# INDEX

<table>
<thead>
<tr>
<th>S.NO</th>
<th>NAME OF THE EXPERIMENT</th>
<th>PAGE.NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COURSE MANAGEMENT SYSTEM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1) PROBLEM ANALYSIS AND PROJECT PLANNING</td>
<td>2-10</td>
</tr>
<tr>
<td></td>
<td>2) SOFTWARE REQUIREMENT ANALYSIS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3) SOFTWARE DESIGNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4) PROTOTYPE MODEL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>E-BIDDING</td>
<td>11-25</td>
</tr>
<tr>
<td></td>
<td>1) PROBLEM ANALYSIS AND PROJECT PLANNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2) SOFTWARE REQUIREMENT ANALYSIS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3) SOFTWARE DESIGNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4) PROTOTYPE MODEL</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ELECTRONIC CASH COUNTER</td>
<td>26-33</td>
</tr>
<tr>
<td></td>
<td>1) PROBLEM ANALYSIS AND PROJECT PLANNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2) SOFTWARE REQUIREMENT ANALYSIS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3) SOFTWARE DESIGNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4) PROTOTYPE MODEL</td>
<td></td>
</tr>
</tbody>
</table>
CASE STUDY: COURSE MANAGEMENT SYSTEM

1. Problem Analysis and Project Planning

Problem Statement:
A course management system (CMS) is a collection of software tools providing an online environment for course interactions. A CMS typically includes a variety of online tools and environments.

Project Statement: The given problem is to model COURSE MANAGEMENT SYSTEM.

Objective:
The main objective of course management system is to provide services between students and faculty. They are an increasingly important part of academic systems in higher education.

2. Software requirement Analysis

Functional Requirements:

Grade Management: The system should be capable of managing student grades.

Group Management: The system should support group management features. Such features are especially important for courses with group projects.

- Ability to create groups
- Integration with homework submissions
- Integration with grade management
- Group maintenance

Non functional Requirements:

Performance Requirements:

- The system shall accommodate high number of courses and users without any fault.

Safety requirements:

- System use shall not cause any harm to human users.

Security Requirements:

- System will use secured database
  - Normal users can just read information but they can’t edit or modify anything.
- System will have different types of users and every user has access constraints.
Software Requirements:

- OperatingSystem : Windows XP/7
- CodingLanguage : JAVA/J2EE
- IDE : My Eclipse8.0
- Database : MYSQL

Hardware Requirements:

- System : 13 and above
- HardDisk : 40 GB
- Ram : 512 Mb
3. Software Designing

UML Diagrams:

i. Use Case Diagram:

```
<table>
<thead>
<tr>
<th>Role</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>view course, view student, manage topic for course, manage tutor information</td>
</tr>
<tr>
<td>Tutor</td>
<td>assign course to tutor, view tutor, view course calendar</td>
</tr>
<tr>
<td>Administrator</td>
<td>manage course, view course, view student, manage tutor information, manage student tutor</td>
</tr>
</tbody>
</table>
```

Diagram:

- System
- Tutor
- Student
- Administrator

Actions:
- assign course to tutor
- view tutor
- view course calendar
- manage course
- view course
- view student
- manage topic for course
- manage tutor information
- manage student tutor
- types of courses

Inclusion:
- <include>
ii. **ClassDiagram:**

[Diagram of class relationships]

- **Course Administrator**
  - +name
  - +password
  - +id
  - +manage course()
  - +view course()
  - +assign tutor()
  - +create course()
  - +manage statement()

- **Student**
  - +student id
  - +student name
  - +qualification
  - +contact no
  - +emailid

- **Course Calendar**
  - +day
  - +start time
  - +end time
  - +start day
  - +end day
  - +tutor name

- **Topic**
  - +topic name
  - +topic id

- **Tutor**
  - +tutor name
  - +tutor id
  - +tutor password

- **Course**
  - +course name
  - +course id
iii. **Deployment Diagram:**

![Deployment Diagram]

iv. **Collaboration Diagram:**

![Collaboration Diagram]
v. SequenceDiagram:

vi. ComponentDiagram:
vii. ActivityDiagram

\[\text{manage code} \]
\[\text{create course} \rightarrow \text{new course} \]
\[\text{create topic} \]
\[\text{assign tutor} \]
\[\text{prepare course calendar} \]
\[\text{modify course} \rightarrow \text{modify course} \]
\[\text{remove course} \rightarrow \text{remove course} \]
\[\text{course completed} \]
4. PrototypeModel:
Import java.awt.*;
Import java.applet.*;
Import java.awt.event.*;
Public class student extends Frame implements Action Listener
{
    String msg;
    Button b1=new Button("Save");
    Label LL1=new Label("student details",Label.CENTER);
    Label l1=new Label("Name:",Label.LEFT);
    Label l2=new Label("age:",Label.LEFT);
    Label l3=new Label("Sex(M/F):",Label.LEFT);
    Label l4=new Label("Address:",Label.LEFT);
    Label l5=new Label("Course:",Label.LEFT);
    Label l6=new Label("Semester:",Label.LEFT);
    Label l7=new Label(" ",Label.RIGHT);
    TextField t1=new TextField();
    Choice c1=new Choice();
    CheckboxGroup cbg=new CheckboxGroup();
    Checkbox ck1=new Checkbox("Male",false,cbg);
    Checkbox ck2=new Checkbox("Female",false,cbg);
    TextArea t2=new TextArea(" ",180,90,TextArea.SCROLLBARS_VERTICAL_ONLY);
    Choice course=new Choice();
    Choice sem=new Choice();
    Choice age=new Choice();
    Public student()
    {
        addWindowListener(new myWindowAdapter());
        setBackground(Color.cyan);
        setForeground(Color.black);
        setLayout(null);
        add(l1);
        add(t1);
        add(l2);
        add(t2);
        add(ck1);
        add(ck2);
        add(course);
        add(age);
        add(b1);
        b1.addActionListener(this);
        add(b1);
        course.add("Bsc.s");
        course.add("Bscmaths");
        course.add("Bsc physics");
        course.add("BA English");
        course.add("BCOM");
        sem.add("1");
        sem.add("2");
        sem.add("3");
        sem.add("14");
case study: E-bidding

1. Problem analysis and project planning:

ProblemStatement:
E-Bidding is the process of bidding through electronic medium. E-Procurement is the buying of goods/services or construction work through electronic medium. The e-bidding system is a flexible solution for supporting lot-based online auctions. The system has been designed to be highly scalable and capable of supporting large number, if bidders in an active auction. To help business with financier’s and purchases in online auctioning system (E-bidding)

**Project statement:** The given problem is to model e-bidding

**Objective:**
An Online Auction is service in which auction users can sell or bid for antique products through the internet.

2. **Software requirement and analysis:**

**Functional requirements:**

i. **System Feature:**
Login module will provide security and authentication to the seller and buyer. This system will only allow the administrator to see the other functionality of the system, who having a valid username and password.

ii. **User Details:**
User module will contain the all information about the seller and buyer who are registered under the online auction system. All user information like name, address, mobile number etc, is handled by the user details module.

iii. **Product Details:**
Product module will contains all the information about the product. All the information about the product name, type, price, bid time, description, photos, maximum bidder names, final bidding price etc

iv. **Seller Details:**
Seller module will contains the information about the seller who are registered under the online auction system, all seller information like name, seller id, email id etc, are handled by the seller details module.

**Non-Functional Requirements:**

**Performance requirement:**
The physical machine to be used in the online auction needs to have internal access in order to connect to the database, since all the data will be stored on the server database which the
software will need to connection.

**Security requirement:**

The online auction system will uses the secure authentication for the online auction system administrator, login id and password is associated with the system administrator to provide security over the system

No detail of the competitor bidder will be shown as it can be a case of fraud.

**Software Quality Attributes:**

The detail of the buyers and sellers is kept secret

No case of fraud is possible

**Software Requirements**

Frontend : ibm eclipse
Codinglanguage : java (jdk 1.6)
Operatingsystem : windows xp/7

**HardwareRequirements:**

PROCESSOR : Pentium IV 2.4GHz
SYSTEMTYPE : 32 bit operating system
HARDDISK : 40GB
RAM : 512MB
3. Software Designing:
   UML Diagrams
   i. Use Case Diagram:

   For Administrator:
ii. ForBidder

- Login
- Search Product
- Purchase Package
- View Detail of Product
- Bid for the Product
- Modify Bid Amount
- Logout
iii. ForSeller

- Login
- Upload Product
- Set the Time & Prize of Product
- View Bid Information
- Logout
iv. SequenceDiagram

[SequenceDiagram Image]
v. Class Diagram
vi. Activity Diagram For Bidder

Bidder

Registered?

Login

Is Valid?

Search Product, Purchase Package, View Detail Product, Bid for the Product, Modify Bid Amount

Logout
vii. Activity diagram for admin

ADMIN

Login

NO

Is valid?

yes

Manage User

Manage Product

Manage Category

Send Updates to Bidders & Suppliers

Manage Bidding

Report Generation

Logout

Logout
viii. Activity diagram for seller

```
Seller

Registered?

yes

Login

NO

Is Valid?

yes

Upload Product

Set time & prize of Product

View Bid Information

Logout
```

Registration

No
ix. State chartdiagram

![State chartdiagram]

x. ComponentDiagram

![ComponentDiagram]
xi. Collaboration Diagram
4. Prototype:

```java
package adt;
import java.util.LinkedList;
import java.util.List;
public class Auction {
    private int auctionID;
    private int quantity;
    private byte closingType;
    private double highestBid;
    private String sellerIP;
    private Item item;
    private List clients;
    private List biddingHistory;
    private int timer;
    private boolean isActive;
    public Auction(int auctionID, int quantity, byte closingType, double highestBid, String sellerIP, Item item, int timer) {
        this.auctionID = auctionID;
        this.quantity = quantity;
        this.closingType = closingType;
        this.highestBid = highestBid;
        this.sellerIP = sellerIP;
        this.item = item;
        this.timer = timer;
        clients = new LinkedList();
        biddingHistory = new LinkedList();
        isActive = true;
    }
    public boolean isActive() {
        return isActive;
    }
    public void setIsActive(boolean isActive) {
        this.isActive = isActive;
    }
    public int getTimer() {
        return timer;
    }
    public void setTimer(int timer) {
        this.timer = timer;
    }
    public Auction() {
        this.auctionID = 0;
        this.quantity = 0;
    }
}
```
this.closingType = 0;
this.highestBid = 0;
this.sellerIP = null;
this.item = null;
clients = new LinkedList();
biddingHistory = new LinkedList();
}

public List getBiddingHistory()
{
    return biddingHistory;
}

public void setBiddingHistory(List biddingHistory)
{
    this.biddingHistory = biddingHistory;
}

public Item getItem()
{
    return item;
}

public void setItem(Item item)
{
    this.item = item;
}

public int getAuctionID()
{
    return auctionID;
}

public void setAuctionID(int auctionID)
{
    this.auctionID = auctionID;
}

public byte getClosingType()
{
    return closingType;
}

public void setClosingType(byte closingType)
{
    this.closingType = closingType;
}

public double getHighestBid()
{
    return highestBid;
}

public void setHighestBid(double highestBid)
{
    this.highestBid = highestBid;
}

public int getQuantity()
{
    return quantity;
}

public void setQuantity(int quantity)
CASE STUDY: ELECTRONIC CASH COUNTER

1. Problem analysis and project planning:
Problem Statement:

This project is mainly developed for the Account Division of a Banking sector to provide better interface of the entire banking transactions. This system is aimed to give a better outlook to the user interfaces and to implement all the banking transactions like:
• Supply of Account Information • New Account Creations • Deposits • Withdraws • Cheque book issues • Stop payments • Transfer of accounts • ReportGenerations.

Project Statement: The given problem is to model ELECTRONIC CASH COUNTER.

Objective:

The main objective of the project is to develop that a number of factors relating to the time the customer spends with a bank. Without usage of technology, The banking sector cannot provide customers with a satisfactory service. So the main objective is to develop an effective service delivery which is new or significantly improved concept that is taken into practice.

So for this project of electronic cash counter we may consider an example of an ATM. Where we study different case study for ATM.

2. Software requirement and analysis:

Functional Requirements:

☐ New accounts creation
☐ Manage the transaction of account
☐ Security issue

Software Requirements:

☐ Operating System : Windows 8/10
☐ Coding Language : JAVA/J2EE
☐ IDE : My Eclipse 8.0
☐ Database : MYSQL

Hardware Requirements:

☐ System : I3 and above.
☐ Hardware : 40GB.
☐ Ram : 512 Mb.
3. Software Designing
   UML Diagrams

   i. Use Case Diagram For ATM:
ii. Class Diagram For ATM:
iii. Sequence Diagram For ATM:

```
interaction SequenceDiagram1

  c: customer
     1: insert card()
     3: request pint()
     4: enterpin()
     6: processing()
     7: option()
     8: withdraw()
     9: request()

  c: atm
     2: verify()
     5: valid()

  c: bank
```

iv. Collaboration Diagram For ATM:

```
interaction collaboration diagram of atm

  customer
     1: login to the system()
     7: system and check balance
     6: issue amount()

  bank
     4: valid()
     3: update account
     2: available on net()

  cashier
     5: deposit amount()
```
v. Activity Diagram For ATM:

- verify access code
- handle incorrect access code
- ask for amount
  - if valid future transactions
  - checking amount
- amount unavailable
  - if amount invalid
  - prepare receipts
  - amount valid
- prepare receipts
- after all stages completed
- final transactions
vi. Component Diagram For ATM:

![Component Diagram For ATM]

vii. Deployment Diagram For ATM:

![Deployment Diagram For ATM]
4. PrototypeModel

```java
import java.util.Scanner;

public class ATM_Transaction {
    public static void main(String args[]) {
        int balance = 5000, withdraw, deposit; Scanner s = new Scanner(System.in); while(true) {
            System.out.println("Automated Teller Machine");
            System.out.println("Choose 1 for Withdraw");
            System.out.println("Choose 2 for Deposit");
            System.out.println("Choose 3 for Check Balance");
            System.out.println("Choose 4 for EXIT");
            System.out.print("Choose the operation you want to perform:"); int n = s.nextInt();
            switch(n) {
                case 1:
                    System.out.print("Enter money to be withdrawn:"); withdraw = s.nextInt();
                    if(balance >= withdraw) {
                        balance = balance - withdraw;
                        System.out.println("Please collect your money");
                    } else {
                        System.out.println("Insufficient Balance");
                    }
                    break;
                case 2:
                    System.out.print("Enter money to be deposited:"); deposit = s.nextInt();
                    balance = balance + deposit;
                    System.out.println("Your Money has been successfully deposited");
                    break;
                case 3:
                    System.out.println("Choose 3 for Check Balance");
                    break;
                default:
                    System.out.println("Invalid choice");
                    break;
            }
        }
    }
}
```
3:
    System.out.println("Balance : "+balance);
    System.out.println(" ");
    break;
    case 4:
        System.exit(0);
    }
    }
    }
}