Design Document, Usability Document and Design Justification

Automated Classroom Evaluation Performance Application

Software Engineering Lab(CS346)
IIT Guwahati

Submitted by-
Kartikay Goel 180101033
Falak Chhikara - 180101023
Alay Chirag Shah - 180101005
Rahul Choudhary - 180101060

Table of Contents

Introduction

Purpose-:

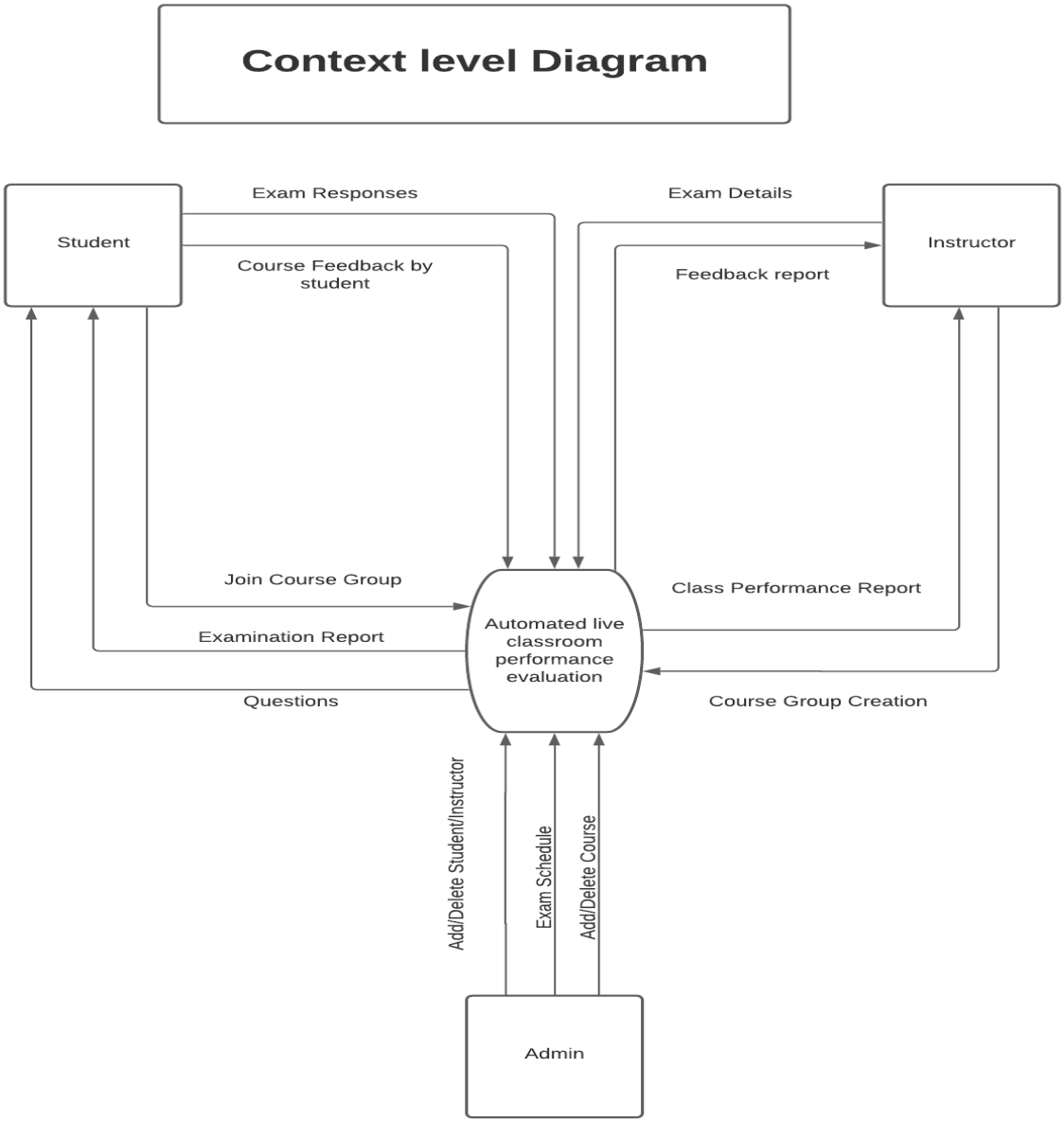
The purpose of this document is to build the base for designing the Automated Class Evaluation Performance Application. The document illustrates Data Flow Diagrams at various levels, the Entity Relationship Diagram, Usability Document in light of the eight golden rules of Schneiderman and Design Justification.

Intended Audience-:

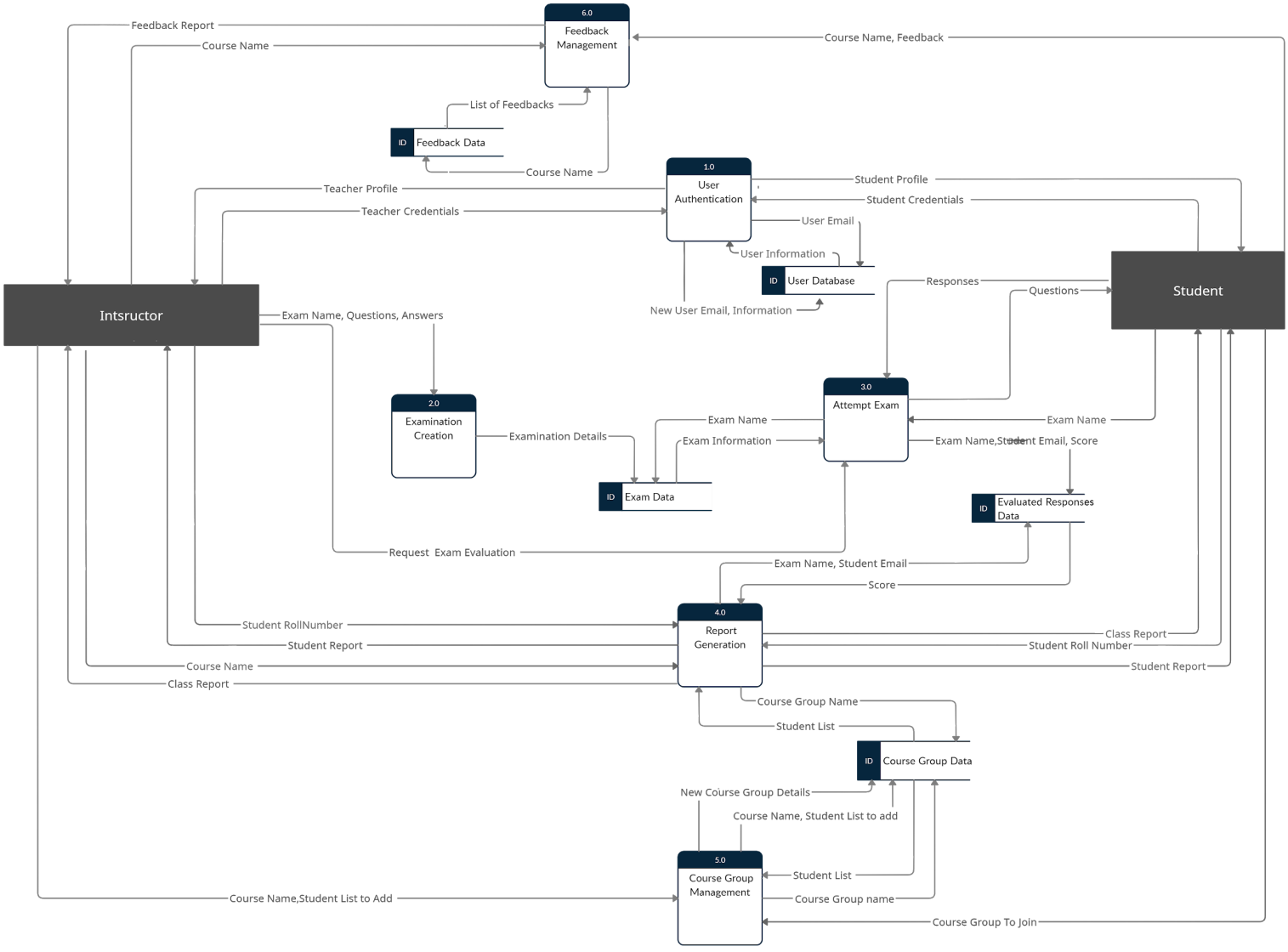
Instructors can use this application to automatically evaluate the score of students in examinations and classes. The intended target audience of this project is college/university instructors and students. This software design document is the second part of a project under the CS345 and CS346 courses - Software Engineering. The intended audience is our course instructor - Dr. Samit Bhattacharya, and the teaching assistants tasked with evaluating this project. This document acts as a guide to check progress regarding the assigned project at various stages.

Data Flow Diagram

Level 0 DFD

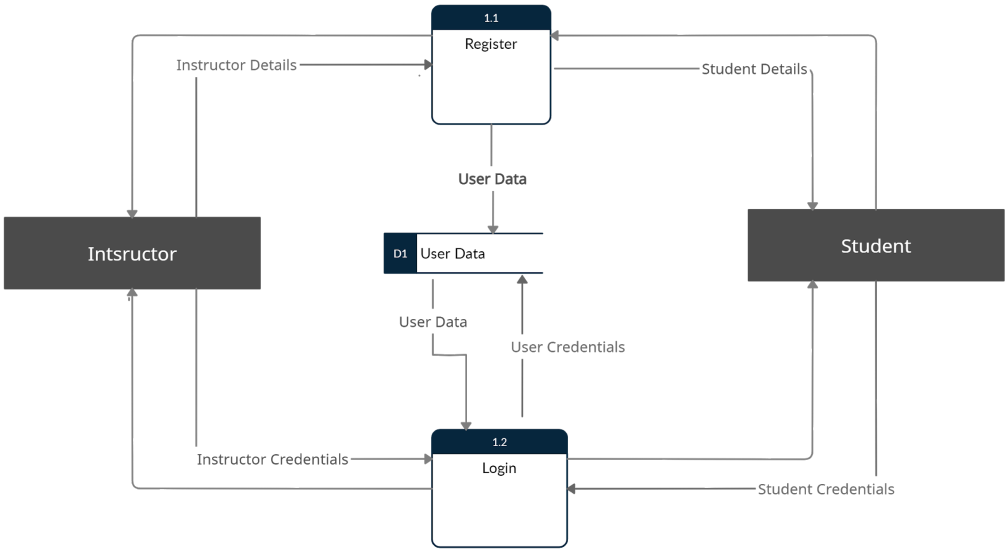


Level 1 DFD

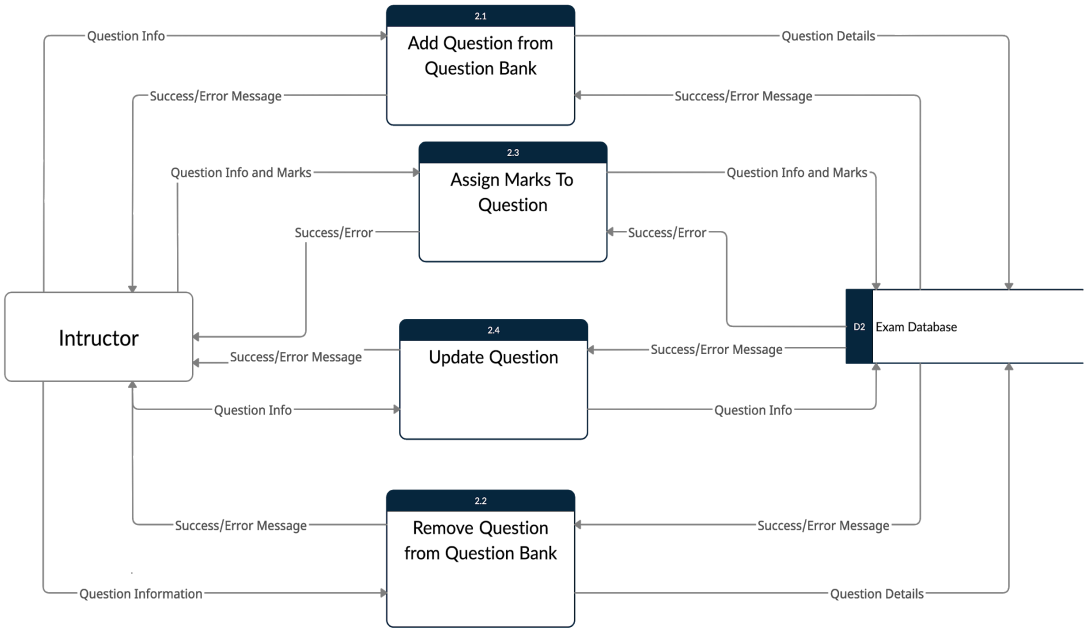

6.0
Feedback Management
Feedback Report
Course Name
Course Name, Feedback
List of Feedbacks
ID
Feedback Data
Course Name
1.0
User Authentication
Teacher Profile
Teacher Credentials
Student Profile
Student Credentials
User Email
User Information
ID
User Database
New User Email, Information
Intsructor
Student
Responses
Questions
Exam Name, Questions, Answers
2.0
Examination Creation
Examination Details
3.0
Attempt Exam
Exam Name
Exam Information
Exam Name
Exam Name,Student Email, Score
ID
Exam Data
ID
Evaluated Responses Data
Request Exam Evaluation
Exam Name, Student Email
Score
4.0
Report Generation
Student RollNumber
Student Report
Course Name
Class Report
Class Report
Student Roll Number
Student Report
Course Group Name
Student List
ID
Course Group Data
New Course Group Details
Course Name, Student List to add
5.0
Course Group Management
Course Name,Student List to Add
Student List
Course Group name
Course Group To Join

Level 2 DFD

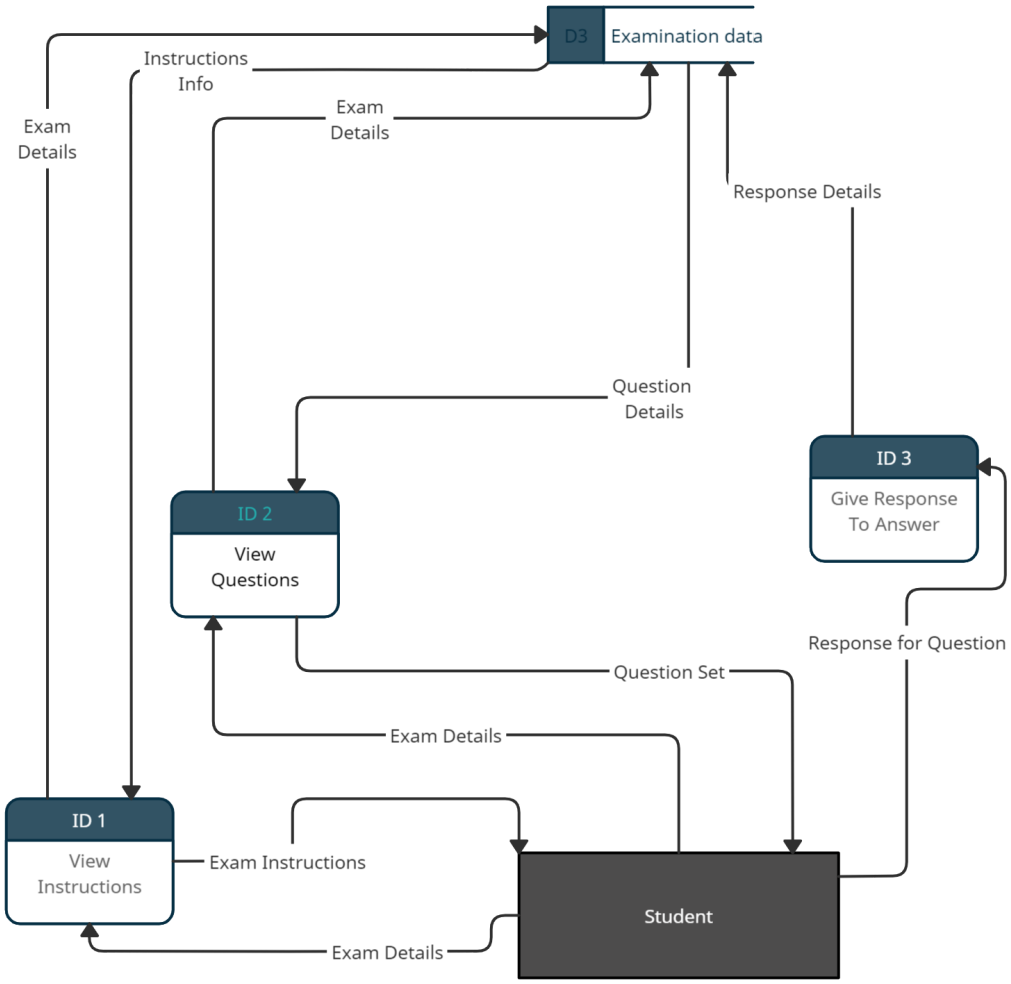
Process 1 - User Authentication



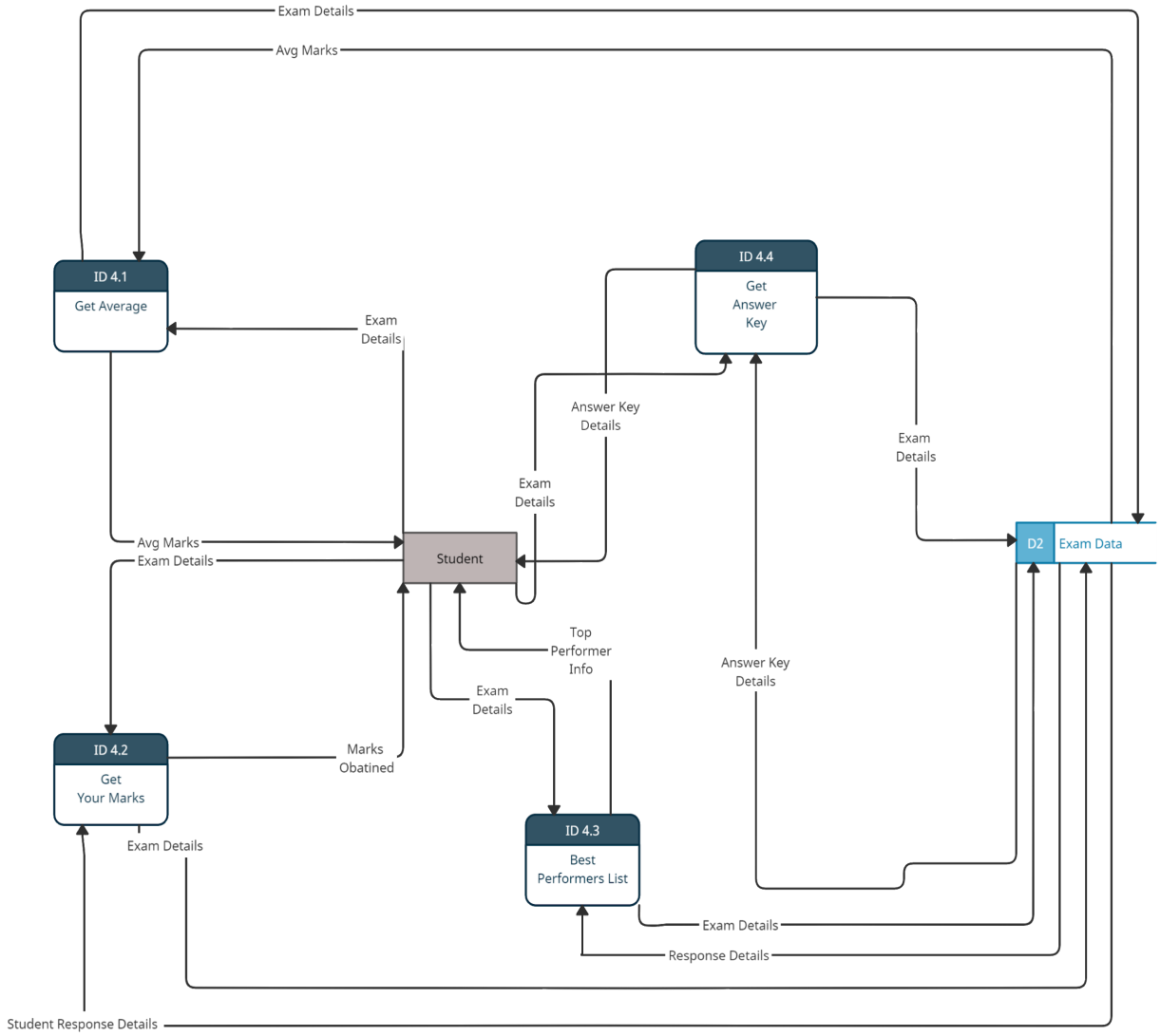
Process 2 - Create Exam



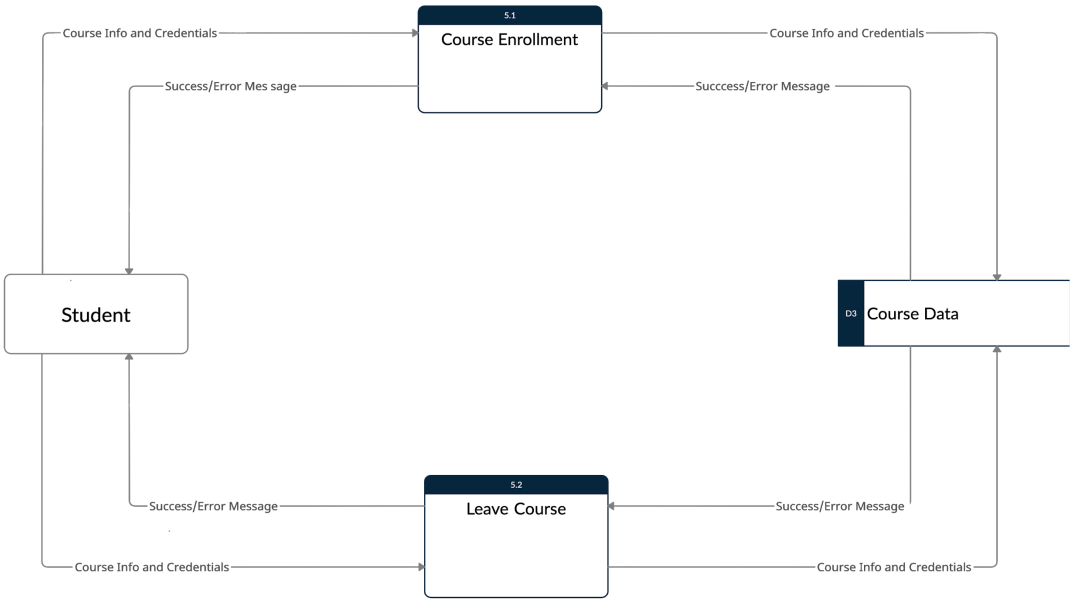
Process 3 - Attempt Exam



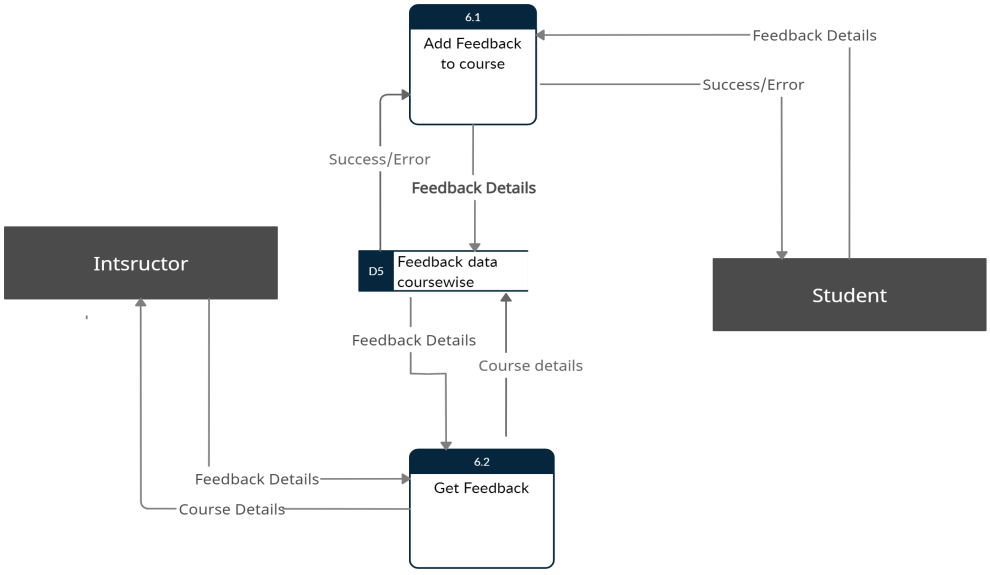
Process 4 - Generate Report



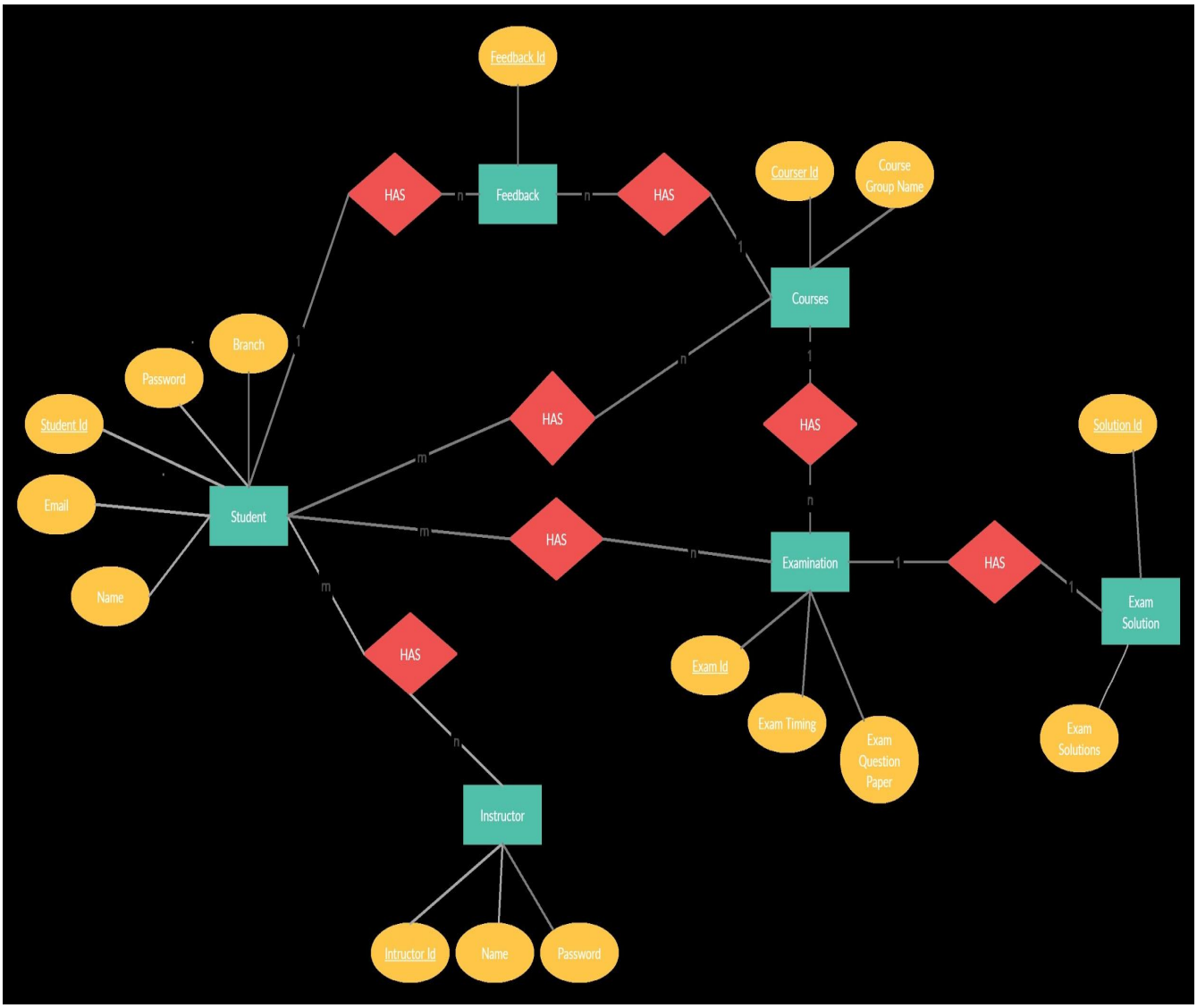
Process 5 - Manage Course Group



Process 6 - Manage Feedback



ER Diagram



The type of relationships are written on the lines in the ER diagram itself.

Usability Document

To improve the usability of an application, we need a very well-designed interface. We have taken care of usability aspects(in the light of eight golden rules of Shneiderman) in the following ways:

1. Strive for consistency:

a. Internal Consistency: The material UI we are going to use will bring a lot of consistency again. Specifications, font faces, colors, and typography will always be the same for all windows that the user encounters. A button with a certain icon on it will perform the same action on all windows. In this way, we hope that users will find the interface very familiar and fluid.

b. External consistency: We plan to follow the visual UI, which is a design interface used by many applications made by Google. In this way we hope that users will find the interface very familiar and fluid. Our automated test evaluation application will have an interface that looks like it offers real offline testing with a timer like a clock. This will make the user experience go hand in hand with giving tests in the exam hall. The visual buttons will have icons that move what they do. For example, the arrow icon of the button that shows the next question, the refresh icon that allows you to reset the answers.

2. Enable frequent users to use shortcuts:

a. Novice: The user will find the portable UI very interesting. The interface will be very suggestive. The interface will be set very clean so that the user can find the tools they want to use easily. Thumbnails will show the action they take. Navigating the thumbnails will show how the user can interact with them. An illustrated help guide will also be available to the user.

b. Expert: Users who have already tried other apps should find the change interesting because the design incorporates useful features from existing apps. The interface of the interface will ensure that the professional user can work efficiently without noticing some difficulty in making changes to our app.

3. Offer informative feedback:

The app is intended to be highly interactive and retrieve useful feedback. The following features will confirm that this is the case.

- Flying over UI clickable objects will change color to indicate that they are clickable. Clicked buttons will change their color.
- Relevant upload barriers and progress barriers will be used during screen loading. This will notify the user of what is happening. Important actions can lead

to confirmation before execution. Example: Before installing the test, the system will upgrade the user to authenticate.

• The color of the question icon will change to green after you attempt that question.

4. Design dialogues to yield closure:

When the actions which the users try to perform are successful, they will receive confirmation. If they fail, users will see a message telling them that the action was not performed, the reason why it was not performed and what they can do about it. For example, when a user submits a test successfully a pop-up message will notify the user that the response has been successfully saved.

5. Simple error handling and offers error prevention:

a. Error handling: Mark text fields where users forgot to provide input on the online sign-in form. The displayed error messages will be easier for the user to forward and will also tell the user what to do to resolve the error. Recovery methods will be available to help the user recover from unintentional actions.

b. Error prevention: With the ability to prevent an error the user will be asked System verification prior to any major actions such as submitting test responses, making them aware of the consequences of their actions. Automatic testing will be performed on the entered data (such as password, email) and warnings or reminders will be given incorrect input formats. We will try to associate red with the wrong actions as much as possible. The on-screen items will be clearly organized to minimize errors.

6. Permit easy reversal of actions:

The following options are provided to reverse actions: The undo option may change the last change made on the page.

• Reset option may postpone the last use of the reset option. The backspace key can be used to delete the last text entered. The erase tool can be used to clear previously drawn lines. The unshare button can be used to unshare a shared document with another user.

7. Support Internal Locus of control:

External compatibility provides the user with the normal comfort of performing tasks. The fact that actions are reversible easily means that the user will be able to quickly revert to whatever they are doing that allows them to perform tasks without fear of constant failure. The feedback provided by the system will provide a sense of security and comfort in users by allowing them to see activities in progress. Good error

management and toolkit tools allow users to easily learn the functionality of the elements. Users will be given enough options to make them feel in control but not so much to make them feel lost.

8. Reduce short-term memory load:
The required learning value will be minimal as the interface will guide the user in each step. Button icons, navigation information over window objects, tips that guide the user on how to do something will mean the user doesn't have to remember a lot of things. The interface will make the experience very fluid. Moreover, all the functionalities offered by the application follow the 7+-2 rule. We have ensured that the user recognizes all the actions instead of recalling them. For example, while signing up, the user is asked to provide its details such as email, password, and name. After successfully submitting the signup request, the user is supposed to verify himself/herself by clicking on a verification link sent to his/her email. Every tech company uses this process of signing up and verification.

Design Justification

COHESION

Module 1 : Authentication

Functions: a. Registration or Signup
b. Login/Logout

Above functions exhibit communication cohesion because they execute over the same database(user database)

Module 2 - Create Exam

Functions:
a. Add Question
b. Remove Question
c. Update Question

Modules show consecutive integration such as DeleteQuestion () and UpdateQuestion () iff when the user uses the AddQuestion () function.
All will show functional integration by being part of the same process (Question Management) The first two tasks indicate temporary integration.

Module 3 : Attempt Exam

Functions:

a. Get Instruction
b. Display Summary
d. Reset chosen option
e. Next Question
f. Answer Question
g. Get Questions

All operations have a temporary connection and communication interface as they all do during continuous testing and work with the same test details.

Module 4 : Generate Report

Functions: a. Get Average
b. Get Rank
c. Get Exam Statistics
d. Get Answer Key
e. Best Performers List
f. Get Marks

All operations have a temporary all-in-one interface that performs simultaneously during test analysis. GetAverage (), getRank () and getMarks () functions will display active integration by being part of the same process.
They all also show interactions by performing the same test data.

Module 5 - Course Group Management

Functions:

a. Enroll in Course
b. Leave Course

The modules show sequential cohesion as LeaveCourse() iff the user uses the function EnrollinCourse().All of them also show communication cohesion for acting on the same course data.

Module 6: Feedback management

Functions:

a. Add Feedback to course

b. Get Feedback

Sequential cohesion is exhibited by this module as getFeedback() works if the user uses the function AddfeebackToCourse().
All of them also show communication cohesion for acting on the same course group data.

COUPLING

1) Control Coupling
Modules Analysis shows control of the Evaluate Exam and Exempt Exam shows control of the Manage Question Bank and Examate Exam shows Control coupling with Manage Attempt Exam as it is completely dependent on each other as the previous module only works if the test data comes from the latest module .

2) Data Coupling
Join / Exit Module and Course FeedBack modules show Data Combination as they both discuss course data.Or the modules dealing with QuestionBank, Exempt Exam, Examate Exam and Analysis also interact with the test data and questionnaire which is why it shows data integration.This design shows high unity and minimal integration so we have completed it as a respectable design.
Each module has more than one combination and the modules show a very small number of interactions so the modules that can be independently sought are active.

BALANCE BETWEEN COHESION AND COUPLING

Our design has a perfect balance of **high cohesion and low coupling,** which is a demanding trait of a good software design**.** Each module in our design shows more than one kind of cohesion (as evident earlier) that enabled functions within the same module to have some kind of commonality. Because of high cohesion and low coupling, the modules of our design can be called **functionally-independent** modules. Our design, is thus, a good stimulation of software engineering demand.