SOFTWARE ENGINEERING
LAB MANUAL

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY
Sheriguda (V), Ibrahimpatnam (M), Hyderabad,
R.R. Dist., Telangana State – 501510.
Department of Computer Science and Engineering

INDEX

1. INTRODUCTION

Objective:

To find the requirement specification (both functional and nonfunctional) of a given Problem.

Step 1:

Introduction:

Identify the product whose software requirements are specified in this document. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem. Describe the different types of user that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

Project Scope

Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. An SRS that specifies the next release of an evolving product should contain its own scope statement as a subset of the long-term strategic product vision.

Step 2:

Overall Description

Product Perspective

Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.

Product Features

Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, is often effective.

User Classes and Characteristics

Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.

Operating Environment

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

Design and Implementation Constraints

Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).

Step 3:

System Features

This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

System Feature 1

Don't really say "System Feature 1." State the feature name in just a few words.

1 Description and Priority

Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).

2 Stimulus/Response Sequences

List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.

3 Functional Requirements

Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services

provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary.

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

Step 4:

External Interface Requirements

User Interfaces

Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.

Hardware Interfaces

Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

Software Interfaces

Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

Communications Interfaces

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

Nonfunctional Requirements

Performance Requirements

If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

Safety Requirements

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.

Security Requirements

Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

Software Quality Attributes

Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

Other Requirements

Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.

2. DATA FLOW DIAGRAM

Data analysis attempts to answer four specific questions:

- What processes make up a system?
- What data are used in each process?
- What data are stored?
- What data enter and leave the system?

Data drive business activities and can trigger events (e.g. new sales order data) or is processed to provide information about the activity. Data flow analysis, as the name suggests, follows the flow of data through business processes and determines how organization objectives are accomplished. In the course of handling transactions and completing tasks, data are input, processed, stored, retrieved, used, changed and output. Data flow analysis studies the use of data in each activity and documents the findings in data flow diagrams, graphically showing the relation between processes and data.

Physical and Logical DFDs

There are two types of data flow diagrams, namely *physical data flow diagrams* and *logical data flow diagrams* and it is important to distinguish clearly between the two:

Physical Data Flow Diagrams

An implementation-dependent view of the current system, showing what tasks are carried out and how they are performed. Physical characteristics can include:

- Names of people
- Form and document names or numbers
- Master and transaction files
- Equipment and devices used

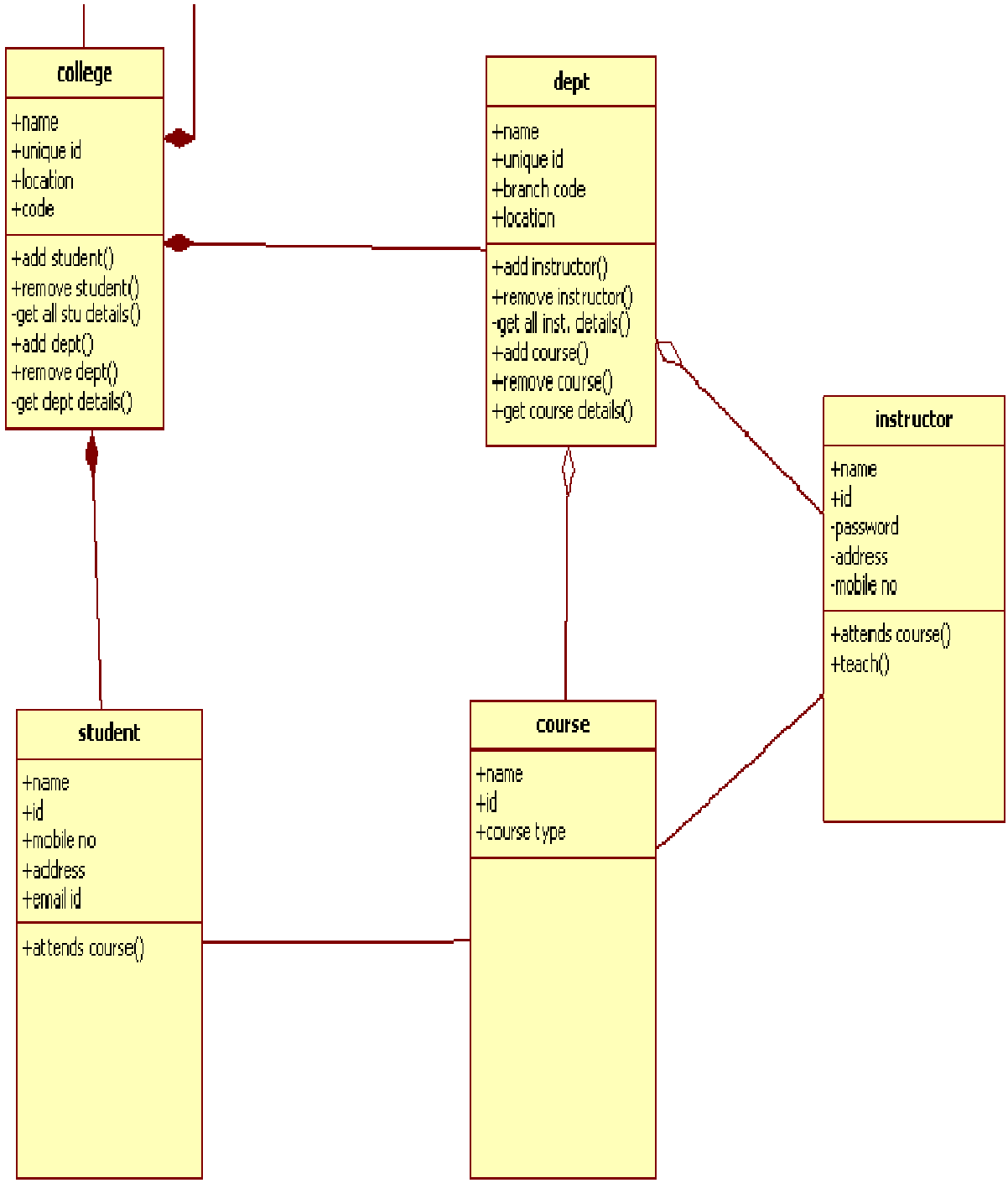
Logical Data Flow Diagrams

An implementation-*in*dependent view of the a system, focusing on the flow of data between processes without regard for the specific devices, storage locations or people in the system. The physical characteristics listed above for physical data flow diagrams will not be specified.

3. SMAPLE DESIGNS

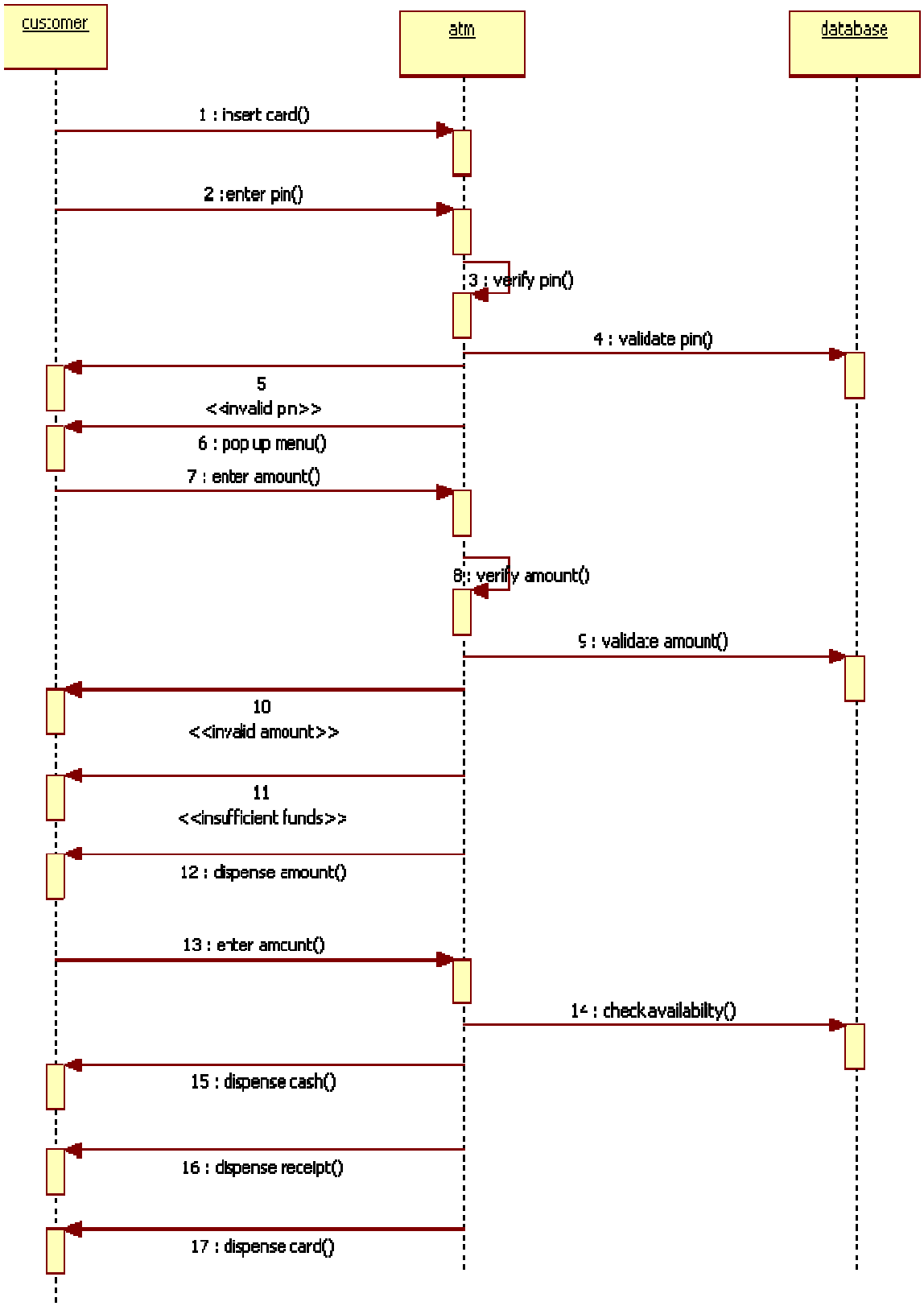
i. CLASS DIAGRAM

I (a) CLASS DIAGRAM FOR COLLEGE MANAGEMENT SYSTEM

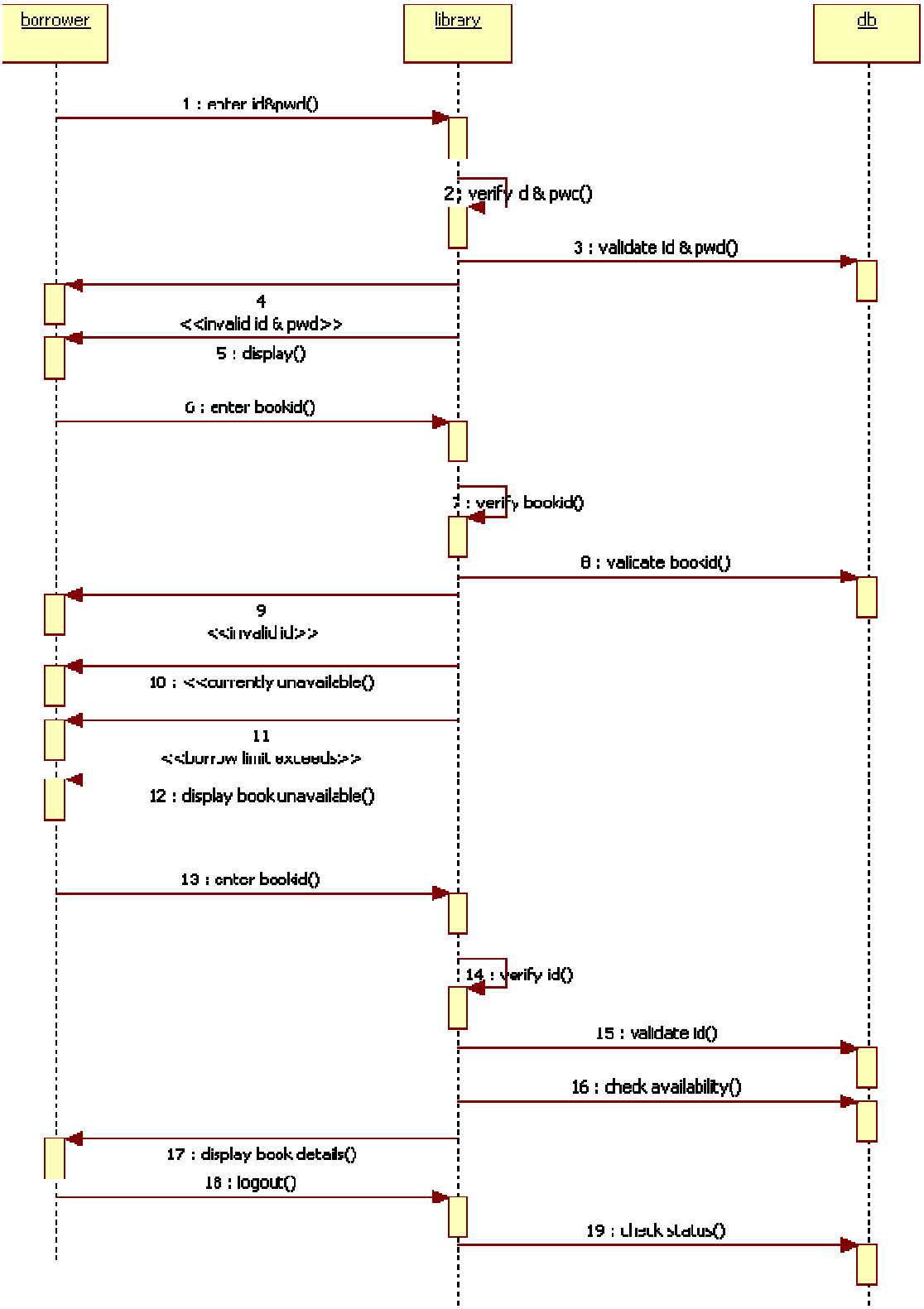


ii. SEQUENCE DIAGRAM

II (a) SEQUENCE DIAGRAM FOR ATM MECHINE

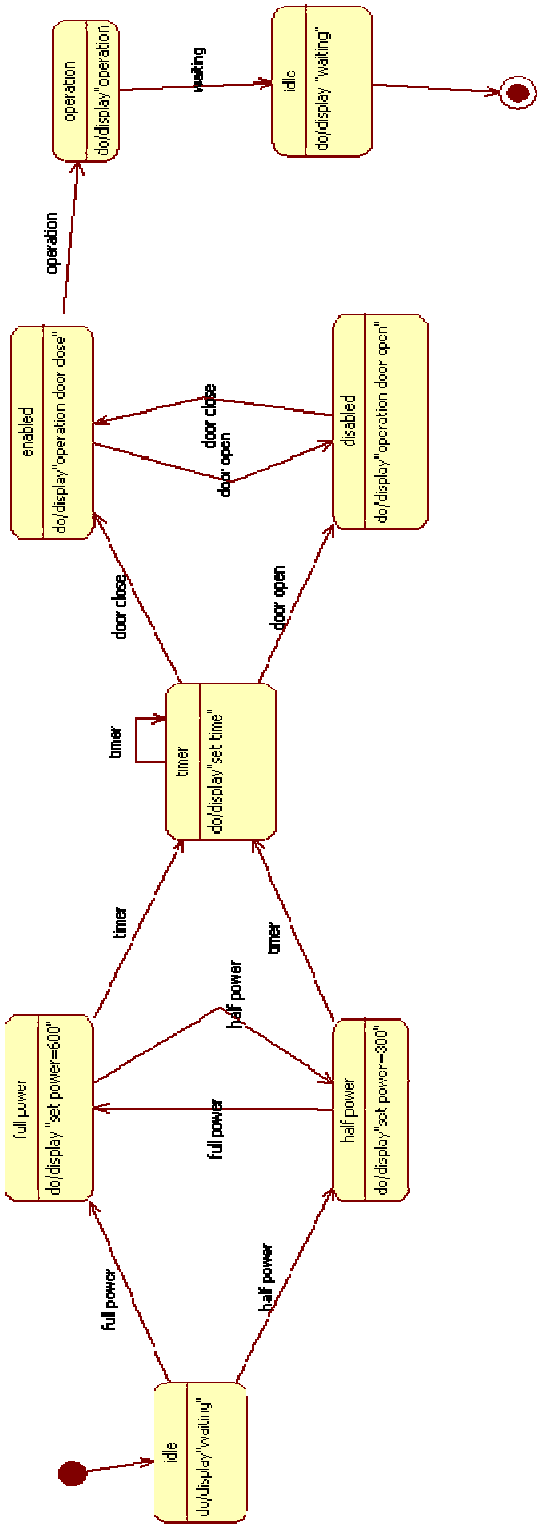


II (b) SEQUENCE DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM



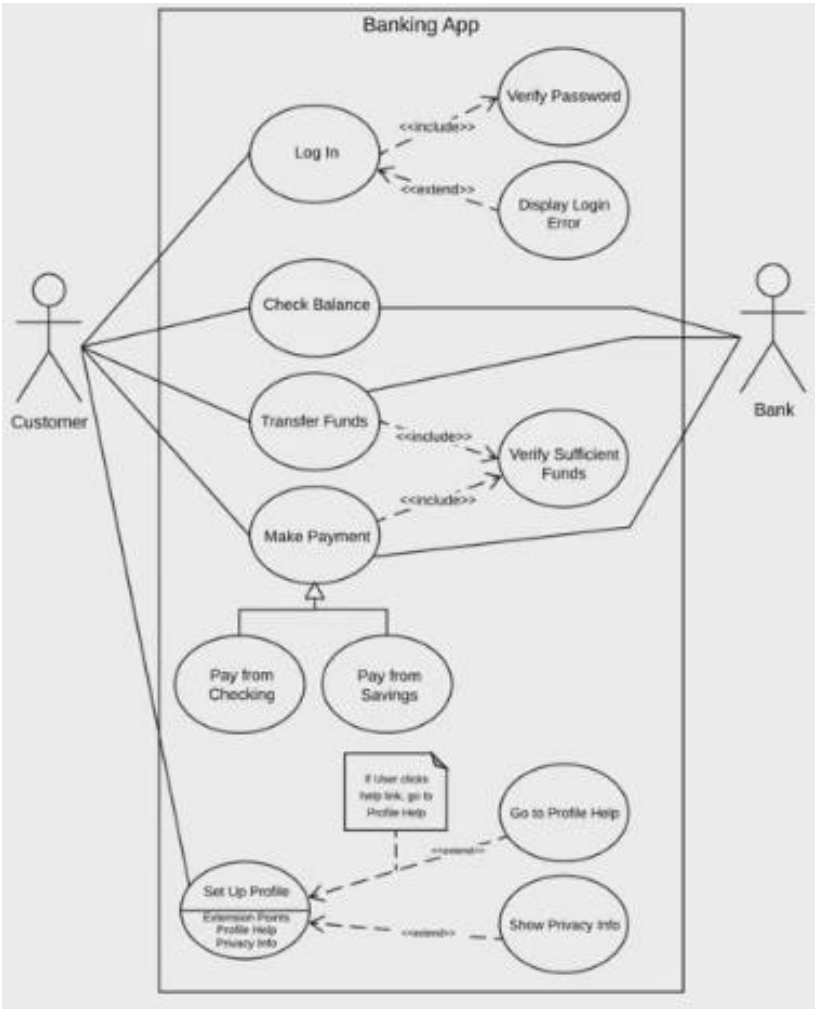
iii. STATE CHART DIAGRAM

III(a) STATE CHART FOR MICRO OVEN



iV. USECASE DIAGRAM

IV(a) USE CASE DIAGRAM FOR BANKING APP



4. Course Management System

Problem Analysis:

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, such as:

An area for **faculty posting** of class **materials** such as course **syllabus** and **handouts** an area for **student posting** of **papers** and other **assignments**

A **grade book** where **faculty** can **record grades** and each **student** can **view** his or her **grades**

An integrated **email tool** allowing **participants** to send announcement **email messages** to the entire **class** or to a subset of the entire class

A **chat tool** allowing **synchronous communication** among class **participants** a threaded discussion board allowing asynchronous communication among participants

In addition, a CMS is typically integrated with other databases in the university so that **students enrolled** in a particular **course** are automatically registered in the CMS as participants in that course.

The Course Management System (CMS) is a **web application** for department personnel, Academic Senate, and **Registrar staff**s to view, enter, and **manage course information** formerly **submitted via paper**.

Departments can use CMS to **create new course** proposals, submit **changes** for **existing courses**, and **track** the **progress** of proposals as they move through the stages of online approval.

Software Requirement Analysis:

Number of Stakeholders in the course management system are 3:

- Students
- faculty
- Departments

General requirements:

Some requirements are shared by all stakeholders

- The system shall provide static course information
- The system shall be able to store static course information
- The system shall be able to represent static course information
- The system shall provide dynamic course information
- The system shall be able to store dynamic course information
- The system shall be able to represent dynamic course information
- The system shall provide a messaging system

Requirements of Students:

Functional requirements:

- The system shall provide the history of attended courses.
- The system shall enable students to subscribe/unsubscribe to courses.
- The system shall enable students to subscribe/unsubscribe to exams.
- The system shall be able to let students upload assignment document files.
- The system shall allow sending messages to individuals, teams or all course participants at once.
- The system shall allow students to create teams.
- The system shall allow students to edit their personal information
- The system shall allow students to change their password
- The system shall provide a password reset function, which resets the password and mails it to the user
- The system shall allow students to view course grade statistics per semester

Non-functional requirements:

Privacy

- The system shall protect the user's privacy
- The system shall prevent students from viewing grades of others
- The system shall provide a user-customizable visibility policy for the personal information

Availability

- The system shall have high availability
- The system shall have downtime at most 4 hours/month
- The system shall have its expected downtime announced at least 48 hours in advance
- The system shall have downtime only during low-intensity hours

User friendliness

- The system will be user friendly
- The system shall have a maximum of 3 clicks to reach any content
- The system shall have a single login to access all content
- The system shall have a consistent UI (in all the views and dialogs, the UI elements behave and are placed in a similar way)

Accessibility

- The system shall have high accessibility
- The system shall be accessible by disabled (blind) users, who should be able to navigate the system and have access to all content and functionality

Security

- The system shall allow only students to change study information of others.
- The system shall allow students to view only their own grade

Interoperability

- The system shall be highly interoperable
- The system shall provide an export to commonly used calendar formats (allowing users to import scheduled lectures into a personal calendar)

Requirements of faculty:

Functional requirements:

- The system shall allow faculty to create courses
- Thesystemshallallow facultyto preparelectureschedules (roster)
- Thesystemshall allow faculty touploadcourse materialfor lectures.
- Thesystemshall allow faculty touploadassignment questions.
- Thesystemshallenablefacultytomanagegrades(insert, update, calculate finalgrade)
- Thesystemshall enablefacultytomailmultiplestudentsat once
- The system shall allow faculty to view all personal information(including pictures) of students in the system
- Thesystemshall allowfacultytomanage course information
- The system shall allow only faculty to manage student teams
- The systemshallenablefacultytocomparegrade statistics with other courses
- The system shall allow faculty to duplicate courses and import materials from other courses into another course, but only from their own courses.

Non-functional requirements:

Security

- The system shall allow faculty tomanage the dynamiccontent visibility (visible for students and faculty, visible for faculty, visibleto self only)
- The system shall allowfaculty to view all grades of all students in the course

Interoperability

- The system shall be able to import BOZ roster information intothecourse roster

Requirements of the Departments:

Functional requirements:

- The system shall allow only the department to manage courses.
- Thesystemshall allow onlythedepartment toappoint faculty to courses.
- Thesystemshall allow only thedepartment to specifythe number of students for a course.
- The system shall allow thedepartment to retrieve all study and personal information of students.
- The system shall allow the department to retrieve all faculty information.
- The system shall allow the department to enter lecturer information.
- The system shall allow the department to calculate grade statistics

- The system shall allow the department to calculate the number of passed students.
- The system shall allow the department to evaluate courses through students by means of a web-survey.
- The system shall automatically synchronize with secondary university systems.
- The system shall not allow users to change information which is contained and maintained by secondary university systems

Non-functional requirements:

Availability

- See Students.

User friendliness

- See Students.

Interoperability

- The system shall be interoperable with secondary university systems.

Extensibility

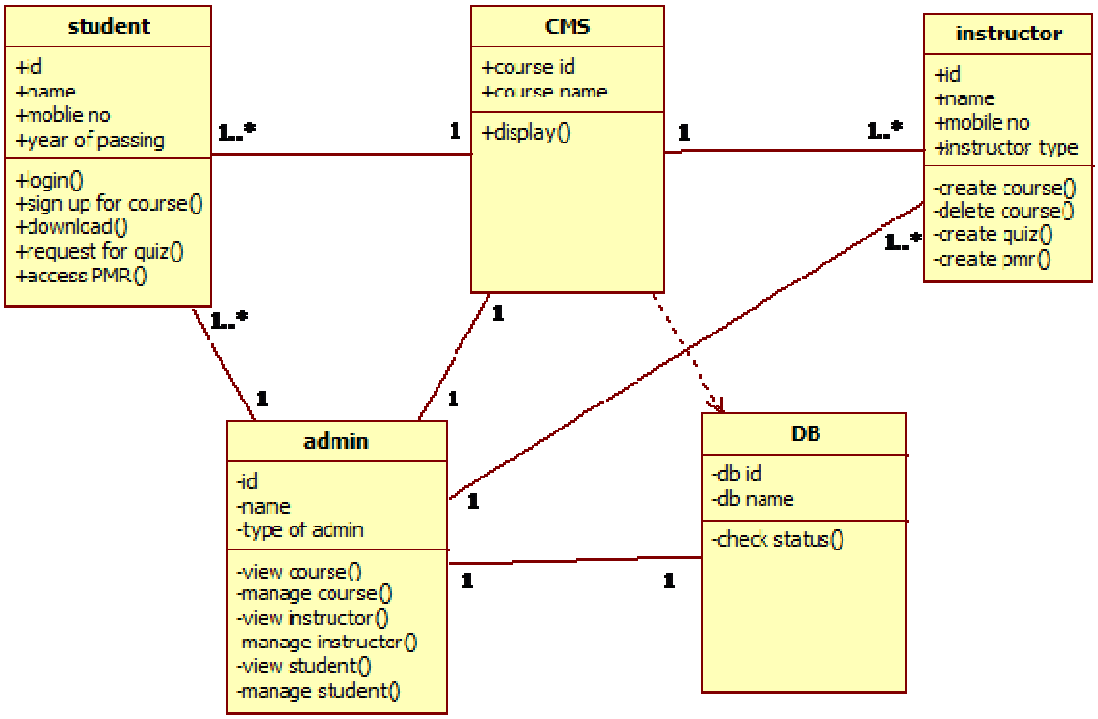
- The system shall allow the department to make exceptions with regard to student enrolment to courses

Data Dictionary:

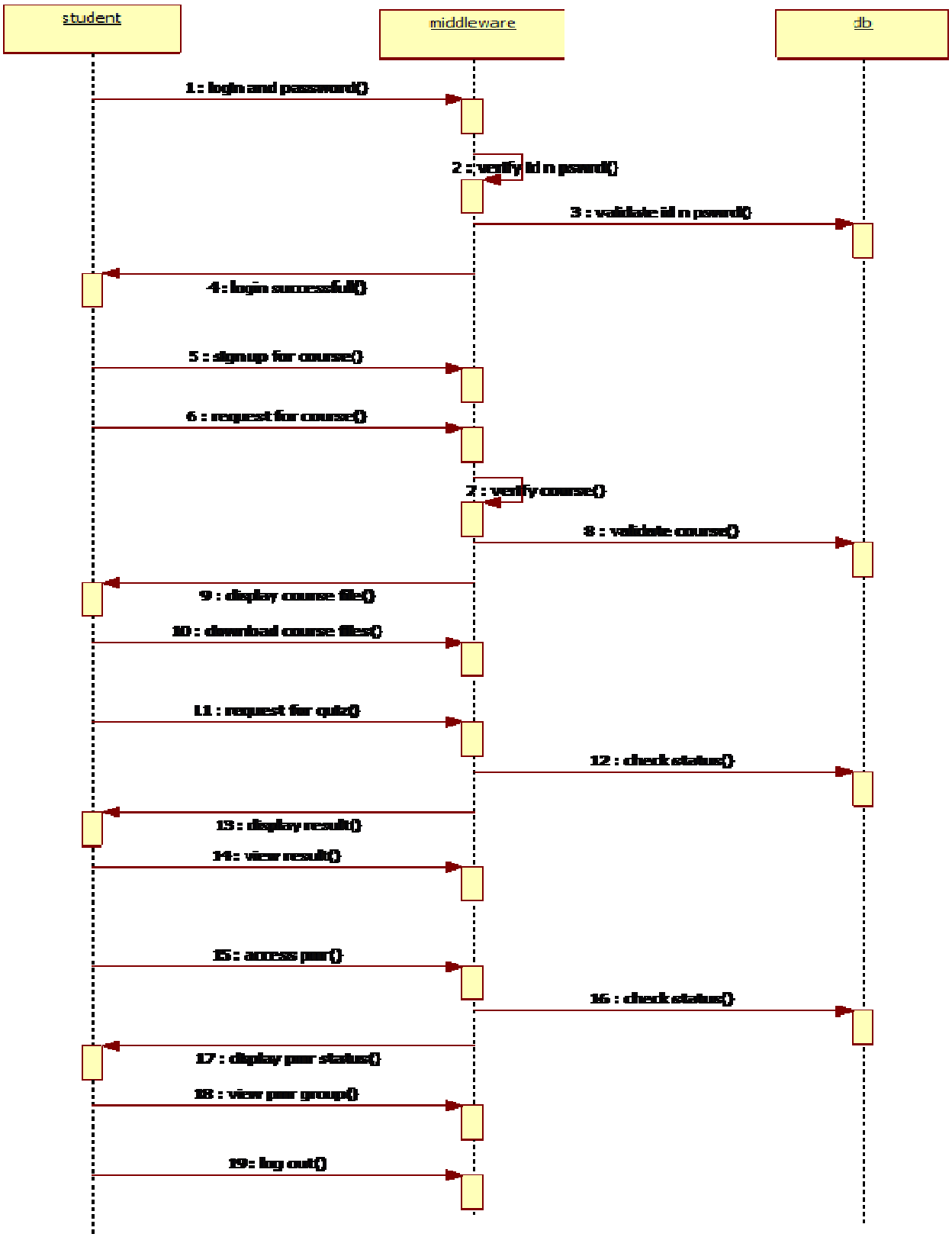
Manage	Managing involves the creation, reading, updating and deleting
PersonalInformation	Information about a person, such as name, address, a picture, interests, etc
Study Information	Information about a person's study progress, such as subscribed courses, grades and exam attempts
CourseInformation	Information of a course which **changes** and **does not change** while a course is given, but between semesters. This includes the faculty, material, assignment etc..
Secondary University Systems	All university systems which are shared by different departments, such as a central address book containing all kinds of personal information

Software Designing:

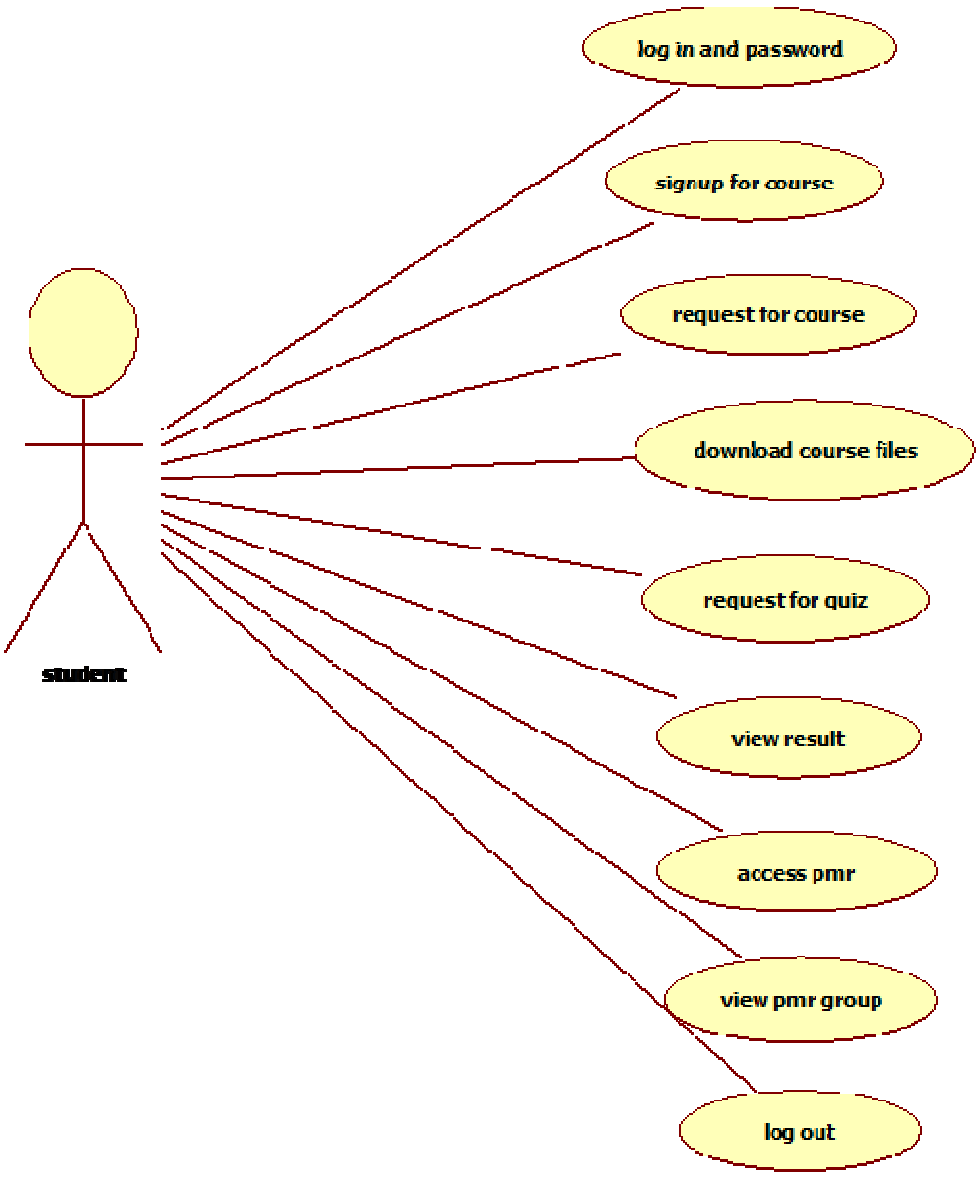
(a) Class diagram



(b) Sequence diagram

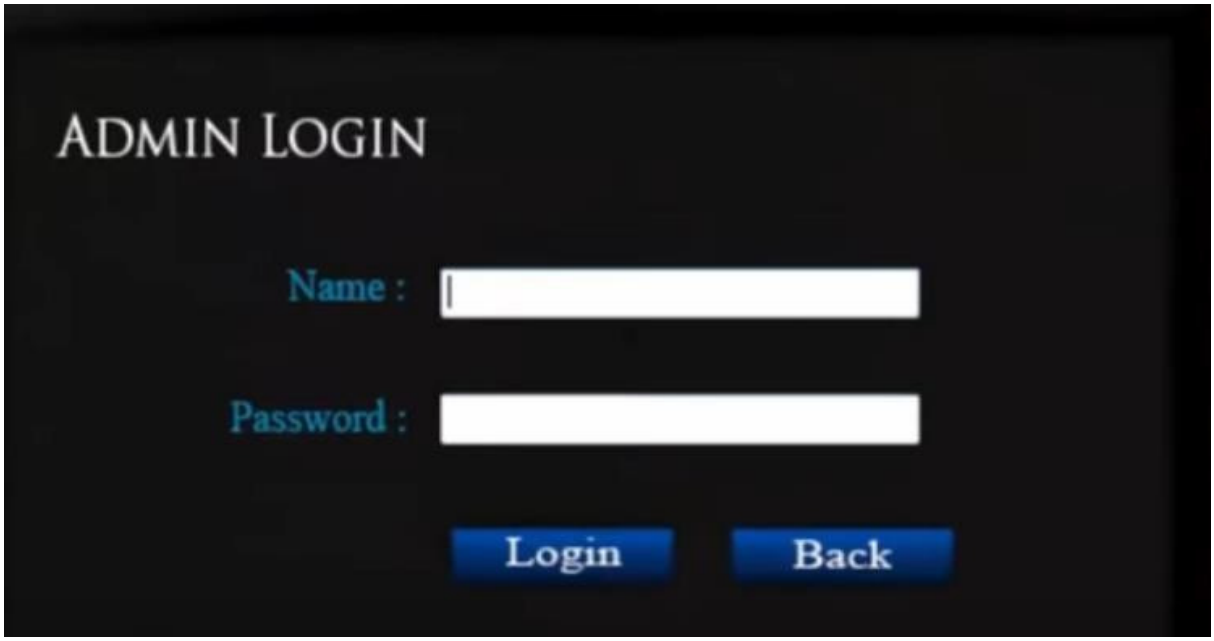


(c) Use case diagram

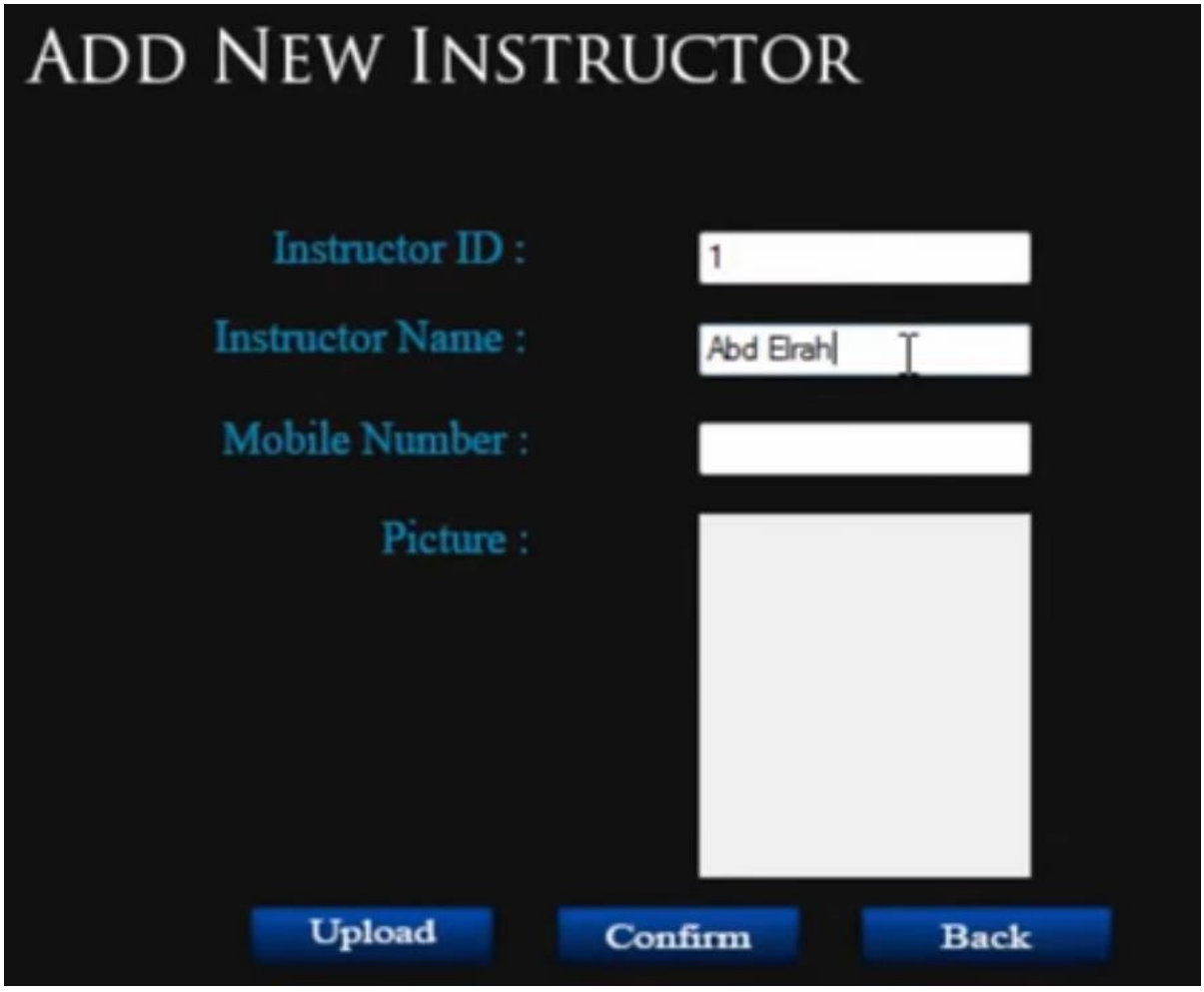


Prototype model:

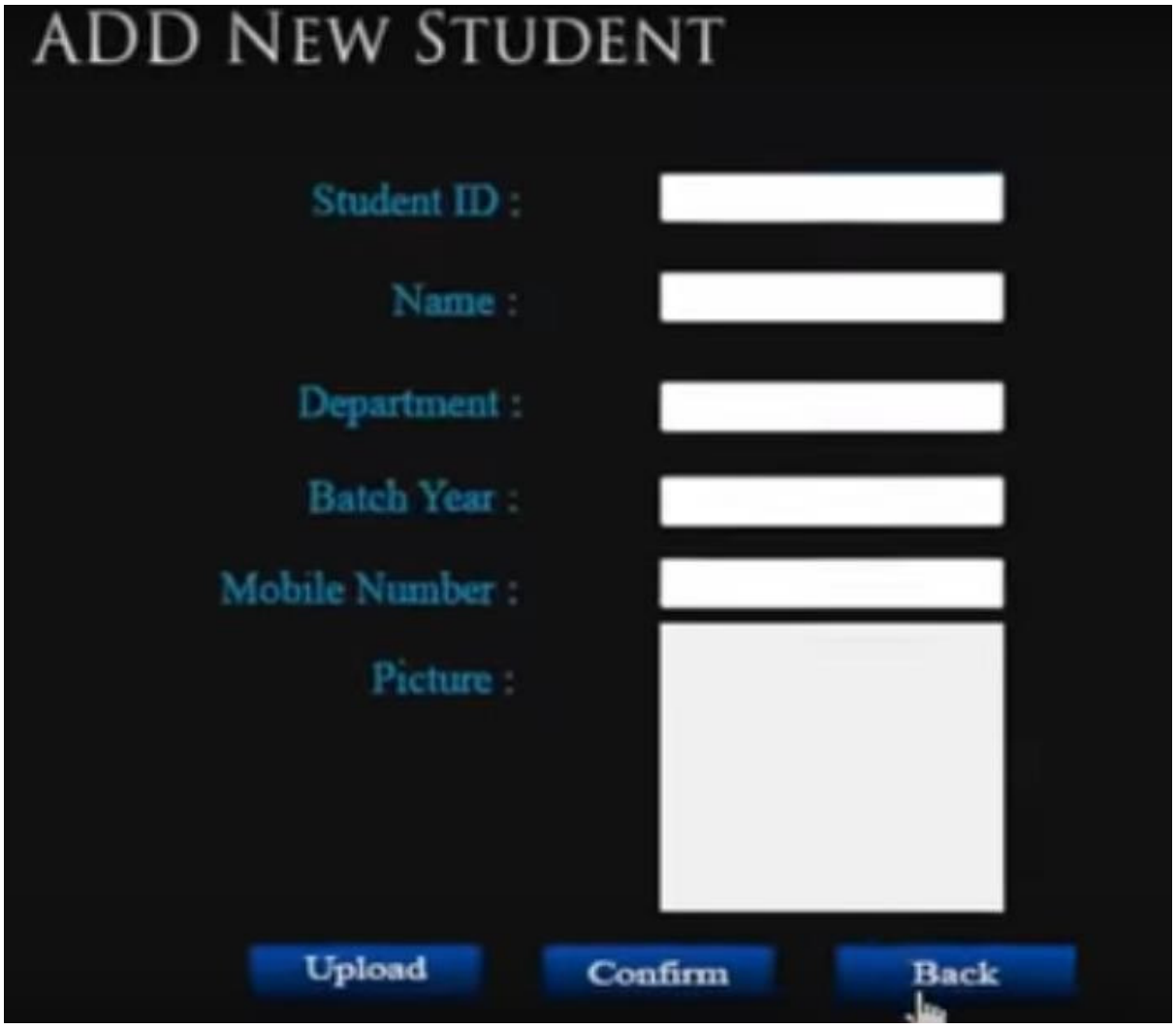
ADMIN LOGIN



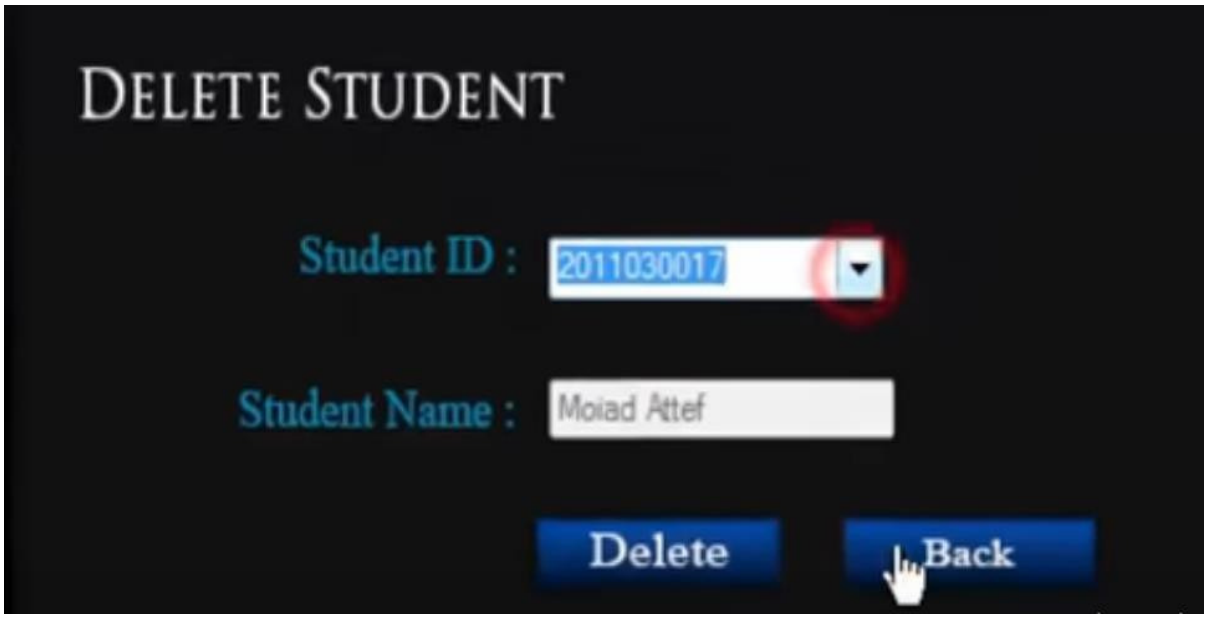
ADD FACULTY



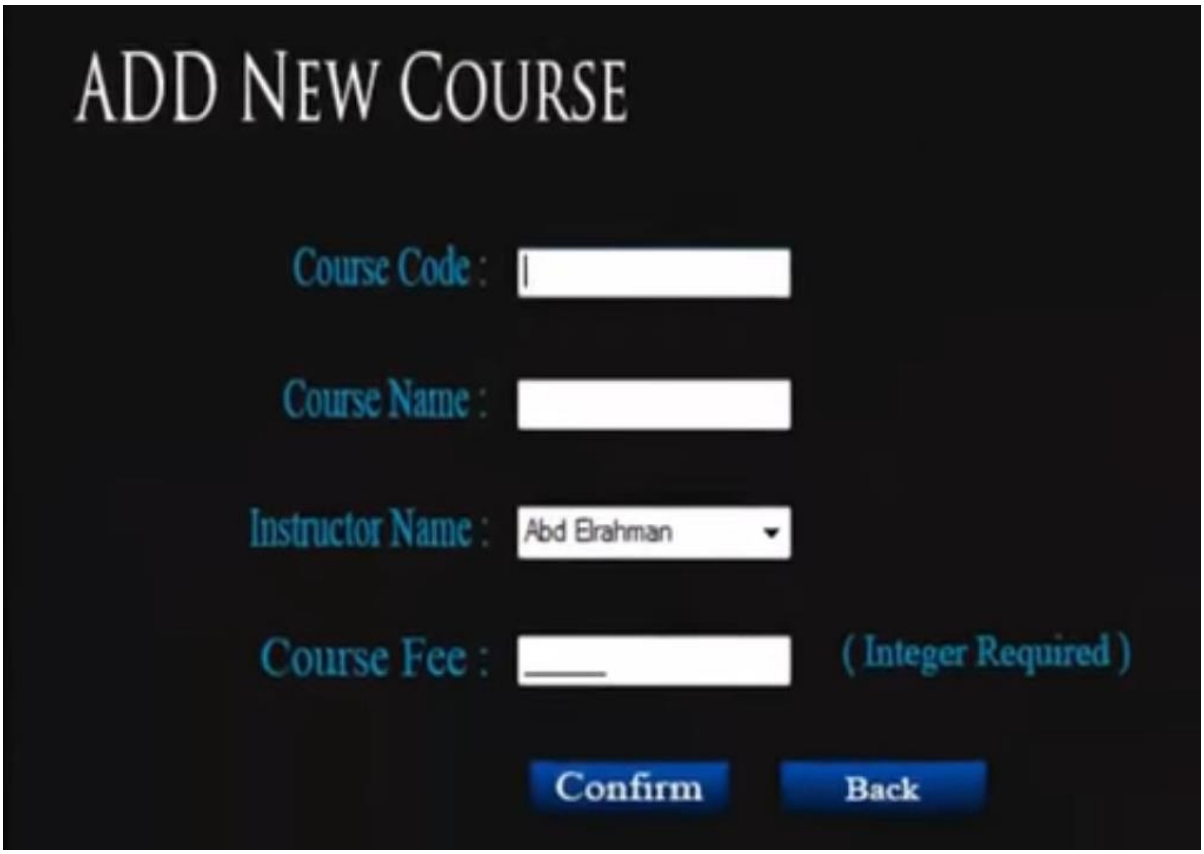
ADD STUDENT



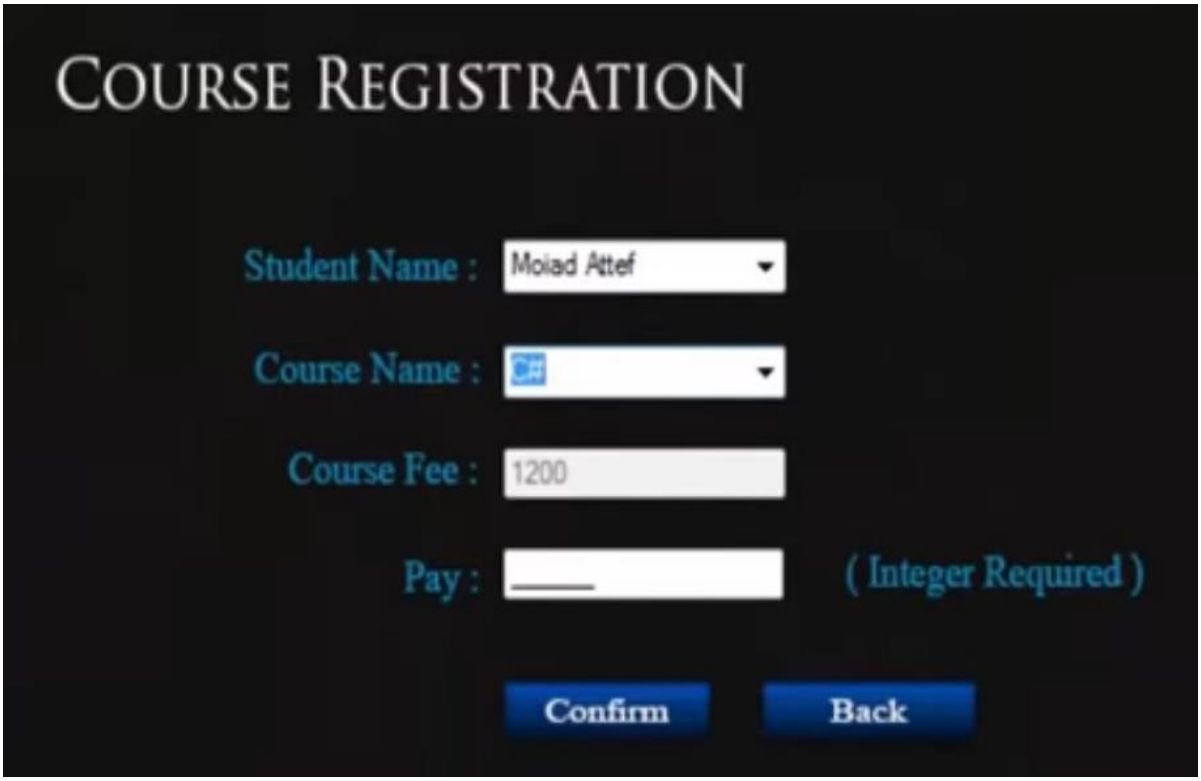
DELETE STUDENT



ADD NEW COURCE



STUDENT COURCE REGISTRATION



STUDENT UPDATING DETAILS

5. Easy Leave

Problem Analysis:

This project is aimed at developing a **web based Leave Management Tool**, which is of importance to either an **organization or a college**.

The **Easy Leave** is an **Intranet** based application that can be accessed throughout the **organization or a specified group/Dept**. This system can be used to **automate** the workflow of **leave applications** and their approvals. The periodic crediting of leave is also automated. There are features like notifications, cancellation of leave, automatic approval of leave, report generators etc in this Tool.

Functional components of the project:

There are **registered people** in the system. Some are approvers. An approver can also be a requestor. In an organization, the **hierarchy** could be **Engineers/Managers/Business Managers/Managing Director** etc. In a **college**, it could be **Lecturer/Professor/Head of the Department/Dean/Principal** etc.

Existing system:

In the existing paper work related to leave management, leaves are maintained using the attendance register for staff. The staff needs to submit their leaves manually to their respective authorities. This increases the paperwork & maintaining the records becomes tedious. Maintaining notices in the records also increases the paperwork.

Issues in Existing System:

It does not ensure security of every record. It increases the redundancy of data and gives various facilities .It leads to loss of data .The staff has to write a letter to its superior for leave which makes it a tedious work for the staff.

Proposed System:

To automate the existing leave management in educational institutes To decrease the paperwork and enable the process with efficient, reliable record maintenance by using centralized database, thereby reducing chances of data loss . To provide for an automated leave management system that intelligently adapts to HR policy of the organization and allows employees and their line managers to manage leaves and replacements for better scheduling of work load & processes.

Software Requirement Analysis:

Number of Stakeholders in the course management system is 2:

- Employee
- Admin

  Types of functionalities in Employee are 2:
  - ✓ General User
  - ✓ Superior User

Requirements of Employee:

General User functinalities:

Functional Requirements:

- Login to the system through the first page of the application.
- Change the password after logging into the system.
- See his/her eligibility details (like how many days of leave he/she is eligible for etc).
- Query the leave balance.
- See his/her leave history since the time he/she joined the company/college.
- Apply for leave, specifying from and to dates, reason for taking leave, and address for communication while on leave and his/her superior's email id.
- See his/her current leave applications and the leave applications that are submitted to him/her for approval or cancellation.
- Withdraw his/her leave application (which has not been approved yet).
- As soon as any operation made by the employee, an automatic email should be sent to the Employee mail id giving details about the action.
- The number of days of leave (as per the assumed leave policy) should be automatically credited to everybody and a notification regarding the same be sent to them automatically.
- An automatic leave-approval facility for leave applications which are older than 2 weeks should be there.

Non functional Requirements:

Understandability

- Get help about the leave system on how to use the different features of the system.

Security

- In this user unable to approve the leaves.
- Unable to see the other employee leave details.

Superior User Functionalities:

Functional Requirements:

- Approve/reject the leave applications that are submitted to him/her.
- Cancel his/her leave (which has been already approved).
- As soon as any operation made by the subordinate, an automatic email should be sent to the superior mail id giving details about the action.
- Need to act as normal user for his/her superiors and act as superior for his/her subordinates.
- Display subordinate details.

Non functional Requirements:

Security

- Able to see only his/her subordinate employee leave details.
- Cannot access same category employee details.

Understandability

- Get help about the leave system on how to use the different features of the system.

Requirements of Admin:

Functional Requirements:

- Create the users.
- Mention the user designation.
- Create designation hierarchy (According to this superior and subordinates will be declared)

Non Functional Requirements:

Extendibility:

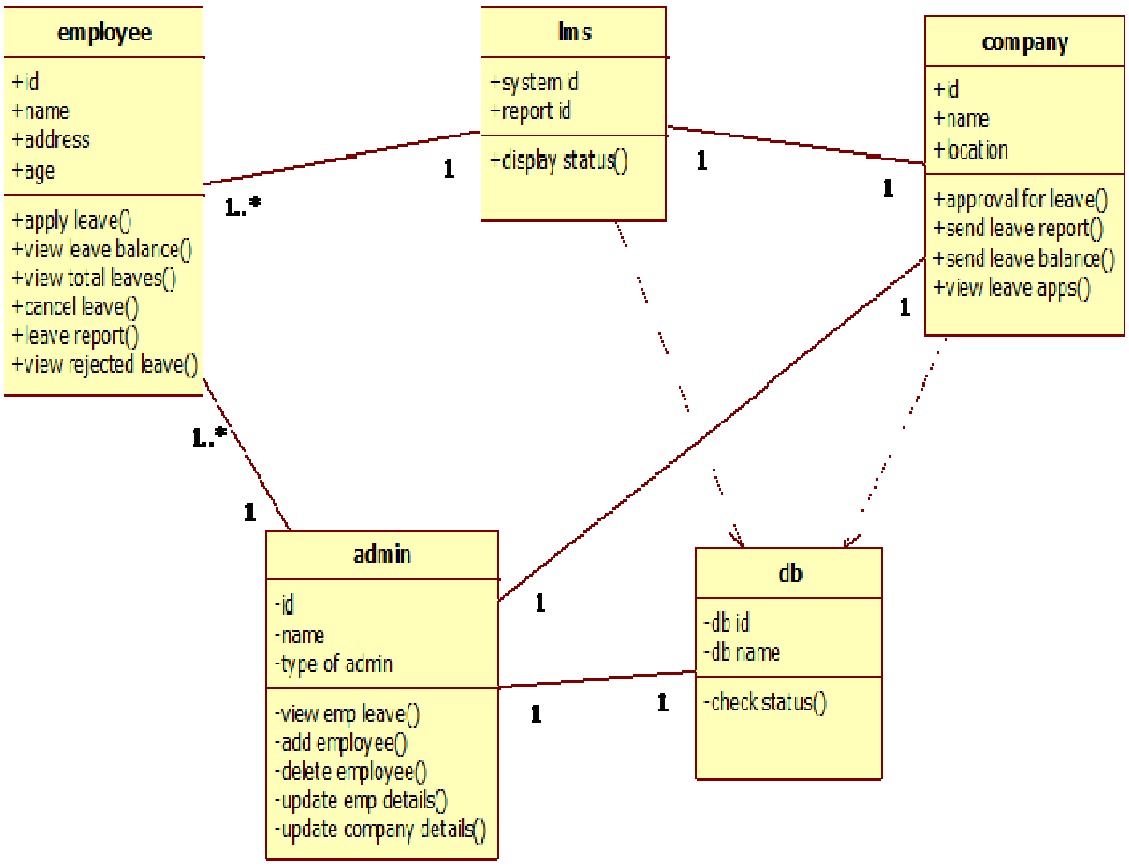
- Can create any number of users.

Maintainability:

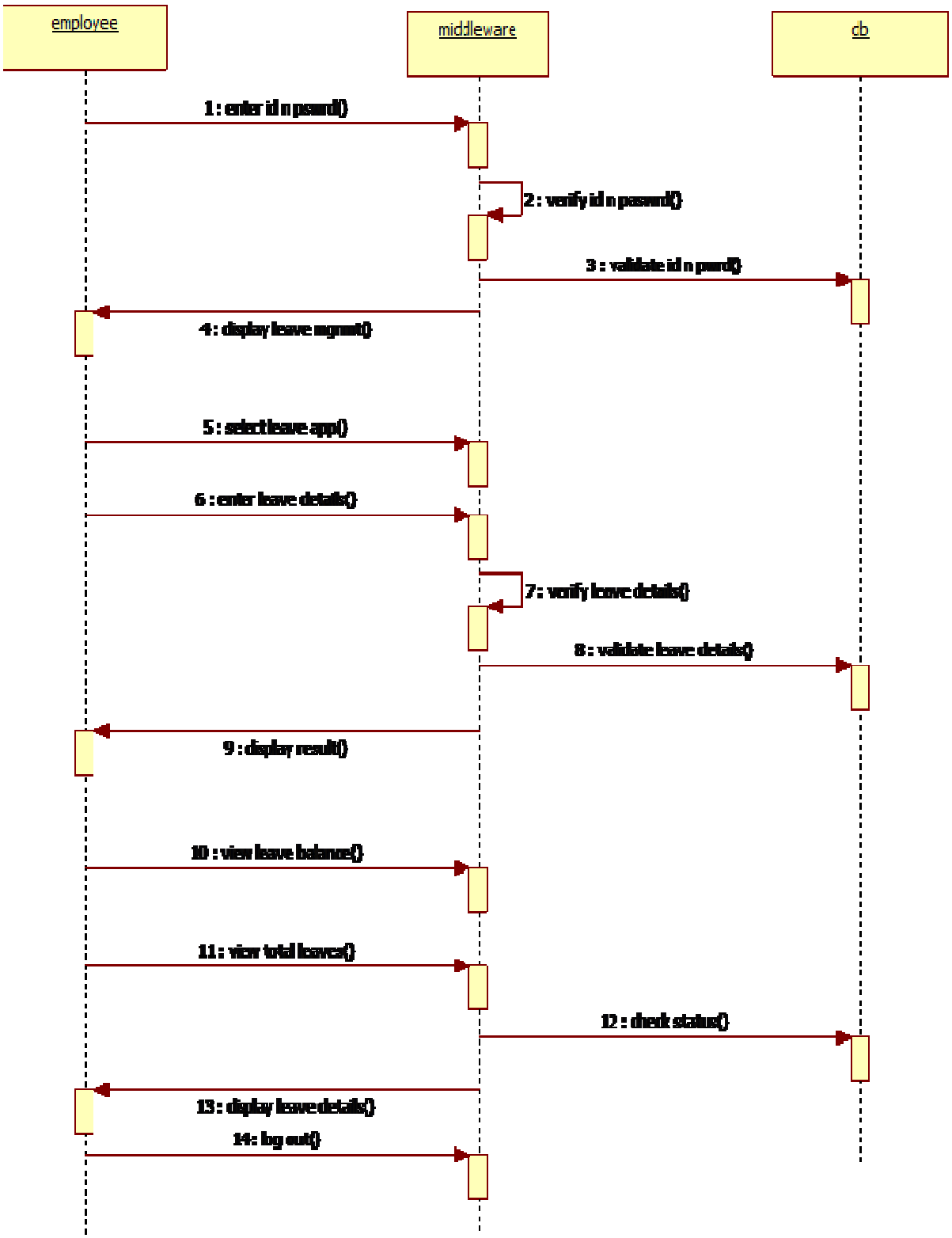
- Has to maintain the subordinate and superior data perfectly.

Software Designing:

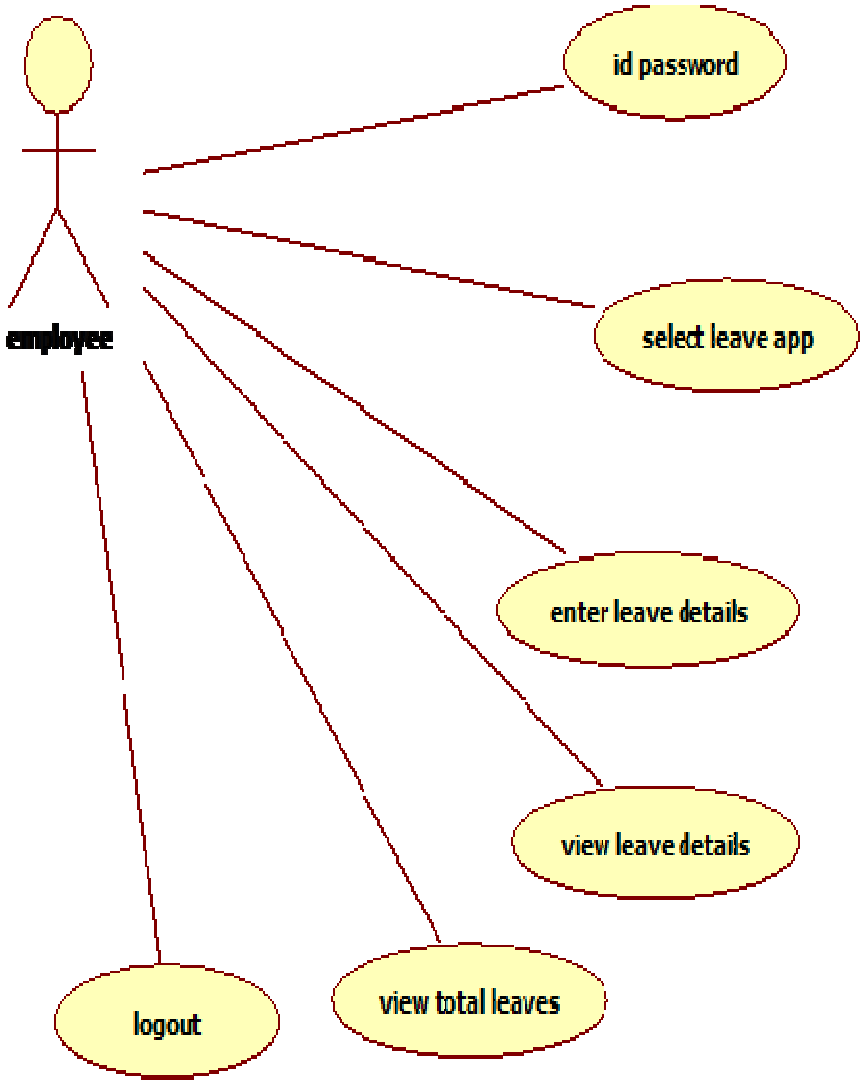
(a) Class diagram



(b) Sequence diagram

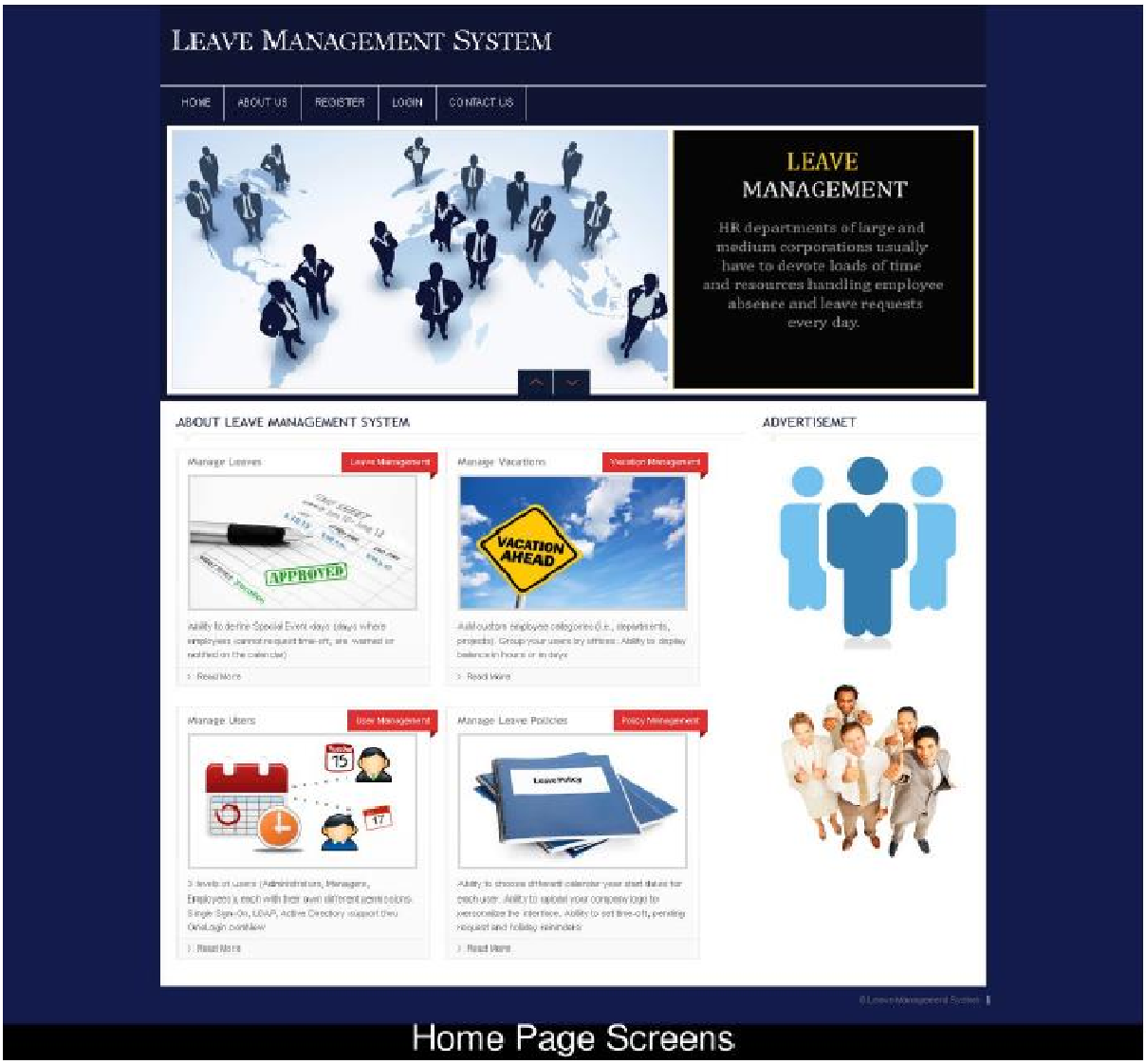


(c) Use case diagram



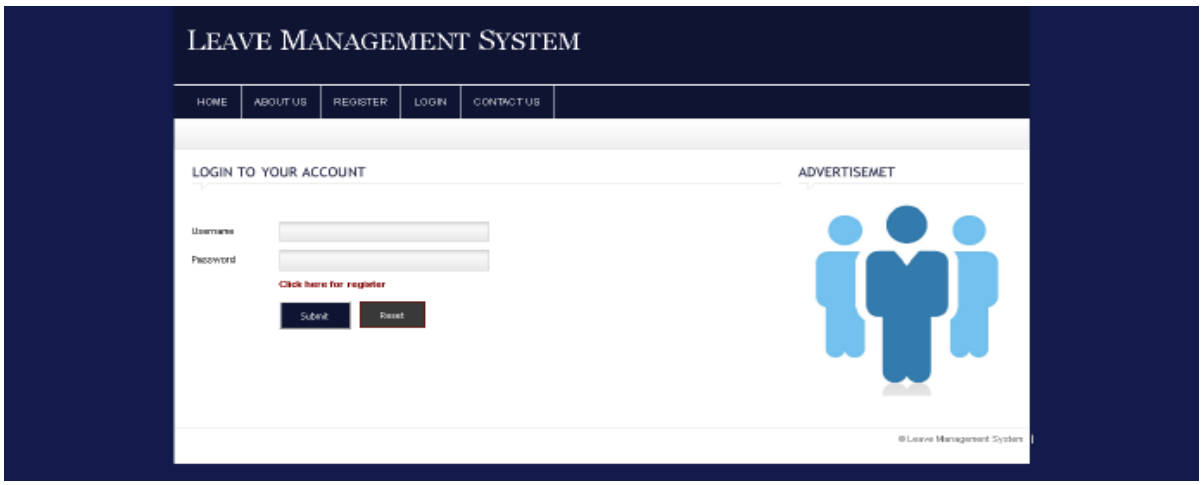
Prototype model:

Home Page

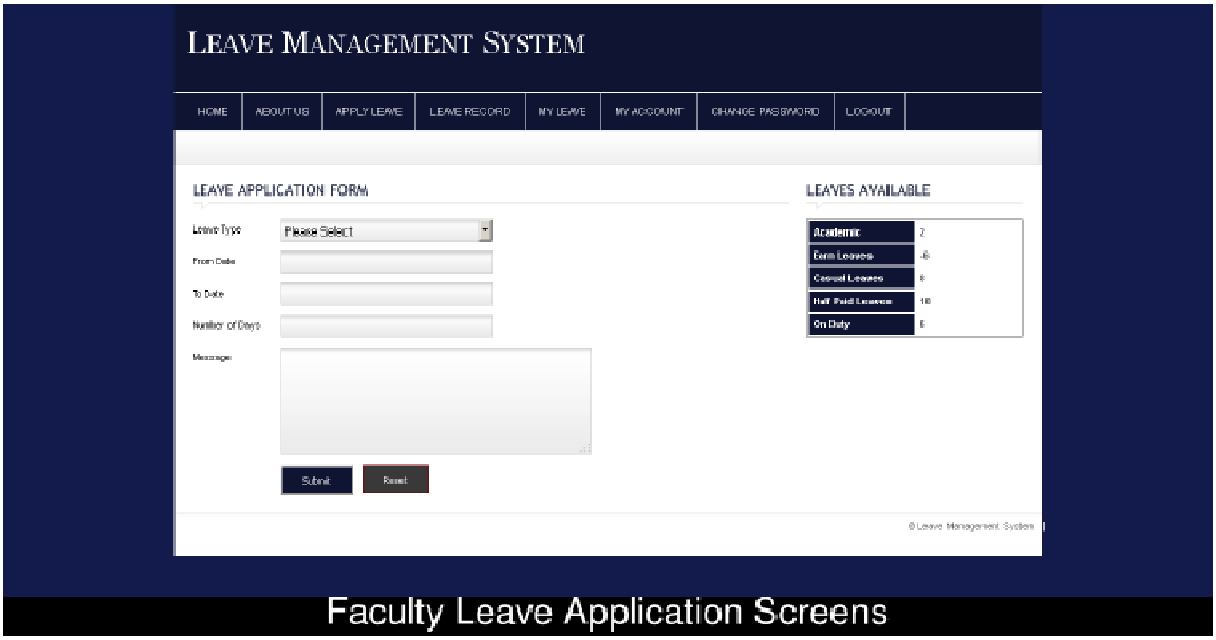


Home Page Screens

Login Page

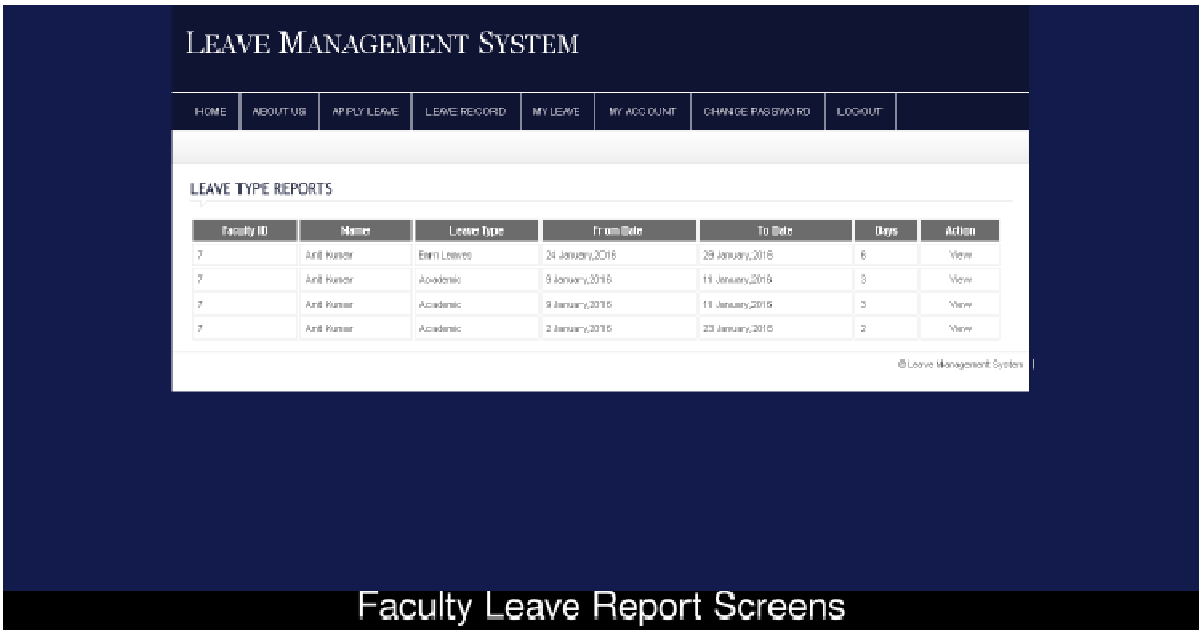


Faculty leave Apply Page



Faculty Leave Application Screens

Faculty leave Report Page

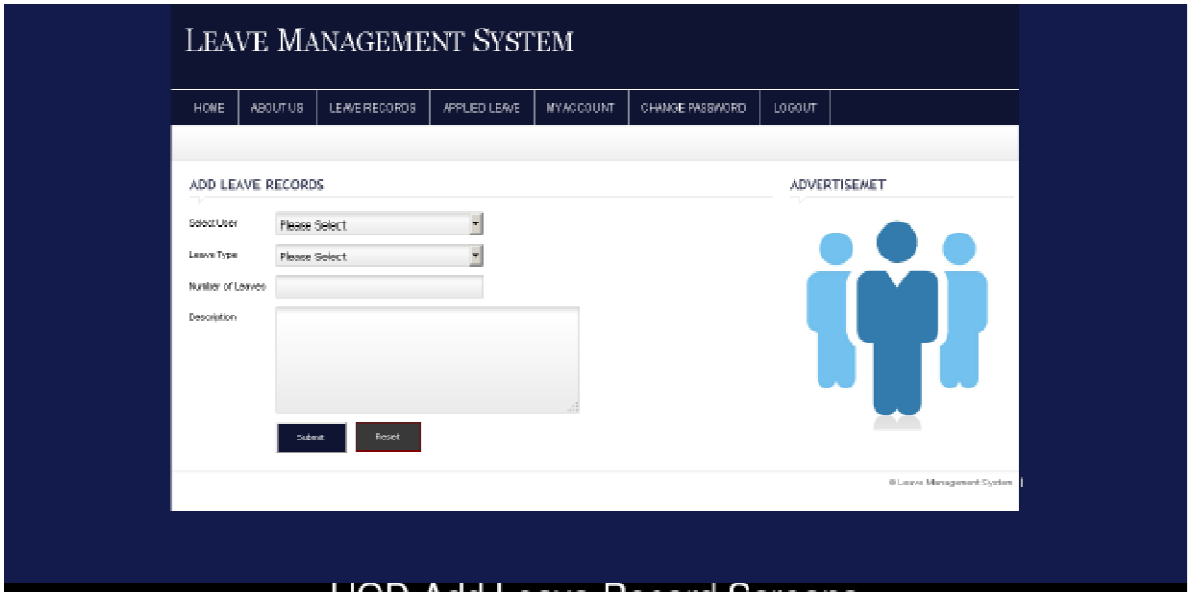


Faculty Leave Report Screens

Admin leave Page

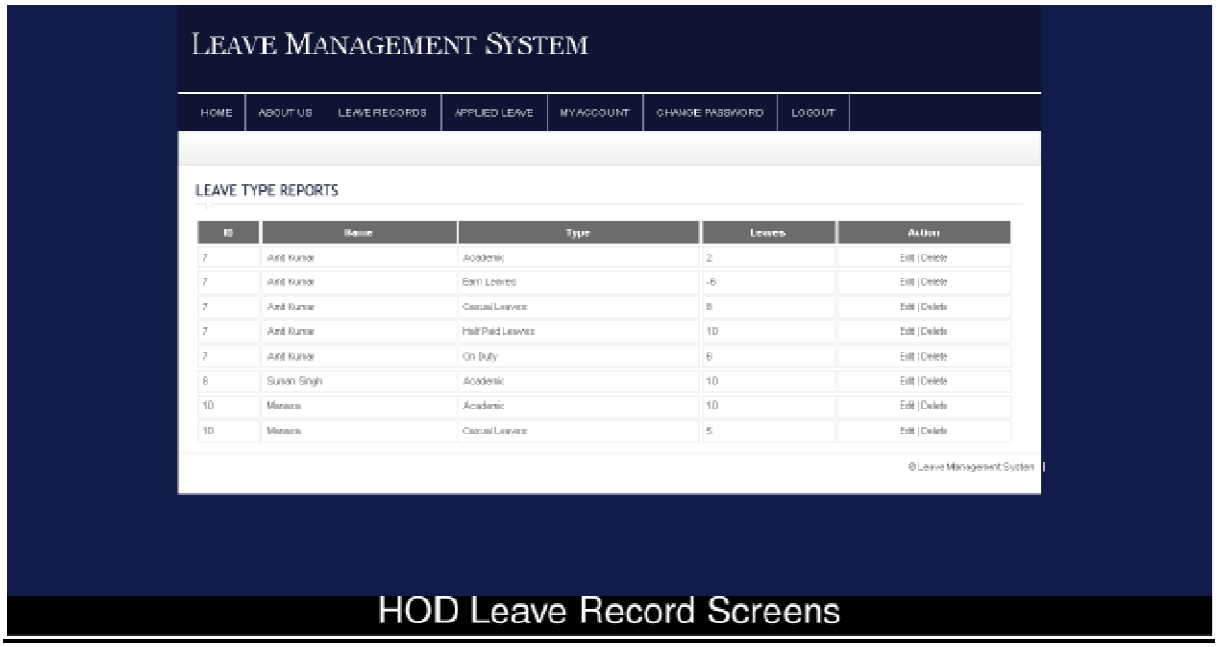
Admin Add Leave Type Screens

Superior Leave Page



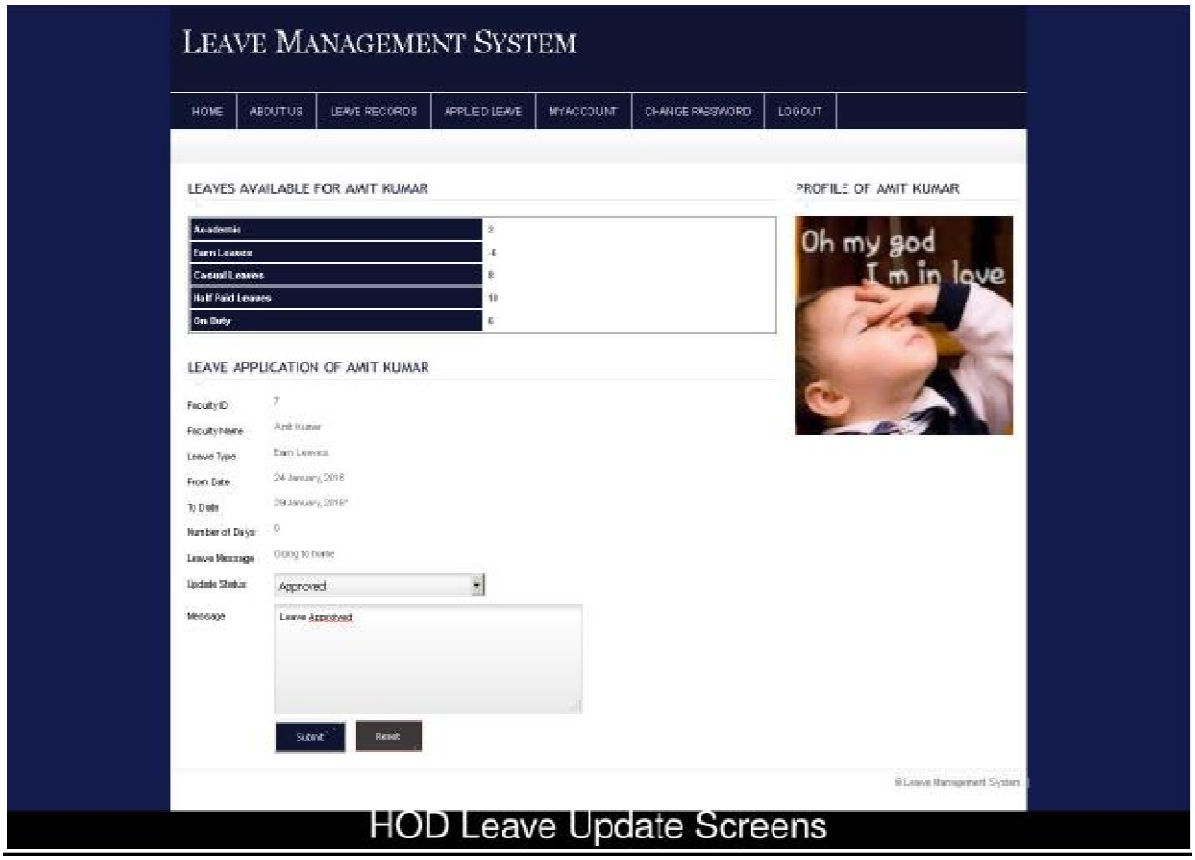
HOD Add Leave Record Screens

Superior Leave Record Page



HOD Leave Record Screens

Superior Leave Approve Page



HOD Leave Update Screens

6. E-Bidding

Problem Analysis:

Auctions are to **sell** a wide variety of **goods**. **Sealed-bid** auctions do **not achieve** efficient **allocations** in general since they do **not allow** the **information** held by **different bidders** to be **shared**.

Typically, in an **auction**, say of the kind used to sell art, the **auctioneer sets** a relatively low **initial price**. This **price** is then **increased until** only one **bidder** is willing to **buy** the object, and the exact manner in which this is done varies. In my model a **bidder** who **drops** out at some price can **reenter at a higher price**.

Proposed system:

- To generate the quick reports.
- To make accuracy and efficient calculations to provide proper information briefly.
- To provide data security.
- To provide huge maintenance of records Flexibility of transactions can be completed in time.

Software Requirement Analysis:

Number of Stakeholders in the course management system is 3:

- Bidder
- Auctioneer
- Admin

General requirements:

Some requirements are shared by **Bidder** and **Auctioneer** stakeholders

- Need to register him/her as a Bidder/auctioneer by providing Personal details.
- Set password at the time of registration.
- Login into the system using email id and password.
- Edit the personal information.
- Can register as both a Bidder and auctioneer.
- Users can edit their profiles.

Requirements for Bidder:

Functional requirements:

- Users can look for a product from a selected category.
- Can bid his/her amount for desired products.
- Can change the bid value.
- Bidder can Drop at any time.
- Bidder can return into the auction at higher price only.
- Final Bid report should be sent to the all Bidders mail ids involved in the action.
- Need to provide the money transaction facility if his/her bid is accepted.

Non Functional Requirements:

Security

- Allowed to see the present bid value only. Not allowed to see the bidders list.
- Not allowed to access other user's data.
- Provide secure e-commerce facility for money transactions.

Reliability:

- Provide good support for money transaction failures.

Usability:

- Easy to understand by the any type of people.
- Provide easy method for bidding.

Requirements for Auctioneer:

Functional requirements:

- Can update his/her product page.
- Can insert a new product with details and can update the product information through edit option.
- Can see their products and bided list through their account page.
- Able to see the bidder's information.
- She/he can start the bidding for own product.
- Able to close the Bidding at any point of time.
- Final Bid report should be sent to the all Bidders mail ids involved in the action.
- Main the bank details for money transaction.

Non Functional Requirements:

Security

- Not allowed to do changes to other user data.
- Not allowed to bid for the product he/she auctioned.
- Secure transactional details.
- Production deletion option not allowed after bidding start.

Requirements for Admin:

Functional requirements:

- Admin can update all product pages.
- Admin can delete the product from the auctioneer List.
- Admin can delete user from user panel.
- Admin can have access in the bid page.
- Admin allowed stop or cancel the bid.
- Admin can delete user accounts.
- Admin can start or close the bid.

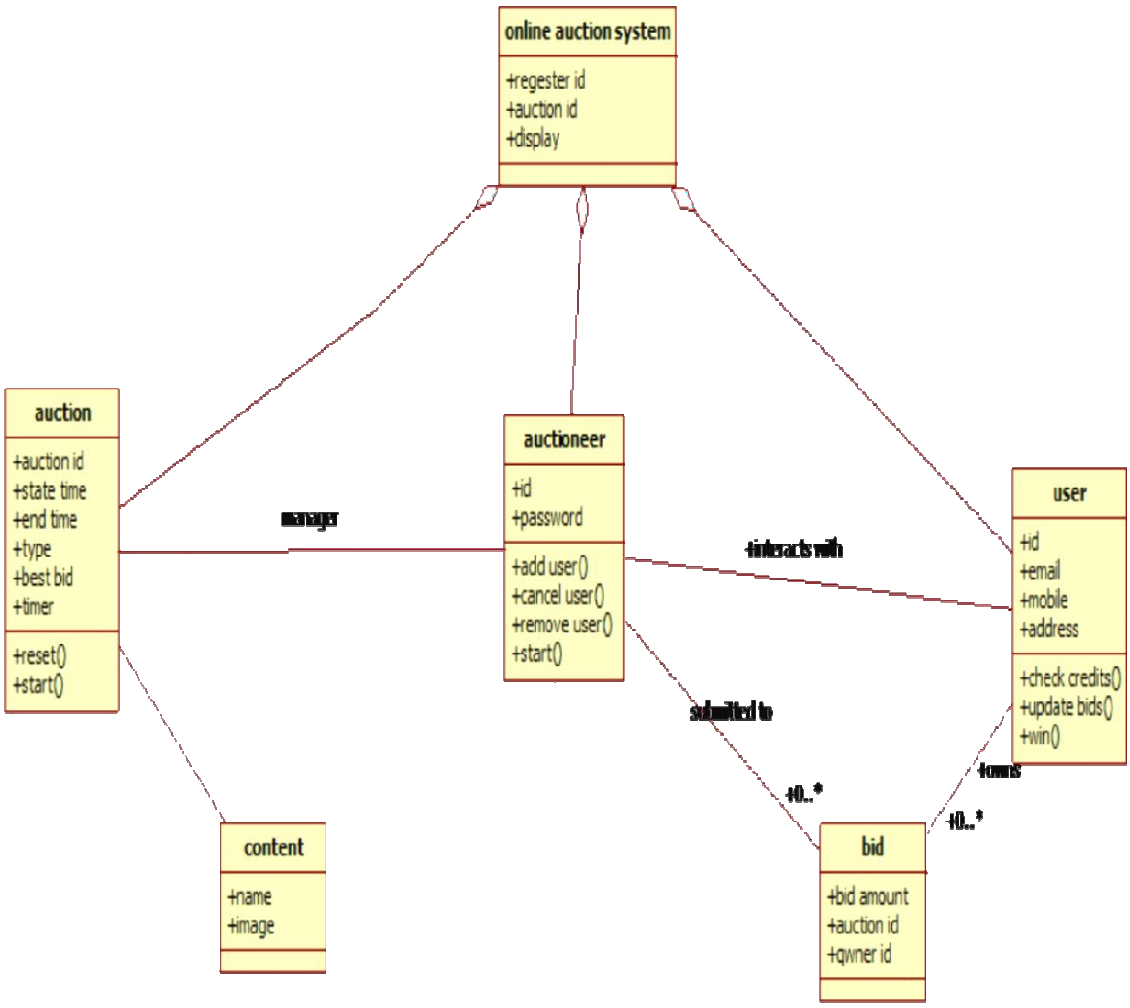
Non Functional Requirements:

Efficiency

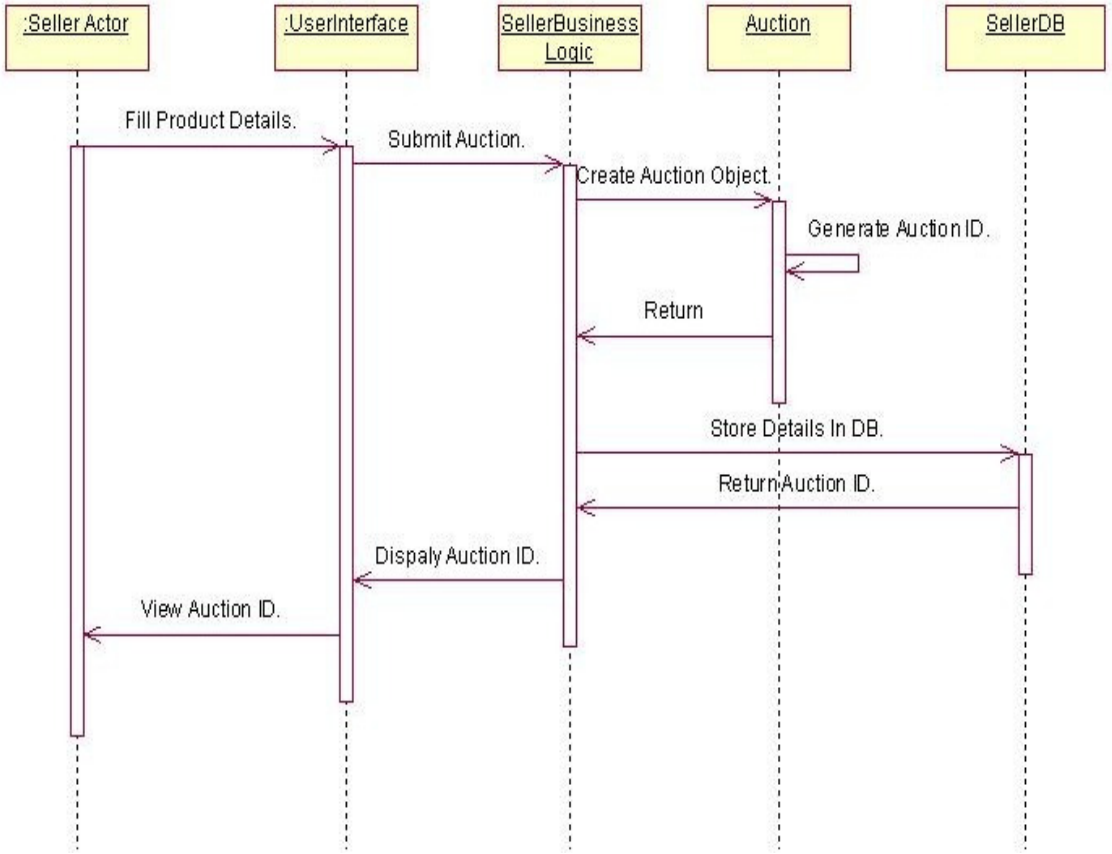
- Auction updates should be accurate.

Software Designing:

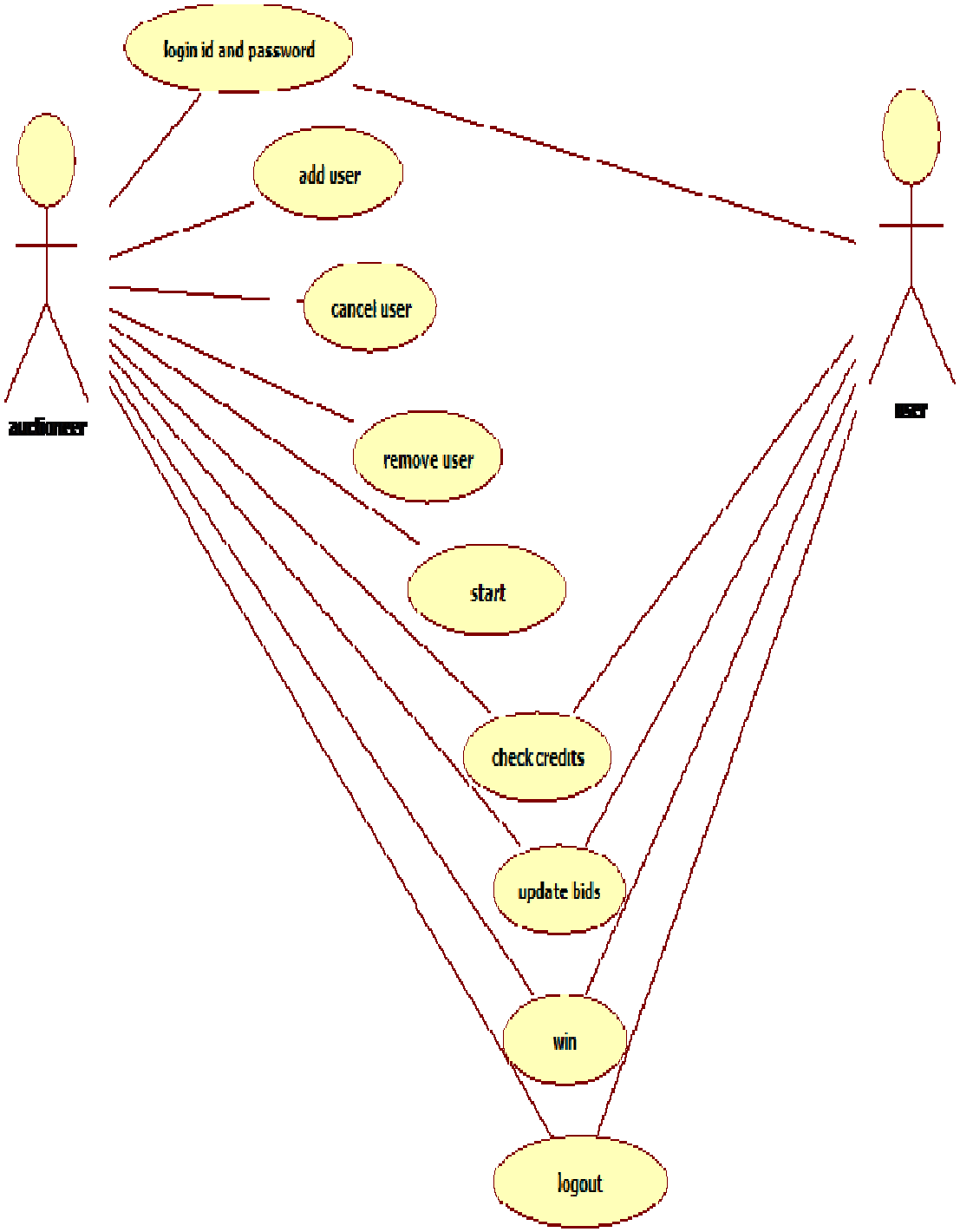
(a) Class diagram



(b) Sequence diagram

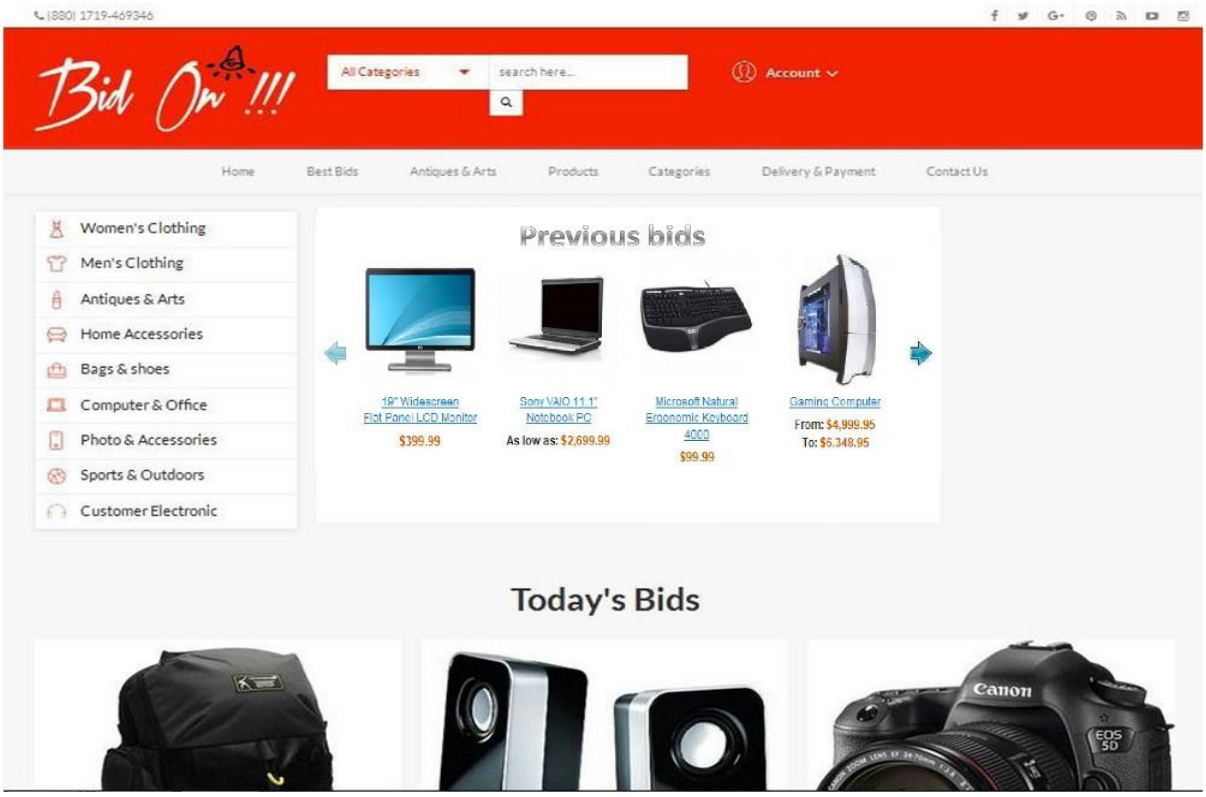


(c) Use case diagram

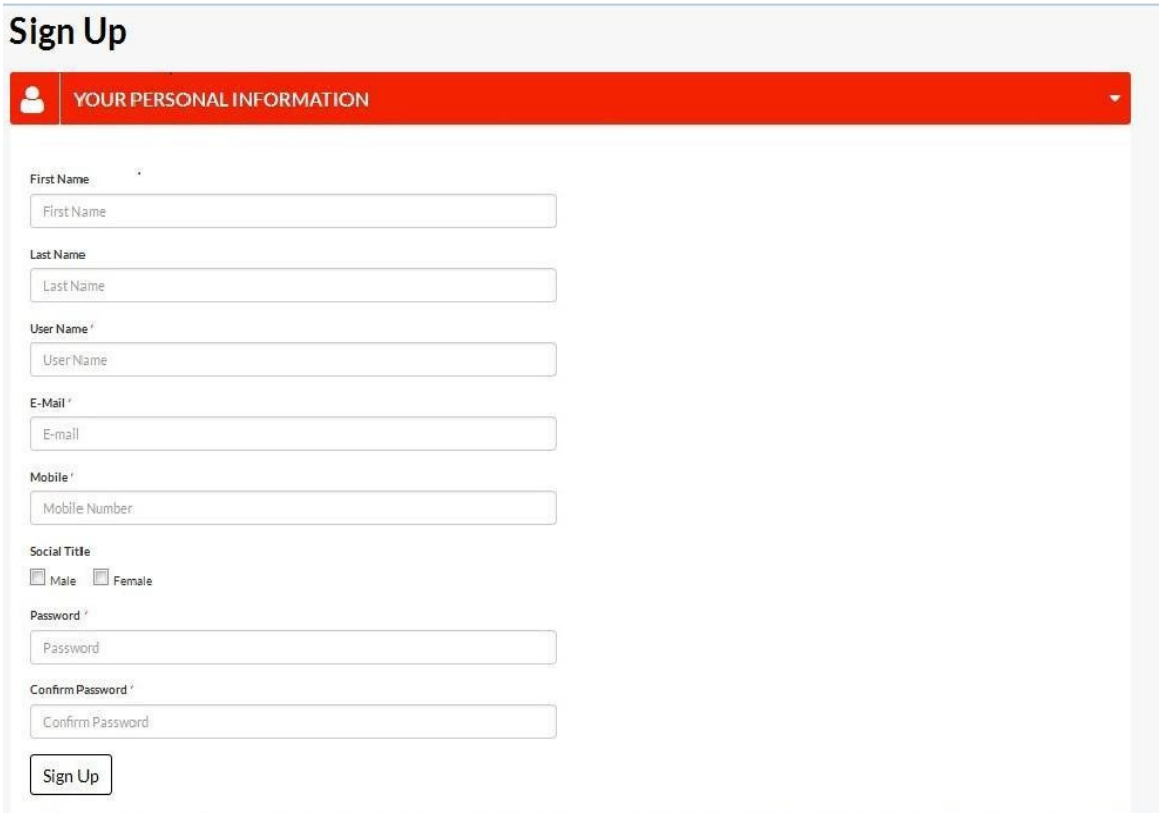


Prototype model:

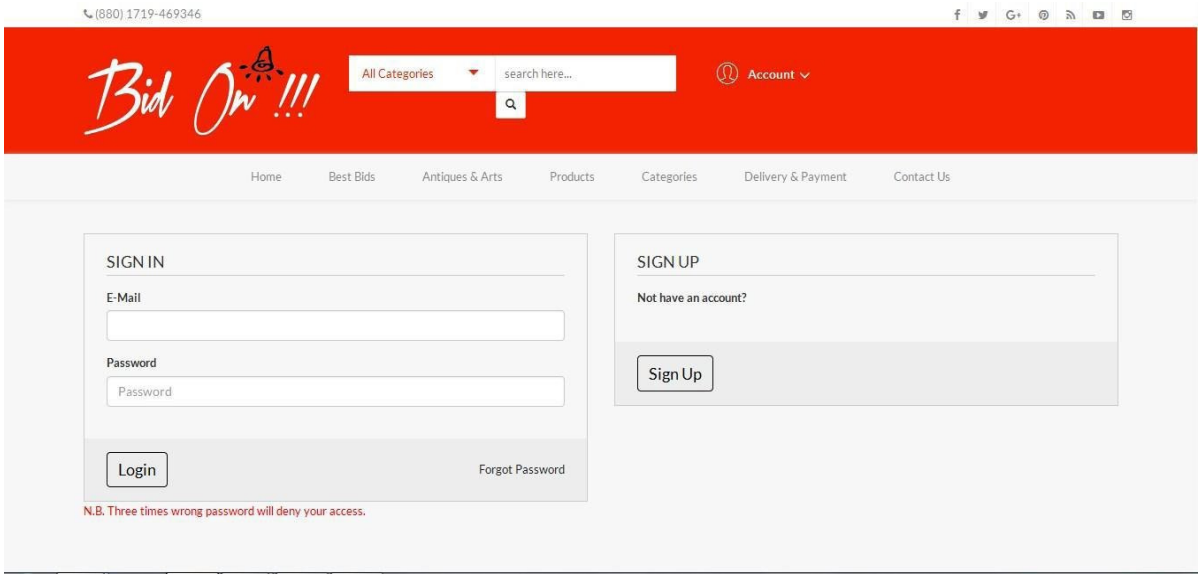
Home Page



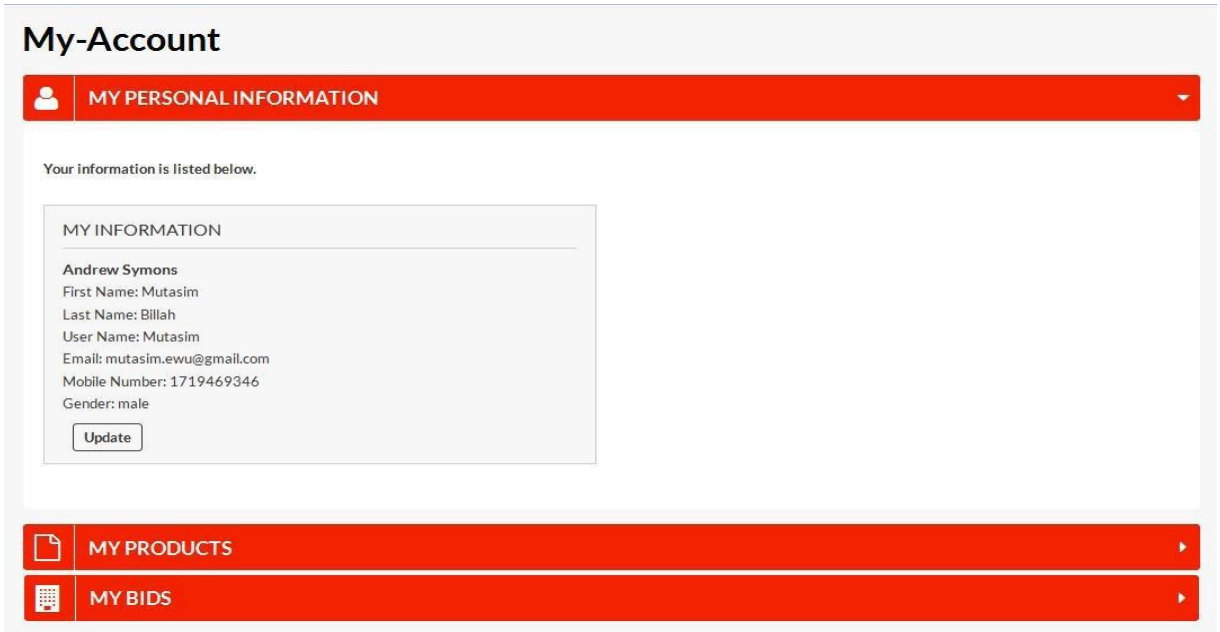
Sign Up Page



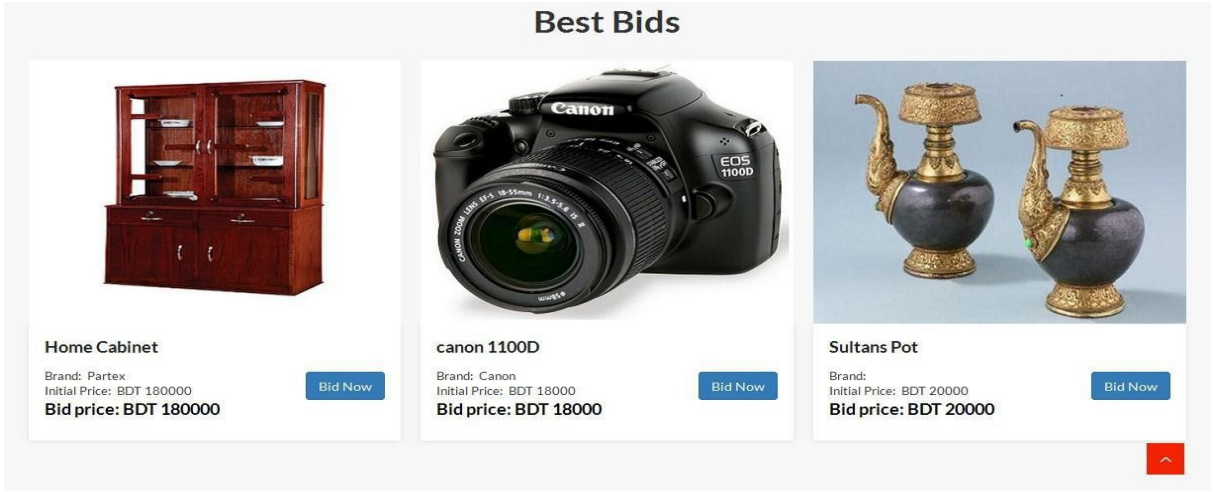
Sign in page



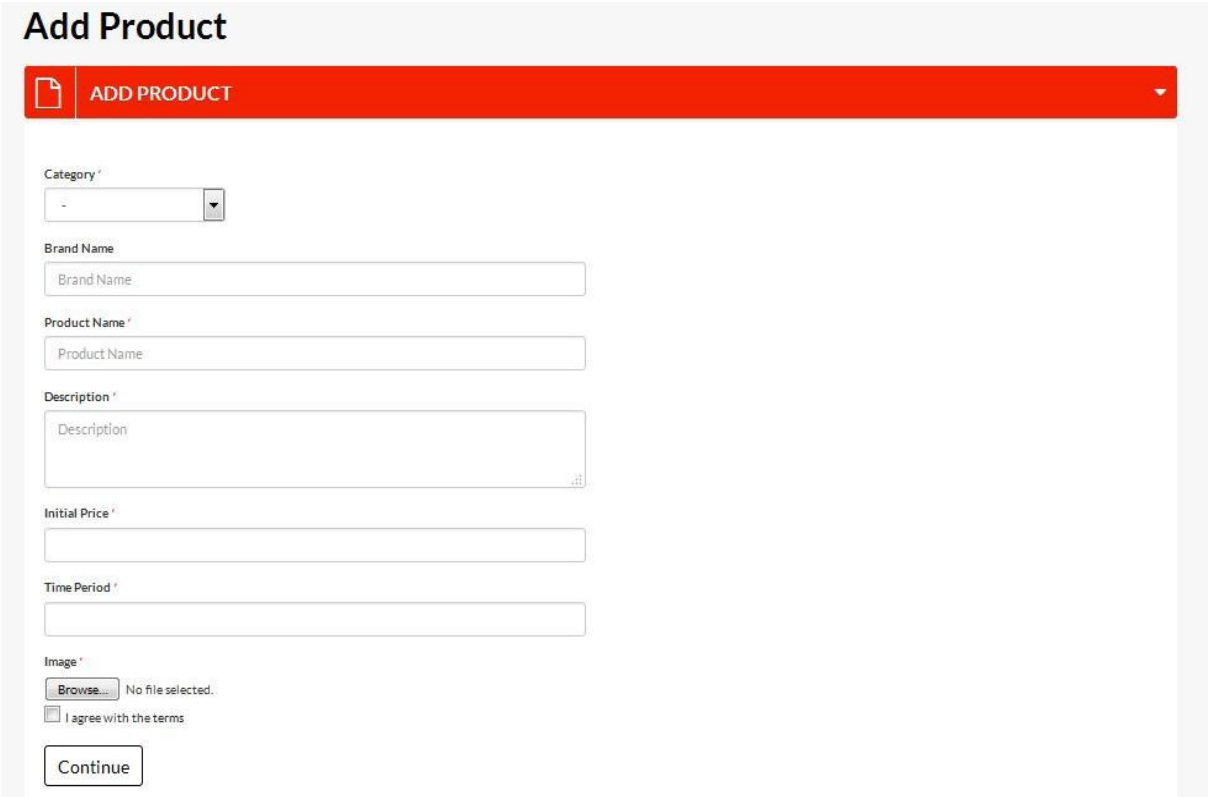
My Account



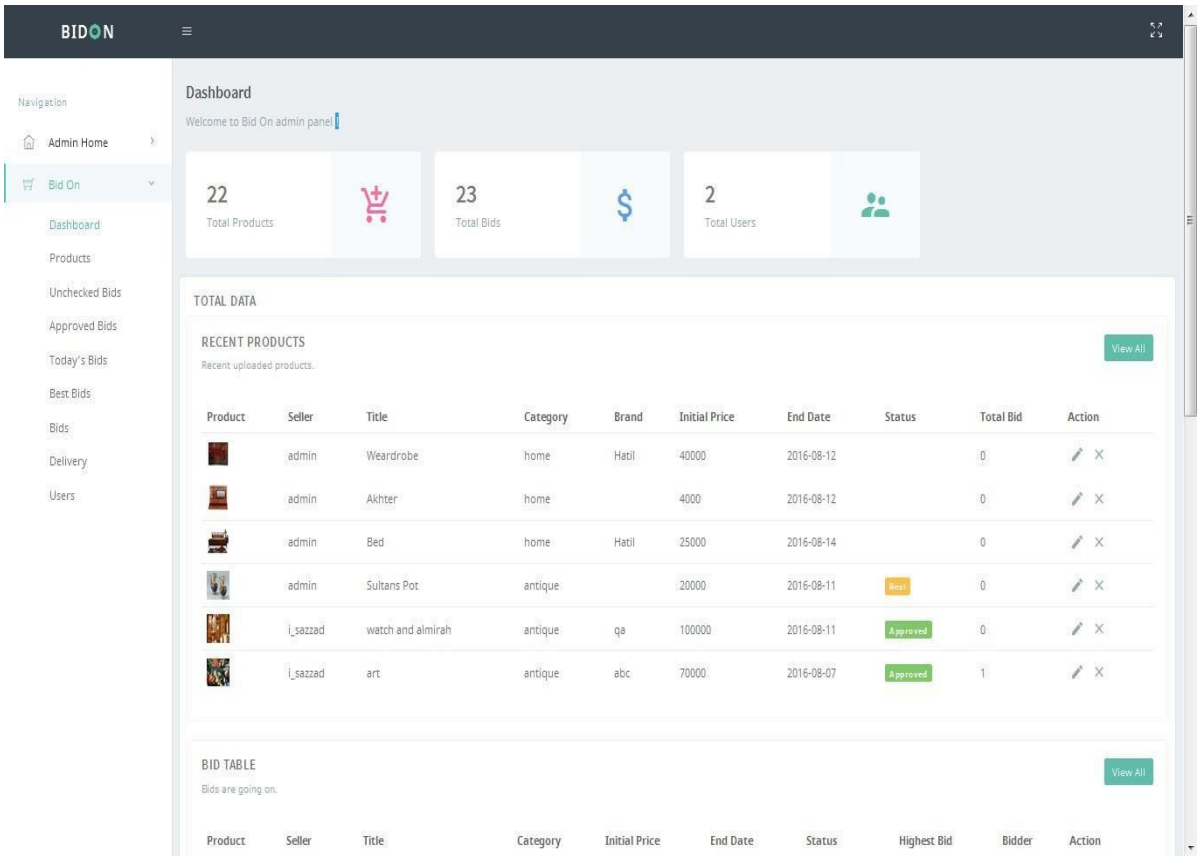
Best Bids



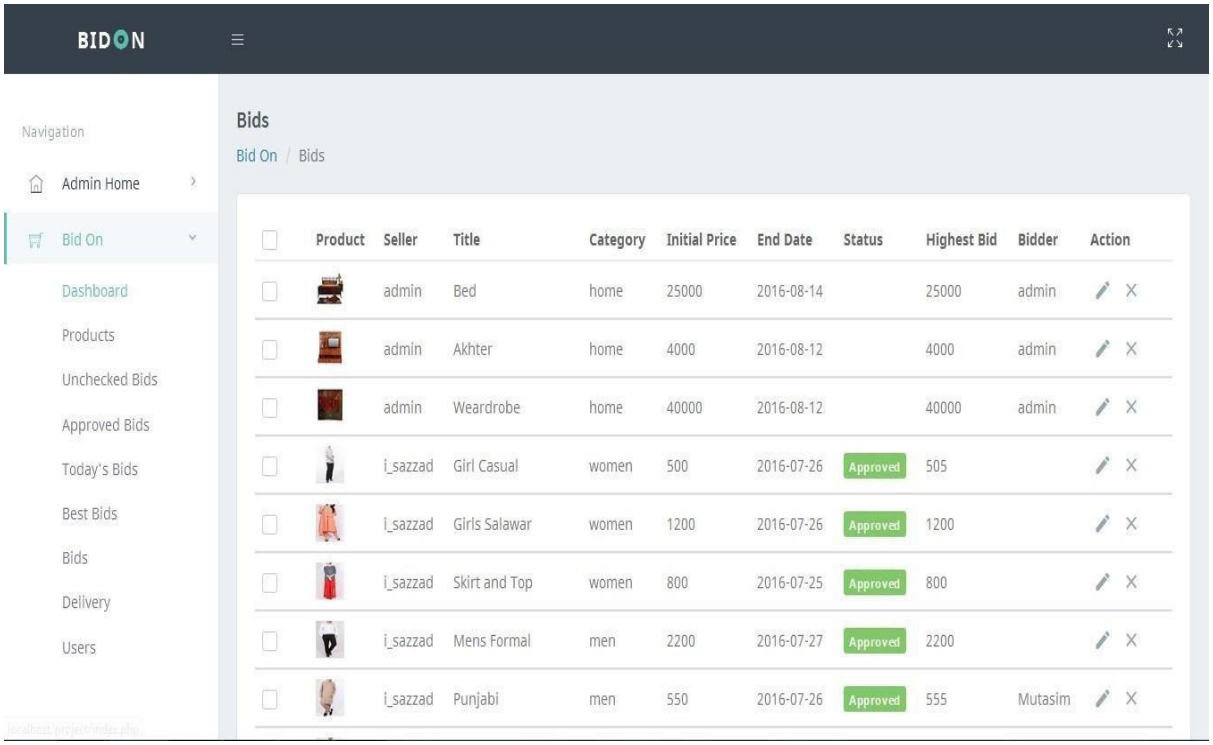
Add Product



Admin Home Page



Bid Page


BIDON
Navigation
Admin Home
Bid On
Dashboard
Products
Unchecked Bids
Approved Bids
Today's Bids
Best Bids
Bids
Delivery
Users
Bids
Bid On / Bids
Product Seller Title Category Initial Price End Date Status Highest Bid Bidder Action
admin Bed home 25000 2016-08-14 25000 admin
admin Akhter home 4000 2016-08-12 4000 admin
admin Weardrobe home 40000 2016-08-12 40000 admin
i_sazzad Girl Casual women 500 2016-07-26 Approved 505
i_sazzad Girls Salawar women 1200 2016-07-26 Approved 1200
i_sazzad Skirt and Top women 800 2016-07-25 Approved 800
i_sazzad Mens Formal men 2200 2016-07-27 Approved 2200
i_sazzad Punjabi men 550 2016-07-26 Approved 555 Mutasim

6. Electronic Cash counter

Problem Analysis:

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 out look to the user interfaces and to implement all the banking transactions:

Some operations include in the project are:

- ✓ **Supply of Account Information**
- ✓ **New Account Creations**
- ✓ **Deposits**
- ✓ **Withdraws**
- ✓ **Cheque book issues**
- ✓ **Stop payments**
- ✓ **Transfer of accounts**
- ✓ **Report Generations**.

These points write with black pen no need to underline

Proposed System:

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

- ➢ User friendliness is provided in the application with various controls.
- ➢ The system makes the overall project management much easier and flexible.
- ➢ Readily upload the latest updates, allows user to download the alerts by clicking the URL.
- ➢ There is no risk of data mismanagement at any level while the project development is under process.
- ➢ It provides high level of security with different level of authentication

Software Requirement Analysis:

Actors Involved in Electronic Cash Payment Systems:

- ➢ **Customers:** Customers use the digital cash payment systems to make purchases.
- ➢ **Dealers:** Dealers have to bear the costs of payment transactions.
- ➢ **Providers for digital payment systems:** Providers are intermediaries between dealers and financial institutions. They provide services and training.
- ➢ **Financial institutions:** Banking systems or organizations that use electronic payment systems.
- ➢ **Trust Centers:** They control digital signature keys, and help to secure customer confidence in certain payment systems. They are responsible for the integrity of transmitted data and authenticity of contractors.

Customer:

Functional requirements:

- Create new accounts.
- Provide account information.
- Money transferring to other users.
- Give multiple payment options. Ex: cash pay, wallets pay, coupons pay, cheque pay.
- Getting the reports of transactions.
- Transaction status report.

Non functional requirements:

Security

- Transaction data must be encrypted. So that no one can access the account information.
- Not to allow user to check the other users account data.

Reliability

- Unsuccessful transaction must recovery immediately.
- Make sure no data loss.

Portability

- Able to do transaction anywhere.

Usability

- Easy to understand and easy to use.

Dealers:

Functional requirements:

- Provide customer contact data and transaction details. (not complete information)
- Getting the reports of transactions.
- Can create account.
- Transaction status report.

Non functional requirements:

Security

- Bank account data hiding.
- Restrict the customer data accessing. Provide necessary data only.

Reliability

- Unsuccessful transaction must recovery immediately.
- Make sure no data loss.

Portability

- Able to do transaction anywhere.

Usability

- Easy to understand and easy to use.

Providers for digital payment systems:

Functional requirements:

- Provide easy and secure interface between customers and dealers.
- Able to stop the transactions if necessary.
- Tracking the transactions.
- Getting transaction report.
- Transaction status report.

Non functional requirements:

Security

- Perfectly managing the dealers and customers accounts.

Interoperability:

- Can communicate with the other bank systems easily.

Financial institutions:

Requirements:

- Able to provide easy accessing to the application.
- Able to accept the online currency.

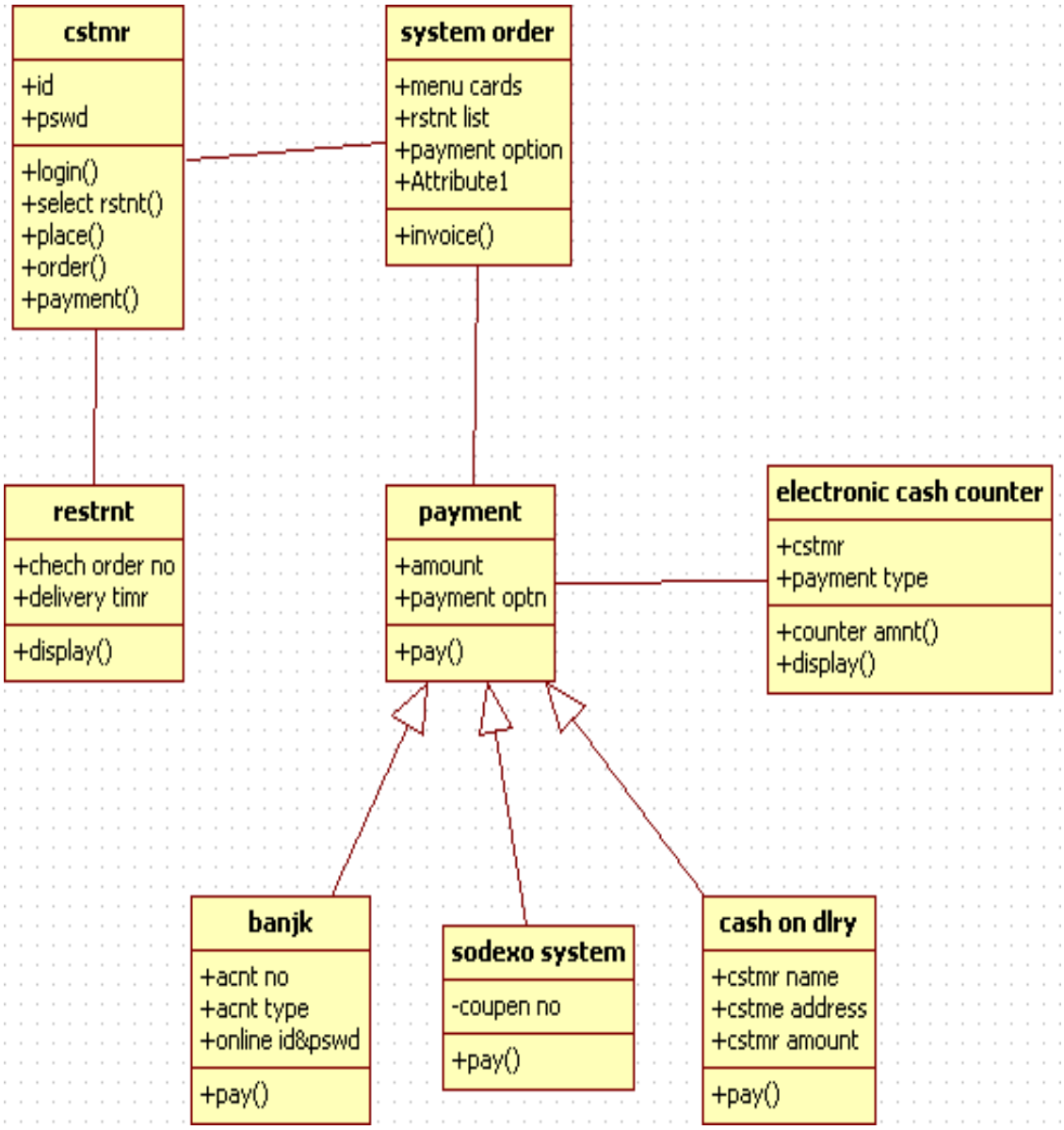
Trust Centers:

Requirements:

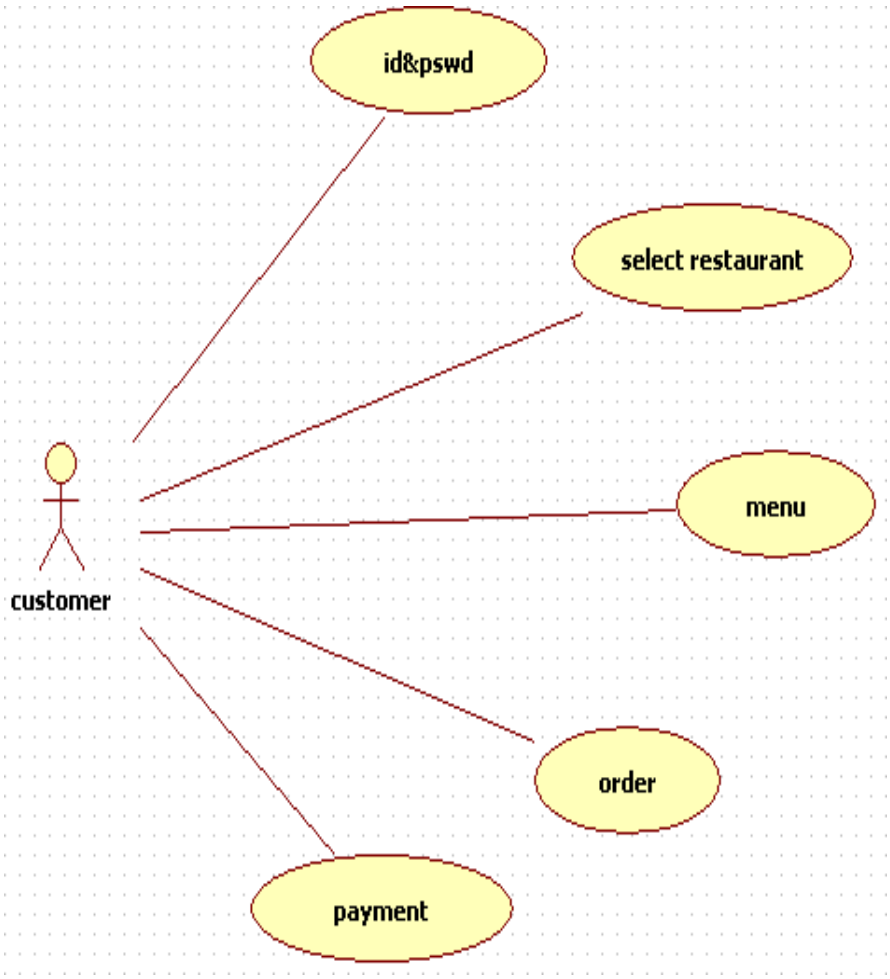
- Generate keys for the transactions.
- Maintain the transaction data privacy by encryption.

Software Designing:

(a) Class diagram

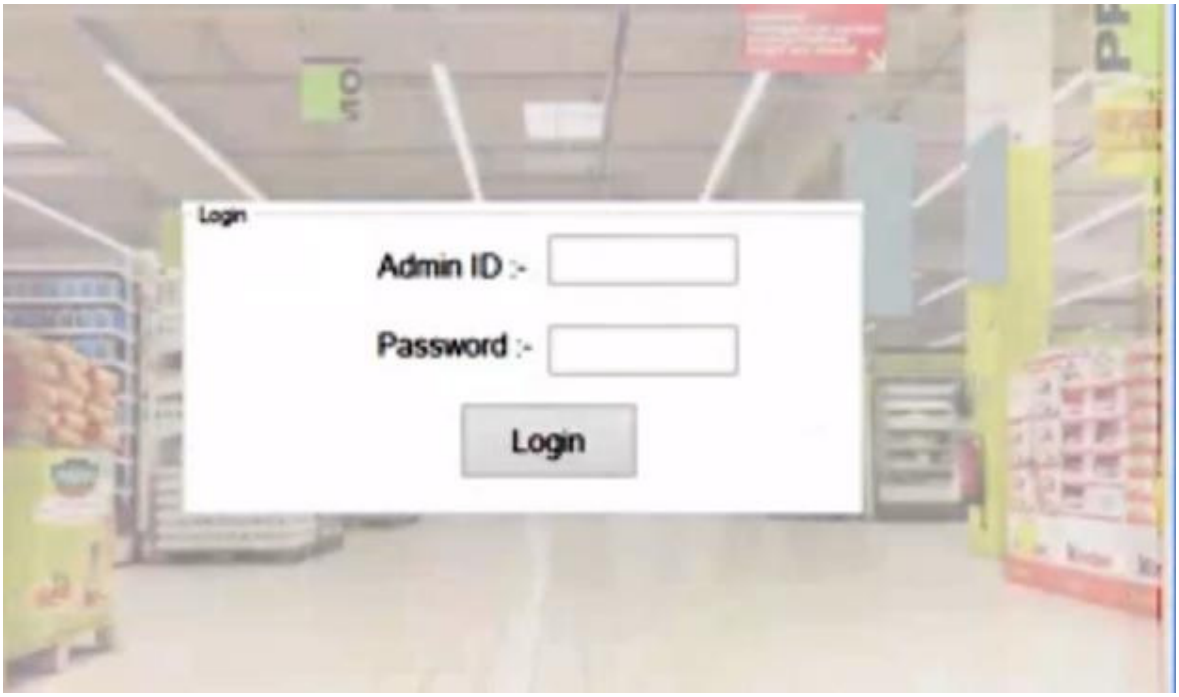


(b) Class diagram



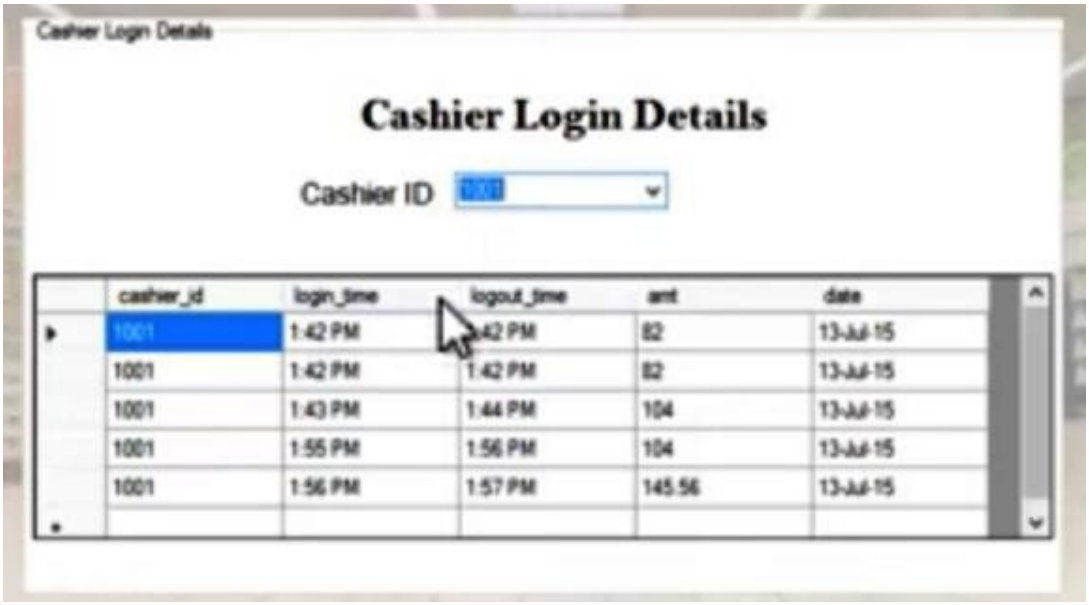
Prototype model:

Login Page



Add cashier Page

Cashier login details Page



Transaction details Page