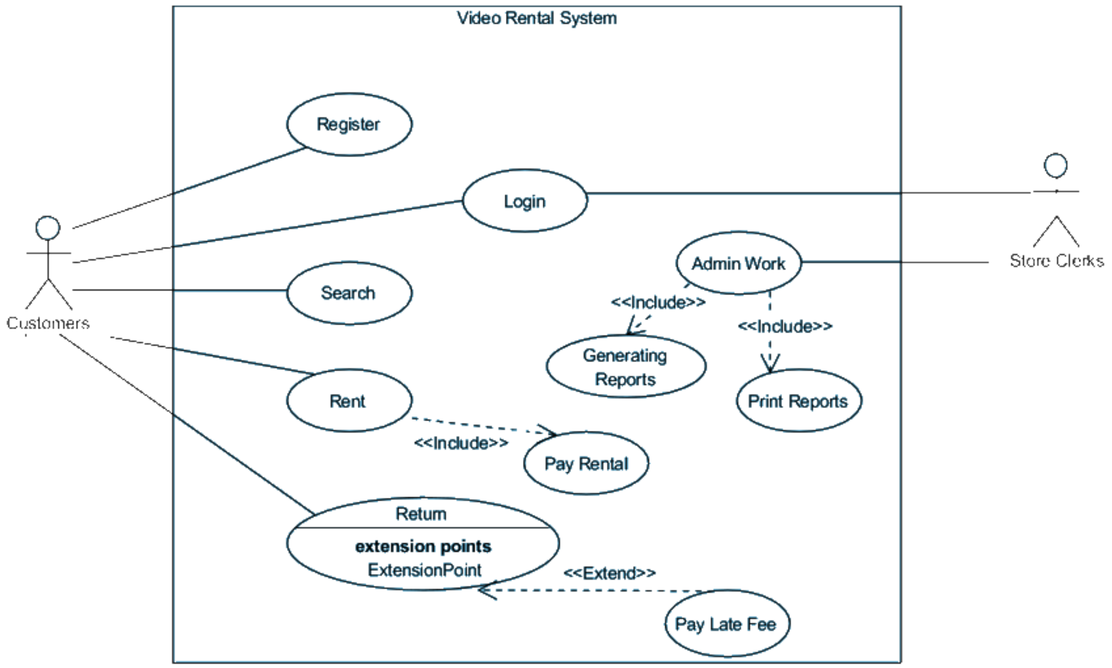
1)

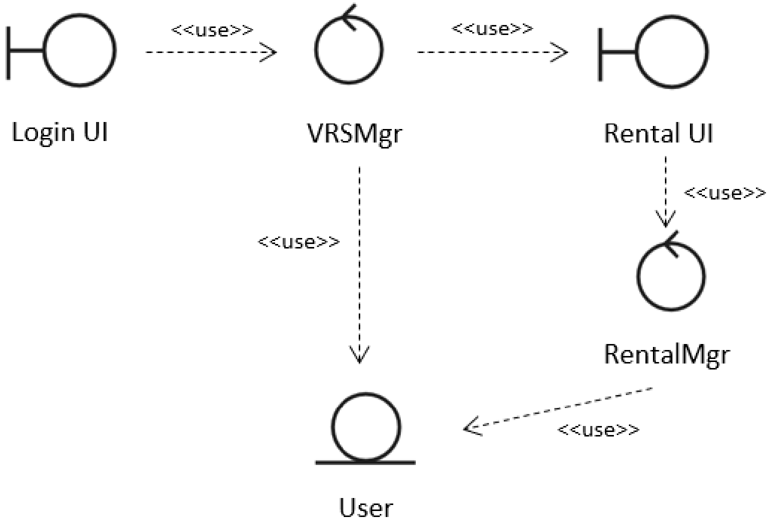
a)



b)

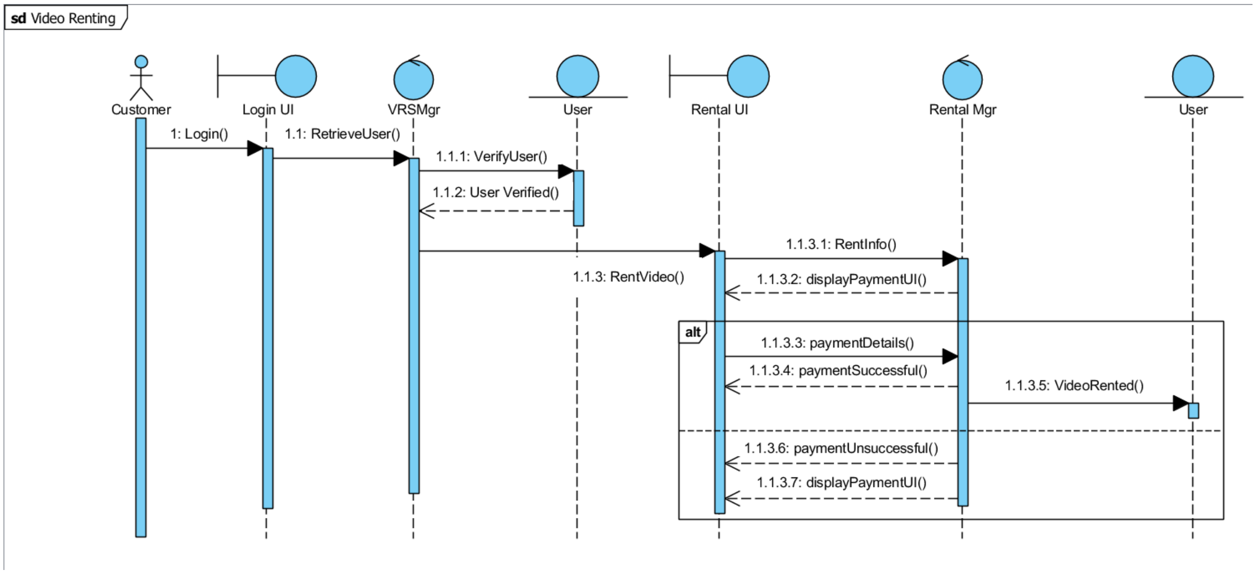
Actor:	Customer
Description:	This use case describes the functionality for renting videos.
Preconditions:	The customer must have an existing account and is logged in. The customer has no unpaid late charges.
Postconditions:	The customer successfully rents video.
Flow of Events:	1. Customer selects "Rent video". 2. System checks if customer is eligible for discount. 3. System displays price of video rental. 4. System prompts for payment. 5. Customer enters credit card details. 6. System verifies payment details. 7. System displays "Video Rented".
Alternative Flows:	AF-S6: Error in payment details. 1. System displays error message. 2. Return to step 4.

c)

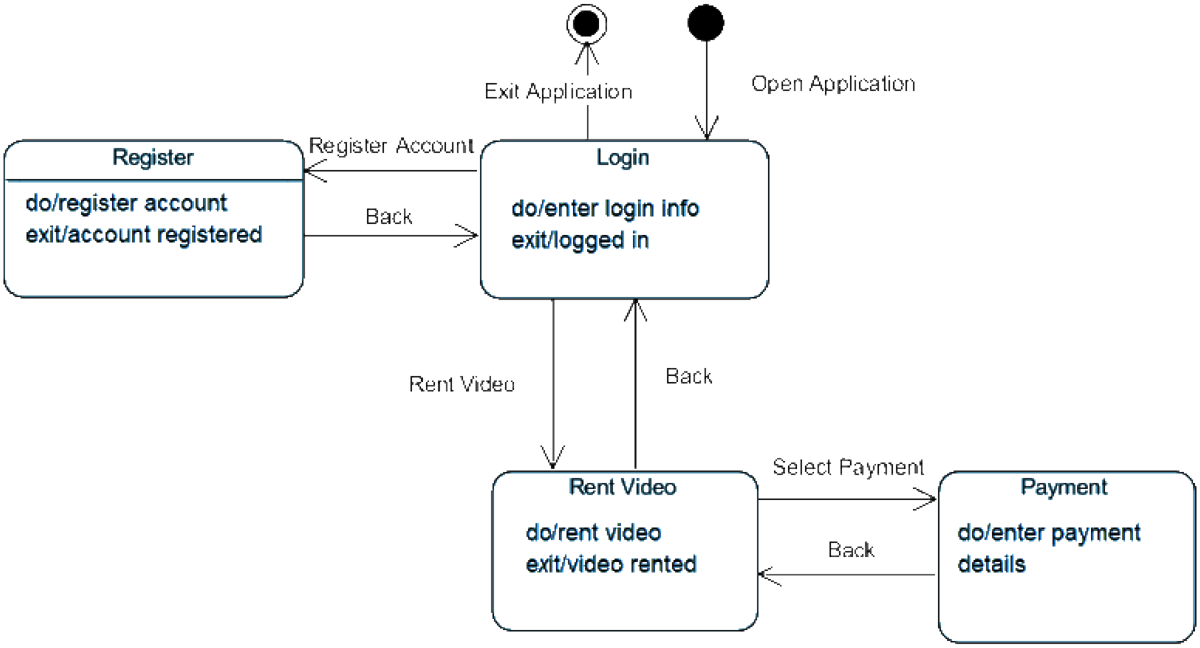

Login UI
<<use>>
VRSMgr
<<use>>
Rental UI
<<use>>
RentalMgr
<<use>>
User
<<use>>

2)

a)

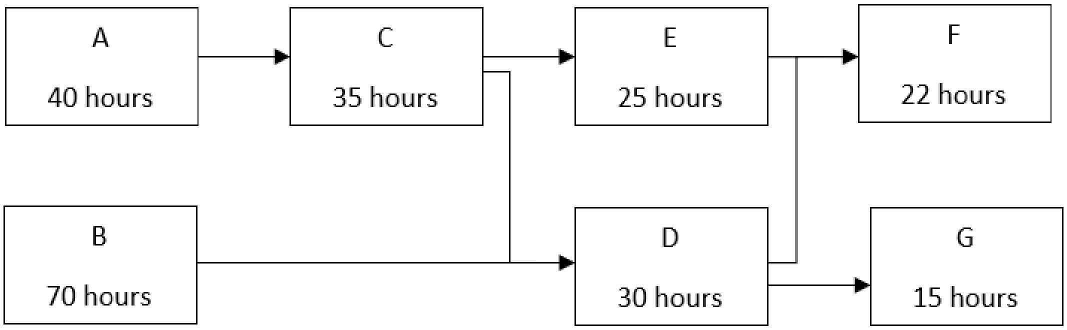

sd Video Renting
Customer
Login UI
VRSMgr
User
Rental UI
Rental Mgr
User
1: Login()
1.1: RetrieveUser()
1.1.1: VerifyUser()
1.1.2: User Verified()
1.1.3: RentVideo()
1.1.3.1: RentInfo()
1.1.3.2: displayPaymentUI()
alt
1.1.3.3: paymentDetails()
1.1.3.4: paymentSuccessful()
1.1.3.5: VideoRented()
1.1.3.6: paymentUnsuccessful()
1.1.3.7: displayPaymentUI()

b)



c)

i)



Path
ACEF = 40+35+25+22 = 122
ACDF = 40+35+30+22 = 127
ACDG = 40+35+30+15 = 120
BDF = 70+30+22 = 122
BDG = 70+30+15 = 115

Critical Path = ACDF
Shortest Duration = 127 hours

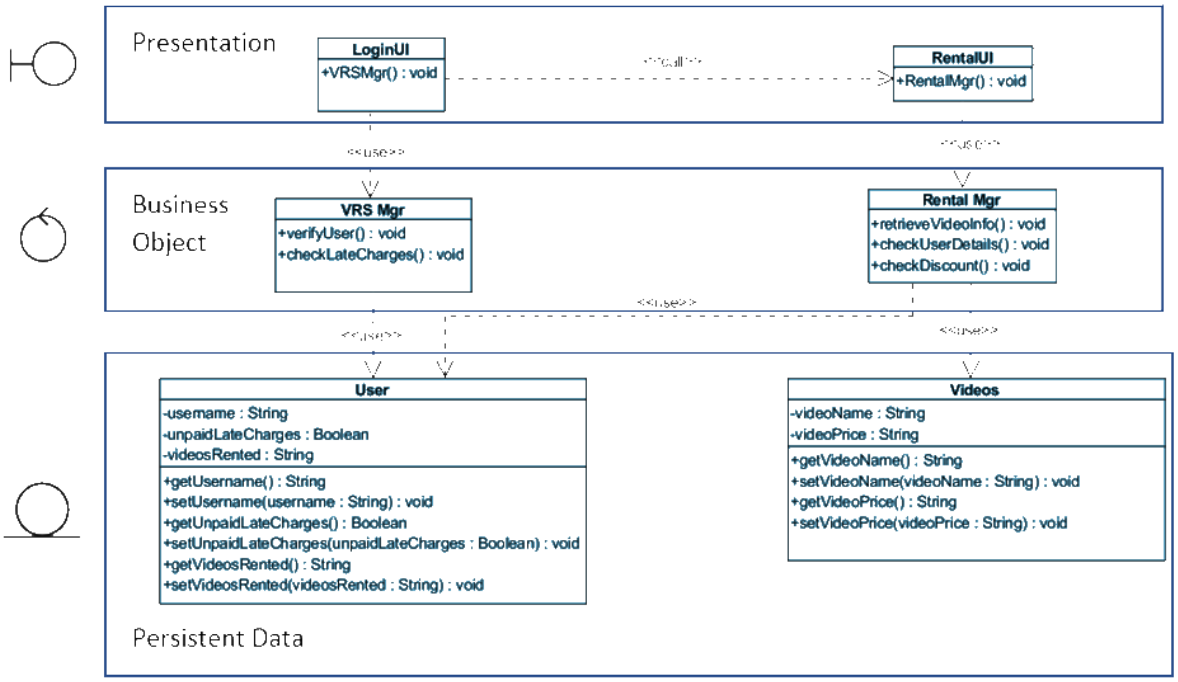
ii) Critical Path = BDF

Shortest Duration = 132 hours

3)

a)

i) Layered Architecture


Presentation
LoginUI
+VRSMgr() : void
<<call>>
RentalUI
+RentalMgr() : void
<<use>>
<<use>>
Business
Object
VRS Mgr
+verifyUser() : void
+checkLateCharges() : void
Rental Mgr
+retrieveVideoInfo() : void
+checkUserDetails() : void
+checkDiscount() : void
<<use>>
<<use>>
<<use>>
User
-username : String
-unpaidLateCharges : Boolean
-videosRented : String
+getUsername() : String
+setUsername(username : String) : void
+getUnpaidLateCharges() : Boolean
+setUnpaidLateCharges(unpaidLateCharges : Boolean) : void
+getVideosRented() : String
+setVideosRented(videosRented : String) : void
Videos
-videoName : String
-videoPrice : String
+getVideoName() : String
+setVideoName(videoName : String) : void
+getVideoPrice() : String
+setVideoPrice(videoPrice : String) : void
Persistent Data

ii) MVC Design Pattern

- Simultaneous development
- Multiple views for a model –Models can have multiple views
- High cohesion
  - MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.
- Low coupling
  - The very nature of the MVC framework is such that there is low coupling among models, views or controllers

b)

i) *Note: Not sure of answer*

1. Define interface specification: operations, arguments, type signatures, and exceptions

2. Reuse: Existing libraries and Design Patterns – choose which design pattern best fits the system

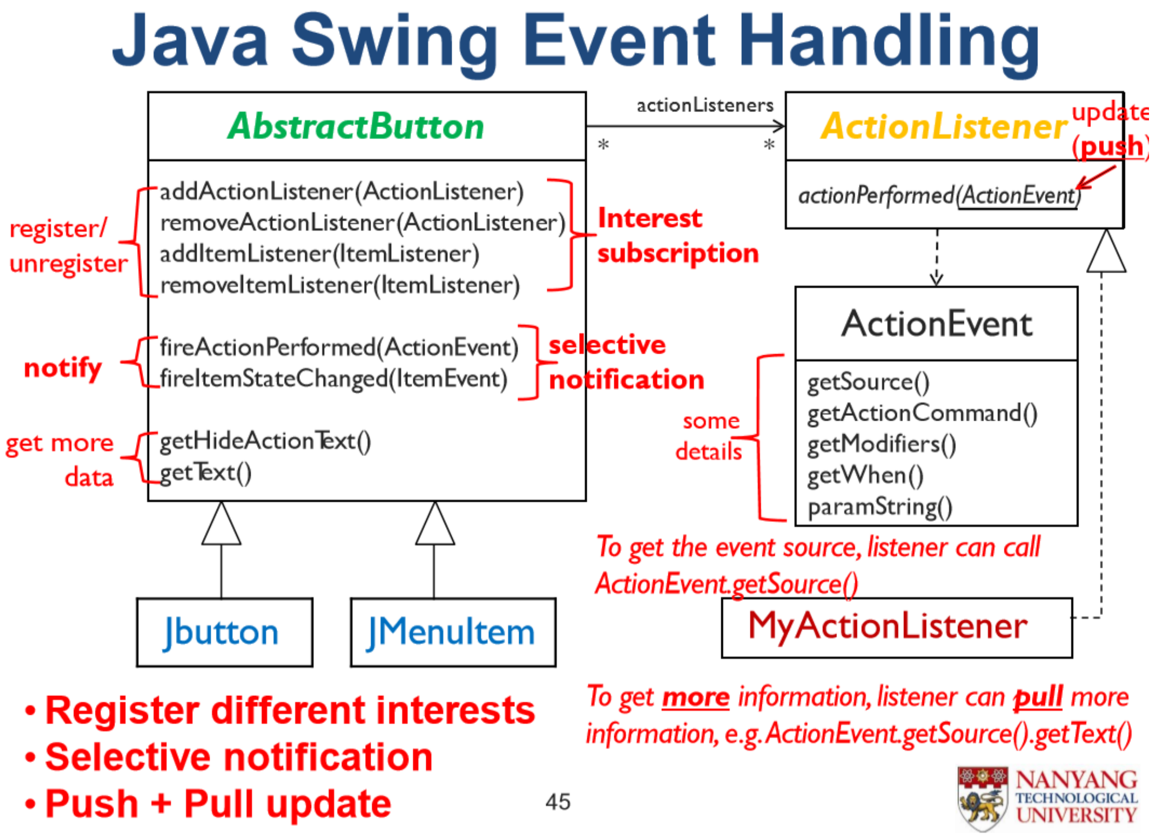
3. Restructuring: to meet the design goal maintainability, flexibility

4. Optimization: speed or memory

ii) Payment scenario in the VRS system. User can choose which credit card payment method(Visa, Mastercard etc) and the system can dynamically load the respective methods.

iii) Pull Update is when subject sends simple notification (via Observer.update()). The Observer calls back (via Subject.getData()) for details explicitly (if Observer is interested) – two way communication.

Push Update is when subject sends detailed information about the change (or update), whether Observer wants it (interested) or not – Observer does not need to call back – one way communication.



Java Swing Event Handling

To get the event source, listener can call ActionEvent.getSource()

To get **more** information, listener can **pull** more information, e.g. ActionEvent.getSource().getText()

- Register different interests
- Selective notification
- Push + Pull update

45

4)

a)

i)

1. Video Name	Valid EC: any name in ASCII format eg. ABC,abc,123,!@# Invalid EC: any characters not in ASCII format eg. Chinese, Arabic, Korean characters etc
2. Type of Video	Valid EC: Movie, TV, Documentary, Cartoon, Music Invalid EC: Any type which is not one of the above, eg. Comedy, Horror etc
3. Year of Video	Valid EC: 1900-2019 Invalid EC: <1900, >2019

ii)

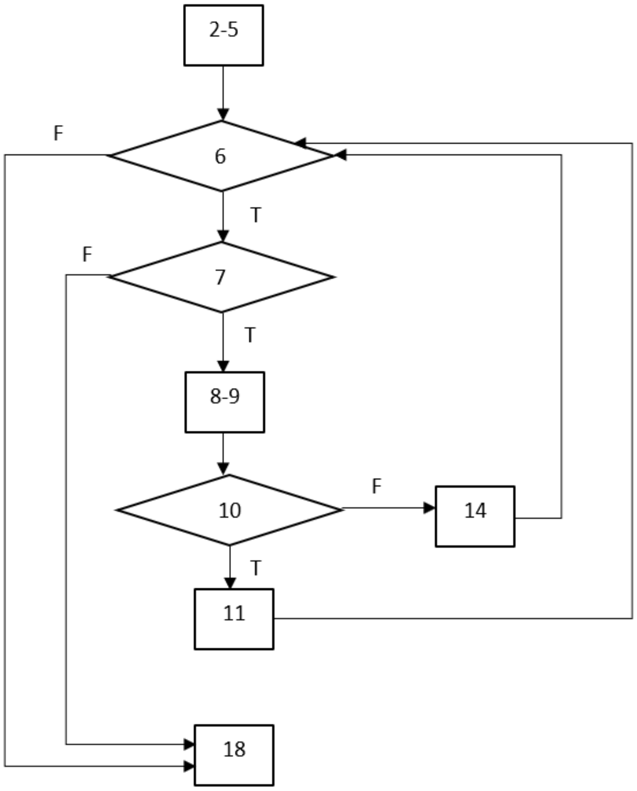
1. Video Name	2. Type of Video	3. Year of Video
No boundaries	No boundaries	Valid EC: ~~1899,~~ 1900, ~~1901~~ ~~2018,~~ 2019, ~~2020~~ Invalid EC: ~~1898~~ 1899 ~~1900~~ ~~2019~~ 2020 ~~2021~~

iii)

Video Name	Type of Video	Year of Video	Expected Result
I’m Not Stupid	Movie	2018	Approve
881	TV	2010	Approve
Gambling King	Documentary	1997	Approve
Jimmy Neutron	Cartoon	2019	Approve
Mozart Collection	Music	1900	Approve
陈龙电影	Movie	2011	Reject
Scary Movie	Horror	2001	Reject
881	TV	1899	Reject
881	TV	2020	Reject

b)

i)



ii) CC = 3+1 = 4

iii)

Path	Customer	Amount	videoRecords.length	returnAmount (line10)	Output
2-5, 6, 18	Andy	10	0	0	10
2-5, 6, 7, 18	Andy	10	1	0	10
2-5, 6, 7, 8-9, 10, 14, 6, 18	Andy	10	1	-1	Late payment for video ABC is unsuccessful
2-5, 6, 7, 8-9, 10, 11, 6, 18	Andy	10	1	1	Late payment for video ABC is successful