Advanced Software Design

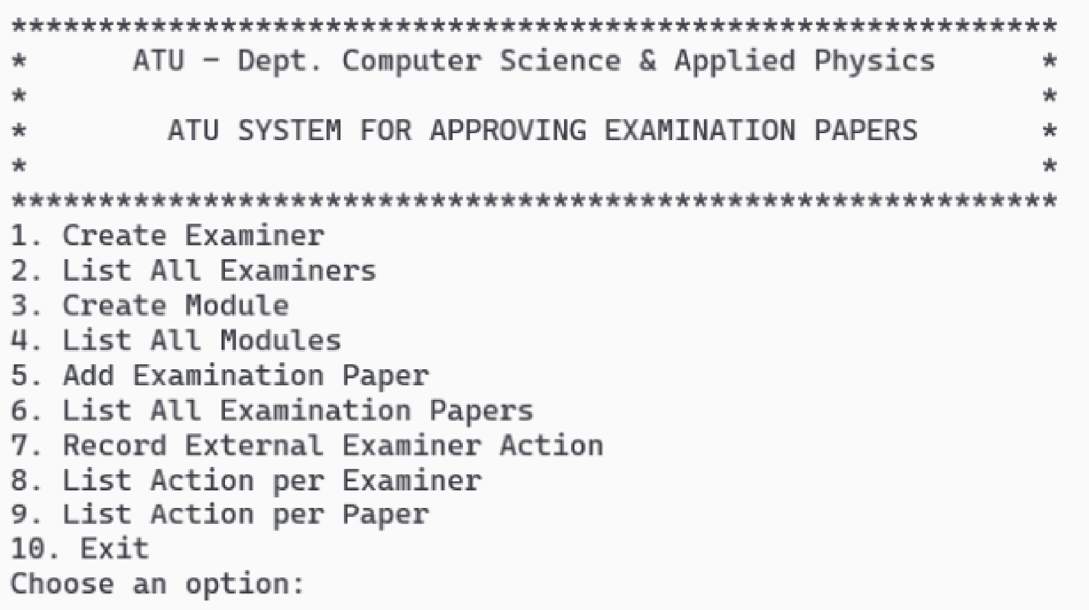
Rodrigo Almeida - G00377123

Introduction

The ATU Examination Paper Management System is an API designed following the principles of Data-Oriented Programming (DOP) and SOLID. This is a project for the Advanced Software Design Module. The aim is to provide a reliable, easy-to-use, and maintainable interface for interacting with examination data.

Key Functionalities

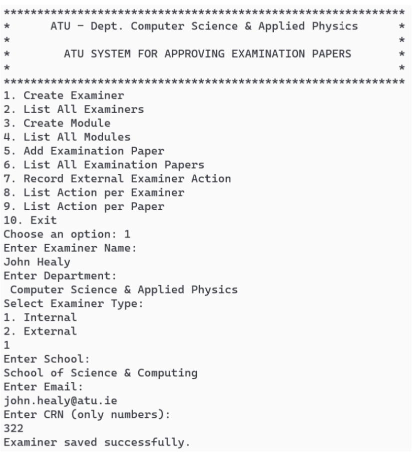
Main Menu



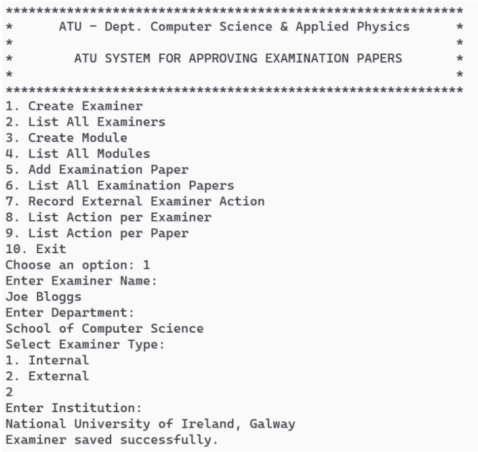
```
************************************************************
*      ATU - Dept. Computer Science & Applied Physics      *
*                                                          *
*        ATU SYSTEM FOR APPROVING EXAMINATION PAPERS       *
*                                                          *
************************************************************
1. Create Examiner
2. List All Examiners
3. Create Module
4. List All Modules
5. Add Examination Paper
6. List All Examination Papers
7. Record External Examiner Action
8. List Action per Examiner
9. List Action per Paper
10. Exit
Choose an option:
```

Examiner Management

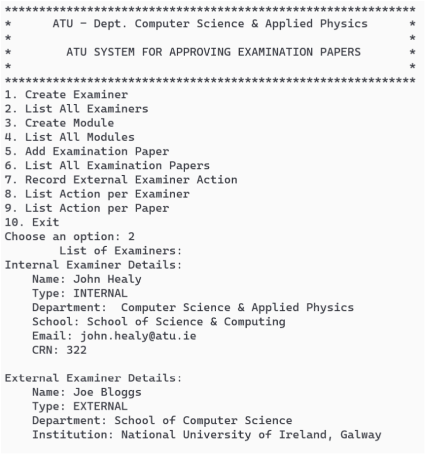
- **Creation and listing of examiners**: Handle operations for internal and external examiners. Includes creation of new examiners and listing all examiners.



```
************************************************************
*      ATU - Dept. Computer Science & Applied Physics      *
*                                                          *
*        ATU SYSTEM FOR APPROVING EXAMINATION PAPERS       *
*                                                          *
************************************************************
1. Create Examiner
2. List All Examiners
3. Create Module
4. List All Modules
5. Add Examination Paper
6. List All Examination Papers
7. Record External Examiner Action
8. List Action per Examiner
9. List Action per Paper
10. Exit
Choose an option: 1
Enter Examiner Name:
John Healy
Enter Department:
 Computer Science & Applied Physics
Select Examiner Type:
1. Internal
2. External
1
Enter School:
School of Science & Computing
Enter Email:
john.healy@atu.ie
Enter CRN (only numbers):
322
Examiner saved successfully.
```



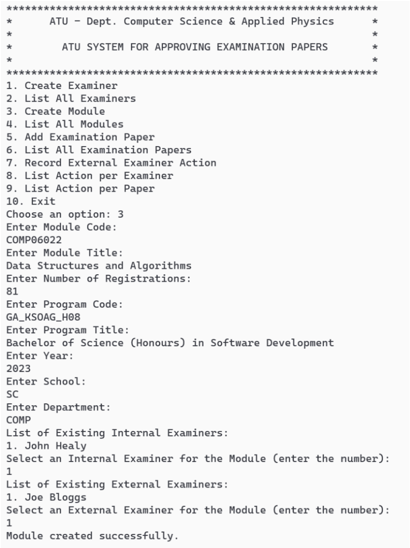
```
************************************************************
*      ATU - Dept. Computer Science & Applied Physics      *
*                                                          *
*        ATU SYSTEM FOR APPROVING EXAMINATION PAPERS       *
*                                                          *
************************************************************
1. Create Examiner
2. List All Examiners
3. Create Module
4. List All Modules
5. Add Examination Paper
6. List All Examination Papers
7. Record External Examiner Action
8. List Action per Examiner
9. List Action per Paper
10. Exit
Choose an option: 1
Enter Examiner Name:
Joe Bloggs
Enter Department:
School of Computer Science
Select Examiner Type:
1. Internal
2. External
2
Enter Institution:
National University of Ireland, Galway
Examiner saved successfully.
```



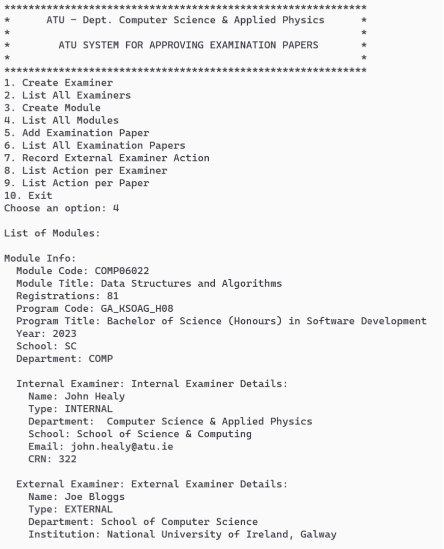
```
************************************************************
*      ATU - Dept. Computer Science & Applied Physics      *
*                                                          *
*        ATU SYSTEM FOR APPROVING EXAMINATION PAPERS       *
*                                                          *
************************************************************
1. Create Examiner
2. List All Examiners
3. Create Module
4. List All Modules
5. Add Examination Paper
6. List All Examination Papers
7. Record External Examiner Action
8. List Action per Examiner
9. List Action per Paper
10. Exit
Choose an option: 2
        List of Examiners:
Internal Examiner Details:
    Name: John Healy
    Type: INTERNAL
    Department:  Computer Science & Applied Physics
    School: School of Science & Computing
    Email: john.healy@atu.ie
    CRN: 322

External Examiner Details:
    Name: Joe Bloggs
    Type: EXTERNAL
    Department: School of Computer Science
    Institution: National University of Ireland, Galway
```

Academic Modules Management

- **Creation and Listing of modules**: Manage academic modules with functionalities for creating new modules and listing all available ones.



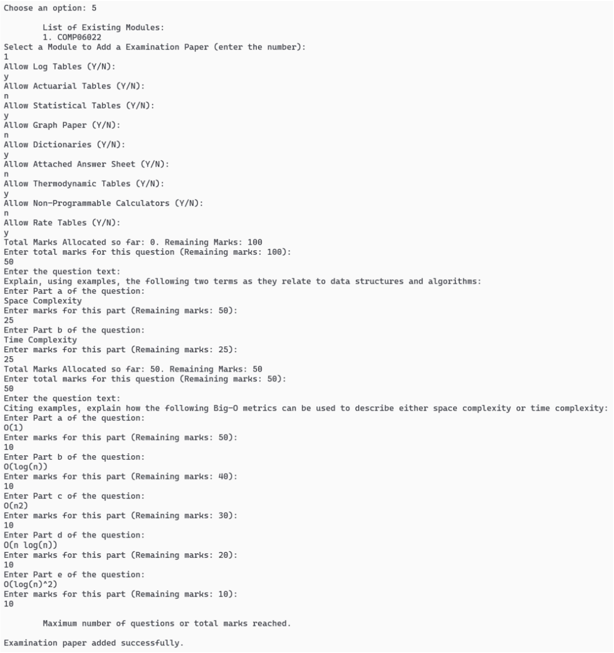
```
************************************************************
*      ATU - Dept. Computer Science & Applied Physics      *
*                                                          *
*        ATU SYSTEM FOR APPROVING EXAMINATION PAPERS       *
*                                                          *
************************************************************
1. Create Examiner
2. List All Examiners
3. Create Module
4. List All Modules
5. Add Examination Paper
6. List All Examination Papers
7. Record External Examiner Action
8. List Action per Examiner
9. List Action per Paper
10. Exit
Choose an option: 3
Enter Module Code:
COMP06022
Enter Module Title:
Data Structures and Algorithms
Enter Number of Registrations:
81
Enter Program Code:
GA_KSOAG_H08
Enter Program Title:
Bachelor of Science (Honours) in Software Development
Enter Year:
2023
Enter School:
SC
Enter Department:
COMP
List of Existing Internal Examiners:
1. John Healy
Select an Internal Examiner for the Module (enter the number):
1
List of Existing External Examiners:
1. Joe Bloggs
Select an External Examiner for the Module (enter the number):
1
Module created successfully.
```



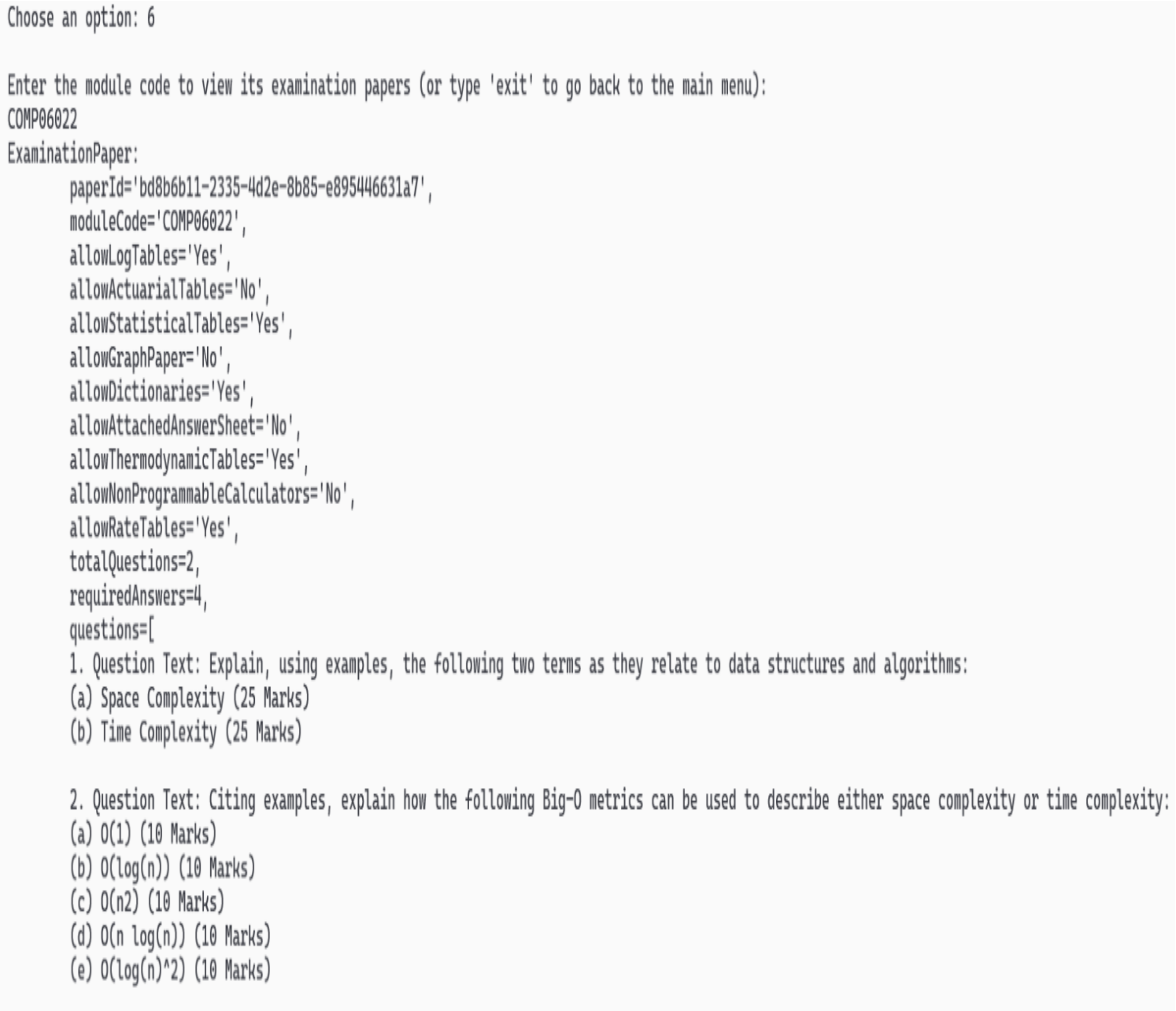
```
************************************************************
*      ATU - Dept. Computer Science & Applied Physics      *
*                                                          *
*        ATU SYSTEM FOR APPROVING EXAMINATION PAPERS       *
*                                                          *
************************************************************
1. Create Examiner
2. List All Examiners
3. Create Module
4. List All Modules
5. Add Examination Paper
6. List All Examination Papers
7. Record External Examiner Action
8. List Action per Examiner
9. List Action per Paper
10. Exit
Choose an option: 4

List of Modules:

Module Info:
  Module Code: COMP06022
  Module Title: Data Structures and Algorithms
  Registrations: 81
  Program Code: GA_KSOAG_H08
  Program Title: Bachelor of Science (Honours) in Software Development
  Year: 2023
  School: SC
  Department: COMP

  Internal Examiner: Internal Examiner Details:
    Name: John Healy
    Type: INTERNAL
    Department:  Computer Science & Applied Physics
    School: School of Science & Computing
    Email: john.healy@atu.ie
    CRN: 322

  External Examiner: External Examiner Details:
    Name: Joe Bloggs
    Type: EXTERNAL
    Department: School of Computer Science
    Institution: National University of Ireland, Galway
```

Examination Paper Management

- **Paper Handling**: Creation of new examination papers and all questions following the module rules. It also lists all papers.



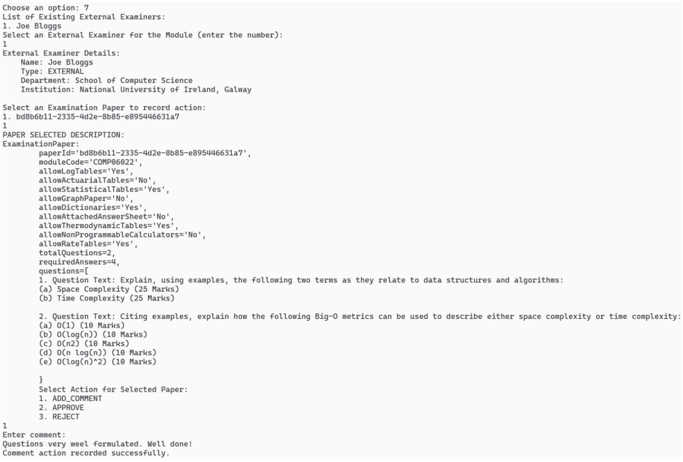
```
Choose an option: 5

        List of Existing Modules:
        1. COMP06022
Select a Module to Add a Examination Paper (enter the number):
1
Allow Log Tables (Y/N):
y
Allow Actuarial Tables (Y/N):
n
Allow Statistical Tables (Y/N):
y
Allow Graph Paper (Y/N):
n
Allow Dictionaries (Y/N):
y
Allow Attached Answer Sheet (Y/N):
n
Allow Thermodynamic Tables (Y/N):
y
Allow Non-Programmable Calculators (Y/N):
n
Allow Rate Tables (Y/N):
y
Total Marks Allocated so far: 0. Remaining Marks: 100
Enter total marks for this question (Remaining marks: 100):
50
Enter the question text:
Explain, using examples, the following two terms as they relate to data structures and algorithms:
Enter Part a of the question:
Space Complexity
Enter marks for this part (Remaining marks: 50):
25
Enter Part b of the question:
Time Complexity
Enter marks for this part (Remaining marks: 25):
25
Total Marks Allocated so far: 50. Remaining Marks: 50
Enter total marks for this question (Remaining marks: 50):
50
Enter the question text:
Citing examples, explain how the following Big-O metrics can be used to describe either space complexity or time complexity:
Enter Part a of the question:
O(1)
Enter marks for this part (Remaining marks: 50):
10
Enter Part b of the question:
O(log(n))
Enter marks for this part (Remaining marks: 40):
10
Enter Part c of the question:
O(n2)
Enter marks for this part (Remaining marks: 30):
10
Enter Part d of the question:
O(n log(n))
Enter marks for this part (Remaining marks: 20):
10
Enter Part e of the question:
O(log(n)^2)
Enter marks for this part (Remaining marks: 10):
10

        Maximum number of questions or total marks reached.

Examination paper added successfully.
```



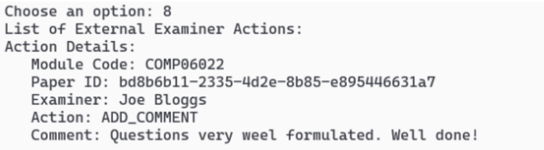
```
Choose an option: 6

Enter the module code to view its examination papers (or type 'exit' to go back to the main menu):
COMP06022
ExaminationPaper:
        paperId='bd8b6b11-2335-4d2e-8b85-e895446631a7',
        moduleCode='COMP06022',
        allowLogTables='Yes',
        allowActuarialTables='No',
        allowStatisticalTables='Yes',
        allowGraphPaper='No',
        allowDictionaries='Yes',
        allowAttachedAnswerSheet='No',
        allowThermodynamicTables='Yes',
        allowNonProgrammableCalculators='No',
        allowRateTables='Yes',
        totalQuestions=2,
        requiredAnswers=4,
        questions=[
        1. Question Text: Explain, using examples, the following two terms as they relate to data structures and algorithms:
        (a) Space Complexity (25 Marks)
        (b) Time Complexity (25 Marks)

        2. Question Text: Citing examples, explain how the following Big-O metrics can be used to describe either space complexity or time complexity:
        (a) O(1) (10 Marks)
        (b) O(log(n)) (10 Marks)
        (c) O(n2) (10 Marks)
        (d) O(n log(n)) (10 Marks)
        (e) O(log(n)^2) (10 Marks)
```

External Examiner Actions

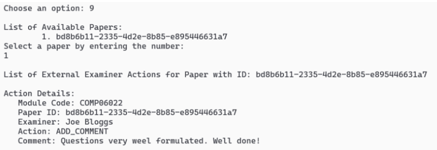
- **Action Recording**: Records `comments`, `approvals`, and `rejections` by external examiners on examination papers. It also lists all actions for the Module and the user could choose to list all actions for an existing paper.



```
Choose an option: 7
List of Existing External Examiners:
1. Joe Bloggs
Select an External Examiner for the Module (enter the number):
1
External Examiner Details:
    Name: Joe Bloggs
    Type: EXTERNAL
    Department: School of Computer Science
    Institution: National University of Ireland, Galway

Select an Examination Paper to record action:
1. bd8b6b11-2335-4d2e-8b85-e895446631a7
1
PAPER SELECTED DESCRIPTION:
ExaminationPaper:
        paperId='bd8b6b11-2335-4d2e-8b85-e895446631a7',
        moduleCode='COMP06022',
        allowLogTables='Yes',
        allowActuarialTables='No',
        allowStatisticalTables='Yes',
        allowGraphPaper='No',
        allowDictionaries='Yes',
        allowAttachedAnswerSheet='No',
        allowThermodynamicTables='Yes',
        allowNonProgrammableCalculators='No',
        allowRateTables='Yes',
        totalQuestions=2,
        requiredAnswers=4,
        questions=[
        1. Question Text: Explain, using examples, the following two terms as they relate to data structures and algorithms:
        (a) Space Complexity (25 Marks)
        (b) Time Complexity (25 Marks)

        2. Question Text: Citing examples, explain how the following Big-O metrics can be used to describe either space complexity or time complexity:
        (a) O(1) (10 Marks)
        (b) O(log(n)) (10 Marks)
        (c) O(n2) (10 Marks)
        (d) O(n log(n)) (10 Marks)
        (e) O(log(n)^2) (10 Marks)

        }
        Select Action for Selected Paper:
        1. ADD_COMMENT
        2. APPROVE
        3. REJECT
1
Enter comment:
Questions very weel formulated. Well done!
Comment action recorded successfully.
```



```
Choose an option: 8
List of External Examiner Actions:
Action Details:
   Module Code: COMP06022
   Paper ID: bd8b6b11-2335-4d2e-8b85-e895446631a7
   Examiner: Joe Bloggs
   Action: ADD_COMMENT
   Comment: Questions very weel formulated. Well done!
```



```
Choose an option: 9

List of Available Papers:
        1. bd8b6b11-2335-4d2e-8b85-e895446631a7
Select a paper by entering the number:
1

List of External Examiner Actions for Paper with ID: bd8b6b11-2335-4d2e-8b85-e895446631a7

Action Details:
   Module Code: COMP06022
   Paper ID: bd8b6b11-2335-4d2e-8b85-e895446631a7
   Examiner: Joe Bloggs
   Action: ADD_COMMENT
   Comment: Questions very weel formulated. Well done!
```

Design Patterns Utilization

- **Singleton Pattern**: An example is the `ExaminerUtil` class, implemented to ensure there is only one instance managing the examiner data throughout the application. This pattern is crucial for centralizing and managing shared resources consistently.

- **Factory Method Pattern**: The methods `createExaminer`, `createInternalExaminer`, and `createExternalExaminer` in `ExaminerUtil` class follow the Factory Method pattern. They encapsulate the object creation process and delegate it to subclasses (Internal and External Examiners).

- **Command Pattern**: The `recordAction` method in `ExternalExaminerServiceImpl` is an example of the Command pattern, where an action (like `Add_Comment`, `Approve`, `Reject`) is encapsulated as an object.

- **Strategy Pattern**: The implementation of interfaces `ModuleService`, `ExaminationPaperService`, `ExternalExaminerService` and `QuestionService` represents the Strategy pattern. Different strategies for handling modules, examination papers, and external examiner functionalities are encapsulated behind these interfaces.

SOLID Principles

- **Single Responsibility Principle (SRP)**: Classes like `ModuleServiceImpl`, `ExaminationPaperServiceImpl`, `ExaminerUtil`, each have a single responsibility.
- **Open/Closed Principle (OCP)**: The system is extendable without the need for modification. For example, adding new types of examiners or new modules can be done without altering existing code, especially due to the use of interfaces.
- **Liskov Substitution Principle (LSP)**: The use of interfaces and inheritance (like `Examiner interface` implemented by `InternalExaminer` and `ExternalExaminer`) shows adherence to LSP.
- **Interface Segregation Principle (ISP)**: The application follows ISP by creating specific interfaces (`ModuleService`, `ExaminationPaperService`, `ExternalExaminerService` and `QuestionService`) for specific functionalities.
- **Dependency Inversion Principle (DIP)**: The use of high-level modules like `Runner` depending on abstractions (`ModuleService`, `ExaminationPaperService`, `ExternalExaminerService` and `QuestionService`) instead of concrete classes.

End