VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

OBJECT ORIENTED MODELLING AND DESIGN

Submitted by

VIGHNESH CHANDRASHEKHAR (1BM19CS182)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
May-2022 to July-2022

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled **"Object Oriented Modelling and Design Lab"** carried out by **Vighnesh Chandrashekhar (1BM19CS182),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements with respect to **Object Oriented Modelling and Design - (20CS6PCOMD)** work prescribed forthe said degree.

Dr. Shyamala G
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

Index Sheet

Sl. No.	Experiment Title	Page No.
1)	College Information System	4-13
2)	Hostel Management System	14-23
3)	Stock Maintenance System	24-33
4)	Coffee Vending Machine	34-43
5)	Online Shopping System	44-53
6)	Railway Reservation System	54-63
7)	Graphics Editor	64-73

1. COLLEGE INFORMATION SYSTEM

Class Diagram:

College_info_lab1.mdj — StarUML (UNREGISTERED)
College
+name: string
+location: string
+type: string
+get_name()
+get_type()
+get_location()
Admin
+name: string
+password: string
+login()
+logout()
Department
+dname: string
+dphone: int
+viewFaculty()
+viewStudent()
+addFaculty()
+addStudent()
+consistsOf
+consistsOf
+has
+has
+has
+contains
+has
Account Section
+fees: float
+ac_name: string
+total_fees()
+due_fees()
+get_account()
Placement cell
+companies: string
+coordinators: string
+types_of_jobs: string
+get_company()
+get_coordinators()
+total_students_placed()
COE
+cname: string
+generateMarkscard()
+issueHallticket()
+displayResults()
Course
+cname: string
+noOfEnrolls: int
+viewDetails()
+viewEnrolls()
Staff
+staff_id: int
+staff_name: string
+designation: string
+salary: float
+get_staffDetails()
Teaching
NonTeaching
+offers
+handles
Examination
+subject
+marks
+gives
Student
+name: string
+usn: string
+age: int
+sem: int
+get_usn()
+get_name()
+get_age()
+get_sem()
+gets
+gets
Facility
+fname: string
CIE
SEE
UG Student
PG Student
Library
+booksection
+getBooks()
+renewCard()
+returnBooks()
+payFine()
+supervises

State Diagram:

pkg CollegeInformationSystem
Login
do/enterDetails
success
LoginSuccessful
do/loginApproved
failed
updateData
getData
UpdateInformation
do/checkDetails
RetriveInformation
do/retriveDetails
Permission
do/checkAdminStatus
denied
granted
user views data
viewInformation
do/view
Admin
do/updateAdminDetails
Staff
do/updateStaffDetails
Students
do/updateStudentDetails
work finished
Commit
Finish
user logs out
Logout
do/redirectLoginPage

Use Case Diagram :

College Information System
Faculty
Student info
Handle course
Exam
«extend»
Generate marks card
«extend»
Issue hall ticket
«extend»
Generate results
COE
Add user
Admin
Student
Admin
Faculty
Student
View marks
Course registration
View course details
Verify student
«include»
Issue books
«include»
Check availability of books
Librarian

Sequence Diagram:

sd College Information System
Student
Interface
Database
Payment
1 : Select menu to pay fees
2 : Search data related to student
3 : Display details
4 : Select option and click submit
5 : Display payment modes available
6 : Select payment mode
7 : Request payment
8 : Validate payment
9 : Make payment
10 : Successful message
11 : Payment successful

Activity Diagram:

act College Information System
User login
Request validation
Wait for response
Network
Receive confirmation
Profile
[fee due]
Fee Payment
Payment gateway
Database manager
Accountant
Validate payment
Update database
Generate receipt
Home page

2. HOSTEL MANAGEMENT SYSTEM

Class Diagram:

Hostel
+noOfRooms: int
+noOfStudents: int
+viewDetails()
Admin
+name: string
+workHours: int
+gender: int
+login()
+logout()
+addHostel()
+addStudent()
Rooms
+roomNo: int
+roomType: string
+addRoom()
+deleteRoom()
+searchRoom()
+viewVacancy()
Mess
+messType: string
+getMessMenu()
Student
+name: string
+phone: int
+roomNo: int
+age: int
+gender: string
+giveFeedback()
+leaveHostel()
Payment
+modeOfPayment: string
+bankDetails: string
+amount: float
+get_paymentDetails()
SingleAccomodation
MultipleAccomodation
+noOfStudents: int
Feedback
+response: string
+viewFeedback()
+editFeedback()
+deleteFeedback()
Secretary
+name: string
+gender: string
+age: int
+phone: int
+supervising()
Warden
+name: string
+gender: string
+age: int
+phone: int
+manageHostel()
+getStudentList()
+consistsOf
+consistsOf
+consistsOf
+consistsOf
+contains
+has
+makes
+give
1
*
1..*

State Diagram:

hostel lab2 state.mdj — StarUML (UNREGISTERED)
stm HostelManagementSystem
Idle
new user
Login: StateMachine2
Login
do/enterUsername
do/enterPassword
verify details
Verification
do/verifyUsername
do/verifyPassword
unsuccessful validation
successful validation
RegisterFaculty: StateMachine3
GetUserDetails
do/getName
do/getDept
do/getPassword
verify
verifyDetails
unsuccessful verification
successful verification
success
UpdateDatabase
failure
Warden
do/getPreferences
unsuccessful verification
verify preferences
Verify
do/verifyHostelName
do/verifyHostelType
allocation unsuccessful
successful verification
CheckDatabase
allocation successful

UseCase Diagram:

Hostel Management System
Collect fee
«include»
Verify Student
Maintain Mess
Maintain Attendance
Approve room allotment
Act on complaint
«extend»
Add room
Edit hostel information
«include»
Delete room
Generate reports
«include»
Maintain accounts
Change mess
Book room
Change hostel
Make complaint
Change room
Fee payment
Create menu
Take baggage to room
Give feedback
Check hostel facilities
Search rooms
Give permissions
Admin
Accountant
Students
Mess Staff
Staff
Warden

Sequence Diagram:

sd HostelManagementSystem
Student
Login
Room
Database
Payment
1 : Enter login details
2 : Login successful
4 : View rooms
3 : Login failed
5 : Show rooms
6 : Book room
7 : Check availability
8 : Is available
9 : Not available
10 : Error message
11 : Make payment
12 : Payment successful
13 : Room allocated
14 : Update database
15 : Error

Activity Diagram:

Student
Database
Payment
Login
fail
[correct credentials]
Book room
Check availability
[not available]
[available]
Allot room
Make payment
Error message
Generate bill
[payment fail]
[payment success]
Calculate cost

3. STOCK MANAGEMENT SYSTEM

Class Diagram:

Vendor
+vendor_id: Int
+name: String
+address: String
+rating: Int
+add_Vendor()
+view()
+delivery()
+sell_produce()
+dealer
1..*
*
+sells
Product
+id: Integer[1]
+name: String[1..*]
+quantity: Integer
+price: Float
+addProduct()
+editProduct()
+deleteProduct()
+searchProduct()
+items
*
*
Customer
+id: Integer
+name: String
+mobile: Integer
+email: String
+address: String
+addCustomer()
+deleteCustomer()
+update()
+displays
1..*
*
*
*
Purchase
+date: String
+code: Integer
+price: Float
+quantity: Float
1
+pays
0..1
Store
+store_id: Integer
+name: String
+address: String
+type: String
+addStore()
+update()
+delete()
Stock
+id: Integer
+item: String
+description: String
+type: String
+quantity: Float
+addStock()
+updateStock()
+deleteStock()
+searchStock()
Bill
+id: Integer
+type: String
+description: String
+customer_id: Integer
+calculate()
+update()
+generate()
+add()
Super Market
+owner: String
+description: String
+size: Float
+edit()
+view()
Departmental Store
+specification: String
+description: String
+view()
Warehouse
+warehouseSize: Float
+type: String
+viewDetails()
+add()
+delete()

State Diagram:

stm StockManagementSystem
Inventory check
do/check available inventory
exit/find deficit or excess item
no trader found / request reject
request for sale or purchase
Sale or purchase request
Contact traders
Find trader for sale or purchase
insufficient balance / request rejected
trader found / request confirmed
Sale or purchase order
Verifies
entry/check the order details
Order follows to accounts
Order
entry/place sale or purchase order

UseCase Diagram:

Stock Maintenance System
Calculate exp stock
Calculate req stock
Place order
View returns
Find invoice
Find factory
Modify product
Pay taxes
Track order
Report quality issues
Search stock
Receive payment
Check order
Search product
Receive shipment with bill
Stock level proper
«extend»
Check quality criteria
«extend»
Detective shipment
Update level
Pay bill
«include»
Shipment error
«extend»
Manager
Company owner
Supplier
Inspector
Inspector clerk
Customer
Shipment clerk

Sequence Diagram:

interaction StockMaintenanceSystem
Employee
Store
Customer
Payment
Vendor
1 : Login
2 : Display items on login successful
3 : Display error on login fail
4 : Select items
5 : Make payment
6 : Payment status returned
7 : Payment successful and order placed
8 : Payment failed
9 : Inform employee about the order details
10 : Record order details
11 : Send order details
12 : Deliver the items
13 : Logout
14 : Logout confirmation

Activity Diagram:

Accept user login
Request validation
Wait for response
Receive confirmation
System ready to receive ordres
Order received
Check inventory
[stock exists]
Accountant
Inventory manager
Sales agent
Validate payment
Update inventory
Generate bill
Deliver item

4. COFFEE VENDING MACHINE

Class Diagram:

«enumeration»
items
coldCoffee
blackCoffee
latte
cappuccino
Money_Box
+coins: int
+coinCount()
+returnCoins()
+deduceCoins()
+total()
+display()
Coin_Slot
+coinValue: Integer
+coinCount: Integer
+countCoin()
+calculateTotal()
Selector
+items: varchar
+prize: Integer
+ingredients: varchar
+select()
+edit()
+display()
Vending_Machine
+id: varchar
+type: String
+name: String
+dispenseCoin()
+dispenseProduct()
+comparePrice()
+checkAvailability()
+updateProduct()
Display
+coins: Integer
+items: varchar
+display()
+selects
+screen
+use
+customer
1
1..*
User
+name: String
+id: varchar
+phoneNumber: real
+address: String
+details()
+edit()
+delete()
Coffee_Machine
+qSugar: Integer
+qMilk: Integer
+coffee: varchar
+addItems()
+recipe()
Product
+productName: String
+pid: varchar
+price: Integer
+update()
+retrieve()
+add()
+coffee
0..1
dispense
+timeTaken: Float
+quantity: Float
Staff
+salary: Float
+workingDays: Integer
+update()
+search()
Boss
+salary: Float
+location: String
+gender: String
+update()
+add()
Others
+occupation: String
+gender: String
+add()
+update()

State Diagram:

stm CoffeeVendingMachine
Idle
coins / set balance
Collecting money
do/add to balance
cancel / refunded coins
[change<0]
select
[item empty]
Compute change
do/test items and compute change
[change=0]
[change>0]
dispense: StateMachine3
Make change
do/make change

stm StateMachine3
Choose row
do/move arm to correct row
arm ready
Choose column
do/move arm to correct column
arm ready
Get item
do/push item off shelf
pushed

UseCase Diagram:

Coffee Vending Machine
«extend»
Buy item
Choose item
«include»
«extend»
Take item
Customer
Payment
«extend»
Dispense charge
Dispense coffee
Black coffee
Cappuccino
Cold coffee
Latte

Sequence Diagram:

sd Coffee Vending Machine
Customer
Product management
Vending machine
Lifeline4
Service man
1 : browse products
2 : product details
3 : select product
4 : customise
5 : show order
6 : deposit coins
7 : verify coins
{1..10 seconds}
8 : dispense change
10 : make coffee
9 : collect change
11 : order processing
12 : dispense coffee
13 : collect coffee
14 : clean machine
15 : add ingredient price
16 : collect revenue
«create»
17 : ask for receipt
Printer
18 : print
19 : send receipt
«destroy»
20 : turn off

Activity Diagram:

act Coffee Vending Machine
Customer
Coffee dispenser
Payment
Select coffee
Check payment
Accept coins
[is available]
Insert coins
Error message
Calculate amount
[not available]
Collect coffee
Make coffee
[amount:price]
Received amount
Get change
Dispense coffee
change:[amount>price]

5. ONLINE SHOPPING SYSTEM

Class Diagram:

WebUser
+loginId: string
+password: string
+state: UserState
+getId()
+getPassword()
+getState()
+has
0..1
1
Customer
+name: string
+phone: int
+email: string
+age: int
+gender: string
+address: string
+edit()
+login()
+logout()
«enumeration»
UserState
new
active
blocked
banned
«enumeration»
OrderStatus
orderPlaced
shipped
delivered
returned
1
+owns
0..1
1
+has
1
ShoppingCart
+created: date
+addItem()
+deleteItem()
+has
1
1
Account
+billingAddress: string
+isClosed: boolean
+open: date
+closed: date
+search()
+edit()
Payment
+total: int
+details: string
+mode: string
+makePayment()
+cancelPayment()
1
LineItem
+quantity: int
+price: int
+has
*
+places
1
*
+completes
1
Product
+name: string
+supplier: string
+getDetails()
+readReview()
+consists
1..*
1
Order
+number: int
+ordered: date
+shipped: date
+shipFrom: string
+status: OrderStatus
+total: int
+checkStatus()
+cancelOrder()
1
NetBanking
+cardNumber: string
+paymentMade: date
COD
+noOfNotes: int

State Diagram:

stm OnlineShoppingSystem
click login
LoggingIn
valid username and password
HomePage
do/display catalogue
select
Catalogue
do/display items
select
click logout
LoggingOut
Item
do/display product details
add item
AddToCart
PaymentSystem
Order
pay
fail
order
order
Quantity
do/change quantity
remove item
change quantity
Cart
do/display items in cart

stm PaymentSystem
amount
Method
do/choose payment method
select
Card
do/enter details
Validation
do/validate details
fail
success / fail
success
Processing
do/process amount from user account to system

UseCase Diagram:

5. Online Shopping System

Use Case Diagram

Online Shopping System
Customer
Login
Add item to cart
Delete item from cart
«extend»
Place order
«include»
Make payment
Payment System
Check authentication
«include»
Make finalised payment
«include»
Add tax
Admin
Maintain stock
Update quantity
Add items
Remove items

Sequence Diagram:

sd OnlineShoppingSystem
Customer
Website
Product
Cart
Payment
1 : Validate login
2 : Login successful
3 : Search product
4 : Product found
5 : Add product to cart
6 : Product added
7 : Order
8 : Request payment
9 : Do payment
10 : Validate payment
11 : Successful
12 : Order placed

Activity Diagram:

Customer
Online shop
Add product to cart
[no]
[finished?]
[yes]
Checkout
Login
[anonymous user]
[user known]
Confirm order
Query order or delivery information
Process payment
Process delivery
Send confirmation

6. RAILWAY RESERVATION SYSTEM

Class Diagram:

RailwayStation
+name: string
+address: string
+getDetails()
1
*
+contains
Train
+name: string
+capacity: int
+getDetails()
1
+takes
*
Admin
+name: string
+password: string
+login()
+logout()
+manageSystem()
1
*
+supervises
Passenger
+name: string
+age: int
+gender: string
+phone: int
+occupation: string
+getDetails()
1
*
+books
TicketBooking
+category: string
+price: int
+addCategory()
+deleteCategory()
1
+pays
1
Payment
+amount: int
+mode: string
+makePayment()
Bank
+name: string
1
+uses
1
+has
1
1
Ticket
+departure: time
+arrival: time
+startPoint: string
+destPoint: string
+book()
+cancel()
General
+noOfTickets: int
+reserve()
+getWaitingList()
Tatkal
+noOfAvailableSeats

State Diagram:

stm RailwayManagementSystem
click login
LoggingIn
do/enter phone number
Waiting
do/wait for confirmation
ChooseTrain
!available
BookTicket
do/select seat
success
PaymentSystem
available
select
valid otp
confirmed
HomePage
do/display trains
Confirmation
exit
ChooseCategory
do/choose type of train
fail
click logout
LoggingOut
available
Filter
do/options to filter based on timings and location
!available

stm PaymentSystem
amount
Method
do/choose payment method
select
Card
do/enter details
Validation
do/validate details
fail
success / fail
success
Processing
do/process amount from user account to system

UseCase Diagram:

6. Railway Reservation System

Use Case Diagram

Railway Reservation System
User
Admin
Railway system
Bank
Register
Login
Check for Availability
«extend»
«include»
Book Ticket
Fill Details
«include»
Make Payment
Cancel Ticket
Online
Offline
«extend»
Refund Money

Sequence Diagram:

sd RailwayReservationSystem
User
Railway Reservation Interface
Ticket Issue System
1 : login
2 : Return Status
3 : Validate
4 : Display Available Seats
5 : Select Seats
6 : Ask passenger details
7 : Enter details
8 : Ask payment details
9 : Make Payment
10 : Request ticket
11 : Issue Ticket
12 : Generate Ticket
13 : Display E-Ticket
14 : Print Ticket
15 : Logout

Activity Diagram:

Passenger
Railway Database
Railway Authority
Bank
User login
Authentication
check if valid
Check Seat Availability
Yes
no
Print Invalid
is available
print booking full
no
yes
option to select number of seats
fill each passenger's details
process reservation request
select payment mode
verify customer account number
pay amount
if valid
no
update reservation details and status
yes
print invalid
print ticket
debit amount

7. GRAPHICS EDITOR SYSTEM

Class Diagram:

GraphicsEditor
+applicationName: string
+newDoc()
+openDoc()
+importDoc()
+exportDoc()
+deleteDoc()
1
*
+madeOf
Document
+name: string
+createDate: date
+lastModified: date
+noOfSheets: int
+newSheet()
+save()
+print()
+deleteSheet()
+consistsOf
1
*
Group
+name: string
+createDate: date
+lastModifiedDate: date
+noOfObjects: int
+addObject()
+deleteObject()
*
+has
1
Sheet
+name: string
+noOfGroups: int
+createData: date
+lastModified: date
+copy()
+paste()
+delete()
+edit()
+addGroup()
1
+has
*
Object
+name: string
+position: point
+color: string
+create()
+delete()
+changeColor()
+zoomIn()
+zoomOut()
Text
+message: string
+font: string
+size: string
+enter()
+remove()
+edit()
ZeroDimension
+dimension: int = 0
OneDimension
+dimension: int = 1
+orientation: degree
TwoDimension
+dimension: int = 2
+orientation: degree
Point
Line
+length: int
Arc
+angle: degree
+diameter: int
Circle
+radius: int
Rectangle
+length: int
+breadth: int
Ellipse
+length: int
+breadth: int

State Diagram:

invalid
Login
do/enter login details
valid
openDocument
do/chooseDocument
create
newDocument
do/createnewDocument
created document
edit
Editor: StateMachine2
edit
addSheets
do/enter no. of sheets
Objects
DrawingObjects
do/choose tools
select objects
delete
GeometricObjects
do/choose shapes
delete
do/select sheet
delete
save
continue editing
SaveDocument
do/select the document

UseCase Diagram:

7. Graphics Editor

Use Case Diagram

Graphics Editor
Edit Document
«extend»
New Document
«include»
Save Document
«include»
«extend»
Delete Document
Graphic Tools
colour
insert
delete
User
graphics editor

Sequence Diagram:

sd GraphicsEditor
User
UI
Toolbox
Palette
Toolbar
1 : create document
2 : new doc created
3 : create tools
4 : tools selected
5 : choose font
6 : font selected
7 : save doc
8 : doc saved
9 : saved

Activity Diagram:

User
Editor
open file
old file
new file
create new file
load existing file
select item
eraser tool
drawing tool
erase diagram
draw
job not done
text
diagram
write text
draw figure
save file
job done
close file