HOSTEL MANAGEMENT SYSTEM

2022-2023
A PROJECT REPORT

Submitted by

NAVEENCHANDRA	**201231522201**
RADHESHA SHERUGAR	**201231522214**
KIRANA	**201231522189**

In partial fulfillment for the award of the degree of

BACHELOR OF COMPUTER APPLICATIONS

Under the guidance of
Ms. DIVYA K
Lecturer
Department of Computer Science

BHANDARKARS' ARTS AND SCIENCE COLLEGE
KUNDAPURA

MANGALORE UNIVERSITY

BHANDARKARS' ARTS AND SCIENCE COLLEGE

KUNDAPURA -576201

Department of Computer Science

Certificate

Certified that the project work entitled

.....HOSTEL MANAGEMENT SYSTEM.....
is a bonafied work carried out by

..... NAVEENCHANDRA, RADHESHA SHERUGAR, KIRANA .....
in partial fulfilment for the award of degree of Bachelor of Computer Applications of the Mangalore University during the year 2022-23. The project report has been approved as it satisfies the academic requirements of project work prescribed for the Bachelor of Computer Applications.

Project Guide

Head of the Department
Department of Computer Science

Name of the Student:	NAVEENCHANDRA,RADHESHA SHERUGAR,KIRANA
Roll Number:	200533,200536,200529
University Register Number:	201231522201, 201231522214, 201231522189

Examiners: 1. **2.**

BHANDARKARS' ARTS AND SCIENCE COLLEGE

KUNDAPURA-576201

ATTENDANCE CERTIFICATE

This is to certify that the following students of sixth semester BCA has got adequate attendance in the project work as stipulated by Mangalore University in BCA regulations.

NAVEENCHANDRA	:201231522201
RADHESH SHERUGAR	:201231522214
KIRANA	:201231522189

PRINCIPAL

DECLARATION

We hereby declare that this project work entitled "HOSTEL MANAGEMENT SYSTEM" has been prepared by us during the year 2022–23 under the guidance of Ms. DIVYA K, Lecturer Department of Computer Science, Bhandarkars' Arts and Science College, Kundapura in the partial fulfillment of BCA degree prescribed by the Mangalore University.

We also declare that this project is the outcome of our own effort, that it has not been submitted to any other university for the award of any degree.

Date:

NAVEENCHANDRA	**201231522201**
RADHESHA SHERUGAR	**201231522214**
KIRANA	**201231522189**

ACKNOWLEDGEMENT

We consider it a privilege to express our profound and gratitude and respect to those who guided us in the successful completion of our project "**HOSTEL MANAGEMENT SYSTEM**".

We, the project team members are deeply indebted to our project guide **Ms. Divya K**, **Lecturer**, **Department of Computer Science** for giving us timely advise and supports during the completion of project.

We would like to express our gratitude to **Mrs. Vijayalaxmi**, **Head of Department**, **Computer Science,** for this kind concern and encouragement during the completion of our project.

We sincerely thankful to **Dr. N. P. Narayana Shetty**, **Principal**, for providing the opportunity and facility to develop this project.

Our sincere thanks to all faculty members of computer science department for their constant encouragement.

We are thankful to our parents and our friends for their whole hearted support and the cooperation.

Thanking You,

Project Team,

- NAVEENCHANDRA
- RADHESHA SHERUGAR
- KIRANA

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

1. INTRODUCTION

1.1 Introduction of the System

1.1.1 Title of the Project

Hostel Management System

1.1.2 Category of the Project

Web Application

1.1.3 Overview of the Project

Hostel Management System is the application that helps hostel administrator manage various aspects of running hostel like student registration, managing rooms, managing wardens and more.

1.2 Background

1.2.1 Introduction of the Company

Not applicable

1.2.2 Brief note on existing System

- Traditionally, the hostel management system has been a manual process, involving a lot of paperwork and time consuming administrative task especially when dealing with large number of student.
- In this system, the administrators are responsible for registering student, managing student records, room allocation and other task.

1.3 Objective of the system

- The Objectives are predetermined goals of the system.
- The main objective of hostel management system is to reduce the paper works and manage all the activities like student record, warden record and other related task on computer.
- This system is efficient and effective.

1.4 Scope of the system

- Scope is a limitation that process faces from beginning to end.
- Student need internet connection while registering in order to get password to login.
- A room can only have a maximum of 4 seats.
- Student cannot quit before the checkout date.
- If student quit before checkout date hostel fees cannot be refundable.

1.5 Structure of the system

1.5.1 Login Module

In this module allows Admin/Warden/Student to log in using username and password.

1.5.2 Register Module

In this module, Student need to provide some personal, educational details and pay the hostel fees to register.

1.5.3 Admin Module

1.5.3.1 Dashboard

In this module, Admin can see all details in brief like total students, total staff, total rooms and some other details.

1.5.3.2 Manage Student

In this module, Admin can view the details of the specific students, remove student, update student details, view all students.

1.5.3.2.1 Update student

In this module, Admin can update the details of the student.

1.5.3.2.2 Delete student

In this module, Admin can delete the student.

1.5.3.2.3 View all student

In this module, Admin can view the details of all the student.

1.5.3.2.4 Search and View student

In this module, Admin can search and view the details of the specific student.

1.5.3.3 Manage Warden

In this module, Admin can view the details of all the warden, delete warden, add and update warden information.

1.5.3.3.1 Add warden

In this module, Admin can add the warden to the hostel.

1.5.3.3.2 Delete warden

In this module, Admin can delete the details of the warden.

1.5.3.3.3 Update warden

In this module, Admin can update the details of the warden.

1.5.3.3.4 View warden

In this module, Admin can view the details of the warden.

1.5.3.4 Manage Rooms

In this module, Admin can view the details of the room, remove room, add room and update room information.

1.5.3.4.1 Add room

In this module, Admin can add the room to the hostel.

1.5.3.4.2 Delete room

In this module, Admin can delete the room details.

1.5.3.4.3 Update room

In this module, Admin can update the details of the room.

1.5.3.4.4 View all room

In this module, Admin can view the details of the room.

1.5.3.5 Manage Food Menu

In this module, Admin can view the details of the food menu and update food menu.

1.5.3.5.1 Update foodmenu

In this module, Admin can update foodmenu.

1.5.3.6 Manage Event

In this module, Admin can upload or view the events/alerts of the hostel.

1.5.3.6.1 Upload event

In this module, Admin can upload event/alerts.

1.5.3.7 View students Payment

In this module, Admin can view payment done by the student.

1.5.4 Warden Module

1.5.4.1 Dashboard

In this module, warden can see all details in brief like total student, total staff, total rooms and some other details.

1.5.4.2 My Profile

In this module, Warden can view their profile.

1.5.4.3 View Student Details

In this module, Based on gender warden can see all students details.

1.5.4.4 Manage Food Menu

In this module, Warden can view the food menu and update food menu.

1.5.4.4.1 Update foodmenu

In this module, Warden can update foodmenu.

1.5.4.5 Manage Event

In this module, Warden can upload or view the events/alerts of the hostel.

1.5.4.5.1 Upload event

In this module, Warden can Upload event/alerts.

1.5.4.6 Meal price

In this module, Warden can calculate the month meal price for the student.

1.5.5 Student Module

1.5.5.1 Dashboard

In this module, Student can see all details in brief like total student, total staff, total rooms and some other details.

1.5.5.2 My Profile

In this module, Student can view their profile and allowed to update some field.

1.5.5.2.1 Update my profile

In this module, student can update their profile.

1.5.5.3 View Food Menu

In this module, Student can view the food menu and his monthly meal fees.

1.5.5.4 View Events

In this module, Student can view events/alerts uploaded by warden or admin.

1.5.5.5 Renewal

In this module, Student can renewal or extend the checkout date.

1.6 System Architecture

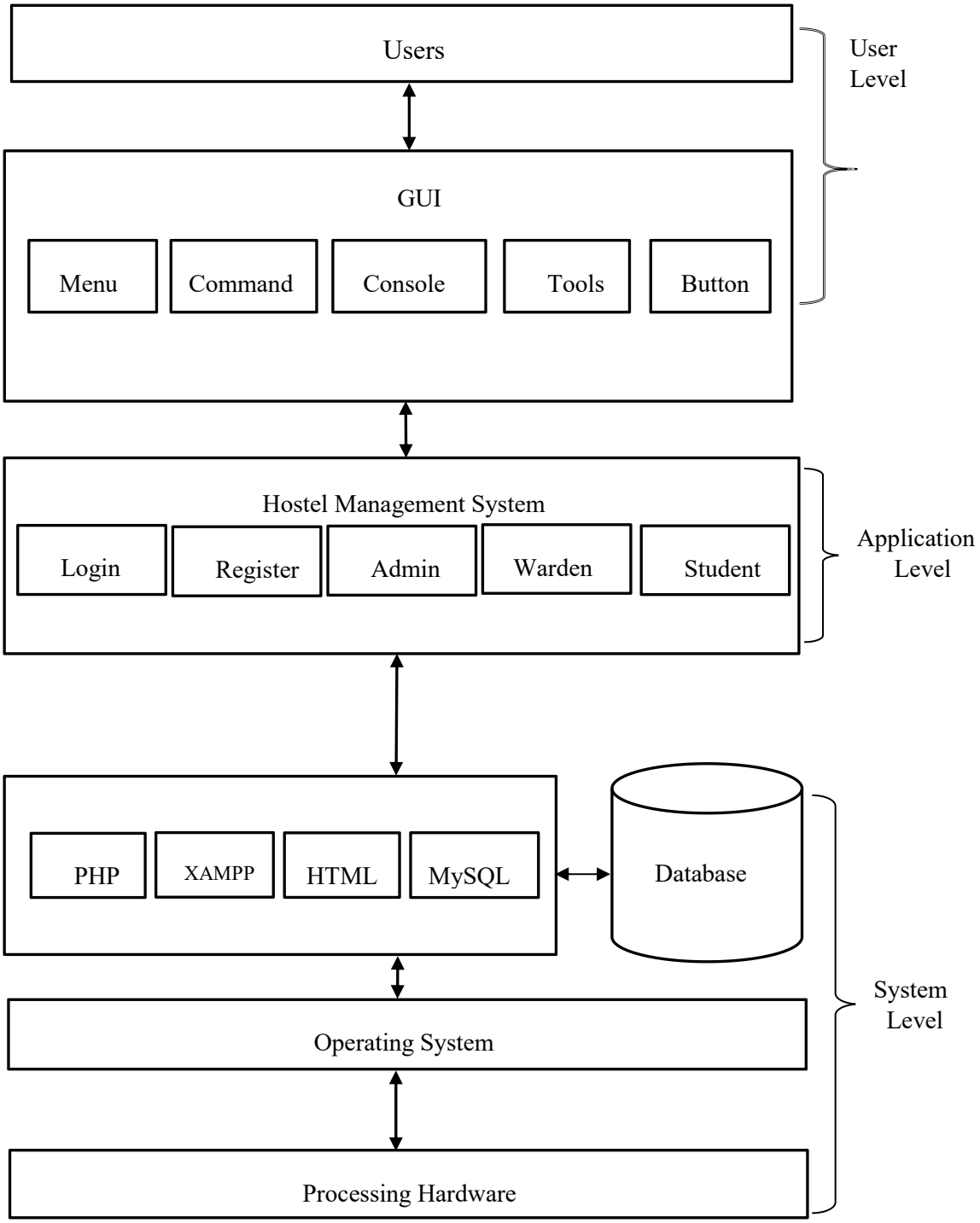


Figure 1.1

1.7 End Users

- Admin
- Warden
- Student

1.8 Software/Hardware need for Development

- Operating System: Windows 7 or higher.
- CPU: Intel core and AMD processor with 64-bit support.
- XAMPP server.
- HTML, CSS, MYSQL, PHP, JavaScript.
- VS code or similar editor for writing code.

1.9 Software/Hardware need for the implementation

- Web browser (latest version of Chrome, Edge, Brave browsers).
- Internet.
- Disk storage: 500 GB hard disk or more.
- RAM-4GB

SOFTWARE REQUIREMENT SPECIFICATON

2. SOFTWARE REQUIREMENT SPECIFICATION

2.1 Introduction

The SRS is a document that completely describes what the proposed system should do without describing how the software will do it. SRS fully describes external behavior of the application or the sab system identifying it. It also describes non function requirements, design, constraints and other factors necessary to provide a complete and comprehensive description of the requirement of the software. The major purpose of the SRS is to bridge the communication gap between user and developer. SRS is the median through which the client and user needs are accurately specified.

Boundaries of software products are defined by set of requirements. The software development team designs, tests and delivers these requirements to a user requirements automatic unit of a software product. From the view point of the users, as a rule requirements are always correct, unambiguous, verifiable and traceable based on the requirements specified the final product can be verify for its correctness. The SRS describes the various system requirement, interfaces, features and function in detail. Software Requirement Specification helps the developer to understand their needs.

2.2 Overall Description

This section describes the function of the project and their aim.It also includes the constraints and the requirement of the project.

2.2.1 Product Perspective

2.2.1.1 System Interfaces

This application runs in the latest version of Chrome or Firefox on Windows.

2.2.1.2 User Interface

This application GUI provides menus, toolbars, buttons, labels, anchor tags etc allowing for easy control by a keyboard, mouse.

2.2.1.3 Hardware Interface

Not applicable.

2.2.1.4 Software Interface

This application allows to work with DML, DDL, DCL query with MySQL.

2.2.1.5 Communication Interface

Internet.

2.2.1.6 Interface with server

This application allows to interface with MYSQL server, Xampp, apache.

2.2.2 Product Functions

- Product function specifies how the given product or software will work. There are 5 main modules Login, Register, Admin, Warden, Student.
- Login module is used to login into the application, a user can login by using their username or user-id and password.
- Register module is used to register and pay the hostel fees by the student.
- Admin module is used to view dashboard, manage student details, manage warden details, manage room details, manage food menu, manage event, view student payments.
- Warden module used to view dashboard, view profile, view student details, manage food menu, manage event, and calculate month meal price of the student.
- Student module used to view dashboard, manage profile, view food menu, view events and renewal.

2.2.3 User characteristics

User should have basic knowledge of how to operate a computer.

2.2.4 General constraints

General Constraint describe how the product operates inside various circumstances and limit the options designers have if building the product.

The developed application should run on various latest versions of Chrome and Firefox on Windows 7 or higher version.

2.2.5 Assumption and Dependencies

These factors are not design constraint on the software but any changes to these factors can affect the requirement in the SRS.

We assume that we have all the requirements like a standard of 500GB Hard disk, 4GB RAM.

2.3 Special Requirements

Not applicable.

2.4 Functional requirements

2.4.1 Login module

a. Input: E-mail id or user-id, password.

b. Process: It will validate the input given by user with the database.

c. Output: Main page will be displayed if the input is matched with database otherwise error message will be displayed.

2.4.2 Register module

a. Input: Name, father name, gender, register number, email, checkout date, mobile number,blood group, caste, college name,course duration, address, aadhar number, room no,card number, cardholder name, expiry year and month, cvv.

b. Process: Reads data and validates and inserts the records to the table.

c. Output: If valid, displays 'payment done and registration successful log details sent to your email' else shows error message.

2.4.3 Admin module

2.4.3.1 Dashboard

a. Input: Button click.

b. Process: Retrieves the data from tables.

c. Output: Displays the total number of students, wardens, rooms etc.

2.4.3.2 Manage student

2.4.3.2.1 Update student

a. Input: Name, father name, gender, register number, mobile number,blood group, caste, college name,course duration,address, aadhar number, room no.

b. Process: Validates the input and updates data to the table.

c. Output: If valid, displays 'Record updated Successfully' else shows error message with description.

2.4.3.2.2 Delete student

a. Input: Register number.

b. Process: If found, Retrieves and updates the student status as "old student".

c. Output: If valid displays 'Deleted Successfully' else shows error message.

2.4.3.2.3 View all students

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays all the student records.

2.4.3.2.4 Search and view student

a. Input: Register Number.

b. Process: If found, Retrieves the data from table.

c. Output: If valid displays the student record else shows error message with description.

2.4.3.3 Manage warden

2.4.3.3.1 Add warden

a. Input: Name, warden id, warden email, gender, aadhar number, address, phone number.

b. Process: Validates the input and inserts records to the table.

c. Output: If valid, displays 'Warden added Successfully' else shows error message with description.

2.4.3.3.2 Delete warden

a. Input: Warden Id.

b. Process: If found, Retrieves and updates the warden status as “old warden”.

c. Output: If valid displays ‘Deleted Successfully’ else shows error message with description.

2.4.3.3.3 Update wardens

a. Input: Name, warden email, gender, aadhar number, address, phone number.

b. Process: Validates the input and updates the data to a table.

c. Output: If valid ,displays ‘Updated Successfully’ else shows error message with description.

2.4.3.3.4 View all wardens

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays all the warden records.

2.4.3.4 Manage room

2.4.3.4.1 Add room

a. Input: Room Id, seater, room for, status.

b. Process: Validates the input and inserts record to the table.

c. Output: If valid, displays ‘Room added Successfully’ else shows error message with description.

2.4.3.4.2 Delete room

a. Input: Room Id.

b. Process: If found, retrieves and removes the record.

c. Output: If valid, displays ‘Room Deleted Successfully’ else shows error message with description.

2.4.3.4.3 Update room

a. Input: seater, room for, status.

b. Process: Validates the input and updates the data to a table.

c. Output: If valid, Displays ‘Room Updated Successfully’ else shows error message with description.

2.4.3.4.4 View all rooms

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays the room records.

2.4.3.5 Manage food menu

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays the food menu.

2.4.3.5.1 Update foodmenu

a. Input: Food names.

b. Process: Validates the input and updates the data to a table.

c. Output: If valid, displays 'food menu updated Successfully' else shows error message with description.

2.4.3.6 Manage event

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays the events.

2.4.3.6.1 Upload event

a. Input: Event description and image.

b. Process: Validates the input and updates the data to a table.

c. Output: If valid, displays 'Event uploaded Successfully' else shows error message.

2.4.3.7 View students payment

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays the hostel fees paid by student.

2.4.4 Warden module

2.4.4.1 Dashboard

a. Input: Button click.

b. Process: Retrieves the data from tables.

c. Output: Displays the total number of students, wardens, rooms etc.

2.4.4.2 My profile

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays warden's record.

2.4.4.3 View student details

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays all the student records.

2.4.4.4 Manage food menu

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays the menu.

2.4.4.4.1 Update menu

a. Input: Food names.

b. Process: Validates the input and updates the data to a table.

c. Output: If valid, displays 'Food menu Updated Successfully' else shows error message.

2.4.4.5 Manage event

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays the events.

2.4.4.5.1 Upload event

a. Input: Event description and image.

b. Process: Validates the input and inserts the data to a table.

c. Output: If valid, displays 'Event Uploaded Successfully' else shows error message.

2.4.4.6 Meal price

a. Input: no of days.

b. Process: retrieves the data and validates the input if valid updates the student month meal price.

c. Output: If valid,displays "student monthly meal price updated" else shows error message.

2.4.5 Student module

2.4.5.1 Dashboard

a. Input: Button click.

b. Process: Retrieves the data from tables.

c. Output: Displays the total number of students, wardens, rooms etc.

2.4.5.2 My profile

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays student's record.

2.4.5.2.1 Update my profile

a. Input: College name, Roll no, Course, course duration, previous year mark, address, password.

b. Process: Validates the input and updates the data to a table.

c. Output: If valid, displays the 'Profile Updated Successfully' else shows error message.

2.4.5.3 View food menu

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays the food menu and students month meal price.

2.4.5.4 View events

a. Input: Button click.

b. Process: Retrieves the data from table.

c. Output: Displays the events/alerts uploaded by admin and wardens.

2.4.5.5 Renewal

a. Input: card number, cardholder name, expiry year and month, cvv, extended month.

b. Process: Validates the input and updates the details to the table.

c. Output: If valid, displays "Renewal successfully completed" else shows error message .

2.5 Design Constraints

2.5.1 Hardware Constraint

The development of this application requires a computer system with 4GB RAM and 500GB Hard disk.

2.5.2 Software Constraint

This application development requires XAMPP, MySQL and Windows Operating System.

2.5.3 Fault Tolerance

Fault tolerance requirements can place a major constraint on how the system is to be displayed. Fault tolerance often make the system more complex and expensive, so they should be minimized.

In this application, we use exception handlers and validation control to avoid the fault tolerance.Fault tolerance is achieved by every input data. If the data is failed during its validation, then that input will be discarded. Only the correct information will be received.

2.5.4 Security

Currently security requirements have become essential and major for all types of the systems. Security requirements place restrictions on the use of certain command control access to database, provide different kinds of access, requirements for different people, require the use of passwords and cryptography techniques and maintain a log of activities in the system.

In this application, We use user id and password to authenticate the user.

2.5.5 Standard Compliance

It specifies the requirements for the standard the system must follow. The standards may include Button, Textbox, Labels, Layouts, Radio Button, Drop down etc,.

2.6 System Attributes

- **Availability**

  Availability refers to the percentage of time that the infrastructure, system, or solution remains operational under normal circumstances in order to serve its intended purpose.

- **Portability**

  Portability, in relation to software, is a measure of how easily an application can be transferred from one computer to another. A computer software application is considered portable to a new environment if the effort required to adapt it to the new environment is within reasonable limits.

- **Reliability**

  Reliability refers to the probability that the system will meet certain performance standards in yielding correct output for a desired time duration.

- **Maintainability**

  Maintainability refers to the ease with which you can repair, improve and understand software code. Software maintenance is a phase in the software development cycle that starts after the customer has received the product.

- **Scalability**

  Software scalability is an attribute of a tool or a system to increase its capacity and functionalities based on its users demand. Scalable software can remain stable while adapting to changes, upgrades, overhauls and resource reduction.

2.7 Other Requirements

Not applicable.

SYSTEM DESIGN

3. SYSTEM DESIGN

3.1 Introduction

- System design is the process of defining the architecture, module interfaces and data for a system to satisfy specified requirements.
- The purpose of the design phase is to plan the solution of the problem specified by the requirement documents.
- This is the first step that moving from problem domain to the solution domain.
- This design of the system is essentially a blueprint or a plan for a solution for the system.

3.2 Assumptions and Constraints

An assumption is a condition you think to be true and a constraint is fixed limitations of project development

- All the functional requirement collected from client are sufficient for the project life-cycle.
- All the Non-functional and Specific requirement specified in SRS is well enough for the development of system.
- Time constraint.

3.3 Functional decomposition

Functional decomposition is the process of taking a complex process and breaking it down into its smaller, simpler parts. Using functional decomposition large or complex functionalities are more easily understood. It is mainly used during project analysis phase, so each phase can be viewed as software. So, this has modular with some sub modules.

3.3.1 System software architecture

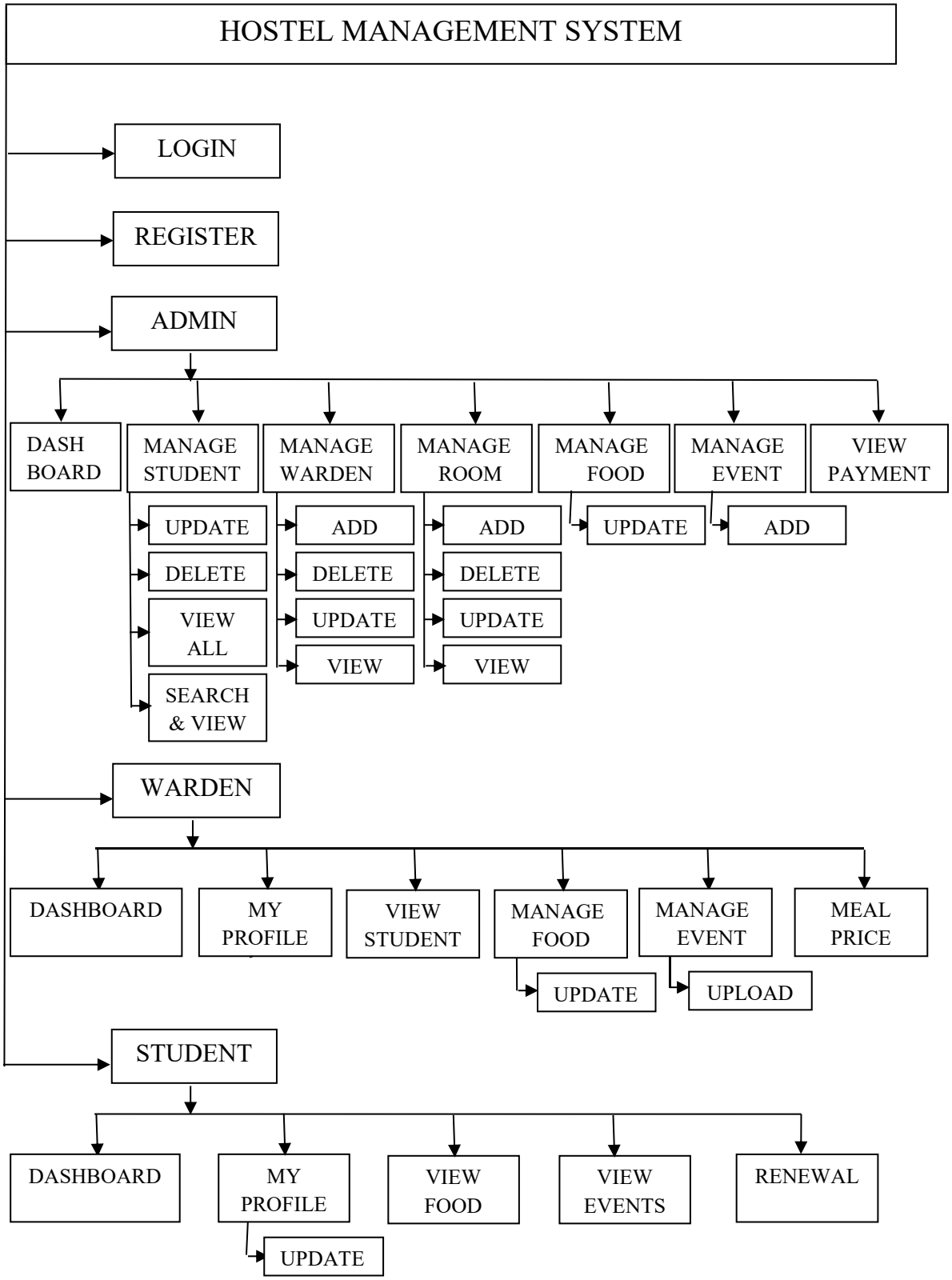


Figure 3.1 System software architecture

3.3.2 System Technical Architecture

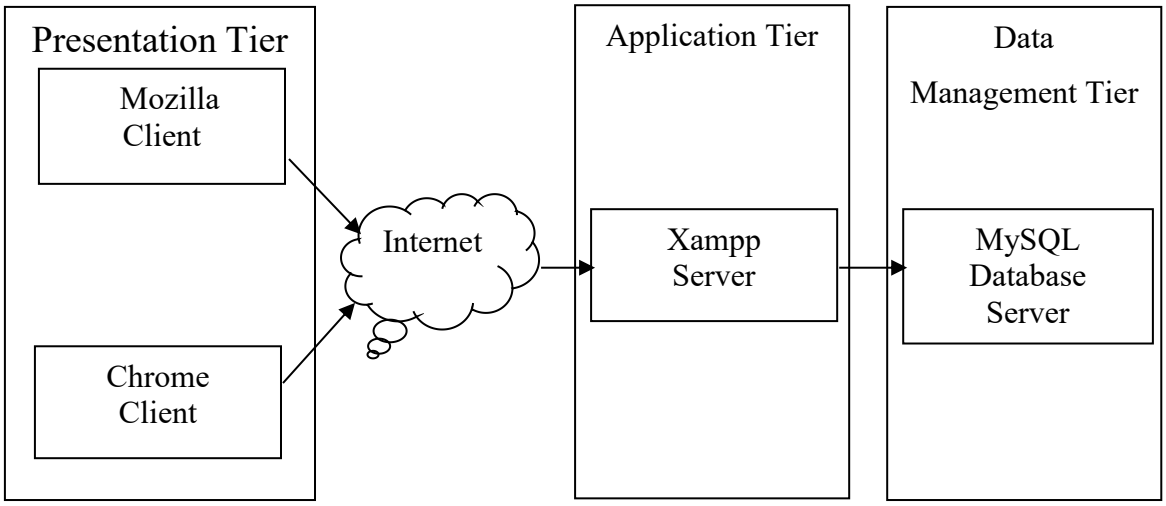


Figure 3.2 System Technical Architecture

3.3.3 System Hardware Architecture

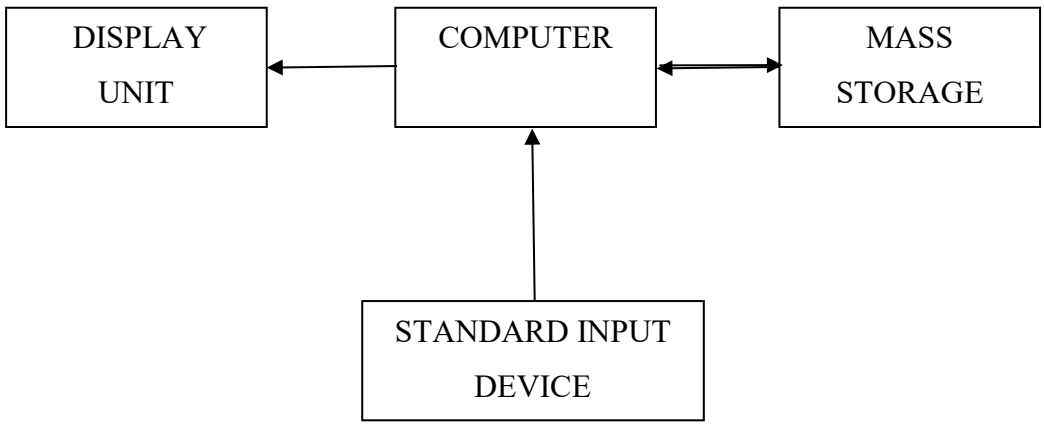


Figure 3.3 System Hardware Architecture

3.3.4 External Interfaces

Not applicable.

3.4 Description of Programs

3.4.1 Context Flow Diagram (CFD)

In CFD entire system is considered as a single process. Context flow diagram shows input and outputs of the system. It shows all the external entities that interact with the system and how the data flows between this external entities and system.

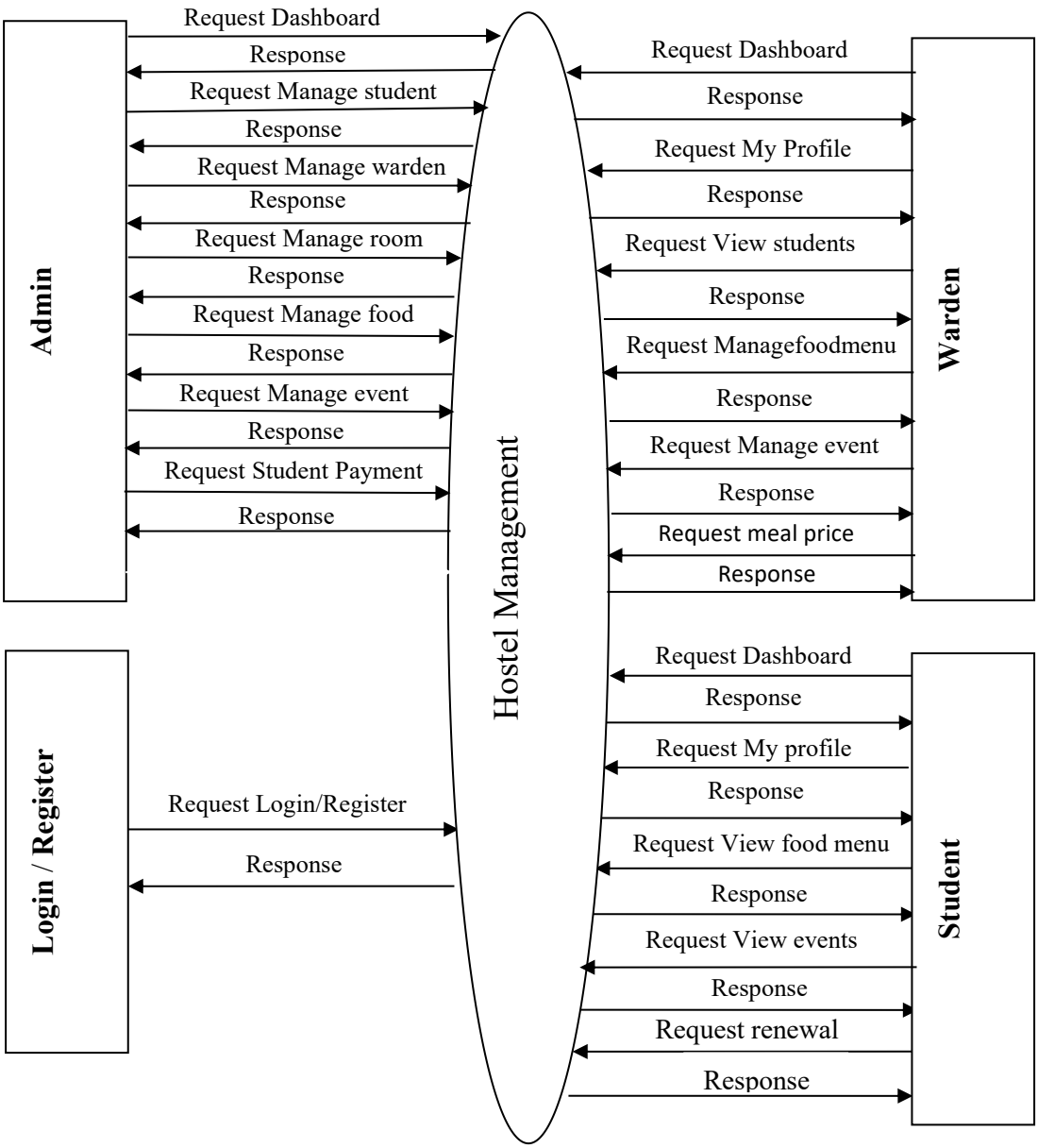


Figure 3.4 Context Flow Diagram

3.4.2 Data Flow Diagrams (DFDs)

Data Flow Diagram (DFD)

Data flow diagram shows the flow of data through system. Data flow diagrams also called the data flow graphs. It views as a function that transforms the inputs into desired outputs. It aims to capture the transformation that take place within a system to the input data so that eventually the output data is produced.

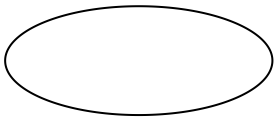
Symbols	Name	Description
	Process	It performs transformation of data from one state to another.
	Source/Sink	It represents the external entity that may be either source or Sink.
	Flow of data	It represent the flow of data source to destination.
	Data Source/Data storage	It is the place where data is stored.

Table 3.1 Data Flow Diagram (DFD)

Level-0 DFD

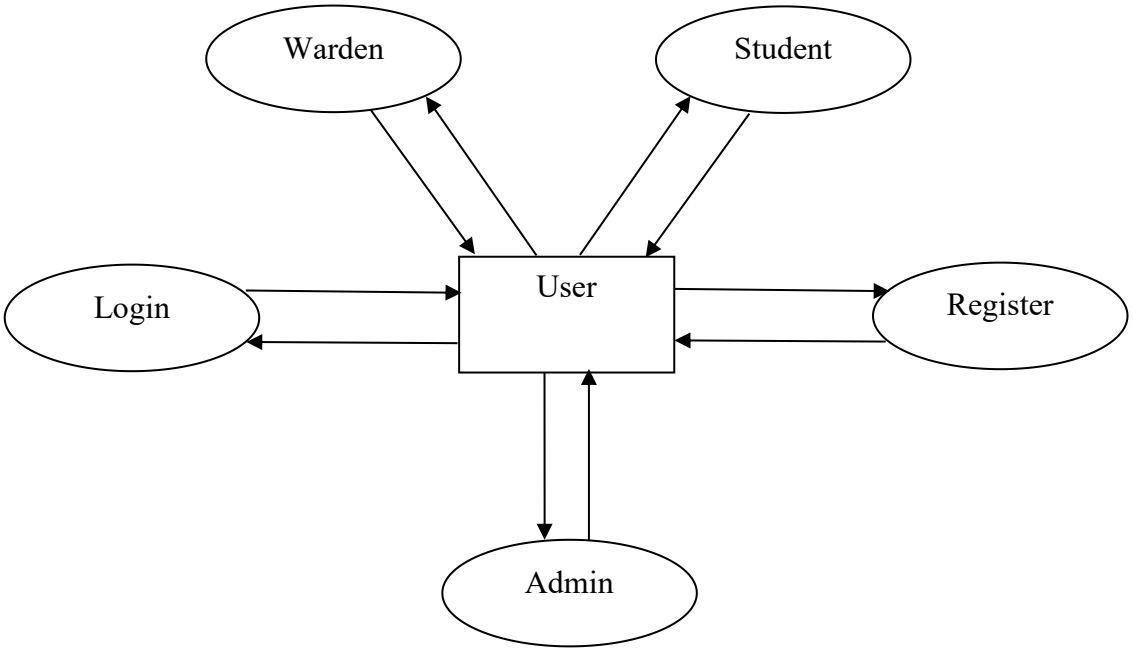


Figure 3.5 DFD Level-0

3.5 Description of Components

3.5.1 Functional Component

3.5.1.1 Login

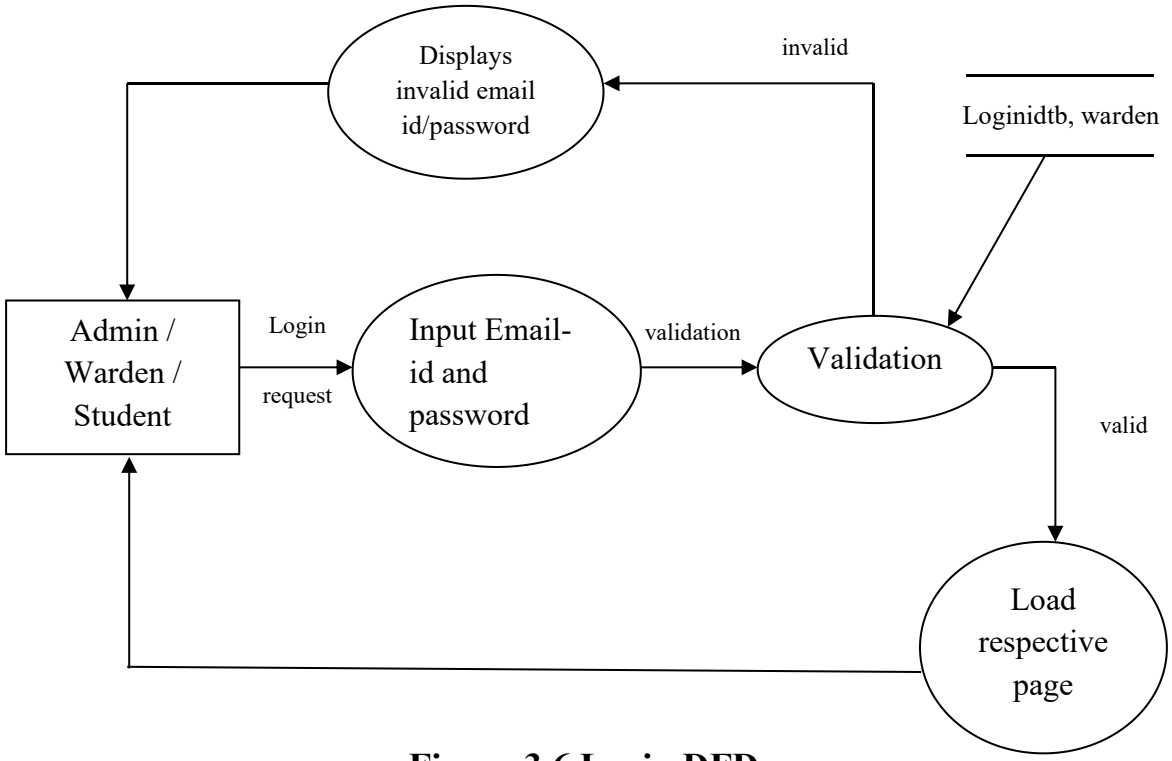


Figure 3.6 Login DFD

3.5.1.1.1 Input: E-mail id, password.

3.5.1.1.2 Process definition: It will validate the input given by user with the database.

3.5.1.1.3 Output: Dashboard will be displayed if the input is matched with database otherwise error message will be displayed with description.

3.5.1.1.4 Interface with other functional components: Register, Add warden.

3.5.1.1.5 Resource Allocation: Loginidtb,warden table.

3.5.1.1.6 User Interface: Textbox, Button, Label.

3.5.1.2 Register

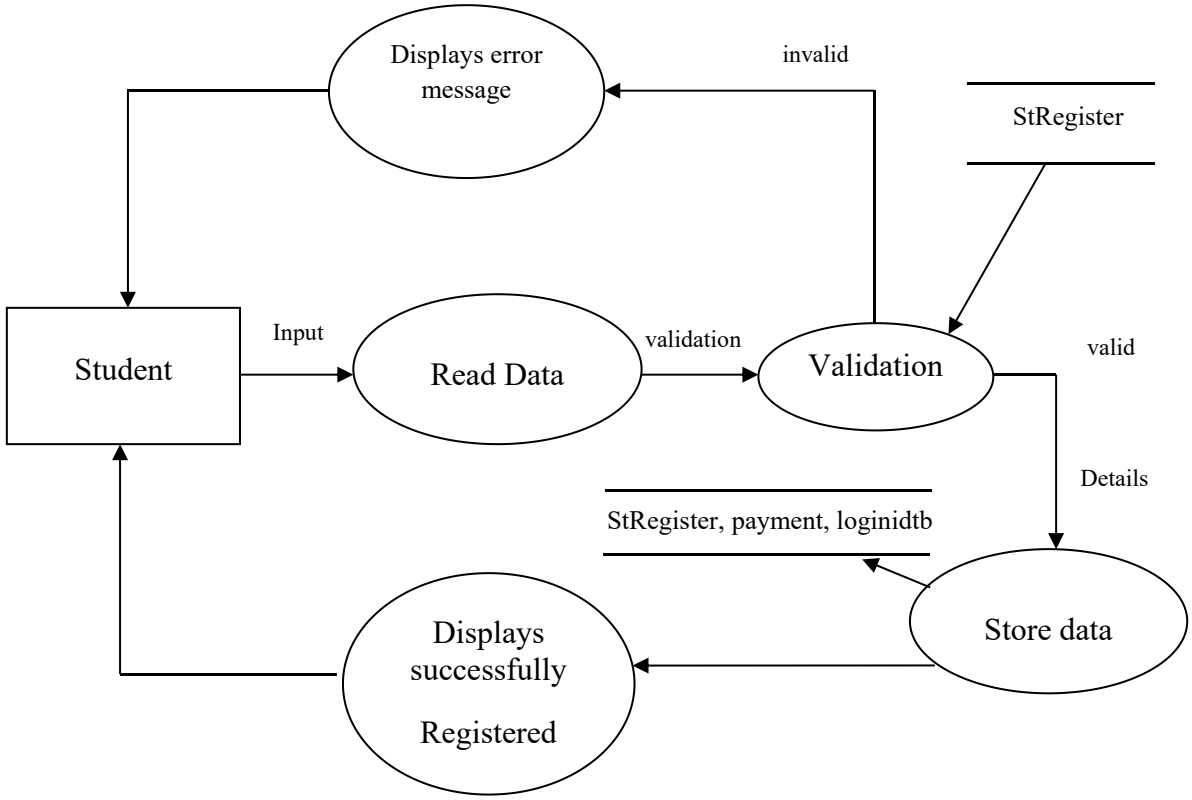


Figure 3.7 Register DFD

3.5.1.2.1 Input: Name, father name, gender, register number, email, staying duration, mobile number,blood group, caste, college name,course duration,address, aadhar number,card number, cardholder name, expiry year and month, cvv, room no.

3.5.1.2.2 Process definition: Reads data and validates.if valid inserts the records to the table.

3.5.1.2.3 Output: If valid, displays 'payment done, Registration Successful' else shows error message with description.

3.5.1.2.4 Interface with other functional components: Independent module.

3.5.1.2.5 Resource Allocation: StRegister,payment, loginidtb table.

3.5.1.2.6 User Interface: Textbox, Button, Label, Drop-down.

3.5.1.3 Admin

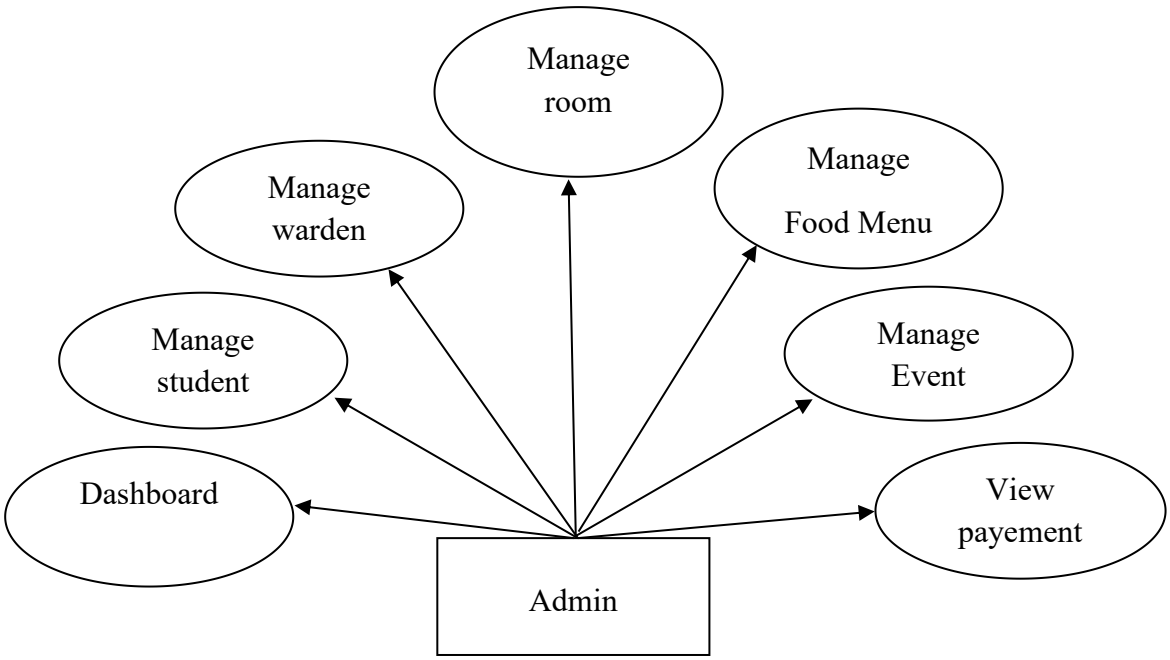


Figure 3.8 Admin DFD

3.5.1.3.1 Dashboard

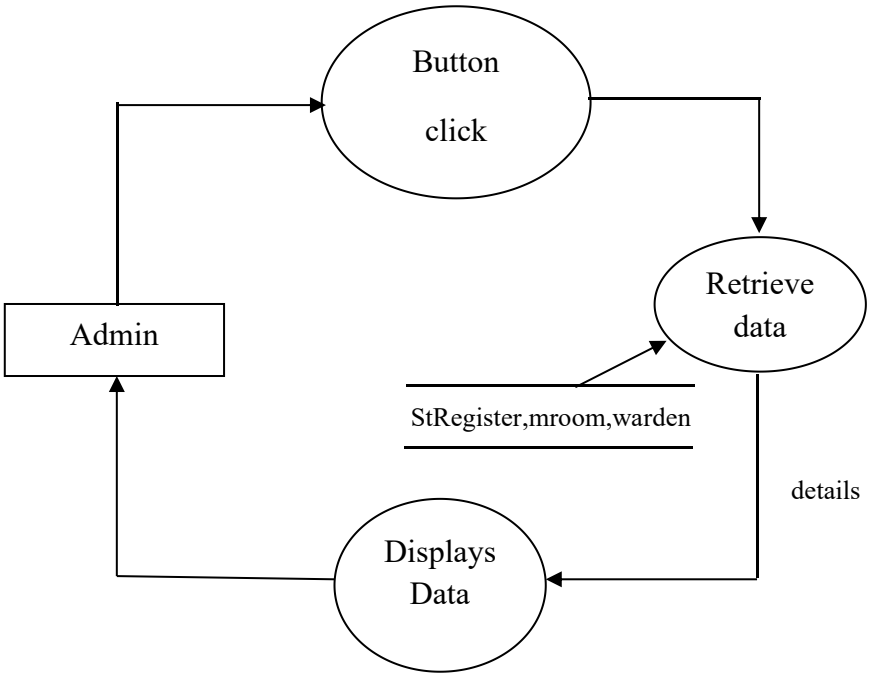


Figure 3.9 Dashboard DFD

3.5.1.3.1.1 Input: Button click.

3.5.1.3.1.2 Process definition: Retrieve the data from database.

3.5.1.3.1.3 Output: Displays details.

3.5.1.3.1.4 Interface with other functional components: Register, Add room, Add warden.

3.5.1.3.1.5 Resource Allocation: StRegister, mroom, warden tables.

3.5.1.3.1.6 User Interface: Button and labels.

3.5.1.3.2 Manage Students

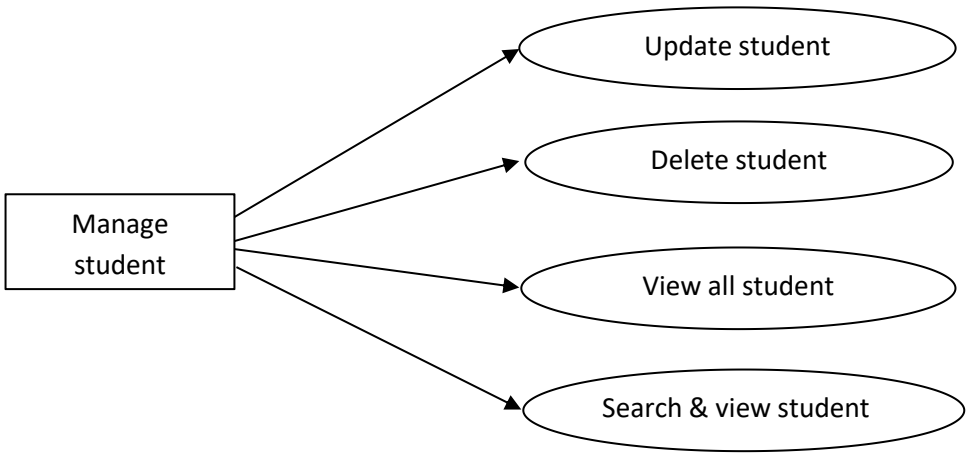


Figure 3.10 Manage Students DFD

3.5.1.3.2.1 Update student

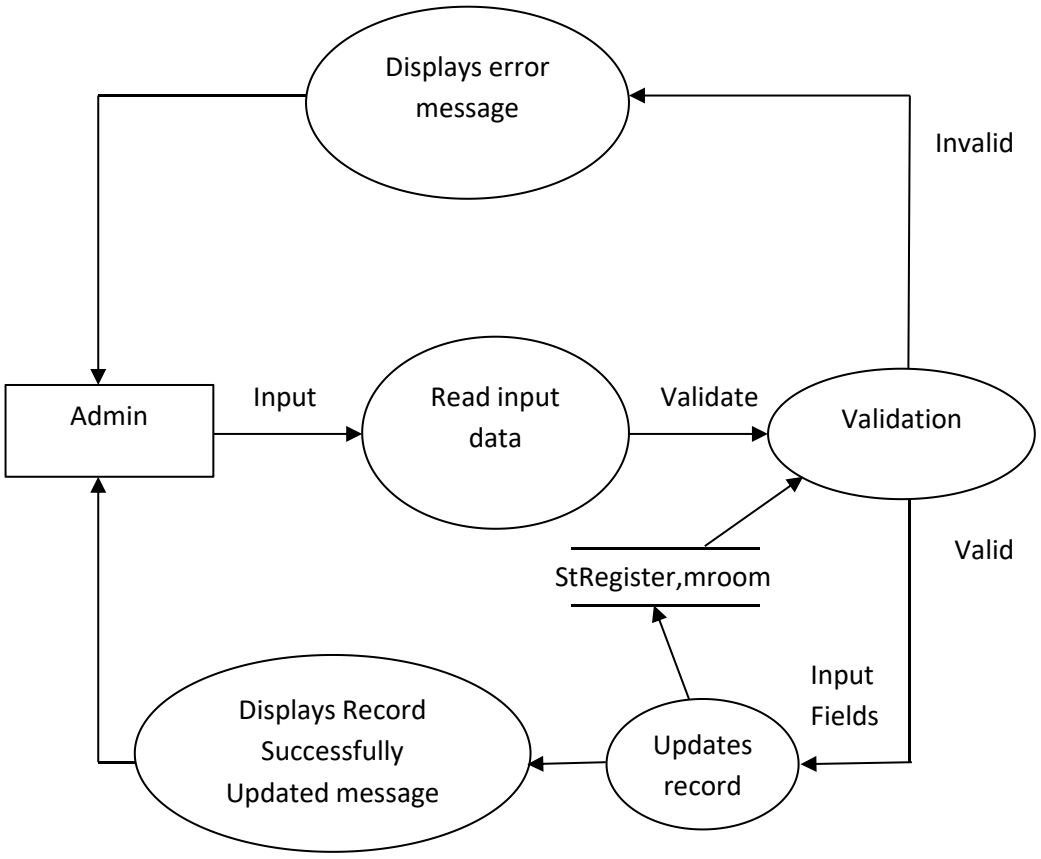


Figure 3.11 Update student DFD

3.5.1.3.2.1.1 Input: Name, father name, gender, register number, mobile number,blood group, caste, college name,course duration,address, aadhar number, room no.

3.5.1.3.2.1.2 Process definition: validates the data if valid updates the record to the table.

3.5.1.3.2.1.3 Output: If valid Displays "record successfully updated" else shows error message with description.

3.5.1.3.2.1.4 Interface with other functional components: Register, Add room.

3.5.1.3.2.1.5 Resource Allocation: StRegister,mroom.

3.5.1.3.2.1.6 User Interface: Button, Label, Textbox, Drop-down

3.5.1.3.2.2 Delete student

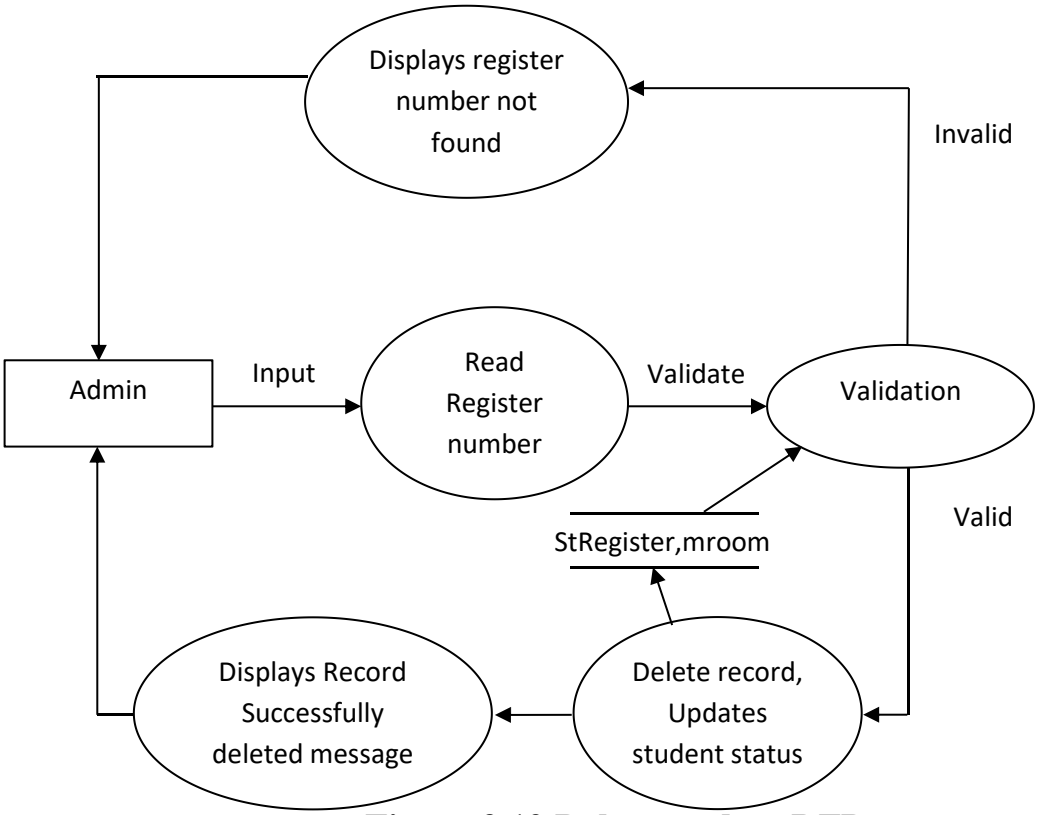


Figure 3.12 Delete student DFD

3.5.1.3.2.2.1 Input: Register number.

3.5.1.3.2.2.2 Process definition: validates the data. if valid deletes record updates student status as “old student” .

3.5.1.3.2.2.3 Output: If valid Displays “record deleted successfully” else shows error message with description.

3.5.1.3.2.2.4 Interface with other functional components: Register.

3.5.1.3.2.2.5 Resource Allocation: StRegister,mroom table.

3.5.1.3.2.2.6 User Interface: Button, Label, Textbox, Drop-down.

3.5.1.3.2.3 View all student

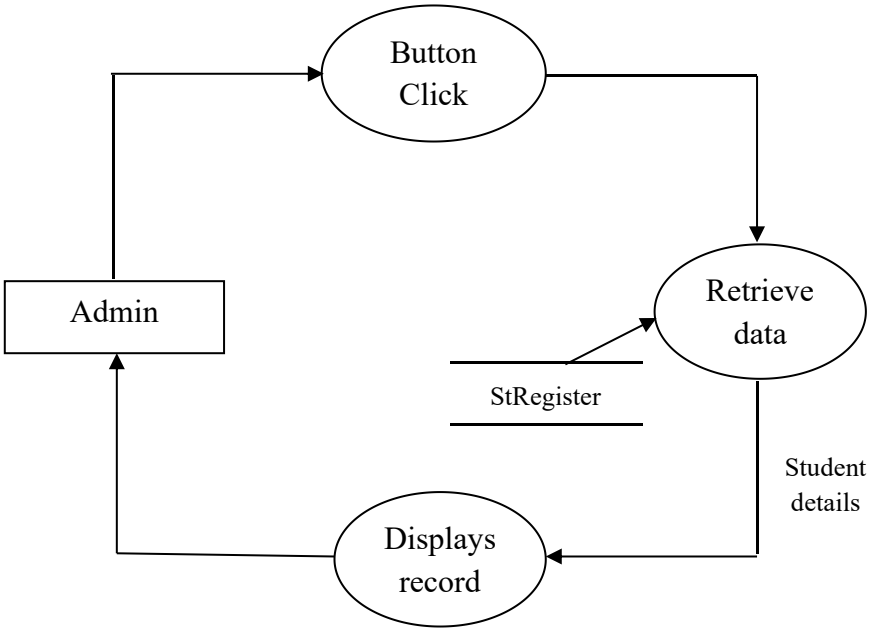


Figure 3.13 View all student DFD

3.5.1.3.2.3.1 Input: Button click.

3.5.1.3.2.3.2 Process definition: Retrieve details from tables.

3.5.1.3.2.3.3 Output: Displays details from tables.

3.5.1.3.2.3.4 Interface with other functional components: Register module.

3.5.1.3.2.3.5 Resource Allocation: StRegister.

3.5.1.3.2.3.6 User Interface: Button, Textbox and labels.

3.5.1.3.2.4 Search and view student

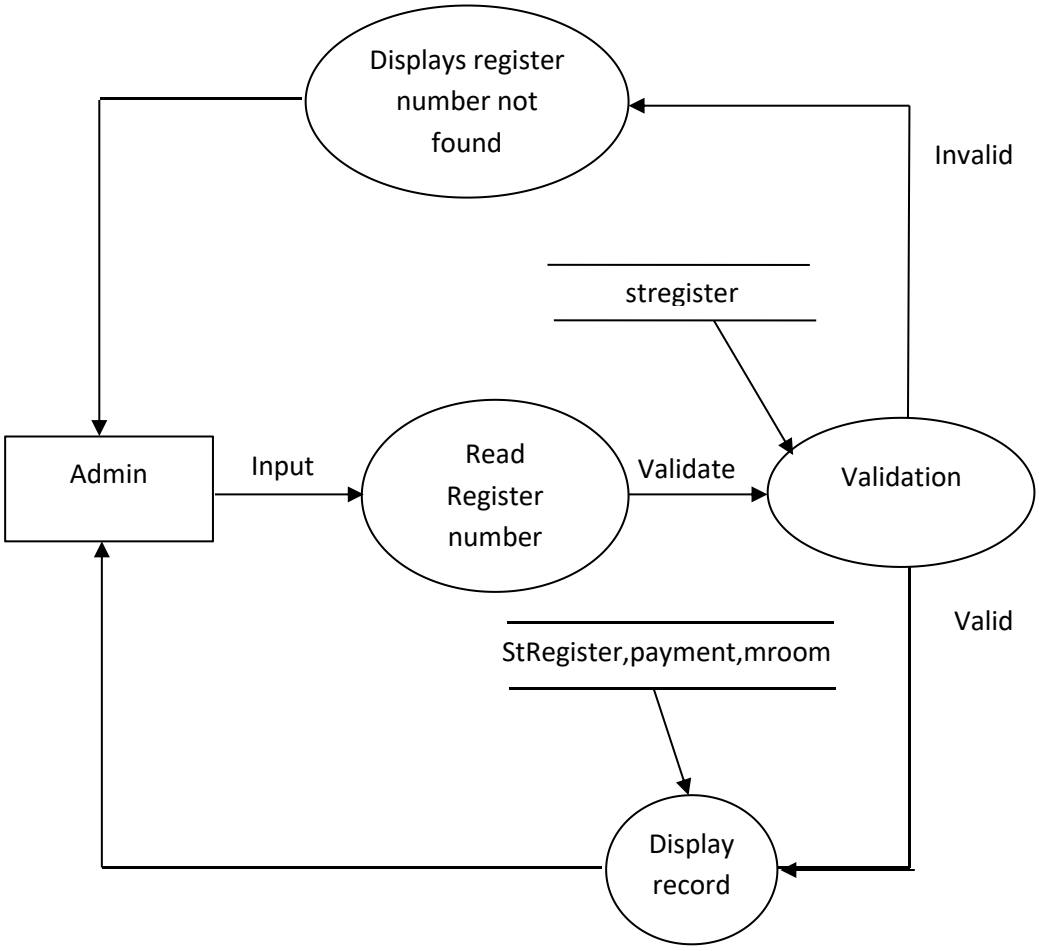


Figure 3.14 Search and view student DFD

3.5.1.3.2.4.1 Input: Register number.

3.5.1.3.2.4.2 Process definition: validates the data.If valid retrieves the details.

3.5.1.3.2.4.3 Output: If found Displays the details else shows error message.

3.5.1.3.2.4.4 Interface with other functional components: Register, Addroom.

3.5.1.3.2.4.5 Resource Allocation: StRegister, payment.

3.5.1.3.2.4.6 User Interface: Button, Textbox and labels.

3.5.1.3.3 Manage wardens

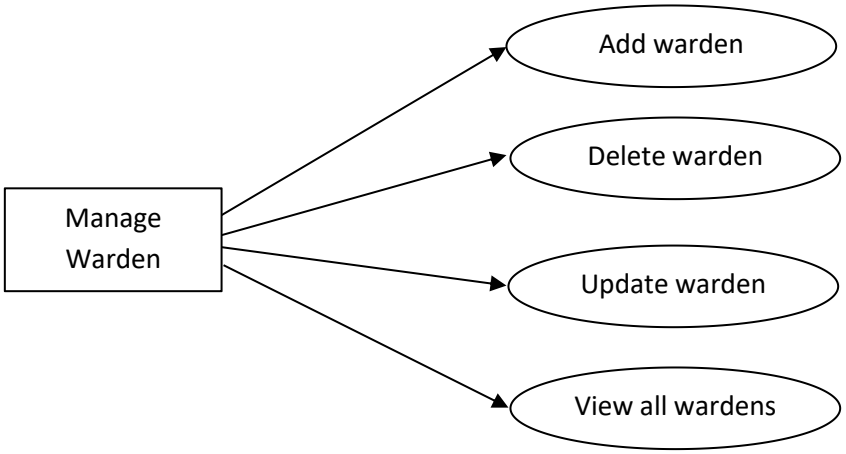


Figure 3.15 Manage wardens DFD

3.5.1.3.3.1 Add warden

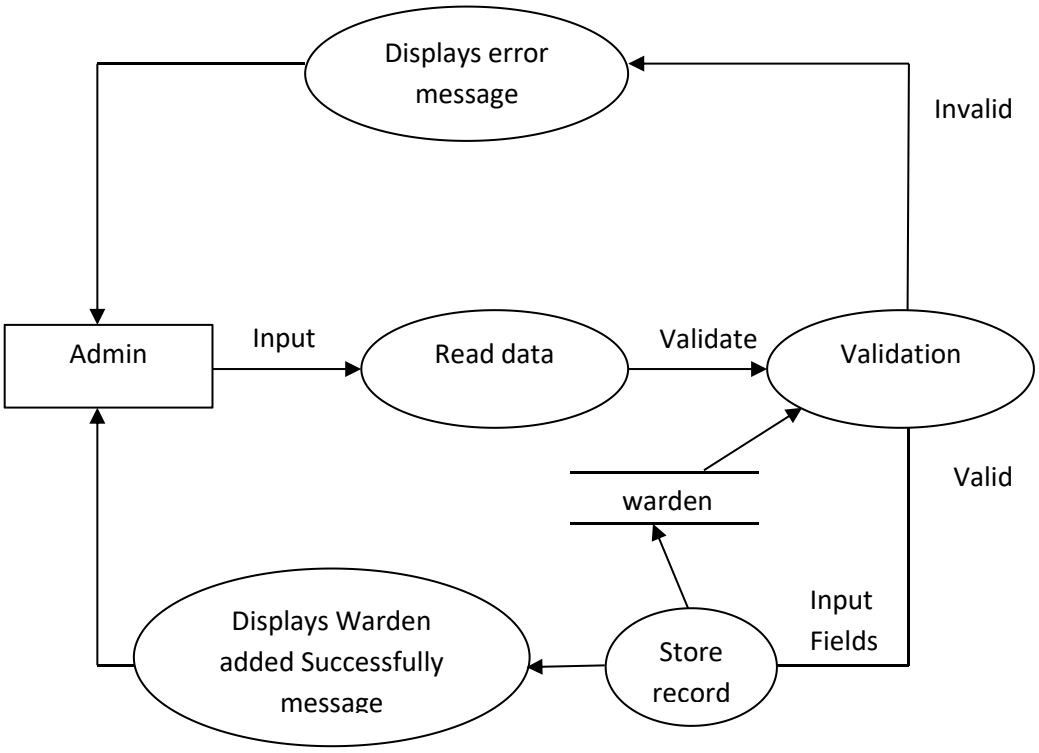


Figure 3.16 Add Warden DFD

3.5.1.3.3.1.1 Input: Warden id, name, address, aadhar number, age, email, gender, phone number.

3.5.1.3.3.1.2 Process definition: Read the data, if valid stores data to the table.

3.5.1.3.3.1.3 Output: if valid Displays "warden added successfully" else shows error message with description.

3.5.1.3.3.1.4 Interface with other functional components: Independent module.

3.5.1.3.3.1.5 Resource Allocation: Warden table.

3.5.1.3.3.1.6 User Interface: Button, Label, Textbox, Drop-down.

3.5.1.3.3.2 Delete warden

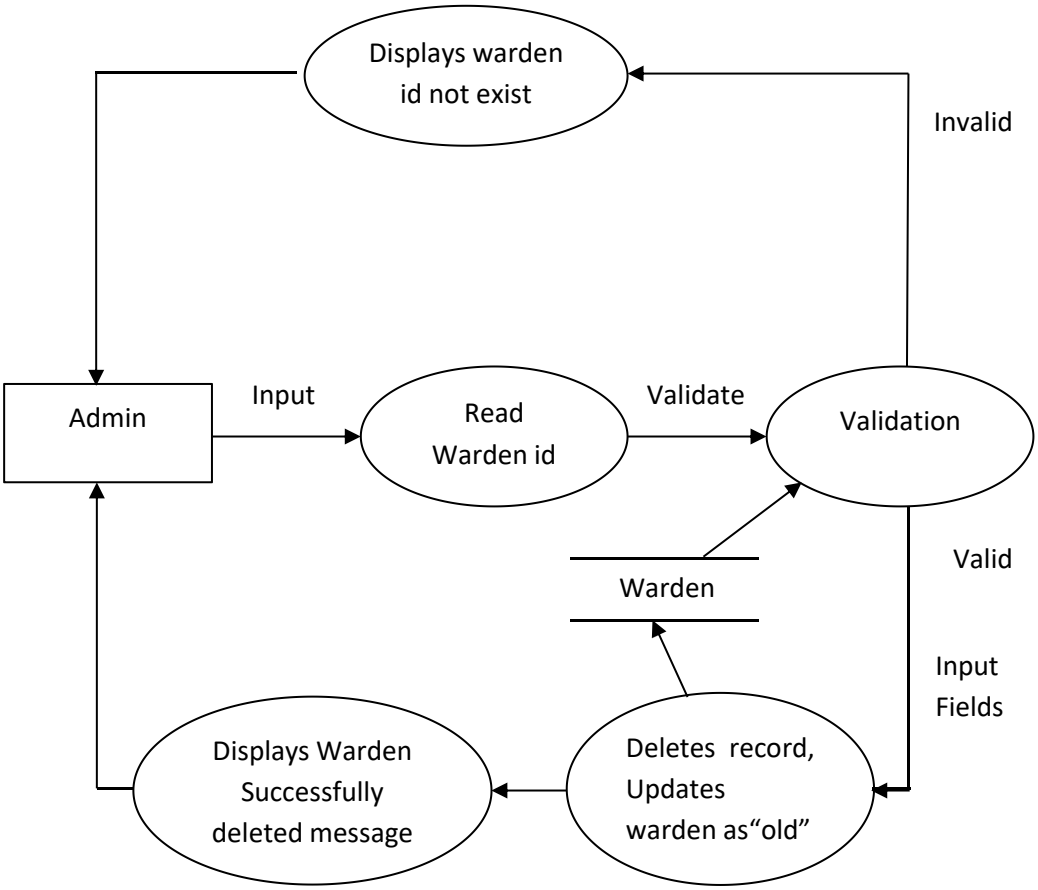


Figure 3.17 Delete warden DFD

3.5.1.3.3.2.1 Input: Warden id.

3.5.1.3.3.2.2 Process definition: reads the data if valid updates the warden as “old” to the table.

3.5.1.3.3.2.3 Output: if valid Displays “record deleted successfully” else shows error message with description.

3.5.1.3.3.2.4 Interface with other functional components: Add warden.

3.5.1.3.3.2.5 Resource Allocation: warden table.

3.5.1.3.3.2.6 User Interface: Button, Label, Textbox.

3.5.1.3.3.3 Update warden

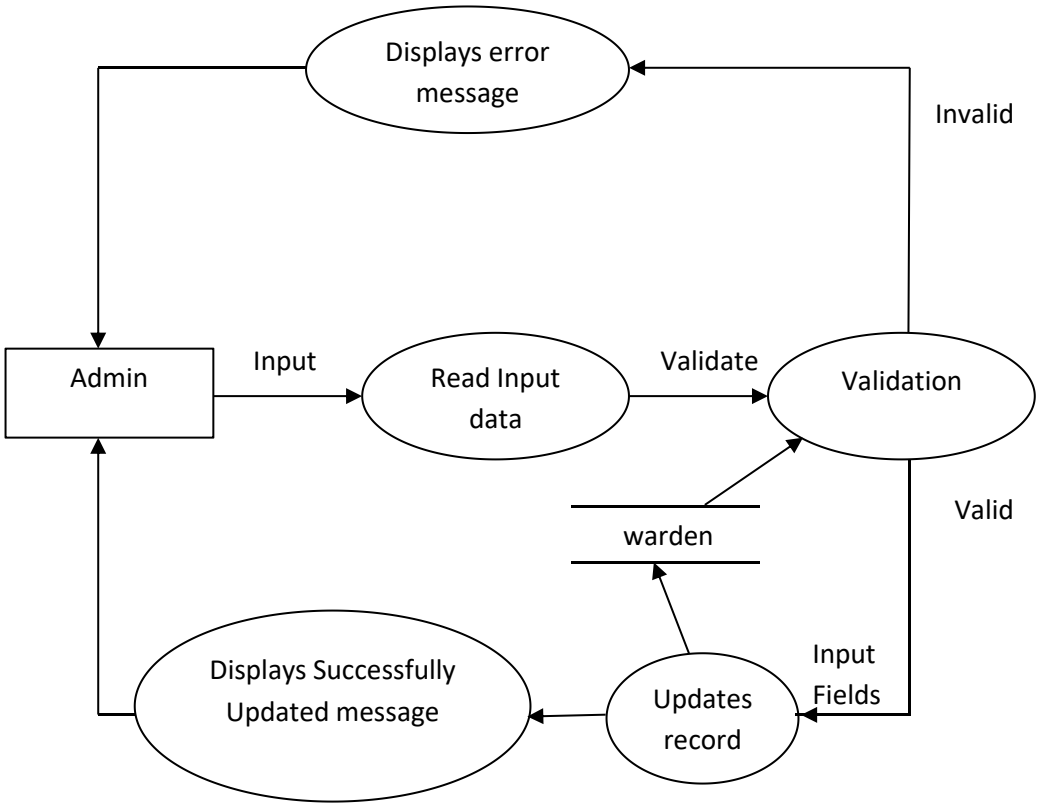


Figure 3.18 Update Warden DFD

3.5.1.3.3.3.1 Input: name, address, aadhar number, age, email, gender, phone number.

3.5.1.3.3.3.2 Process definition: validate the data, if valid updates the record to the table.

3.5.1.3.3.3.3 Output: if valid displays "record updated successfully" else shows error message with description.

3.5.1.3.3.3.4 Interface with other functional components: Add warden.

3.5.1.3.3.3.5 Resource Allocation: warden table.

3.5.1.3.3.3.6 User Interface: Button, Label, Textbox, Drop-down.

3.5.1.3.3.4 View all warden

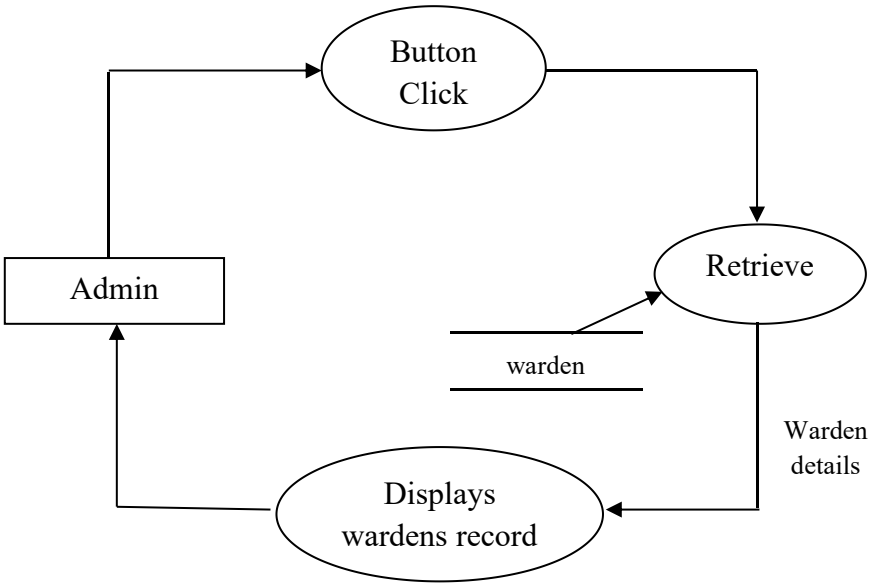


Figure 3.19 View warden DFD

3.5.1.3.3.4.1 Input: Button click.

3.5.1.3.3.4.2 Process definition: Retrieves details from table.

3.5.1.3.3.4.3 Output: Displays details of all warden.

3.5.1.3.3.4.4 Interface with other functional components: Add warden.

3.5.1.3.3.4.5 Resource Allocation: warden table.

3.5.1.3.3.4.6 User Interface: Button, Textbox and labels.

3.5.1.3.4 Manage room

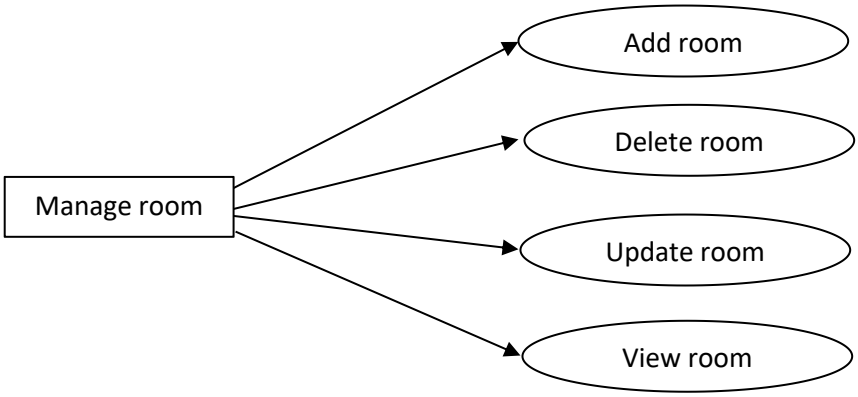


Figure 3.20 Manage room DFD

3.5.1.3.4.1 Add room

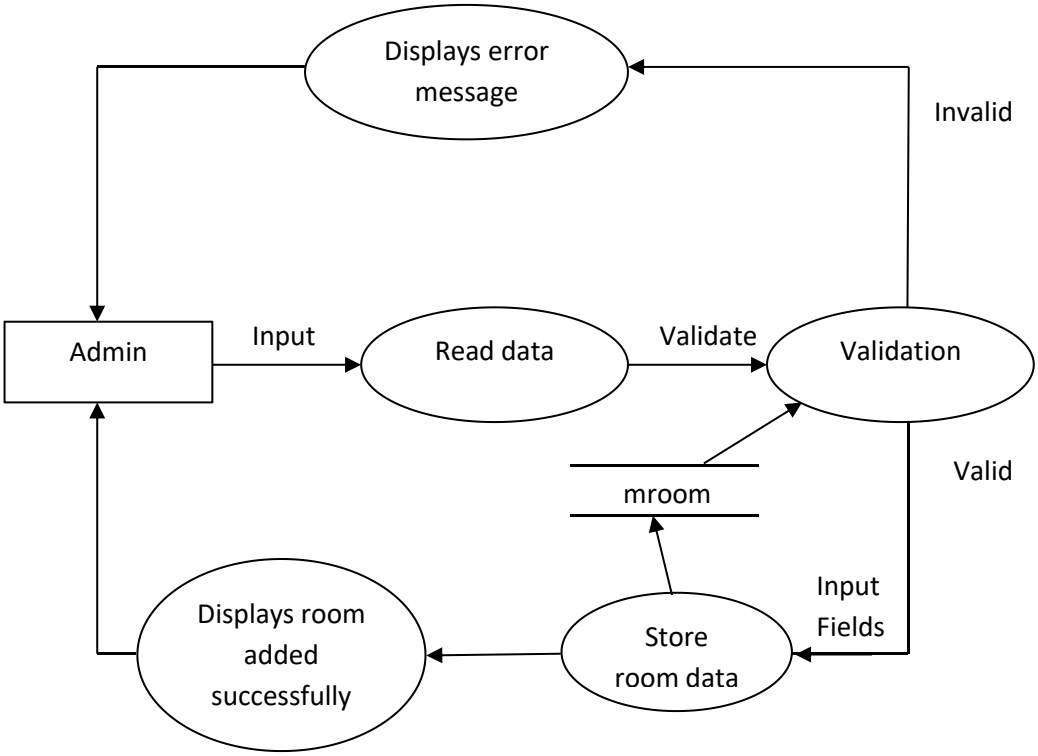


Figure 3.21 Add room DFD

3.5.1.3.4.1.1 Input: Room id, gender, status, seater .

3.5.1.3.4.1.2 Process definition: Reads the data and validates. If valid insert record into the table.

3.5.1.3.4.1.3 Output: if valid Displays “room added successfully” else shows error message.

3.5.1.3.4.1.4 Interface with other functional components: Independent module.

3.5.1.3.4.1.5 Resource Allocation: Mroom table.

3.5.1.3.4.1.6 User Interface: Button, Label, Textbox, Drop-down.

3.5.1.3.4.2 Delete room

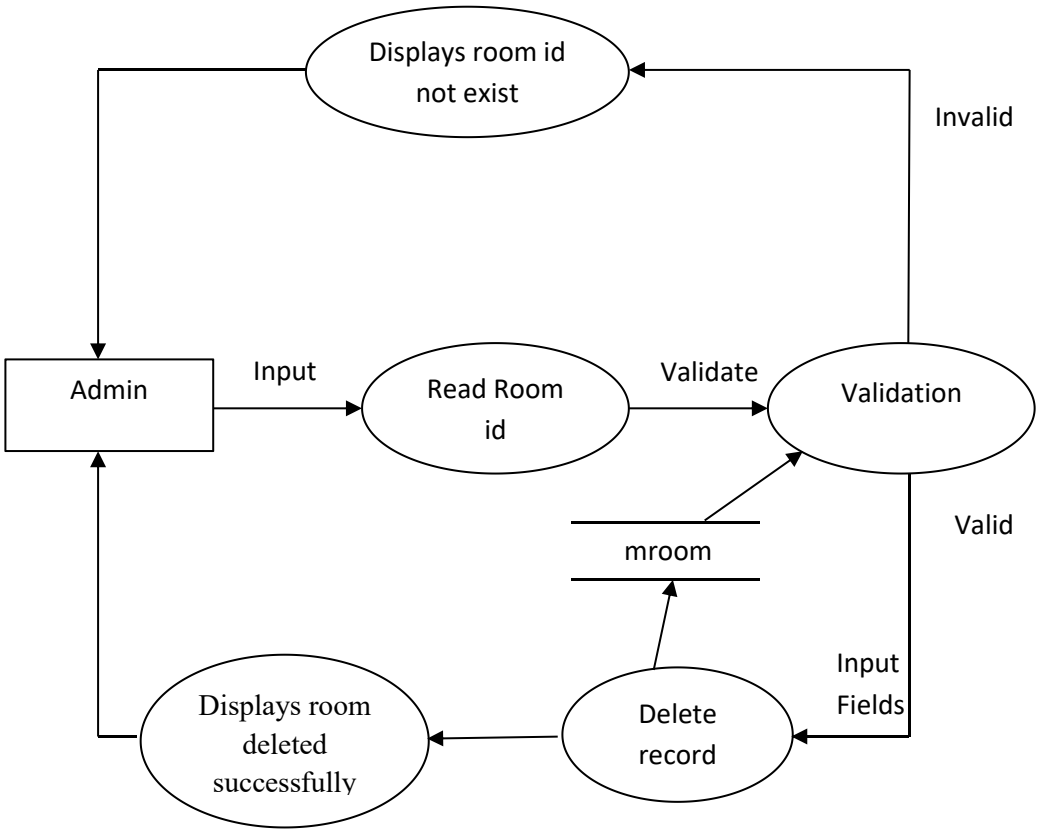


Figure 3.22 Delete room DFD

3.5.1.3.4.2.1 Input: Room id.

3.5.1.3.4.2.2 Process definition: validates the data if valid, deletes the record from the table.

3.5.1.3.4.2.3 Output: if valid Displays “room deleted successfully” else shows error message with description.

3.5.1.3.4.2.4 Interface with other functional components: Add room.

3.5.1.3.4.2.5 Resource Allocation: Mroom table.

3.5.1.3.4.2.6 User Interface: Button, Label, Textbox.

3.5.1.3.4.3 Update room

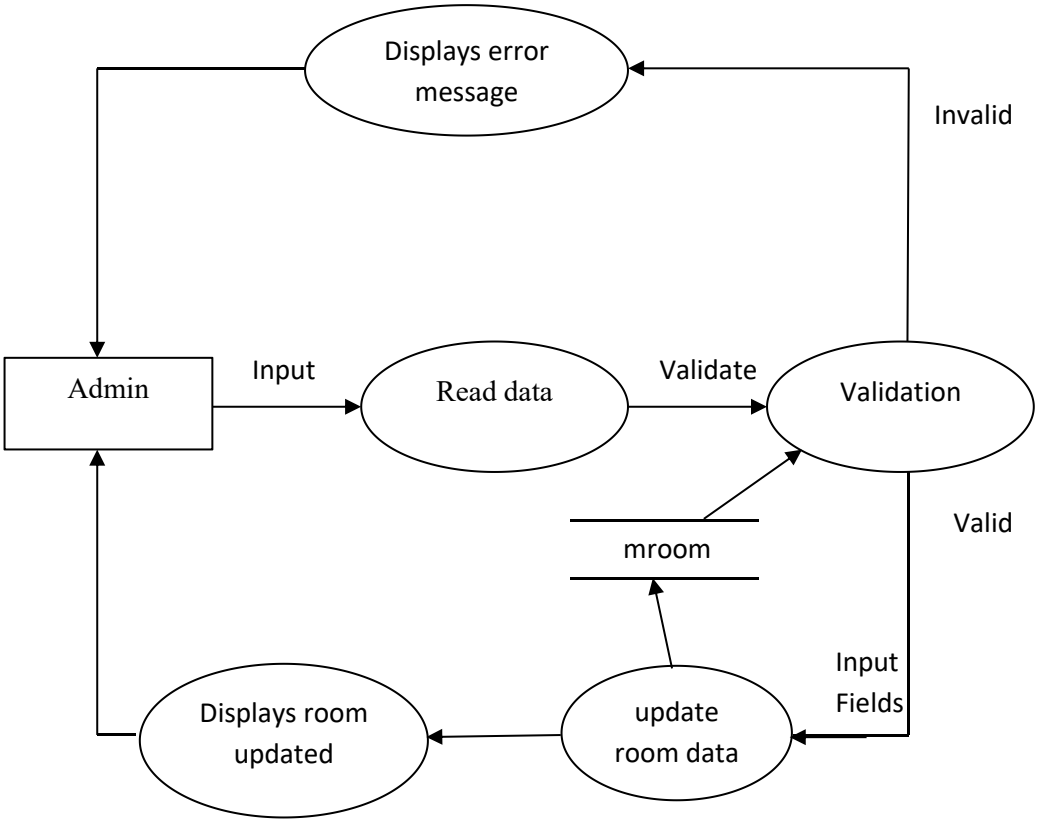


Figure 3.23 Update room DFD

3.5.1.3.4.3.1 Input: gender, status, seater .

3.5.1.3.4.3.2 Process definition: validates the data if valid updates the record to the table.

3.5.1.3.4.3.3 Output: if valid Displays “room updated successfully” else shows error message with description.

3.5.1.3.4.3.4 Interface with other functional components: Add room.

3.5.1.3.4.3.5 Resource Allocation: Mroom table.

3.5.1.3.4.3.5 User Interface: Button, Label, Textbox, Drop-down.

3.5.1.3.4.4 View all room

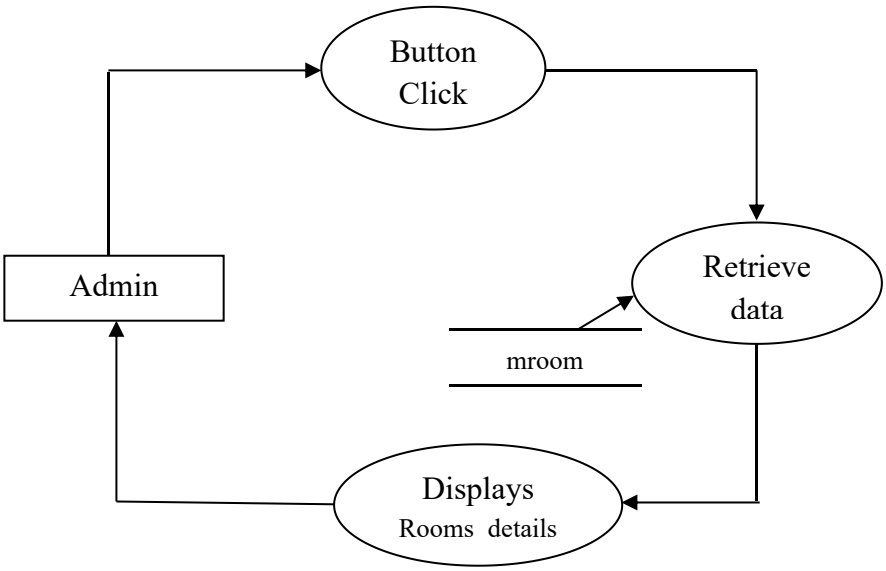


Figure 3.24 View all room DFD

3.5.1.3.4.4.1 Input: Button click.

3.5.1.3.4.4.2 Process definition: Retrieve details from table.

3.5.1.3.4.4.3 Output: Displays room details from table.

3.5.1.3.4.4.4 Interface with other functional components: Add room.

3.5.1.3.4.4.5 Resource Allocation: Mroom table.

3.5.1.3.4.4.6 User Interface: Button and labels.

3.5.1.3.5 Manage Food menu

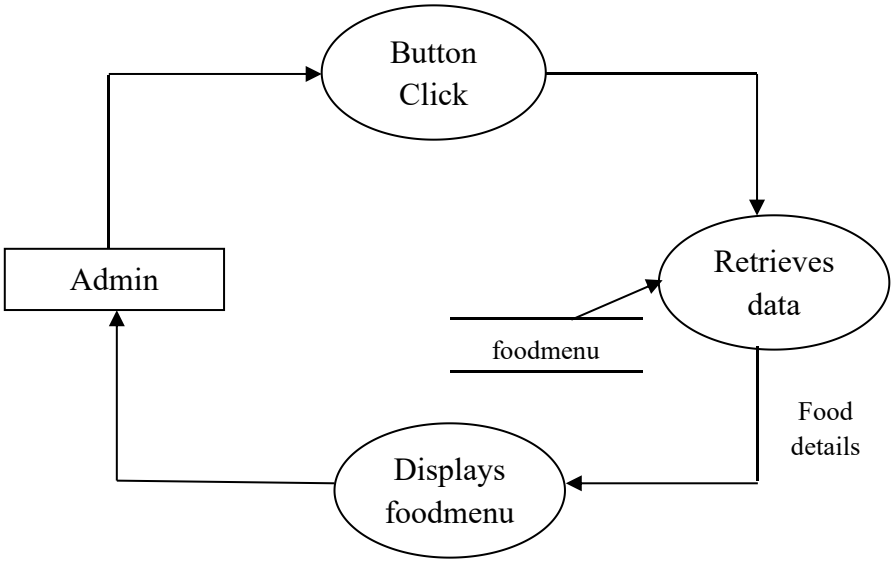


Figure 3.25 Manage food menu DFD

3.5.1.3.5.1 Input: Button click.

3.5.1.3.5.2 Process definition: Retrieve details from table.

3.5.1.3.5.3 Output: Displays food details.

3.5.1.3.5.4 Interface with other functional components: Independent module.

3.5.1.3.5.5 Resource Allocation: foodmenu table.

3.5.1.3.5.6 User Interface: Button, Textbox and labels.

3.5.1.3.5.7 Update foodmenu

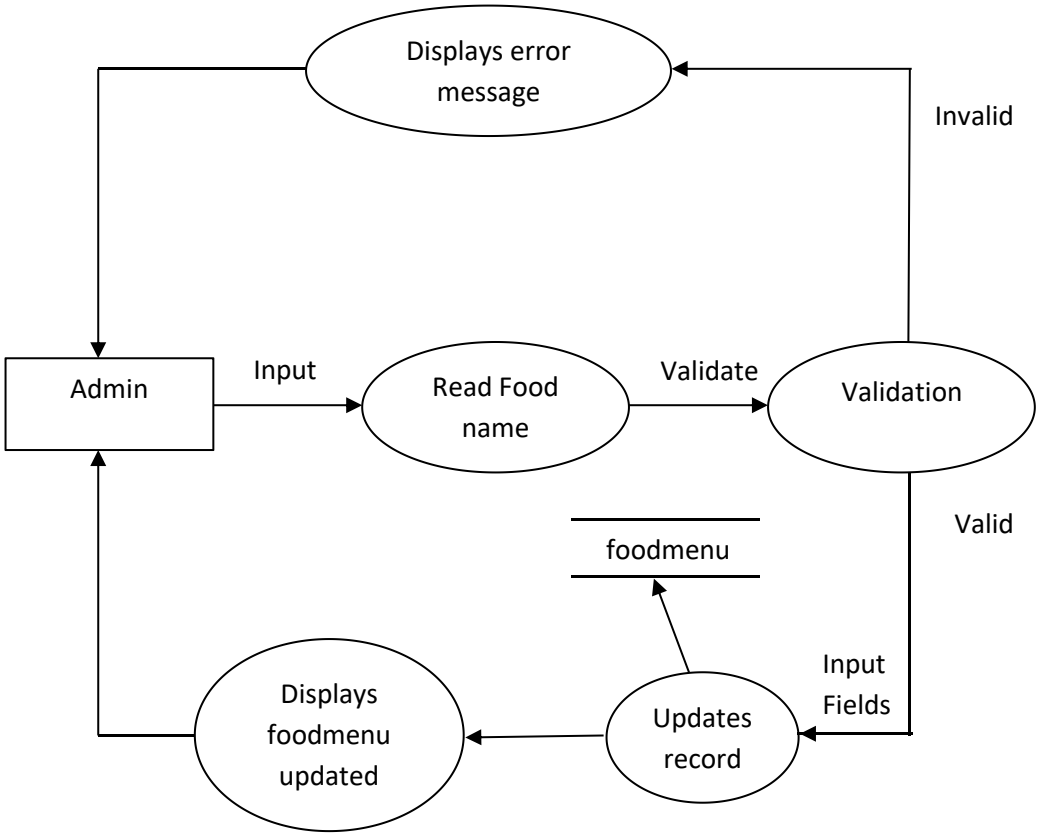


Figure 3.26 Update foodmenu DFD

3.5.1.3.5.7.1 Input: Food names.

3.5.1.3.5.7.2 Process definition: validates the data , if valid updates the record to the table.

3.5.1.3.5.7.3 Output: If valid Displays "foodmenu updated successfully" else shows error message.

3.5.1.3.5.7.4 Interface with other functional components: Manage food menu.

3.5.1.3.5.7.5 Resource Allocation: foodmenu table.

3.5.1.3.5.7.6 User Interface: Button, Label, Textbox.

3.5.1.3.6 Manage event

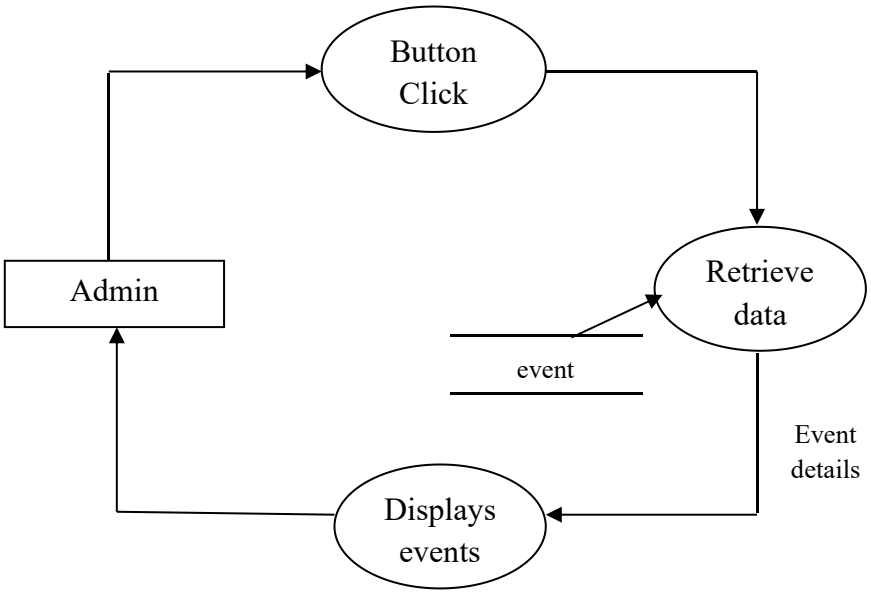


Figure 3.27 Manage event DFD

3.5.1.3.6.1 Input: Button click.

3.5.1.3.6.2 Process definition: Retrieves details from table.

3.5.1.3.6.3 Output: Displays events details .

3.5.1.3.6.4 Interface with other functional components: Independent module.

3.5.1.3.6.5 Resource Allocation: event table.

3.5.1.3.6.6 User Interface: Button, Textbox and labels.

3.5.1.3.6.7 Upload event

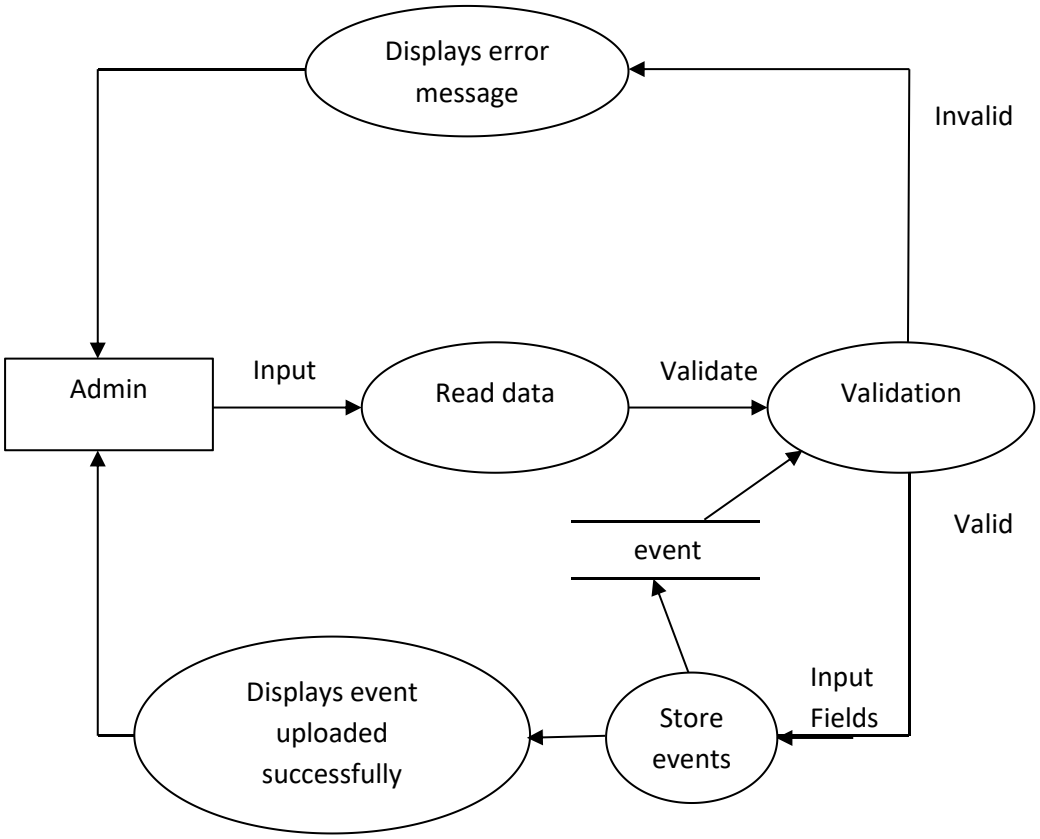


Figure 3.28 Upload event DFD

3.5.1.3.6.7.1 Input: Image file, event description.

3.5.1.3.6.7.2 Process definition: validates the data, if valid inserts the record to the table.

3.5.1.3.6.7.3 Output: if valid Displays “event uploaded successfully” else shows error message with description.

3.5.1.3.6.7.4 Interface with other functional components: Manage event.

3.5.1.3.6.7.5 Resource Allocation: event table.

3.5.1.3.6.7.6 User Interface: Button, Label, Textbox.

3.5.1.3.7 View students payment

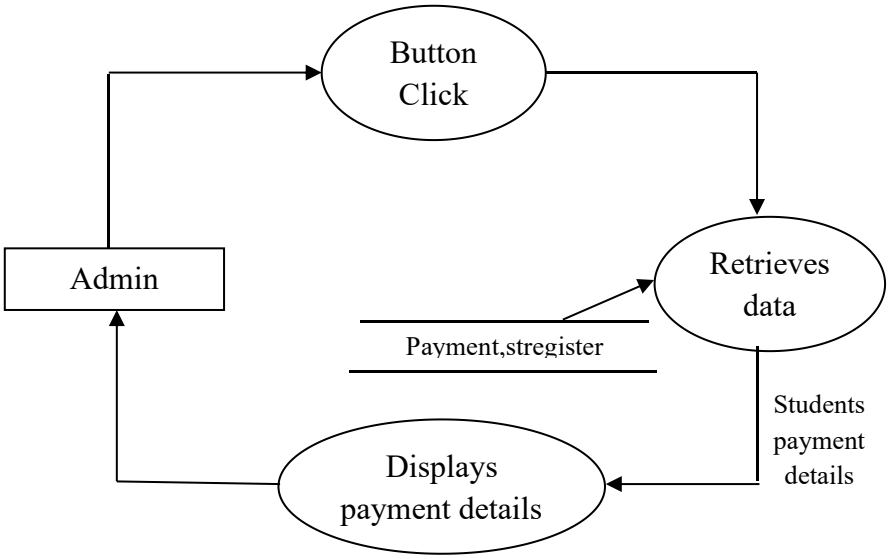


Figure 3.29 View students payment DFD

3.5.1.3.7.1 Input: Button click.

3.5.1.3.7.2 Process definition: Retrieve details from table.

3.5.1.3.7.3 Output: Displays payment details of the student.

3.5.1.3.7.4 Interface with other functional components: register, meal price, renewal.

3.5.1.3.7.5 Resource Allocation: stregister ,payment table.

3.5.1.3.7.6 User Interface: Button, Textbox and labels.

3.5.1.4 Warden

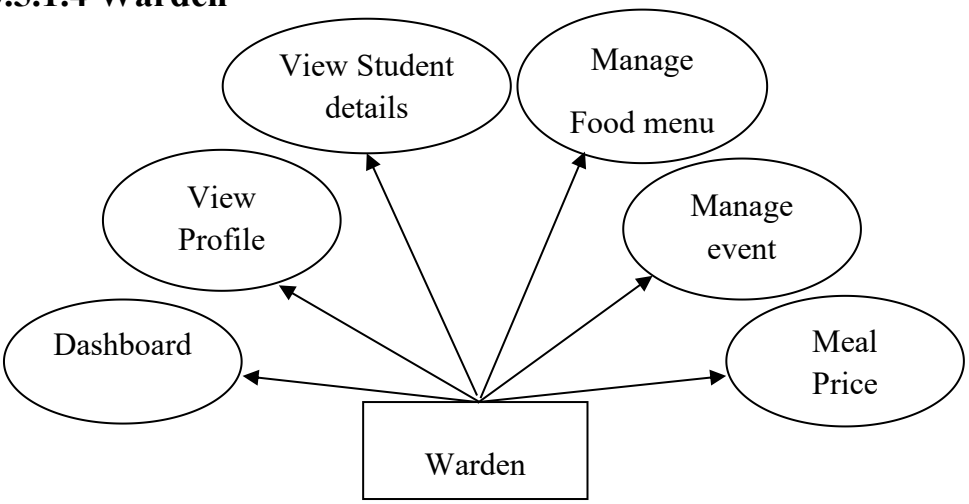


Figure 3.30 Warden DFD

3.5.1.4.1 Dashboard

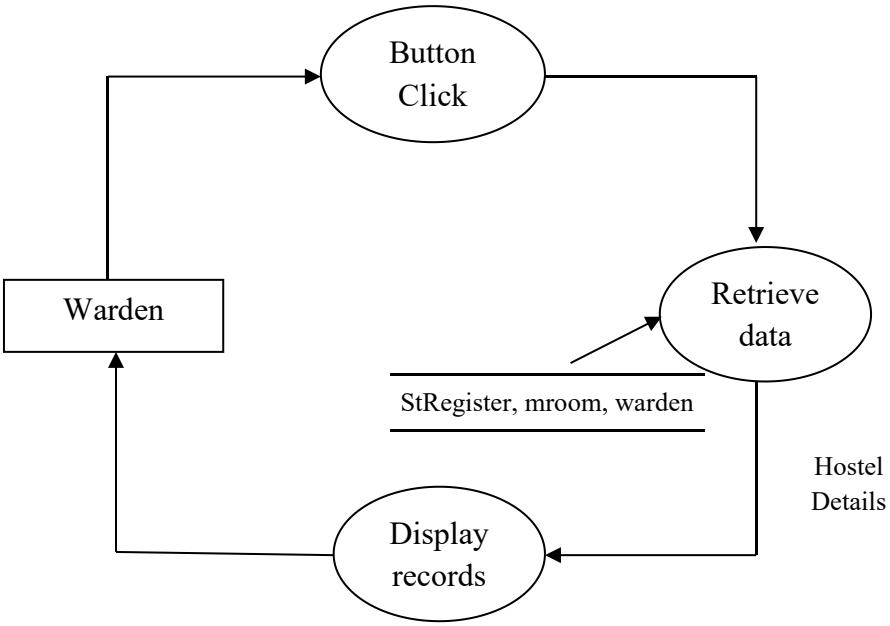


Figure 3.31 Dashboard DFD

3.5.1.4.1.1 Input: Button click.

3.5.1.4.1.2 Process definition: Retrieves details from tables.

3.5.1.4.1.3 Output: Displays details.

3.5.1.4.1.4 Interface with other functional components: Register, Add warden, Add room.

3.5.1.4.1.5 Resource Allocation: StRegister, mroom, warden tables.

3.5.1.4.1.6 User Interface: Button, Textbox and labels.

3.5.1.4.2 My Profile

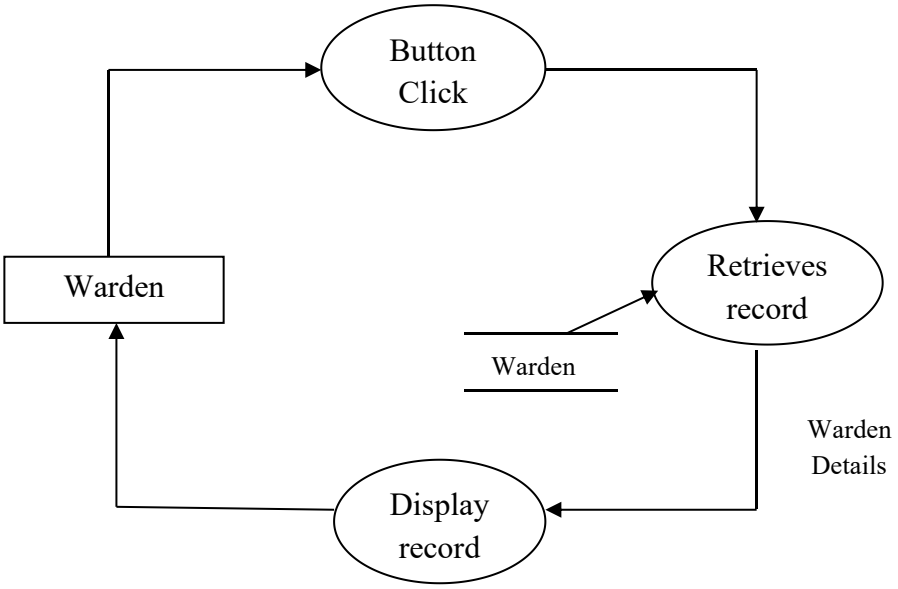


Figure 3.32 My Profile DFD

3.5.1.4.2.1 Input: Button click.

3.5.1.4.2.2 Process definition: Retrieve details from table.

3.5.1.4.2.3 Output: Displays warden details.

3.5.1.4.2.4 Interface with other functional components: Add warden.

3.5.1.4.2.5 Resource Allocation: Warden table.

3.5.1.4.2.6 User Interface: Buttons, Textbox and labels.

3.5.1.4.3 View student details

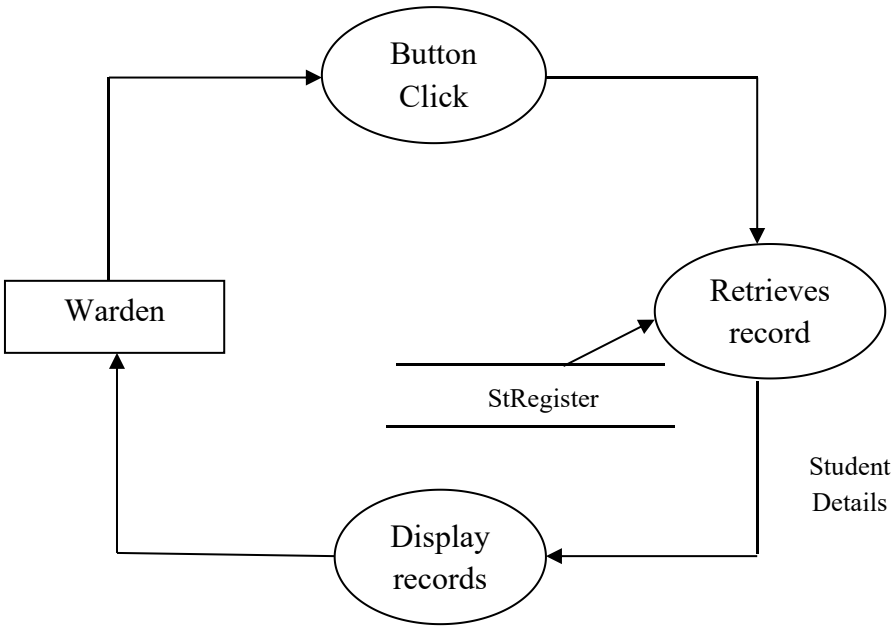


Figure 3.33 View student details DFD

3.5.1.4.3.1 Input: Button click.

3.5.1.4.3.2 Process definition: Retrieve students details from table.

3.5.1.4.3.3 Output: Displays all the student details.

3.5.1.4.3.4 Interface with other functional components: Register.

3.5.1.4.3.5 Resource Allocation: StRegister table.

3.5.1.4.3.6 User Interface: Buttons, Textbox and labels.

3.5.1.4.4 Manage Food menu

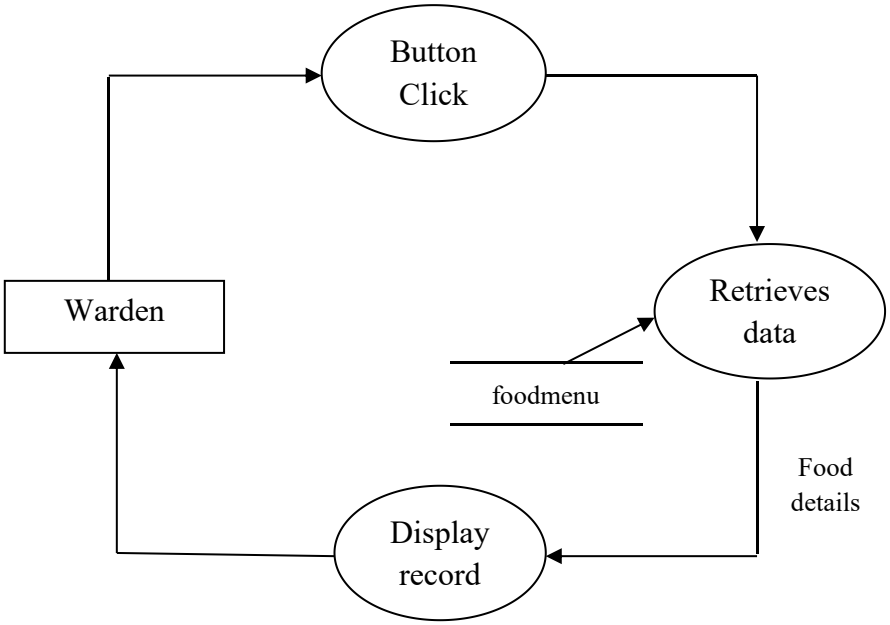


Figure 3.34 Manage food menu DFD

3.5.1.4.4.1 Input: Button click.

3.5.1.4.4.2 Process definition: Retrieves details from table.

3.5.1.4.4.3 Output: Displays food details from table.

3.5.1.4.4.4 Interface with other functional components: Independent module.

3.5.1.4.4.5 Resource Allocation: foodmenu table.

3.5.1.4.4.6 User Interface: Button, Textbox and labels.

3.5.1.4.4.7 Update food menu

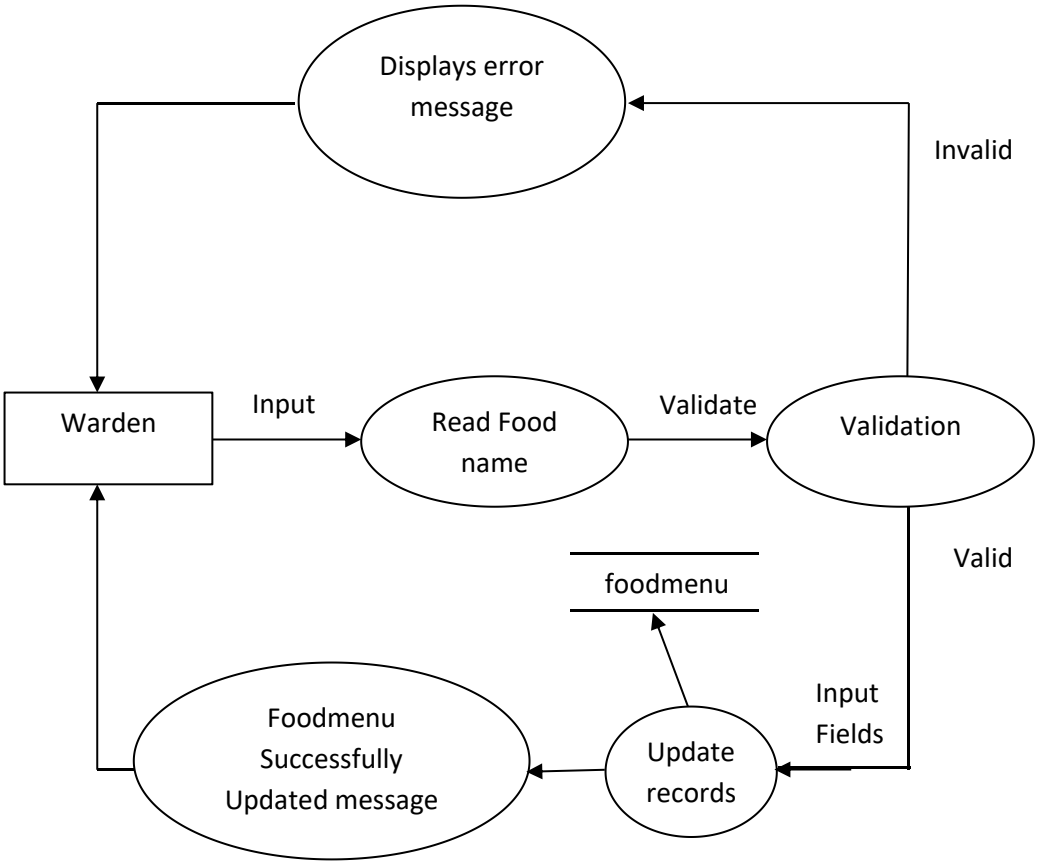


Figure 3.35 Update food menu DFD

3.5.1.4.4.7.1 Input: Food names.

3.5.1.4.4.7.2 Process definition: validates the data if valid updates the record to the table.

3.5.1.4.4.7.3 Output: if valid Displays "food menu successfully updated" else shows error message with description.

3.5.1.4.4.7.4 Interface with other functional components: Manage food menu.

3.5.1.4.4.7.5 Resource Allocation: foodmenu table.

3.5.1.4.4.7.6 User Interface: Button, Label, Textbox.

3.5.1.4.5 Manage event

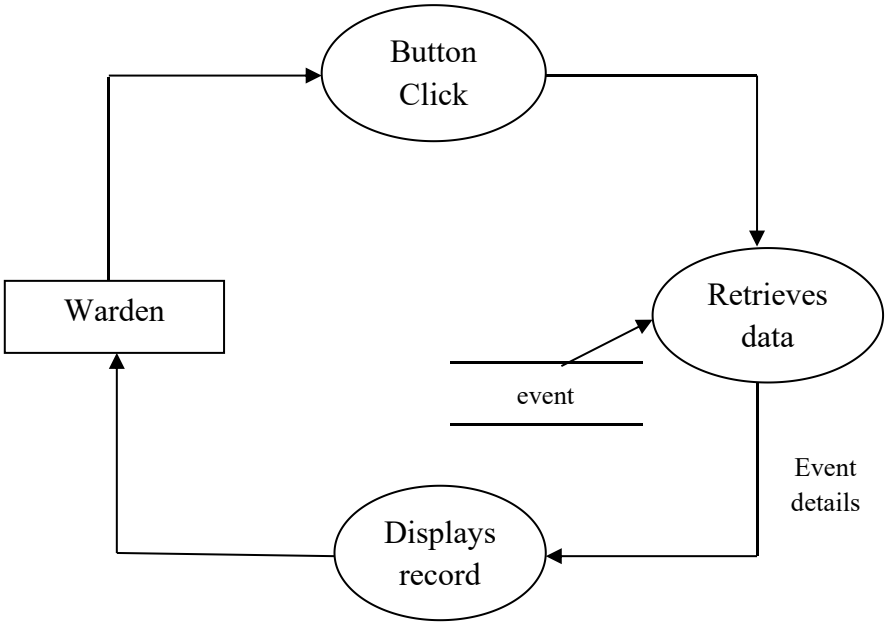


Figure 3.36 manage event DFD

3.5.1.4.5.1 Input: Button click.

3.5.1.4.5.2 Process: Retrieve details from table.

3.5.1.4.5.3 Output: Displays event details from table.

3.5.1.4.5.4 Interface with other functional components: Independent module.

3.5.1.4.5.5 Resource Allocation: event table.

3.5.1.4.5.6 User Interface: Button, Textbox and labels.

3.5.1.4.5.7 Upload event

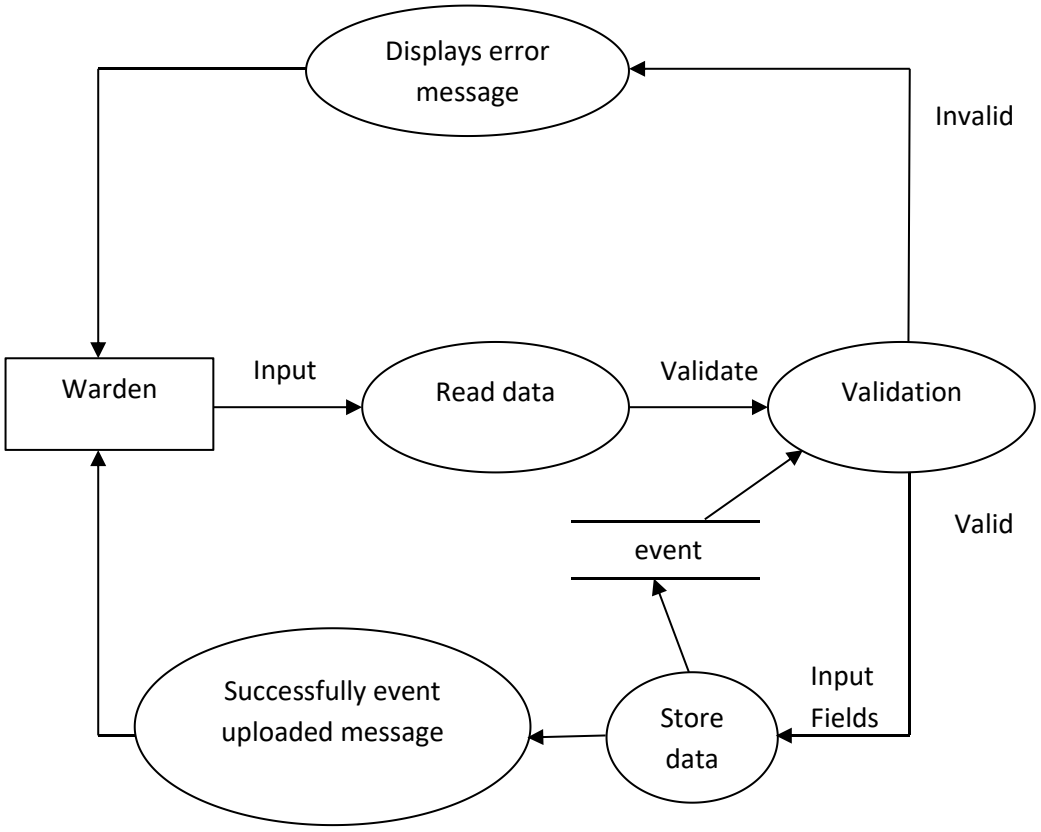


Figure 3.37 Upload event DFD

3.5.1.4.5.7.1 Input: Image file, event description .

3.5.1.4.5.7.2 Process definition: validates the data , if valid inserts the record to the table.

3.5.1.4.5.7.3 Output: if valid Displays “event uploaded successfully” else shows error message with description.

3.5.1.4.5.7.4 Interface with other functional components: Manage event.

3.5.1.4.5.7.5 Resource Allocation: event table.

3.5.1.4.5.7.6 User Interface: Button, Label, Textbox.

3.5.1.4.6 Meal price

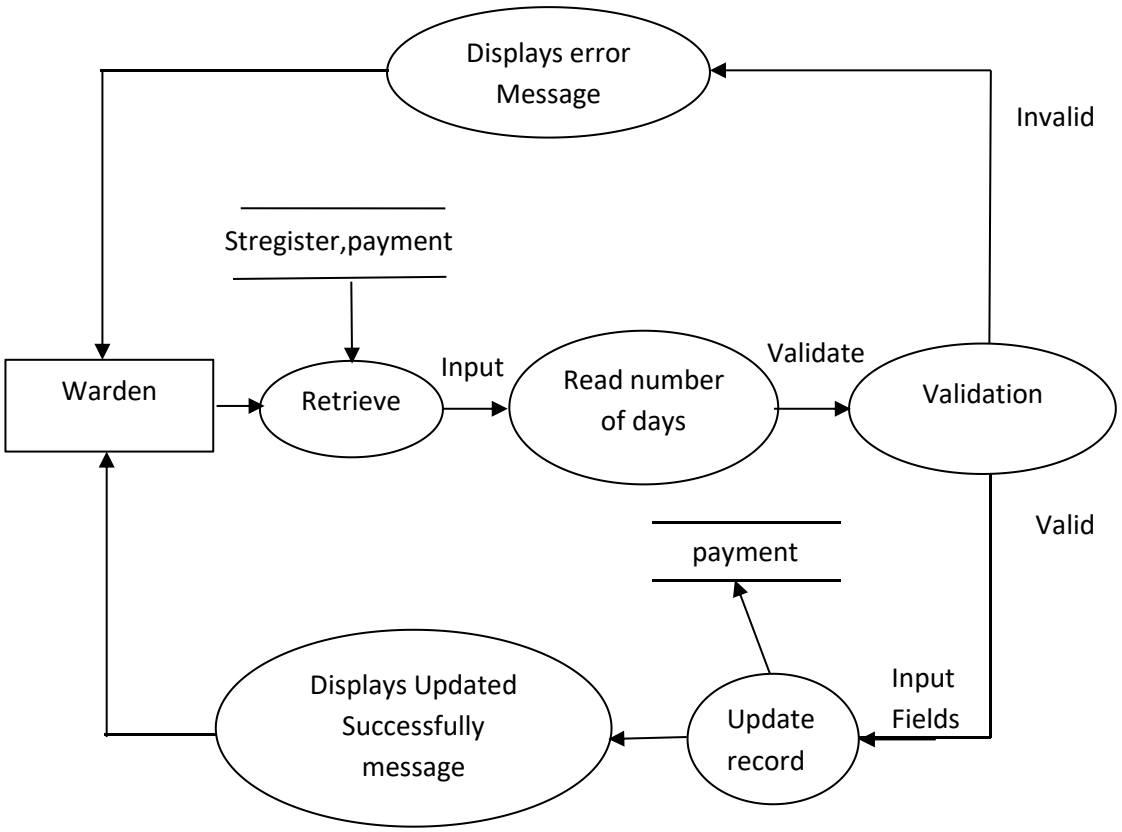


Figure 3.38 Meal price DFD

3.5.1.4.6.1 Input: Number of days.

3.5.1.4.6.2 Process definition: retrieves details and validates the data , if valid updates the month meal price of the student to the table.

3.5.1.4.6.3 Output: if valid Displays “updated successfully” else shows error message with description.

3.5.1.4.6.4 Interface with other functional components: Register ,add warden.

3.5.1.4.6.5 Resource Allocation: stregister, payment table.

3.5.1.4.5.7.6 User Interface: Button, Label, Textbox.

3.5.1.5 Student

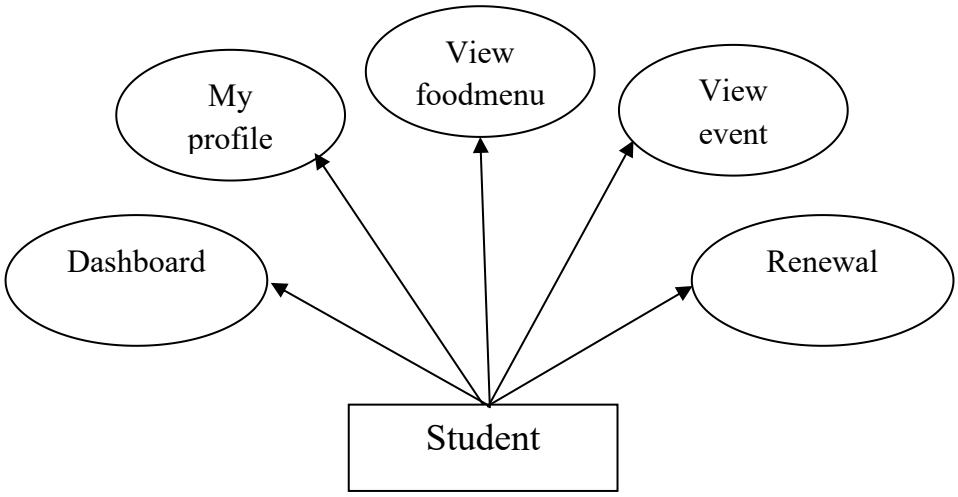


Figure 3.39 Student DFD

3.5.1.5.1 Dashboard

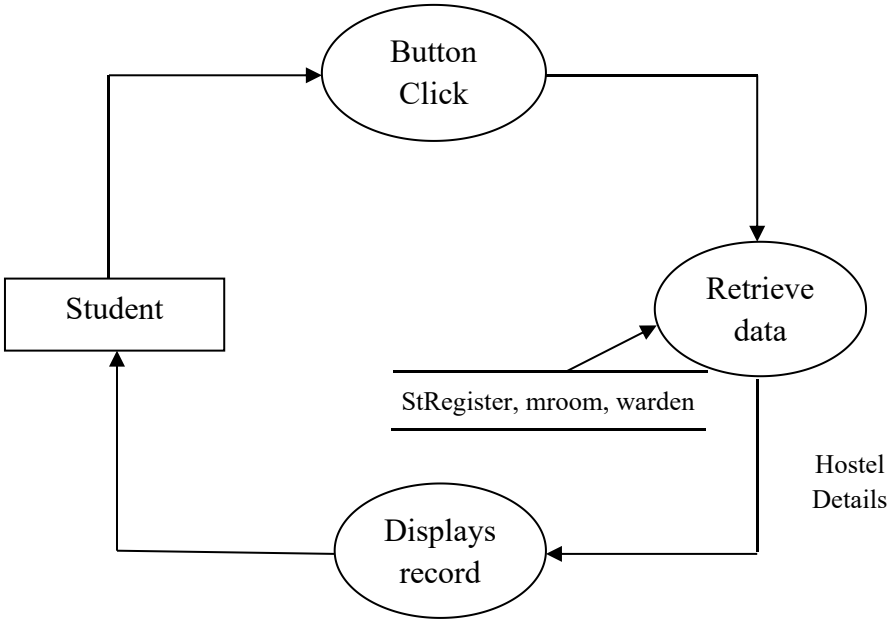


Figure 3.40 Dashboard DFD

3.5.1.5.1.1 Input: button click.

3.5.1.5.1.2 Process definition: Retrieve details from tables.

3.5.1.5.1.3 Output: Displays details.

3.5.1.5.1.4 Interface with other functional components: Register, Add warden, Add room.

3.5.1.5.1.5 Resource Allocation: StRegister, mroom, warden tables.

3.5.1.5.1.6 User Interface: Button and labels.

3.5.1.5.2 My profile

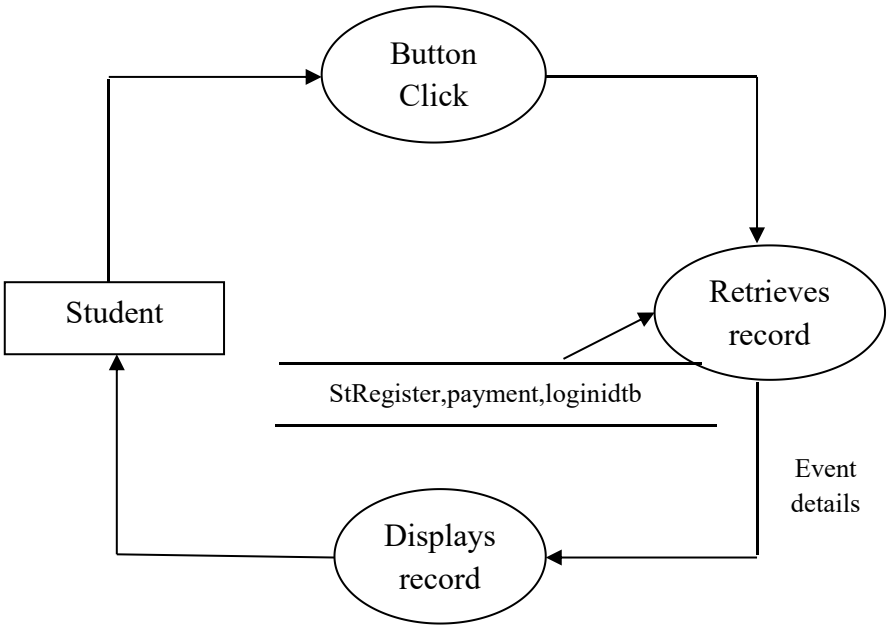


Figure 3.41 My Profile DFD

3.5.1.5.2.1 Input: Button click.

3.5.1.5.2.2 Process definition: Retrieves details from tables.

3.5.1.5.2.3 Output: Displays details from tables.

3.5.1.5.2.4 Interface with other functional components: Register, Add room.

3.5.1.5.2.5 Resource Allocation: StRegister,payment, loginidtb tables.

3.5.1.5.2.6 User Interface: Button, Textbox and labels.

3.5.1.5.2.7 Update my profile

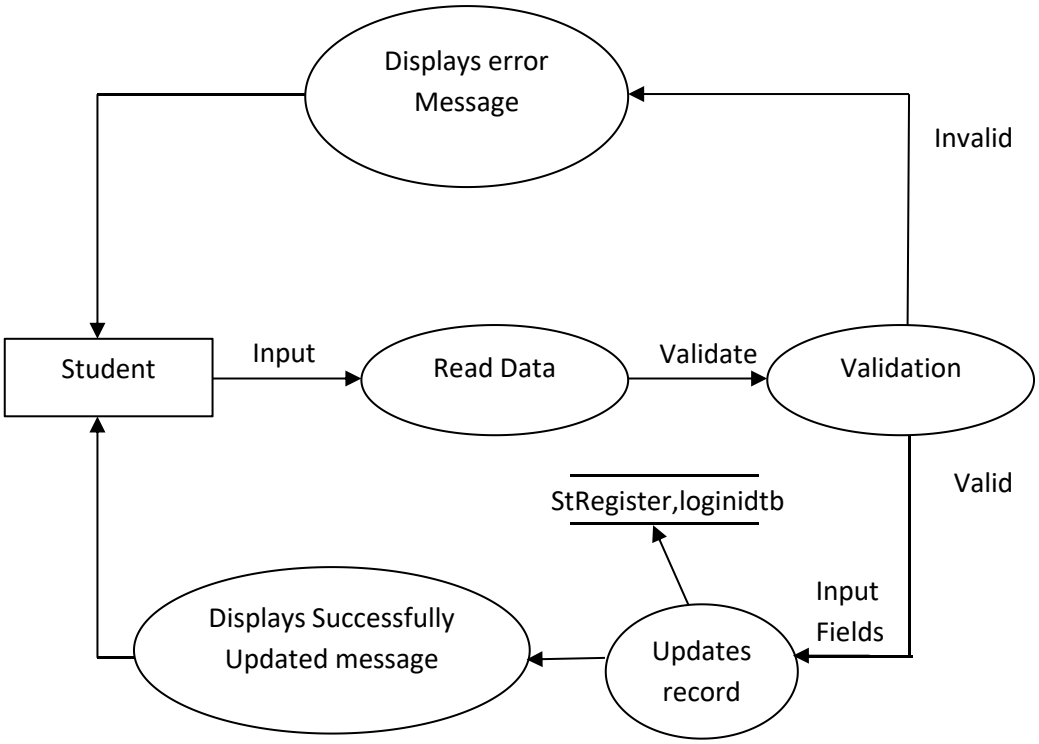


Figure 3.42 Update my profile DFD

3.5.1.5.2.7.1 Input: College name, Roll no, Course ,course duration,previous year mark,address,password.

3.5.1.5.2.7.2 Process definition: Reads data. If valid updates the record to the table.

3.5.1.5.2.7.3 Output: if valid Displays "successfully profile Updated" message else shows error message with description.

3.5.1.5.2.7.4 Interface with other functional components: Register, my profile.

3.5.1.5.2.7.5 Resource Allocation: StRegister loginidtb table.

3.5.1.5.2.7.6 User Interface: Button, Textbox and labels.

3.5.1.5.3 View Food Menu

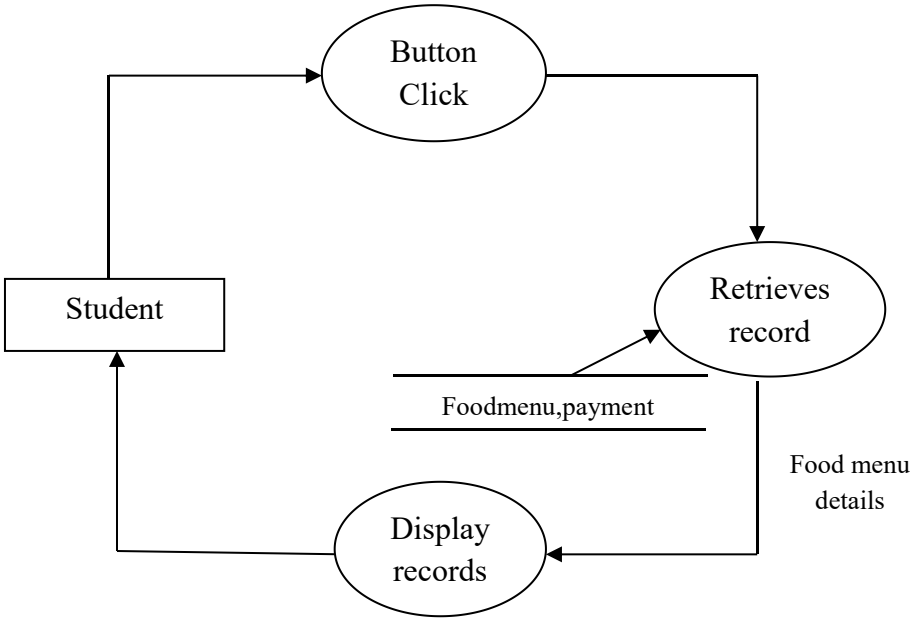


Figure 3.43 View food menu DFD

3.5.1.5.3.1 Input: Button click.

3.5.1.5.3.2 Process definition: Retrieve details from table.

3.5.1.5.3.3 Output: Displays food details and month meal price from table.

3.5.1.5.3.4 Interface with other functional components: Manage food menu.

3.5.1.5.3.5 Resource Allocation: foodmenu ,payment table.

3.5.1.5.3.6 User Interface: Button, Textbox and labels.

3.5.1.5.4 View events

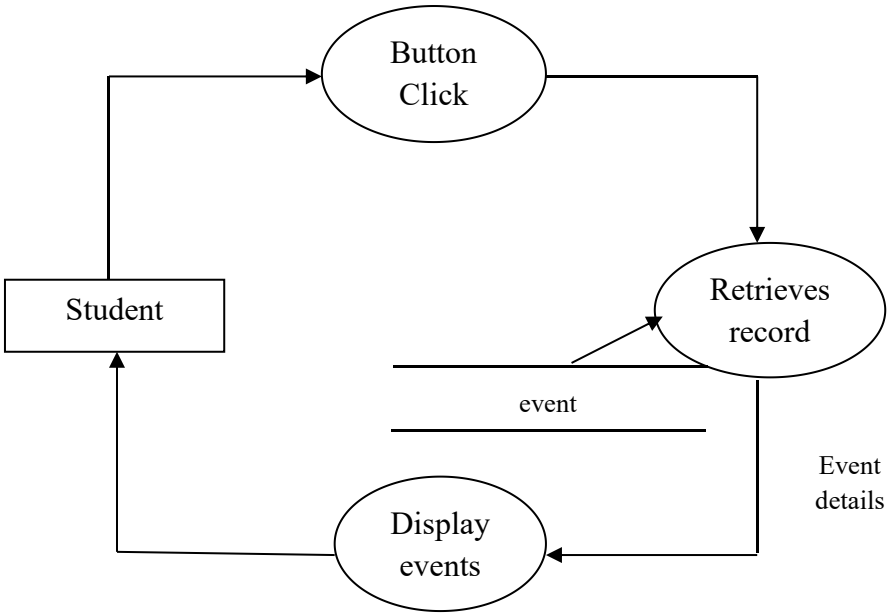


Figure 3.44 View Event DFD

3.5.1.5.4.1 Input: Button click.

3.5.1.5.4.2 Process definition: Retrieve details from table.

3.5.1.5.4.3 Output: Displays event details from table.

3.5.1.5.4.4 Interface with other functional components: Manage event.

3.5.1.5.4.5 Resource Allocation: event table.

3.5.1.5.4.6 User Interface: Button, Textbox and labels.

3.5.1.5.5 Renewal

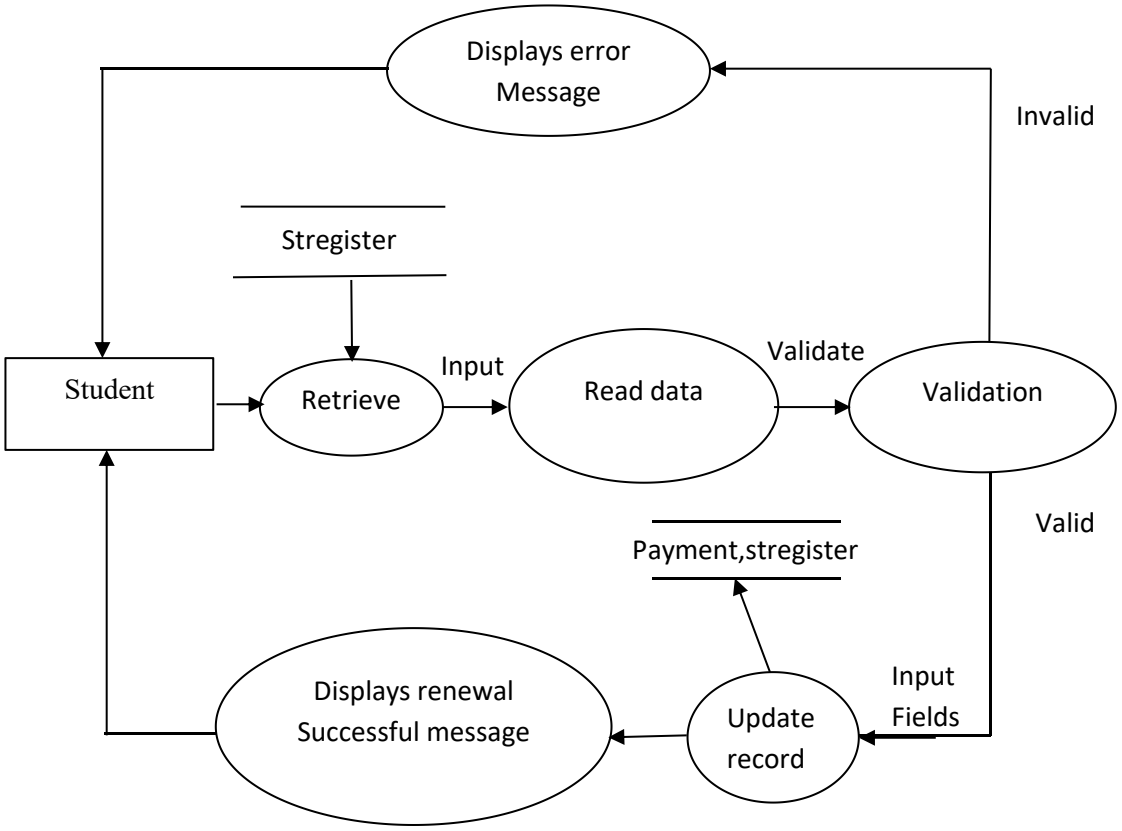


Figure 3.45 Renewal DFD

3.5.1.5.5.1 Input: card number, cardholder name, expiry year and month, cvv, extended month.

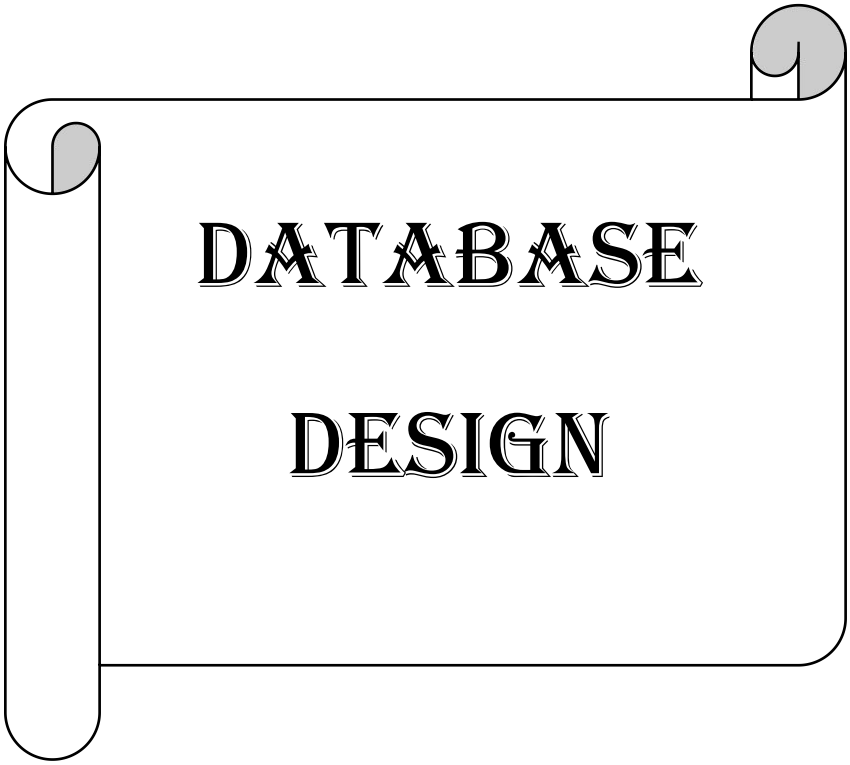
3.5.1.5.5.2 Process definition: Retrieves data and validates input ,if valid, Updates the details to the table.

3.5.1.5.5.3 Output: if valid Displays “renewal successful” else shows error message.

3.5.1.5.5.4 Interface with other functional components: Register.

3.5.1.5.5.5 Resource Allocation: stregister, payment table.

3.5.1.5.5.6 User Interface: Button, Textbox and labels.


DATABASE
DESIGN

4. DATABASE DESIGN

4.1 Introduction

Database is a collection of related data. Relational database stores data in a table or relations. The data stored in a relation are arranged in records. Each record consists of set of attributes. Fields can be referred to characteristics of records. This document describes the table that is used to design software, its attributes, data types, constraints and relationship among those tables.

The design process consists of the following steps:

- Determine the purpose of your database.
- Find and organize the information required.
- Divide the information into tables.
- Turn information items into columns..
- Specify primary keys.
- Set up the table relationships. ...
- Refine your design.
- Apply the normalization rules.

4.2 Purpose and scope

Purpose

- **Avoid Redundant Data**

  The table in the database should be constructed following standards and with utmost dedication. It should have different fields and minimize redundant data. The table should always have a Primary Key that would be a unique id.

- **Faultless Information**

  The database should follow the standards and conventions and provide meaningful information useful to the organization.

- **Data Integrity**

  Integrity assists in guaranteeing that the values are valid and faultless. Data Integrity is set to tables, relationships, etc.

- **Modify**

  The database developed should be worked upon with the conventions and standards, so that it can be easily modified whenever the need arise.

- **Scope**
  - Normalization of Database.
  - Imposing Integrity Constraint.
  - Establishing the Relation between the tables.
  - Accessing the data from multiple tables.

4.3 Database Identification

The identification of database by unique name given to the various database objects. The identifier is the name of database object. The following are the various database objects.

- Tables
- Columns
- Views
- Sequences
- Indexes
- Stored Procedures

4.4 Schema information

The database schema is its structure described in a formal language supported by the database management system (DBMS). The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases).

In a relational database, the schema defines the tables, fields, relationships, views, indexes, packages, procedures, function, queues, triggers, types, sequences, materialized views.

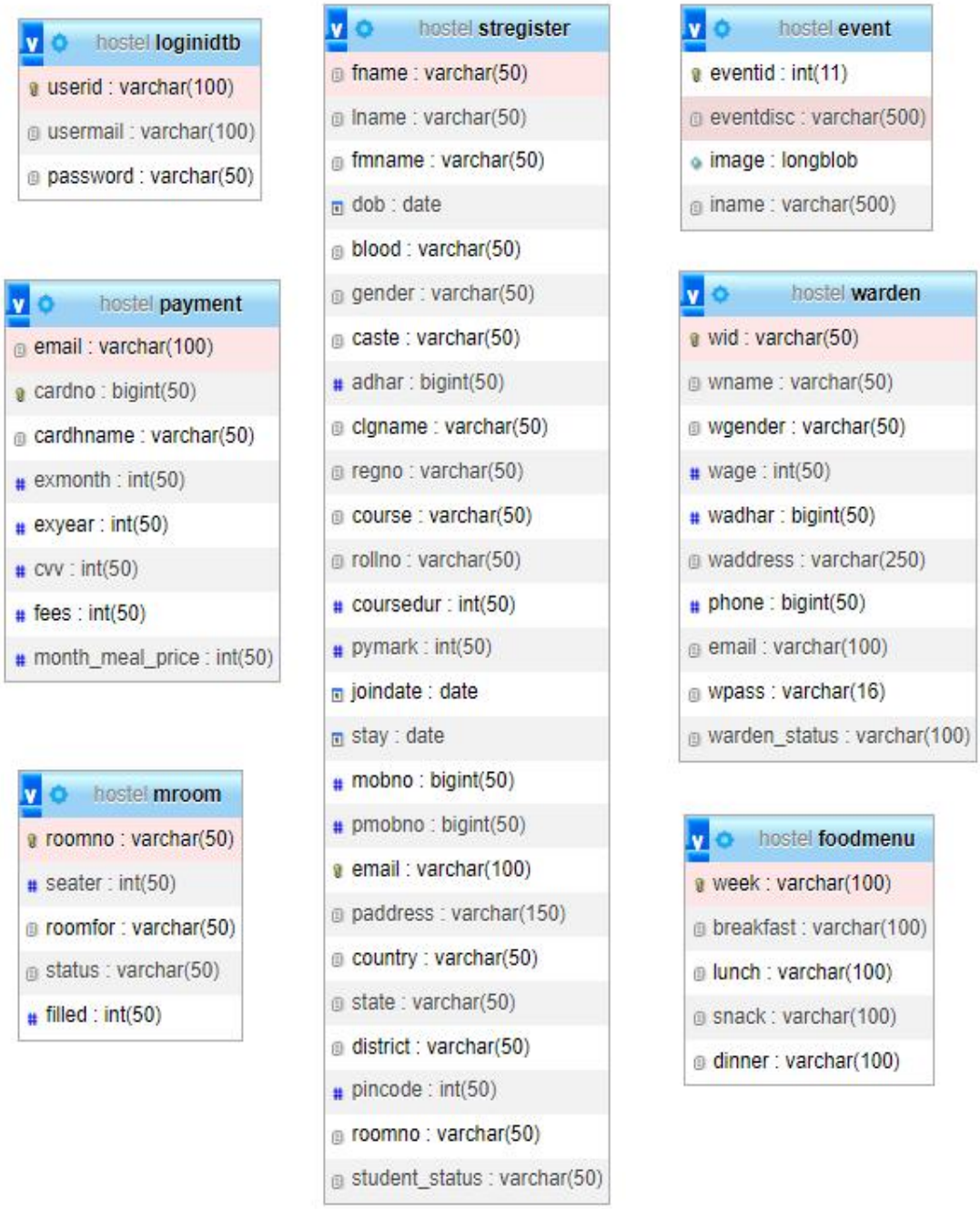


Figure 4.1 Schema

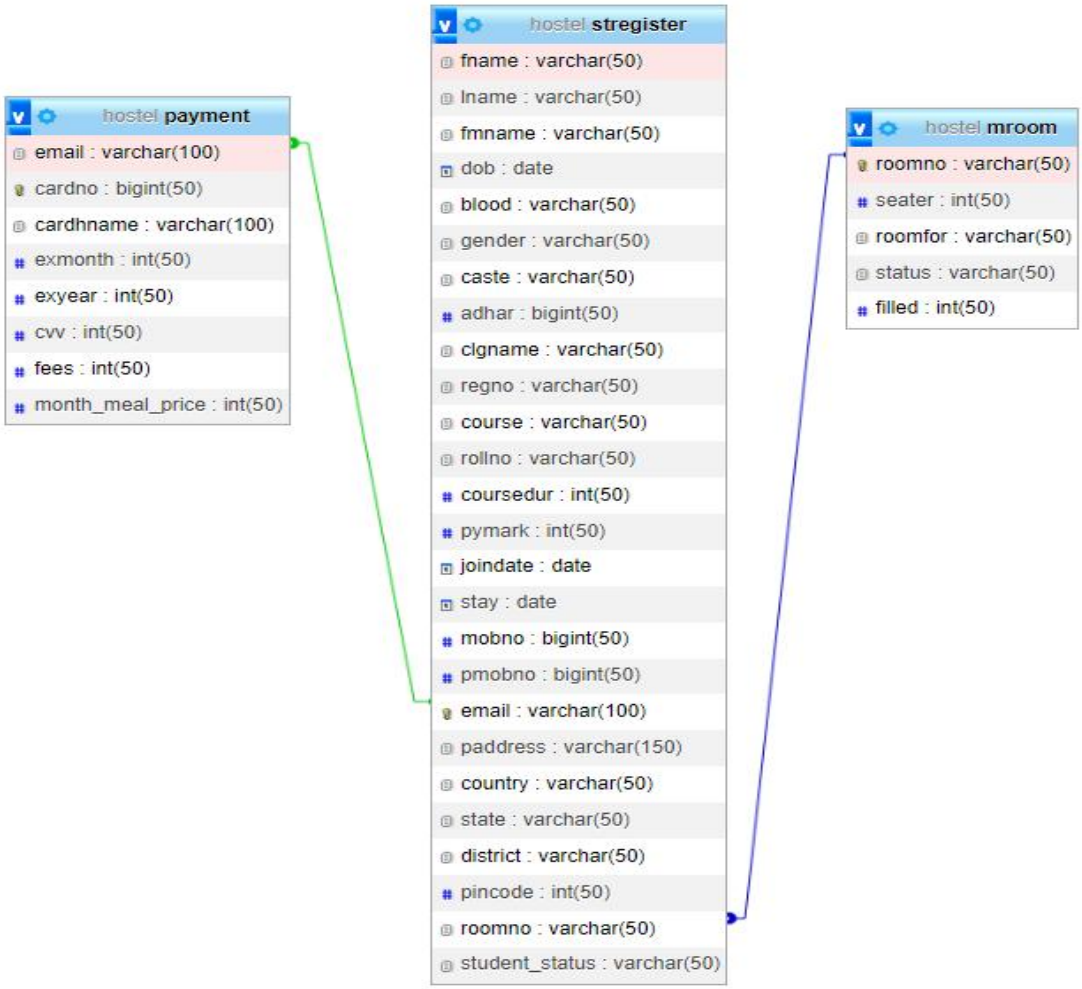


Figure 4.2 Schema Relationship

4.5 Table definition

Column name	Data type	Size	Constraint	Description
Userid	Varchar	100	Not null, primary key	ID of the user
Usermail	Varchar	100	Not null	Email of the student
password	Varchar	50	Not null	Password of the student

Table 4.1 Loginidtb

Column name	Data type	Size	Constraint	Description
Fname	Varchar	50	Not null	Students first name
Lname	Varchar	50	Not null	Students last name
fmname	Varchar	50	Not null	Students Father / mother name
Dob	Date		Not null	Students Date of birth
blood	Varchar	50	Not null	Students Blood type
Gender	Varchar	50	Not null	Students gender
Caste	Varchar	50	Not null	Students catse
Adhar	Bignt	50	Not null	Students aadhar number
Clgname	Varchar	50	Not null	Students college
Regno	Varchar	50	Not null	Students register number

Course	Varchar	50	Not null	Students course name
Rollno	Varchar	50	Not null	Students roll number
Cousedur	Int	50	Not null	Students course duration
Pymark	Int	50	Not null	Students previous year mark
Joindate	date		Not null	Students registered date
Stay	date		Not null	Student checkout date
Mobno	Bignt	50	Not null	Students mobile number
Pmobno	Bigint	50	Not null	Students parent mobile number
Email	Varchar	100	Not null, Primary key	Students email id
Paddress	Varchar	150	Not null	Permanent address of the student

Country	Varchar	50	Not null	Students country
State	Varchar	50	Not null	Students state
District	Varchar	50	Not null	Students district
Pincode	Int	50	Not null	Students pincode
roomno	Varchar	50	Not Null, Foreign key	Students booked room
Student_status	Varchar	50	Not null	Old student or current student

Table 4.2 Stregister

Column name	**Data type**	**Size**	**Contraints**	**Description**
Email	Varchar	100	Foreign key, not null	Students email
Cardno	Bigint	50	Primary key, not null	Card number
Cardhname	Varchar	100	Not null	Card holder number
Exmonth	Int	50	Not null	Card expiry month

Exyear	Int	50	Notnull	Card expiry year
Cvv	Int	50	Not null	Card cvv
Fees	Int	50	Not null	Students fees for hostel
Month_meal_price	Int	50		Students monthly meal price

Table 4.3 Payment

Column name	**Data type**	**Size**	**Contraints**	**Description**
Week	Varchar	100	Primary key, Not null	week
breakfast	Varchar	100	Not null	Food for breakfast
Lunch	Varchar	100	Not null	Food for lunch
Snack	Varchar	100	Not null	Food for snack
Dinner	Varchar	100	Not null	Food for dinner

Table 4.4 Foodmenu

Column name	Data type	Size	Contraints	Description
Eventid	Int	11	Primary key, not null, auto increment	Event id
eventdisc	Varchar	500	Not null	Description about the Event
image	longblob	100	Not null	Image of event
iname	Varchar	500	Not null	Image name

Table 4.5 Event

Column name	Data type	Size	Contraints	Description
roomno	varchar	50	Primary key, not null	Room number
seater	int	50	Not null	Number of seats in the room
roomfor	Varchar	50	Not null	Room for boys or girls
status	Varchar	50	Not null	Status of room whether it is filled or available

filled	Int	50	Not null	Seat filled in a room

Table 4.6 Mroom

Column name	Data type	Size	Contraints	Description
wid	varchar	50	Primary key, Not null	Warden id
wname	Varchar	50	Not null	Warden name
wgender	Varchar	50	Not null	Warden gender
wage	Int	50	Not null	Age of warden
wadhar	Bignt	50	Not null	Warden aadhar number
waddress	Varchar	250	Not null	Address of warden
Phone	Bigint	50	Not null	Phone number of warden
Email	Varchar	100	Not null	Wardens email
Wpass	Varchar	16	Not null	Wardens password

Warden_status	Varchar	100	Not null	Old or current warden

Table 4.7 warden

4.6 Physical design

The physical design is where you translate schemas into actual database structures.

- Entity to Table
- Tuples to rows
- Attribute to Column
- Primary Key and Alternate Key to Unique Index
- Domain into constraints

4.7. Data Dictionary

A data dictionary is a file or a set of files that includes a database's metadata. The data dictionary hold records about other objects in the database, such as data ownership, data relationships to other objects, and other data.

The data dictionary, in general, includes information about the following:

- Name of the data item
- Aliases
- Description/purpose
- Related data items
- Range of values
- Data structure definition

4.8 ER diagram

ER-modeling is a data modeling method used in software engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling method are called Entity-Relationship Diagrams or ER diagrams or ERDs.

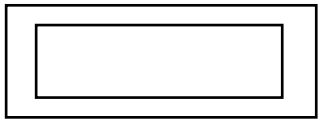
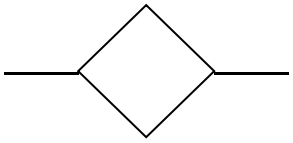
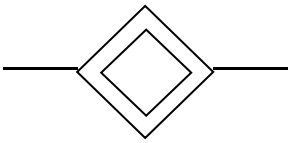
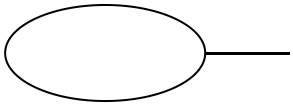
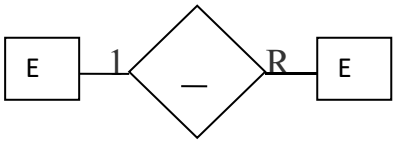
Symbols	Conversion
	Entity
	Weak entity
	Relation
	Identity Reltion
	Attribute
	Derived Attribute
	Cardinality ratio 1:N from E1:E2 in R.

Table 4.8 ER-Diagram

Components of an ER Diagrams

1. Entity

An entity can be a real-world object, either animate or inanimate, that can be merely identifiable.

An entity is denoted as a rectangle in an ER diagram. For example, in a school database, students, teachers, classes, and courses offered can be treated as entities. All these entities have some attributes or properties that give them their identity.

Entity Set

An entity set is a collection of related types of entities.

Strong Entity

An entity with uniquely identified by its attribute.

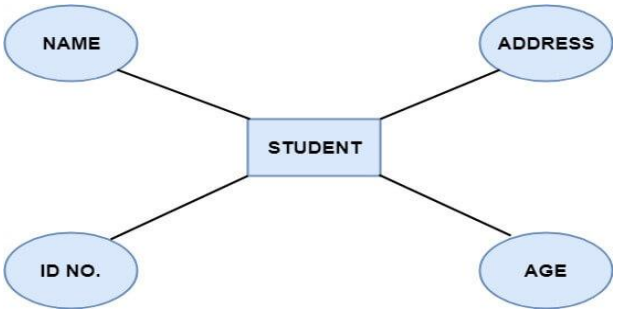
Weak Entity

In a relational database, a week entity is an entity that cannot be uniquely identified by its attributes alone.

2. Attributes

Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

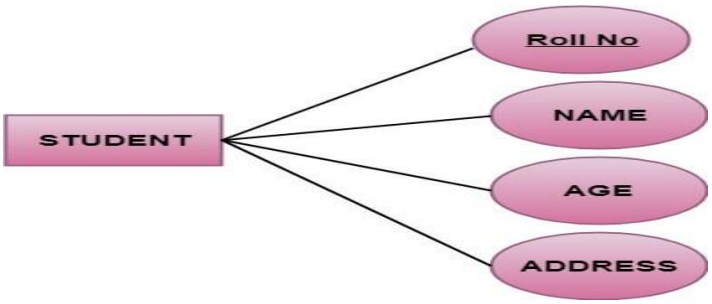
There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.



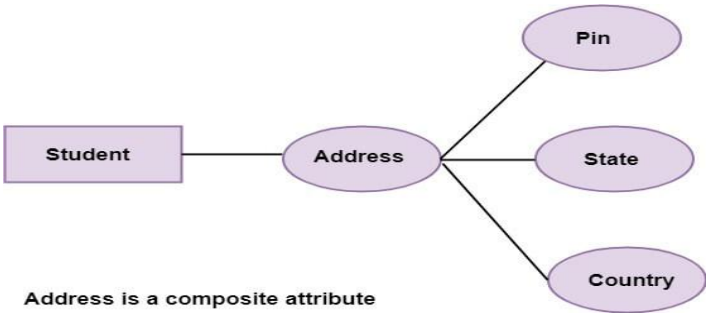
There are four types of Attributes:

1. Key attribute
2. Composite attribute
3. Single-valued attribute
4. Multi-valued attribute
5. Derived attribute

1. Key attribute: Key is an attribute or collection of attributes that uniquely identifies an entity among the entity set. For example, the roll_number of a student makes him identifiable among students.

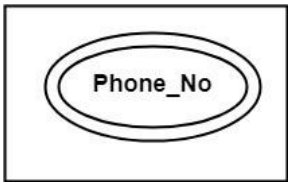


2. Composite attribute: An attribute that is a combination of other attributes is called a composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other characteristics such as pin code, state, country.

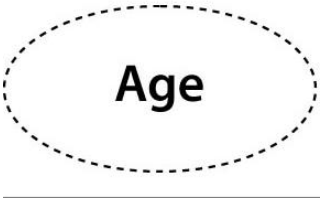


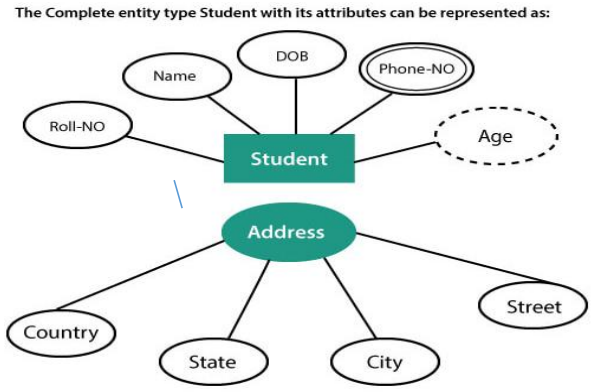
3. Single-valued attribute: Single-valued attribute contain a single value. For example, Social_Security_Number.

4. Multi-valued Attribute: If an attribute can have more than one value, it is known as a multi-valued attribute. Multi-valued attributes are depicted by the double ellipse. For example, a person can have more than one phone number, email-address, etc.



5. Derived attribute: Derived attributes are the attribute that does not exist in the physical database, but their values are derived from other attributes present in the database. For example, age can be derived from date_of_birth. In the ER diagram, Derived attributes are depicted by the dashed ellipse.





3. Relationships

The association among entities is known as relationship. Relationships are represented by the diamond-shaped box. For example, an employee works_at a department, a student enrolls in a course. Here, Works_at and Enrolls are called relationships.

Degree of a relationship set

The number of participating entities in a relationship describes the degree of the relationship. The three most common relationships in E-R models are:

1. Unary (degree1)
2. Binary (degree2)
3. Ternary (degree3)

1. Unary relationship: This is also called recursive relationships. It is a relationship between the instances of one entity type. For example, one person is married to only one person.

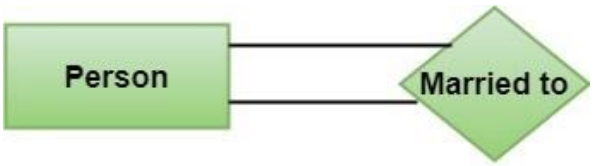


Fig: Unary Relationship

2. Binary relationship: It is a relationship between the instances of two entity types. For example, the Teacher teaches the subject.

Fig: Binary Relationship

3. Ternary relationship: It is a relationship amongst instances of three entity types. In fig, the relationships "**may have**" provide the association of three entities, i.e., TEACHER, STUDENT, and SUBJECT. All three entities are many-to-many participants. There may be one or many participants in a ternary relationship.

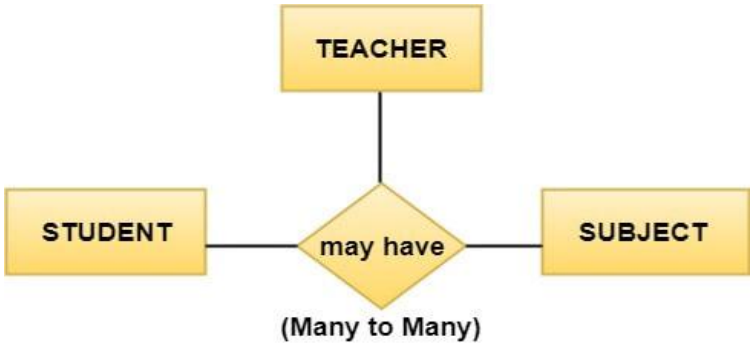


Fig: Ternary Relationship

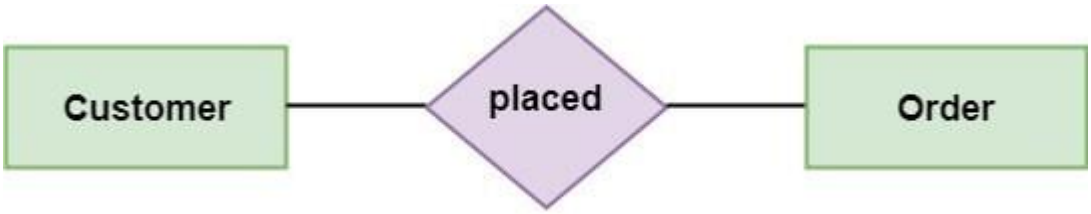
Cardinality Ratio

Cardinality describes the number of entities in one entity set, which can be associated with the number of entities of other sets via relationship set.

Types of Cardinalities

1. One to One: One entity from entity set A can be contained with at most one entity of entity set B and vice versa. Let us assume that each student has only one student ID, and each student ID is assigned to only one person. So, the relationship will be one to one.

2. One to many: When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationships. For example, a client can place many orders; a order cannot be placed by many customers.

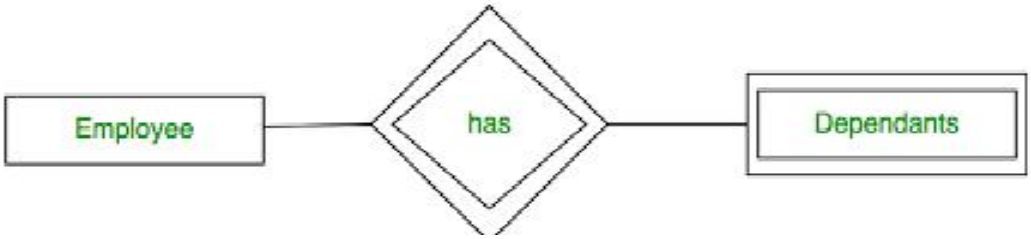


3. Many to One: More than one entity from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A. For example - many students can study in a single college, but a student cannot study in many colleges at the same time.

4. Many to Many: One entity from A can be associated with more than one entity from B and vice-versa. For example, the student can be assigned to many projects, and a project can be assigned to many students.

Identifying relationship

An identiffying relationship is a relationship between two entities in which an instance of a child entity is identified through its association with a parent entity, which means the child entity is dependent on the parent entity for its identity and cannot exits without it.



Participation Contraints

The participation constraint specifies the number of instance of an entity can participate in a relationship set.

Total participation – Each entity is involved in the relatioship. Total participation is represented by double lines.

Partial participation – Not all entities are involved in the relatioship. Partial participation is represented by single lines.

Example :

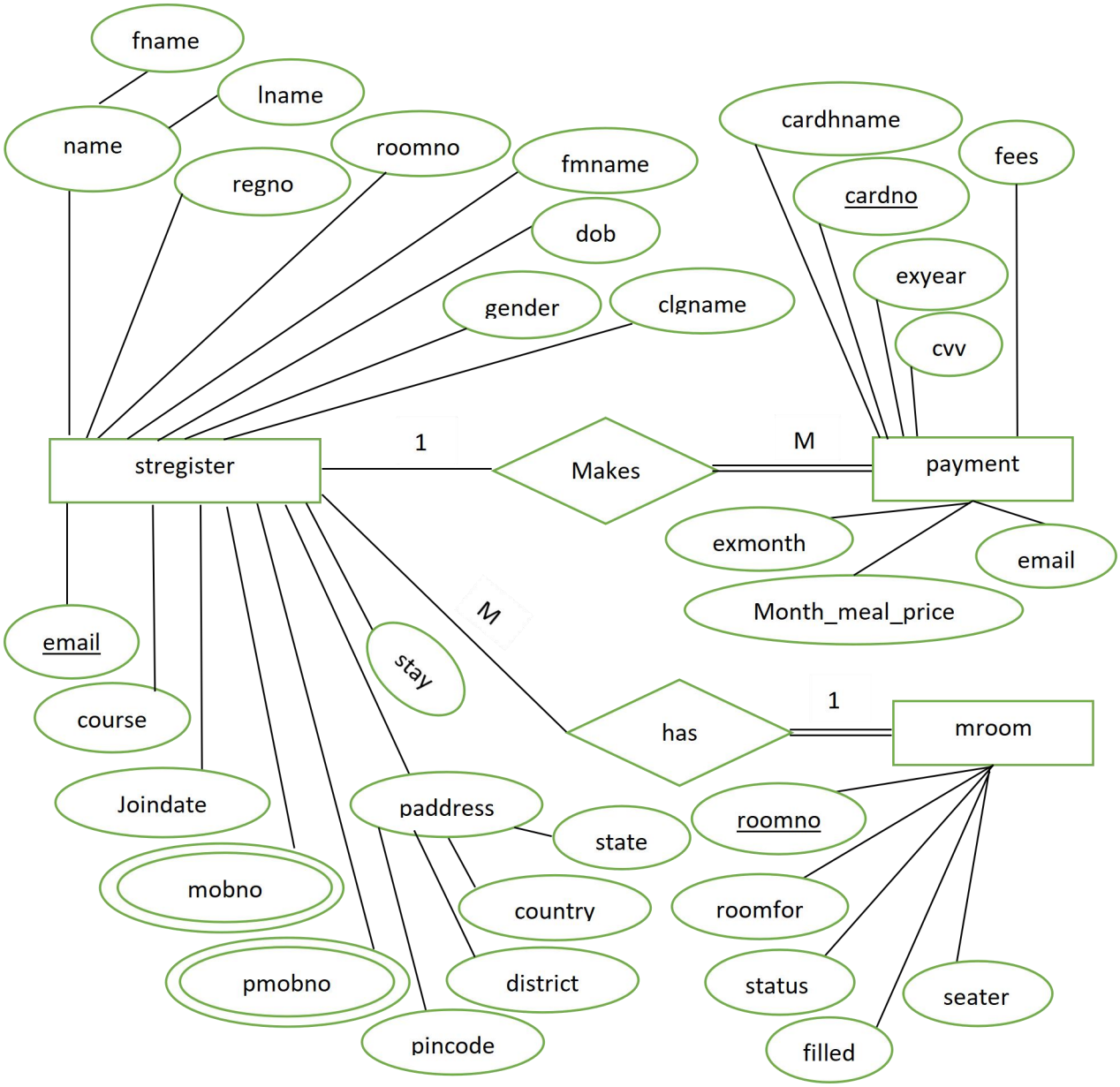


Figure 4.3 ER-Diagram

4. 9. Database Administration

4.9.1 DBMS System information

A database is an organized collection of structured information, or data typically stored electronically in a computer system. A data base is usually controlled by a database management system(DBMS).

4.9.2 DBMS configuration

(Steps for configuration of DBMS)

Steps for configure apache and mysql in XAMPP

1. In phpmyadmin, click the Users tab at the top.
2. Find the row that has User root and Host 127.0.0.1.
3. Click Edit Privileges.
4. Click Change password.
5. Enter the password twice (write it down somewhere if you're not sure you can remember it)
6. Click the Go button

4.9.3. Support software required

Mysql Required XAMPP

Software Requirements

The following operating systems are officially supported:

- Windows 7 (64-bit, Professional level or higher)
- Mac OS X 10.6.1+
- Ubuntu 9.10 (64bit)
- Ubuntu 8.04 (32bit/64bit)

4.9.4 Hardware (Storage) requirements

(Specify the Disk place Required)

The minimum hardware requirements are:

- Hard Disk:1 GB Required 500GB(Recommended)
- CPU: Intel Core or Xeon 3GHz (or Dual Core 2GHz) or equal AMD CPU
- Cores: Single (Dual/Quad Core is recommended)
- RAM: 4 GB (6 GB recommended)
- Graphic Accelerators: nVidia or ATI with support of OpenGL 1.5 or higher
- Display Resolution: 1280×1024 is recommended, 1024×768 is minimum.

4.9.5 Backup and recover

Recovery is **the process of restoring a database to the correct state in the event of a failure**

Database backup is **a way to protect and restore a database**. It is performed through database replication and can be done for a database or a database server.

Using phpMyAdmin to Back Up or Restore MySQL

If you're running phpMyAdmin backing up and **restoring your MySQL database** is simple.

The **export** function is used as a backup, and the **import** function is used to restore.

Step 1: Create a MySQL Database Backup

1. Open phpMyAdmin. On the directory tree on the left, click the database you want to back up.

This should open the directory structure in the right-hand window. You'll also notice that, in the directory tree on the left, all the assets under the main database are highlighted.

2. Click **Export** on the menu across the top of the display.

You'll see a section called "Export Method." Use **Quick** to save a copy of the whole database. Choose **Custom** to select individual tables or other special options.

Leave the **Format** field set to **SQL,** unless you have a good reason to change it.

3. Click **Go.** If you select **Quick,** your web browser will download a copy of the database into your specified downloads folder. You can copy that to a safe location.

Step 2: Clear the Old Database Information

It's important to clear out old data before restoring a backup. If there's any old data, it isn't overwritten when you restore. This can create duplicate tables, causing errors and conflicts.

1. Open phpMyAdmin, on the navigation pane on the left, choose the database you want to restore.

2. Click the **check all** box near the bottom. Then, use the drop-down menu labeled **With selected** to select **Drop.**

3. The tool should prompt you to confirm that you want to go forward. Click **yes.**

This will get rid of all the existing data, clearing the way for your restoration.

Step 3: Restore Your Backed up MySQL Database

In phpMyAdmin, the **Import** tool is used to restore a database.

1. On the menu across the top, click **Import**.

2. The first section is labeled **File to import.** A couple of lines down, there's a line that starts with "Browse your computer," with a button labeled **Choose File.** Click that button.

3. Use the dialog box to navigate to the location where you've saved the export file that you want to restore. Leave all the options set to default. (If you created your backup with different options, you can select those here.)

4. Click **Go.**

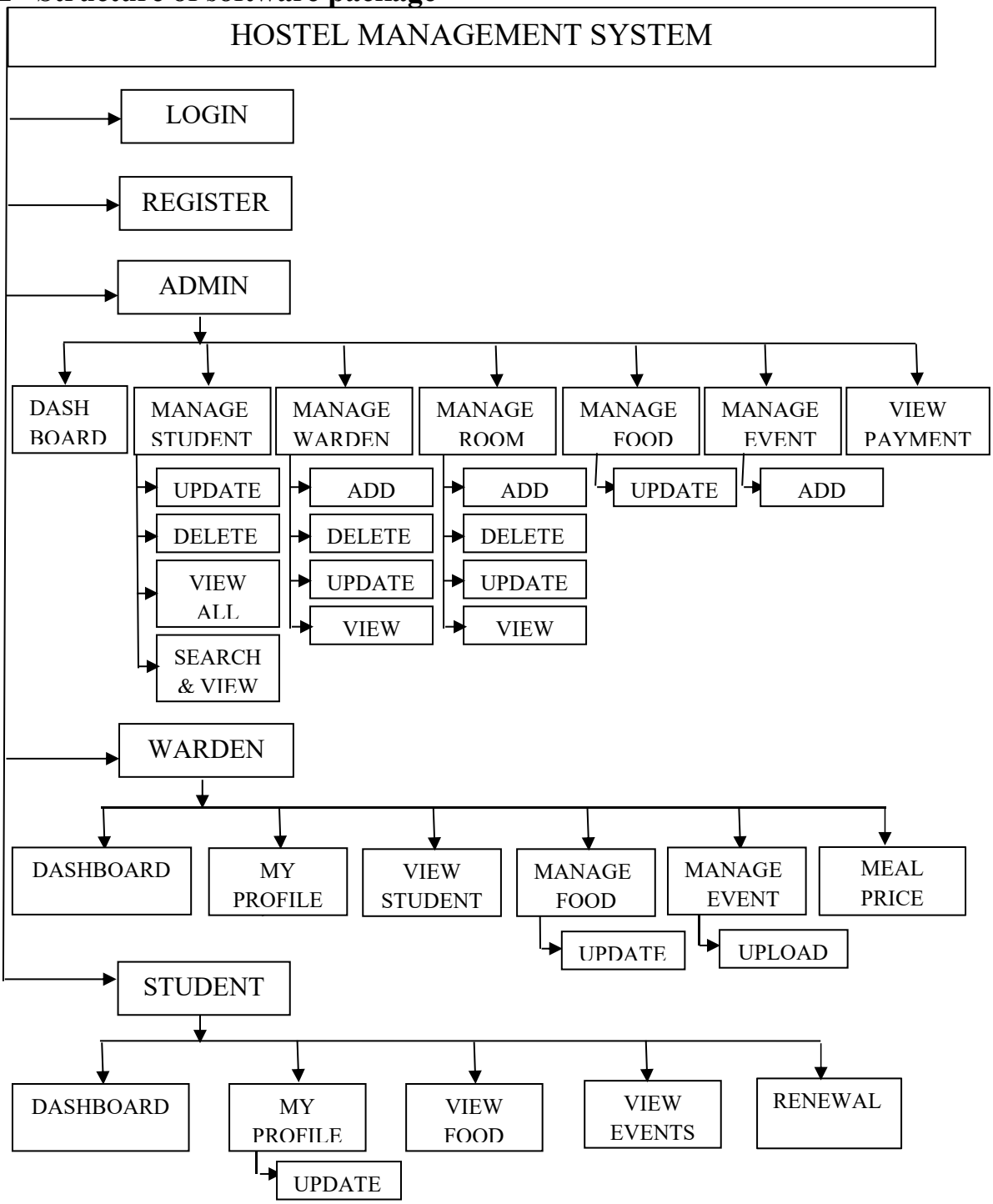
DETAILED DESIGN

5. DETAILED DESIGN

5.1 Introduction

During detailed design, the internal logic of each modules specified in system design is decided. During this phase further details of the modules are decided. Design of each of the modules usually specified in a high level description language which is independent of the language in which software eventually be implemented

5.2 Structure of software package



5.3 Module decomposition of software

Structure chart:

Structure chart is a top-down modular design, consist of squares representing different models in a system and lines .Structure chart shows how program has been partitioned into manageable modules hierarchy and organization of those modules and communicational interface.

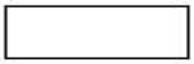
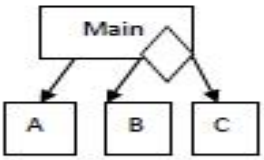
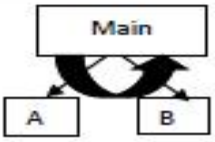
Symbol	Name	Process
	Data flow	Show the direction flow of data.
	Control flow	Shows the direction of flow control.
	Processing	Shows manipulation, calculation and processing.
	Module Invocation	It represent subordinate module being invoked by superior ordinate module.
	Condition invocation	It indicates that the invocation of subordinates Module depends on the evaluation of a condition.
	Iteration	It represent the iteration

Table 5.1 Structure chart

Flow chart:

Flow chart is a graphical representation of solution to the given problems. A Flowchart is pictorial representation of an algorithm, workflow or process. The diagrammatic representation illustrates a solution model to given problem. It uses the following symbol.

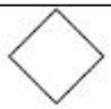
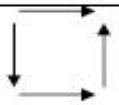
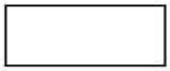
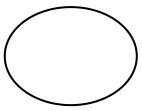
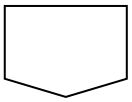
Symbol	Name	Purpose
	Terminator	It indicates the start and end process.
	Input/output	Input / output data
	Decision	It represents a comparison or question that determines an alternate path to be followed.
	Flow direction	Shows the direction of data flow.
	Processing	It represents manipulation, calculation or information processing.
	Direction access storage	File storage
	Preparation(Looping)	An instruction or Group of instruction
	In-page	
	Off- page	
	Delay	

Table 5.2 Flowchart

5.3.1 Login

5.3.1.1 Input: Email id and password

5.3.1.2 Procedural details

Algorithm:

Step 1: start
Step 2: Input username and password
Step 3: IF input is valid THEN
IF email id exist then
Load to dashboard
ELSE
Display" invalid username or password"
GOTO Step 2
END IF
ELSE
Display "invalid input message"
GOTO Step 2
END IF
Step 5: End

5.3.1.3 File I/O interfaces: Loginidtb,warden .

5.3.1.4 Outputs: after successful authentication dashboard will be loaded

5.3.1.5 Implementation aspects: textbox, label, button.

5.3.2 Register

5.3.2.1 Input: Name, father name, gender, register number, email, checkout date duration, mobile number,blood group, caste, college name,course duration,address, aadhar number,cardno, cardholder name, expiry month and year, cvv, roomno .

5.3.2.2 Procedural details

Algorithm:

Step 1: start

Step 2: input Name, father name, gender, register number, email, checkout date duration, mobile number,blood group, caste, college name,course duration,address aadhar number, cardno, cardholder name, expiry month and year, cvv, roomno.

Step 3: Reads and validates the input

Step 4: IF input is valid THEN

IF student exists THEN

Displays"already registered with this email/register number message"

GOTO Step 2

ELSE

GOTO Step 5

END IF

ELSE

Displays "invalid input message"

GOTO Step 2

END IF

Step 5: Inserts the record to the database.

Step 6: Displays "Payment Done,Registration successful ".

Step 7: End

5.3.2.3 File I/O interfaces: stregister, payment,loginidtb.

5.3.2.4 Outputs: Store the data to the database and displays "Payment done,Registration successful".

5.3.2.5 Implementation aspects: Button, Textbox and labels.

5.3.3 Admin

5.3.3.1 Dashboard

5.3.3.1.1 Input: Button click

5.3.3.1.2 Procedural details

Flow chart:

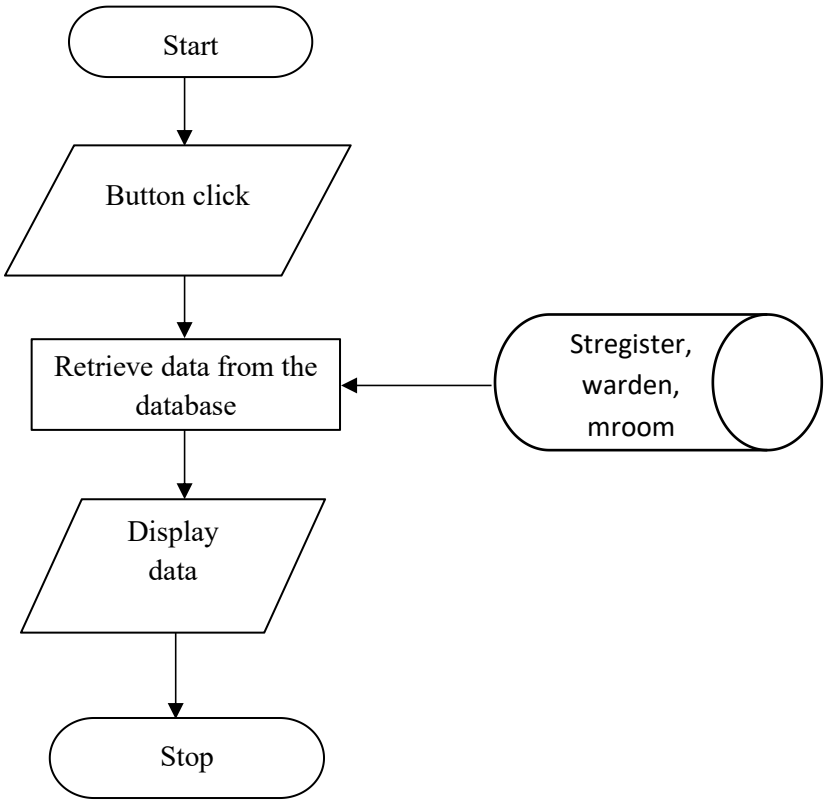


Figure 5.1 Dashboard Flowchart

5.3.3.1.3 File I/O interfaces: stregister, warden, mroom.

5.3.3.1.4 Outputs: Displays the total number of students, wardens, rooms.

5.3.3.1.5 Implementation aspects: Button, Textbox and labels.

5.3.3.2 Manage Student

5.3.3.2.1 Update Student

5.3.3.2.1.1 Input: Name, father name, gender, register number, mobile number,blood group, caste, college name,course duration,address, aadhar number, room no.

5.3.3.2.1.2 Procedural details

Algorithm:

Step 1: start

Step 2: Input student details.

Step 3: Reads and validates the input

Step 4: IF input is valid THEN

IF student exists THEN

GOTO Step 5

ELSE

Displays “student does not exist”

GOTO Step 2

END IF

ELSE

Displays “invalid input message”

GOTO Step 2

END IF

Step 5: Updates the record to the database.

Step 6: Displays “Record updated successfully ”.

Step 7: End

5.3.3.2.1.3 File I/O interfaces: stregister, mroom.

5.3.3.2.1.4 Outputs: Updates record to the database and displays Record updated successfully.

5.3.3.2.1.5 Implementation aspects: Button, Textbox and labels

5.3.3.2.2 Delete Student

5.3.3.2.2.1 **Input:** Register Number

5.3.3.2.2.2 **Procedural details**

Flowchart:

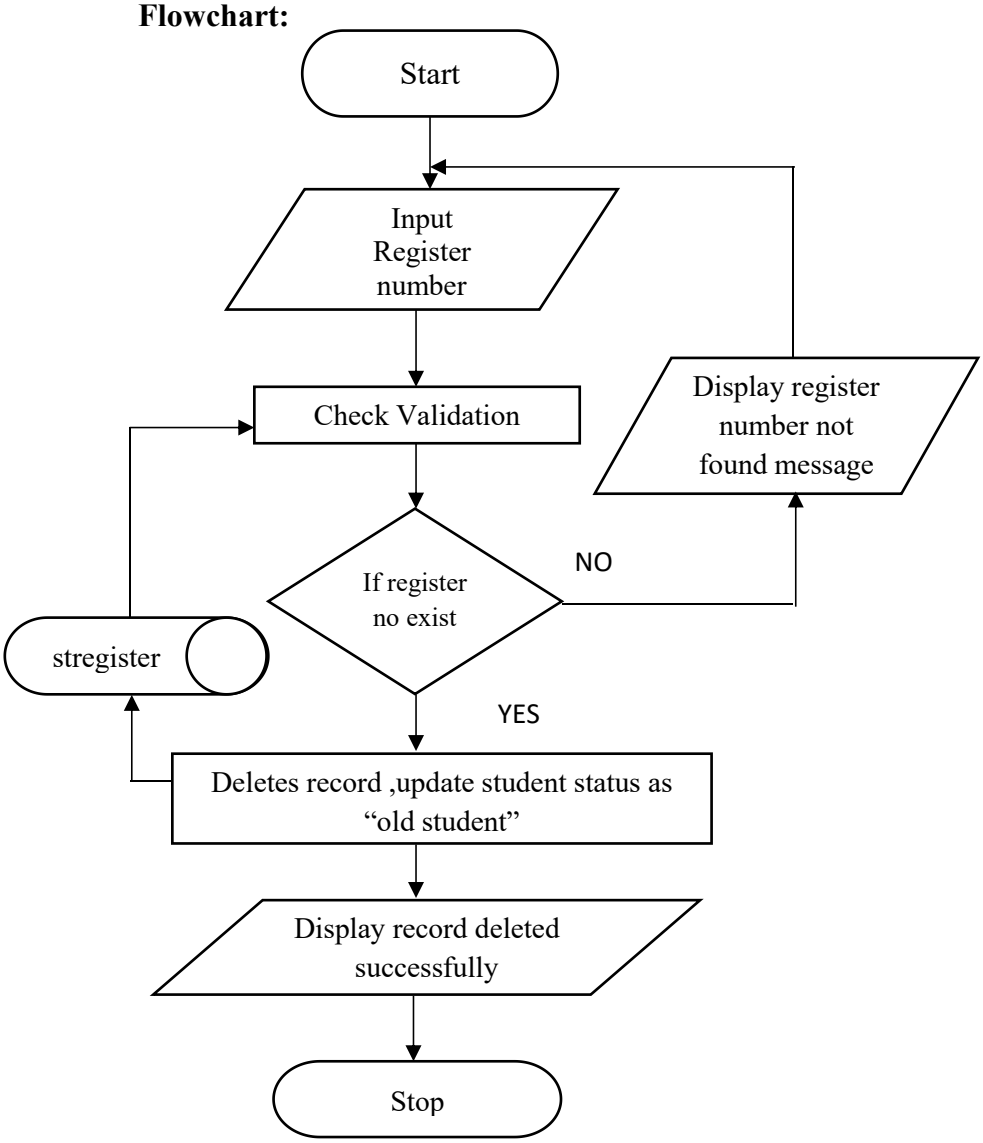


Figure 5.2 Delete student Flowchart

5.3.3.2.2.3 File I/O interfaces: stregister.

5.3.3.2.2.4 Outputs: Displays deleted successfully message.

5.3.3.2.2.5 Implementation aspects: Button, Textbox and labels

5.3.3.2.3 View all student

5.3.3.2.3.1 Input: Button click

5.3.3.2.3.2 Procedural details

Structured chart:

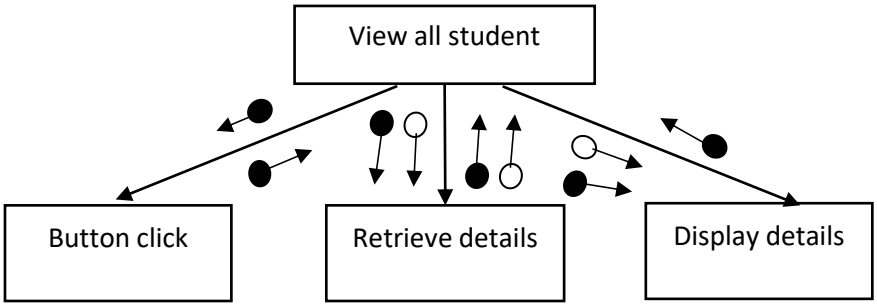


Figure 5.3 View all student Structure chart

5.3.3.2.3.3 File I/O interfaces: stregister.

5.3.3.2.3.4 Outputs: Displays all the students record from database.

5.3.3.2.3.5 Implementation aspects: Button, Textbox and labels

5.3.3.2.4 Search and view student

5.3.3.2.4.1 Input: Register Number

5.3.3.2.4.2 Procedural details

Algorithm:

Step 1: Start

Step 2: Input Register number

Step 3: Reads and validates the input

Step 4: IF input is valid THEN

IF register number exists THEN

GOTO Step 5

ELSE

Display “register number not found”

GOTO Step 2

END IF

ELSE

Displays “invalid input message”

GOTO Step 2

END IF

Step 5: retrieves the record from the database and displays the record.

Step 6: End

5.3.3.2.4.3 File I/O interfaces: stregister,payment.

5.3.3.2.4.4 Outputs: Displays details of the student

5.3.3.2.4.5 Implementation aspects: Button, Textbox and labels

5.3.3.3 Manage Warden

5.3.3.3.1 Add warden

5.3.3.3.1.1 Input: Name, warden id, warden email, gender, aadhar number, address, phone number.

5.3.3.3.1.2 Procedural details

Algorithm:

Step 1: start

Step 2: Input Warden details.

Step 3: Reads and validates the input

Step 4: IF input is valid THEN

IF warden exists THEN

Displays"Warden with this id already exists"

GOTO Step 2

ELSE

GOTO Step 5

END IF

ELSE

Displays "invalid input message"

GOTO Step 2

END IF

Step 5: Inserts the record to the database.

Step 6: Displays "Warden added successfully ".

Step 7: End

5.3.3.3.1.3 File I/O interfaces: warden.

5.3.3.3.1.4 Outputs: inserts record to the database displays successfully inserted message.

5.3.3.3.1.5 Implementation aspects: Button, Textbox and labels

5.3.3.3.2 Update warden

5.3.3.3.2.1 Input: Name, warden email, gender, aadhar number, address, phone number.

5.3.3.3.2.2 Procedural details

Algorithm:

Step 1: start

Step 2: Input warden details.

Step 3: Reads and validates the input

Step 4: IF input is valid THEN

IF warden id exists THEN

GOTO Step 5

ELSE

Displays “warden does not exist”

GOTO Step 2

END IF

ELSE

Displays “invalid input message”

GOTO Step 2

END IF

Step 5: Updates the record to the database.

Step 6: Displays “Record updated successfully ”.

Step 7: End

5.3.3.3.2.3 File I/O interfaces: warden.

5.3.3.3.2.4 Outputs: Updates the record to the database and displays record updated successfully message.

5.3.3.3.2.5 Implementation aspects: Button, Textbox and labels

5.3.3.3.3 Delete warden

5.3.3.3.3.1 **Input:** warden id

5.3.3.3.3.2 Procedural details

Flowchart:

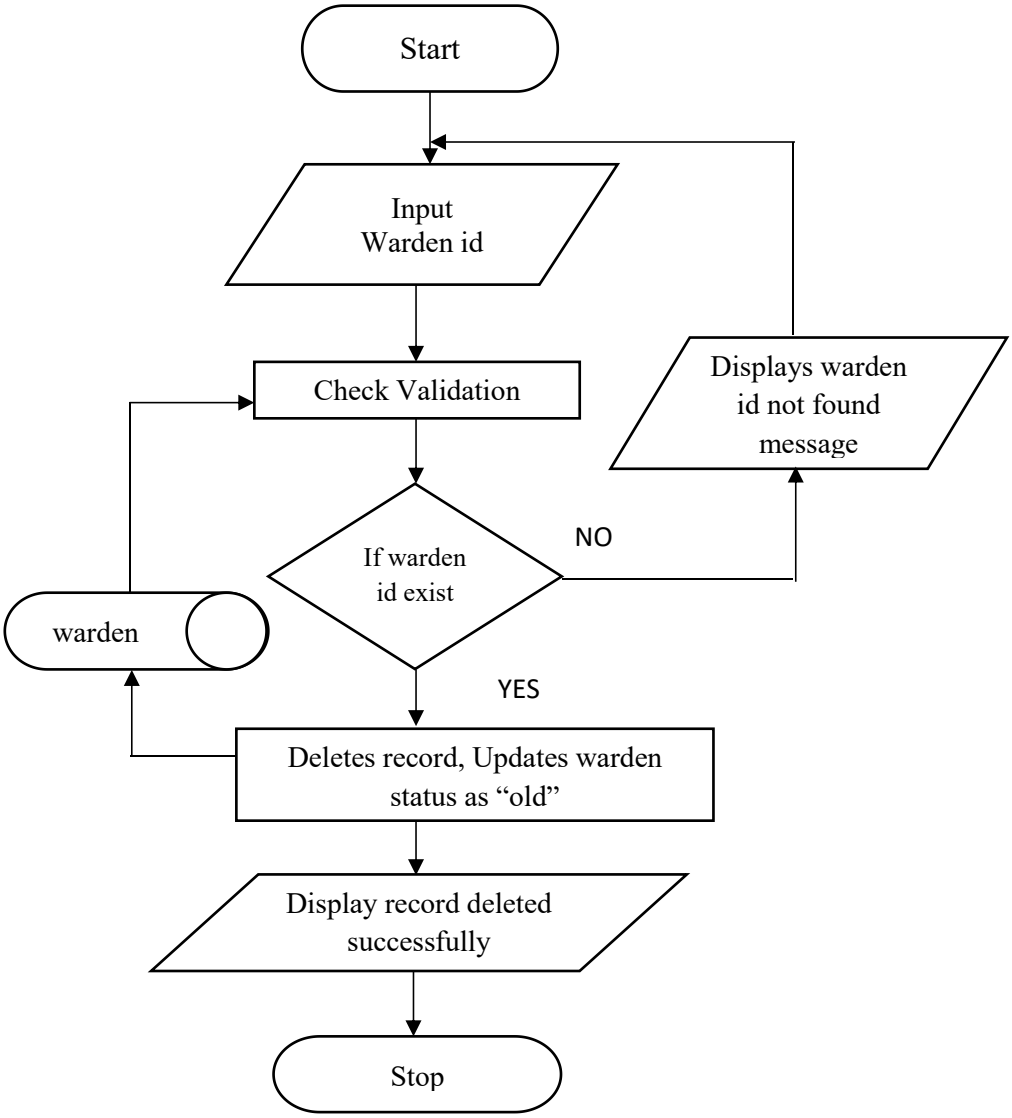


Figure 5.4 Delete warden Flowchart

5.3.3.3.3.3 **File I/O interfaces:** warden.

5.3.3.3.3.4 **Outputs:** Updates record and displays "record successfully deleted" message

5.3.3.3.3.5 **Implementation aspects:** Button, Textbox and labels

5.3.3.3.4 View all warden

5.3.3.3.4.1 Input: Button click

5.3.3.3.4.2 Procedural details

Structured chart:

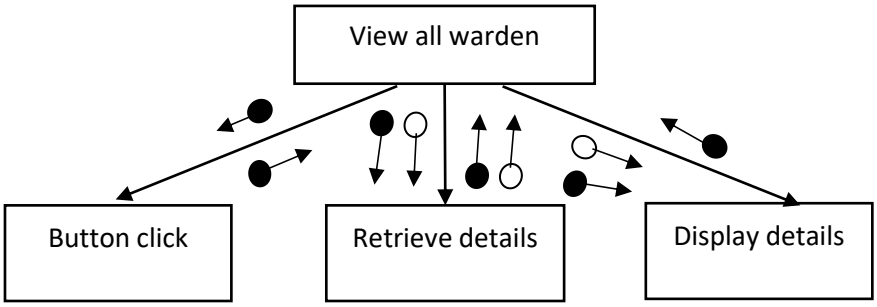


Figure 5.5 View all warden structure chart

5.3.3.3.4.3 File I/O interfaces: warden.

5.3.3.3.4.4 Outputs: Displays all the wardens record from database.

5.3.3.3.4.5 Implementation aspects: Button, Textbox and labels

5.3.3.4 Manage room

5.3.3.4.1 Add room

5.3.3.4.1.1 Input: Room Id, seater, room for, status.

5.3.3.4.1.2 Procedural details

Algorithm:

Step 1: start

Step 2: Input room details.

Step 3: Reads and validates the input

Step 4: IF input is valid THEN

IF room id exists THEN

Displays"room with this id already exists"

GOTO Step 2

ELSE

GOTO Step 5

END IF

ELSE

Displays "invalid input message"

GOTO Step 2

END IF

Step 5: Inserts the record to the database.

Step 6: Displays "Room added successfully ".

Step 7: End

5.3.3.4.1.3 File I/O interfaces: mroom.

5.3.3.4.1.4 Outputs: inserts record to the database and displays "room added successfully" message

5.3.3.4.1.5 Implementation aspects: Button, Textbox and labels

5.3.3.4.2 Update room

5.3.3.4.2.1 Input: seater, room for, status.

5.3.3.4.2.2 Procedural details

Flowchart:

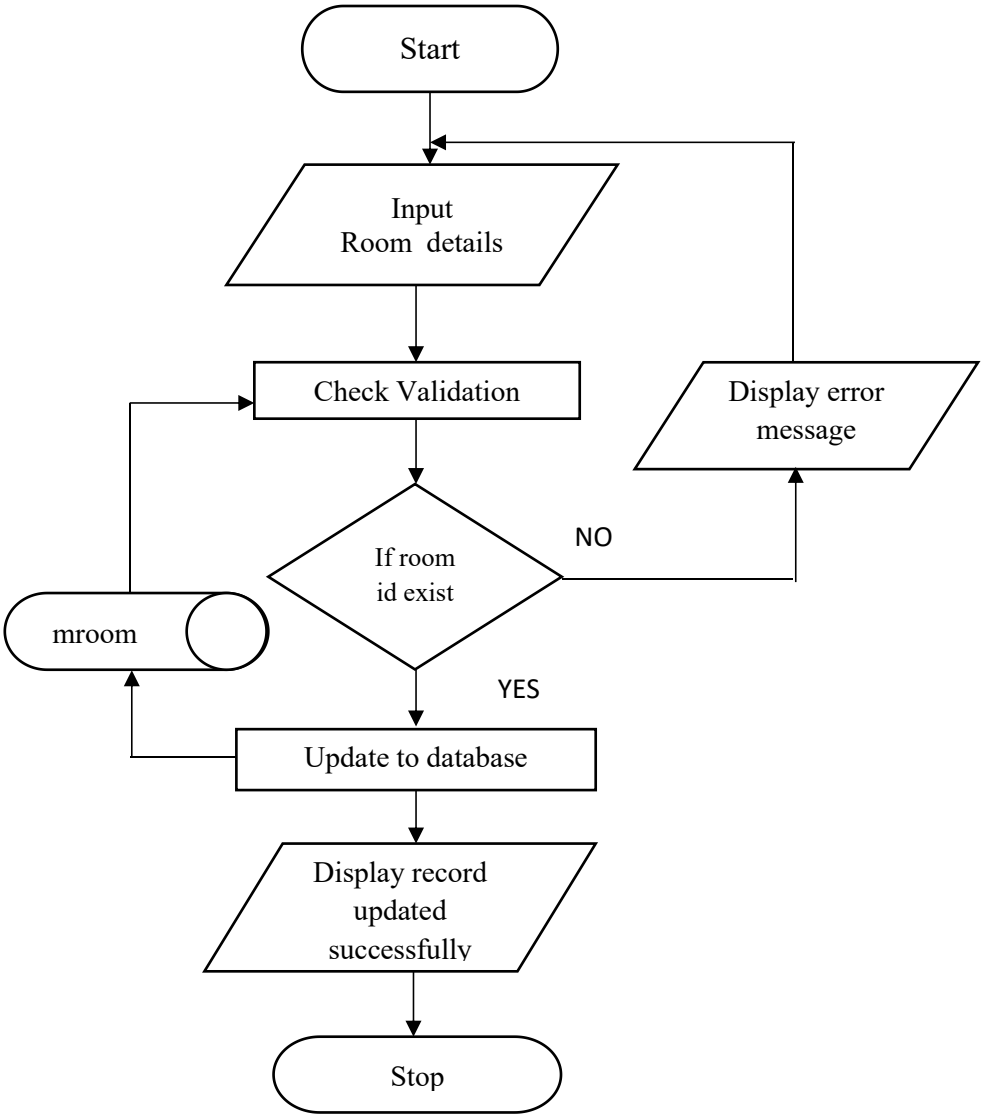


Figure 5.6 Update room Flowchart

5.3.3.4.2.3 File I/O interfaces: mroom.

5.3.3.4.2.4 Outputs: updates record to the database and displays ”record updated successfully” message.

5.3.3.4.2.5 Implementation aspects: Button, Textbox and label

5.3.3.4.3 Delete room

5.3.3.4.3.1 Input: Room id

5.3.3.4.3.2 Procedural details

Flowchart:

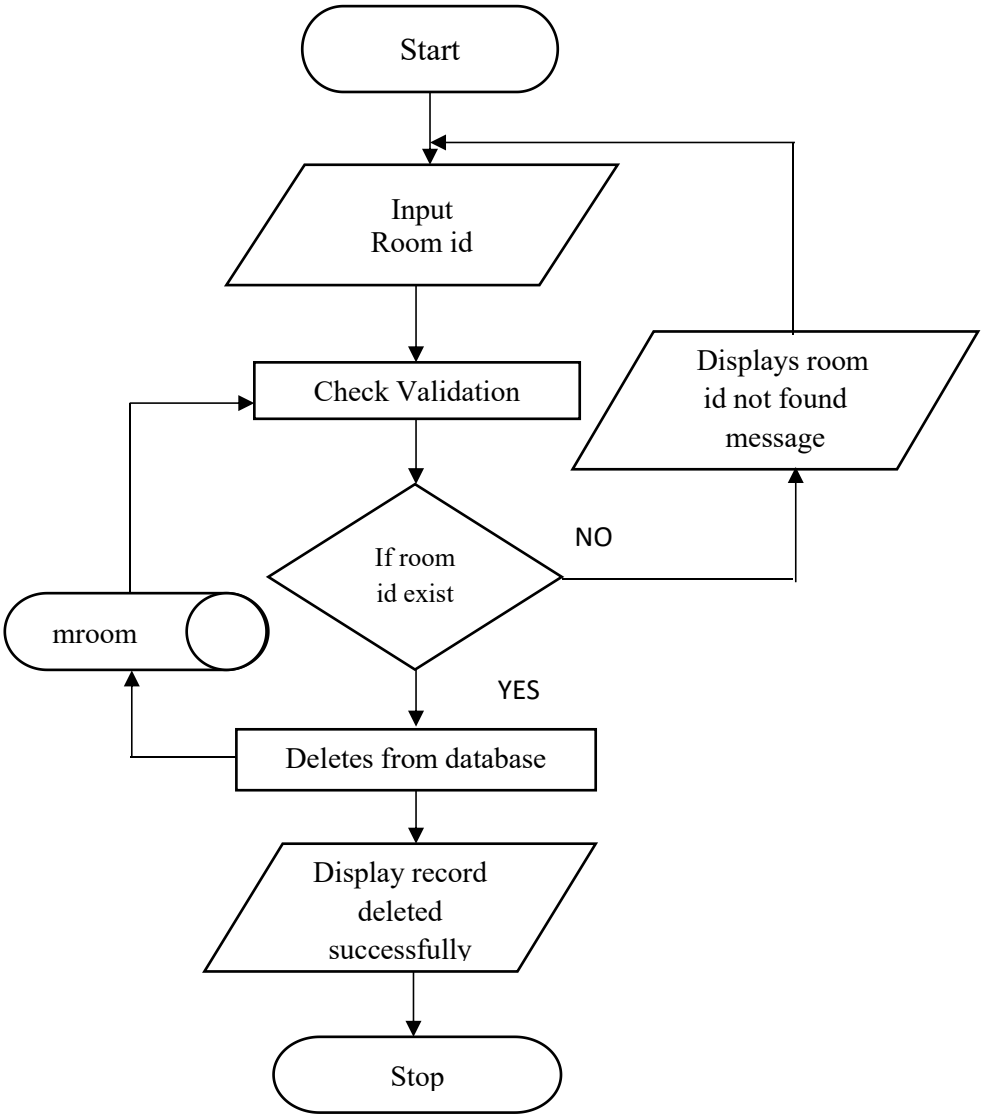


Figure 5.7 delete room Flowchart

5.3.3.4.3.3 File I/O interfaces: mroom.

5.3.3.4.3.4 Outputs: Deletes record from database and displays room deleted successfully message

5.3.3.4.3.5 Implementation aspects: Button, Textbox and labels

5.3.3.4.4 View all room

5.3.3.4.4.1 Input: Button click

5.3.3.4.4.2 Procedural details

Structured chart:

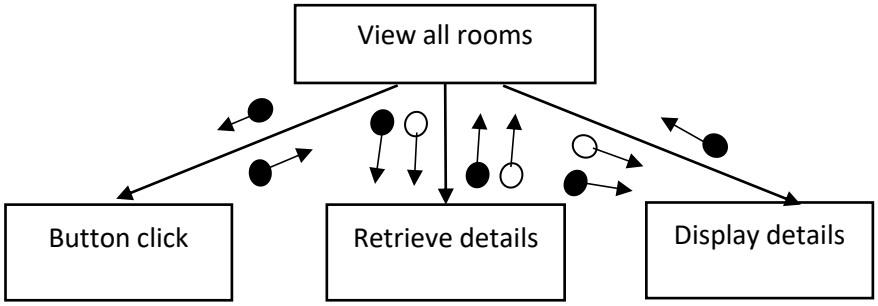


Figure 5.8 View all room structure chart

5.3.3.4.4.3 File I/O interfaces: mroom.

5.3.3.4.4.4 Outputs: Displays all the rooms record from database

5.3.3.4.4.5 Implementation aspects: Button, Textbox and labels

5.3.3.5 Manage food menu

5.3.3.5.1 Input: Button click

5.3.3.5.2 Procedural details

Algorithm:

Step 1: Start

Step 2: Input Button click

Step 3: Retrieves the record from database and displays the records

Step 4:Exit

5.3.3.5.3 File I/O interfaces: foodmenu.

5.3.3.5.4 Outputs: Displays all the record from database

5.3.3.5.5 Implementation aspects: Button, Textbox and labels

5.3.3.5.6 Update Food menu

5.3.3.5.6.1 Input: Food names

5.3.3.5.6.2 Procedural details

Flowchart:

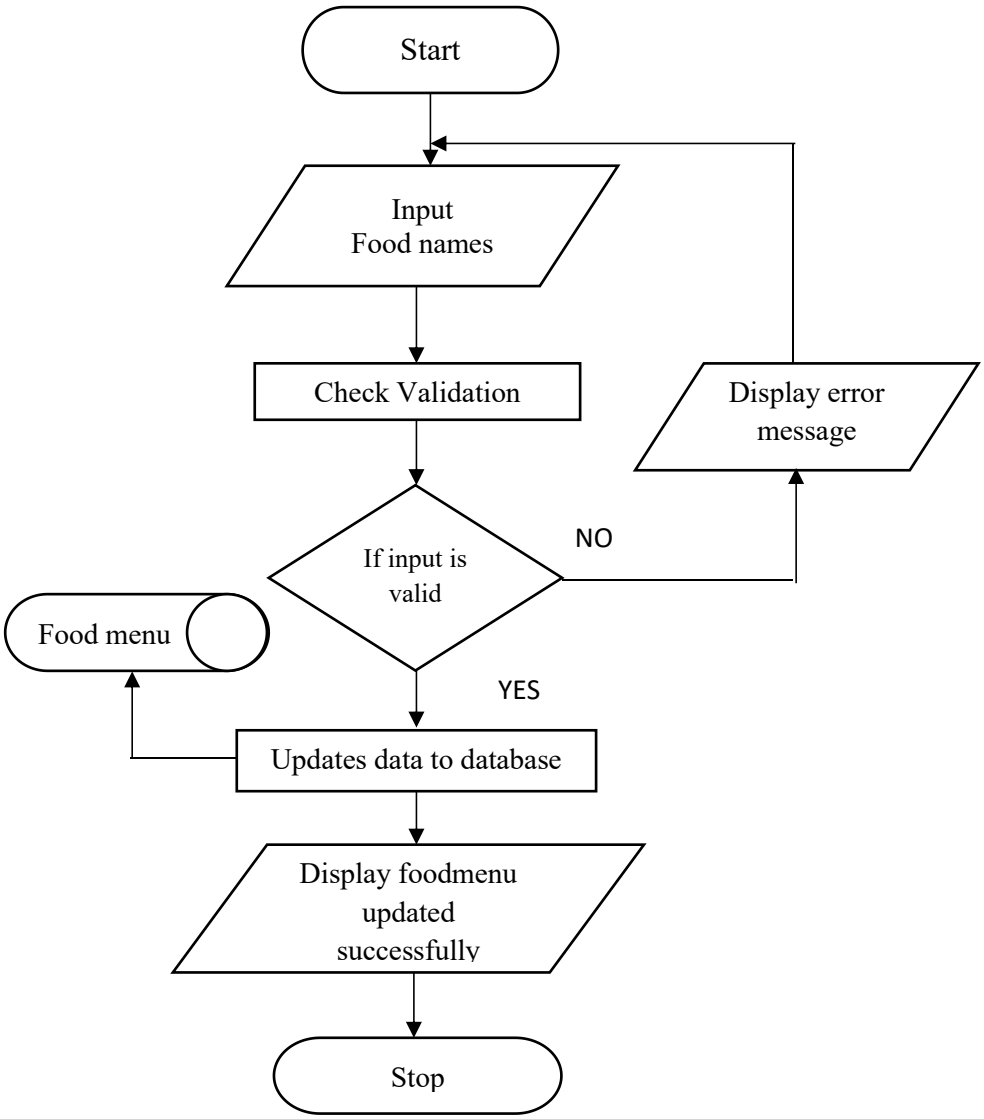


Figure 5.9 Update food menu Flowchart

5.3.3.5.6.3 File I/O interfaces: foodmenu.

5.3.3.5.6.4 Outputs: Updates data to the database and displays successfully updated message

5.3.3.5.6.5 Implementation aspects: Button, Textbox and labels

5.3.3.6 Manage event

5.3.3.6.1 Input: Button click

5.3.3.6.2 Procedural details

Structured chart:

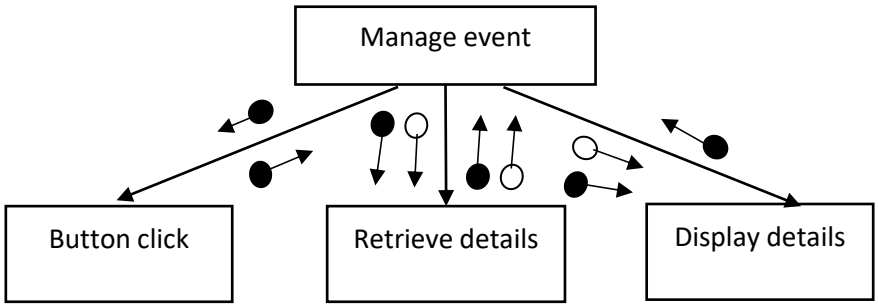


Figure 5.10 Manage event structure chart

5.3.3.6.3 File I/O interfaces: event.

5.3.3.6.4 Outputs: Displays all the record from database

5.3.3.6.5 Implementation aspects: Button, Textbox and labels

5.3.3.6.6 Upload event

5.3.3.6.6.1 Input: Event description and image.

5.3.3.6.6.2 Procedural details

Algorithm:

Step 1: Start

Step 2: Input Event description and image.

Step 3: Reads and validates the input

Step 4: IF input is valid THEN

IF image exists THEN

Displays"event with this image name already exists"

GOTO Step 2

ELSE

GOTO Step 5

END IF

ELSE

Displays "invalid input message"

GOTO Step 2

END IF

Step 5: Inserts the record to the database.

Step 6: Displays "event uploaded successfully ".

Step 7: End

5.3.3.6.6.3 File I/O interfaces: event.

5.3.3.6.6.4 Outputs: inserts the record to the database and displays event uploaded successfully message

5.3.3.6.6.5 Implementation aspects: Button, Textbox and label.

5.3.3.7 View student payments

5.3.3.6.1 Input: Button click

5.3.3.6.2 Procedural details

Structured chart:

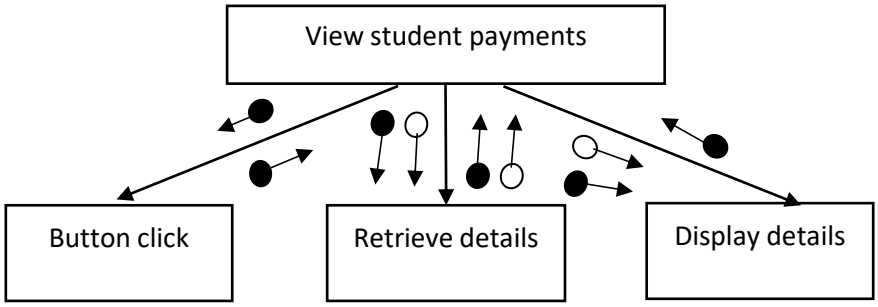


Figure 5.11 View student payments structure chart

5.3.3.6.3 File I/O interfaces: stregister, payment.

5.3.3.6.4 Outputs: Displays all the payment records of the student

5.3.3.6.5 Implementation aspects: Button, Textbox and labels

5.3.4 Warden

5.3.4.1 Dashboard

5.3.4.1.1 Input: Button click

5.3.4.1.2 Procedural details

Flowchart:

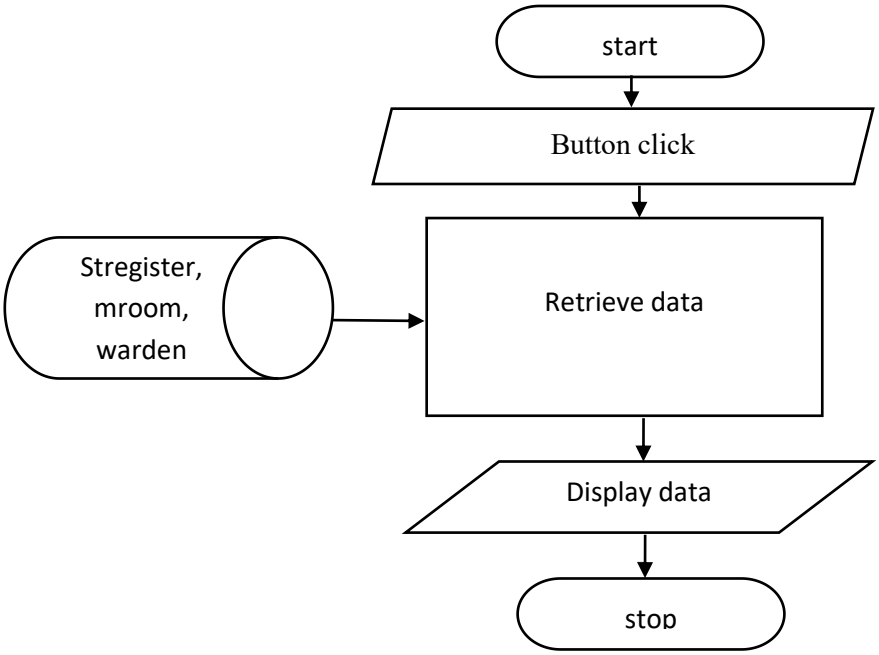


Figure 5.12 Dashboard Flowchart

5.3.4.1.3 File I/O interfaces: stregister, mroom, warden.

5.3.4.1.4 Outputs: Displays the total number of students, wardens, rooms etc.

5.3.4.1.5 Implementation aspects: Button, Textbox and labels.

5.3.4.2 My profile

5.3.4.2.1 Input: Button click

5.3.4.2.2 Procedural details

Structured chart:

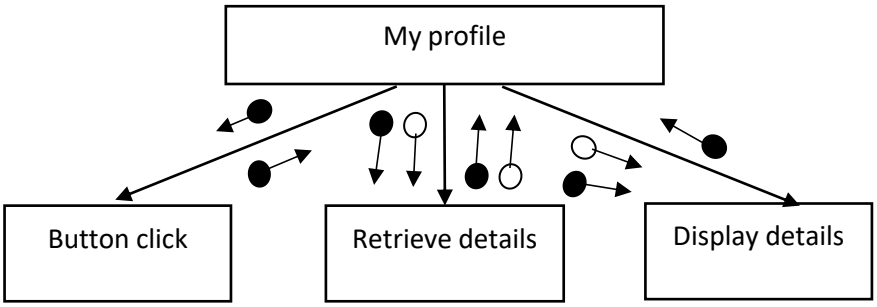


Figure 5.13 My profile Structure chart

5.3.4.2.3 File I/O interfaces: warden.

5.3.4.2.4 Outputs: Displays the warden's details from the database.

5.3.4.2.5 Implementation aspects: Button, Textbox and labels.

5.3.4.3 View student details

5.3.4.3.1 Input: Button click

5.3.4.3.2 Procedural details

Algorithm:

Step 1: Start

Step 2: Input Button click

Step 3: Retrieves the record from database and displays the records

Step 4:Exit

5.3.4.3.3 File I/O interfaces: stregister.

5.3.4.3.4 Outputs: Displays all the students record from database

5.3.4.3.5 Implementation aspects: Button, Textbox and labels

5.3.4.4 Manage food menu

5.3.4.4.1 Input: Button click

5.3.4.4.2 Procedural details

Flowchart:

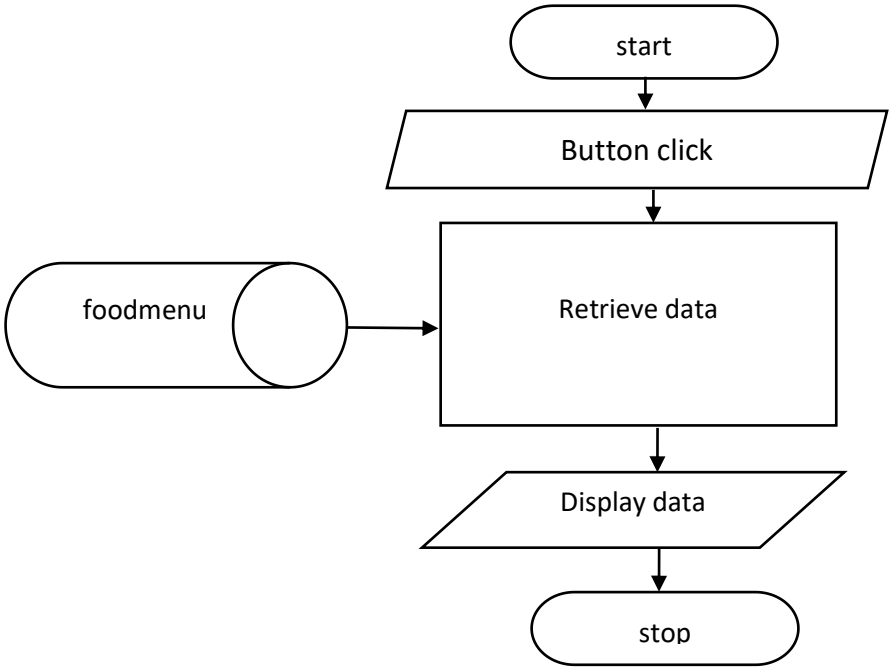


Figure 5.14 Manage food menu Flowchart

5.3.4.4.3 File I/O interfaces: foodmenu.

5.3.4.4.4 Outputs: Displays all the food detials from database

5.3.4.4.5 Implementation aspects: Button, Textbox and labels

5.3.4.4.6 Update Food menu

5.3.4.4.6.1 Input: Food names

5.3.4.4.6.2 Procedural details

Algorithm:

Step 1: Start

Step 2: Input food names.

Step 3: Reads and validates the input

Step 4: IF input is valid THEN

GOTO Step 5

ELSE

Displays “invalid input message”

GOTO Step 2

END IF

Step 5: Updates the record to the database.

Step 6: Displays “food menu updated successfully ”.

Step 7: End

5.3.4.4.6.3 File I/O interfaces: Foodmenu.

5.3.4.4.6.4 Outputs: Updates the record to the database and displays foodmenu updated successfully message.

5.3.4.4.6.5 Implementation aspects: Button, Textbox and labels

5.3.4.5 Manage event

5.3.4.5.1 Input: Button click

5.3.4.5.2 Procedural details

Structured chart:

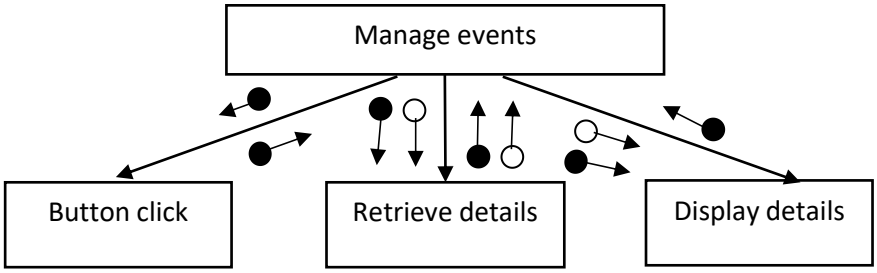


Figure 5.15 manage event structure chart

5.3.4.5.3 File I/O interfaces: event.

5.3.4.5.4 Outputs: Displays all the record from database

5.3.4.5.5 Implementation aspects: Button, Textbox and labels

5.3.4.5.6 Upload event

5.3.4.5.6.1 Input: Event description and image.

5.3.4.5.6.2 Procedural details

Algorithm:

Step 1: Start

Step 2: Input Event description and image.

Step 3: Reads and validates the input

Step 4: IF input is valid THEN

IF image exists THEN

Displays"event with this image name already exists"

GOTO Step 2

ELSE

GOTO Step 5

END IF

ELSE

Displays “invalid input message”

GOTO Step 2

END IF

Step 5: Inserts the record to the database.

Step 6: Displays “event uploaded successfully ”.

Step 7: End

5.3.4.5.6.3 File I/O interfaces: event.

5.3.4.5.6.4 Outputs: inserts the record to the database and displays event uploaded successfully message.

5.3.4.4.6.5 Implementation aspects: Button, Textbox and labels

5.3.4.6 Meal price

5.3.4.6.1 Input: Number of days

5.3.4.6.2 Procedural details

Algorithm:

Step 1: Start

Step 2: retreive details

Step 3: Input number of days.

Step 4: Reads and validates the input

Step 5: IF input is valid THEN

GOTO Step 6

ELSE

Displays “invalid input message”

GOTO Step 3

END IF

Step 6: Updates the record to the database.

Step 7: Displays “meal price updated successfully ”.

Step 8: End

5.3.4.6.3 File I/O interfaces: payment, stregister.

5.3.4.6.4 Outputs: Updates the record to the database and displays “meal price updated successfully” message.

5.3.4.6.5 Implementation aspects: Button, Textbox and labels

5.3.5 Student

5.3.5.1 Dashboard

5.3.5.1.1 Input: Button click

5.3.5.1.2 Procedural details

Flowchart:

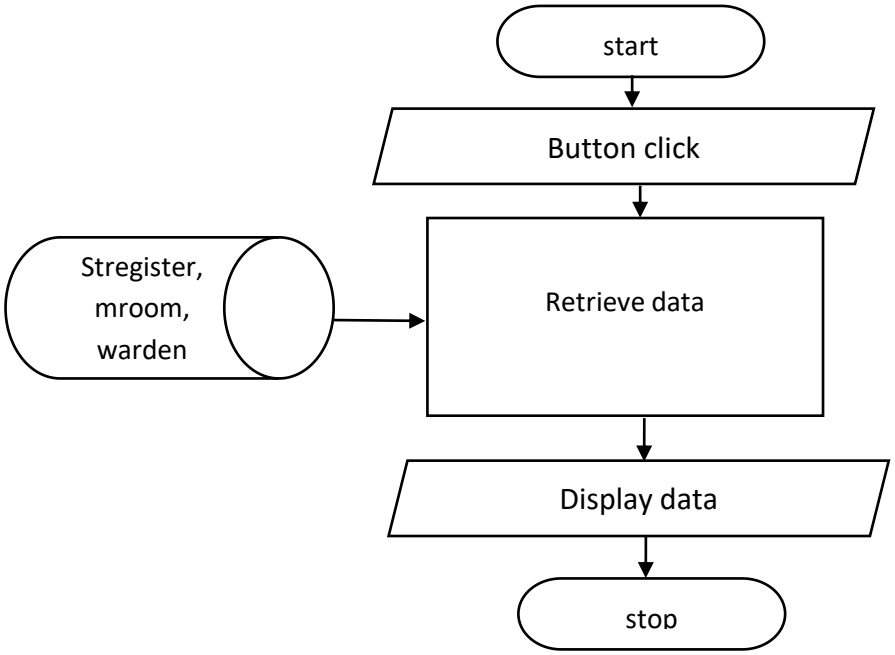


Figure 5.16 Dashboard Flowchart

5.3.5.1.3 File I/O interfaces: stregister, mroom, warden.

5.3.5.1.4 Outputs: Displays the total number of students, wardens, rooms etc.

5.3.4.1.5 Implementation aspects: Button, Textbox and labels.

5.3.5.2 My profile

5.3.5.2.1 Input: Button click

5.3.5.2.2 Procedural details

Algorithm:

Step 1: start

Step 2: input button click

Step 3: retrieves data from database and displays.

Step 4: End

5.3.5.2.3 File I/O interfaces: stregister, payment, loginidtb.

5.3.5.2.4 Outputs: Displays the student's details from the database.

5.3.5.2.5 Implementation aspects: Button, Textbox and labels

5.3.5.2.6 Update my profile

5.3.5.2.6.1 Input: College name, Roll no, Course, course duration, previous year mark, address, password.

5.3.5.2.6.2 Procedural details

Flowchart:

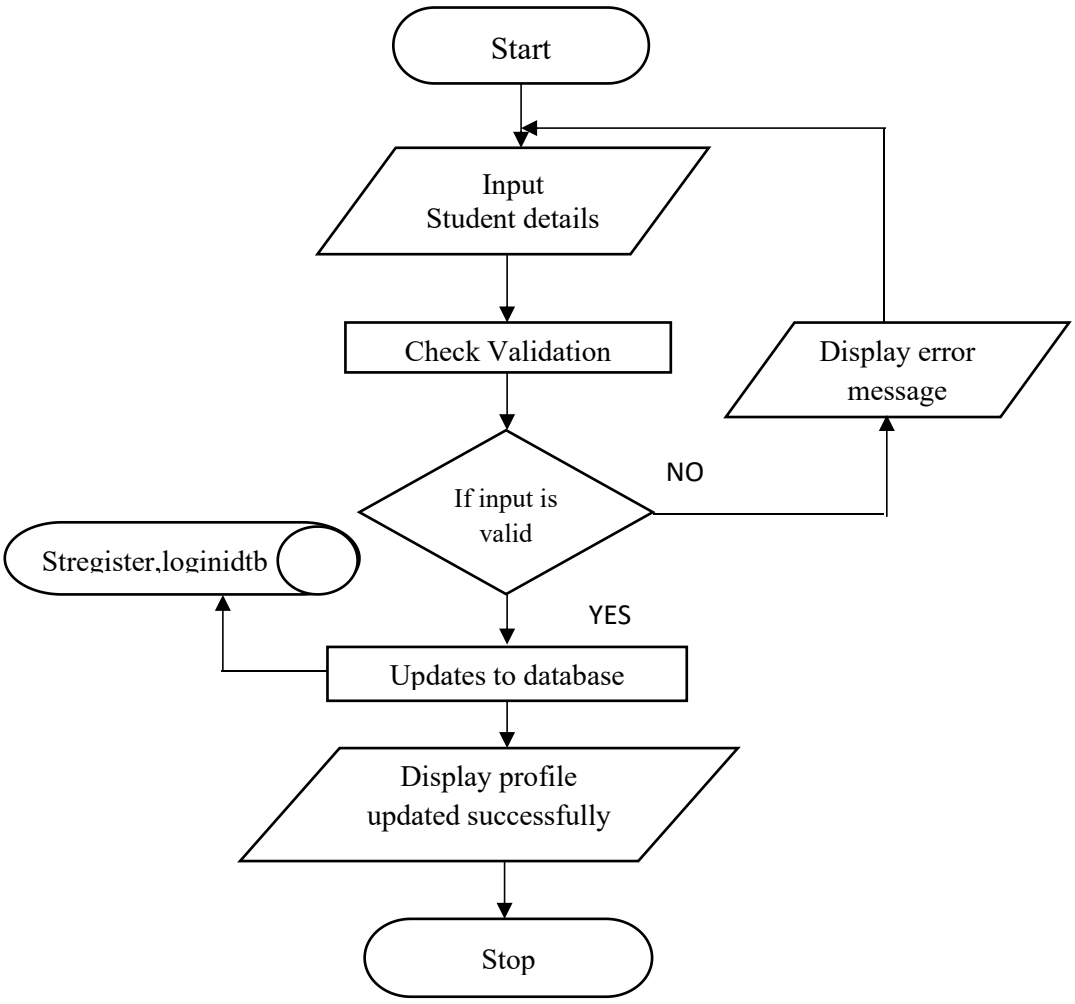


Figure 5.17 Update my profile Flowchart

5.3.5.2.6.3 File I/O interfaces: stregister, loginidtb.

5.3.5.2.6.4 Outputs: Updates data to databases and displays profile updated successfully message

5.3.5.2.6.5 Implementation aspects: Button, Textbox and labels

5.3.5.3 View food menu

5.3.5.3.1 Input: Button click

5.3.5.3.2 Procedural details

Structured chart:

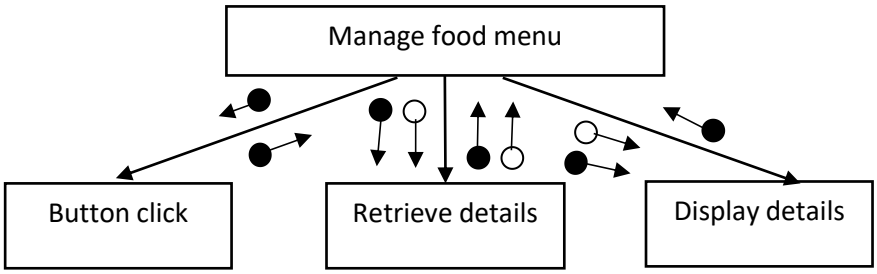


Figure 5.18 View food menu structure chart

5.3.5.3.3 File I/O interfaces: foodmenu, payment.

5.3.5.3.4 Outputs: Displays the record from database

5.3.5.3.5 Implementation aspects: Button, Textbox and labels

5.3.5.4 View event

5.3.5.4.1 Input: Button click

5.3.5.4.2 Procedural details

Flowchart:

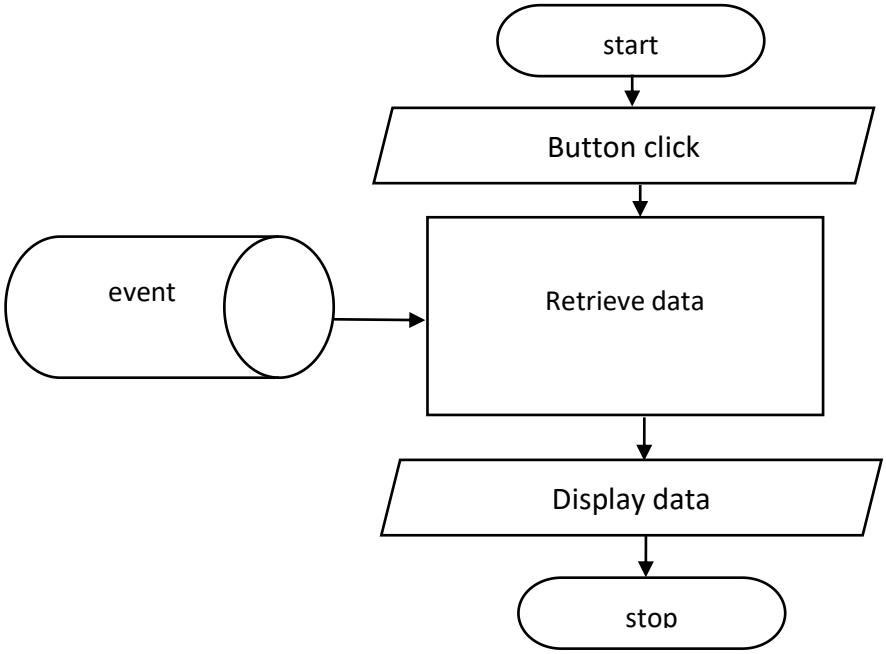


Figure 5.19 View event Flowchart

5.3.5.4.3 File I/O interfaces: event.

5.3.5.4.4 Outputs: Displays all the event record from database

5.3.5.5.5 Implementation aspects: Button, Textbox and labels

5.3.5.5 Renewal

5.3.5.5.1 Input: card number, cardholder name, expiry year and month, cvv, extended month.

5.3.5.5.2 Procedural details

Algorithm:

Step 1: Start

Step 2: retrieve details

Step 3: Input card number, cardholder name, expiry year and month, cvv, extended month.

Step 4: Reads and validates the input

Step 5: IF input is valid THEN

GOTO Step 6

ELSE

Displays “invalid input message”

GOTO Step 3

END IF

Step 6: Updates the details to the database.

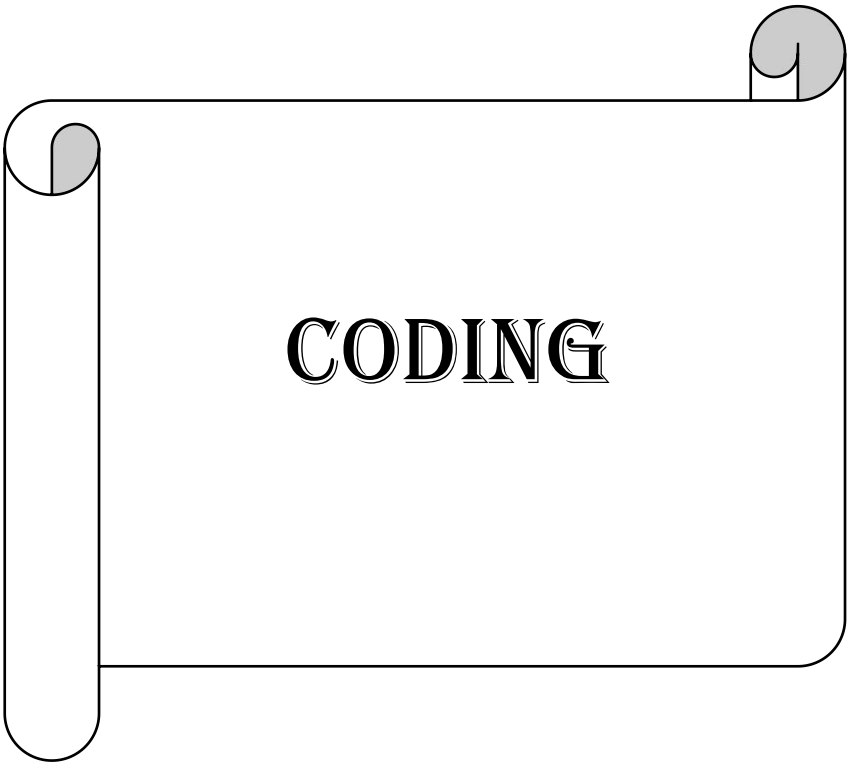
Step 7: Displays “Renewal successful”.

Step 8: End

5.3.5.5.3 File I/O interfaces: Stregister, payment.

5.3.5.5.4 Outputs: Updates the record to the database and displays “renewal successful” message.

5.3.5.5.5 Implementation aspects: Button, Textbox and labels


CODING

6. CODING

- **Login**

```
<?php

session_start();
if(isset($_POST['loginbut']))
{
 $inputadminuser=$_POST['user'];
 $inputadminpass=$_POST['pass'];

 $con=new mysqli("localhost","root","","hostel");
 if($con->connect_error)
 {
  echo"<script>alert('Could Not Connect');</script>";
  echo"<script>window.location.href='index.html';</script>";
 }

 if(isset($_GET['username']))
 {
  $username="";
 }
 else
 {
  $username=$inputadminuser;
 }

 $adminun="admin";
 $adminue="dkiran4661@gmail.com";
 $adminp="admin";
 //admin login
 if($inputadminuser==$adminun or $inputadminuser==$adminue)
 {
```

```
    if($inputadminpass==$adminp)
    {

echo'<script>window.location.href="http://localhost/hostel/admin/admindashboard.php";
</script>';
    }
    else
    {
      echo"<script>alert('Invalid Password');</script>";
      echo"<script>window.location.href='index.html';</script>";
    }
  }
  else
  {

    $abc = "SELECT *
    FROM stregister
    JOIN loginidtb ON stregister.email = loginidtb.usermail
    WHERE (loginidtb.usermail = '".$inputadminuser."' OR loginidtb.userid =
'".$inputadminuser."')";
    $idrslt=$con->query($abc);
    $bqry="select * from warden where email='".$inputadminuser."' or
wid='".$inputadminuser."'";
    $srslt=$con->query($bqry);


    if($idrslt->num_rows!=0)
    {
      $rr=$idrslt->fetch_assoc();
      $upass=$rr['password'];
      if($inputadminpass==$upass)
      {
```

```
echo'<script>window.location.href="http://localhost/hostel/student/studentdashboard.php?
username='.$username.'";</script>';
     }
     else
     {
       echo"<script>alert('Invalid Password');</script>";
       echo"<script>window.location.href='index.html';</script>";
     }
   }
   elseif($srslt->num_rows!=0)
   {
     $zz=$srslt->fetch_assoc();
     $upass=$zz['wpass'];
     if($inputadminpass==$upass)
     {

echo'<script>window.location.href="http://localhost/hostel/warden/wardendashboard.ph
p?username='.$username.'";</script>';
     }
     else
     {
       echo"<script>alert('Invalid Password');</script>";
       echo"<script>window.location.href='index.html';</script>";
     }
   }
   else
   {
       echo"<script>alert('Invalid Username');</script>";
       echo"<script>window.location.href='index.html';</script>";
   }
  }
}
?>
```

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8" />
  <link href="https://fonts.googleapis.com/css?family=Lato:400,700&display=swap" rel="stylesheet">
  <link rel="stylesheet" type="text/css" href="styles.css" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
 </head>
 <body>
  <div class="rect1l">

  </div>

  <div class="rect2l"></div>

  <div class="rect3l"></div>
  <form method="POST" action="index.php">
  <div class="login-div">
   <div class="logo"></div>
   <div class="title">HOSTEL MANAGEMENT</div>
   <div class="sub-title">LOGIN</div>
   <div class="fields">
    <div class="username"><svg fill="#999" viewBox="0 0 1024 1024"><path class="path1" d="M896 307.2h-819.2c-42.347 0-76.8 34.453-76.8 76.8v460.8c0 42.349 34.453 76.8 76.8 76.8h819.2c42.349 0 76.8-34.451 76.8-76.8v-460.8c0-42.347-34.451-76.8-76.8-76.8zM896 358.4c1.514 0 2.99 0.158 4.434 0.411l-385.632 257.090c-14.862 9.907-41.938 9.907-56.802 0l-385.634-257.090c1.443-0.253 2.92-0.411 4.434-0.411h819.2zM896 870.4h-819.2c-14.115 0-25.6-11.485-25.6-25.6v-438.566l378.4 252.267c15.925 10.618 36.363 15.925 56.8 15.925s40.877-5.307 56.802-15.925l378.398-252.267v438.566c0 14.115-11.485 25.6-25.6
```

```
25.6z"></path></svg><input type="username" class="user-input" id="user" name="user" required placeholder="username" /></div>
      <div class="password"><svg fill="#999" viewBox="0 0 1024 1024"><path class="path1" d="M742.4 409.6h-25.6v-76.8c0-127.043-103.357-230.4-230.4-230.4s-230.4 103.357-230.4 230.4v76.8h-25.6c-42.347 0-76.8 34.453-76.8 76.8v409.6c0 42.347 34.453 76.8 76.8 76.8h512c42.347 0 76.8-34.453 76.8-76.8v-409.6c0-42.347-34.453-76.8-76.8-76.8zM307.2 332.8c0-98.811 80.389-179.2 179.2-179.2s179.2 80.389 179.2 179.2v76.8h-358.4v-76.8zM768 896c0 14.115-11.485 25.6-25.6 25.6h-512c-14.115 0-25.6-11.485-25.6-25.6v-409.6c0-14.115 11.485-25.6 25.6-25.6h512c14.115 0 25.6 11.485 25.6 25.6v409.6z"></path></svg><input type="password" class="pass-input" id="pass" name="pass" placeholder="password" required/></div>
    </div>
    <div class="bal">
    <button class="signin-button" name="loginbut">Login</button>
   </div>
    <div class="link">
     <h5><a href="http://localhost/hostel/login/fpas.html">Forgot password?</a> </h5>
</form>

    </div>
    <div class="rect1r">

    </div>

    <div class="rect2r"></div>

    <div class="rect3r"></div>
   </div>
  </body>
</html>
```

- **Register**

```
<?php
   session_start();
   if(isset($_POST['register']))
   {
    $fname=$_POST['fname'];
    $lname=$_POST['lname'];
    $fmname=$_POST['fmname'];
    $dob=$_POST['dob'];
    $blood=$_POST['blood'];
    $gender=$_POST['gender'];
    $caste=$_POST['caste'];
    $adhar=$_POST['adhar'];

    $_SESSION['name']=$fname;
    $_SESSION['gen']=$gender;

    $clgname=$_POST['clgname'];
    $regno=$_POST['regno'];
    $course=$_POST['course'];
    $rollno=$_POST['rollno'];
    $coursedur=$_POST['coursedur'];
    $pymark=$_POST['pymark'];
    $joindate=$_POST['joindate'];
    $stay=$_POST['stay'];

    $_SESSION['rollid']=$rollno;
    $_SESSION['regid']=$regno;
    $_SESSION['staypay']=$stay;
    $_SESSION['jdate']=$joindate;

    $mobno=$_POST['mobno'];
```

```
$pmobno=$_POST['pmobno'];
$regemail=$_POST['regemail'];
$paddress=$_POST['paddress'];
$country=$_POST['country'];
$state=$_POST['state'];
$district=$_POST['district'];
$pincode=$_POST['pincode'];

$student_status="ACTIVE";
$roomid="";
$_SESSION['emid']=$regemail;
//setting connection and checking for existence of email
$con=new mysqli("localhost","root","","hostel");
if($con->connect_error)
{
  echo"<script>alert('Could Not Connect');</script>";
  echo"<script>window.location.href='home.html';</script>";
}
$qry="select * from stregister where email='".$regemail."' or regno='".$regno."'";
$idrslt=$con->query($qry);

if($idrslt->num_rows==1)
{
  echo"<script>alert('Already applied with this Email/Register number');</script>";
  echo"<script>window.location.href='http://localhost/hostel/home/home.html';</script>";
}
elseif($mobno==$pmobno)
{
  echo"<script>alert('Personal Number and Parent Number should be different');</script>";
  echo"<script>window.location.href='register.html';</script>";

}
else
{
```

```
$iqry="insert into stregister values('".$fname."','".$lname."','".$fmname."','".$dob."','".$blood."','".$gender."','".$caste."','".$adhar."','".$clgname."','".$regno."','".$course."','".$rollno."','".$coursedur."','".$pymark."','".$joindate."','".$stay."','".$mobno."','".$pmobno."','".$regemail."','".$paddress."','".$country."','".$state."','".$district."','".$pincode."','".$roomid."','".$student_status."')";
$ins=$con->query($iqry);
//if inserted successfully
if($ins)
{
 //sending email with mail function, message:successfull registration
 $receiver = "$regemail";
 $subject = "Applied for Hostel Room";
 $body = "Dear $fname,\nWelcome to our Hostel.\nYou have Successfully submitted the Application to the Hostel.\nComplete the Payment procedure to book a room.\nThank You";
 $sender = "From:dkiran4661@gmail.com";
 if(mail($receiver, $subject, $body, $sender))
 {
  echo"<script>alert('Applied successfully, Goto the Payment section to complete registration');</script>";
  echo"<script>window.location.href='pay1.php';</script>";
 }
 else
 {
  echo"<script>alert('Your offline to recieve Email, But successfully submitted Application. Goto the Payment section to complete registration');</script>";
  echo"<script>window.location.href='pay1.php';</script>";
 }
}
else
{
 echo'<script>alert("Some Error Occured During The Submission");</script>';
 echo"<script>window.location.href='pay1.php';</script>";
}
```

```
    }
  }
?>



<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="register.css">
  <title>Document</title>
<script>
 function backbutton(){
        window.location.href='http://localhost/hostel/home/home.html';

 }
 function db() {
  var button = document.getElementById("sub");
  button.disabled = true;

 }
 function validateInput(event) {
  const input = event.target;
  const value = input.value;
  const sanitizedValue = value.replace(/[0-9]/g, '');
  input.value = sanitizedValue;
 }
```

```
function sanitizeBloodGroup(event) {
  const input = event.target;
  let value = input.value.toUpperCase(); // Convert to uppercase for consistent comparison

  // List of valid blood groups
  const validBloodGroups = ['A', 'B', 'AB', 'O', 'A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'];

  // Check if the input value is a valid blood group
  if (!validBloodGroups.includes(value)) {
    value = ''; // Clear the input value if it is not valid
  }

  input.value = value;
}




function disableSubmitButton() {
  var submitButton = document.getElementById('sub');

  setTimeout(function() {
    submitButton.disabled = true;
  }, 10);
}
document.addEventListener("DOMContentLoaded", function() {
  // Get today's date
  var today = new Date();

  // Format the date as "YYYY-MM-DD"
  var formattedDate = today.toISOString().split('T')[0];

  // Set the value of the date input field
  document.getElementById("jdate").value = formattedDate;
```

```
  });

  function updateMinCheckoutDate() {
    // Get the selected joining date
    var joiningDateInput = document.getElementById("jdate");
    var joiningDate = new Date(joiningDateInput.value);

    // Calculate the minimum checkout date (6 months after joining date)
    var minCheckoutDate = new Date(joiningDate);
    minCheckoutDate.setMonth(joiningDate.getMonth() + 6);

    // Format the date as "YYYY-MM-DD"
    var formattedDate = minCheckoutDate.toISOString().split('T')[0];

    // Set the minimum value of the checkout date input field
    document.getElementById("checkoutDate").setAttribute("min", formattedDate);

    // Reset the value of the checkout date input field
    document.getElementById("checkoutDate").value = "";

    // Enable dates starting from the minimum date in the date picker
    document.getElementById("checkoutDate").removeAttribute("disabled");
  }

</script>
<style>
    input{
        text-transform: uppercase;
    }
</style>
</head>
<body onload="updateMinCheckoutDate()">

    <div class="container">
```

```
<form method="POST" action="register.php" onsubmit="disableSubmitButton()">
    <div class="logo">
    <h1 class="hdng">REGISTER</h1>
    </div>

    <div class="pd">

        <div class="pdalign">


            <input class="name" type="text" name="fname" placeholder="FIRST NAME" id="myInput" oninput="validateInput(event)" required>
            <input class="fname" type="text" name="lname" placeholder="LAST NAME" id="myInput" oninput="validateInput(event)">
            <input class="fname" type="text" name="fmname" placeholder="FATHER / MOTHER NAME" required id="myInput" oninput="validateInput(event)">
            <input class="dob" type="date" name="dob" title="BIRTH DATE" placeholder="DATE OF BIRTH" min="1990-01-01" max="2010-01-01" required>
            <input class="name" type="text" name="blood" placeholder="BLOOD GROUP" required id="myBloodg" oninput="sanitizeBloodGroup(event)">

            <select class="selalgn" name="gender" id="" title="Select Gender">
            <!-- <input class="fname" type="text" placeholder="GENDER"> -->
            <option >MALE</option>
            <option >FEMALE</option>
        </select>
            <input class="fname" type="text" placeholder="CASTE" name="caste" required id="myInput" oninput="validateInput(event)">
            <input class="fname" type="number" min="100000000000" max="999999999999" placeholder="AADHAR NUMBER" name="adhar" required>

        </div>
    </div>
```

```
</div>
<div class="eq">
  <div class="eqalign">
    <input class="name" type="text" placeholder="COLLEGE NAME" name="clgname" required id="myInput" oninput="validateInput(event)">
    <input class="fname" type="text" placeholder="REGISTER NUMBER" name="regno" required>
    <select id="" name="course" title="Select Course">
      <option>BCA</option>
      <option>BBA</option>
      <option>BCOM</option>
      <option>BSC</option>
      <option>BA</option>
    </select>
    <input class="fname" type="text" placeholder="ROLL NUMBER" name="rollno" required>
    <input class="name" type="number" min="1" max="4" placeholder="COURSE DURATION" name="coursedur" required>
    <input class="fname" type="number" min="35" max="100" placeholder="PREVIOUS YEAR MARKS IN %" name="pymark" required>
    <input class="join" type="date" title="JOINING DATE" placeholder="JOINING DATE" name="joindate" id="jdate" readonly>
    <input class="fname" type="date" name="stay" title="CHECKOUT DATE" id="checkoutDate">
    </div>
</div>
<div class="ad">
  <div class="adalign">
    <input class="name" type="number" min="1000000000" max="9999999999" placeholder="MOBILE NUMBER" name="mobno" required>
    <input class="fname" type="number" min="1000000000" max="9999999999" placeholder="PARENT'S MOBILE NUMBER" name="pmobno" required>
    <input class="dob" type="email" placeholder="EMAIL" name="regemail">
```

```
            <input class="fname" type="text" placeholder="PERMANENT ADDREESS" name="paddress" required>
            <input class="name" type="text" placeholder="COUNTRY" name="country" required id="myInput" oninput="validateInput(event)">
            <input class="fname" type="text" placeholder="STATE" name="state" required id="myInput" oninput="validateInput(event)">
            <input class="dob" type="text" placeholder="DISTRICT" name="district" required id="myInput" oninput="validateInput(event)">
            <input class="fname" type="number" min="100000" max="999999" placeholder="PINCODE" name="pincode" required>
            </div>

        </div>

        <div class="balgn">
        <button class="signin" name="register" id="sub" >SUBMIT</button>

      </form>

        <button class="back" onclick="backbutton();" >BACK</button>
      </div>

   </div>

</body>
</html>
```

- **Dashboard**

```
<?php
session_start();
$con=new mysqli("localhost","root","","hostel");
if($con->connect_error)
{
 echo"<script>alert('Could Not Connect');</script>";
 echo"<script>window.location.href='home.html';</script>";
}

@$us=$_SESSION['usn'];

//number of student
$qry="select * from stregister";
$stc=$con->query($qry);
$stcount=$stc->num_rows;

//number of Boys
$qry="select * from stregister where gender='MALE'";
$stb=$con->query($qry);
$stboys=$stb->num_rows;

//number of Boys
$qry="select * from stregister where gender='FEMALE'";
$stg=$con->query($qry);
$stgirls=$stg->num_rows;

//number of available rooms
$qry="select * from mroom where status='AVAILABLE'";
$availmroom=$con->query($qry);
$availroomcount=$availmroom->num_rows;
```

```
//number of rooms
$qry="select * from mroom";
$mroom=$con->query($qry);
$roomcount=$mroom->num_rows;

//number of boys and girls rooms
$qry="select * from mroom where roomfor='MALE'";
$mroomb=$con->query($qry);
$broomcount=$mroomb->num_rows;

$qry="select * from mroom where roomfor='FEMALE'";
$mroomg=$con->query($qry);
$groomcount=$mroomg->num_rows;


//NUMBER OF FILLED BOYS AND GIRLS ROOMS
$qry="select * from mroom where status='FULL' and roomfor='MALE'";
$mroombf=$con->query($qry);
$bfroomcount=$mroombf->num_rows;

$qry="select * from mroom where status='FULL' and roomfor='FEMALE'";
$mroomgf=$con->query($qry);
$gfroomcount=$mroomgf->num_rows;


//no of wardens
$qry="select * from warden";
$warden=$con->query($qry);
$total_warden=$warden->num_rows;



//number of male warden
$qry="select * from warden where wgender='MALE'";
```

```
    $mwarden=$con->query($qry);
    $malewarden=$mwarden->num_rows;

    //number of female warden
    $qry="select * from warden where wgender='FEMALE'";
    $fwarden=$con->query($qry);
    $femalewarden=$fwarden->num_rows;



?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="dash.css">
    <link rel="stylesheet" href="fontawesome-free-6.4.0-web/css/all.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.1/css/all.min.css">

    <title>Document</title>
</head>
<body>
    <div class="admin">
        <i class="fa fa-wallet"></i>
        <h1>ADMIN</h1>
    </div>

    <div class="sidebar">
        <div class="info">
```

```
        <div class="img"></div>
        <labe class="hiadmin">Hi,<span>Admin</span></label>
        <div class="emma"><label class="email" for="">dkiran@gmail.com</label></div>
    </div>
    <div class="menu">

        <a href="admindashboard.php" class="active"><i class="fas fa-graduation-cap"></i> DASHBOARD</a>
        <a href="managestudent.html"><i class="fas fa-graduation-cap"></i> MANAGE STUDENTS</a>
        <a href="managewarden.html"><i class="fas fa-graduation-cap"></i> MANAGE WARDENS</a>

        <a href="manageroom.html" ><i class="fas fa-graduation-cap"></i> MANAGE ROOMS</a>
        <a href="managefood.php"><i class="fas fa-graduation-cap"></i> MANAGE FOODS</a>
        <a href="manageevent.php"><i class="fas fa-graduation-cap"></i> MANAGE EVENTS</a>
        <a href="spayment.php"><i class="fas fa-graduation-cap"></i> PAYMENT</a>

    </div>
</div>

<div class="navbar">
<h1 class="h1">Dashboard</h1>
<a class="logout" href="http://localhost/hostel/home/load.html"> LOG OUT <i class="fas fa-sign-out-alt"></i> </a>
</div>

<div class="cards">
```

```
    <div class="card1">
        <i class="fas fa-user-circle"></i>
        <h1><?php echo $stcount; ?></h1>
        <label>Total No Of Students</label>
    </div>
    <div class="card2">
        <i class="fas fa-user"></i>
        <h1><?php echo $total_warden; ?></h1>
        <label>Total No Of Wardens</label>
    </div>
    <div class="card3">
        <i class="fas fa-bed"></i>
        <h1><?php echo $roomcount; ?></h1>
        <label>Total No Of Rooms</label>
    </div>
    <div class="card4">
        <i class="fas fa-bed"></i>
        <h1><?php echo $availroomcount;?></h1>
        <label>Available Rooms</label>
    </div>
</div>

<div class="cd">
    <div class="cd1">
        <label class="student" for="">STUDENTS</label>
        <i class="fas fa-male"></i>
        <label class="boys" for=""><?php echo $stboys;?></label>
        <label class="bdata"for="">BOYS</label>
        <div class="div"></div>
        <i class="fas fa-female"></i>
        <label class="girls" for=""><?php echo $stgirls;?></label>
        <label class="gdata"for="">GIRLS</label>
    </div>
    <div class="cd2">
```

```
        <label class="student" for="">WARDENS</label>
        <i class="fas fa-male"></i>
        <label class="boys" for=""><?php echo $malewarden ; ?></label>
        <label class="bdata"for="">MALE</label>
        <div class="div"></div>
        <i class="fas fa-female"></i>
        <label class="girls" for=""><?php echo $femalewarden; ?></label>
        <label class="gdata"for="">FEMALE</label>
    </div>
    <div class="cd3">
        <label class="student" for="">ROOMS</label>
        <i class="fas fa-male"></i>
        <label class="boys" for=""><?php echo $broomcount; ?></label>
        <label class="bdata"for="">BOYS</label>
        <div class="div"></div>
        <i class="fas fa-female"></i>
        <label class="girls" for=""><?php echo $groomcount; ?></label>
        <label class="gdata"for="">GIRLS</label>
    </div>
    <div class="cd4">
        <label class="student" for="">ROOMS</label>
        <i class="fas fa-male"></i>
        <H1>FILLED</H1>
        <label class="boys" for=""><?php echo $bfroomcount;?></label>
        <label class="bdata"for="">BOYS</label>
        <div class="div"></div>
        <i class="fas fa-female"></i>
        <label class="girls" for=""><?php echo $gfroomcount;?></label>
        <label class="gdata"for="">GIRLS</label>
    </div>
</div>
</body>
</html>
```

USER INTERFACE

7. USER INTERFACE

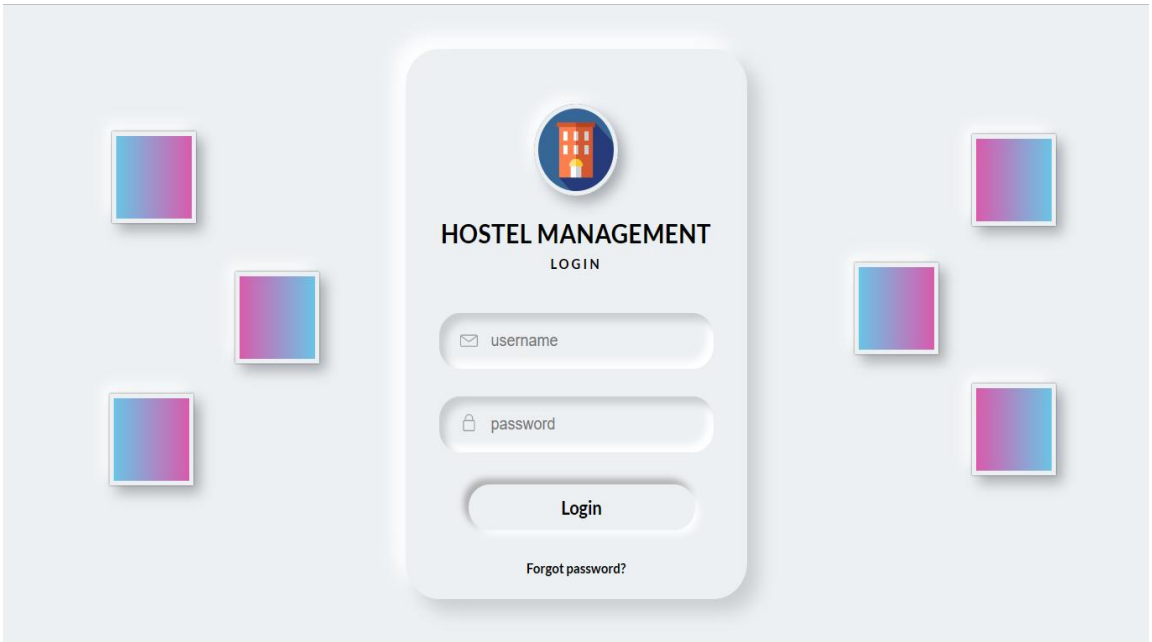


Figure 7.1 login

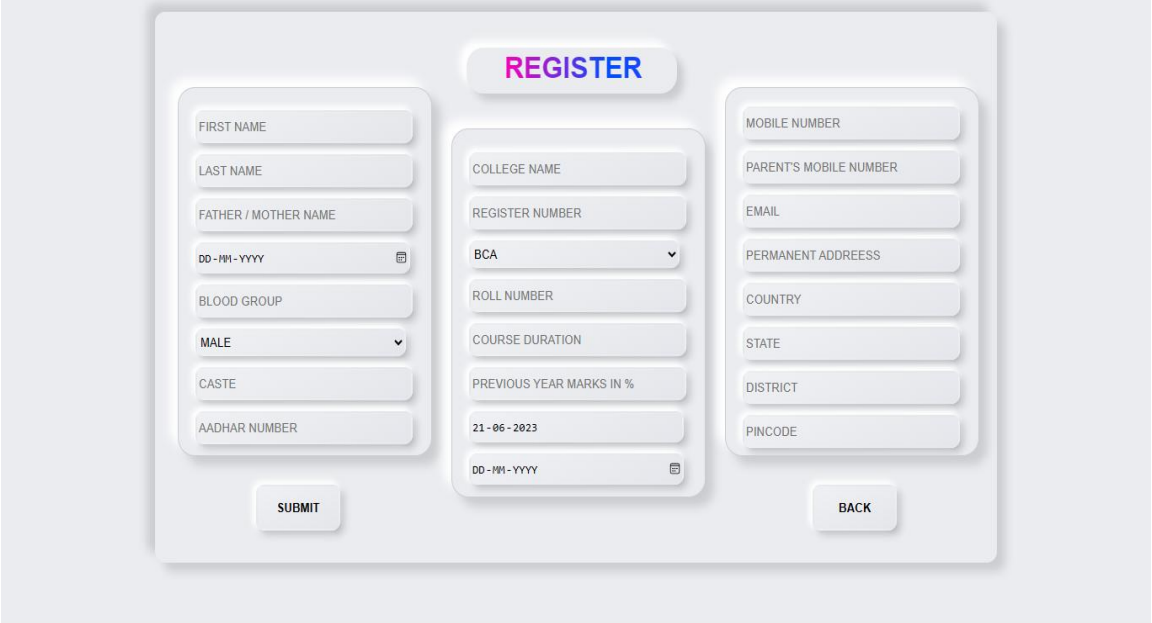


Figure 7.2 Register

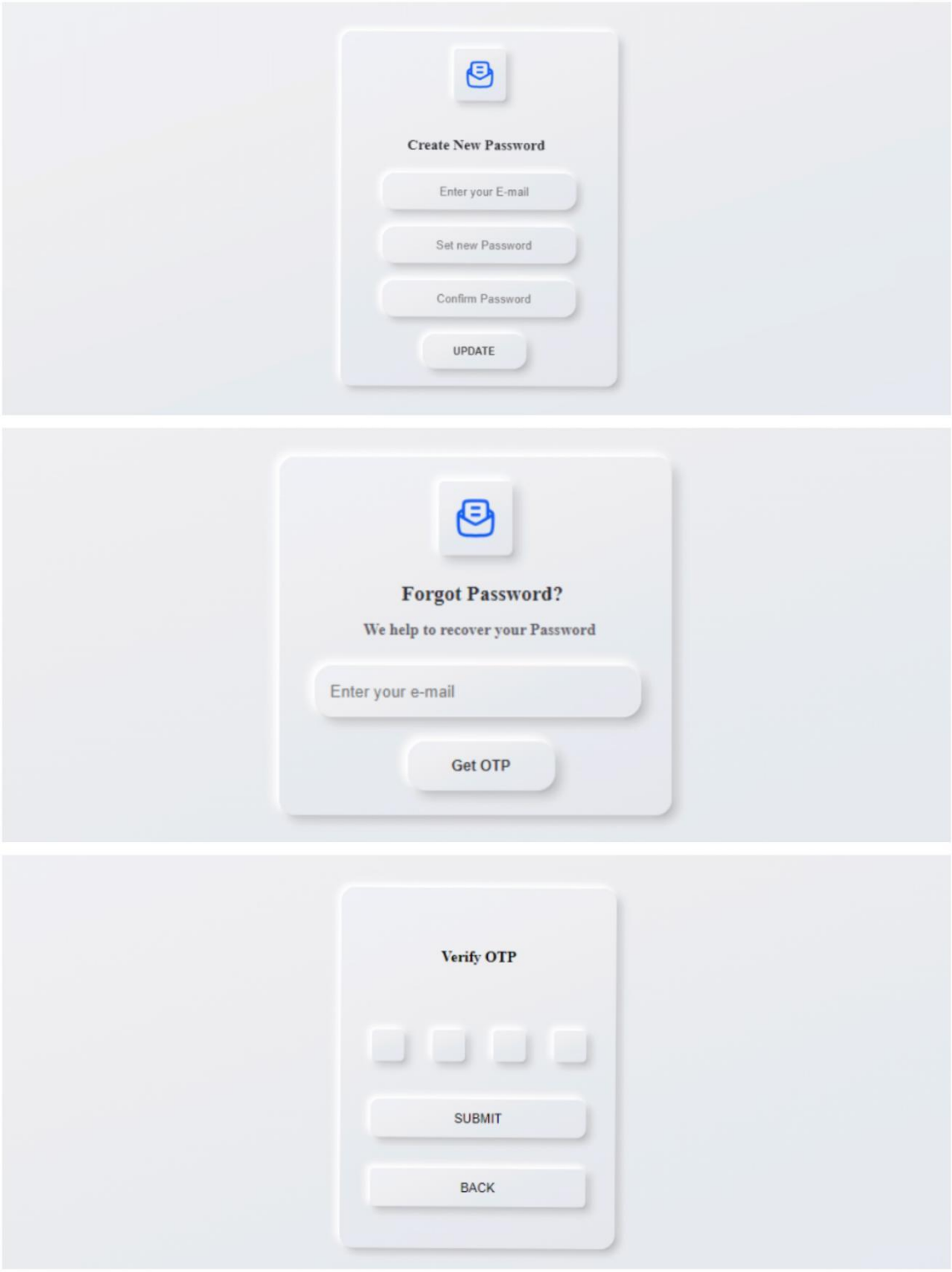


Figure 7.3 Forgot password

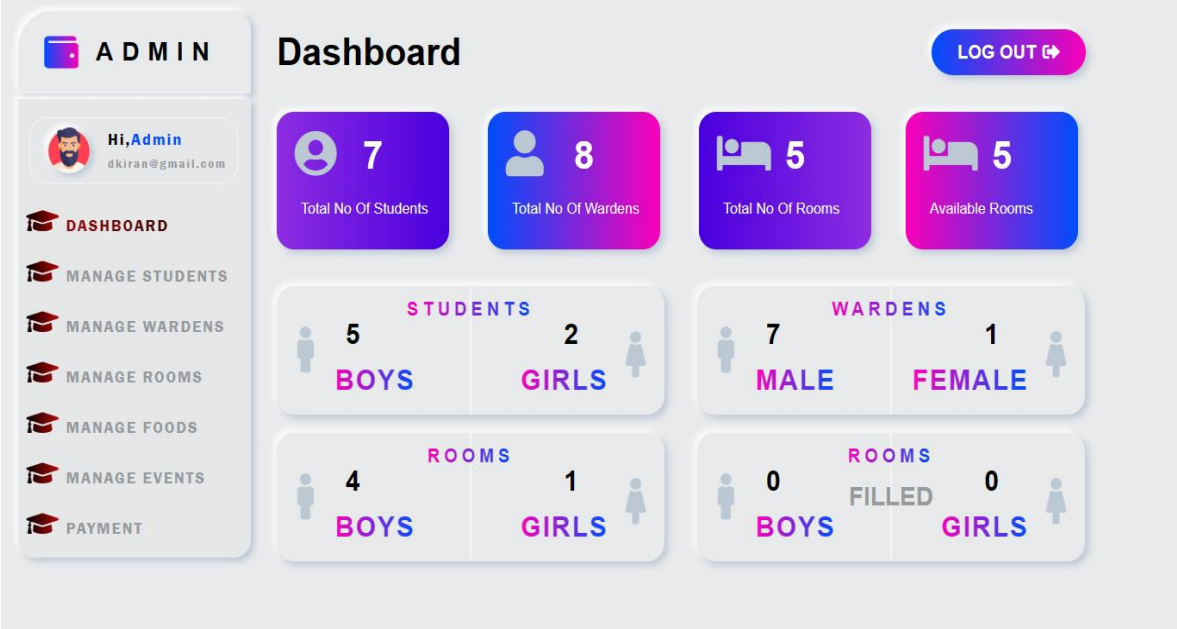


Figure 7.4 Dashboard

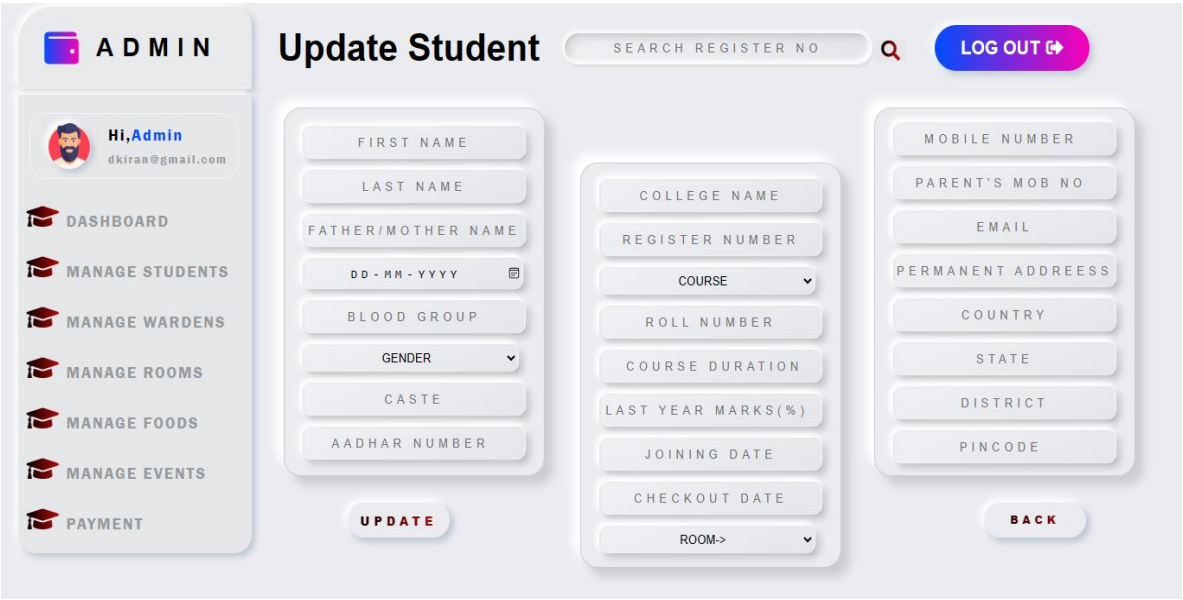


Figure 7.5 Update Student

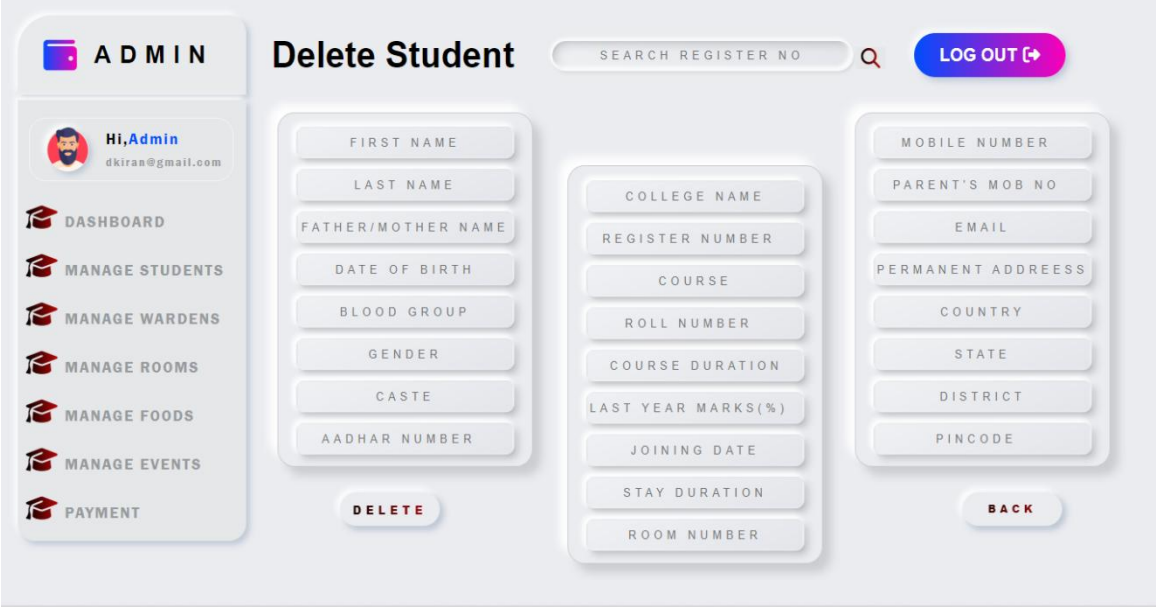


Figure 7.6 Delete Student

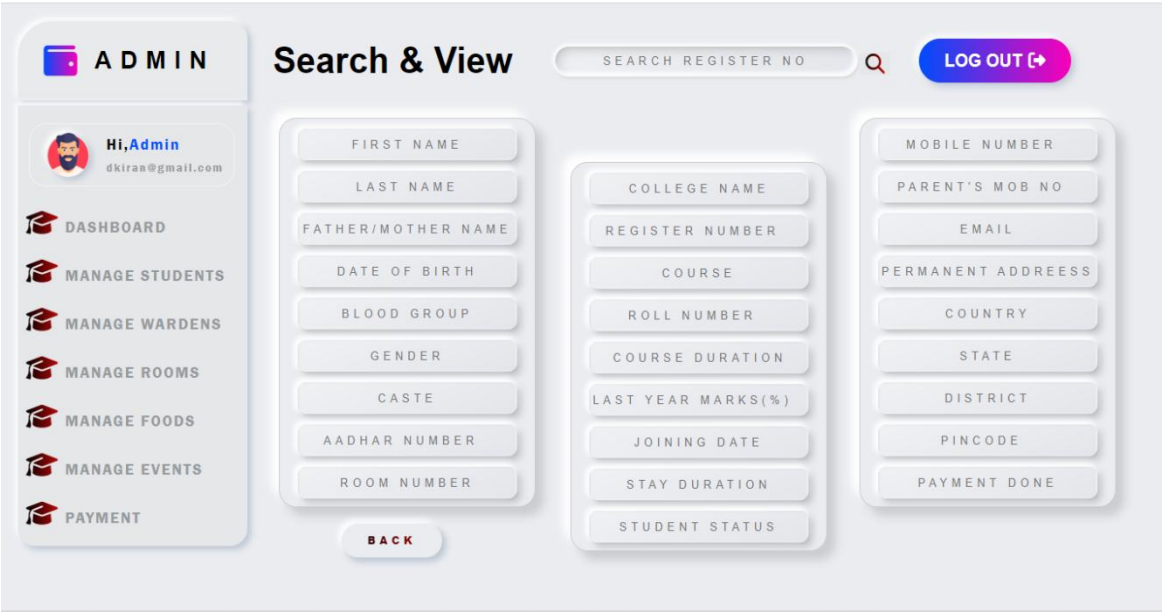


Figure 7.7 Search and view student

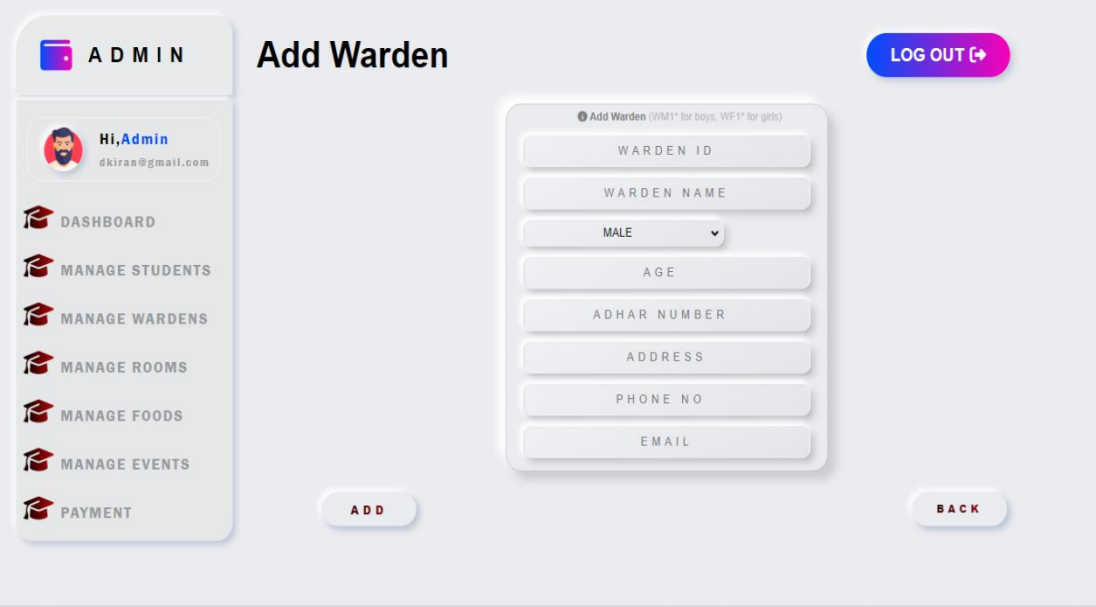


Figure 7.8 Add Warden

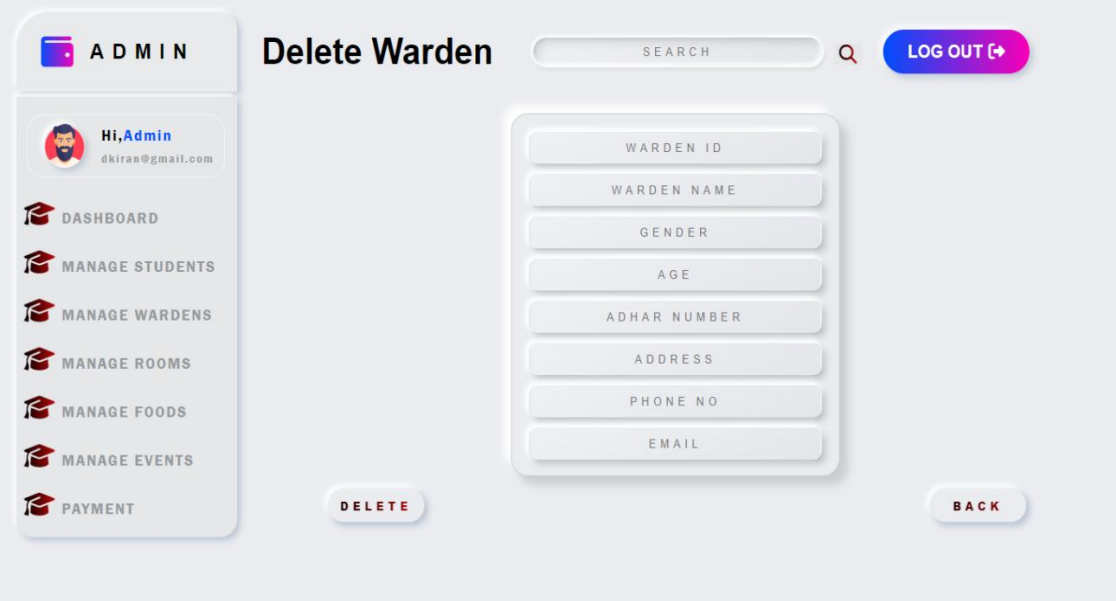


Figure 7.9 Delete warden

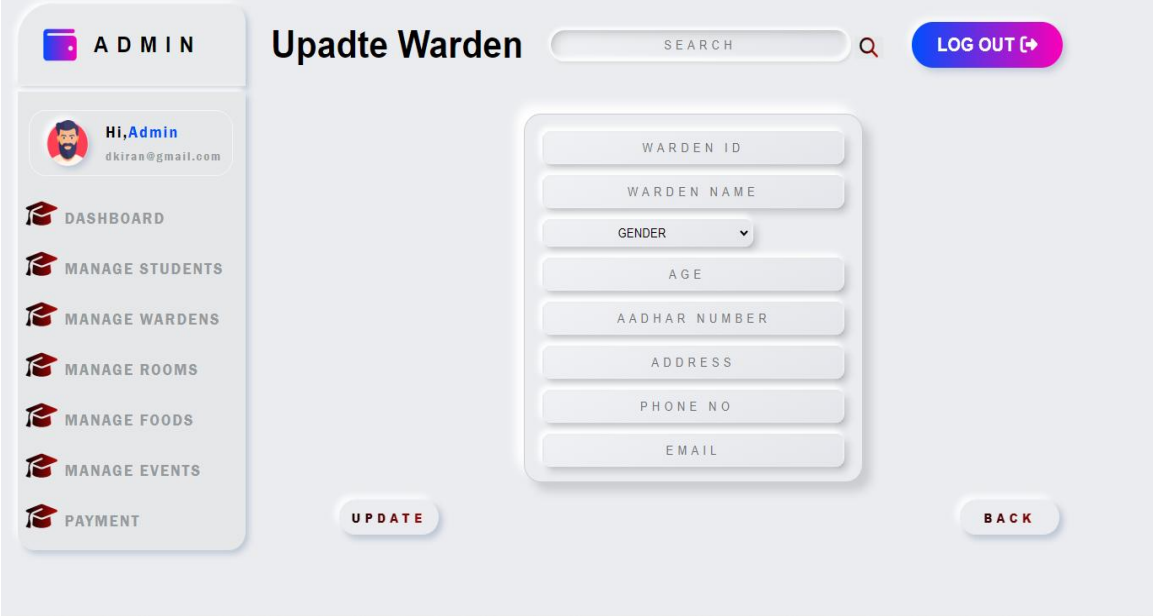


Figure 7.10 Update Warden

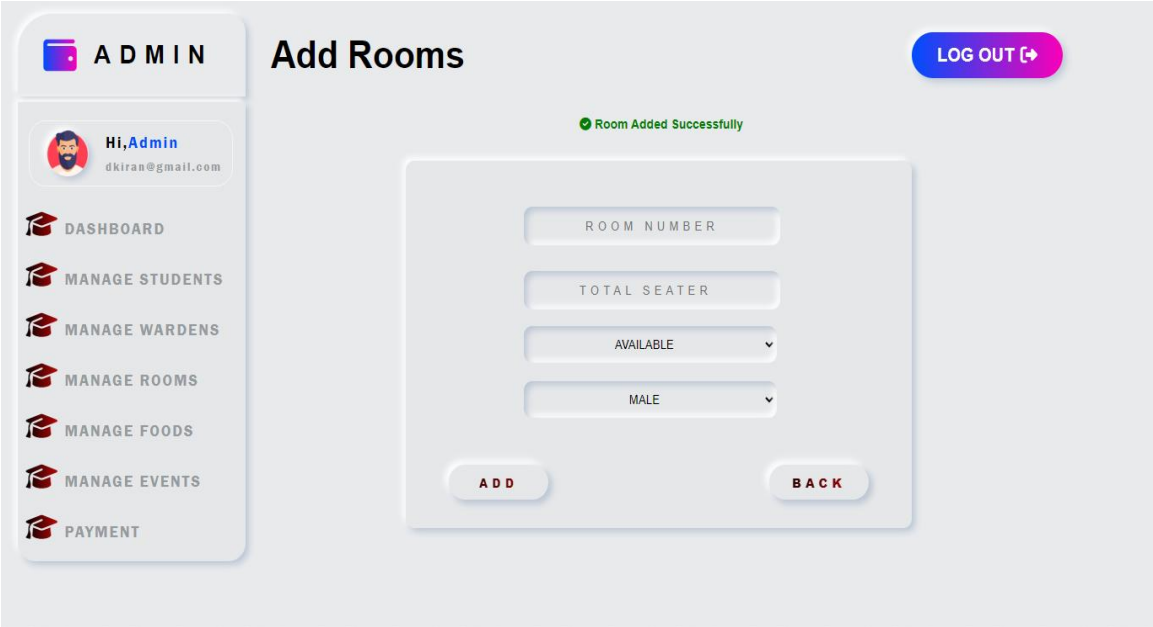


Figure 7.11 Add Room

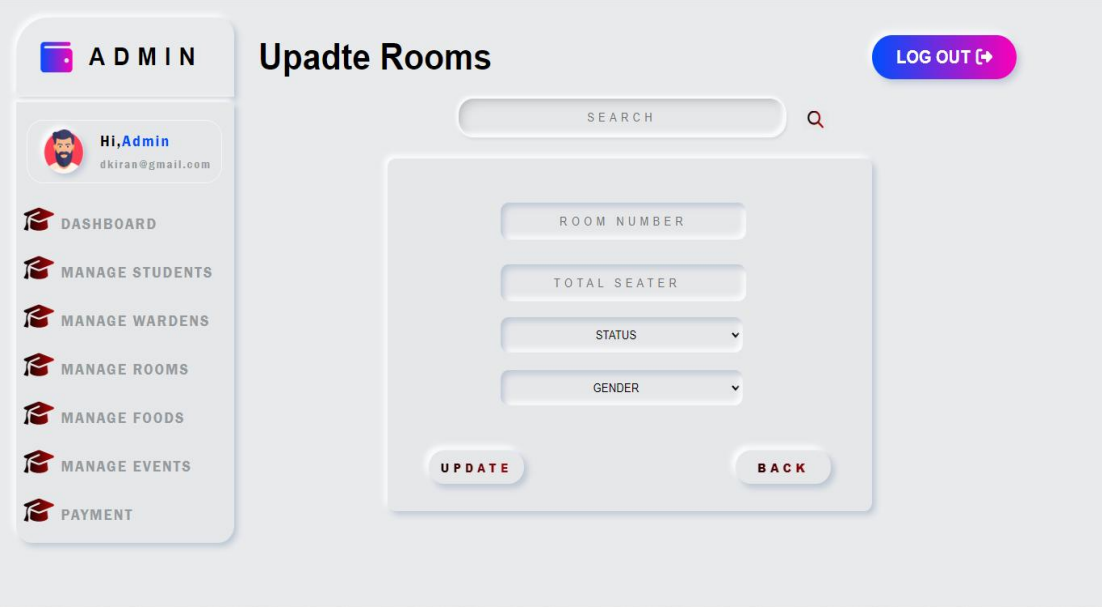


Figure 7.12 Update room

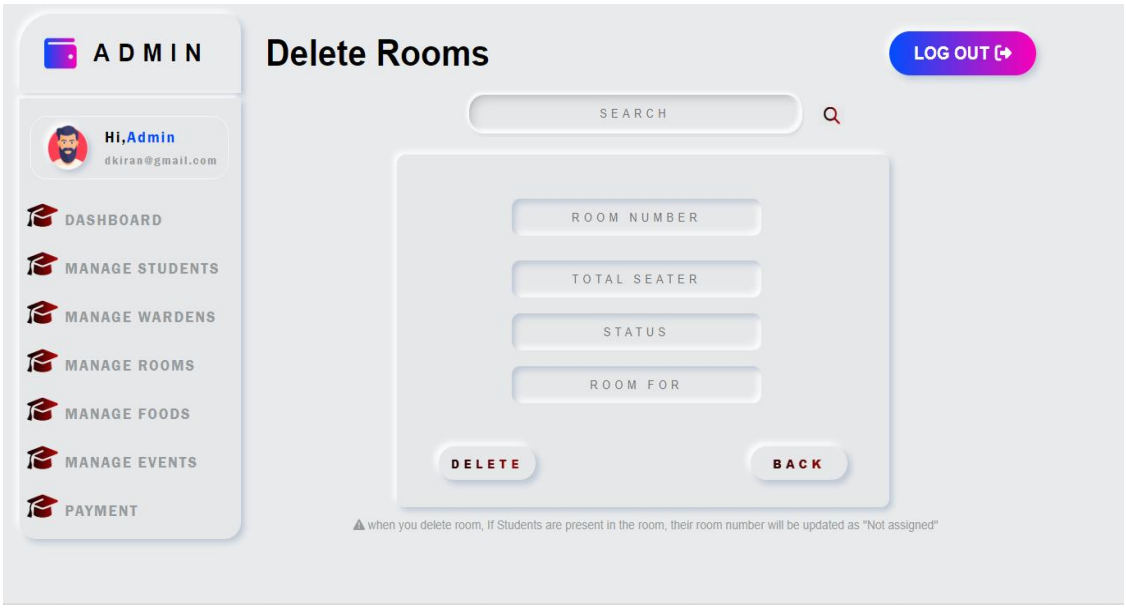


Figure 7.13 Delete Room

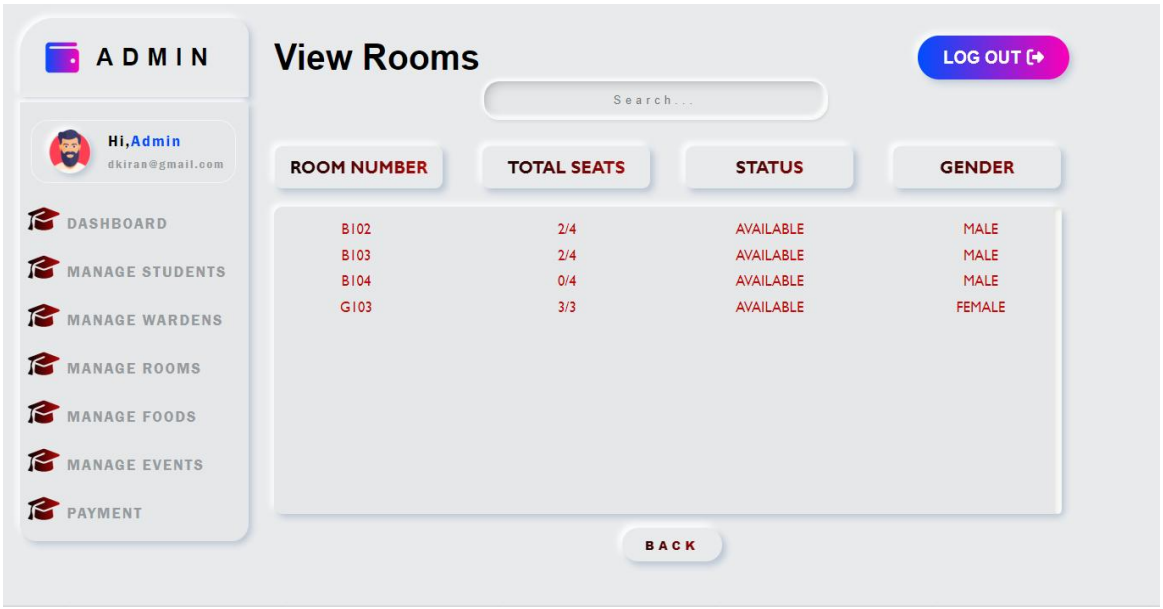


Figure 7.14 View room

Figure 7.15 Manage food menu

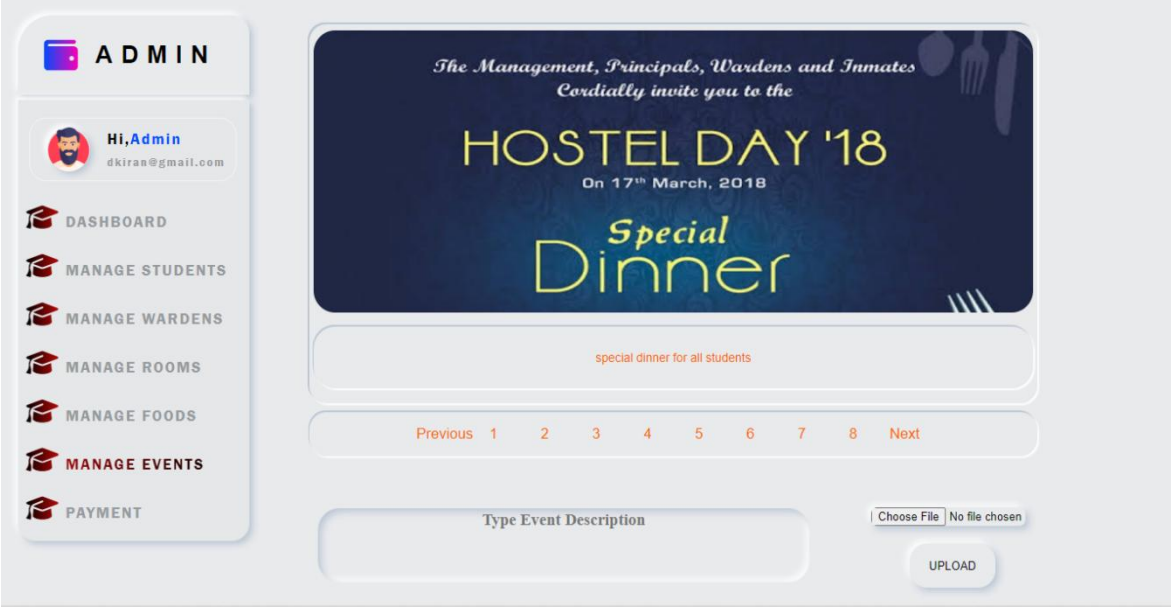


Figure 7.16 Manage event

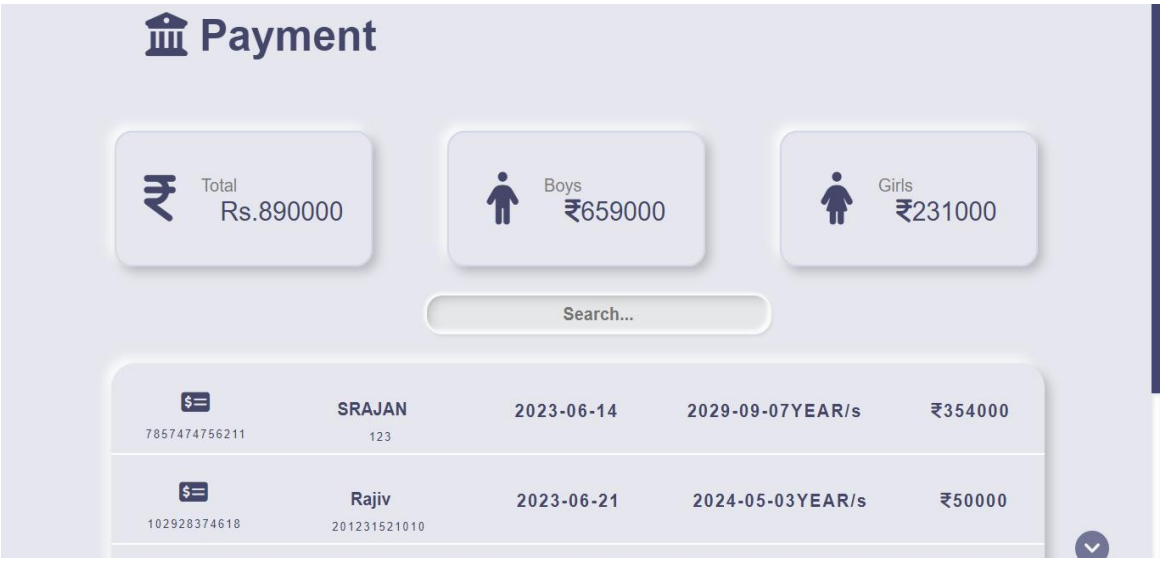


Figure 7.17 View students payment

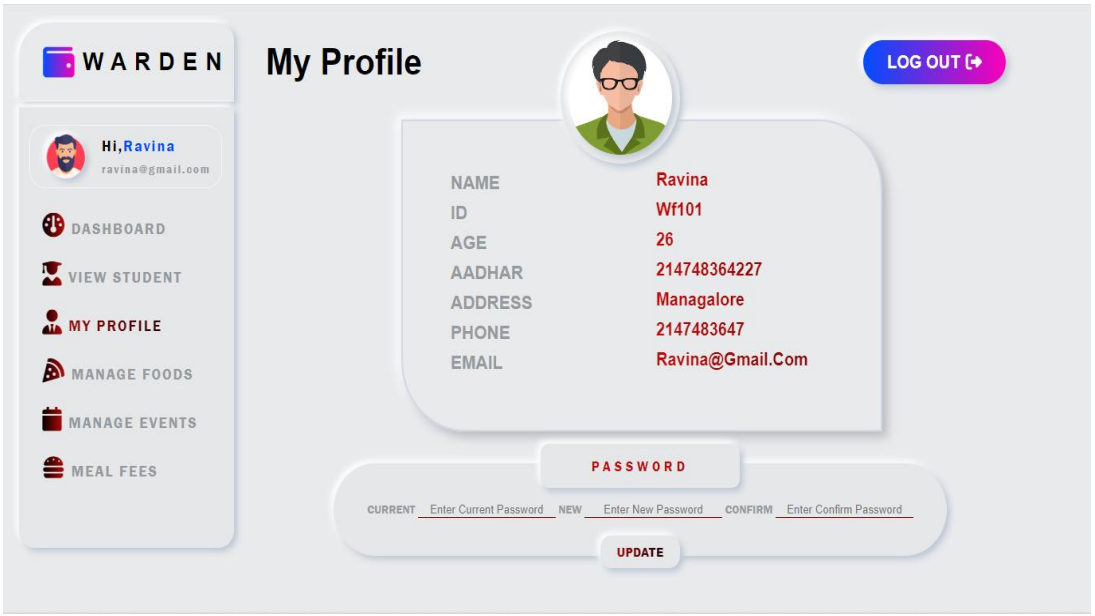


Figure 7.18 My profile

Figure 7.19 Meal Price

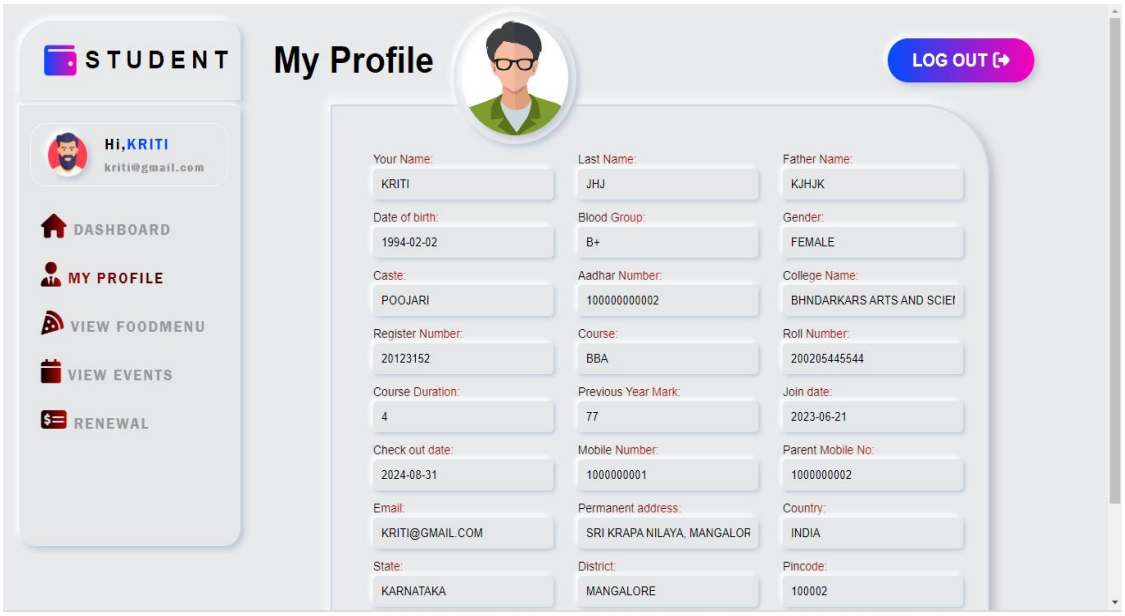


Figure 7.20 student profile

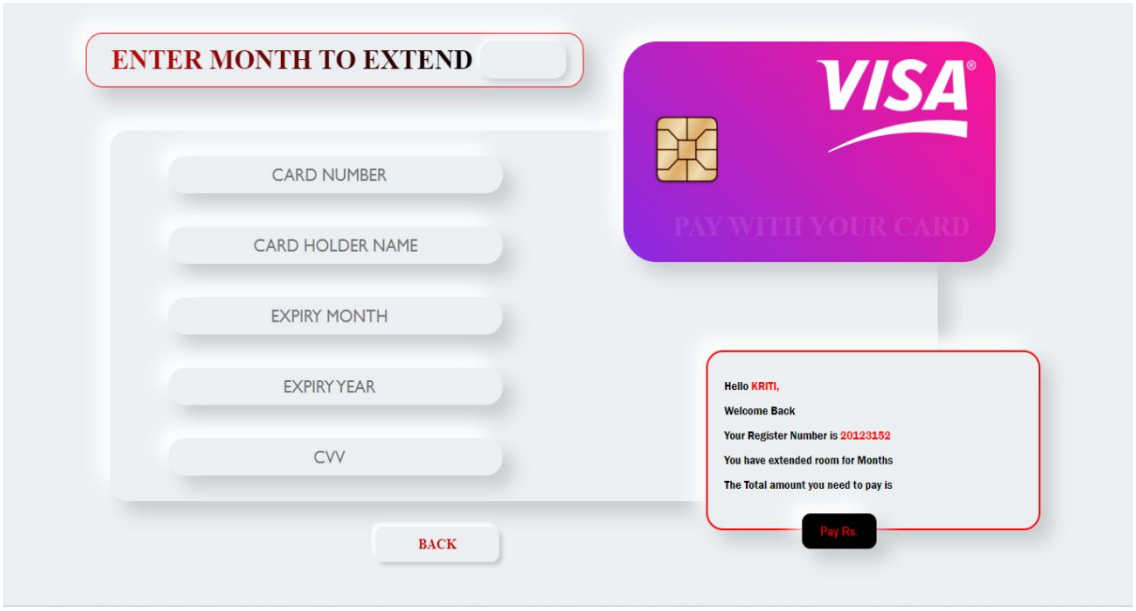


Figure 7.21 Renewal

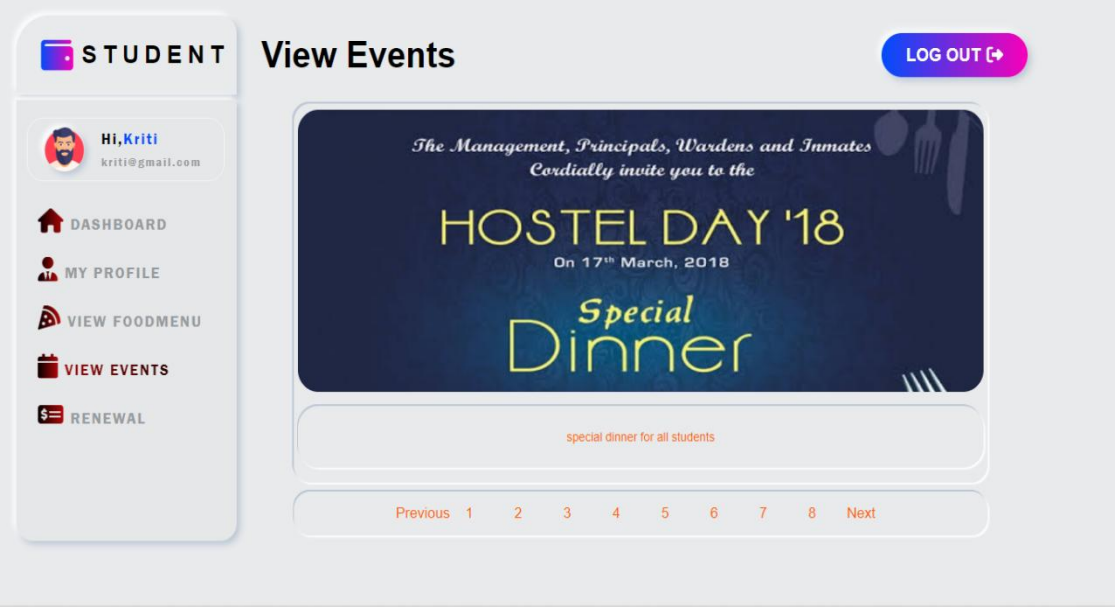


Figure 7.22 View event

TESTING

8. TESTING

8.1 Introduction

Testing is the major quality control measures and during the software development it is used to detect errors that could have occurred during any of the phase like requirement analysis, design, coding. The goal of the testing is to uncover errors in the program.

8.2 Levels of Testing

Testing is done in different levels which includes the following.

- Unit Testing
- Integration Testing
- System testing
- Acceptance testing

  - **Unit Testing**

    In Unit testing each module gets tested during the coding phase itself. The purpose is to exercise the different parts of the module code to detect the coding errors.

  - **Integration Testing**

    After new testing the modules are gradually integrated into sub systems. It is performed to detect design errors by focusing on testing the interconnection between modules.

  - **System Testing**

    System is tested against the system requirement if all the requirements are met and if the system performs as specified by the requirement.

  - **Acceptance Testing**

    It is performed to demonstrate to the client on real life data of the client, the operation of the system.

8.3 Test Case

It is the input that tests the genuineness of the program and successful execution of the test case revels. that there are no errors in the program that are under testing. it is a set of conditions or variables under which tester will determine whether an application or software is working currently

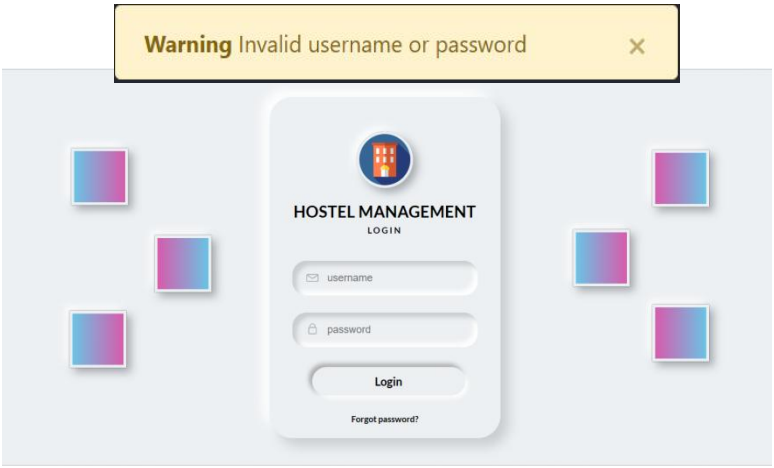
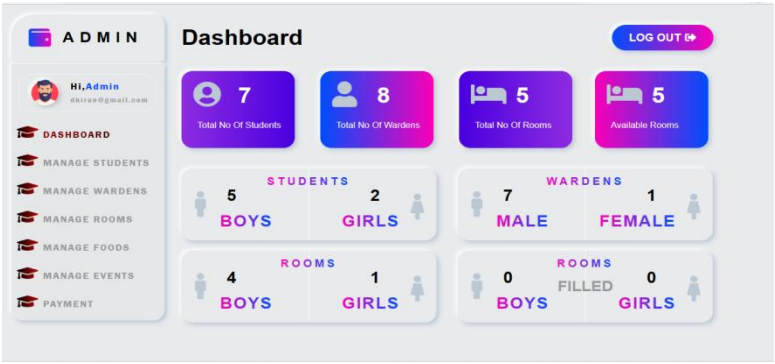
Test case-ID	01
Test Case Title	Login
Purpose of testing	Testing the login
Test data	User Id / Email Id and Password
Steps	Step:1 IF User Id / Email Id and Password is valid THEN Step:2 REDIRECT to Dashboard page Step:3 ELSE DISPLAY Login Unsuccessful.
Expected Output	Invalid output:  Valid output: 

Table 8.1 Login

Test case-ID	02
Test Case Title	Register
Purpose of testing	To register student to the hostel.
Test data	Name, parent name, gender, register number, email, checkout date, mobile number,blood group, caste, college name,course duration,address aadhar number, cardno, cardholder name, expiry month and year, cvv, roomno.
Steps	Step:1 IF provided data are valid THEN Step:2 DISPLAY Registration,payment done. login details are sent to the email Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output: 

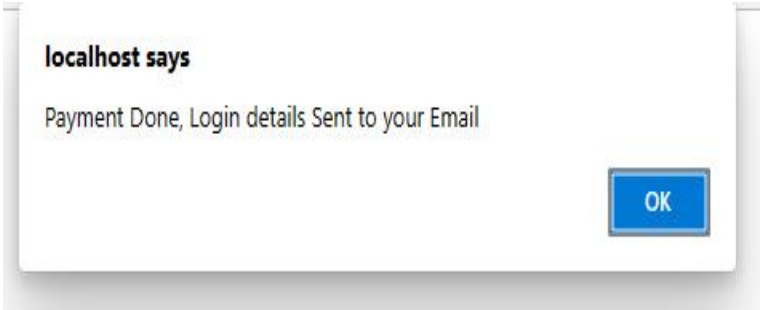
	 Valid output: 

Table 8.2 Register

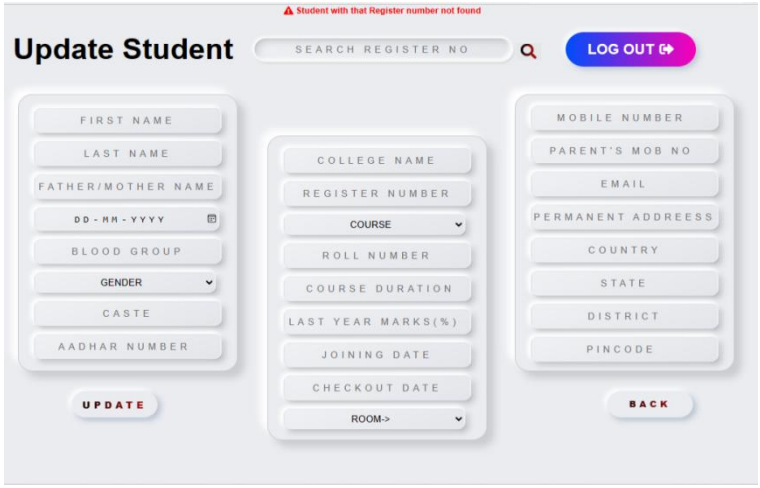
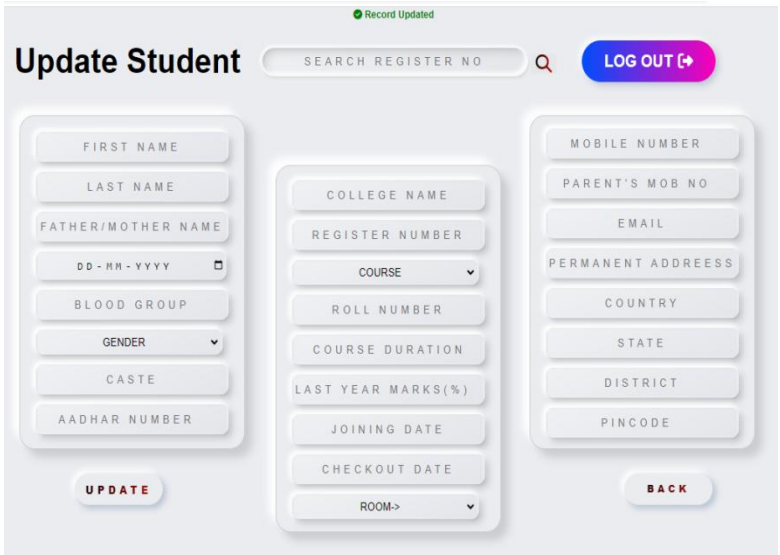
Test case-ID	03
Test Case Title	Update student
Purpose of testing	To Update the student details.
Test data	Name, father name, gender, register number, mobile number,blood group, caste, college name,course duration,address, aadhar number, room no.
Steps	Step:1 IF student details are valid THEN Step:2 DISPLAY Record updated successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.3 Update student

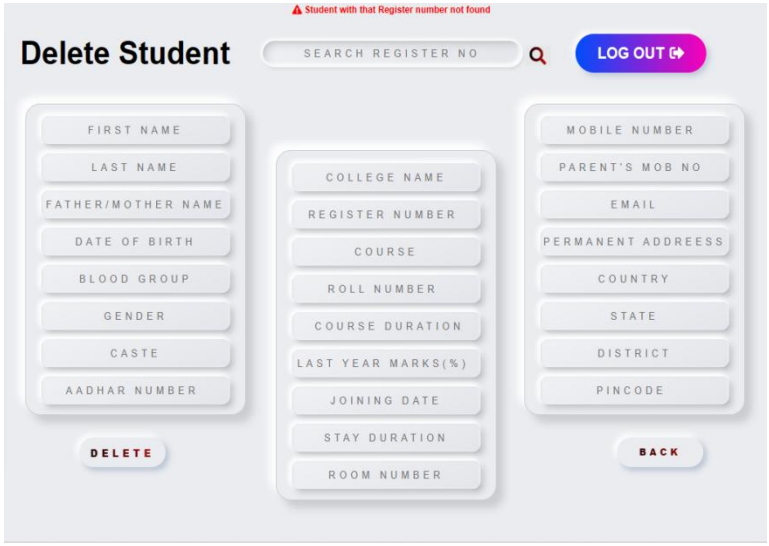
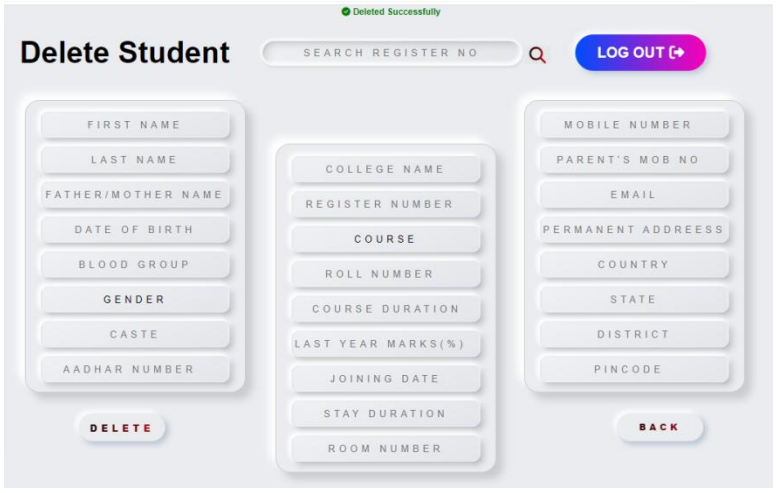
Test case-ID	04
Test Case Title	Delete student
Purpose of testing	To update the student as old student.
Test data	Register number.
Steps	Step:1 IF Register Number is valid THEN Step:2 DISPLAY Record deleted successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.4 Delete student

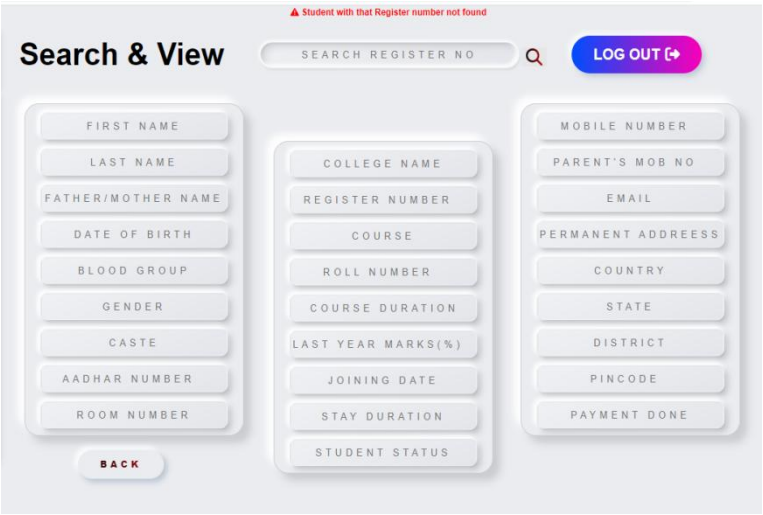
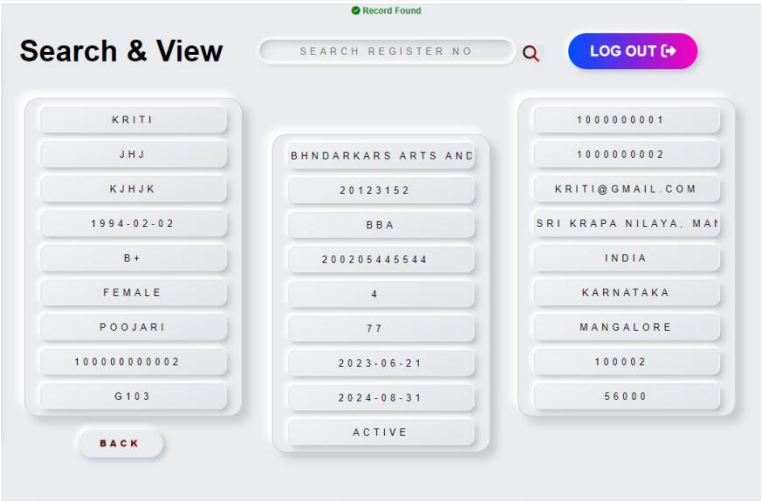
Test case-ID	05
Test Case Title	Search and view student
Purpose of testing	To view a particular student
Test data	Register number.
Steps	Step:1 IF Register Number is valid THEN Step:2 DISPLAY Record found message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.5 Search and view student

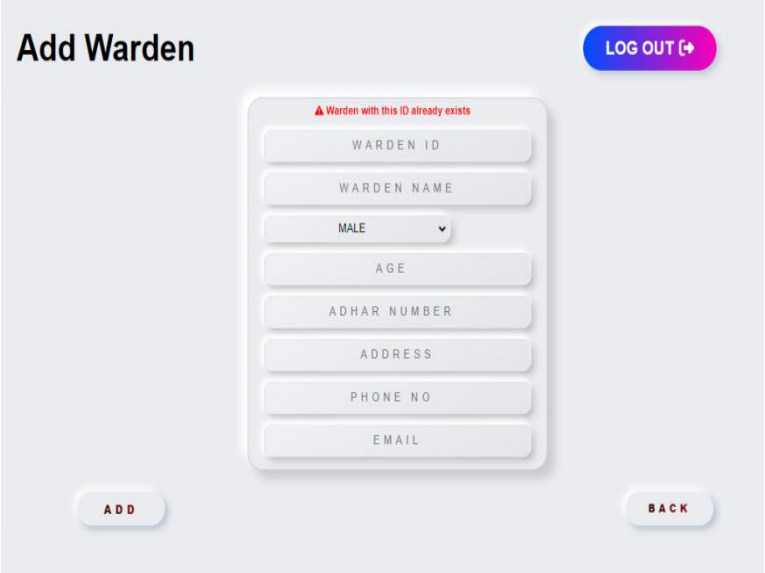
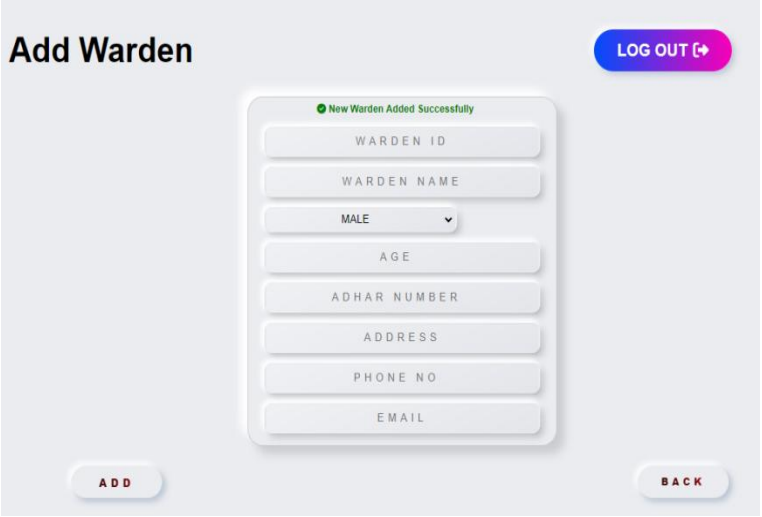
Test case-ID	06
Test Case Title	Add warden
Purpose of testing	To add the warden to the hostel
Test data	Name, warden id, warden email, gender, aadhar number, address, phone number.
Steps	Step:1 IF data are valid THEN Step:2 DISPLAY warden added successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.6 Add warden

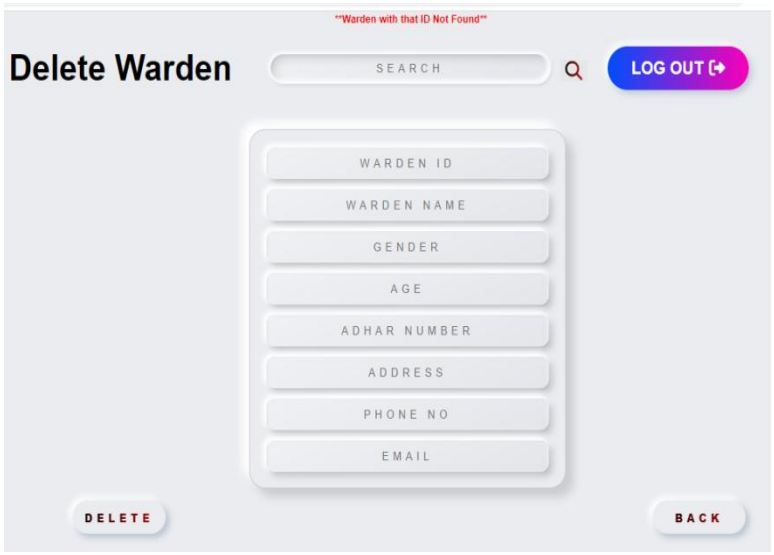
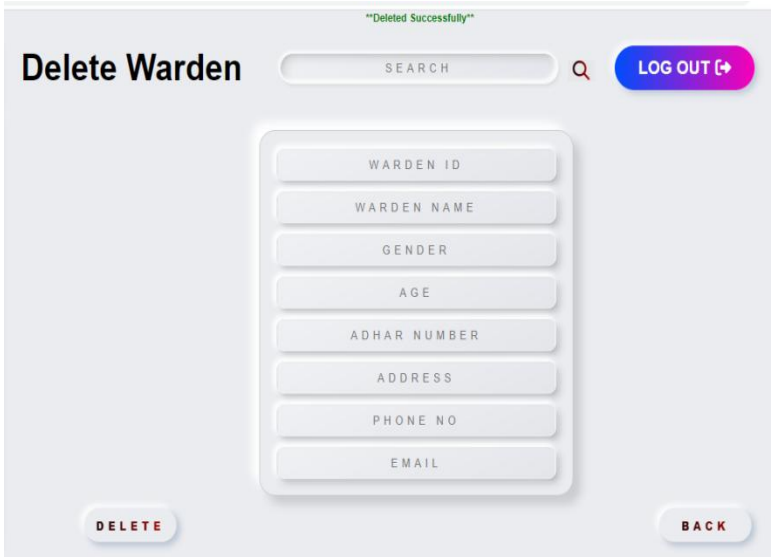
Test case-ID	07
Test Case Title	Delete warden
Purpose of testing	To update the warden as old warden.
Test data	Warden id
Steps	Step:1 IF warden id is valid THEN Step:2 DISPLAY warden deleted successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.7 Delete warden

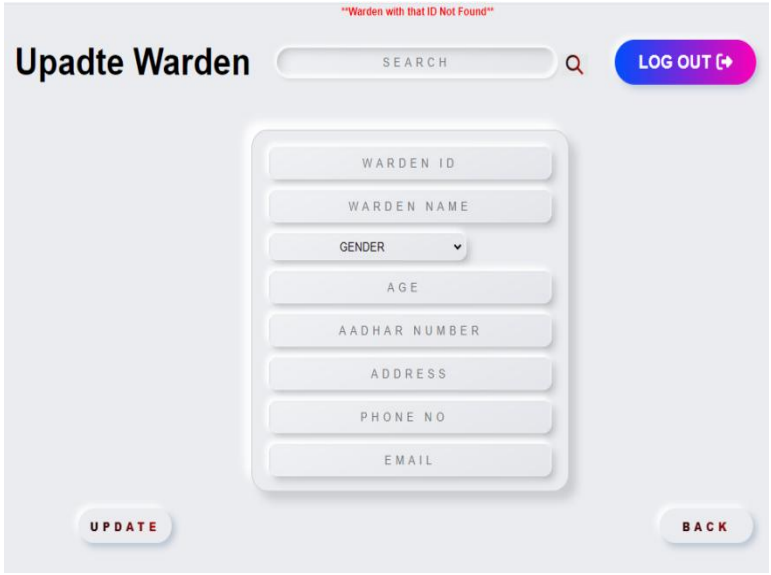
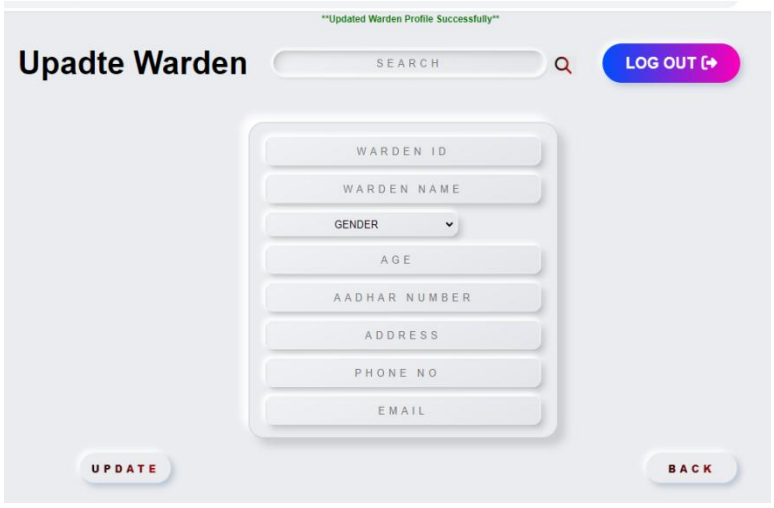
Test case-ID	08
Test Case Title	Update warden
Purpose of testing	To update the warden details.
Test data	Name, warden email, gender, aadhar number, address, phone number
Steps	Step:1 IF warden details are valid THEN Step:2 DISPLAY warden profile updated successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.8 Update Warden

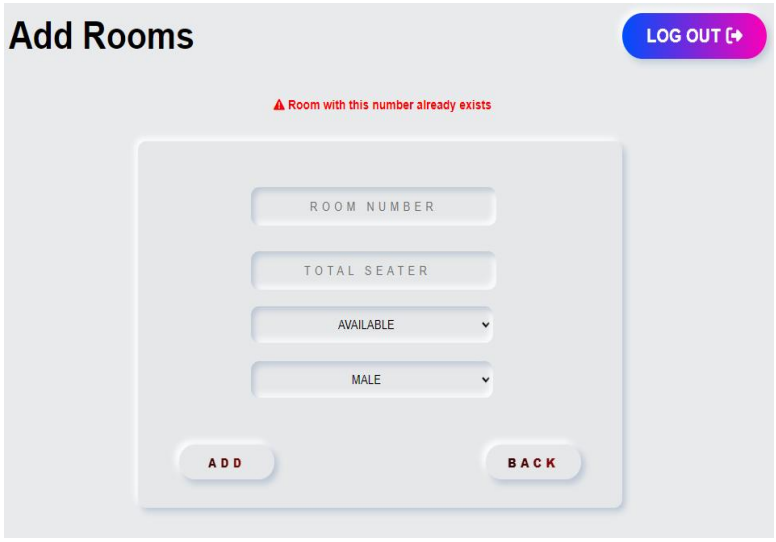
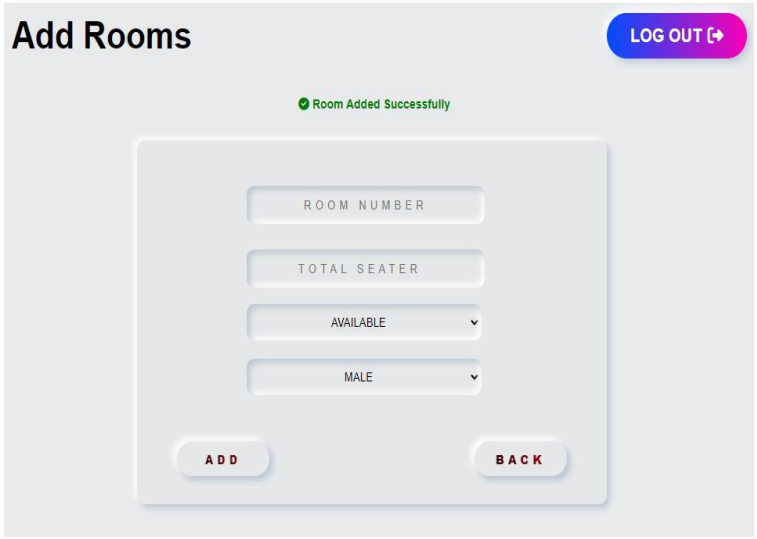
Test case-ID	09
Test Case Title	Add room
Purpose of testing	To add the room to the hostel
Test data	Room Id, seater, room for, status.
Steps	Step:1 IF data are valid THEN Step:2 DISPLAY room added successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.9 Add room

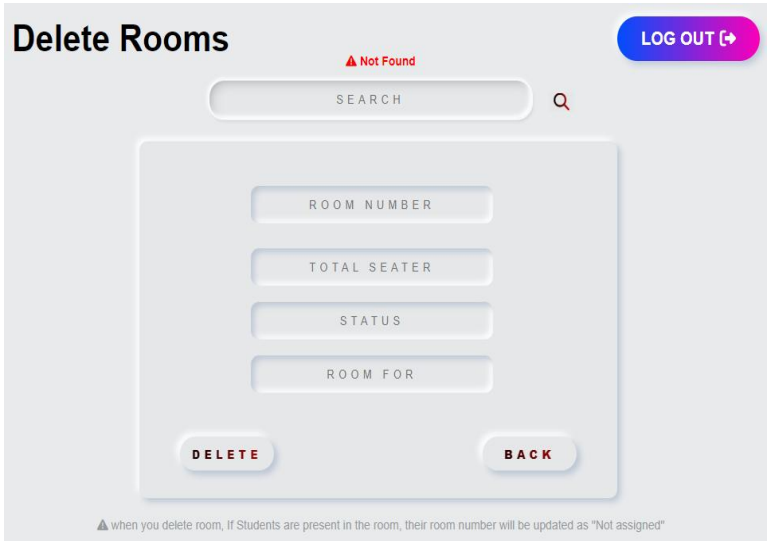
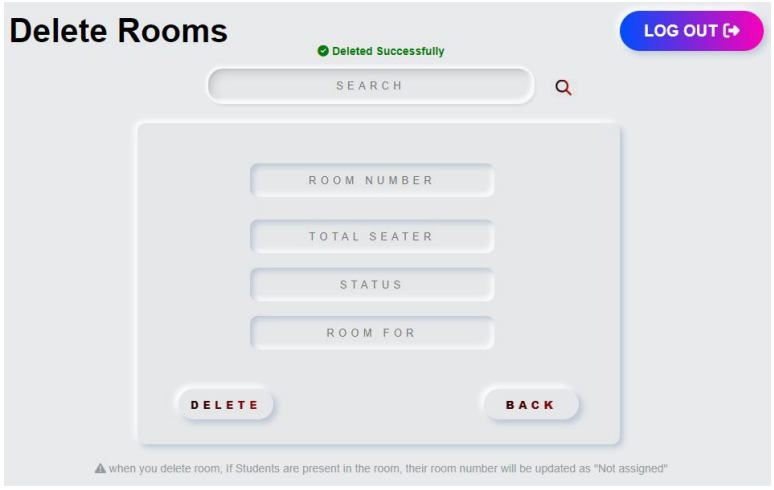
Test case-ID	10
Test Case Title	Delete room
Purpose of testing	To delete the room.
Test data	Room Id.
Steps	Step:1 IF Room id is valid THEN Step:2 DISPLAY room deleted successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.10 Delete room

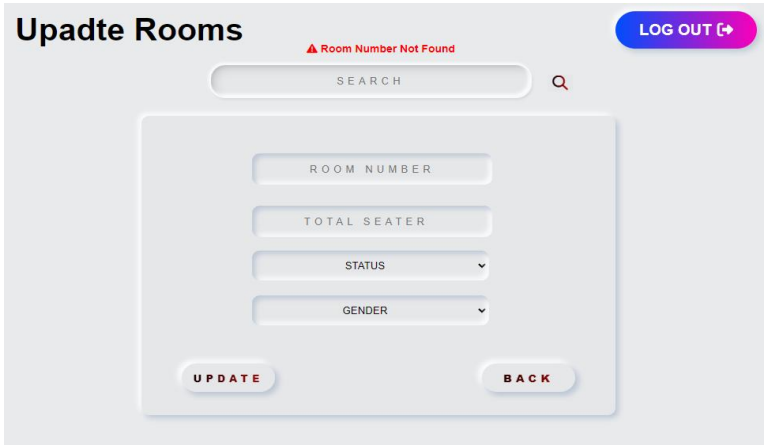
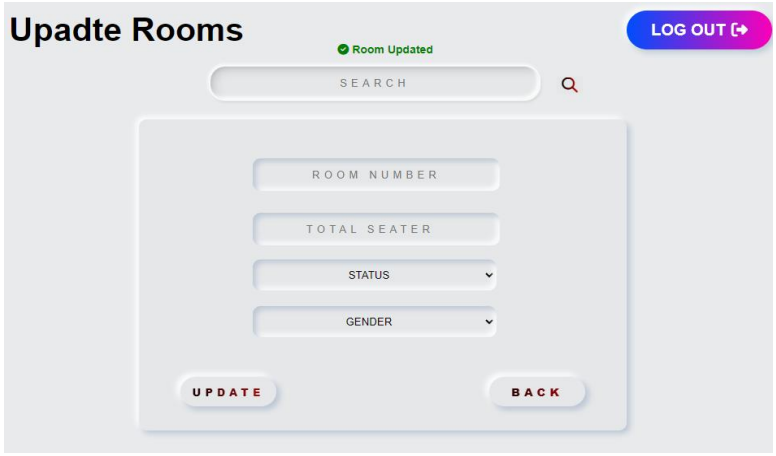
Test case-ID	11
Test Case Title	Update room
Purpose of testing	To update the room.
Test data	Room Id.
Steps	Step:1 IF data are valid THEN Step:2 DISPLAY room updated successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.11 Update room

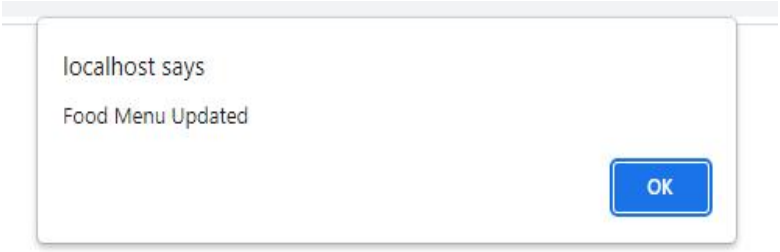
Test case-ID	12
Test Case Title	Update food menu
Purpose of testing	To update the food menu.
Test data	Food names.
Steps	Step:1 IF data are valid THEN Step:2 DISPLAY food menu updated successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.12 Update food menu

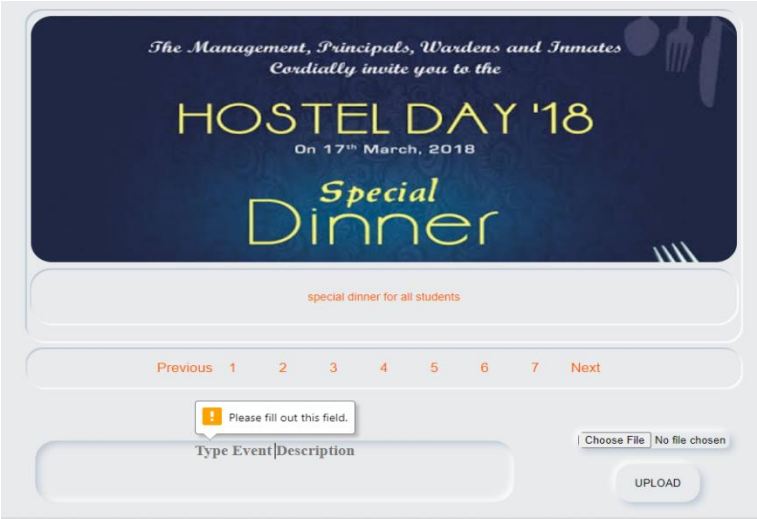
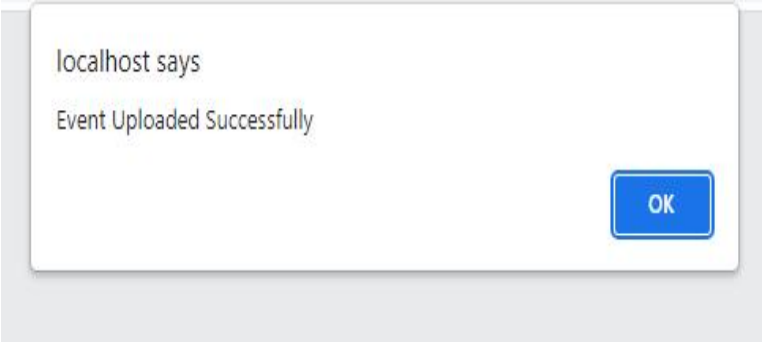
Test case-ID	13
Test Case Title	Upload event
Purpose of testing	To upload the event.
Test data	Event name and description, image.
Steps	Step:1 IF data are valid THEN Step:2 DISPLAY event uploaded successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.13 Upload event

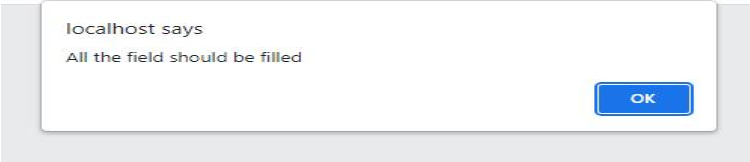
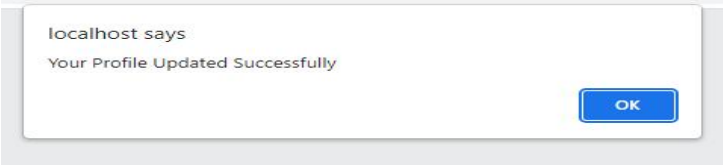
Test case-ID	14
Test Case Title	Update my profile
Purpose of testing	To update the students profile.
Test data	College name, Roll no, Course, course duration, previous year mark, address,password.
Steps	Step:1 IF data are valid THEN Step:2 DISPLAY Profile updated successfully message. Step:3 ELSE DISPLAY warning message with description.
Expected Output	Invalid output:  Valid output: 

Table 8.14 Update my profile

CONCLUSION

In conclusion, this project was successfully implemented using PHP.It capable of managing hostel records such as Managing students,Managing wardens, Managing rooms, Managing events, Managing food and payments.

This software is efficient maintaining student, room and wardens details and can easily perform activities operation on student, room and wardens records and also works to handles the information of room available in hostel.This software also reduces the workload of hostel.

Moreover, this project helped for us to understand Software Development life Cycle(SDLC.),Time bound work, team spirit and preparing project document, project testing, GUI designing and presentation.

In addition to that, we learned PHP, HTML, CSS, JavaScript, MYSQL.

Finally concluded that we tried to fulfill the objectives of project work and goal of our project **"HOSTEL MANAGEMENT SYSTEM"**.

NAVEENCHANDRA	201231522201
RADHESHA SHERUGAR	201231522214
KIRANA	201231522189

LIMITATIONS

- Student need internet connection while registering in order to get password to login.
- A room can only have a maximum of 4 seats.
- Student cannot quit before the checkout date.
- If student quit before checkout date hostel fees cannot be refundable.

FUTURE SCOPE

- Hostel management system can become more automated and integrated with other system.
- Security is crucial aspect of hostel management system. Future system can incorporate advanced security features like biometric authentication and facial recognition.
- Improving communication and collaboration between hostel administrator, student, parent/guardians. This is achieved through instant messaging, notification system and centralized communication platforms.

ABBREVIATIONS AND ACRONYMS

- SRS - Software Requirement Specification.
- DFD - Data Flow Diagram.
- CFD - Context Flow Diagram.
- RAM - Random Access Memory.
- ROM - Read Only Memory.
- GUI - Graphical User Interface.
- HTML - Hypertext Mark-up Language.
- CSS - Cascading Style Sheet.
- PHP - Hypertext Pre-Processor.
- XAMPP - Extended Apache Mariadb Perl.
- SQL - Structured Query Language.
- MYSQL - My Structured Query Language.
- DBMS - Database Management System.
- ER - Entity Relationship.
- CPU - Central Processing Unit.
- SDLC - Software Development Life Cycle.
- DML - Data Manipulation Language.
- DDL - Data Definition Language.
- DCL - Data Control Language.

BIBLIOGRAPHY

Referenced books

- Integrated approach to software engineering - Pankaj Jatole.
- HTML and CSS-3 made simple -Ivan Bayross, BPB publications.
- PHP and MySQL Web development - Luke Welling, Laura Thomson

Referenced sites

- www.geeksforgeeks.com
- www.W3school.com
- www.github.com
- www.tutorialspoint.com