Criterion B: Design

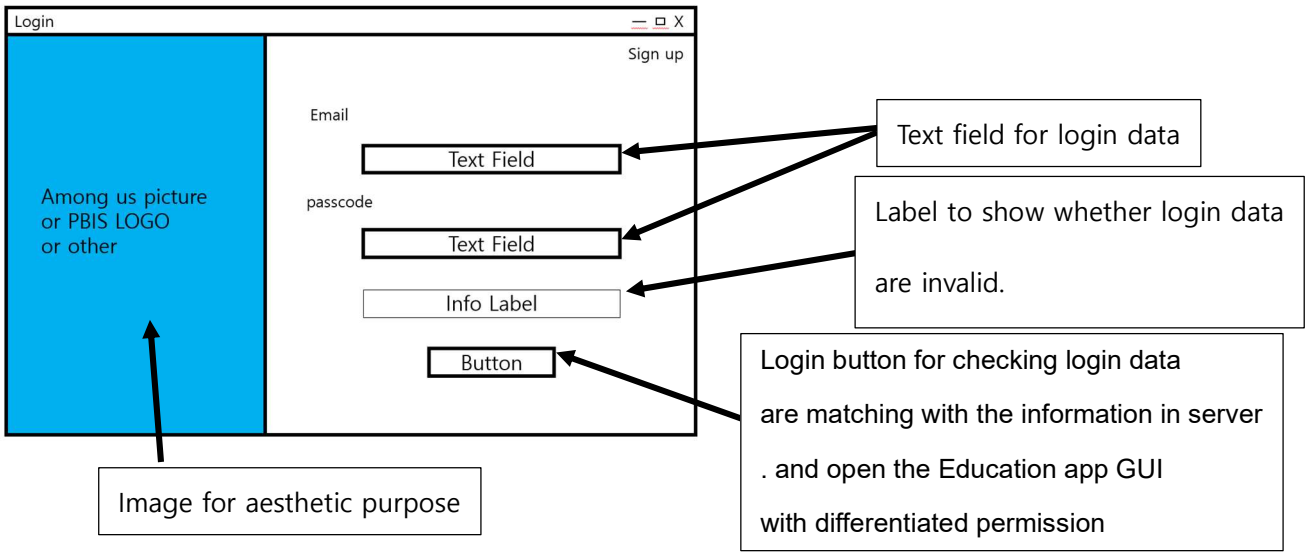
Libraries

Library name	Description
Java swing	It is useful for producing lightweight desktop applications since it can be used to construct window-based apps.
Json	it is a common data interchange format that may be used to transfer data between systems and platforms that do not immediately comprehend one another.
Jcalendar	This library helps to create a calendar in swing GUIs. The user may choose a date using the calendar, and developers can utilize the calendar for additional purposes afterword.
MySQL connecter	It gives client programs access to the MySQL server. Through this connection, it is possible to save, access, and perform queries.
junit-platform-console-standalone	Testing frameworks can be launched on the JVM using the JUnit Platform as a foundation. When creating or debugging programs, this is useful.

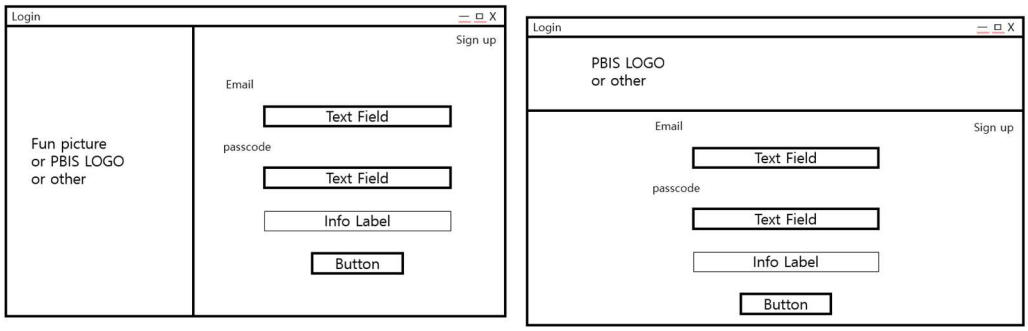
Design of GUIs

Login Form

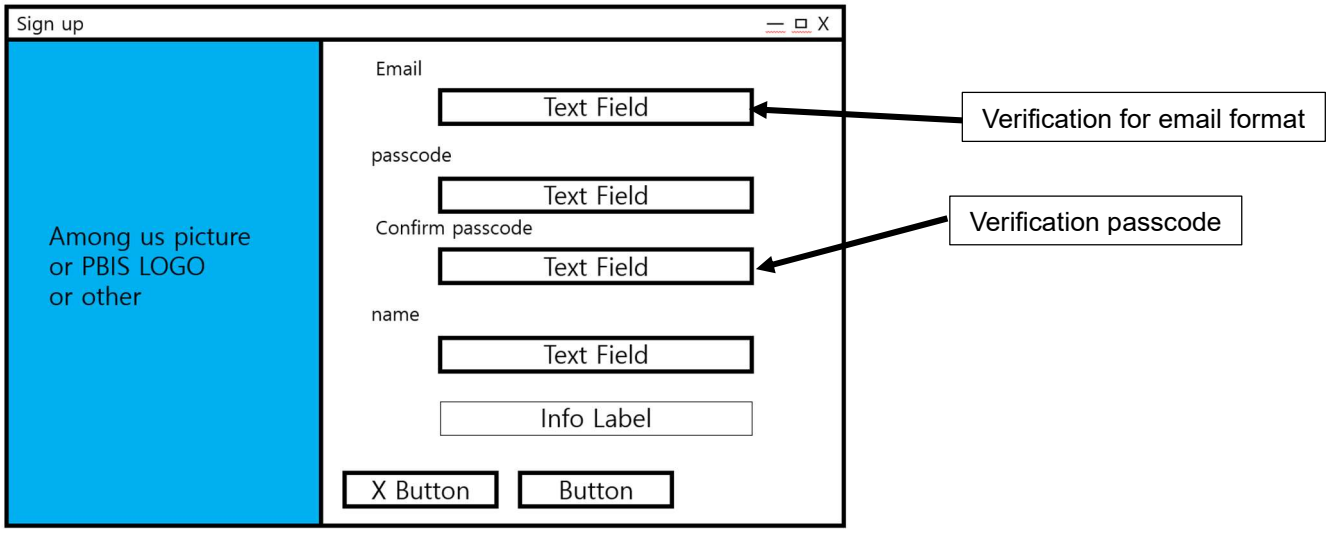
All designs are created by using Microsoft PowerPoint



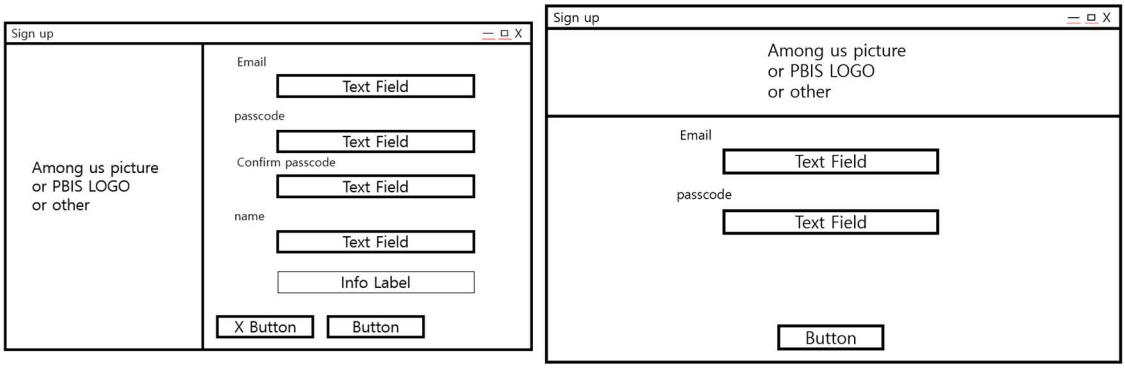
Previous version and another idea



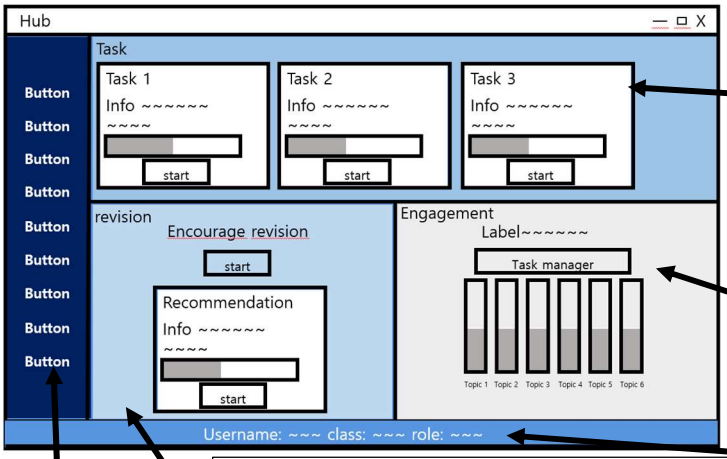
Sign up



Previous version and another idea



Hub



Task box that shows info, result gauge bar And start button.

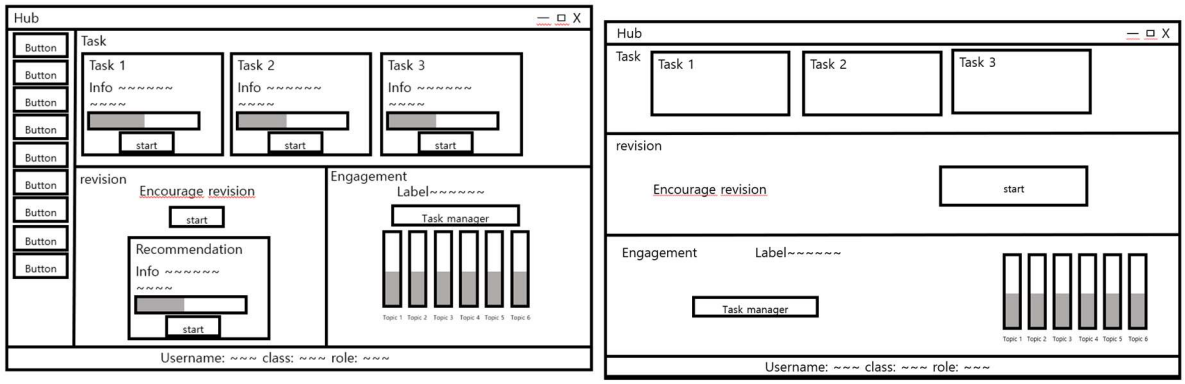
Engagement panel that has users predict grades gauge bar by topic, and button that connects with task manager.

Label that identifies users' info

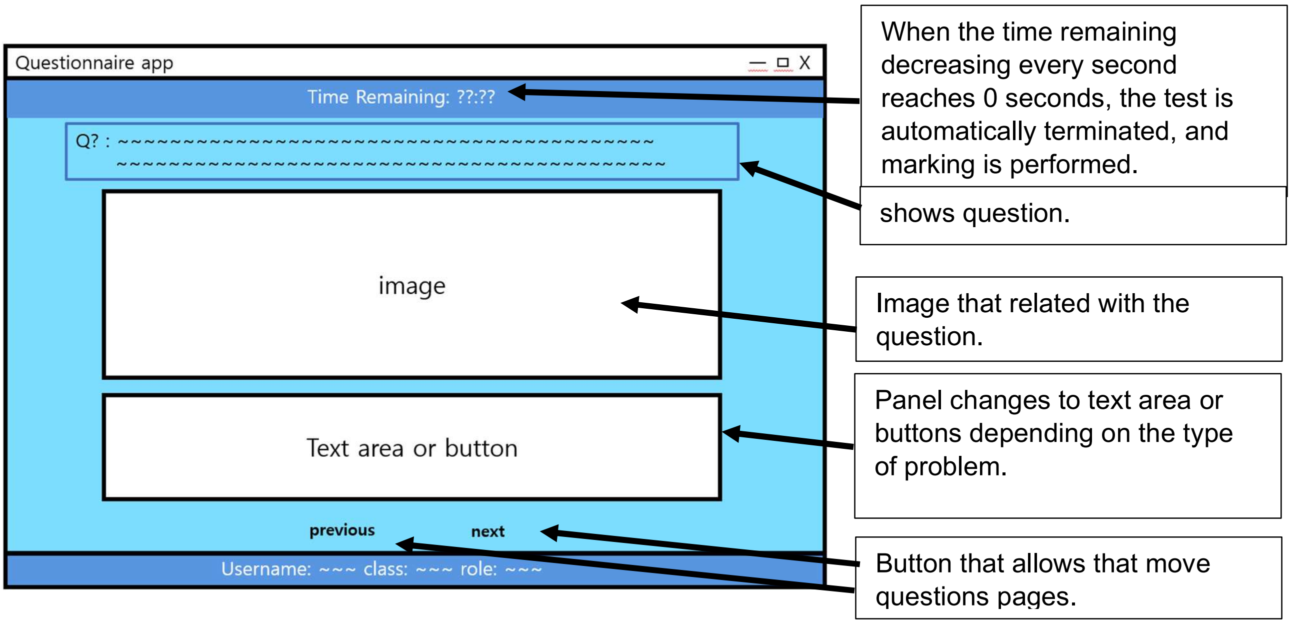
revision panel that has recommended task specialized for user, and button that connects with revision page.

Buttons for the additional feature

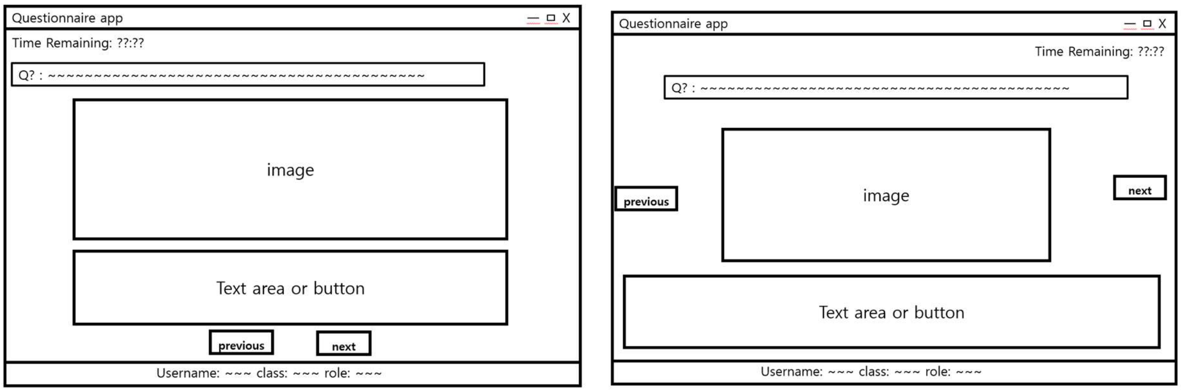
Previous version and another idea



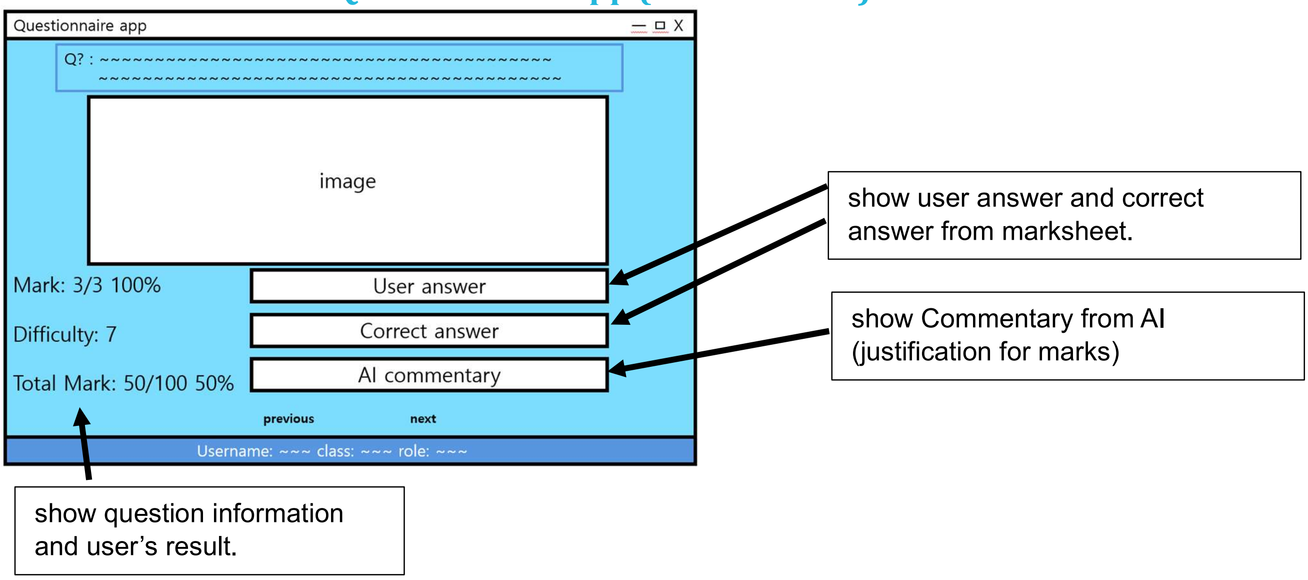
Questionnaire app (test mode)



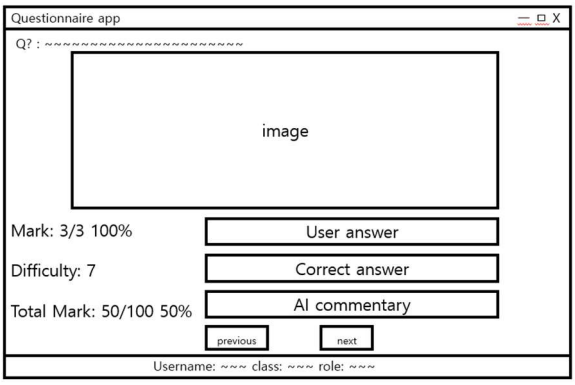
Previous version and another idea

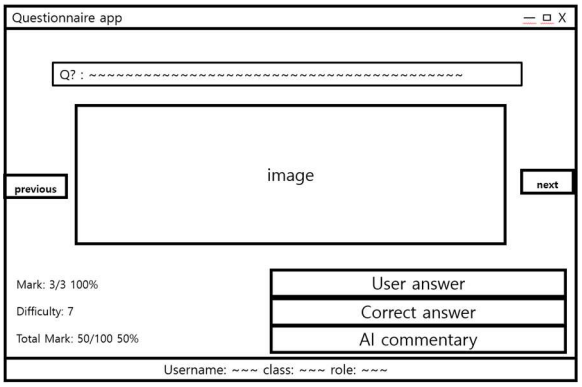


Questionnaire app (review mode)

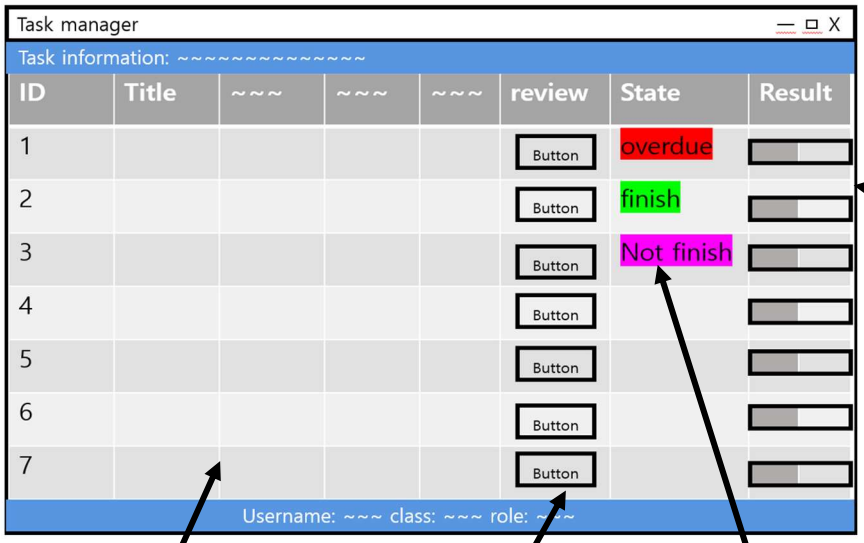


Previous version and another idea





Task Manager



Guage bar to display result.

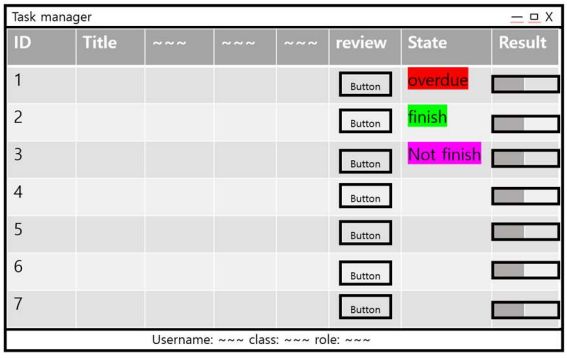
Task info

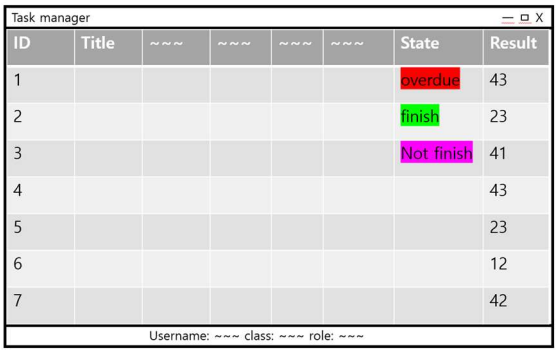
button that connects with

Questionnaire app (review mode)

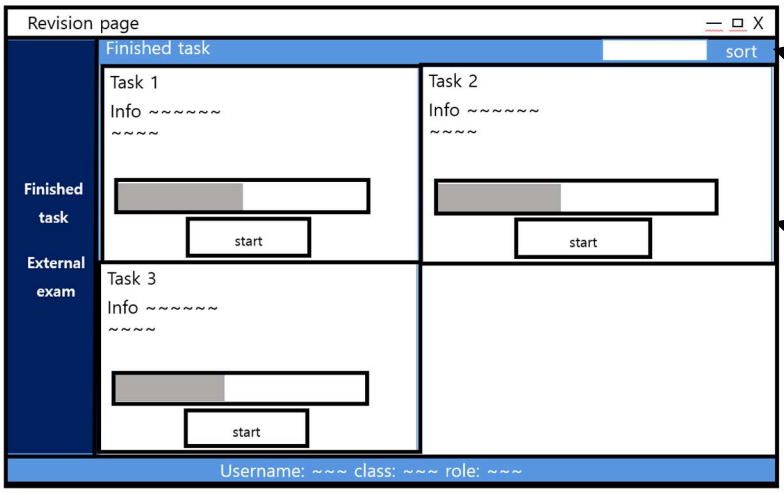
Task state shows that user finished the task on time or time passed.

Previous version and another idea





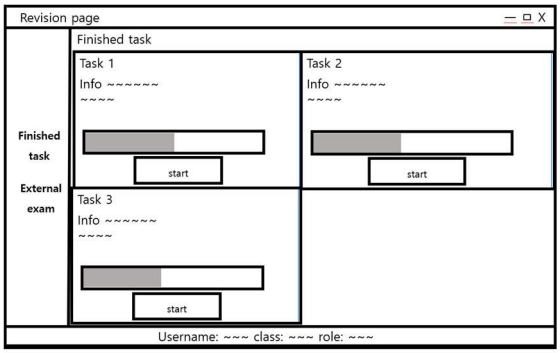
Revision page

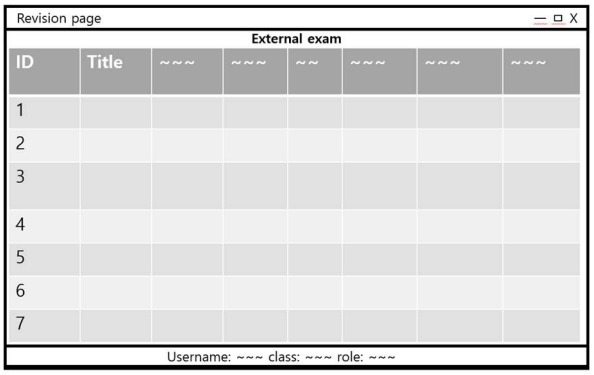


Searching bar and sort system for student

Show finished test or external examset and allow students revise with them.

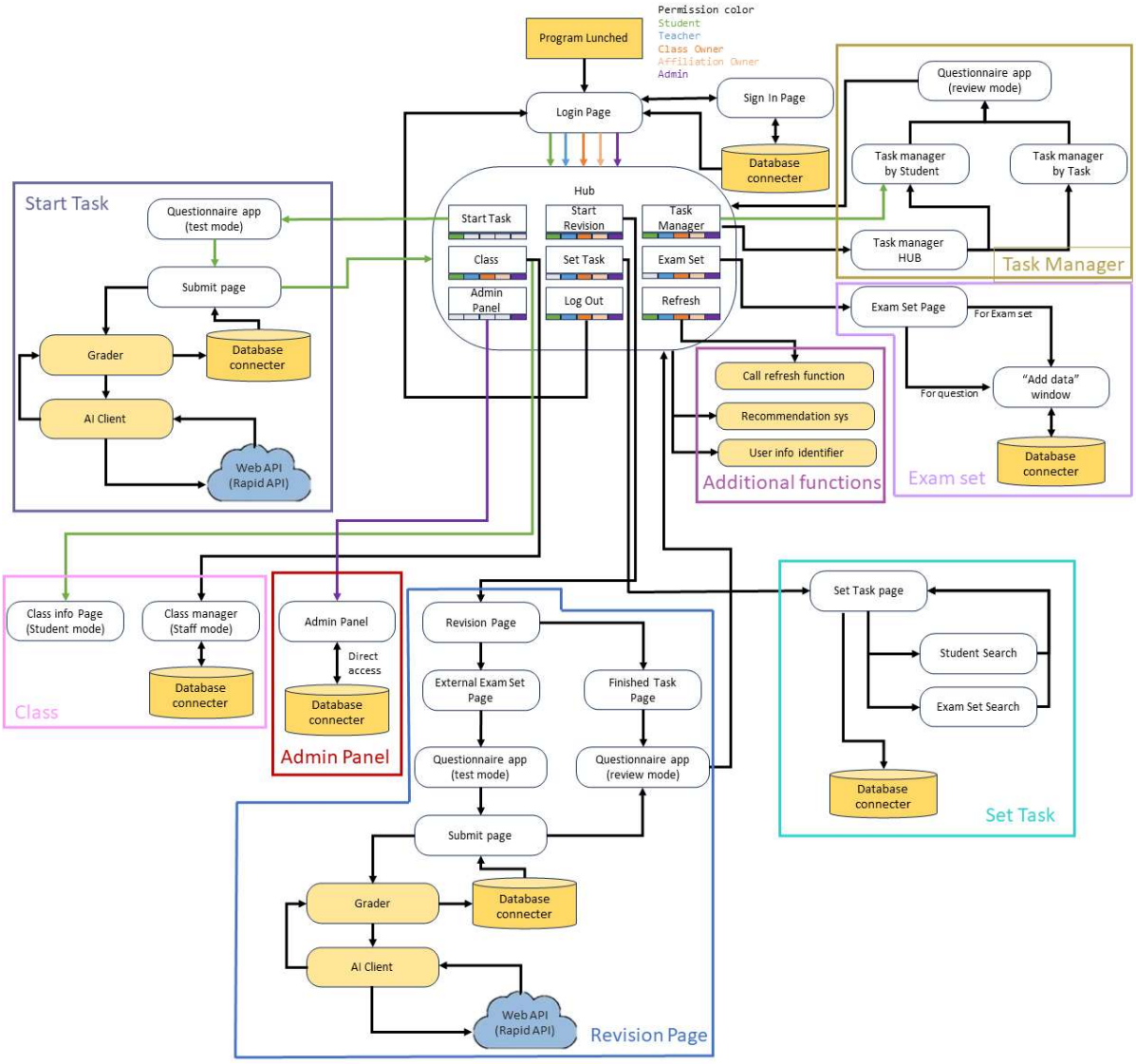
Previous version and another idea





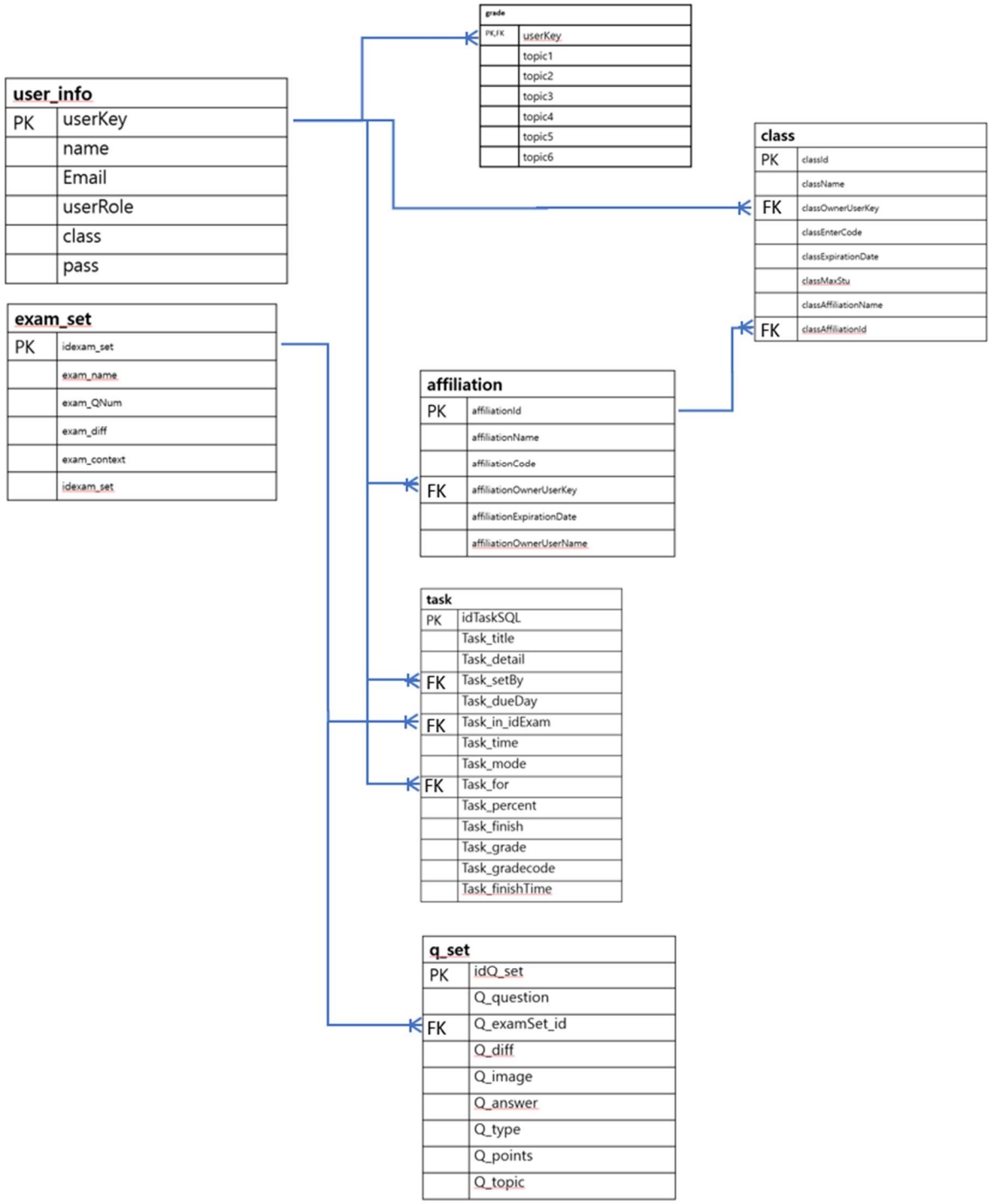
Functionality

Hierarchical UML diagram



Database Structure

Entity Relation (ER) Diagram



Data Dictionary

user_info			
Field Name	Data Type	Validation Rules	Description
userKey	int	• Primary Key • Not Null • Auto Increment	maintain a unique identifier for a user
name	varchar (255)	• Not Null	Name to be displayed in the app
Email	varchar (255)	• Not Null	Email used as ID when logging in
userRole	varchar (255)	• Not Null	save the user's permission (Admin as A, Teacher as T, Student as S, Class Owner as CO, Affiliation Owner as AO)
class	varchar (255)		class name
pass	varchar (255)	• Not Null	logging in data
classId	int	• Foreign Key	identify which class this user in

q_set			
Field Name	Data Type	Validation Rules	Description
idQ_set	int	• Primary Key • Not Null • Auto Increment	maintain a unique identifier for a question
Q_question	text (65535)	• Not Null	--
Q_examSet_id	int	• Not Null • Foreign Key	Identify the exam set to which this question belongs by ID
Q_diff	varchar (255)	• Not Null	--
Q_image	blob (65535)	• Not Null	Store images related to the vagina as binary data
Q_answer	text (65535)	• Not Null	-
Q_type	varchar (255)	• Not Null	Save the type of question by symbolizing 'e' as essay, 'w' as short subjective writing, and 's' as multiple choice question.
Q_points	varchar (255)	• Not Null	--
Q_topic	int	• Not Null	Save which of the six IGCSE physics topics this question relates to

task			
Field Name	Data Type	Validation Rules	Description
idTaskSQL	int	• Primary Key • Not Null • Auto Increment	maintain a unique identifier for a task
Task_title	varchar (255)	• Not Null	--
Task_detail	varchar (255)		--
Task_setBy	varchar (255)	• Not Null • Foreign Key	Task_setBy stores the userKey of the user who set the task.
Task_dueDay	varchar (255)	• Not Null	Compare Task_dueDay and Task_finishTime to see if the student has completed this task on time or over time
Task_in_idExam	int	• Not Null • Foreign Key	Identifies the exam_set to be used in the task
Task_time	varchar (255)	• Not Null	Set time limit
Task_mode	varchar (255)	• Not Null	for extension.
Task_for	varchar (255)	• Not Null • Foreign Key	who the task was set up for
Task_percent	varchar (255)		Indicates the mark received by the user as %
Task_finish	varchar (255)	• Not Null	If this task is not finished, save 0 if it is finished, save 1
Task_grade	varchar (255)	• Not Null	Rate user performance base on Task_percent on IGCSE grade boundaries
Task_gradecode	text (65535)		Convert and store test results into code in the format |idQ_set&user answer&correct answer from mark sheet&awarded Mark&max Mark&difficulty&AI commentary|
Task_finishTime	varchar (255)		Compare Task_dueDay and Task_finishTime to see if the student has completed this task on time or over time

grade			
Field Name	Data Type	Validation Rules	Description
userKey	int	• Primary Key • Not Null • Auto Increment • Foreign Key	identify user and extend user_info
topic1	varchar (255)	• Not Null	store predict grade for topic 1
topic2	varchar (255)	• Not Null	store predict grade for topic 2
topic3	varchar (255)	• Not Null	store predict grade for topic 3
topic4	varchar (255)	• Not Null	store predict grade for topic 4
topic5	varchar (255)	• Not Null	store predict grade for topic 5
topic6	varchar (255)	• Not Null	store predict grade for topic 6

class			
Field Name	Data Type	Validation Rules	Description
classId	int	• Primary Key • Not Null • Auto Increment	maintain a unique identifier for a class
className	varchar (255)	• Not Null	-
classOwnerUserKey	int	• Not Null • Foreign Key	Identify who the classOwner is by storing the userKey.
classEnterCode	varchar (255)	• Not Null	Save the code that students should use to join the class
classExpirationDate	varchar (255)	• Not Null	After classExpirationDate, data related to this class is deleted
classMaxStu	varchar (255)	• Not Null	Save class Max Student numbers
classAffiliationName	varchar (255)		Save the name of the affiliation
classAffiliationId	int	• Foreign Key	Identify which Affiliation this class belongs to

affiliation			
Field Name	Data Type	Validation Rules	Description
affiliationId	int	• Primary Key • Not Null • Auto Increment	maintain a unique identifier for an affiliation
affiliationName	varchar (255)	• Not Null	-
affiliationCode	varchar (255)	• Not Null	Save the code that class should use to join the affiliation
affiliationOwnerUserKey	int	• Not Null • Foreign Key	Identify who the affiliation Owner is by storing the userKey.
affiliationExpirationDate	varchar (255)	• Not Null	After the affiliation Expiration Date, the data related to this affiliation is deleted
affiliationOwnerUserName	varchar (255)	• Not Null	Save the name of the affiliation Owner

exam_set			
Field Name	Data Type	Validation Rules	Description
idexam_set	int	• Primary Key • Not Null	maintain a unique identifier for an exam_set
exam_name	varchar (255)	• Not Null	--
exam_QNum	varchar (255)	• Not Null	Save Question Count
exam_diff	varchar (255)	• Not Null	Store the difficulty of the overall examset
exam_context	varchar (255)	• Not Null	Save the additional description

Data structure

examSetData 2D array

The 2d array will collect all the examSet and Question from the database and insert them into here.

ExamSet Object1	Question Object1	Question Object2	Question Object3	....
ExamSet Object2	Question Object1	Question Object2	Question Object3	....
ExamSet Object3	Question Object1	Question Object2	Question Object3	....
ExamSet Object4	Question Object1	Question Object2	Question Object3	....
ExamSet Object5	Question Object1	Question Object2	Question Object3	....

In this 2D array the first column has an ExamSet Object, and the rest of columns has as many stores Question Objects as they make up the ExamSet.

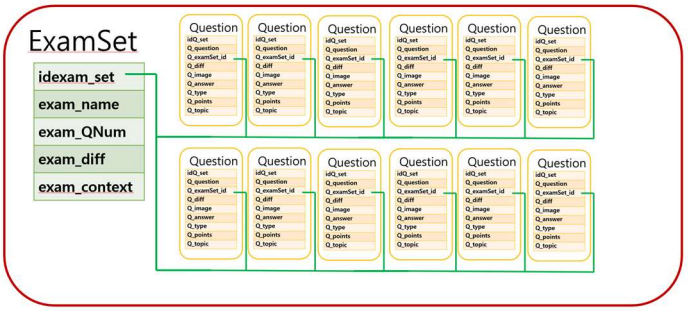
Distance Hash table(dictionary)

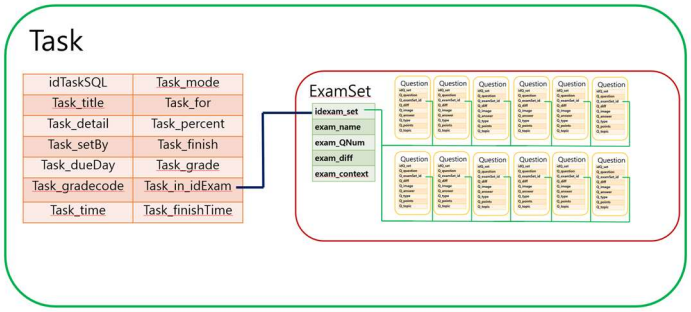
Key	value
ExamSet ID 1 →	12
ExamSet ID 2 →	32
ExamSet ID 3 →	42
ExamSet ID 4 →	34

In this hash table(dictionary), key column in one row has an ExamSet ID, and the value column has vector distance between user and the certain ExamSet.

Relationship between Task Object, ExamSet Object and Question object.

All diagrams are created by using Microsoft PowerPoint





Key Algorithms

exponential function

When a user solves a problem in the grading system, if the difference between the user's expected score and the difficulty of the problem is large, it affects the expected score more, and if the difference is small, I wanted to affect the expected score only a little. To implement this, it was necessary to use an exponential function.

```
function PowerFunction(base, power):
    if power == 1:
        return base
    else if power > 0:
        return base * PowerFunction(base, power - 1)
    else if power < 0:
        return 1.0 / PowerFunction(base, -power)
    else:
        return 1.0
```

If power is one, return base info, because power one of any base is base itself.

Call PowerFuntion itself with one less power.

When the power is positive ($b^p = b \cdot b^{p-1}$)

Call PowerFuntion. And take reciprocal itself.

When the power is positive ($b^{-p} = \frac{1}{b^p}$)

Return 1 when the power is 0($b^0 = 1$) (in this condition, else can hold only when power is 0)

KNN algorithm

It allows to calculate similarity between user and ExamSet, and recommend user a perfect ExamSet to use for revise.

Calculate similarity of two object by calculating vector difference (Euclidean distance) in 6-dimension:

```
ExamSetList = [examSet object, question Object, question Object, question Object, ……],
TopicalExpectiedGradeList = [12,34,63,41,70,42]
ExamSetDifficultyList = [0,0,0,0,0,0]
topicQuestionCountlist = [0,0,0,0,0,0]

function calculate_vector_difference (ExamSetList, TopicalExpectiedGradeList):

    for i from 1 to length(ExamSetList)
        topic = questionObject.getTopic()
        diff = 10*questionObject.getDiffculty()
        ExamSetDifficultyList[topic -1] += diff
        topicQuestionCountlist[topic – 1] += 1
    end for

    for j from 0 to length(ExamSetDifficultyList)
        ExamSetDifficultyList[j] /= topicQuestionCountlist[j]
    End for


    distance = 0.0
    for i from 0 to length(TopicalExpectiedGradeList) - 1:
        distance += (ExamSetDifficultyList [i] - TopicalExpectiedGradeList [i])^2
    distance = sqrt(distance)
    return distance
end function
```

Get topic, difficulty (change scale to compare difficulty (0~7) with expected grade(0~70), and get sum of diff list in ExamSetDifficultyList and get topicQuestionCountlist

ExamSetDifficultyList store the average difficulty of all questions by topic

Calculate Euclidean distance.

$$Euclidean\ Distance = \sqrt{\sum_{i=1}^{n}(P_i - Q_i)^2}, \qquad n = 6$$

Recommend k ExamSets with hash table(dictionary) and 2D array

```
.TopicalExpectiedGradeList = [12,34,63,41,74,42]

examSetData =[[examSet object, question Object, question Object, question Object, ……],
              [examSet object, question Object, question Object, question Object, ……],
              ~~
              ~~
              [examSet object, question Object, question Object, question Object, ……]]

function recommendExamSet(k):
    // Create an empty dictionary to store distances
    distances = {}

    for ExamSetList in examSetData:
        examSet = ExamSetList [0]
        examSetID = examSet.getIdexam_set()


        distance = calculateDistance(TopicalExpectiedGradeList, ExamSetList)
        distances[examSetID] = distance
    end for
    sortedDistances = sort(distances by value)
    recommendedExamSetIDsList = []

    for entry in getFirstKEntries(sortedDistances, k):
        recommendedExamSetIDsList.add(entry.key)
    end for

    return recommendedExamSetIDsList
```

Go through all examSetData list. Get a ExamSetList

Get an examSet Object.

Get an examSetID by calling getIdexam_set()

Calculate the distance between TopicalExpectiedGradeList and setInfo.

Store the distance in the dictionary.

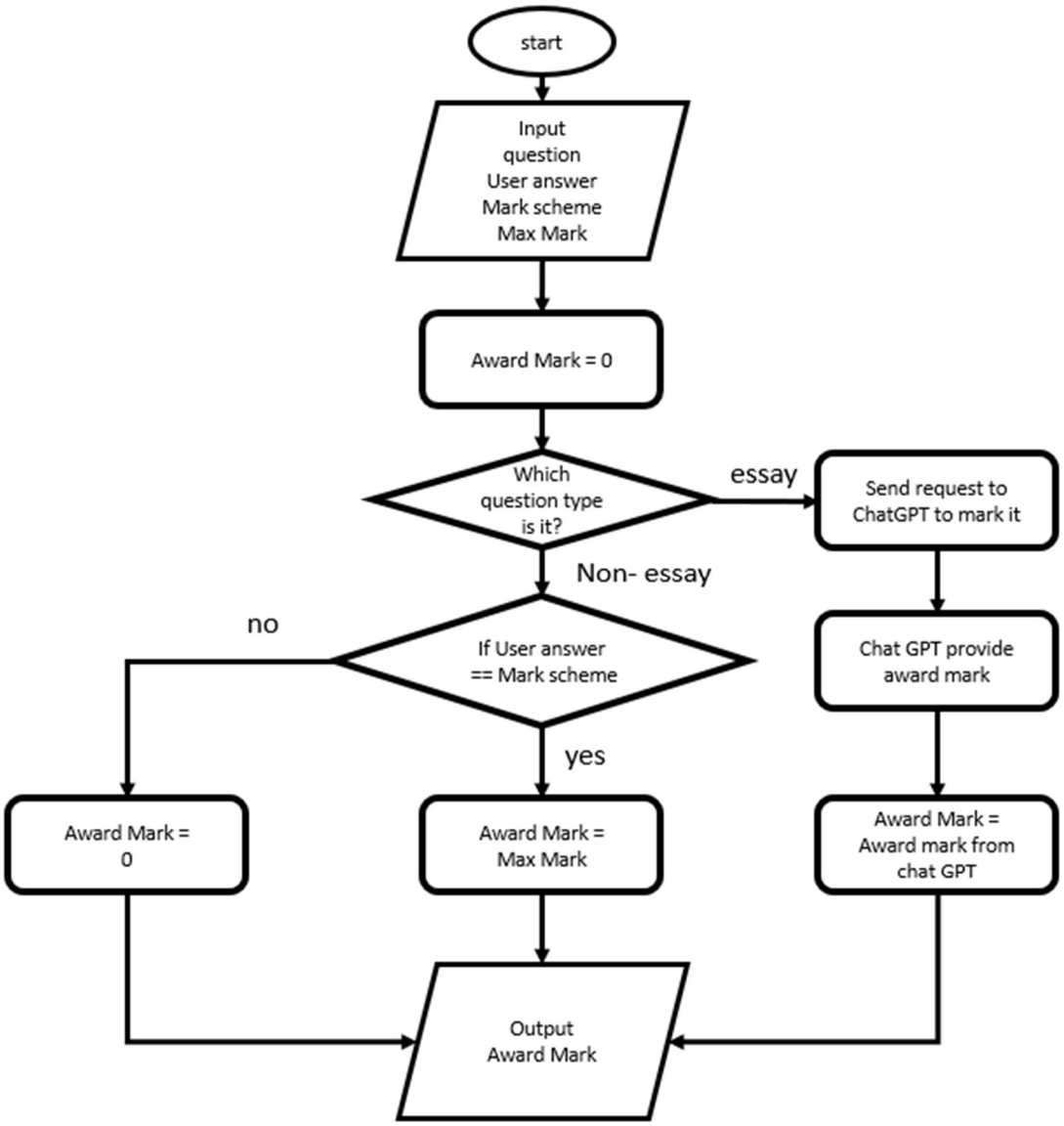
Sort the distances by their values and find k-nearest neighbors.

Get the first k entries from the sorted distances.

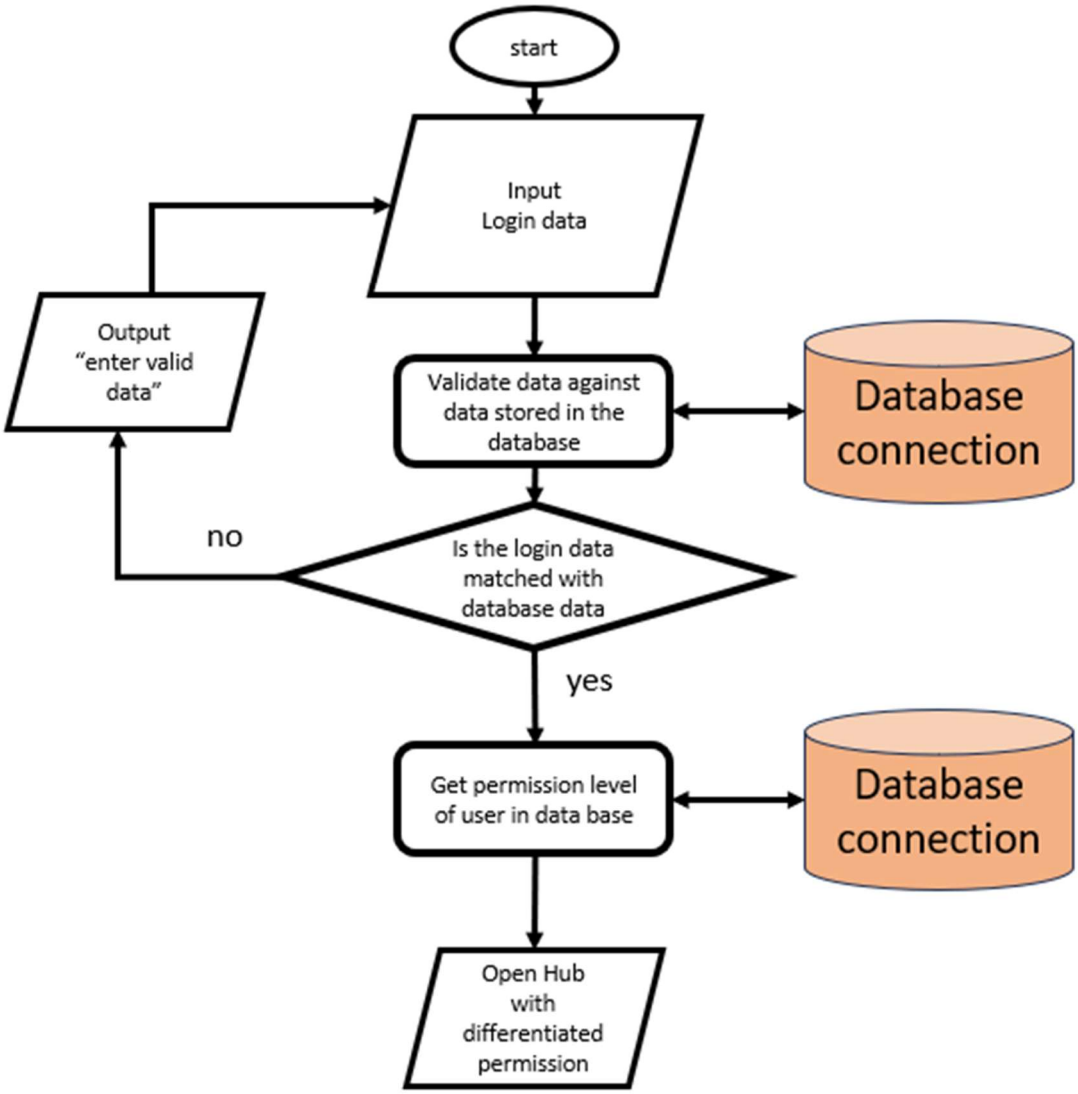
Flowcharts

All flowcharts are created by using Microsoft PowerPoint

Grading system



Login system



Test plan

Success Criterion	Test Description	Expected Outcome	Actual outcome
4	Logging in, with different permissions. and let user use different services by user's permission	When users log in with student permission, the hub allows them to use only examination service. When a user logs in with teacher permission, the hub allows them to use task setting.	When user logs in with student permission, the hub allows them to use only revision service, examination service, task manager service, class info service. When user logs in with teacher permission, the hub allows them to use rest of service (task setting service, class manage service, etc.).
1,2	Teacher can set up a task for the student, let the student access it,	Students can perform tasks with the questionnaire app. The teacher can create a task by class.	Students can perform tasks with the questionnaire app. The questionary app is an intensive GUI that keeps students focused. The teacher can create a task and assign it by a class and a student.
7	Check the auto grading system	Chat GPT grades students' answers so that the teacher doesn't have to grade them	Chat GPT receives information such as questions, student answers, mark scheme, and max mark. Then it marks and returns the award mark and justification accordingly
8	Check the automatic recommendation system	The average expected grade of the user is compared with the average difficulty of the ExamSet, and the ExamSet with the least difference is recommended.	The KNN algorithm compares the vector of the user's topical expected grade with the vector of the topical difficulty of the examSet, and then recommends the examSet, which has the least vector distance.
6	Students can try all the examSet stored in the database as well as the assigned task.	Using the revision service, students can search and try all the examSet stored in the database.	Using the revision service, students can search and try all the examSet stored in the database. And after trying examSet, show the results and check IA commentary what students missed and did well for effective revision.
5	check students' progress	Using the task manager service, the teacher can check the information of the task that the student did and each student's score in a table.	The teacher checks the score that the student received for each task with a gauge bar. Then, use the review button in the table to check the answers, mark scheme, and AI commission. The teacher can also see what tasks each student has completed.
2	allow for teachers to make examSet consist with questions	Teacher can save questions, mark scheme, max mark in database and create an examSet that brings these things together.	Teacher can save questions, mark scheme, max mark, image in database and create an examSet that brings these things together