**OPEN EXAMINATION SYSTEM**

*Project Report*

*submitted to Keshav Mahavidyalaya, University of Delhi*

*in partial fulfillment of the requirements*

*for the Course*

*DSC 15: Software Engineering*

*B.Sc. (H) Computer Science*

|  |
| --- |
| Submitted by: Ishaan Sharma (16136) [22035570110]  and Dhruv Mirchandani(16103) [22035570103] |
|  |
|  | Supervisor: |
|  | **Dr. Anjali Thukral** |
|  |  |



2024

Department of Computer Science

Keshav Mahavidyalaya

University of Delhi

**CERTIFICATE**

This is to certify that the work embodied in this project entitled **“OPEN examination system”** has been carried out in the Department of Computer Science, Keshav Mahavidyalaya, University of Delhi. The work presented in this project is original and has not been submitted in part or full to this or any other University for any diploma or degree.

**Name: Ishaan Sharma Dhruv Mirchandani**

**Examination Rollno: 22035570110 22035570103**

**Class Rollno: 16136 16103**

Dr. Anjali Thukral

(Supervisor)

**Content**

**Problem Statement 4**

**Process Model 4**

**1. Software Requirement Specification 5**

**1.1 Overall Description 5**

**1.1.1 Product Functions 5**

**1.1.2 User Characteristics 5**

**1.1.3 General Constraints 5**

**1.1.4 Assumptions and Dependencies 5**

**1.2 External Interface Requirements 6**

**1.2.1 User Interfaces 6**

**1.2.2 Hardware Interfaces 6**

**1.2.3 Software Interfaces 6**

**1.3 Functional Requirements 6**

**1.3.1 FR 1 6**

**1.3.2 FR 2 6**

**1.3.3 FR n 6-7**

**1.4 Performance Requirement 7**

**1.5 Design Constraints 7**

**1.6 Data Flow Diagram 8**

**1.7 Data Dictionary 9**

**2. Estimations 10**

**2.1 Function Points 10**

**2.2 Efforts 11**

**3. Scheduling (Gantt Chart) 11**

**4. Risk Management 12-13**

**5. Design 14**

**5.1 System Design 14-21**

**5.2 Data Design 21-23**

**6. Coding 24-25**

**7. Testing 26-29**

**8. References 30**

**Problem Statement**

The Open Examination System is designed to facilitate online examinations, allowing students to take exams from remote locations while ensuring integrity, fairness, and efficiency. The system should support creating, scheduling, conducting, and evaluating online exams for multiple subjects. The platform will include interfaces for both examiners (to create and manage exams) and students (to take exams). The system will also maintain records, track performance, and provide analytics to educators.

# 

# Process Model

The Open Examination System will follow an **incremental model**. The system will be developed in a series of small, manageable modules, with each increment delivering a fully functional set of features. This allows for thorough testing and feedback at every stage, ensuring continuous improvement.

# Software Requirement Specification

## Overall Description

### Use Case Diagram

### User Characteristics

 **Students**: Aged 18-25, with basic computer knowledge, able to navigate web-based systems.

 **Examiners/Faculty**: Educators who create and evaluate exams, proficient in subject knowledge and comfortable with online tools.

 **Admin**: Technical staff who manage system settings, user accounts, and oversee the general administration of the platform.

### General Constraints

 The system should support up to 1000 concurrent users during peak times.

 The exam duration must be enforced strictly, with automatic submission when time expires.

 Exams should support various question formats (MCQ, essay, true/false, etc.).

 The system must work across multiple platforms (web, mobile).

### Assumptions and Dependencies

* Users will have reliable internet access during exams.
* The system will be deployed on cloud infrastructure to ensure scalability.
* Third-party services, such as email or SMS, will be used for sending notifications.

## 

## External Interface Requirements

### User Interfaces

 **Student Dashboard**: View upcoming exams, take exams, and review results.

 **Examiner Dashboard**: Create, schedule, and evaluate exams.

 **Admin Dashboard**: Manage users, system settings, and oversee exam sessions.

### Hardware Interfaces

 The system should be accessible via standard devices like desktops, laptops, tablets, and mobile phones.

 The system should support camera access to allow remote proctoring.

### Software Interfaces

* **Database**: Integration with a relational database for storing exam, user, and result data.
* **Email/SMS API**: Integration with third-party services for notifications.
* **Authentication**: Integration with OAuth or other SSO (Single Sign-On) providers for secure logins.

## Functional Requirements

**FR 1: Exam Information Management**

* Allow exam providers to input exam details, including:
  + Exam title, description, duration, schedule, and pass criteria.
* Store exam-related data in the **Exam Database** (D1).
* Provide examinees with access to exam details and instructions.
* Collect and store examinees’ answers for evaluation.

**FR 2: Examinees Management**

* Maintain an **Examinees Database** (D2) with details such as:
  + Examinee personal information (e.g., name, contact, ID).
  + Exam enrollment and participation records.
* Facilitate the addition, deletion, or modification of examinee information.
* Generate and provide a list of examinees for a given exam.

**FR 3: Examination Evaluation**

* Evaluate examinees’ submitted answers using pre-defined criteria.
* Process raw exam data to calculate scores and determine pass/fail status.
* Store evaluation results in the **Reports Database** (D1).

**FR 4: Examination Reports**

* Generate examination reports for exam providers and examinees, including:
  + Individual scores and pass/fail status.
  + Detailed performance breakdown (if applicable).
* Allow retrieval and presentation of reports through the system.
* Ensure secure storage of reports in the **Reports Database**.

**FR 5: Data Flow and Notifications**

* Ensure seamless data transfer between the **Exam Information Management**, **Examinees Management**, and **Examination Reports** modules.
* Notify examinees about exam schedules, results, and updates via email or system notifications.

## Performance Requirement

 The system should handle up to 1000 concurrent users without performance degradation.

 Exam results for objective-type questions should be displayed immediately upon submission.

 Subjective questions should be graded within 48 hours by the examiner.

 The system should have a response time of fewer than 2 seconds for any user action.

## Design Constraints

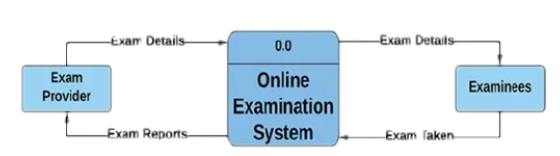
 The system must be web-based with support for responsive design.

 It should follow accessibility guidelines (e.g., WCAG 2.1) to ensure usability for all students.

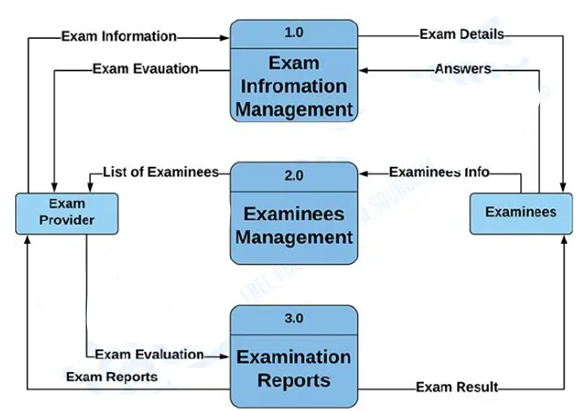
 The system must use secure connections (HTTPS) and encrypt sensitive data (user information, exam responses).

## Data Flow Diagram

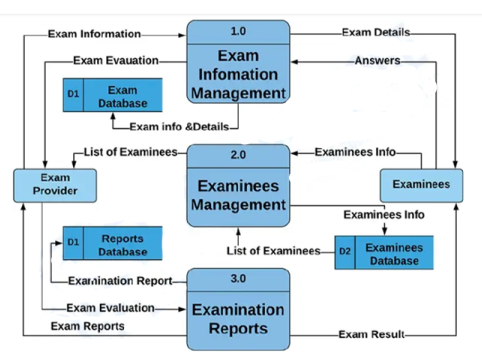
Level 0



Level 1

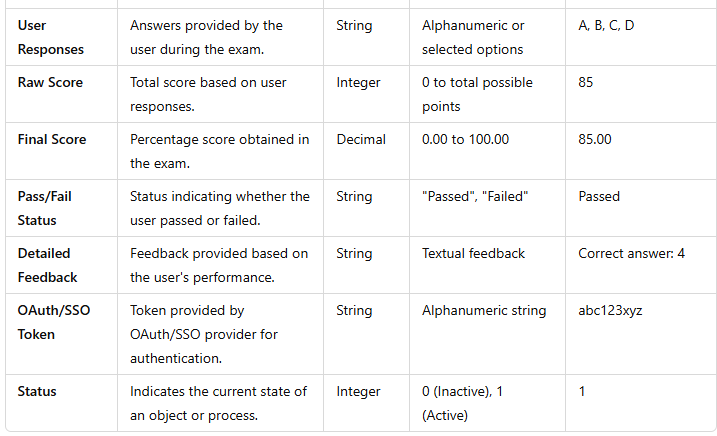


Level 2



## Data Dictionary



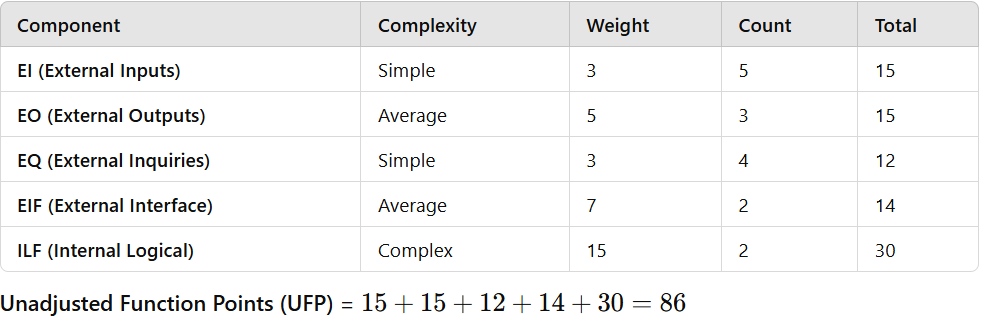


# Project Cost and Efforts

Project cost estimation and effort planning are critical for delivering the **Open Examination System** on time and within budget.

**2.1 Effort Estimation Using Function Points**

* **Function Points (FPs)** are calculated based on inputs like external inputs (EIs), external outputs (EOs), and the complexity of the system. Each component is assigned a weight based on its complexity.

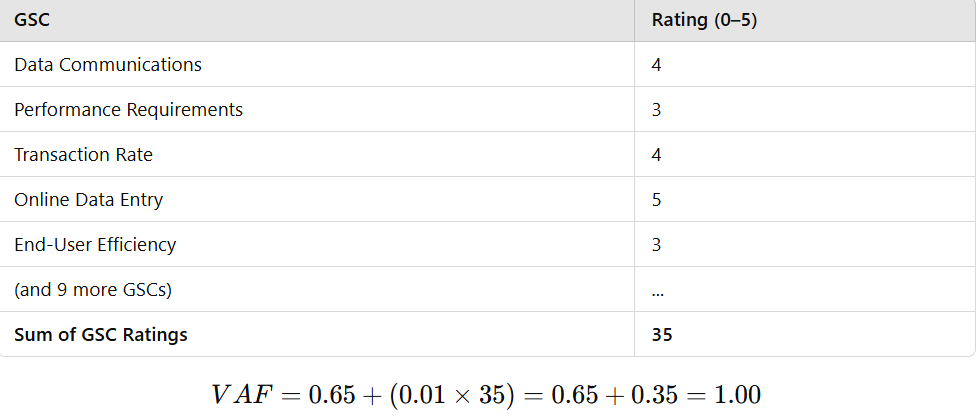


**Calculate Value Adjustment Factor (VAF)**

The VAF is determined using 14 General System Characteristics (GSCs), rated on a scale of 0 to 5, based on their impact on the project. These factors include items like data communication, system complexity, and reusability.

**VAF=0.65+(0.01×Sum of GSC ratings)**

Example GSCs and Ratings:

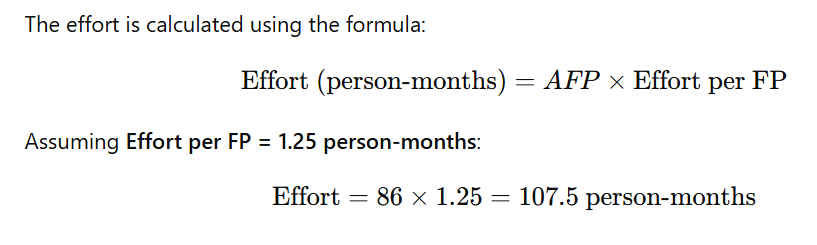
****

**Calculate Adjusted Function Points (AFP)**

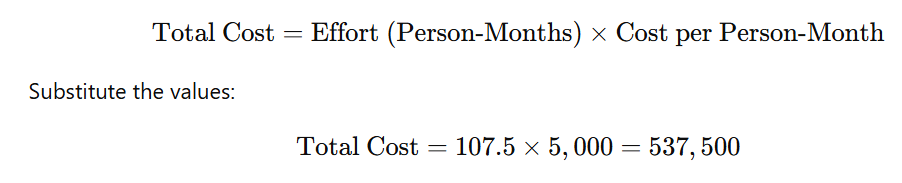
AFP = UFP × VAF

AFP = 86×1.00=86

## Effort Calculation (Using FP)



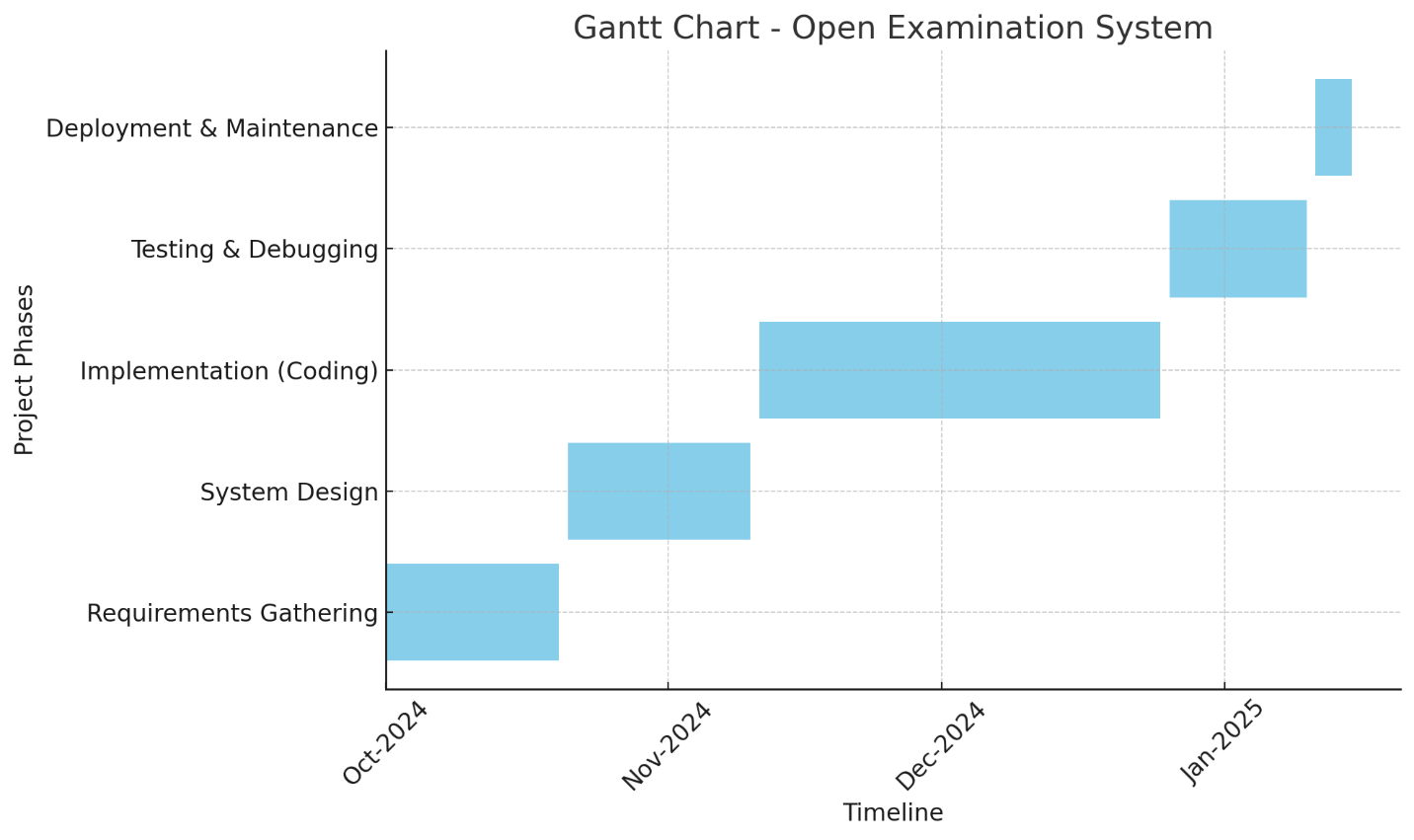
**3.Cost Estimation**



# Scheduling

**Timeline Chart (Gantt Chart)**

**Project Phases**



**4. Risk Management Plan for Open Examination System**

1. Objective

To identify, assess, and mitigate risks associated with the development and deployment of the Open Examination System, ensuring project success while maintaining system integrity and usability.

2. Risk Identification

A. Technical Risks

1. Server Downtime: Risk of server failure during peak usage.
2. Third-Party Integration Failures: Issues with APIs for email, SMS, or OAuth authentication.
3. Data Security Breach: Unauthorized access to sensitive user or exam data.
4. System Performance: Latency or crashes when handling high user traffic.

B. Operational Risks

1. Exam Scheduling Errors: Mismanagement by admins leading to incorrect exam timings.
2. Result Calculation Errors: Incorrect configurations affecting exam evaluations.
3. Proctoring Issues: Failure in camera access or cheating detection mechanisms.

C. User Risks

1. User Accessibility Issues: Difficulty navigating the platform, especially for students or examiners.
2. Connectivity Problems: Internet disruptions for remote users during examinations.

D. Project Risks

1. Scope Creep: Additional requirements leading to delays and increased costs.
2. Resource Availability: Lack of sufficient developers or testers.
3. Timeline Delays: Failure to meet deadlines due to unforeseen challenges.

3. Risk Assessment



| Risk | Impact | Likelihood | Priority |
| --- | --- | --- | --- |
| Server Downtime | High | Likely | Critical |
| Third-Party Integration Failures | Medium | Possible | Moderate |
| Data Security Breach | High | Rare | High |
| User Accessibility Issues | Medium | Likely | Moderate |
| Internet Connectivity Problems | High | Likely | Critical |
| Scope Creep | High | Possible | High |

4. Risk Mitigation Strategies

A. Technical Risks

1. Deploy the system on a scalable cloud platform with redundancy to handle peak traffic.
2. Test all third-party API integrations thoroughly during development.
3. Implement end-to-end encryption for all sensitive data and enforce strong access controls.
4. Conduct load testing to identify performance bottlenecks.

B. Operational Risks

1. Provide detailed training to admins on scheduling and exam management.
2. Automate result calculations wherever possible to minimize errors.
3. Integrate a reliable proctoring tool to manage exam integrity.

C. User Risks

1. Design a user-friendly responsive interface compatible with web and mobile.
2. Create a 24/7 support system to help users with connectivity or usability issues.

D. Project Risks

1. Use agile methodologies to accommodate scope changes without major disruptions.
2. Allocate additional buffer time in the project timeline for contingencies.
3. Maintain an updated risk register to track and manage risks dynamically.

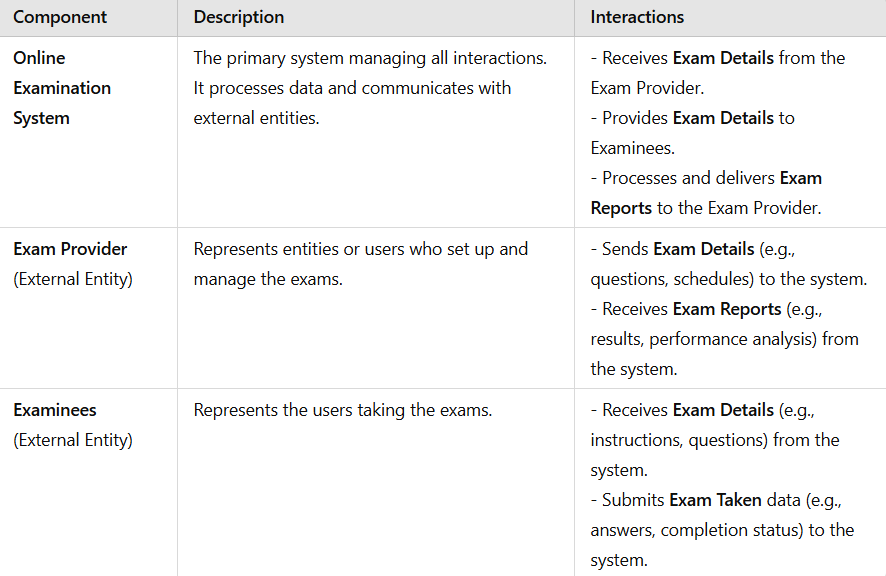
5. Risk Monitoring and Review

* Weekly Meetings: Review progress and new risks with the development team.
* Key Metrics: Track system uptime, bug reports, and user feedback.
* Risk Register: Document and regularly update identified risks, mitigation strategies, and current status.

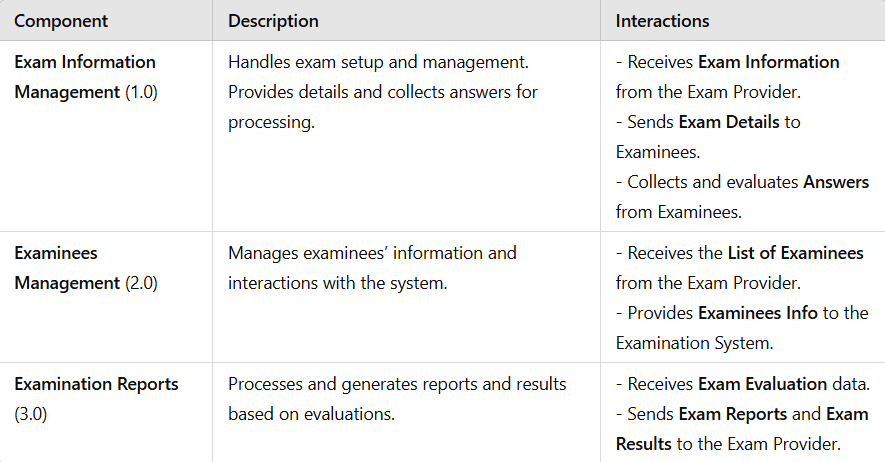
**5.1 System Design**

**Software Design (Functional) Traceable to DFD Modules**

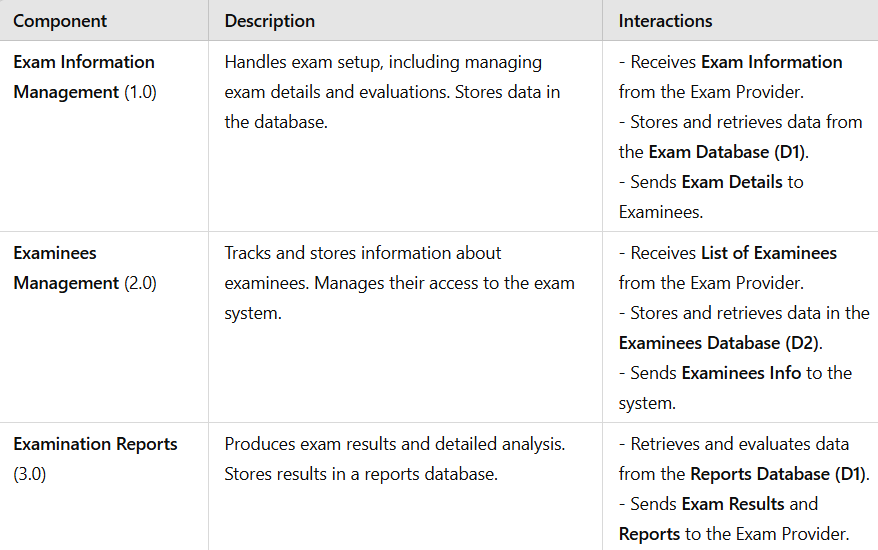
**DFD Level 0 Modules and Functional Design**

****

**DFD Level 1 Modules and Corresponding Components**

****

**DFD Level 2 Modules and Corresponding Components**

****

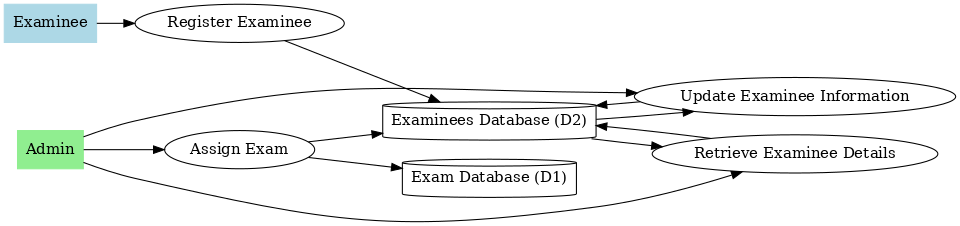
**Examinees Management Module explored**

This module serves as the bridge between examinees and the examination system. It is responsible for managing all data related to examinees, from registration to assigning exams and handling updates.

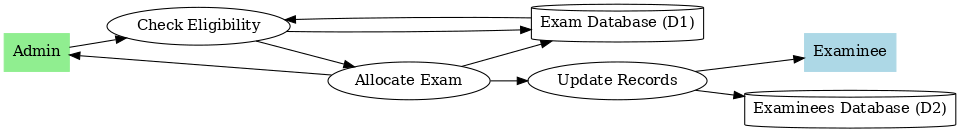
Key Functionalities

1. Register Examinee
   * Purpose: To add a new examinee to the system.
   * Steps:
     + Collect personal details (e.g., name, contact, email, ID).
     + Validate the data for completeness and uniqueness.
     + Save the information in the Examinees Database (D2).
   * Data Flow:
     + Inputs: Examinee details (provided by the user).
     + Output: Confirmation of successful registration.
     + Data Store Interaction: Creates a new record in D2.
2. Update Examinee Information
   * Purpose: To modify existing information about an examinee.
   * Steps:
     + Retrieve the examinee's current data using their ID or contact details.
     + Allow admin or examinee to update fields (e.g., address, email, phone number).
     + Save changes in the Examinees Database (D2).
   * Data Flow:
     + Inputs: Examinee ID, updated details.
     + Output: Success or error message based on operation.
     + Data Store Interaction: Updates an existing record in D2.
3. Assign Exam
   * Purpose: To allocate an exam to an eligible examinee.
   * Steps:
     + Check eligibility criteria from the Exam Database (D1) (e.g., prerequisites, available slots).
     + Assign the exam to the examinee if all criteria are met.
     + Update the records in both D1 (exam allocation) and D2 (assigned exams for the examinee).
   * Data Flow:
     + Inputs: Examinee ID, exam details.
     + Outputs: Confirmation of exam assignment or error (e.g., no slots available).
     + Data Store Interaction: Updates D1 and D2 with the assignment details.
4. Retrieve Examinee Details
   * Purpose: To fetch details about an examinee for reports or admin verification.
   * Steps:
     + Receive search criteria (e.g., ID, name).
     + Query the Examinees Database (D2) for matching records.
     + Return results in a readable format (e.g., a report or JSON output).
   * Data Flow:
     + Inputs: Search criteria (e.g., examinee ID).
     + Output: Examinee details.
     + Data Store Interaction: Reads data from D2.
5. Responsibility of Data Stores
6. Examinees Database (D2):
   * Stores all information about registered examinees.
   * Handles updates, retrievals, and exam assignment data.
7. Exam Database (D1):
   * Stores details about available exams, including eligibility criteria and assigned seats.
   * Interacts with the "Assign Exam" process.
8. Scenarios and Use Cases
9. Scenario 1: Registering a New Examinee
   * An examinee submits their details via the web portal.
   * The system checks for duplicates, validates the data, and adds the record to D2.
10. Scenario 2: Updating an Examinee's Information
    * The admin updates the email and phone number of an examinee via the admin panel.
    * The system retrieves the record, updates it, and saves changes in D2.
11. Scenario 3: Assigning an Exam
    * The admin assigns an exam to an examinee after confirming eligibility.
    * The system updates D1 to reduce available slots and records the assignment in D2.
12. Scenario 4: Generating Examinee Reports
    * The admin requests the details of examinees who registered for a specific exam.
    * The system queries D2 for examinees with matching exam assignments and generates a report.

Level 2 DFD



Level 3 DFD



**Call and Return Architecture**,

This architectural style is based on function calls and their returns, which makes it easier to translate processes and data flows from the DFD into software components.

Mapping Steps

1. Define the Call and Return Architecture

* The Call and Return Architecture is a top-down approach where:
  + Higher-level modules call lower-level modules to perform specific tasks.
  + Control flows downward as calls are made, and results/data flow upward as functions return values.

2. Identify the Main Module

* The main control module is the "Examinees Management System".
  + This module oversees the high-level processes like "Register Examinee," "Update Examinee Info," "Assign Exam," and "Retrieve Examinee Details."

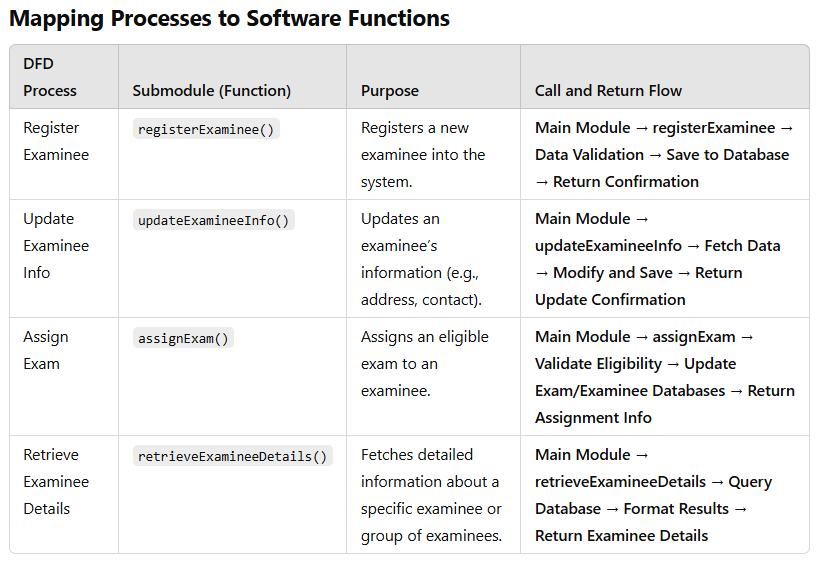
3. Break Down the Submodules

Each subprocess from the DFD (e.g., Register Examinee) becomes a function or submodule in the Call and Return design. Here’s how it maps:

1. Examinees Management (Main Module)
   * Calls submodules for specific functionalities.

Submodules:

* + registerExaminee()
  + updateExamineeInfo()
  + assignExam()
  + retrieveExamineeDetails()



**Structured Chart for "Register Examinee"**

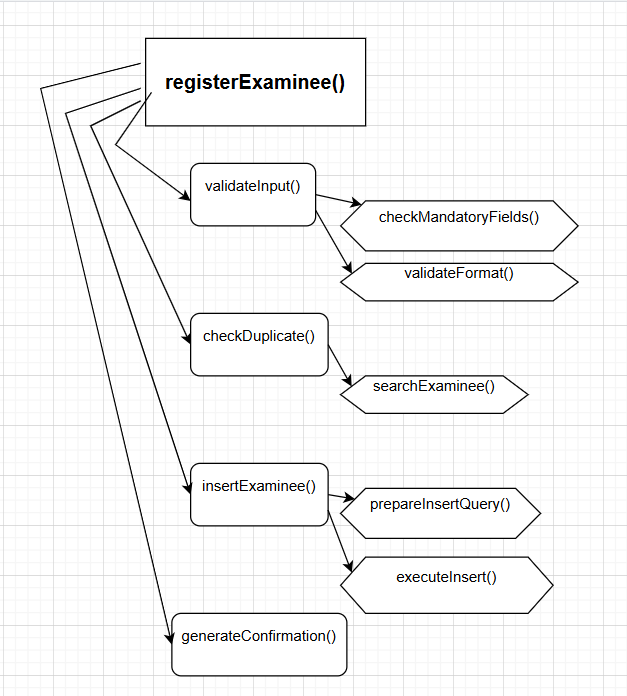
Main Module: Examinees Management

The Examinees Management module contains the submodule registerExaminee() as part of its processes. This is further broken down into smaller sub-functions.

Hierarchy in Structured Chart

1. Main Function: registerExaminee()
   * The central function that manages the examinee registration process.
   * Delegates specific tasks to helper functions.
2. Structured Chart

Below is the hierarchy of functions and responsibilities:

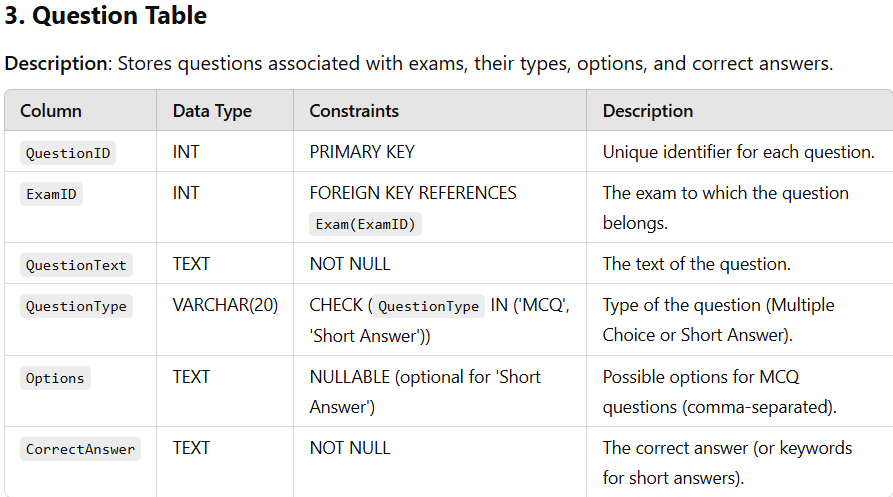


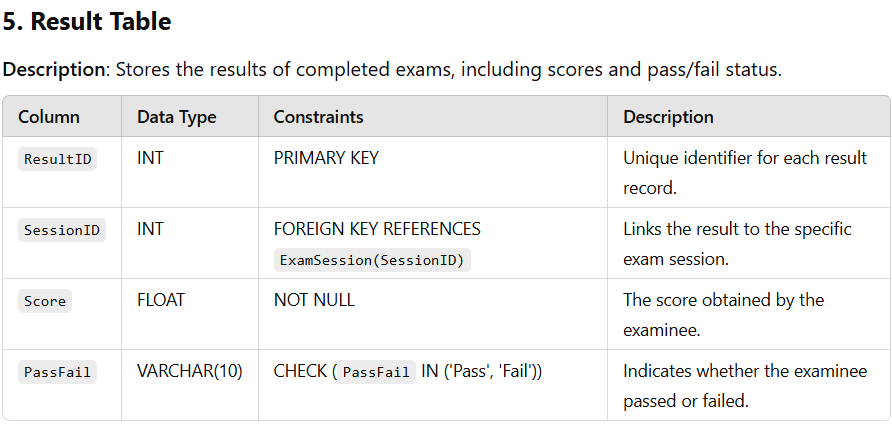
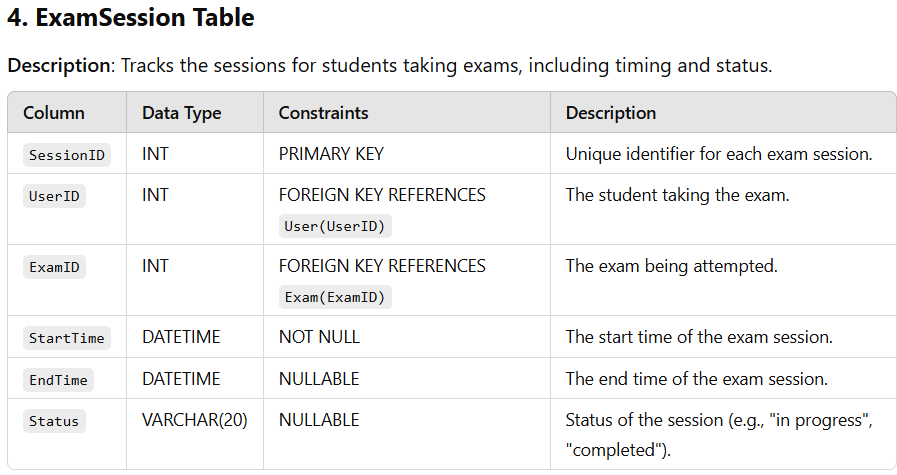
**Descriptions of Submodules**

1. registerExaminee()
   * Responsibility: Main control function for the examinee registration process.
   * Tasks:
     + Calls validateInput() to ensure data is valid.
     + Checks for duplicates using checkDuplicate().
     + Adds the examinee to the database using insertExaminee().
     + Generates a confirmation message for successful registration.
2. validateInput()
   * Responsibility: Ensures input data meets all criteria.
   * Tasks:
     + Calls checkMandatoryFields() to ensure required fields (name, email, ID) are present.
     + Validates specific formats like email and phone using validateFormat().
3. checkDuplicate()
   * Responsibility: Prevents duplicate registrations.
   * Tasks:
     + Uses searchExaminee() to query the database for an existing record with the same ID or email.
4. insertExaminee()
   * Responsibility: Adds new examinee details to the database.
   * Tasks:
     + Formats the data into an SQL query using prepareInsertQuery().
     + Executes the query using executeInsert() to store the data in the Examinees Database (D2).
5. generateConfirmation()
   * Responsibility: Creates a success or error message for the user interface.

**5.2 Data Design**

# 





**Data Flow Summary**

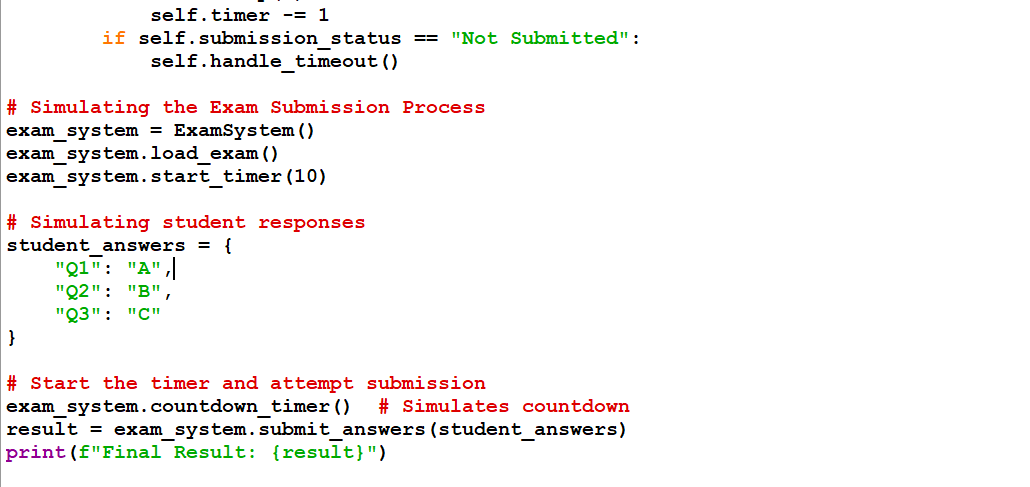
1. **User Registration/Login**:
   * Data from the User table is used for login and session management.
2. **Exam Creation and Scheduling**:
   * Admins use the Exam and Question tables to create and schedule exams.
3. **Exam Execution**:
   * Students start an exam, creating a new record in the ExamSession table.
4. **Result Generation**:
   * The system evaluates answers and stores the result in the Result table.

**6.CODING**

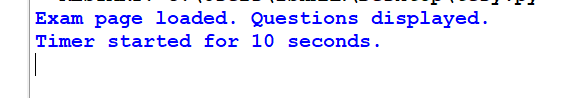
Overview:

This Python code simulates an exam system where a student can load the exam, submit answers within a given time, and have their submission validated. It handles the timer, submission, validation, and errors timeout or invalid answers).





**OUTPUT**



Steps:

1.Initialization:

The ExamSystem class is initialized with a timer and submission status.

2.Loading the Exam:

The load\_exam() method displays the exam questions to the student.

3.Starting the Timer:

start\_timer(duration) starts a countdown for the exam, with the provided duration in seconds.

4.Submitting Answers:

submit\_answers(answers) handles the answer submission process. It checks if the time is still available:

If time is available, it proceeds to validate the submission.

If the time is expired, it calls handle\_timeout().

5.Validation:

validate\_submission(answers) checks if the answers are in a valid format (a non-empty dictionary).

If valid, the answers are saved, and a success message is shown.

If invalid, handle\_errors() is called to return an error.

Handling Timeout:

If the timer expires, handle\_timeout() marks the submission as failed.

6.Countdown:

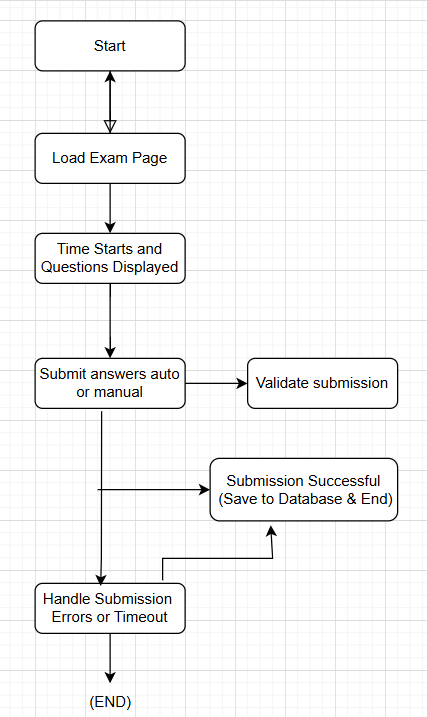
The countdown\_timer() method simulates the countdown, decreasing the timer every second. If the student doesn't submit before time runs out, a timeout is triggered.

7.Simulation:

The code runs a simulation where the student answers three questions and attempts to submit them within a 10-second timer.

**7.Testing**

Flow graph



**1. Cyclomatic Complexity**

Cyclomatic complexity is a software metric that measures the number of independent paths through the program's source code. It is calculated using the formula:

Cyclomatic Complexity(CC)=E−N+2P\text{Cyclomatic Complexity} (CC) = E - N + 2PCyclomatic Complexity(CC)=E−N+2P

Where:

* EEE = number of edges
* NNN = number of nodes
* PPP = number of connected components (usually 1)

**Flow Graph Breakdown:**

* **Nodes (N)**:
  1. Start
  2. Load Exam Page
  3. Timer Starts & Questions Displayed
  4. Submit Answers
  5. Validate Submission
  6. Submission Successful
  7. Handle Submission Errors or Timeout
  8. End

Total nodes = **8**.

* **Edges (E)**:
  1. Start → Load Exam Page
  2. Load Exam Page → Timer Starts & Questions Displayed
  3. Timer Starts & Questions Displayed → Submit Answers
  4. Submit Answers → Validate Submission
  5. Validate Submission → Submission Successful
  6. Validate Submission → Handle Submission Errors or Timeout
  7. Submission Successful → Save to Database & End
  8. Handle Submission Errors or Timeout → End

Total edges = **8**.

* **Connected Components (P)**: The graph is a single connected component, so P=1P = 1P=1.

**Cyclomatic Complexity Calculation:**

CC=E−N+2P=8−8+2(1)=2CC = E - N + 2P = 8 - 8 + 2(1) = 2CC=E−N+2P=8−8+2(1)=2

Thus, the **cyclomatic complexity** is **2**.

**2. Number of Independent Paths**

The number of independent paths is equal to the **Cyclomatic Complexity**, which is **2**.

**Independent Paths**

The independent paths represent the unique control flow paths through the program. For a flow graph with Cyclomatic Complexity 2, there are **two independent paths**:

1. **Path 1**: Start → Load Exam Page → Timer Starts & Questions Displayed → Submit Answers → Validate Submission → Submission Successful → Save to Database & End
2. **Path 2**: Start → Load Exam Page → Timer Starts & Questions Displayed → Submit Answers → Validate Submission → Handle Submission Errors or Timeout → End

These two paths represent:

* Path 1: A successful exam submission.
* Path 2: An error or timeout during submission.

**3. Show Independent Paths**

Here are the independent paths visually broken down:

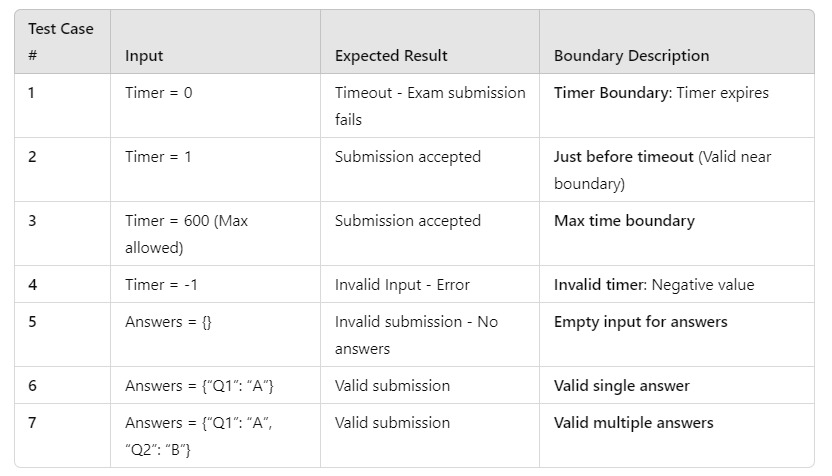
1. **Path 1**:
   * Start → Load Exam Page → Timer Starts & Questions Displayed → Submit Answers → Validate Submission → Submission Successful → Save to Database & End
2. **Path 2**:
   * Start → Load Exam Page → Timer Starts & Questions Displayed → Submit Answers → Validate Submission → Handle Submission Errors or Timeout → End

**Boundary Value Analysis (BVA) and Equivalence Class Partitioning (ECP)**

**Boundary Value Analysis (BVA)**

Boundary value analysis tests the boundaries of inputs, where errors are more likely to occur. Here are the relevant boundary conditions for the **exam submission process** and **timer** validation:

1. **Timer Boundary Values**:
   * **Valid Boundary Values**:
     + Timer = 0 (boundary where the timer expires)
     + Timer = 1 (just before the timer expires)
     + Timer = Max allowed time (e.g., 600 seconds)
   * **Invalid Boundary Values**:
     + Timer < 0 (invalid value)
2. **Answer Submission Boundary Values**:
   * **Valid Boundary Values**:
     + Answers = {“Q1”: “A”, “Q2”: “B”} (valid submission with minimal data)
   * **Invalid Boundary Values**:
     + Empty answers = {} (invalid submission)
     + Malformed answers = {“Q1”: “A”} (incomplete or invalid submission)

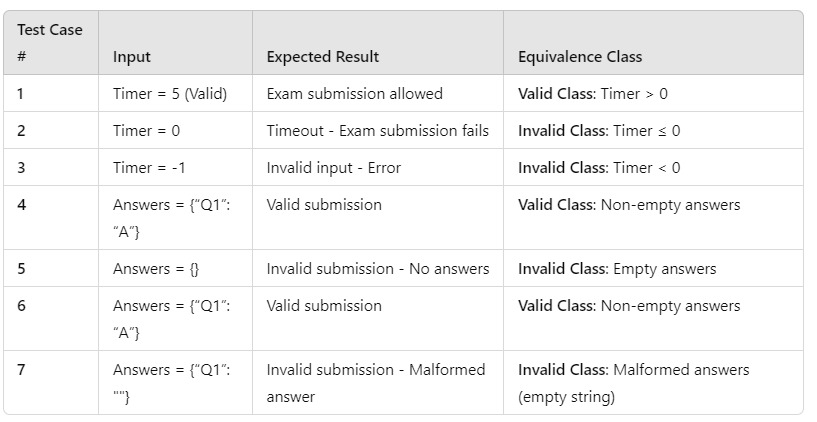


**Equivalence Class Partitioning (ECP)**

Equivalence class partitioning divides inputs into valid and invalid classes. For this system, we consider the following equivalence classes:

1. **Timer**:
   * **Valid Class**: Timer > 0 (time remaining for submission)
   * **Invalid Class**: Timer = 0 or Timer < 0 (timeout or invalid time)
2. **Answers**:
   * **Valid Class**: Non-empty answers (e.g., {“Q1”: “A”, “Q2”: “B”})
   * **Invalid Class**: Empty answers (e.g., {}), Malformed answers (e.g., {“Q1”: “”})

By testing one representative value from each equivalence class, we ensure we cover a wide range of scenarios without redundant tests.



**References**

* 1. **Software Engineering: A Practitioner's Approach** by Roger S. Pressman
  2. **Stack Overflow**
  3. **Github**
  4. **GeeksforGeeks**
  5. **ChatGPT**