sender, EventArgs e)
{
    frmAcctSearch frm = new frmAcctSearch();
    frm.ShowDialog();
txtAcctCode.Text = GobalDeclarations.srchAcct;
txtAcctName.Text = CnBCodesDAL.LoadAcctInfo(GobalDeclarations.srchAcct);
}

private void btnDocSrch_Click(object sender, EventArgs e)
{
    frmDocSearch frm = new frmDocSearch();
    GobalDeclarations.vType = "CnBCode";
    frm.ShowDialog();
    DataTable dtSearch = CnBCodesDAL.Search(GobalDeclarations.DocNo);
    CnBCodesBLL.ShowRecord(cmbAccType, txtCBCode, txtAcctCode, txtBankAcctTitle, txtAcctName, txtBankAcctNo, cmbAcctType, txtBrCode, txtBrName, txtAddress, txtCity, 0, dtSearch);
}
```
A.7. CnBCodesDAL.cs

```csharp
public static class CnBCodesDAL
{
    public static string GenerateCode()
    {
        string query = "SELECT ISNULL(MAX(CnBCode),'00000') CnBCode FROM CnBCode";
        string res = Generics_Functions.GenerateCode(query);
        return res;
    }

    public static string Add(string AccType, string CnBCode, string AcctCode, string BName, string BAccountNo, string BankAcctType, string BrnchCod, string BrnchName, string Address, string City)
    {
        string query = "INSERT INTO CnBCode (CnBCode, AcctType, AcctCode, BName, BAccountNo, BankAcctType, BrnchCod, BrnchName, Address, City) VALUES ('" + CnBCode + "," + AccType + "," + AcctCode + "," + BName + "," + BAccountNo + "," + BankAcctType + "," + BrnchCod + "," + BrnchName + "," + Address + "," + City + ");"
        string isSuccess = Generics_Functions.ExecuteNonQuery(query);
        return isSuccess;
    }

    public static string Update(string AccType, string CnBCode, string AcctCode, string BName, string BAccountNo, string BankAcctType, string BrnchCod, string BrnchName, string Address, string City)
    {
        string query = "UPDATE CnBCode SET AcctType='" + AccType + ", AcctCode='" + AcctCode + ", BName='" + BName + ", BAccountNo='" + BAccountNo + ", BrnchCod='" + BrnchCod + ", BrnchName='" + BrnchName + ", Address='" + Address + ", City='" + City + " WHERE CnBCode='" + CnBCode + ");"
        string isSuccess = Generics_Functions.ExecuteNonQuery(query);
        return isSuccess;
    }

    public static string Delete(string CnBCode)
    {
        string query = "DELETE FROM CnBCode WHERE CnBCode='" + CnBCode + ");"
        string isSuccess = Generics_Functions.ExecuteNonQuery(query);
        return isSuccess;
    }

    public static DataTable Display()
    {
        string query = "SELECT CnBCode, AcctType, AcctCode, BName, BAccountNo, BrnchCod, BrnchName, BankAcctType, Address, City FROM CnBCode";
        DataTable dt = Generics_Functions.LoadDataTableMemory(query);
        return dt;
    }

    public static string LoadAcctInfo(string AccCode)
    {
        string query = "SELECT AccName FROM Accounts WHERE AccCode='" + AccCode + ");"
        string res = Generics_Functions.FetchAcctDescription(query);
        return res;
    }

    public static DataTable Search(string CnBCode)
    {
        string query = "SELECT CnBCode, AcctType, AcctCode, BName, BAccountNo, BrnchCod, BrnchName, BankAcctType, Address, City FROM CnBCode WHERE CnBCode LIKE '%" + CnBCode + ");"
        DataTable dt = Generics_Functions.LoadDataTableMemory(query);
        return dt;
    }

    public static void FillAcctTypeCnB(ComboBox cmbAcctTypeCnB)
    {
        string query = "SELECT Code, AccountType FROM AcctType WHERE AccountTypeID = 1"
        Generics_Functions.FillDropDown(cmbAcctTypeCnB, "Code", "AccountType", query);
    }

    public static void FillAcctTypeCS(ComboBox cmbAcctTypeCnB)
    {
        string query = "SELECT Code, AccountType FROM AcctType WHERE AccountTypeID = 2"
        Generics_Functions.FillDropDown(cmbAcctTypeCnB, "Code", "AccountType", query);
    }
}
```
public partial class BankReceiptVoucher : Form
{
    public DataTable dt;
    public BankReceiptVoucher()
    {
        InitializeComponent();
    }
}

private void BankReceiptVoucher_Load(object sender, EventArgs e)
{
    BankReceiptVoucher frm = new BankReceiptVoucher();
    string lAccountType = frmMain.gAccountType.ToString();

    if (lAccountType == "BRV")
    {
        Style.Mth_ControlStyle(frm, "Bank Receipt Voucher", lblHeading, pbLeft);
        lblType.Text = "BRV";
        lblChqRefNo.Text = "Cheque No.";
        gbxBankInfo.Visible = true;
        gbxCashInfo.Visible = false;
        BRVDAL.LoadBanks(cmbBankName);
        BRVDAL.LoadTransAccts(cmbAcctCode);
        LoadRecords();
        BRV BLL.ShowRecord(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, dgvAcct, GlobalDeclarations.currentRow, dt);
    }
    if (lAccountType == "BPV")
    {
        Style.Mth_ControlStyle(frm, "Bank Payment Voucher", lblHeading, pbLeft);
        lblType.Text = "BPV";
        lblChqRefNo.Text = "Cheque No.";
        gbxBankInfo.Visible = true;
        gbxCashInfo.Visible = false;
        BPVDAL.LoadBanks(cmbBankName);
        BPVDAL.LoadTransAccts(cmbAcctCode);
        LoadRecords();
        BPV BLL.ShowRecord(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, dgvAcct, GlobalDeclarations.currentRow, dt);
    }
    if (lAccountType == "CRV")
    {
        Style.Mth_ControlStyle(frm, "Cash Receipt Voucher", lblHeading, pbLeft);
        lblType.Text = "CRV";
        lblChqRefNo.Text = "Ref No.";
        gbxBankInfo.Visible = false;
        gbxCashInfo.Visible = true;
        CRVDAL.LoadBanks(cmbCashName);
        CRVDAL.LoadTransAccts(cmbAcctCode);
        LoadRecords();
        CRV BLL.ShowRecord(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, GlobalDeclarations.currentRow, dt);
    }
    if (lAccountType == "CPV")
    {
        Style.Mth_ControlStyle(frm, "Cash Payment Voucher", lblHeading, pbLeft);
        lblType.Text = "CPV";
        lblChqRefNo.Text = "Ref No.";
        gbxBankInfo.Visible = false;
        gbxCashInfo.Visible = true;
    }
private void btnDocSrch_Click(object sender, EventArgs e)
{
    frmDocSearch frm = new frmDocSearch();
    frm.ShowDialog();
    if (lblType.Text == "BRV")
    {
        DataTable dtSearch = BRVDAL.Search(GobalDeclarations.DocNo, GobalDeclarations.vType, GobalDeclarations.isDate, GobalDeclarations.vDate);
        BRVBLL.ShowRecord(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, 0, dtSearch);
    }
    if (lblType.Text == "BPV")
    {
        DataTable dtSearch = BPVDAL.Search(GobalDeclarations.DocNo, GobalDeclarations.vType, GobalDeclarations.isDate, GobalDeclarations.vDate);
        BPVBLL.ShowRecord(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, 0, dtSearch);
    }
    if (lblType.Text == "CRV")
    {
        DataTable dtSearch = CRVDAL.Search(GobalDeclarations.DocNo, GobalDeclarations.vType, GobalDeclarations.isDate, GobalDeclarations.vDate);
        CRVBLL.ShowRecord(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, 0, dtSearch);
    }
    if (lblType.Text == "CPV")
    {
        DataTable dtSearch = CPVDAL.Search(GobalDeclarations.DocNo, GobalDeclarations.vType, GobalDeclarations.isDate, GobalDeclarations.vDate);
        CPVBLL.ShowRecord(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, 0, dtSearch);
    }
}

private void btnAcctSrch_Click(object sender, EventArgs e)
{
    try
    {
        frmAcctSearch frm = new frmAcctSearch();
        frm.ShowDialog();
        cmbAcctCode.SelectedValue = GobalDeclarations.srchAcct;
        txtAcctTitle.Text = CnBCodesDAL.LoadAcctInfo(GobalDeclarations.srchAcct);
    }
    catch {}
}

private void btnNew_Click(object sender, EventArgs e)
{
    if (lblType.Text == "BRV")
    {
        BRVBLL.UnLocksControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct);
        BRVBLL.ClearControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, txtAcctTitle, txtChqNo, dtChqDate, txtNarration, txtAmount, dgvAcct);
    }
    if (lblType.Text == "BPV")
    {

if (lblType.Text == "CRV")
{
    CRVBLL.UnLocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, pnldgvAcct);
    CRVBLL.ClearControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, cmbAcctCode, txtAcctTitle, txtChqNo, dtChqDate, txtNarration, txtAmount, dgvAcct);
}

if (lblType.Text == "CPV")
{
    CPVBLL.UnLocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, pnldgvAcct);
    CPVBLL.ClearControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, cmbAcctCode, txtAcctTitle, txtChqNo, dtChqDate, txtNarration, txtAmount, dgvAcct);
    GlobalDeclarations.MenuItemsNew(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
}

public int GridColumnSum()
{
    int sum = 0;
    foreach (DataGridViewRow row in dgvAcct.Rows)
    {
        int c = 5;
        sum = Convert.ToInt32(sum) + Convert.ToInt32(row.Cells[c].Value);
    }
    return sum;
}

private void btnSave_Click(object sender, EventArgs e)
{
    if (lblType.Text == "BRV")
    {
        if (BRVBLL.FieldChecks(cmbBankName.Text, dgvAcct) == "true")
        {
            string isSuccess = BRVDAL.Add(txtDocNo.Text, Convert.ToString(dtDocDate.Value.ToShortDateString()), lblType.Text, cmbBankName.SelectedValue.ToString(), txtBankNarration.Text);
            if (isSuccess == "true")
            {
                for (int i = 0; i < dgvAcct.Rows.Count; i++)
                {
                }
            }
            MessageBox.Show("Record successfully Added");
            LoadRecords();
            BRVBLL.LocksControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, pnldgvAcct);
            GlobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
        }
        else
        {
            MessageBox.Show(isSuccess.ToString());
        }
    }
    else
    {
        MessageBox.Show(isSuccess.ToString());
    }
}
```csharp
if (lblType.Text == "BPV")
{
    if (BPVBLL.FieldChecks(cmbBankName.Text, dgvAcct) == "true")
    {
        txtDocNo.Text = BPVDAL.GenerateCode(lblType.Text);
        string isSuccess = BPVDAL.Add(txtDocNo.Text, Convert.ToString(dtDocDate.Value.ToShortDateString()), cmbBankName.SelectedValue.ToString(), txtBankNarration.Text);
        if (isSuccess == "true")
        {
            for (int i = 0; i < dgvAcct.Rows.Count; i++)
            {
            }
            if (isSuccess == "true")
            {
                MessageBox.Show("Record successfully Added");
                LoadRecords();
                BPVBLL.LocksControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, pnldgvAcct);
                GlobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
            }
        }
        else
        {
            MessageBox.Show(isSuccess.ToString());
        }
    }
    else
    {
        MessageBox.Show(isSuccess.ToString());
    }
}
else
{
    MessageBox.Show(isSuccess.ToString());
}

else
{
    MessageBox.Show(isSuccess.ToString());
}

if (lblType.Text == "CRV")
{
    if (CRVBLL.FieldChecks(cmbCashName.Text, dgvAcct) == "true")
    {
        txtDocNo.Text = CRVDAL.GenerateCode(lblType.Text);
        string isSuccess = CRVDAL.Add(txtDocNo.Text, Convert.ToString(dtDocDate.Value.ToShortDateString()), cmbCashName.SelectedValue.ToString(), txtBankNarration.Text);
        if (isSuccess == "true")
        {
            for (int i = 0; i < dgvAcct.Rows.Count; i++)
            {
            }
            if (isSuccess == "true")
            {
                MessageBox.Show("Record successfully Added");
                LoadRecords();
                CRVBLL.LocksControls(txtDocNo, dtDocDate, cmbCashName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, pnldgvAcct);
                GlobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
            }
        }
        else
        {
            MessageBox.Show(isSuccess.ToString());
        }
    }
    else
    {
        MessageBox.Show(isSuccess.ToString());
    }
}
```

```csharp
    isSuccess = CRVDAL.AddDetailSngRow(txtDocNo.Text, lblType.Text, Convert.ToString(dtDocDate.Value.ToShortDateString()), cmbCashName.SelectedValue.ToString(), txtBankNarration.Text, GridColumnSum());
    if (isSuccess == "true")
    {
        MessageBox.Show("Record successfully Added");
        LoadRecords();
        CRVBLL.LocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, pnldgvAcct);
        GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
    }
    else
    MessageBox.Show(isSuccess.ToString());
    }
    else
    MessageBox.Show(isSuccess.ToString());
    }
}
if (lblType.Text == "CPV")
{
    if (CPVBLL.FieldChecks(cmbCashName.Text, dgvAcct) == "true")
    {
        txtDocNo.Text = CPVDAL.GenerateCode(lblType.Text);
        string isSuccess = CPVDAL.Add(txtDocNo.Text, Convert.ToString(dtDocDate.Value.ToShortDateString()), lblType.Text, cmbCashName.SelectedValue.ToString(), txtBankNarration.Text);
        if (isSuccess == "true")
        {
            for (int i = 0; i < dgvAcct.Rows.Count; i++)
            {
                if (isSuccess == "true")
                {
                    MessageBox.Show("Record successfully Added");
                    LoadRecords();
                    CPVBLL.LocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, pnldgvAcct);
                    GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
                }
                else
                MessageBox.Show(isSuccess.ToString());
                }
            else
            MessageBox.Show(isSuccess.ToString());
            }
    }
    }
}
public void LoadRecords()
{
    if (lblType.Text == "BRV")
    dt = BRVDAL.Display();
    if (lblType.Text == "BPV")
```
    dt = BPVDAL.Display();
if (lblType.Text == "CRV")
    dt = CRVDAL.Display();
if (lblType.Text == "CPV")
    dt = CPVDAL.Display();
}

private void btnEdit_Click(object sender, EventArgs e)
{
if (lblType.Text == "BRV")
    BRVBLL.UnLocksControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo,
txtBranchCode, txtBranchName, txtBankNarration, pnldgvAcct);
if (lblType.Text == "BPV")
    BPVBLL.UnLocksControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo,
txtBranchCode, txtBranchName, txtBankNarration, pnldgvAcct);
if (lblType.Text == "CRV")
    CRVBLL.UnLocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration,
pnldgvAcct);
if (lblType.Text == "CPV")
    CPVBLL.UnLocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration,
pnldgvAcct);
}

    GobalDeclarations.MenuItemsEdit(btnNew, btnSave, btnEdit, btnUpdate, btnDelete,
btnCancel, 1);
}

private void btnUpdate_Click(object sender, EventArgs e)
{
if (lblType.Text == "BRV")
{
if (BRVBLL.FieldChecks(cmbBankName.Text, dgvAcct) == "true")
{
    string isSuccess = BRVDAL.Update(txtDocNo.Text,
Convert.ToString(dtDocDate.Value.ToShortDateString()), lblType.Text,
cmbBankName.SelectedValue.ToString(), txtBankNarration.Text);
    if (isSuccess == "true")
    {
        for (int i = 0; i < dgvAcct.Rows.Count; i++)
        {
            isSuccess = BRVDAL.AddDetailGrid(txtDocNo.Text, lstType.Text,
Convert.ToString(dtDocDate.Value.ToShortDateString()),
            if (isSuccess == "true")
            {
                isSuccess = BRVDAL.AddDetailSngRow(txtDocNo.Text, lstType.Text,
Convert.ToString(dtDocDate.Value.ToShortDateString()), cmbBankName.SelectedValue, txtBankNarration.Text, GridColumnSum());
                if (isSuccess == "true")
                {
                    MessageBox.Show("Record successfully Updated");
LoadRecords();
BRVBLL.UnLocksControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo,
txtBranchCode, txtBranchName, txtBankNarration, pnldgvAcct);
GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit,
btnUpdate, btnDelete, btnCancel, 1);
                }
                else
                MessageBox.Show(isSuccess.ToString());
            }
            else
            MessageBox.Show(isSuccess.ToString());
        }
        else
        MessageBox.Show(isSuccess.ToString());
    }
}
else
    MessageBox.Show(isSuccess.ToString());
}
else
    MessageBox.Show(isSuccess.ToString());
}
if (lblType.Text == "BPV")
{
if (BPVBLL.FieldChecks(cmbBankName.Text, dgvAcct) == "true")
{
    string isSuccess = BPVDAL.Update(txtDocNo.Text, 
            Convert.ToString(dtDocDate.Value.ToShortDateString()), lblType.Text, 
            cmbBankName.SelectedValue.ToString(), txtBankNarration.Text); 
    if (isSuccess == "true")
    {
        for (int i = 0; i < dgvAcct.Rows.Count; i++)
            isSuccess = BPVDAL.AddDetailGrid(txtDocNo.Text, lblType.Text, 
        if (isSuccess == "true")
        {
            isSuccess = BPVDAL.AddDetailSngRow(txtDocNo.Text, lblType.Text, 
                Convert.ToString(dtDocDate.Value.ToShortDateString()), cmbBankName.SelectedValue.ToString(), 
                txtBankNarration.Text, GridColumnSum());
            if (isSuccess == "true")
            {
                MessageBox.Show("Record successfully Updated");
                LoadRecords();
                BPVBLL.LocksControls(txtDocNo.Text, lblType.Text, 
                    txtBranchCode, txtBranchName, txtBankNarration, pnldgvAcct);
                GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, 
                        btnUpdate, btnDelete, btnCancel, 1);
            }
            else
                MessageBox.Show(isSuccess.ToString());
        }
        else
            MessageBox.Show(isSuccess.ToString());
    }
    else
        MessageBox.Show(isSuccess.ToString());
}
if (lblType.Text == "CRV")
{
    if (CRVBLL.FieldChecks(cmbCashName.Text, dgvAcct) == "true")
    {
        string isSuccess = CRVDAL.Update(txtDocNo.Text, 
            Convert.ToString(dtDocDate.Value.ToShortDateString()), lblType.Text, 
            cmbCashName.SelectedValue.ToString(), txtBankNarration.Text); 
            if (isSuccess == "true")
        {
            isSuccess = CRVDAL.AddDetailGrid(txtDocNo.Text, lblType.Text, 
        if (isSuccess == "true")
        {
            isSuccess = CRVDAL.AddDetailSngRow(txtDocNo.Text, lblType.Text, 
                Convert.ToString(dtDocDate.Value.ToShortDateString()), cmbCashName.SelectedValue.ToString(),
                txtBankNarration, pnldgvAcct);
            GobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, 
                        btnUpdate, btnDelete, btnCancel, 1);
        }
        else
            MessageBox.Show(isSuccess.ToString());
    }
    else
        MessageBox.Show(isSuccess.ToString());
}
else
    MessageBox.Show(isSuccess.ToString());
}

if (lblType.Text == "CPV")
{
    if (CPVBLL.FieldChecks(cmbCashName.Text, dgvAcct) == "true")
    {
        string isSuccess = CPVDAL.Update(txtDocNo.Text, Convert.ToString(dtDocDate.Value.ToShortDateString()), lblType.Text, cmbCashName.SelectedValue.ToString(), dgvAcct.Rows.Count(), txtBankNarration.Text);
        if (isSuccess == "true")
        {
            for (int i = 0; i < dgvAcct.Rows.Count; i++)
            {
                if (isSuccess == "true")
                {
                    MessageBox.Show("Record successfully Updated");
                    LoadRecords();
                    CPVBLL.LocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, pnlDgvAcct);
                    GlobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
                }
            }
        }
        else
            MessageBox.Show(isSuccess.ToString());
    }
    else
            MessageBox.Show(isSuccess.ToString());
}

private void btnDelete_Click(object sender, EventArgs e)
{
    string isSuccess;
    if (lblType.Text == "BRV")
    {
        isSuccess = BRVDAL.Delete(txtDocNo.Text, lblType.Text);
        if (isSuccess == "true")
        {
            MessageBox.Show("Record successfully Deleted");
            LoadRecords();
        BRVDAL.LocksControls(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
        BRVDAL.Previous(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, pnlDgvAcct);
        GlobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, 1);
        }
    }
    else
            MessageBox.Show(isSuccess.ToString());
}
if (lblType.Text == "BPV")
{
    isSuccess = BPVDAL.Delete(txtDocNo.Text, lblType.Text);
    if (isSuccess == "true")
    {    }
```csharp
    MessageBox.Show("Record successfully Deleted");
    LoadRecords();
    BPVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
    BPVBLL.LocksControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, pnlDgvAcct);
    GlobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, i);
}
else
    MessageBox.Show(isSuccess.ToString());
if (lblType.Text == "CRV")
{
    isSuccess = CRVDAL.Delete(txtDocNo.Text, lblType.Text);
    if (isSuccess == "true")
    {
        MessageBox.Show("Record successfully Deleted");
        LoadRecords();
        CRVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
        CRVBLL.LocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, pnlDgvAcct);
        GlobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, i);
    }
else
    MessageBox.Show(isSuccess.ToString());
}
else
    MessageBox.Show(isSuccess.ToString());
if (lblType.Text == "CPV")
{
    isSuccess = CPVDAL.Delete(txtDocNo.Text, lblType.Text);
    if (isSuccess == "true")
    {
        MessageBox.Show("Record successfully Deleted");
        LoadRecords();
        CPVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
        CPVBLL.LocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, pnlDgvAcct);
        GlobalDeclarations.MenuItemsSavUpdDel(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, i);
    }
else
    MessageBox.Show(isSuccess.ToString());
}

private void btnCancel_Click(object sender, EventArgs e)
{
    if (lblType.Text == "BRV")
    {
        BRVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
        BRVBLL.LocksControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, pnlDgvAcct);
    }
else
    MessageBox.Show(isSuccess.ToString());
if (lblType.Text == "BPV")
{
    BPVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
    BPVBLL.LocksControls(txtDocNo, dtDocDate, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, pnlDgvAcct);
}
else
    MessageBox.Show(isSuccess.ToString());
if (lblType.Text == "CRV")
{
    CRVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
}
CRVBLL.LocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, pnldgvAcct);
}  
if (lblType.Text == "CPV")
{
    CPVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
    CPVBLL.LocksControls(txtDocNo, dtDocDate, cmbCashName, txtBankNarration, pnldgvAcct);
}
GlobalDeclarations.MenuItemsDefault(btnNew, btnSave, btnEdit, btnUpdate, btnDelete, btnCancel, l);
}

private void btnFirst_Click(object sender, EventArgs e)
{
if (lblType.Text == "BRV")
    BRVBLL.First(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "BPV")
    BPVBLL.First(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "CRV")
    CRVBLL.First(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "CPU")
    CPVBLL.First(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
}

private void btnPrevious_Click(object sender, EventArgs e)
{
if (lblType.Text == "BRV")
    BRVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "BPV")
    BPVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "CRV")
    CRVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "CPU")
    CPVBLL.Previous(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
}

private void btnNext_Click(object sender, EventArgs e)
{
if (lblType.Text == "BRV")
    BRVBLL.Next(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "BPV")
    BPVBLL.Next(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "CRV")
    CRVBLL.Next(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "CPU")
    CPVBLL.Next(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
}

private void btnLast_Click(object sender, EventArgs e)
{
if (lblType.Text == "BRV")
    BRVBLL.Last(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "BPV")
    BPVBLL.Last(txtDocNo, dtDocDate, lblType, cmbBankName, txtAcctNo, txtBranchCode, txtBranchName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "CRV")
    CRVBLL.Last(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
if (lblType.Text == "CPU")
    CPVBLL.Last(txtDocNo, dtDocDate, lblType, cmbCashName, txtBankNarration, dgvAcct, dt);
private void cmbBankName_SelectedIndexChanged(object sender, EventArgs e)
{
    DataTable dtBank = null;
    try
    {
        if (lblType.Text == "BRV")
            dtBank = BRVDAL.FetchAccounts(cmbBankName.Text);
        if (lblType.Text == "BPV")
            dtBank = BPVDAL.FetchAccounts(cmbBankName.Text);
        txtAcctNo.Text = dtBank.Rows[0][0].ToString();
        txtBranchCode.Text = dtBank.Rows[0][1].ToString();
        txtBranchName.Text = dtBank.Rows[0][2].ToString();
    }
    catch (Exception ex)
    {
    }
}
private void btnAdd_Click(object sender, EventArgs e)
{
    dgvAcct.DataSource = null;
    dgvAcct.Rows.Add(cmbAcctCode.Text, txtAcctTitle.Text, txtChqNo.Text, dtChqDate.Value.ToShortDateString(), txtNarration.Text, txtAmount.Text);
    if (lblType.Text == "BRV")
        BRVBLL.ClearPanelControls(cmbAcctCode, txtAcctTitle, txtChqNo, dtChqDate, txtNarration, txtAmount);
    if (lblType.Text == "BPV")
        BPVBLL.ClearPanelControls(cmbAcctCode, txtAcctTitle, txtChqNo, dtChqDate, txtNarration, txtAmount);
    if (lblType.Text == "CRV")
        CRVBLL.ClearPanelControls(cmbAcctCode, txtAcctTitle, txtChqNo, dtChqDate, txtNarration, txtAmount);
    if (lblType.Text == "CPV")
        CPVBLL.ClearPanelControls(cmbAcctCode, txtAcctTitle, txtChqNo, dtChqDate, txtNarration, txtAmount);
}
Appendix B. NetBeans IDE 6.0 Implementation

B.1. DBConnection.java

```java
public class DBConnection {
    Connection conn = null;

    public DBConnection() { // Constructor
        try {
            String conStr = "jdbc:sqlserver://SYSTEM;databaseName=CnB";
            Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
            conn = DriverManager.getConnection(conStr, "sa", "");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        return conn;
    }

closeConnection() {
    try {
        if(conn!=null)
            conn.close();
        conn=null;
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

B.2. AccountCodesDAL.java

public class AccountCodesDAL {
    Generics_Functions gf = new Generics_Functions();
    public String Add(String AccCode, String AccName)
    {
        String query, isSuccess = "";
        ResultSet dt;
        try {
            query = "INSERT INTO Accounts (AccCode, AccName) VALUES ('" + AccCode + ", " + AccName + ")";
            isSuccess = gf.ExecuteQuery(query);
        }
        catch (Exception e) {
            System.err.println("Got an exception! ");
            System.err.println(e.getMessage);
            System.exit(1);
        }
        return isSuccess;
    }
}
```
try {
    String query = "UPDATE Accounts SET AccName='" + AccName + '" WHERE AccCode='" + AccCode + '"';
    isSuccess = gf.ExecuteQuery(query);
} catch (Exception e) {
    System.err.println("Got an exception! ");
    System.err.println(e.getMessage());
    System.exit(1);
} return isSuccess;

public String Delete(String AccCode) {
    String isSuccess="";
    try {
        String query = "DELETE FROM Accounts WHERE AccCode='" + AccCode + '"';
        isSuccess = gf.ExecuteQuery(query);
    } catch (Exception e) {
        System.err.println("Got an exception! ");
        System.err.println(e.getMessage());
        System.exit(1);
    } return isSuccess;
}

public ResultSet Display() {
    String query = "SELECT * FROM Accounts";
    ResultSet dt = gf.LoadDataTableMemory(query);
    return dt;
}

B.3. AccountSearch.java
public class AccountSearch {
Generics_Functions gf = new Generics_Functions();
public ResultSet FindRecord() {
    String query = "";
    ResultSet dt = gf.LoadDataTableMemory(query);
    return dt;
}

public ResultSet Display() {
    String query = "";
    ResultSet dt = gf.LoadDataTableMemory(query);
    return dt;
}

B.4. CnBCodesDAL.java
public class CnBCodesDAL {
Generics_Functions gf=new Generics_Functions();
public String GenerateCode()
{
    String query = "SELECT ISNULL(MAX(CnBCode),0)00000') CnBCode FROM CnBCode";
    String res = gf.GenerateCode(query);
    return res;
}

public String Add(String CnBCode, String AcctType, String AcctCode, String BName, String BAccountNo, String BrnchCod, String BrnchName, String BankAcctType, String Address, String City)
{
    String query = "INSERT INTO CnBCode (CnBCode, AcctType, AcctCode, BName, BAccountNo, BrnchCod,
    BrnchName, BankAcctType, Address, City) VALUES (" + CnBCode + "," + AcctType + "," + AcctCode + "," + BName + "," + BAccountNo + "," + BrnchCod + "," + BrnchName + "," + BankAcctType + "," + Address + "," + City + ");
    String isSuccess = gf.ExecuteQuery(query);
    return isSuccess;
}

public String Update(String CnBCode, String AcctType, String AcctCode, String BName, String BAccountNo, String BrnchCod, String BrnchName, String BankAcctType, String Address, String City)
{
    String query = "UPDATE CnBCode SET AcctType='" + AcctType + ", AcctCode='" + AcctCode + ", BName='" + BName + ", BAccountNo='" + BAccountNo + ", BrnchCod='" + BrnchCod + ", BrnchName='" + BrnchName + ",
    BankAcctType='" + BankAcctType + ", Address='" + Address + ", City='" + City + ", WHERE CnBCode='" + CnBCode + ";
    String isSuccess = gf.ExecuteQuery(query);
    return isSuccess;
}

public String Delete(String CnBCode)
{
    String query = "DELETE FROM CnBCode WHERE CnBCode='" + CnBCode + ";
    String isSuccess = gf.ExecuteQuery(query);
    return isSuccess;
}

public ResultSet Display()
{
    String query = "SELECT CnBCode, AcctType, AcctCode, BName, BAccountNo, BrnchCod, BrnchName,
    BankAcctType, Address, City FROM CnBCode";
    ResultSet dt = gf.LoadResultSetMemory(query);
    return dt;
}

public ResultSet LoadAcctInfo(String AccCode)
{
    String query = "SELECT AccName FROM WHERE AccCode='" + AccCode + ";
    ResultSet res = gf.FetchAcctDescription(query);
    return res;
}

public ResultSet Search(String CnBCode)
{
    String query = "SELECT CnBCode, AcctType, AcctCode, BName, BAccountNo, BrnchCod, BrnchName,
    BankAcctType, Address, City FROM CnBCode WHERE CnBCode='" + CnBCode + ";
    ResultSet dt = gf.LoadResultSetMemory(query);
    return dt;
}

B.5. Generic_Functions.java

public class Generics_Functions {

    public ResultSet FetchAcctDescription(String QueryString)
    {
        ResultSet rs = null;
        try
        {  
            // Code for fetching account description
        }
        catch (Exception e)
        {
            // Handle exception
        }
    }

}
public ResultSet LoadResultSetMemory(String QueryString)
{
    ResultSet rs = null;
    try
    {
        DBConnection ObjDBConn = new DBConnection();
        Connection conn = ObjDBConn.DBConnection();
        if (conn != null)
        {
            java.sql.Statement st = conn.createStatement();
            rs = st.executeQuery(QueryString);
        }
    }
    catch (Exception e)
    {
        System.err.println("Got an exception! ");
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return rs;
}

public String GenerateCode(String Query)
{
    ResultSet rs = null;
    String newCode = null;
    try
    {
        DBConnection ObjDBConn = new DBConnection();
        Connection conn = ObjDBConn.DBConnection();
        if (conn != null)
        {
            java.sql.Statement st = conn.createStatement();
            rs = st.executeQuery(Query);
            while (rs.next())
            {
                newCode = rs.getString(1);
            }
        }
    }
    catch (Exception e)
    {
        System.err.println("Got an exception! ");
        System.err.println(e.getMessage());
    }
    return newCode;
}
System.exit(1);
}
return newCode;
}
public ResultSet LoadDataTableMemory(String Query)
{
    ResultSet rs = null;
    try
    {
        DBConnection ObjDBConn = new DBConnection();
        Connection conn = ObjDBConn.DBConnection();
        if (conn != null)
        {
            java.sql.Statement st = conn.createStatement();
            rs = st.executeQuery(Query);
        }
    }
    catch (Exception e)
    {
        System.err.println("Got an exception! ");
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return rs;
}

public String ExecuteQuery(String QueryString)
{
    try
    {
        DBConnection ObjDBConn = new DBConnection();
        Connection conn = ObjDBConn.DBConnection();
        if (conn != null)
        {
            java.sql.Statement st = conn.createStatement();
            st.executeUpdate(QueryString);
            conn.close();
        }
    }
    catch (Exception e)
    {
        System.err.println("Got an exception! ");
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "true";
}

public void FillDropDown(JComboBox ComboBox, String DisplayMember, String QueryString)
{
    try
    {
        DBConnection ObjDBConn = new DBConnection();
        Connection conn = ObjDBConn.DBConnection();
        if (conn != null)
        {
            ResultSet rs = st.executeQuery(QueryString);
            ComboBox.removeAllItems();
            rs.next();
            ComboBox.addItem(rs.getString(1));
        }
    }
    catch (Exception e)
    {
        System.err.println("Got an exception! ");
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
while(rs.next())
    ComboBox.addItem(rs.getString(1));
}
}catch (Exception e)
{
    System.err.println("Got an exception! ");
    System.err.println(e.getMessage());
    System.exit(1);
}

B.6. BRVDAL.java
public class BRVDAL {
  private Generics_Functions gf = new Generics_Functions();
  public String GenerateCode()
  {
    String query = "SELECT ISNULL(MAX(CnBCode),'00000') CnBCode FROM CnBCode";
    String res = gf.GenerateCode(query);
    return res;
  }
  public void LoadBanks(JComboBox comboBox)
  {
    String query = "SELECT AcctCode, BName from CnBCode";
    gf.FillDropDown(comboBox, "BName", query);
  }
  public ResultSet FetchAccounts(String BName)
  {
    String query = "SELECT BName, AcctCode, BAccountNo, BrnchCod, BrnchName from CnBCode WHERE BName = " + BName + ";"
    ResultSet dt = gf.LoadDataTableMemory(query);
    return dt;
  }
  public void LoadTransAccts(JComboBox comboBox)
  {
    String query = "SELECT AccCode, AccName from Accounts";
    gf.FillDropDown(comboBox, "AccCode", query);
  }
  public String Add(String vNo, String vDate, String vType, String Bank, String TaxAcct, String TaxAmount, String BankNarration, String TaxNarration, int count, String AcctCode, String CheqRefNo, String CheqRefDate, String Narration, double Dabit, String AcctGLCode)
  {
    double Credit = 0; int i;
    String query = "INSERT INTO vMain (vNo, vDate, vType, Bank, TaxAcct, TaxAmount, BankNarration, TaxNarration) VALUES (" + vNo + "," + i + 1 + "," + vType + "," + vDate + "," + Bank + "," + TaxAcct + "," + TaxAmount + "," + BankNarration + "," + TaxNarration + ");"
    String isSuccess = gf.ExecuteQuery(query);
    if (isSuccess == "true")
    {
      for (i = 0; i <= count; i++)
      {
        query = "INSERT INTO vDetail (vNo, sNo, vType, vDate, AcctCode, CheqRefNo, CheqRefDate, Narration, Dabit) VALUES (" + vNo + "," + i + 1 + "," + vType + "," + vDate + "," + AcctCode + "," + CheqRefNo + "," + CheqRefDate + "," + Narration + "," + Dabit + ");"
        isSuccess = gf.ExecuteQuery(query);
        Credit = ((int) Credit) + ((int) Dabit);
      }
    }
  }
}

83
isSuccess = gf.ExecuteQuery(query);
}
return isSuccess;

public String Update(String vNo, String vDate, String vType, String Bank, String TaxAcct, String TaxAmount, String BankNarration, String TaxNarration, int count, String AcctCode, String CheqRefNo, String CheqRefDate, String Narration, double Dabit, String AcctGLCode)
{
    double Credit = 0; int i;
    String query = "UPDATE vMain SET vDate='" + vDate + ", vType='" + vType + ", Bank='" + Bank + ", TaxAcct='" + TaxAcct + ", TaxAmount='" + TaxAmount + ", BankNarration='" + BankNarration + ", TaxNarration='" + TaxNarration + " WHERE vNo='" + vNo + "';"
    String isSuccess = gf.ExecuteQuery(query);
    if (isSuccess == "true")
    {
        query = "DELETE FROM vMain WHERE vNo='" + vNo + ";"
        isSuccess = gf.ExecuteQuery(query);
        for (i = 0; i <= count; i++)
        {
            query = "INSERT INTO vDetail (vNo, sNo, vType, vDate, AcctCode, CheqRefNo, CheqRefDate, Narration, Dabit) VALUES ('" + vNo + ", " + i + 1 + ", vType='" + vType + ", vDate='" + vDate + ", AcctCode='" + AcctCode + ", CheqRefNo='" + CheqRefNo + ", " + CheqRefDate + ", " + Narration + ", " + Credit + ");"
            isSuccess = gf.ExecuteQuery(query);
            Credit = ((int) Credit) + ((int) Dabit);
        }
        query = "INSERT INTO vDetail (vNo, sNo, vType, vDate, AcctCode, CheqRefNo, CheqRefDate, Narration, Credit) VALUES ('" + vNo + ", " + i + 1 + ", vType='" + vType + ", vDate='" + vDate + ", AcctCode='" + AcctCode + ", CheqRefNo='" + CheqRefNo + ", " + CheqRefDate + ", " + Narration + ", " + Credit + ");"
        isSuccess = gf.ExecuteQuery(query);
    }
    return isSuccess;
}

public String Delete(String vNo)
{
    String query = "DELETE FROM vDetail WHERE vNo='" + vNo + "';"
    String isSuccess = gf.ExecuteQuery(query);
    if (isSuccess == "true")
    {
        query = "DELETE FROM vMain WHERE vNo='" + vNo + "';"
        isSuccess = gf.ExecuteQuery(query);
    }
    return isSuccess;
}

public ResultSet Display()
{
    String query = "SELECT M.vNo, M.vDate, M.vType, M.Bank, M.TaxAcct, M.TaxAmount, M.BankNarration, M.TaxNarration, D.sNo, D.AcctCode, D.CheqRefNo, D.CheqRefDate, D.Narration, D.Dabit FROM vMain M INNER JOIN vDETAIL D ON M.vNo = D.vNo";
    ResultSet dt = gf.LoadDataTableMemory(query);
    return dt;
}

public ResultSet Search(String vNo)
{
    String query = "SELECT M.vNo, M.vDate, M.vType, M.Bank, M.TaxAcct, M.TaxAmount, M.BankNarration, M.TaxNarration, D.sNo, D.AcctCode, D.CheqRefNo, D.CheqRefDate, D.Narration, D.Dabit FROM vMain M INNER JOIN vDETAIL D ON M.vNo = D.vNo WHERE M.vNo='" + vNo + "';"
    ResultSet dt = gf.LoadDataTableMemory(query);
    return dt;
}
public void Print()
{
}
}

B.7. BPVDAL.java

public class BPVDAL {
    Generics_Functions gf = new Generics_Functions();
    public String GenerateCode()
    {
        String query = "SELECT ISNULL(MAX(CnBCode),’00000’) CnBCode FROM CnBCode";
        String res = gf.GenerateCode(query);
        return res;
    }
    public void LoadBanks(JComboBox comboBox)
    {
        String query = "SELECT AcctCode, BName from CnBCode";
        gf.FillDropDown(comboBox, "BName", query);
    }
    public ResultSet FetchAccounts(String BName)
    {
        String query = "SELECT BName, AcctCode, BAccountNo, BrnchCod, BrnchName from CnBCode WHERE BName = ‘" + BName + ‘’’;
        ResultSet dt = gf.LoadDataTableMemory(query);
        return dt;
    }
    public void LoadTransAccts(JComboBox comboBox)
    {
        String query = "SELECT AccCode, AccName from Accounts";
        gf.FillDropDown(comboBox, "AccCode", query);
    }
    public String Add(String vNo, String vDate, String vType, String Bank, String TaxAcct, String TaxAmount, String BankNarration, String TaxNarration, int count, String AcctCode, String CheqRefNo, String CheqRefDate, StringNarration, double Dabit, String AcctGLCode)
    {
        double Credit = 0; int i;
        String query = "INSERT INTO vMain (vNo, vDate, vType, Bank, TaxAcct, TaxAmount, BankNarration, TaxNarration) VALUES (" + vNo + "," + vDate + "," + vType + "," + Bank + "," + TaxAcct + "," + TaxAmount + "," + BankNarration + "," + TaxNarration + ");"
        String isSuccess = gf.ExecuteQuery(query);
        if (isSuccess == "true")
        {
            for (i = 0; i <= count; i++)
            {
                query = "INSERT INTO vDetail (vNo, sNo, vType, vDate, AcctCode, CheqRefNo, CheqRefDate, Narration, Dabit) VALUES (" + vNo + "," + i + "," + vType + "," + vDate + "," + AcctCode + "," + CheqRefNo + "," + CheqRefDate + "," + Narration + "," + Dabit + ");"
                isSuccess = gf.ExecuteQuery(query);
                Credit = ((int) Credit) + ((int) Dabit);
            }
        }
        String query = "INSERT INTO vDetail (vNo, sNo, vType, vDate, AcctCode, CheqRefNo, CheqRefDate, Narration, Credit) VALUES (" + vNo + "," + i + "," + vType + "," + vDate + "," + AcctCode + "," + CheqRefNo + "," + CheqRefDate + "," + Narration + "," + Credit + ");"
        isSuccess = gf.ExecuteQuery(query);
        return isSuccess;
    }
}
public String Update(String vNo, String vDate, String vType, String Bank, String TaxAcct, String TaxAmount, String BankNarration, String TaxNarration, int count, String AcctCode, String CheqRefNo, String CheqRefDate, String Narration, double Dabit, String AcctGLCode) {
    double Credit = 0; int i;
    String query = "UPDATE vMain SET vDate='" + vDate + "', vType='" + vType + "', Bank='" + Bank + "', TaxAcct='" + TaxAcct + "', TaxAmount='" + TaxAmount + "', BankNarration='" + BankNarration + "', TaxNarration='" + TaxNarration + "'
WHERE vNo='" + vNo + "';"
    String isSuccess = gf.ExecuteQuery(query);
    if (isSuccess == "true") {
        query = "DELETE FROM vMain WHERE vNo='" + vNo + "';"
        isSuccess = gf.ExecuteQuery(query);
        for (i = 0; i <= count; i++) {
            query = "INSERT INTO vDetail (vNo, sNo, vType, vDate, AcctCode, CheqRefNo, CheqRefDate, Narration, Dabit) VALUES ('" + vNo + ", '" + i + 1 + "', '" + vType + "', '" + vDate + "', '" + AcctCode + ", '" + CheqRefNo + "', '" + CheqRefDate + "', '" + Narration + ", '" + Dabit + "');"
            isSuccess = gf.ExecuteQuery(query);
            Credit = ((int) Credit) + ((int) Dabit);
        }
        query = "INSERT INTO vDetail (vNo, sNo, vType, vDate, AcctCode, CheqRefNo, CheqRefDate, Narration, Credit) VALUES ('" + vNo + "', '" + 1 + 1 + ", '" + vType + "', '" + vDate + "', '" + AcctGLCode + ", '" + CheqRefNo + "', '" + CheqRefDate + "', '" + Narration + "', '" + Credit + "');"
        isSuccess = gf.ExecuteQuery(query);
    }
    return isSuccess;
}

public String Delete(String vNo) {
    String query = "DELETE FROM vDetail WHERE vNo='" + vNo + ";"
    String isSuccess = gf.ExecuteQuery(query);
    if (isSuccess == "true") {
        query = "DELETE FROM vDetail WHERE vNo='" + vNo + ";"
        isSuccess = gf.ExecuteQuery(query);
    }
    return isSuccess;
}

public ResultSet Display() {
    String query = "SELECT M.vNo, M.vDate, M.vType, M.Bank, M.TaxAcct, M.TaxAmount, M.BankNarration, M.TaxNarration, D.sNo, D.AcctCode, D.CheqRefNo, D.CheqRefDate, D.Narration, D.Dabit FROM vMain M INNER JOIN vDetail D ON M.vNo = D.vNo";
    ResultSet dt = gf.LoadDataTableMemory(query);
    return dt;
}

public ResultSet Search(String vNo) {
    String query = "SELECT M.vNo, M.vDate, M.vType, M.Bank, M.TaxAcct, M.TaxAmount, M.BankNarration, M.TaxNarration, D.sNo, D.AcctCode, D.CheqRefNo, D.CheqRefDate, D.Narration, D.Dabit FROM vMain M INNER JOIN vDetail D ON M.vNo = D.vNo WHERE M.vNo '=' + vNo + ";"
    ResultSet dt = gf.LoadDataTableMemory(query);
    return dt;
}