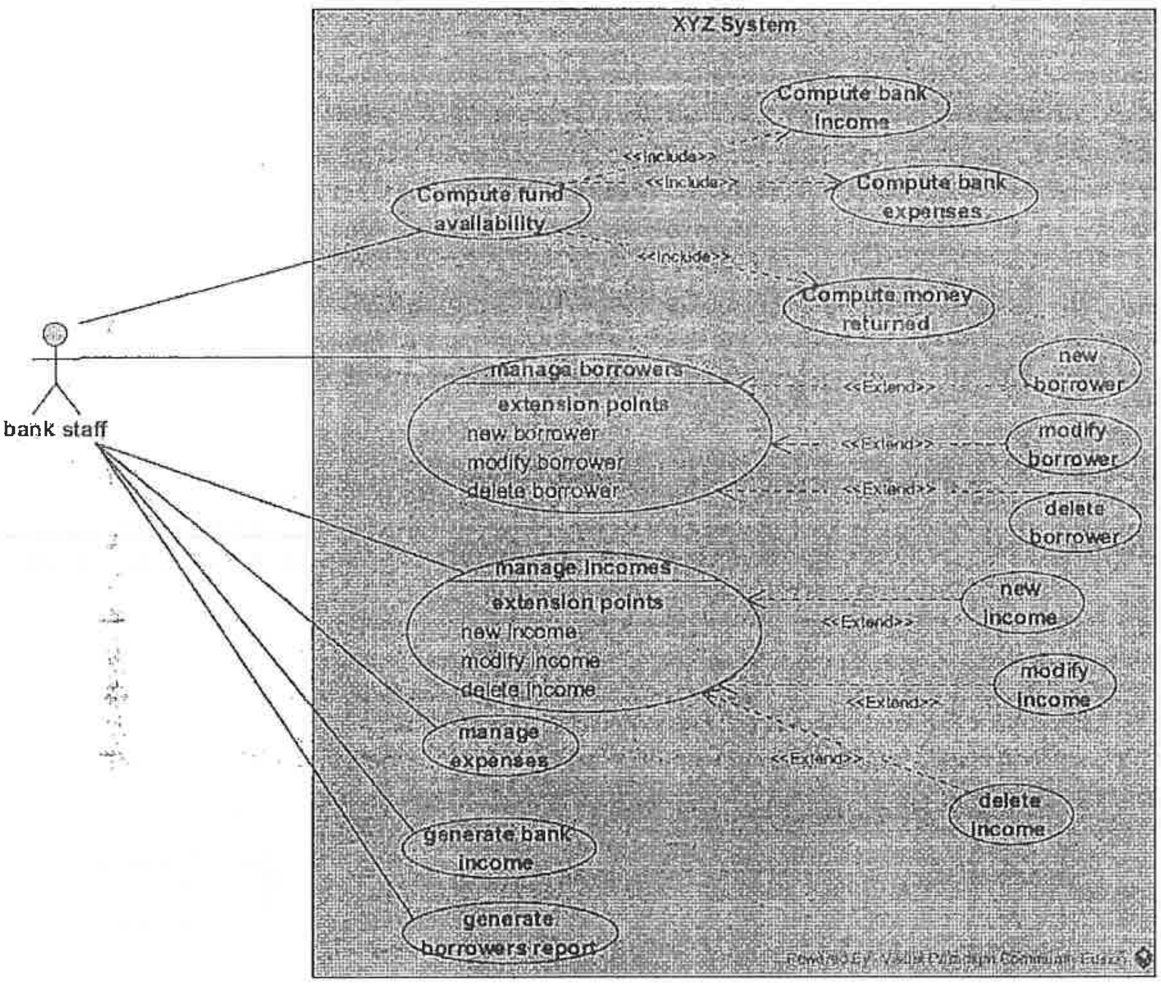
Solver: Yew Shao Jie

Email Address: Yews0012@e.ntu.edu.sg

1.

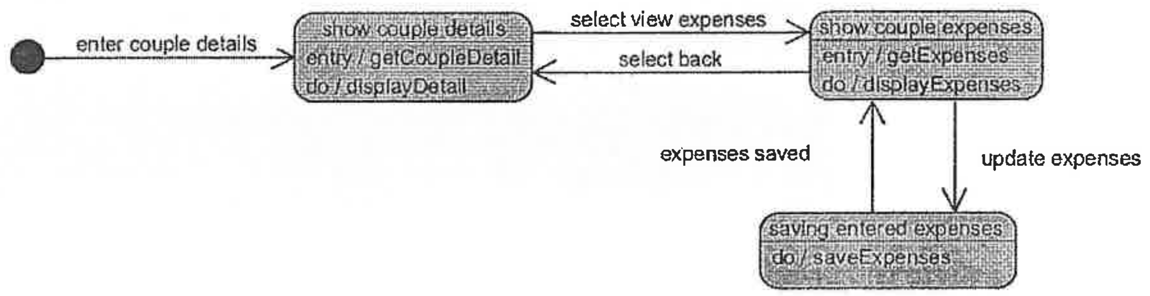
(a)



(b)

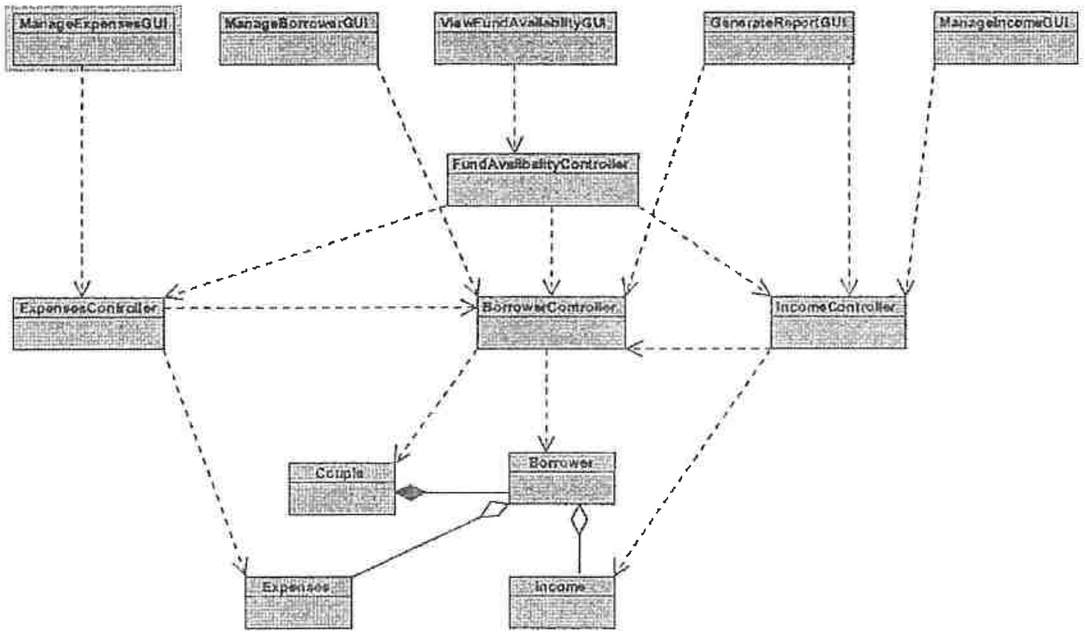
ID	Compute Fund Availability
Actor	Bank Staff
Description	It computes the funds availability when couple request to lend money from bank. It computes based on the bank income, the bank expenses and the amount of money returned by the borrowers. This includes the compute bank income use case, compute bank expenses use case and Compute returned money use case.
Pre condition	A couple is selected for computing the fund availability.
Post condition	Display the fund availability of the couple selected
Sequence	1. The bank staff enter the requested money of the selected couple and submit 2. The system invokes compute bank income of the selected couple 3. The system invokes compute bank expenses of the selected couple 4. The system invokes compute returned money of the selected couple 5. The system computes the fund availability for the couple 6. If the requested money is lower or equal to the fund availability of the couple, the requested money is displayed
Alternative	6a. If the requested money is more than the fund availability, display the fund availability.

(c)

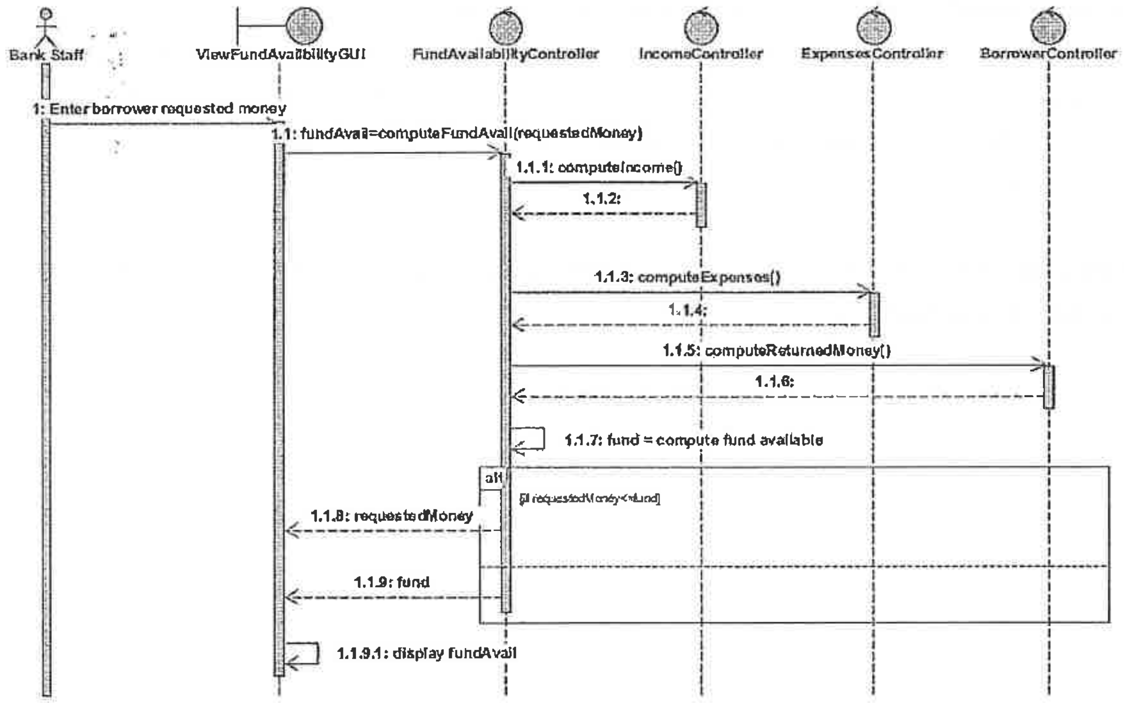


2.

(a)


ManageExpensesGUI
ManageBorrowerGUI
ViewFundAvailabilityGUI
GenerateReportGUI
ManageIncomeGUI
FundAvailabilityController
ExpensesController
BorrowerController
IncomeController
Couple
Borrower
Expenses
Income

(b)


Bank Staff
ViewFundAvailbilityGUI
FundAvailabilityController
IncomeController
ExpensesController
BorrowerController
1: Enter borrower requested money
1.1: fundAvail=computeFundAvail(requestedMoney)
1.1.1: computeIncome()
1.1.2:
1.1.3: computeExpenses()
1.1.4:
1.1.5: computeReturnedMoney()
1.1.6:
1.1.7: fund = compute fund available
alt
[if requestedMoney<=fund]
1.1.8: requestedMoney
1.1.9: fund
1.1.9.1: display fundAvail

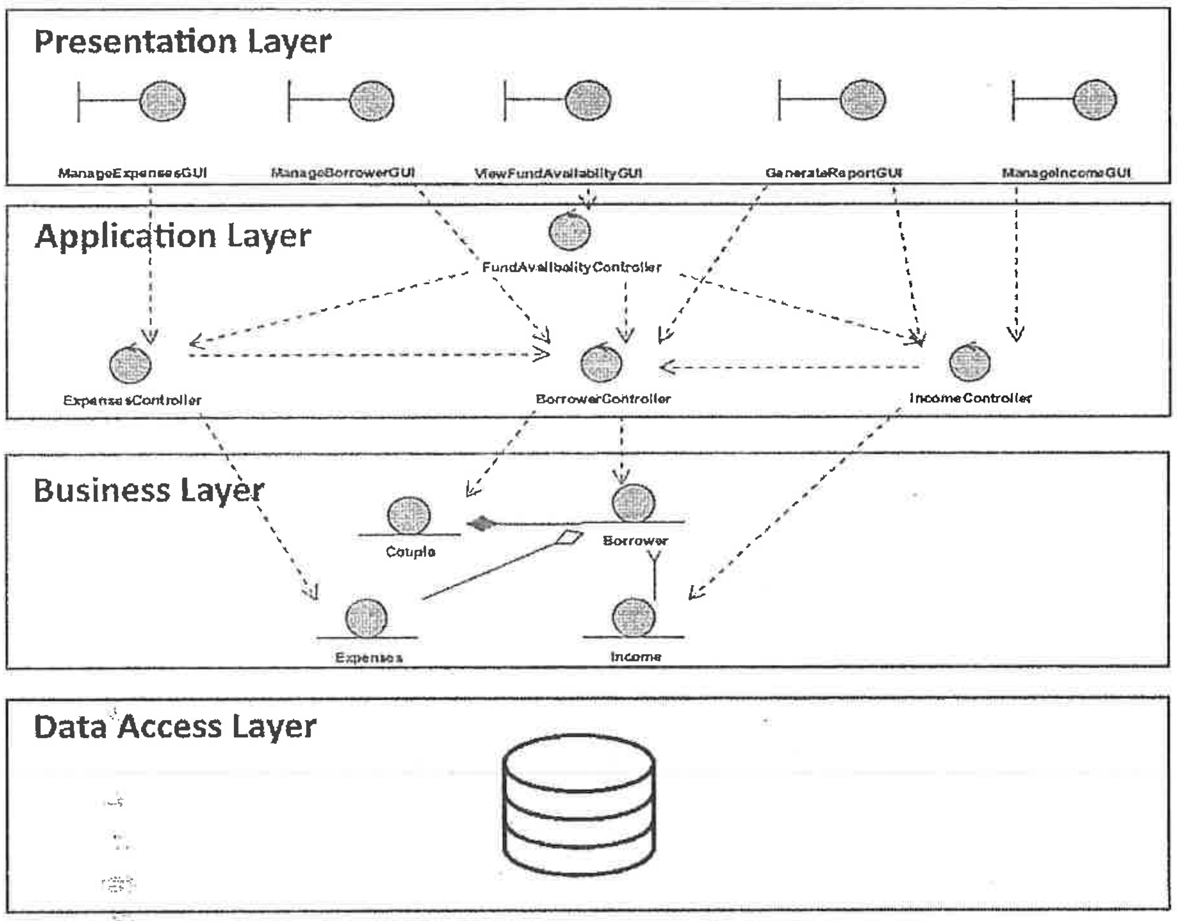
(c)
Example of software project team roles:

- **Project manager** is the overall responsibility for the successful initiation, planning, design, execution, monitoring, controlling and closure of a project.
- **Quality assurance manager** ensure that products meet certain standards of quality. Quality assurance manager plan, direct or coordinate quality assurance programs and formulate quality control policies.
- **Software Architect** is a software expert who makes high-level design choices and dictates technical standards, including software coding standards, tools, and platforms.

3
(a)
(i) Multitier architecture
(ii)
Improved Data Integrity: Data corruption through client applications can be eliminated as the data passed in the middle tier for database updations ensures its validity.
Enhanced Security: Through the implementation of several layers, enhances the data security on a service-by-service basis. As clients do not interact with the database directly, it provides less risk and confliction with unauthorized data. The placement of the business logic on a centralized server makes the data more secure.
Hidden Database Structure: The actual structure of the database often remains hidden from requesters enabling any change of the database to be transparent. Thus a process in the middle tier which exchanges data with other applications can sustain its current interface while a modification of the underlying database structure.

Complexity: Due to the componentization of the tiers, the complex structure is difficult to implement or maintain.

(iii)



(iv)

Entities (model) Objects representing system data, often from the domain model.
Boundaries (view/service collaborator) Objects that interface with system actors (e.g. a user or external service). Windows, screens and menus are examples of boundaries that interface with users.
Controls (controller) Objects that mediate between boundaries and entities. These serve as the glue between boundary elements and entity elements, implementing the logic required to manage the various elements and their interactions. It is important to understand that you may decide to implement controllers within your design as something other than objects – many controllers are simple enough to be implemented as a method of an entity or boundary class for example.

(b)

(i) **Design Problems**

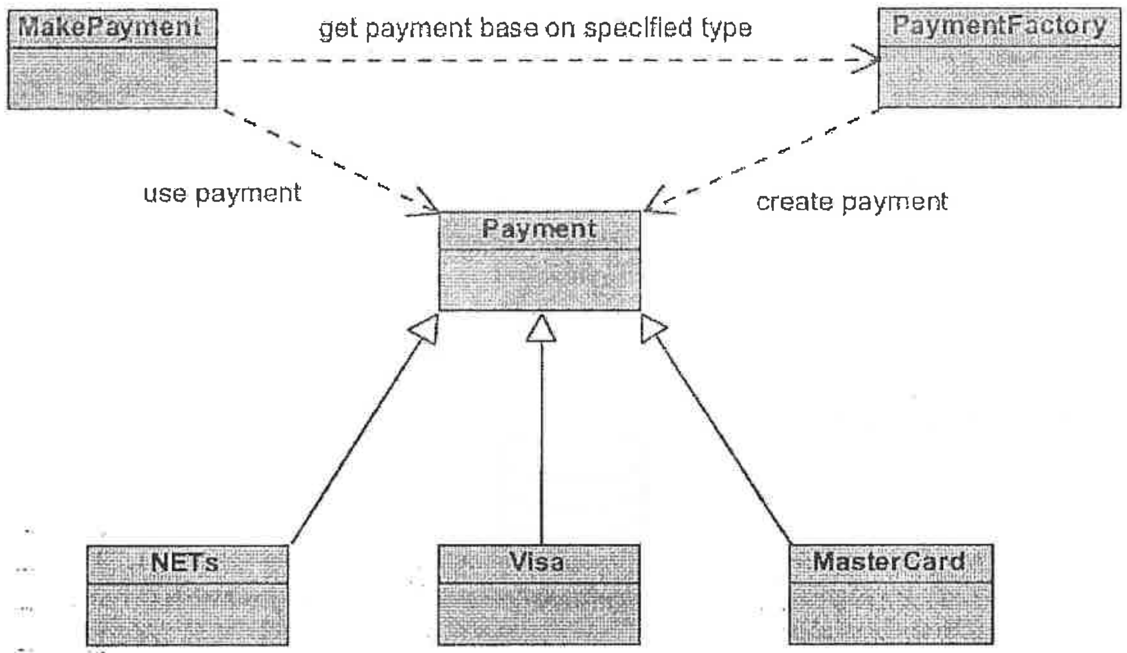
A class should be configured with an algorithm instead of implementing an algorithm directly.

An algorithm should be selected and exchanged at run-time.

(ii)

Strategy Pattern

(iii)



MakePayment: The client that uses Payment class to peform payment

PaymentFactory: This class creates the instance of Payment base on the selected payment by client

Payment: The superclass that acts as interface for all the strategy class.

NETs, Visa, MasterCard: Classes that inherited Payment class. Payment is used as an interface to switch between this classes during runtime. Each of this class is delegated to a type of strategy.

4.
(a)
(i)
Equivalence classes:

C1 Partition	**Class**
Fulltime Employed	Valid
Not Fulltime Employed	Invalid

C2 Partition	**Class**
<0	Invalid
0-99999	Valid
>99999	Invalid

(ii)

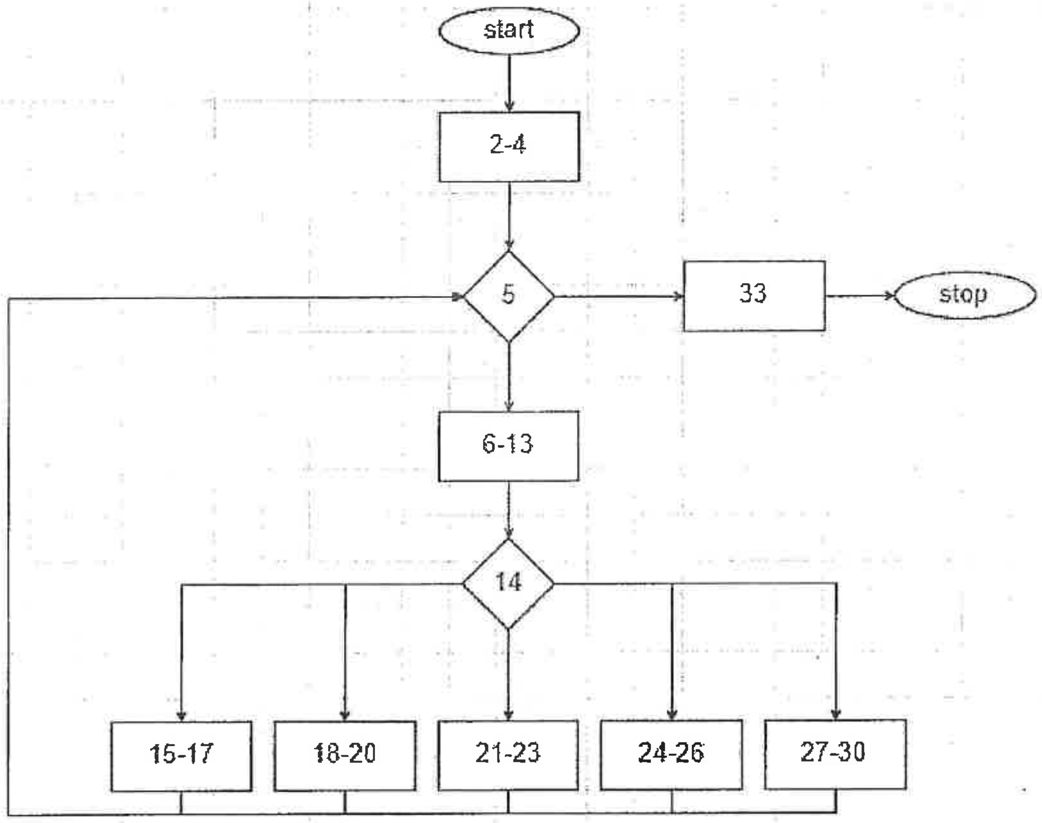
C2 Partition	**Class**	**Boundary Values**
Less than 0	Invalid	-1
Between 0 and 99999	Valid	0, 99999
More than 99999	Invalid	100000

(iii)

C1	**C2**	**Expected Result**
Fulltime Employed	0	Valid
Fulltime Employed	99999	Valid
Fulltime Employed	-1	Invalid
Fulltime Employed	100000	Invalid
Not Fulltime Employed	0	Invalid
Not Fulltime Employed	99999	Invalid

(b)

(i)



(ii)

CC: E – N+2*P

E = 16 – 12 + 2 = 6

(iii)

Path no.	**Path**	**Input**	**Expected Result**
1	2-4,5,6-13,14,15-17,5,33	Choice: 1	Add a new borrower
2	2-4,5,6-13,14,18-20,5,33	Choice: 2	Open Manage Borrower window
3	2-4,5,6-13,14,21-23,5,33	Choice: 3	Borrower deleted
4	2-4,5,6-13,14,24-26,5,33	Choice: 4	Exit the method
5	2-4,5,6-13,14,27-30,5,33	Choice: 0	Not a valid choice
6	2-4,5,33	Not possible to test as default done= false	