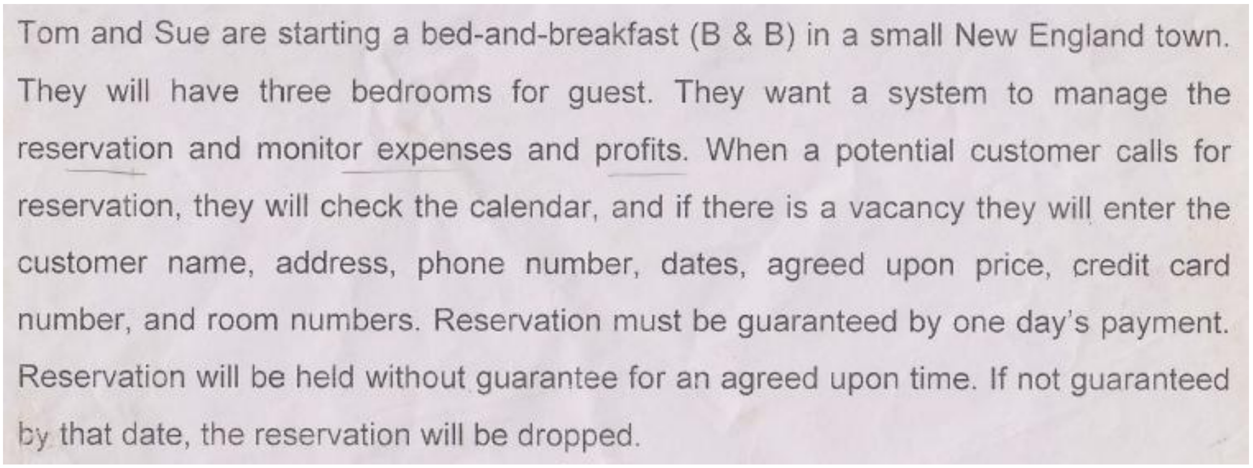
Case Study - 1


Tom and Sue are starting a bed-and-breakfast (B & B) in a small New England town. They will have three bedrooms for guest. They want a system to manage the reservation and monitor expenses and profits. When a potential customer calls for reservation, they will check the calendar, and if there is a vacancy they will enter the customer name, address, phone number, dates, agreed upon price, credit card number, and room numbers. Reservation must be guaranteed by one day's payment. Reservation will be held without guarantee for an agreed upon time. If not guaranteed by that date, the reservation will be dropped.

Customer Functional Requirement

1. **Reservation Management**
   - **3.1.1: Customers shall inquire about room availability.**
     - **Input**: Desired check-in and check-out dates.
     - **Processing**: Query the reservation system's calendar for room availability.
     - **Output**: Confirmation of available or unavailable rooms for the given dates.
     - **Precondition**: The system must have up-to-date room availability data.
     - **Postcondition**: Room availability is provided to the customer.
     - **Exception Scenario**: If the system fails to retrieve availability data, an error message is shown.
   - **3.1.2: Customers shall provide reservation details.**
     - **Input**: Name, address, phone number, check-in and check-out dates, agreed price, credit card details, and room number.
     - **Processing**: Validate the input and temporarily store the reservation data in the system.
     - **Output**: Confirmation of reservation submission with a unique reservation ID.

    - **Precondition**: Room availability is confirmed.
    - **Postcondition**: The reservation is stored and awaiting guarantee.
    - **Exception Scenario**: Invalid or missing input results in an error message.
  - **3.1.3: Customers shall guarantee their reservation with payment.**
    - **Input**: Payment of one day's agreed price.
    - **Processing**: Process the payment and associate it with the reservation ID.
    - **Output**: Confirmation of reservation guarantee.
    - **Precondition**: Reservation details are present in the system.
    - **Postcondition**: Reservation is confirmed and guaranteed.
    - **Exception Scenario**: Payment failure results in the reservation remaining unguaranteed.

Staff Functional Requirement

1. **Reservation Management**
   - **3.2.1: Staff shall check the calendar for room availability.**
     - **Input**: Desired dates for availability query.
     - **Processing**: Search the system's calendar to verify if rooms are vacant.
     - **Output**: List of available rooms or a notification of unavailability.
     - **Precondition**: The calendar must have updated room data.
     - **Postcondition**: Room availability is displayed for the desired dates.
     - **Exception Scenario**: Calendar data is outdated or inaccessible.
   - **3.2.2: Staff shall add new reservations to the system.**
     - **Input**: Customer name, address, phone number, check-in and check-out dates, agreed price, credit card information, and room number.

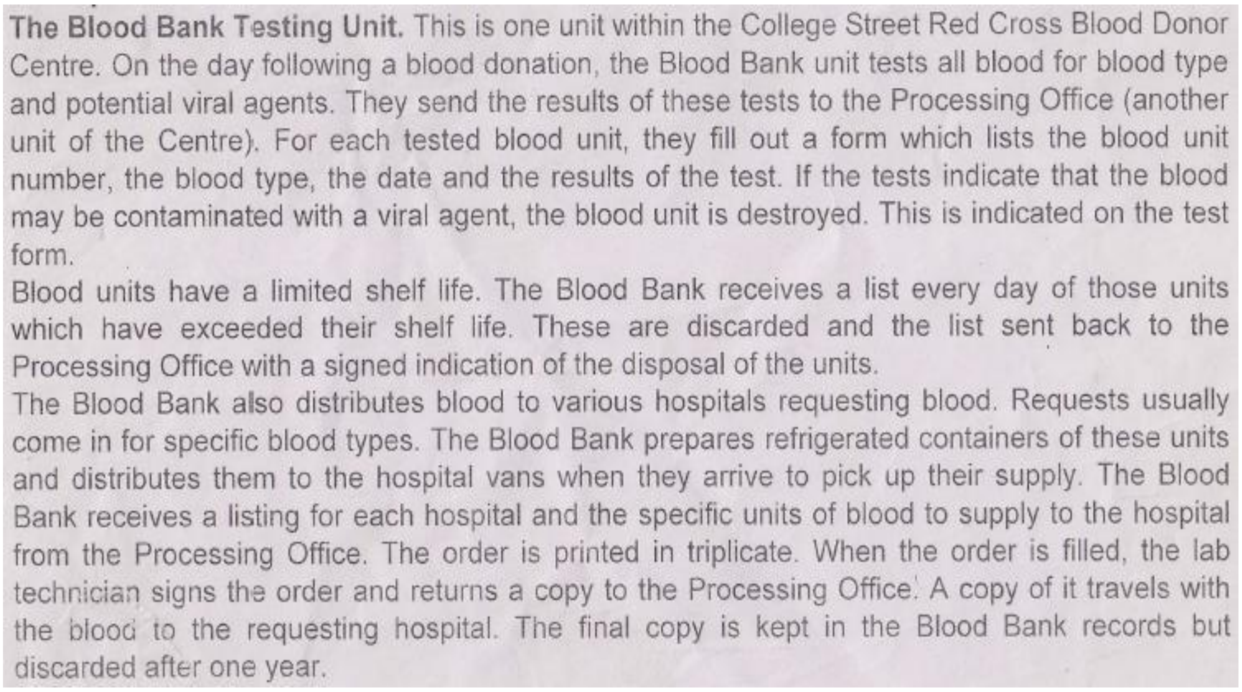
        - **Processing**: Validate input and create a new reservation record in the system.
        - **Output**: Confirmation of reservation creation with a unique reservation ID.
        - **Precondition**: Room availability is confirmed.
        - **Postcondition**: Reservation details are stored in the system.
        - **Exception Scenario**: Invalid or missing input results in an error message.

    - **3.2.3: The system shall monitor reservations for guarantee deadlines.**
        - **Input**: Reservation records with deadline details.
        - **Processing**: Periodically check if the guarantee payment has been made before the deadline.
        - **Output**: Notification for unguaranteed reservations that will be dropped.
        - **Precondition**: The reservation system is operational.
        - **Postcondition**: Unguaranteed reservations are dropped automatically after the deadline.
        - **Exception Scenario**: System failure leads to missed deadline checks.

2. **Expense and Profit Management**
    - **3.2.4: The system shall track expenses and profits.**
        - **Input**: Revenue from reservations and operational expense data.
        - **Processing**: Calculate and record profit or loss for a given period.
        - **Output**: Reports showing expense, revenue, and profit.
        - **Precondition**: Reservation and expense records are available.
        - **Postcondition**: Financial data is accessible for analysis.
        - **Exception Scenario**: Missing or incomplete data leads to inaccurate calculations.

3. **Room Management**
    - **3.2.5: Staff shall manage room availability.**
        - **Input**: Room status updates (occupied, vacant, or under maintenance).
        - **Processing**: Update the calendar and system records accordingly.
        - **Output**: Updated room availability for accurate reservation management.
        - **Precondition**: Accurate room status information is provided.
        - **Postcondition**: System reflects real-time room availability.
        - **Exception Scenario**: Incorrect updates lead to overbooking or mismanagement.

Case Study-2



The Blood Bank Testing Unit. This is one unit within the College Street Red Cross Blood Donor Centre. On the day following a blood donation, the Blood Bank unit tests all blood for blood type and potential viral agents. They send the results of these tests to the Processing Office (another unit of the Centre). For each tested blood unit, they fill out a form which lists the blood unit number, the blood type, the date and the results of the test. If the tests indicate that the blood may be contaminated with a viral agent, the blood unit is destroyed. This is indicated on the test form.

Blood units have a limited shelf life. The Blood Bank receives a list every day of those units which have exceeded their shelf life. These are discarded and the list sent back to the Processing Office with a signed indication of the disposal of the units.

The Blood Bank also distributes blood to various hospitals requesting blood. Requests usually come in for specific blood types. The Blood Bank prepares refrigerated containers of these units and distributes them to the hospital vans when they arrive to pick up their supply. The Blood Bank receives a listing for each hospital and the specific units of blood to supply to the hospital from the Processing Office. The order is printed in triplicate. When the order is filled, the lab technician signs the order and returns a copy to the Processing Office. A copy of it travels with the blood to the requesting hospital. The final copy is kept in the Blood Bank records but discarded after one year.

Blood Bank Functional Requirements

1. Blood Testing and Disposal

- **3.1.1: The Blood Bank unit shall test donated blood for blood type and**

potential viral agents.

  - **Input: Blood units collected from donors.**
  - **Processing: Analyze blood for type and potential viral agents, then record the results.**
  - **Output: A completed test form listing blood unit number, blood type, test date, and results.**
  - **Precondition: Blood units are collected and stored for testing.**
  - **Postcondition: Test results are sent to the Processing Office for further action.**
  - **Exception Scenario: Invalid blood samples result in an error notification and retesting.**

- **3.1.2: The Blood Bank unit shall destroy contaminated blood.**

  - **Input: Blood test results indicating contamination.**
  - **Processing: Dispose of contaminated blood and record the disposal on the test form.**
  - **Output: Updated test form indicating the blood unit has been destroyed.**
  - **Precondition: Blood test results confirm contamination.**
  - **Postcondition: Contaminated blood is no longer stored in inventory.**
  - **Exception Scenario: Improper disposal protocols result in an error report.**

2. Shelf Life Management

- **3.2.1: The Blood Bank shall track and discard expired blood units.**
  - **Input: Daily list of blood units exceeding their shelf life.**
  - **Processing: Discard expired units and record their disposal.**
  - **Output: A signed list sent back to the Processing Office, confirming disposal.**
  - **Precondition: Blood unit shelf life data is updated daily.**
  - **Postcondition: Expired units are no longer part of inventory.**
  - **Exception Scenario: Missing or outdated shelf life data results in errors.**

3. Blood Distribution to Hospitals

- **3.3.1: The Blood Bank shall process hospital requests for specific blood types.**

  - **Input: Request list from hospitals specifying required blood types and quantities.**
  - **Processing: Match requests with available blood units and prepare orders.**
  - **Output: Refrigerated containers with requested blood units.**

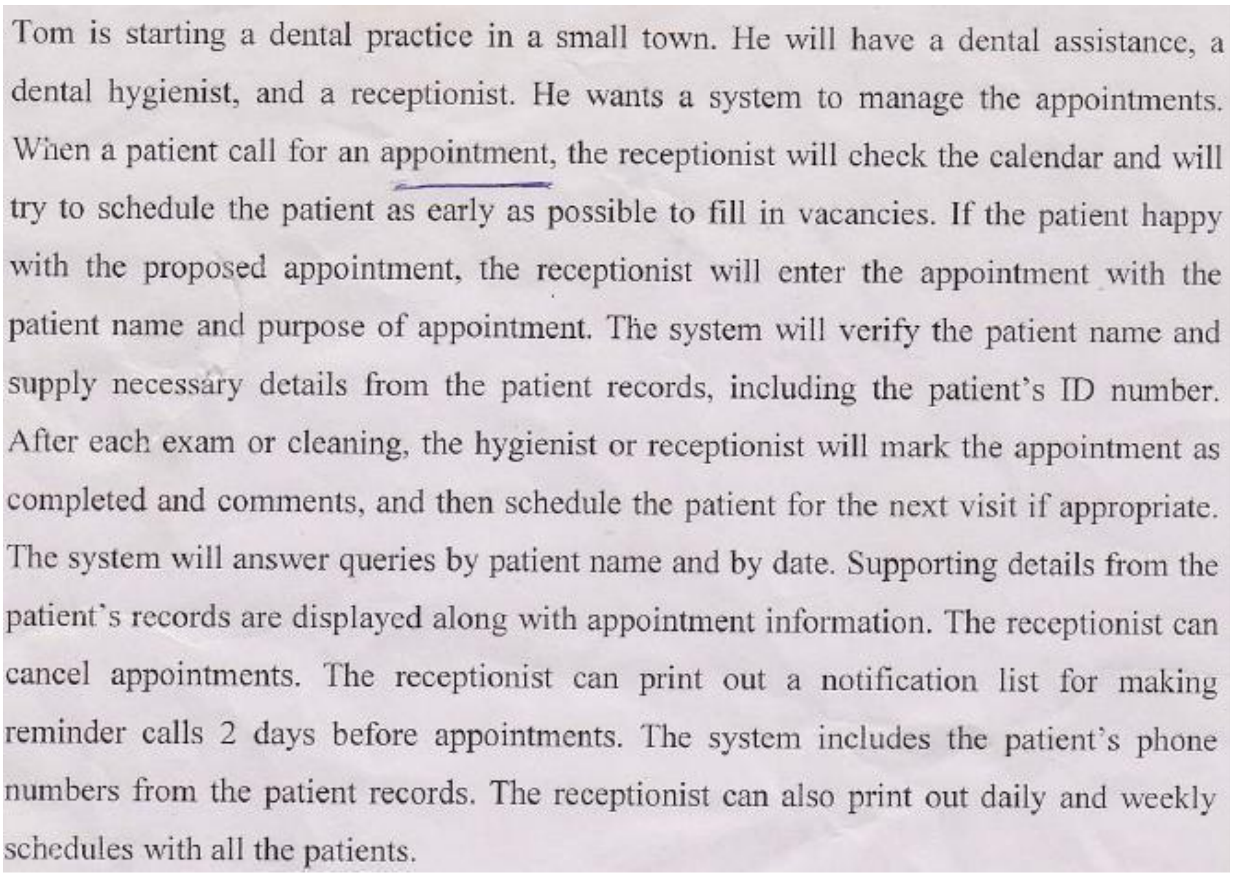
    - **Precondition: Blood units matching the request are available.**
    - **Postcondition: Prepared orders are ready for hospital pickup.**
    - **Exception Scenario: Insufficient stock for requests results in partial fulfillment or notification.**

- **3.3.2: The Blood Bank shall generate and manage blood distribution orders.**
    - **Input: Hospital request details from the Processing Office.**
    - **Processing: Print the order in triplicate, with one copy traveling with the blood, one copy returned to the Processing Office, and one stored in the Blood Bank records.**
    - **Output: Completed order forms distributed appropriately.**
    - **Precondition: Hospital requests are approved and processed.**
    - **Postcondition: Blood distribution orders are recorded and fulfilled.**
    - **Exception Scenario: Order processing errors result in delays or inaccuracies.**

- **3.3.3: The Blood Bank shall store final order records for one year.**

    - **Input: Completed blood distribution orders.**
    - **Processing: Archive order records in the Blood Bank database for one year.**
    - **Output: Records accessible for audits or future reference.**
    - **Precondition: Distribution orders are fulfilled and recorded.**
    - **Postcondition: Archived records are retrievable for one year.**
    - **Exception Scenario: Storage system failure results in data loss.**

Case Study – 3



Tom is starting a dental practice in a small town. He will have a dental assistance, a dental hygienist, and a receptionist. He wants a system to manage the appointments. When a patient call for an appointment, the receptionist will check the calendar and will try to schedule the patient as early as possible to fill in vacancies. If the patient happy with the proposed appointment, the receptionist will enter the appointment with the patient name and purpose of appointment. The system will verify the patient name and supply necessary details from the patient records, including the patient's ID number. After each exam or cleaning, the hygienist or receptionist will mark the appointment as completed and comments, and then schedule the patient for the next visit if appropriate. The system will answer queries by patient name and by date. Supporting details from the patient's records are displayed along with appointment information. The receptionist can cancel appointments. The receptionist can print out a notification list for making reminder calls 2 days before appointments. The system includes the patient's phone numbers from the patient records. The receptionist can also print out daily and weekly schedules with all the patients.

Dental Practice System Functional Requirements

1. **Appointment Management**
   - **3.1.1: The system shall allow the receptionist to schedule patient appointments as early as possible based on calendar availability.**
     - **Input**: Patient call with desired date and purpose of the appointment.
     - **Processing**: Search for the earliest available time slot and propose it to the patient.
     - **Output**: Confirmation of the scheduled appointment, including date, time, and purpose.
     - **Precondition**: Calendar must be updated with current appointments and available slots.
     - **Postcondition**: Appointment is added to the schedule.
     - **Exception Scenario**: No available slots; the system informs the patient and suggests alternate dates.'

- **3.1.2: The system shall verify and retrieve patient records for appointment scheduling.**
  - **Input**: Patient name and/or ID number.
  - **Processing**: Validate input and fetch patient details from the database.
  - **Output**: Patient details, including ID and history, displayed to the receptionist.
  - **Precondition**: Patient records exist in the system.
  - **Postcondition**: Patient details are linked to the scheduled appointment.
  - **Exception Scenario**: Invalid or missing patient records prompt an error message.
- **3.1.3: The receptionist shall cancel appointments if requested.**
  - **Input**: Appointment ID or patient name.
  - **Processing**: Locate the appointment in the system and remove it.
  - **Output**: Confirmation of successful cancellation.
  - **Precondition**: Appointment exists in the system.
  - **Postcondition**: Appointment is removed from the schedule.
  - **Exception Scenario**: Invalid appointment ID results in an error notification.

2. **Post-Appointment Management**
   - **3.2.1: The hygienist or receptionist shall mark appointments as completed and add comments.**
     - **Input**: Appointment ID, status update, and optional comments.
     - **Processing**: Update appointment status to "completed" and save comments.
     - **Output**: Confirmation of status update.
     - **Precondition**: Appointment exists in the system.
     - **Postcondition**: Appointment marked as completed with comments added to the record.
     - **Exception Scenario**: Missing appointment ID results in a failure to update status.
   - **3.2.2: The system shall allow scheduling for the next visit if necessary.**

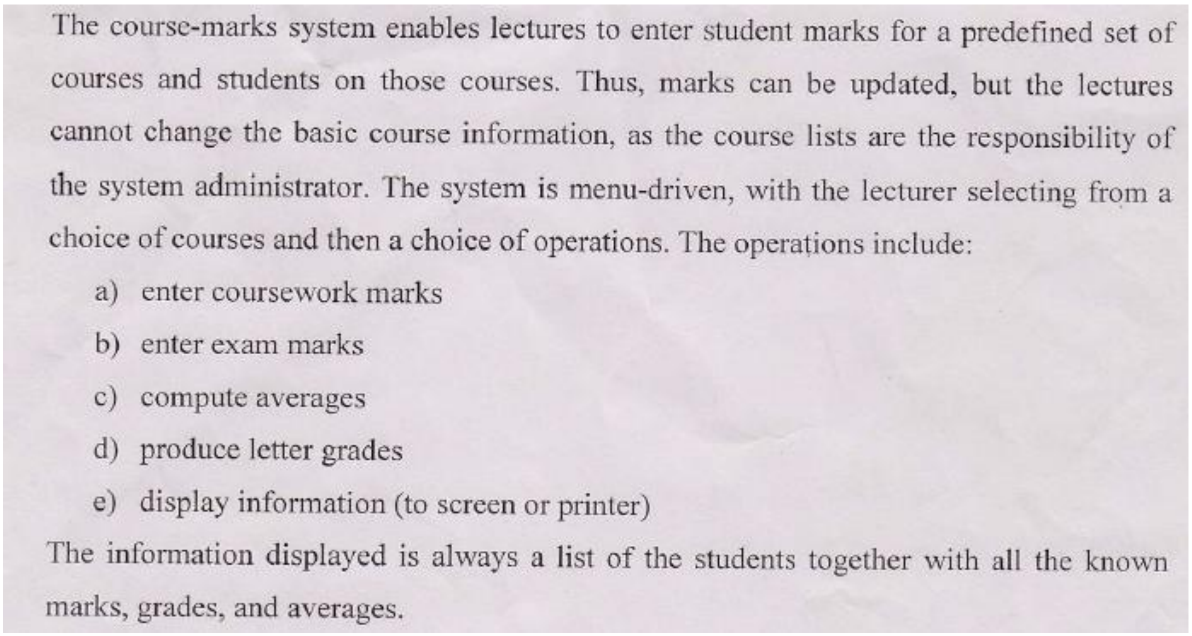
        - **Input**: Patient name or ID, and suggested date for the next visit.
        - **Processing**: Check availability and schedule the next appointment.
        - **Output**: Confirmation of the next visit scheduled.
        - **Precondition**: Patient's previous appointment is marked as completed.
        - **Postcondition**: Next appointment is added to the schedule.
        - **Exception Scenario**: No suitable date results in an error notification to the receptionist.

3. **Patient Records and Queries**
    - **3.3.1: The system shall allow queries by patient name or date to display patient records and appointments.**
        - **Input**: Query parameters such as patient name or date.
        - **Processing**: Search the database for matching records.
        - **Output**: Display of patient records, including appointment information.
        - **Precondition**: Database contains patient and appointment records.
        - **Postcondition**: Relevant records are retrieved and displayed.
        - **Exception Scenario**: No matching records prompt an error notification.

4. **Notification and Reminder Management**
    - **3.4.1: The receptionist shall generate a notification list for reminder calls 2 days before appointments.**
        - **Input**: Date for upcoming appointments.
        - **Processing**: Fetch patient contact information for the specified date.
        - **Output**: A list of patient phone numbers for reminder calls.
        - **Precondition**: Appointments are scheduled for the specified date.
        - **Postcondition**: Notification list is generated and accessible.
        - **Exception Scenario**: No appointments for the specified date result in an empty list.
    - **3.4.2: The receptionist shall print daily and weekly schedules.**
        - **Input**: Selected date range for the schedule.
        - **Processing**: Retrieve appointments for the specified date range.

- **Output**: Printed schedule with patient details and appointment times.
- **Precondition**: Appointments exist for the selected date range.
- **Postcondition**: Schedule is printed successfully.
- **Exception Scenario**: No appointments result in a notification of an empty schedule.

Case Study-4



The course-marks system enables lectures to enter student marks for a predefined set of courses and students on those courses. Thus, marks can be updated, but the lectures cannot change the basic course information, as the course lists are the responsibility of the system administrator. The system is menu-driven, with the lecturer selecting from a choice of courses and then a choice of operations. The operations include:

a) enter coursework marks
b) enter exam marks
c) compute averages
d) produce letter grades
e) display information (to screen or printer)

The information displayed is always a list of the students together with all the known marks, grades, and averages.

Coursework Marks Management

3.1.1: The system shall allow lecturers to enter coursework marks for students in predefined courses.

- **Input: Course name, student ID, and coursework marks.**
- **Processing: Validate the course and student ID, then store the coursework marks in the system.**
- **Output: Confirmation of successful entry or error message if the operation fails.**
- **Precondition: The course and student information must exist in the system.**
- **Postcondition: Coursework marks are updated for the student in the specified course.**
- **Exception Scenario: Invalid course or student ID results in an error notification.**

Exam Marks Management

3.2.1: The system shall allow lecturers to enter exam marks for students in predefined courses.

- **Input: Course name, student ID, and exam marks.**
- **Processing: Validate the course and student ID, then store the exam marks in the system.**
- **Output: Confirmation of successful entry or error message if the operation fails.**
- **Precondition: The course and student information must exist in the system.**
- **Postcondition: Exam marks are updated for the student in the specified course.**
- **Exception Scenario: Invalid course or student ID results in an error notification.**

Averages Calculation

3.3.1: The system shall compute the average marks for students in a course.

- **Input: Course name.**
- **Processing: Retrieve coursework and exam marks for all students in the course, then compute the average.**
- **Output: List of students with their average marks.**
- **Precondition: Coursework and exam marks must exist for the students in the course.**
- **Postcondition: Averages are displayed or printed.**
- **Exception Scenario: Missing marks result in an error notification.**

Grade Production

3.4.1: The system shall produce letter grades for students based on their average marks.

- **Input: Course name.**
- **Processing: Retrieve average marks, apply grading criteria, and assign letter grades.**
- **Output: List of students with their letter grades.**
- **Precondition: Average marks must be computed for students in the course.**
- **Postcondition: Letter grades are displayed or printed.**
- **Exception Scenario: Missing average marks result in an error notification.**

Information Display

3.5.1: The system shall display information about students, including marks, grades, and averages.

- **Input: Course name.**
- **Processing: Retrieve all known details about students in the course.**

- **Output: Comprehensive list of students with marks, grades, and averages displayed on the screen or sent to the printer.**
- **Precondition: Student records with marks and grades must exist in the system.**
- **Postcondition: Information is displayed or printed successfully.**
- **Exception Scenario: Missing student records or data results in an error notification.**