

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:	Fonseka	Student ID:	20546924
Other name(s):	Gampalawaduge Anne Shivoli		
Unit name:	Operating System	Unit ID:	COMP3008
Lecturer / unit coordinator:	Dr. Sajib Mistry	Tutor:	Ms. Dilani Lunugalage
Date of submission:	24/05/2021	Which assignment?	(Leave blank if the unit has only one assignment.)

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: G. Anne S. Fonseka Date of signature: 24/05/2021

(By submitting this form, you indicate that you agree with all the above text.)

Bank Application Document

CONTENT

01. Summary of three tier Architecture.....	03
02. How to run the Bank Application.....	04
03. Analysis of the Bank Application.....	04
04. The Objects and functions of the Bank Application.....	05
05. The output of the Bank Application.....	07
06. Design Pattern.....	09
07. UML Design Description.....	10
08. UML diagram.....	10
09. Difficulties.....	14
10. Suggestions and further improvements.....	14
11. References.....	14

01. Summary of three tier Architecture.

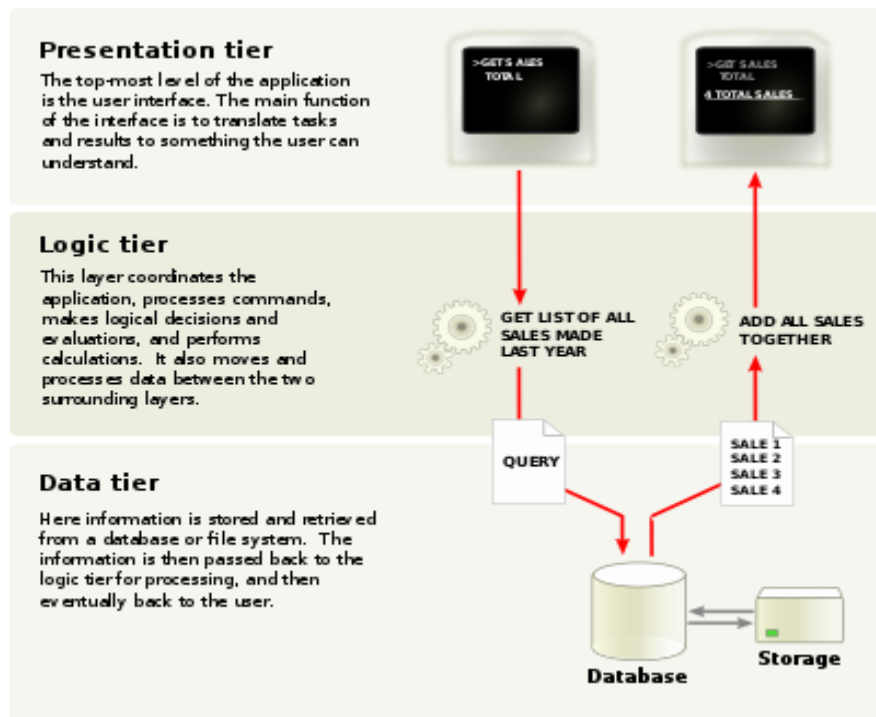


Figure 01: Three tier architecture concept [1].

i. The Data Tier

This tier will mainly consist of the Bank Database objects related to the Bank Application and security provided to DLL to prevent from unauthorized access, because hackers will be able to steal/hack money and confidential data and information.

BankDB Object is created to communicate with the BankDB system itself and this will create three interfaces, namely; User Interface, Account interface and Transaction interface and these interfaces will communicate among data tier and BankDB interface.

By selecting respective interface, user will be able to create Users, Accounts and Transactions. In addition to that, BankDB will provide two major functions, namely; SaveToDisk() and ProcessAllTransactions().

ii. The Business Tier

The business tier is connected to the data tier by using .NET Remoting. The business tier includes the features such as;

- Search transaction per account.

- Checking transactions and withdrawals (whether it has enough money) before doing transactions.

iii. The Presentation Tier

According to the client tier, user will be able to create or select the Banking Application, Some functions are;

- Create and manage account.
- Create and handle account.
- Generate and handle their own user record.

02. How to run the Bank Application

01. Download the folder from the Blackboard.
02. Extract the Zip file.
03. Select the .sln file
04. Open the selected file in visual studio 2019
05. Compile the code

03. Analysis of the Bank Application

With regard to the Bank Application System, I pay my attention to develop the distributed architecture for the Bank Application project. According to given scenario of Banking System, it is needed to be developed by using the layered architecture. Hence, layers are categorized as; Business layer, Data layer and presentation layer.

Defined business component layers are remote services such as;

1. User service
2. Account service
3. Transaction service

User interface

By using the given interface of UserAccessInterfacemethods are categorized as follows; CreateUser() and get the user Reference number.

1. Select the first name and the last name of the registered user.
2. Get the full name of the given registered user.
3. Require all registered users.

Account interface

1. An account number should be assigned to the users
2. All accounts associated the user.
3. Deposit money (funds) of the user.
4. Withdraw money from the user account
5. Check the Account's owner.

Transaction interface

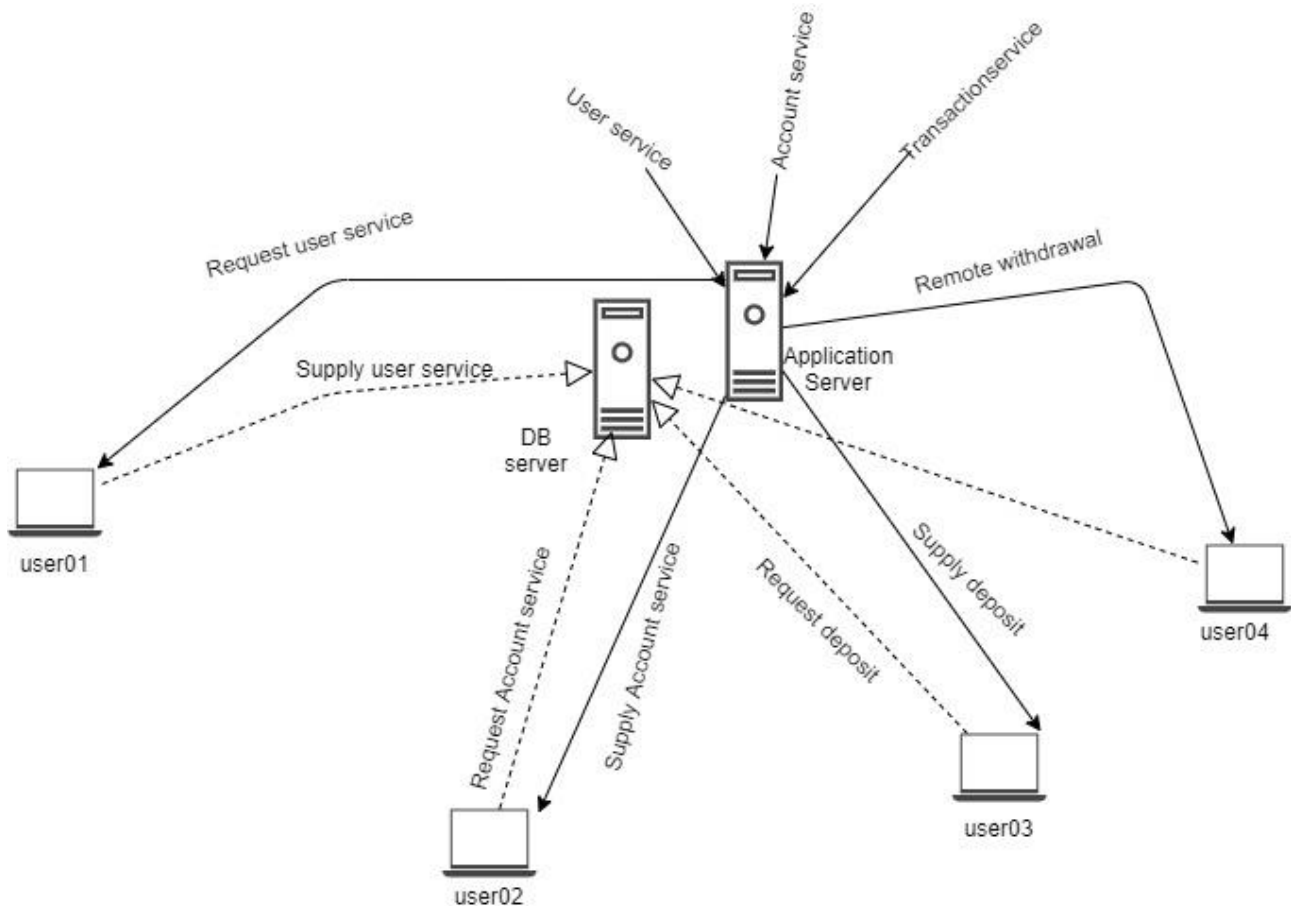
1. Transfer funds/money from one account to another account.
2. Day end transaction processes.
3. Account reconciliation.

Data Layer

Given BankDB.dll support to database functionality of the respective datatable, user, Accounts, and all the transactions.

Presentation Layer

User Interface design and input data validation are checked in the presentation layer.



Services can be deploy in the remote environment and data an be keep separately.

NOTE: Project depicts the local host

04. objects and functions of the Bank Application

- **void SaveToDisk()** – Save the current state of the of the Bank system to a file.
- **void ProcessAllTransactions()** – Processes transactions which are pending.
- **TransactionAccessInterface GetTransactionInterface()** – Allow Transaction access object for access to the Transaction subsystem.

- **UserAccessInterface GetUserAccess()** - Allow User access object for access to the User subsystem.
 - **AccountAccessInterface GetAccountAccess()** - Allow Account access object for access to the Account subsystem.
-
- **Void SelectAccount(uint accountID)** – Select an account
 - **List<unit>GetAccountIDByUser(uint userID)** – List accounts with respect to the user.
 - **Unit CreateAccount(unit userID)** – Create an account and return the ID
 - **Void Deposit(unit amount)** – Deposit money in selected account
 - **Void Withdraw(uint amount)** – Withdraw money from selected account
 - **Unit GetBalance()** – Get Balance from a respective account
 - **Unit GetOwner()** – Get the userID of a respective account
-
- **Void Selectuser(uint userID)** – Select a User
 - **List<unit> GetUsers()** – Get list of registered users
 - **Unit CreateUser()** – Create new user and an ID related to the user
 - **Void GetUserName(out string fname, out string lname)** – Get the name of the selected user
 - **Void SetUserName(string fname, string lname)** – Set the name of the selected user
-
- **Void selectTransaction(uint TransactionID)** – select a transaction
 - **List<unit> GetTransactions()** – Get list of transaction
 - **Unit GetAmount()** – Create transaction and return it's ID
 - **Unit GetSendrAcct()** – Get the ID of the sender of selected transaction
 - **Unit GetRecvrAcct()** –Get the ID of the receiver of selected transaction
 - **Void setAmount(uint amont)** –Set the amount transferred from selected transaction
 - **Void SetSendr()** – Get the ID of the sender of the selected transaction
 - **Void SetRecvr()** – Set the ID of the Receiver of the selected transaction
-
- **CreateAccount()** – Introduce the Bank account to the user
 - **DepositCash()** – Deposit the money by the registered user
 - **WithdrawMoney()** - Withdraw the money by the registered user
 - **DisplayAllAccToUser()** – Display all the account to the registered user
 - **DisplayBalBelongToUser()** – Display Balance belong to the registered user
 - **DisplayAccOwnerGivernAccountID()** – Display Account ID to the owner of the respective Account
 - **Interface()** – Interface consist with collection of several functionalities related to DBSercies
 - **TransferProcess()** – Transferring funds from one account to another. Eg: Standing order
 - **CreateBankUser()** – Create a registered Bank user
 - **GetBankUser ()**– Get the registered bank user
 - **GetAllUsers ()**– Get all the registered bank user

05. The output of the Bank Application

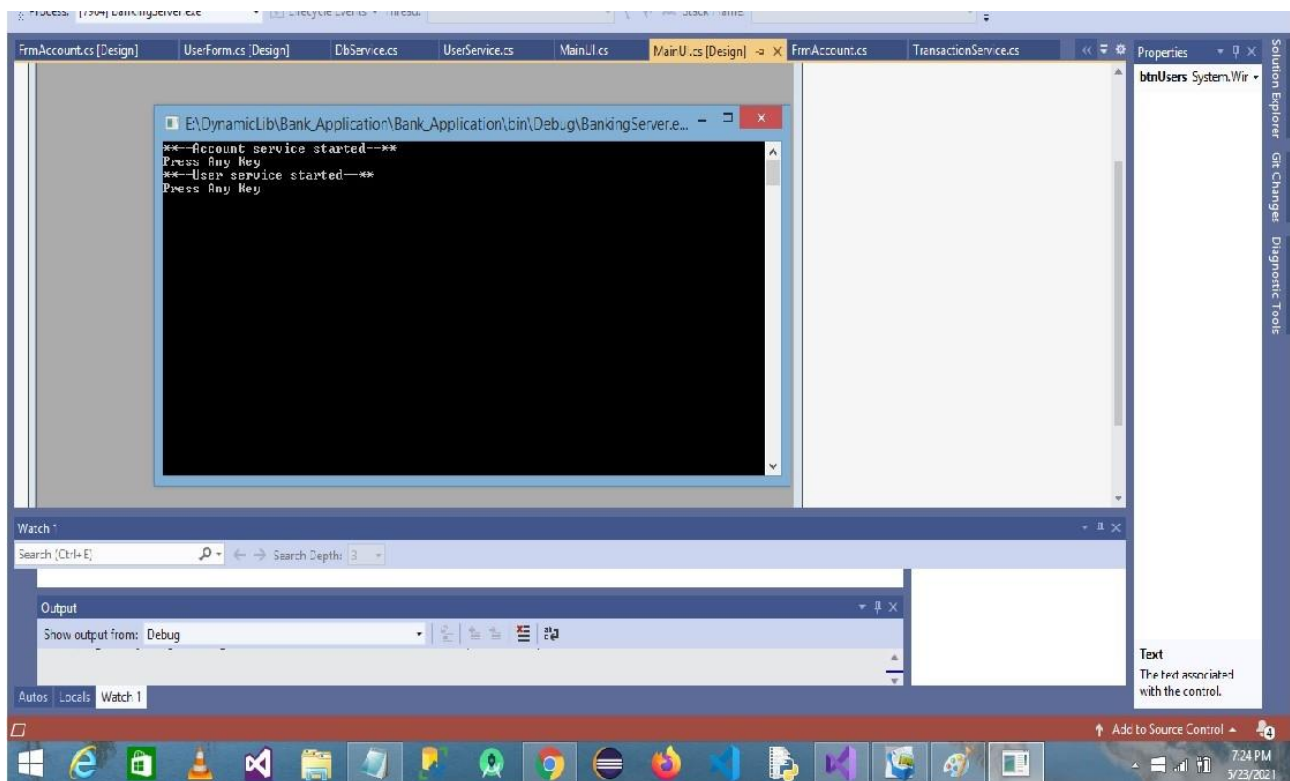


Figure 01: Commence of the Remote services

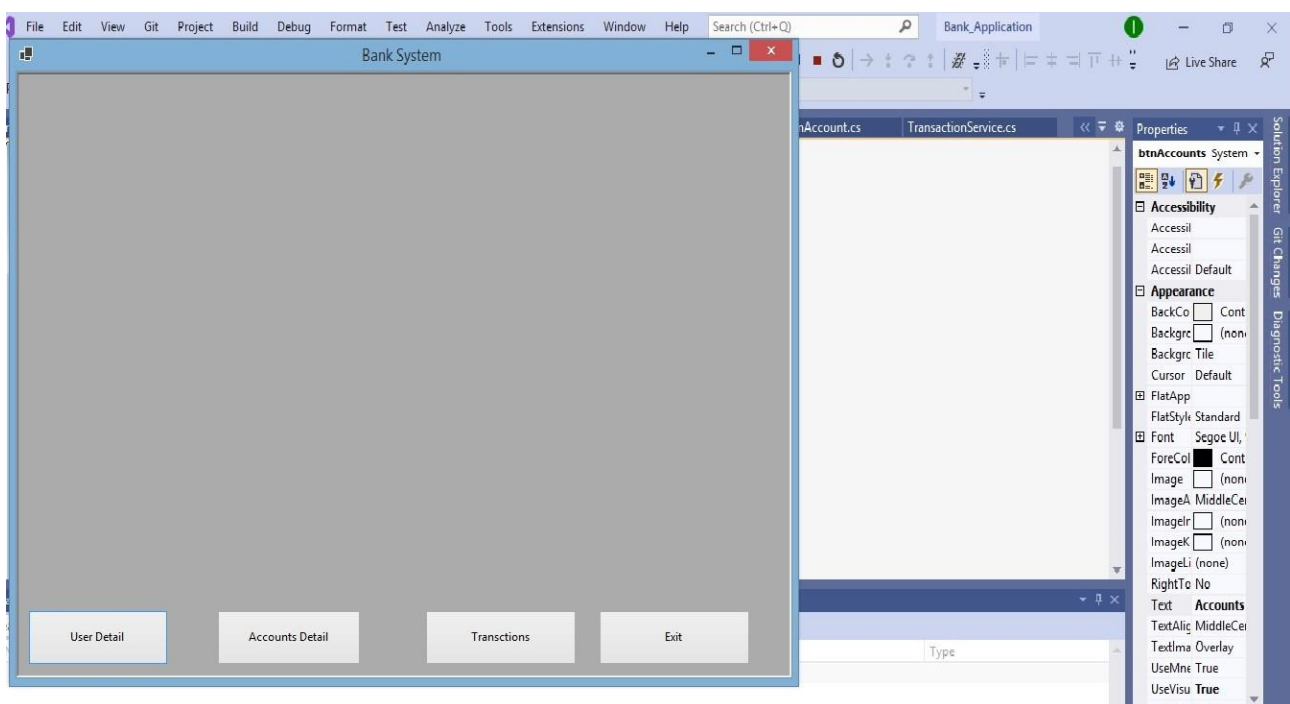
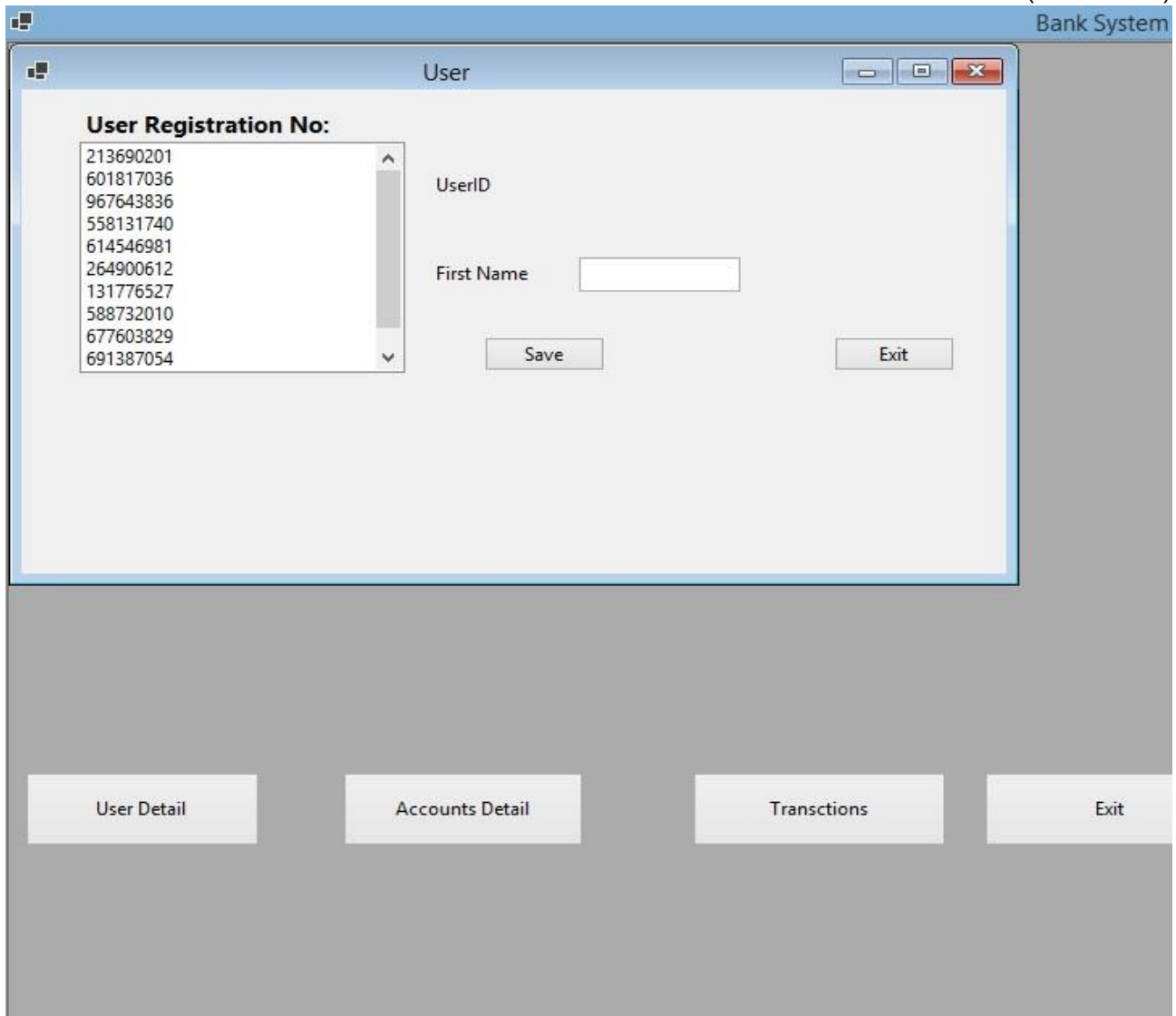


Figure 02: Main interface



The image shows a software interface for a 'Bank System'. At the top, a blue header bar contains the text 'Bank System' on the right. Below this is a window titled 'User' with standard Windows-style controls (minimize, maximize, close). Inside the 'User' window, there is a section titled 'User Registration No:' followed by a list box containing ten numerical IDs: 213690201, 601817036, 967643836, 558131740, 614546981, 264900612, 131776527, 588732010, 677603829, and 691387054. To the right of the list box are two labels: 'UserID' and 'First Name', each followed by an empty text input field. Below these fields are two buttons: 'Save' and 'Exit'. The background of the main application window is gray and features four buttons at the bottom: 'User Detail', 'Accounts Detail', 'Transctions', and 'Exit'.

Bank System

User

User Registration No:

213690201
601817036
967643836
558131740
614546981
264900612
131776527
588732010
677603829
691387054

UserID

First Name

Save

Exit

User Detail

Accounts Detail

Transctions

Exit

Figure 03: User registration interface

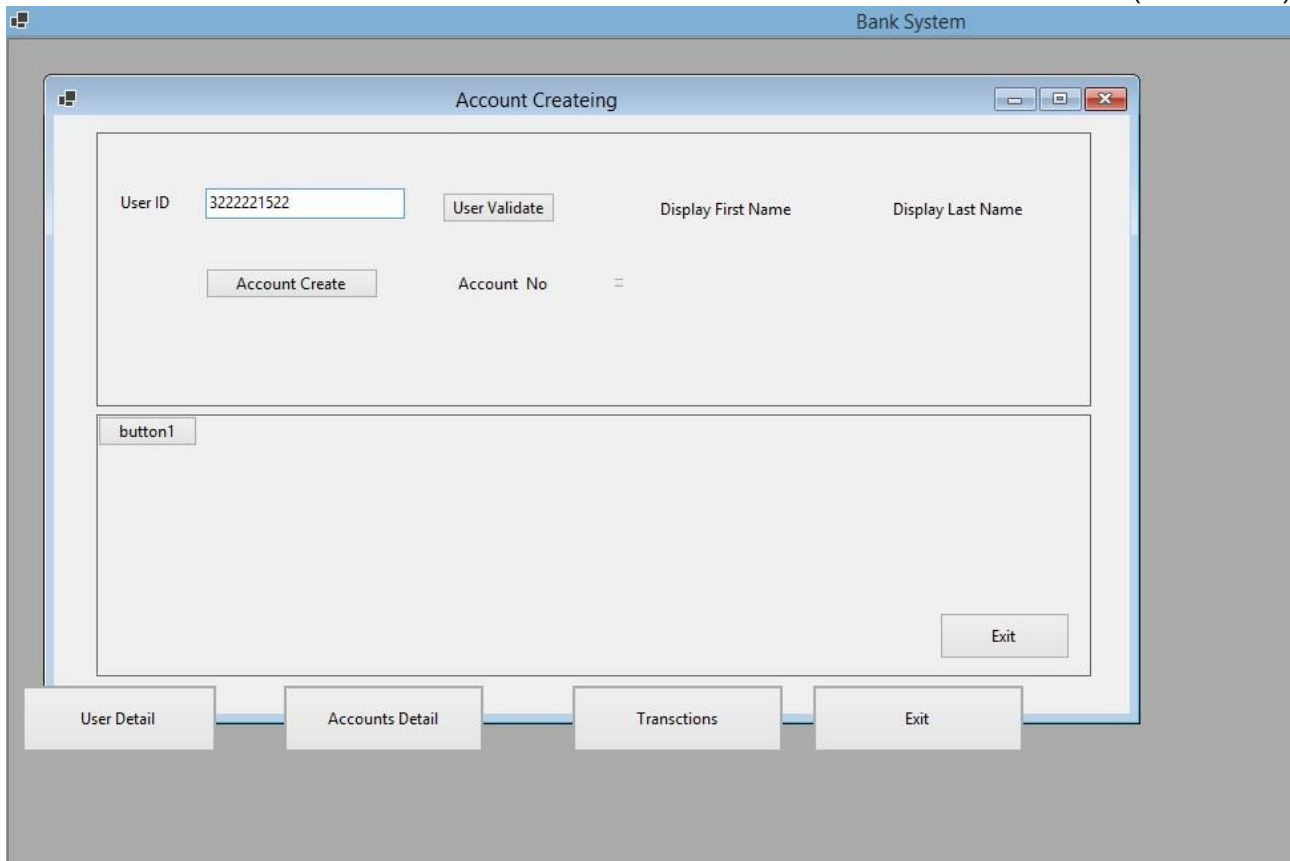


Figure 04: Accounts Registration interface

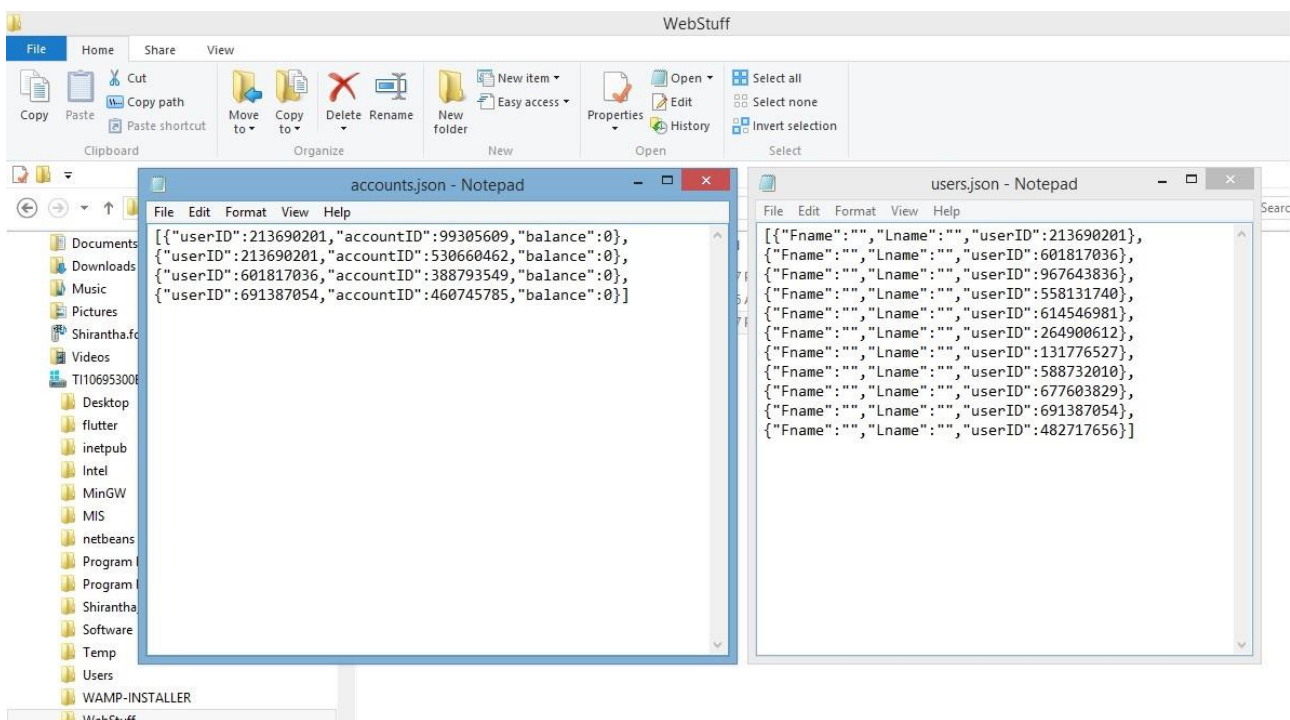


Figure 05: Output of the databases

06. Design Pattern

Connection of the database frequently use all over the project, in order to minimize the redundancy, I have already apply **Singleton Design pattern** technique. Similarly **Façade design pattern** can also

07. UML Design Description

When considering about design outline, there are two major design types [2], namely; structural design diagrams and behavioural design diagrams and Structural diagrams illustrate the technical view of the Bank application, while behavioural diagrams represents the involvement /interaction between the end user and the system.

(Behavioural diagrams)

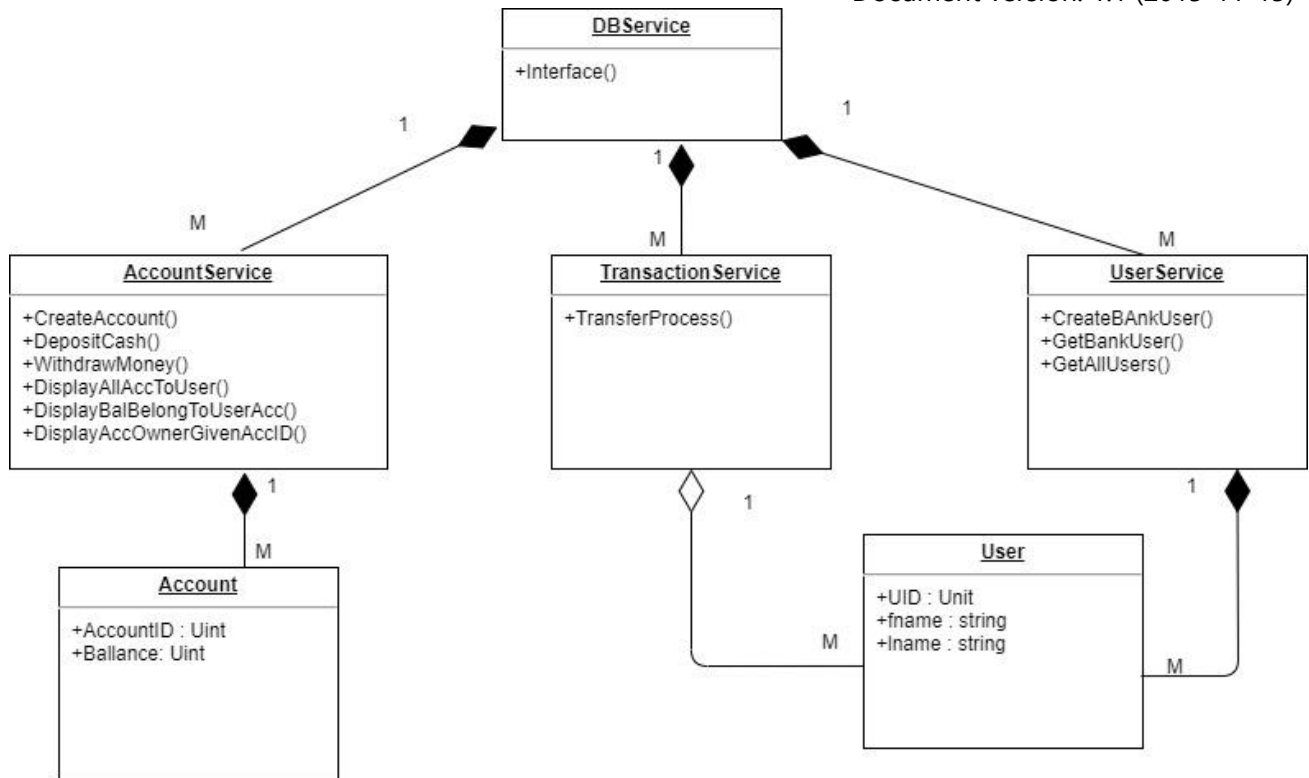
With regard to the Bank Application Report (document), there are two behavioral diagrams, such as; activity diagram and use case diagram. As the first step, the user will be able to insert required data to the system, after system accept the user inputs, system will generate a profile for their customer, and if user want to change or update his/her profile, the system will provide that facility to them. As the second main process, if user wanted generate an account he/she can insert data and information related to account details, then the system will create an account for the users and similarly, user will be able to maintain his/her account. Third process is related to transaction, when the customer of this bank insert data and information related to transaction, the system will create transaction and as other processes, the customer is able to manage transactions.

(Structural diagrams)

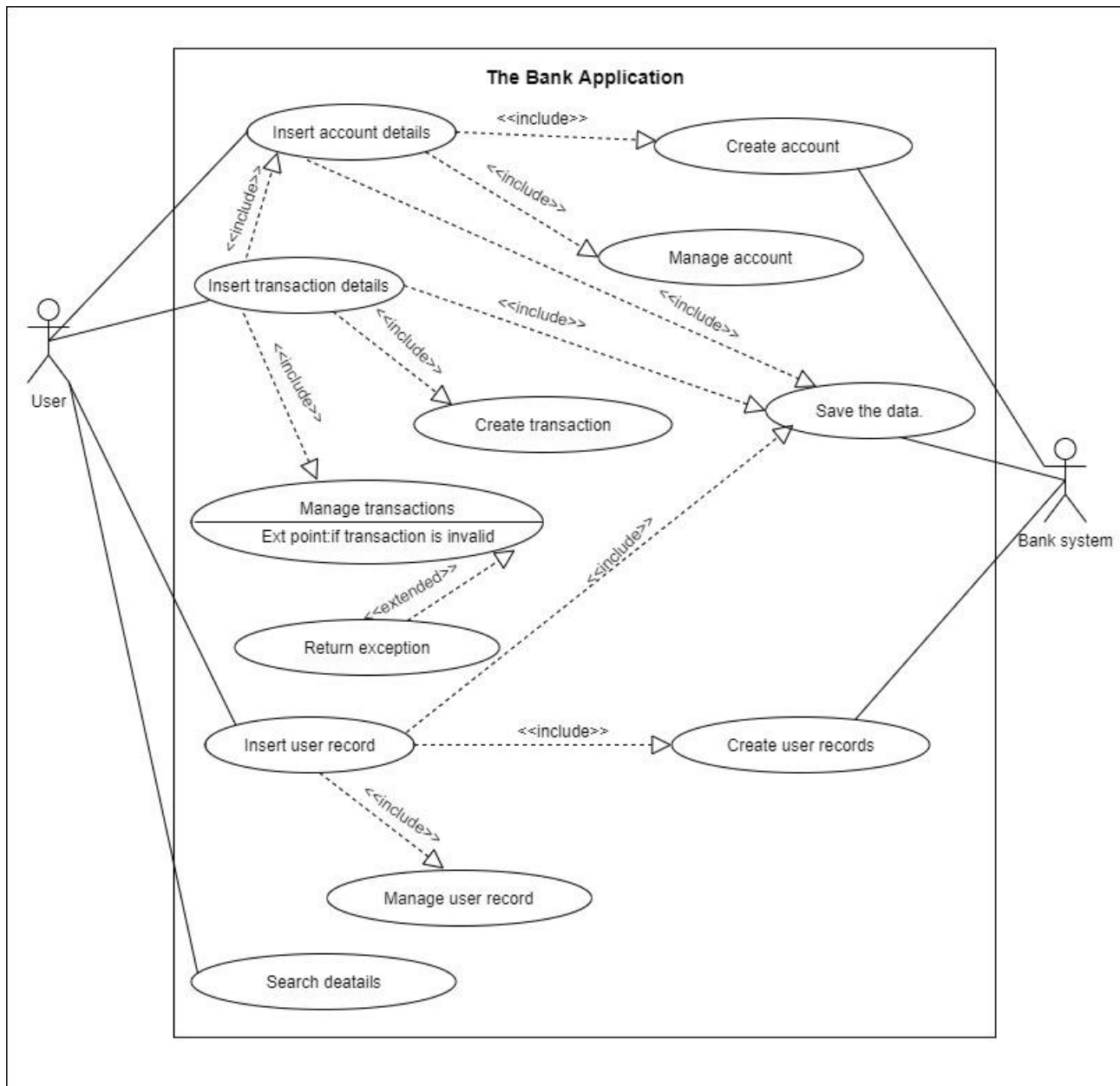
When considering structural diagram there are basically six classes, namely; AccountService, DBService, TransactionService, UserService, user and account in my Bank application Project. AccountService class obtains data from User class and Account class and Account class mainly generate methods related to account handling. For example: CreateAccount(), DepositCash(), WithdrawMoney() and etc... DBService class generate an Instance() method and that will affect to AccountService class, TransactionService class and UserService class. Similarly, UserService class will obtain data from User class. User class includes attributes such as; UID, fname and lname and Account class consist of attributes such as; AccountID and Balance.

08. UML diagram

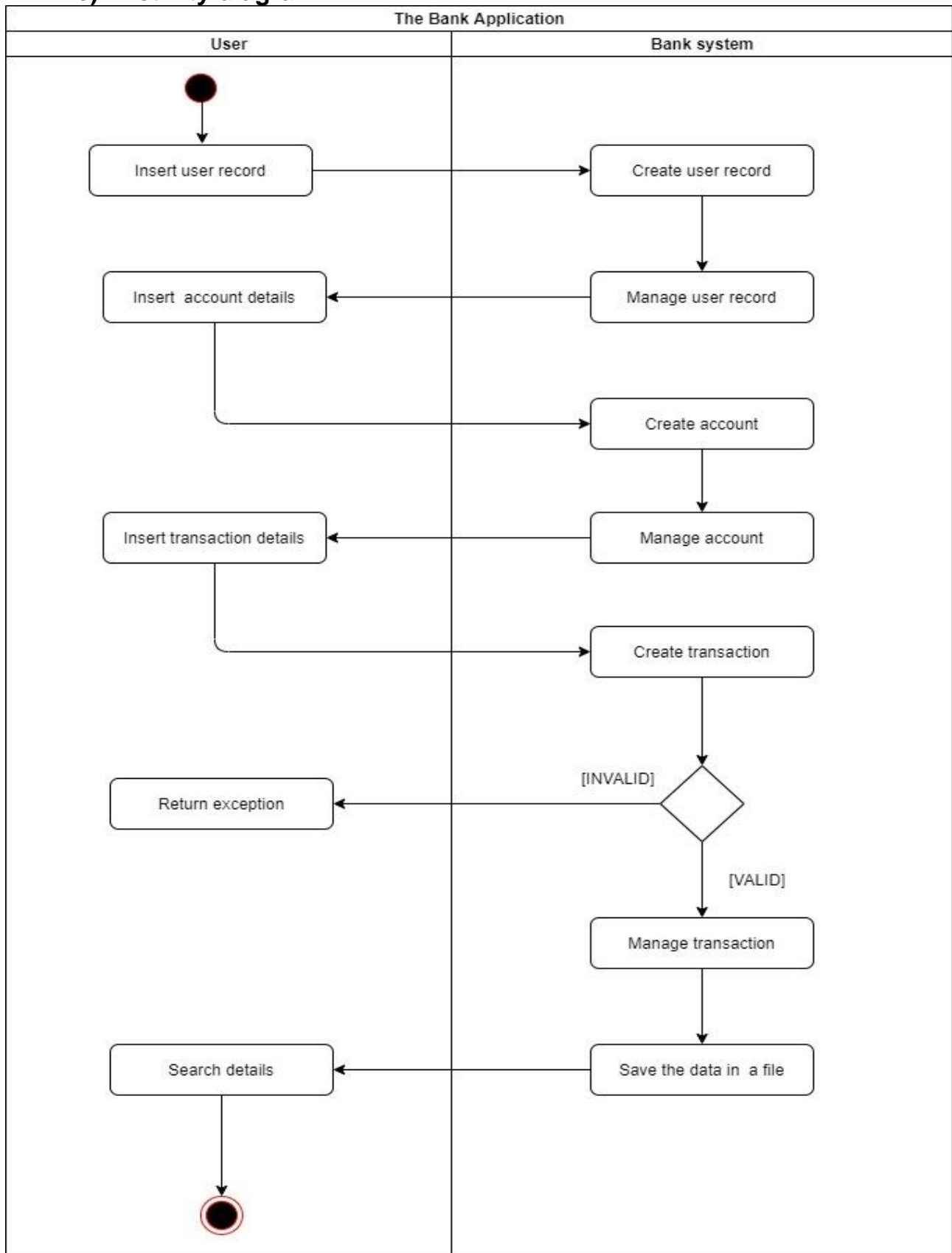
a) Class diagram



b) Use case diagram



c) Activity diagram



09. Difficulties

- I couldn't obtain the required outcome when I apply the BankDLL because the respective First name (fname) and the Last name (lname) didn't set into the BankDB object.

10. Suggestions and further improvements

- User Interface developments
- User Input validations
- Business rule validations

11. REFERENCES

[1] Three tier architecture diagram

https://en.wikipedia.org/wiki/Multitier_architecture#/media/File:Overview_of_a_three-tier_application_vectorVersion.svg

[2]UML diagrams

<https://creately.com/blog/diagrams/uml-diagram-types-examples/>

[3] Remote service component

<https://www.c-sharpcorner.com/article/net-remoting-in-a-simple-way/>

[4] Remote server component

<https://docs.microsoft.com/en-us/troubleshoot/dotnet/csharp/create-remote-server>

[5] .NET Rremoting

<https://www.youtube.com/watch?v=3Qt7TTS1u4A&t=970s>

[6] .NET Rremoting services and Web service

https://www.youtube.com/watch?v=3Qt7TTS1u4A&list=RDCMUCCTVrRB5KpliK6V2GGVsR1Q&start_radio=1&rv=3Qt7TTS1u4A&t=981

[7] Façade design pattern

https://www.tutorialspoint.com/design_pattern/facade_pattern.htm

