irDevelopers.com [irDevelopers.com]

Examination System

Data Dictionary

2021-01-15

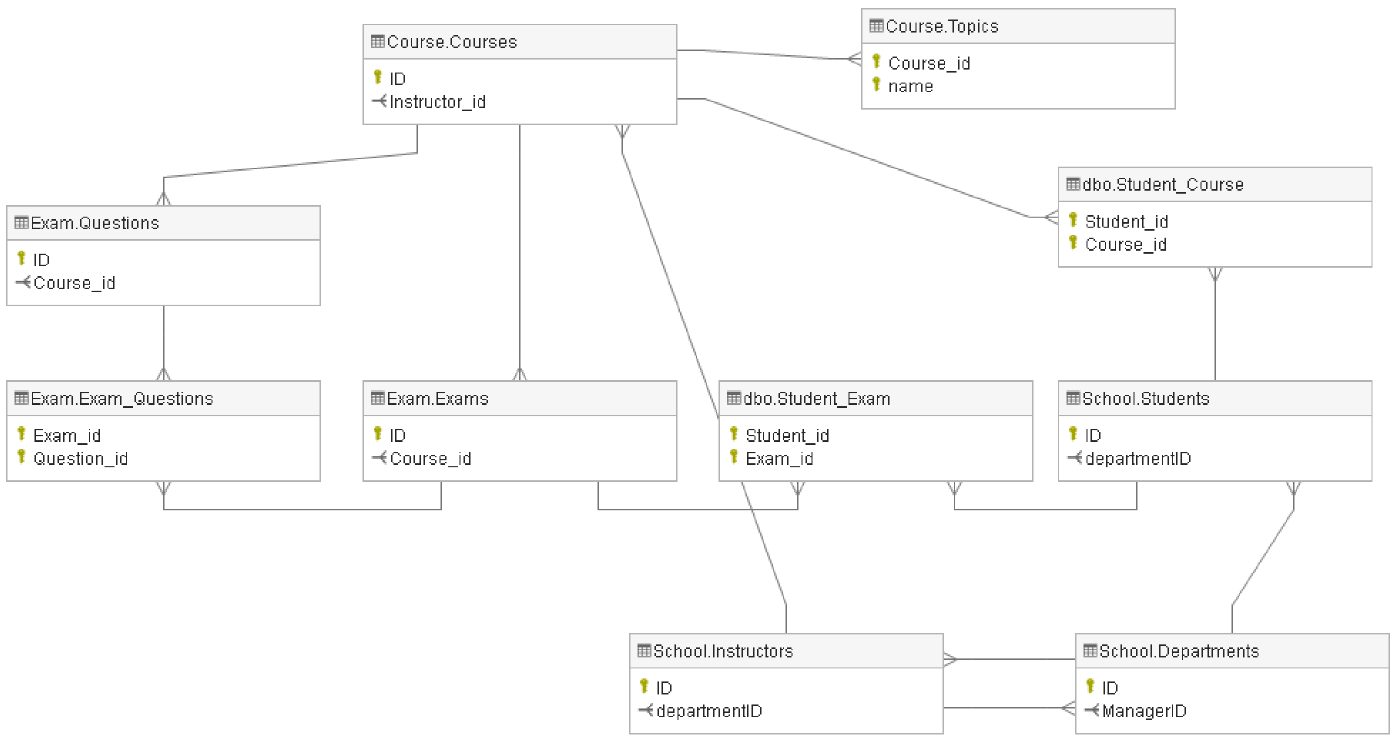
Table of contents

Legend

- 🔑 Primary key
- 🔑 Primary key disabled
- 🔑 User-defined primary key
- 🔑 Unique key
- 🔑 Unique key disabled
- 🔑 User-defined unique key
- ⚡ Active trigger
- ⚡ Disabled trigger
- ≻— Many to one relation
- ≻— User-defined many to one relation
- —≺ One to many relation
- —≺ User-defined one to many relation
- — One to one relation
- — User-defined one to one relation
- →@ Input
- @→ Output
- ↔@ Input/Output
- ▣ Uses dependency
- ▣ User-defined uses dependency
- ▣ Used by dependency
- ▣ User-defined used by dependency

Examination System

1. erd


Course.Courses
ID
Instructor_id
Course.Topics
Course_id
name
dbo.Student_Course
Student_id
Course_id
Exam.Questions
ID
Course_id
Exam.Exam_Questions
Exam_id
Question_id
Exam.Exams
ID
Course_id
dbo.Student_Exam
Student_id
Exam_id
School.Students
ID
departmentID
School.Instructors
ID
departmentID
School.Departments
ID
ManagerID

2. Other

2.1. Tables

2.1.1. Table: Course.Courses

Columns

		Name	Data type	Description / Attributes
▦	🔑	ID	int	Identity / Auto increment
▦		name	nvarchar(50)	
▦		duration	int	
▦		Instructor_id	int	Nullable References: School.Instructors

Links to

	Table	Join	Title / Name / Description
≻—	School.Instructors	Course.Courses.Instructor_id = School.Instructors.ID	FK_Courses_Instructors

Linked from

	Table	Join	Title / Name / Description
—≺	Exam.Exams	Course.Courses.ID = Exam.Exams.Course_id	FK_Exams_Courses
—≺	Exam.Questions	Course.Courses.ID = Exam.Questions.Course_id	FK_Questions_Courses
—≺	dbo.Student_Course	Course.Courses.ID = dbo.Student_Course.Course_id	FK_Student_Course_Courses
—≺	Course.Topics	Course.Courses.ID = Course.Topics.Course_id	FK_Topics_Courses

Unique keys

	Columns	Name / Description
🔑	ID	PK_Courses

2.1.2. Table: Course.Topics

Columns

		Name	Data type	Description / Attributes
		Course_id	int	**References**: Course.Courses
		name	nvarchar(50)	

Links to

	Table	Join	Title / Name / Description
	Course.Courses	**Course.Topics**.Course_id = Course.Courses.ID	FK_Topics_Courses

Unique keys

	Columns	Name / Description
	Course_id, name	PK_Topics

2.1.3. Table: dbo.Student_Course

Columns

		Name	Data type	Description / Attributes
▥	🔑	Student_id	int	**References**: School.Students
▥	🔑	Course_id	int	**References**: Course.Courses

Links to

	Table	Join	Title / Name / Description
≻—	Course.Courses	**dbo.Student_Course.**Course_id = Course.Courses.ID	FK_Student_Course_Courses
≻—	School.Students	**dbo.Student_Course.**Student_id = School.Students.ID	FK_Student_Course_Students

Unique keys

	Columns	Name / Description
🔑	Student_id, Course_id	PK_Student_Course

2.1.4. Table: dbo.Student_Exam

Columns

		Name	Data type	Description / Attributes
▦	🔑	Student_id	int	**References**: School.Students
▦	🔑	Exam_id	int	**References**: Exam.Exams
▦		answers	nvarchar(50)	**Nullable**
▦		results	int	**Nullable**

Links to

	Table	Join	Title / Name / Description
⤚	Exam.Exams	**dbo.Student_Exam**.Exam_id = Exam.Exams.ID	FK_Student_Exam_Exams
⤚	School.Students	**dbo.Student_Exam**.Student_id = School.Students.ID	FK_Student_Exam_Students

Unique keys

	Columns	Name / Description
🔑	Student_id, Exam_id	PK_Student_Exam

2.1.5. Table: dbo.Users

Columns

		Name	Data type	Description / Attributes
▦	🔑	ID	int	Identity / Auto increment
▦		Name	nvarchar(20)	
▦		Password	varchar(45)	
▦		role	varchar(20)	Nullable

Unique keys

	Columns	Name / Description
🔑	ID	PK_Users

2.1.6. Table: Exam.Exam_Questions

Columns

		Name	Data type	Description / Attributes
▦	🔑	Exam_id	int	**References**: Exam.Exams
▦	🔑	Question_id	int	**References**: Exam.Questions

Links to

	Table	Join	Title / Name / Description
⤚	Exam.Exams	**Exam.Exam_Questions.**Exam_id = Exam.Exams.ID	FK_Exam_Questions_Exams
⤚	Exam.Questions	**Exam.Exam_Questions.**Question_id = Exam.Questions.ID	FK_Exam_Questions_Questions

Unique keys

	Columns	Name / Description
🔑	Exam_id, Question_id	PK_Exam_Questions

2.1.7. Table: Exam.Exams

Columns

		Name	Data type	Description / Attributes
▦	🔑	ID	int	**Identity / Auto increment**
▦		ModelAnswer	nvarchar(50)	**Nullable**
▦		MCQ_Count	int	
▦		TF_Count	int	
▦		Course_id	int	**Nullable** **References**: Course.Courses

Links to

	Table	Join	Title / Name / Description
≻—	Course.Courses	**Exam.Exams.**Course_id = Course.Courses.ID	FK_Exams_Courses

Linked from

	Table	Join	Title / Name / Description
—≺	Exam.Exam_Questions	**Exam.Exams.**ID = Exam.Exam_Questions.Exam_id	FK_Exam_Questions_Exams
—≺	dbo.Student_Exam	**Exam.Exams.**ID = dbo.Student_Exam.Exam_id	FK_Student_Exam_Exams

Unique keys

	Columns	Name / Description
🔑	ID	PK_Exams

2.1.8. Table: Exam.Questions

Columns

		Name	Data type	Description / Attributes
	🔑	ID	int	Identity / Auto increment
		question_Statment	nvarchar(MAX)	
		type	nvarchar(5)	
		modelAnswer	nvarchar(5)	
		option1	nvarchar(MAX)	Nullable
		option2	nvarchar(MAX)	Nullable
		option3	nvarchar(MAX)	Nullable
		option4	nvarchar(MAX)	Nullable
		Course_id	int	Nullable References: Course.Courses

Links to

	Table	Join	Title / Name / Description
	Course.Courses	Exam.Questions.Course_id = Course.Courses.ID	FK_Questions_Courses

Linked from

	Table	Join	Title / Name / Description
	Exam.Exam_Questions	Exam.Questions.ID = Exam.Exam_Questions.Question_id	FK_Exam_Questions_Questions

Unique keys

	Columns	Name / Description
🔑	ID	PK_Questions

2.1.9. Table: School.Departments

Columns

		Name	Data type	Description / Attributes
		ID	int	**Identity / Auto increment**
		name	nvarchar(50)	
		ManagerID	int	**Nullable** **References**: School.Instructors

Links to

	Table	Join	Title / Name / Description
	School.Instructors	**School.Departments.**ManagerID = School.Instructors.ID	FK_Departments_Manager

Linked from

	Table	Join	Title / Name / Description
	School.Instructors	**School.Departments.**ID = School.Instructors.departmentID	FK_Instructors_Departments
	School.Students	**School.Departments.**ID = School.Students.departmentID	FK_Students_Departments

Unique keys

	Columns	Name / Description
	ID	PK_Departments

2.1.10. Table: School.Instructors

Columns

		Name	Data type	Description / Attributes
▦	🔑	ID	int	Identity / Auto increment
▦		name	nvarchar(50)	
▦		departmentID	int	Nullable References: School.Departments

Links to

	Table	Join	Title / Name / Description
≻—	School.Departments	School.Instructors.departmentID = School.Departments.ID	FK_Instructors_Departments

Linked from

	Table	Join	Title / Name / Description
—≺	Course.Courses	School.Instructors.ID = Course.Courses.Instructor_id	FK_Courses_Instructors
—≺	School.Departments	School.Instructors.ID = School.Departments.ManagerID	FK_Departments_Manager

Unique keys

	Columns	Name / Description
🔑	ID	PK_Instructors

2.1.11. Table: School.Students

Columns

		Name	Data type	Description / Attributes
▤	🔑	ID	int	Identity / Auto increment
▤		name	nvarchar(50)	
▤		level	int	
▤		departmentID	int	Nullable References: School.Departments

Links to

	Table	Join	Title / Name / Description
≻	School.Departments	School.Students.departmentID = School.Departments.ID	FK_Students_Departments

Linked from

	Table	Join	Title / Name / Description
≺	dbo.Student_Course	School.Students.ID = dbo.Student_Course.Student_id	FK_Student_Course_Students
≺	dbo.Student_Exam	School.Students.ID = dbo.Student_Exam.Student_id	FK_Student_Exam_Students

Unique keys

	Columns	Name / Description
🔑	ID	PK_Students

2.2. Procedures

2.2.1. Procedure: dbo.P_CorrectExam

Input/Output

	Name	Data type	Description
→@	examID	int	
→@	studentID	int	

Script

```
-- Correct an Exam
Create Proc P_CorrectExam @examID int, @studentID int
as
           declare @studentAnswer varchar(100), @modelAnswer varchar(100)

           exec P_StudentExam_SelectAnswers @examID , @studentID, @studentAnswer out
           exec P_Exams_SelectModelAnswer @examID, @modelAnswer out

           declare @t table(id int,modelAnswer char, studentAnswer char)

           insert into @t
           select modelAnswer.id ,modelAnswer.Character, studentAnswer.Character
           from split_string_to_rows(@modelAnswer) as modelAnswer join split_string_to_rows(@studentAnswer) as studentAnswer
           on modelAnswer.id = studentAnswer.id

           declare @questionCount int
           select @questionCount = count(id) from @t

           declare @result int
           select @result = ceiling(count(t1.id) * 100.0 / @questionCount) from @t t1 join @t t2 on t1.id = t2.id and
t1.modelAnswer = t2.studentAnswer

           exec P_StudentExam_UpdateResult @result, @examID , @studentID
```

2.2.2. Procedure: dbo.p_course_delete

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
--delete row from course table with id
create proc p_course_delete @id int
as
           if Exists(select Course_id from Exam.Exams where Course_id = @id)
                      return 0 --can't delete because its in Exam table
           else
           Begin
                      exec P_Topic_DeleteCourseTopics @id
                      delete from Course.Courses
                      where id=@id
                      return 1 -- deleted sucsessfully
           END
```

2.2.3. Procedure: dbo.p_courses_insert

Input/Output

	Name	Data type	Description
→@	name	varchar(50)	
→@	duration	int	
→@	ins_id	int	

Script

```
--insert name,duration and instructor_id in the course table and returns course id
create proc p_courses_insert @name varchar(50),@duration int,@ins_id int
as
	begin try
		insert into Course.Courses(name,duration,instructor_id)
		values(@name,@duration,@ins_id)
		return @@IDENTITY
	end try
	begin catch
		return 0 -- Can't insert Course
	end catch
```

2.2.4. Procedure: dbo.p_courses_select

Script

```
/************** Course Procedures ********************/

--select the data from course table
create proc p_courses_select
as
          select * from course.Courses
```

2.2.5. Procedure: dbo.p_courses_update

Input/Output

	Name	Data type	Description
→@	id	int	
→@	name	varchar(50)	
→@	duration	int	
→@	ins_id	int	

Script

```
--update name,duration and instructor_id in the course table
create proc p_courses_update @id int,@name varchar(50),@duration int ,@ins_id int
as
	begin try
		update Course.Courses
		set [name]=@name,
			duration=@duration,
			Instructor_id=@ins_id
		where id=@id
		return @id
	end try
	begin catch
		return 0
	end catch
```

2.2.6. Procedure: dbo.P_Department_delete

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
-- delete department
create Proc P_Department_delete @id int
as
          if Exists(select departmentID from School.Instructors)
          delete from School.Departments where ID = @id
```

2.2.7. Procedure: dbo.P_Department_insert

Input/Output

	Name	Data type	Description
➔@	Name	nvarchar(50)	
➔@	mangerID	int	

Script

```
--*****************Departments***********************

-- insert new department
create Proc P_Department_insert @Name nvarchar(50), @mangerID int
as
		insert into School.Departments(name, ManagerID) values(@Name, @mangerID)
```

2.2.8. Procedure: dbo.P_Department_select

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
-- select all departments
create Proc P_Department_select @id int
as
          select * from School.Departments
```

2.2.9. Procedure: dbo.P_Department_Update

Input/Output

	Name	Data type	Description
→@	id	int	
→@	Name	nvarchar(50)	
→@	mangerID	int	

Script

```
-- update department info
create Proc P_Department_Update @id int, @Name nvarchar(50), @mangerID int
as
           if EXISTS (select ID from School.Departments where ID=@id )
           BEGIN
                      update School.Departments set name = @Name, ManagerID = @mangerID where ID = @id
                      return 1
           end
           else
                      return 0
```

2.2.10. Procedure: dbo.P_Exam_Delete

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
-- delete an Exam
Create Proc P_Exam_Delete @id int
AS
		if Exists ( select Exam_id from Student_Exam where Exam_id = @id)
				return 0
		else
		Begin
				exec P_ExamQuestions_deleteExam @id
				Delete from Exam.Exams
				Where ID = @id
				return 1
		end
```

2.2.11. Procedure: dbo.P_Exam_Insert

Input/Output

	Name	Data type	Description
→@	mcq_count	int	
→@	tf_count	int	
→@	course_id	int	
→@→	examID	int	

Script

```
-- Add new Exam returns Exam ID
Create Proc P_Exam_Insert @mcq_count int, @tf_count int, @course_id int, @examID int out
AS
           insert into Exam.Exams(MCQ_Count,TF_Count,Course_id)
           Values(@mcq_count,@tf_count,@course_id)
           select @examID = @@IDENTITY
```

2.2.12. Procedure: dbo.P_Exam_Select

Script

```
/************** Exam Procedures ********************/

-- Returns all exams
Create Proc P_Exam_Select
AS
           Select * From Exam.Exams
```

2.2.13. Procedure: dbo.P_Exam_Select_With_Course_id

Input/Output

	Name	Data type	Description
→@	course_id	int	

Script

```
-- Returns all exams related to a specific course
Create Proc P_Exam_Select_With_Course_id @course_id int
AS
          Select ID From Exam.Exams where Course_id = @course_id
```

2.2.14. Procedure: dbo.P_Exam_Update

Input/Output

	Name	Data type	Description
→@	id	int	
→@	mcq_count	int	
→@	tf_count	int	
→@	course_id	int	

Script

```
-- Update an Exam
Create Proc P_Exam_Update @id int, @mcq_count int, @tf_count int, @course_id int
AS
           if Exists( select Exam_id from Student_Exam where Exam_id = @id)
                      return 0
           else
           begin
                      Update Exam.Exams
                      set MCQ_Count = @mcq_count,
                      TF_Count = @tf_count,
                      Course_id = @course_id
                      Where ID = @id
                      -- Call Proc to delete from Exam_Question where exam = exam
                      exec P_ExamQuestions_deleteExam @id
                      -- regenerateExam @exam_id int, @mcq_count in, @tf_count int
                       exec P_reGenerateExam @id, @mcq_count, @tf_count, @course_id
                      return 1
           end
```

2.2.15. Procedure: dbo.P_Exam_UpdateModelAnswer

Input/Output

	Name	Data type	Description
→@	examID	int	

Script

```
-- Generate model answer for exam
Create Proc P_Exam_UpdateModelAnswer @examID int
as
           declare c1 Cursor
           for select modelAnswer from Exam.Exam_Questions eq join Exam.Questions q on eq.Question_id = q.ID  where Exam_id = 
@examID
           for read only

           declare @Answer char,@ModelAnswers varchar(100)=''
           open c1
           fetch c1 into @Answer
           while @@FETCH_STATUS=0
                      begin
                                 set @ModelAnswers=concat(@ModelAnswers,@Answer)
                                 fetch c1 into @Answer
                      end

           update Exam.Exams set ModelAnswer = @ModelAnswers where ID = @examID
           close c1
           deallocate c1
```

2.2.16. Procedure: dbo.P_ExamQuestions_deleteExam

Input/Output

	Name	Data type	Description
→@	exam_id	int	

Script

```
/************** Exam_Question *********************/
-- delete all references for a specific exam
create Proc P_ExamQuestions_deleteExam @exam_id int
AS
           delete from Exam.Exam_Questions where Exam_id = @exam_id
```

2.2.17. Procedure: dbo.P_ExamQuestions_Insert

Input/Output

	Name	Data type	Description
→@	exam_id	int	
→@	mcq_count	int	
→@	tf_count	int	
→@	course_id	int	

Script

```
-- insert questions to exam
Create Proc P_ExamQuestions_Insert @exam_id int, @mcq_count int, @tf_count int, @course_id int
AS
           insert into Exam.Exam_Questions
                      exec P_Question_SelectMCQ @mcq_count, @course_id, @exam_id
           insert into Exam.Exam_Questions
                      exec P_Question_SelectTF @tf_count, @course_id, @exam_id
```

2.2.18. Procedure: dbo.P_Exams_SelectModelAnswer

Input/Output

	Name	Data type	Description
→@	exam_id	int	
→@→	answers	varchar(50)	

Script

```
-- return model anwser for  specific exam
Create Proc P_Exams_SelectModelAnswer @exam_id int, @answers varchar(50) out
AS
           select @answers = ModelAnswer from Exam.Exams where ID = @exam_id
```

2.2.19. Procedure: dbo.P_GenerateExam

Input/Output

	Name	Data type	Description
→@	courseID	int	
→@	TF_Count	int	
→@	MCQ_Count	int	

Script

```
-- Generate New Exam
Create Proc P_GenerateExam @courseID int, @TF_Count int, @MCQ_Count int
as
           declare @examID int

           exec P_Exam_Insert @mcq_count,@tf_count,@courseID,@examID out

           exec P_ExamQuestions_Insert @examID, @MCQ_Count, @TF_Count,@courseID

           exec P_Exam_UpdateModelAnswer @examID
           select @examID as [Exam ID]
```

2.2.20. Procedure: dbo.p_Instructor_Delete

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
--------------------------------------------------------------------------------------------------------
Create proc p_Instructor_Delete  @id int
as
          if EXISTS (select ManagerID from School.Departments where ManagerID=@id )
          begin
                    update School.Departments
                    set ManagerID =NULL
                    Where ManagerID = @id
          end

          delete from School.Instructors
          where ID = @id
          return 1
```

2.2.21. Procedure: dbo.p_Instructors_Insert

Input/Output

	Name	Data type	Description
→@	name	nvarchar(50)	
→@	departmentID	int	

Script

```
/****************Instructor*******************/

create proc p_Instructors_Insert @name nvarchar(50),@departmentID int
as
           INSERT INTO School.Instructors VALUES(@name,@departmentID)
```

2.2.22. Procedure: dbo.p_Instructors_Select

Script

```
------------------------------------------------------------------------------------------------------------
create proc p_Instructors_Select
as
          select * from School.Instructors
------------------------------------------------------------------------------------------------------------
```

2.2.23. Procedure: dbo.p_Instructors_Update

Input/Output

	Name	Data type	Description
→@	instructorID	int	
→@	departmentID	int	
→@	instructorName	nvarchar(50)	

Script

```
----------------------------------------------------------------------------------------------------------------

create proc p_Instructors_Update @instructorID int,@departmentID int,@instructorName nvarchar(50)
as
           if EXISTS (select ID from School.Instructors where ID=@instructorID )
           BEGIN
                      update School.Instructors set departmentID=@departmentID, name=@instructorName
                      where ID = @instructorID
                      return 1
           END
           ELSE
                      return 0
```

2.2.24. Procedure: dbo.p_Instructors_UpdateDepartmentID

Input/Output

	Name	Data type	Description
→@	instructorID	int	
→@	departmentID	int	

Script

```
create proc p_Instructors_UpdateDepartmentID @instructorID int, @departmentID int
as
           if EXISTS (select ID from School.Instructors where ID=@instructorID )
           BEGIN
                      update School.Instructors set departmentID=@departmentID where ID=@instructorID
                      return 1
           END
           ELSE
                      return 0
```

2.2.25. Procedure: dbo.p_Instructors_UpdateName

Input/Output

	Name	Data type	Description
→@	instructorID	int	
→@	name	nvarchar(50)	

Script

```
----------------------------------------------------------------------------------------------------------------

create proc p_Instructors_UpdateName @instructorID int,@name nvarchar(50)
as
           if EXISTS (select ID from School.Instructors where ID=@instructorID )
           BEGIN
                      update School.Instructors set name=@name where ID=@instructorID
                      return 1
           END
           ELSE
                      return 0
```

2.2.26. Procedure: dbo.P_Question_delete

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
-- Procedure to delete a Question returns 1 if sucsess, 0 if exists in Exam_Questions
Create Proc P_Question_delete @id int
AS
	if Exists(select Question_id from Exam.Exam_Questions where Question_id = @id)
		return 0
	else
	BEGIN
		Delete from [Exam].Questions
		where ID = @id
		return 1
	end
```

2.2.27. Procedure: dbo.P_Question_insertMCQ

Input/Output

	Name	Data type	Description
→@	question_statment	nvarchar(50)	
→@	model_answer	nchar(2)	
→@	option1	nvarchar(50)	
→@	option2	nvarchar(50)	
→@	option3	nvarchar(50)	
→@	option4	nvarchar(50)	
→@	course_id	int	

Script

```
-- Procedure to inserts a MCQ Question
Create Proc P_Question_insertMCQ  @question_statment nvarchar(50), @model_answer nchar(2), @option1 nvarchar(50), @option2
nvarchar(50), @option3 nvarchar(50), @option4 nvarchar(50), @course_id int
AS
           Insert into [Exam].Questions
           values(@question_statment,'MCQ',@model_answer,@option1,@option2,@option3,@option4,@course_id)
```

2.2.28. Procedure: dbo.P_Question_insertTF

Input/Output

	Name	Data type	Description
→@	question_statment	nvarchar(50)	
→@	model_answer	nchar(2)	
→@	course_id	int	

Script

```
-- Procedure to inserts a True-False Question
Create Proc P_Question_insertTF  @question_statment nvarchar(50), @model_answer nchar(2), @course_id int
AS
           Insert into [Exam].Questions(question_Statment,[type], modelAnswer,Course_id)
           values(@question_statment,'TF',@model_answer,@course_id)
```

2.2.29. Procedure: dbo.P_Question_Select

Script

```
/******************* Question Procedures ******************/

-- Procedure returns all Questions
Create Proc P_Question_Select
AS
           Select * From Exam.Questions
```

2.2.30. Procedure: dbo.P_Question_SelectMCQ

Input/Output

	Name	Data type	Description
→@	mcq_count	int	
→@	course_id	int	
→@	exam_id	int	

Script

```
-- Procedure returns all MCQ Questions
Create Proc P_Question_SelectMCQ @mcq_count int ,@course_id int, @exam_id int
AS
           Select top(@mcq_count)@exam_id, ID
           From Exam.Questions
           Where type ='MCQ' AND Course_id = @course_id
           order by newid()
```

2.2.31. Procedure: dbo.P_Question_SelectTF

Input/Output

	Name	Data type	Description
→@	tf_count	int	
→@	course_id	int	
→@	exam_id	int	

Script

```
-- Procedure returns all True-False Questions
Create Proc P_Question_SelectTF @tf_count int, @course_id int, @exam_id int
AS
           Select top(@tf_count)@exam_id, ID
           From Exam.Questions
           Where type ='TF' AND Course_id = @course_id
           order by newid()
```

2.2.32. Procedure: dbo.P_Question_Update

Input/Output

Name		Data type	Description
→@	id	int	
→@	question_statment	nvarchar(50)	
→@	type	nvarchar(10)	
→@	model_answer	nchar(2)	
→@	option1	nvarchar(50)	
→@	option2	nvarchar(50)	
→@	option3	nvarchar(50)	
→@	option4	nvarchar(50)	
→@	course_id	int	

Script

```sql
-- Procedure to update a question using its id
-- returns 1 if update sucsess, 0 if question Exists in Exam.
Create Proc P_Question_Update @id int, @question_statment nvarchar(50), @type nvarchar(10), @model_answer nchar(2), @option1
nvarchar(50), @option2 nvarchar(50), @option3 nvarchar(50), @option4 nvarchar(50), @course_id int
AS
           if Exists(select Question_id from Exam.Exam_Questions where Question_id = @id)
                      return 0
           else
           BEGIN
                      Update [Exam].Questions
                      set question_Statment = @question_statment,
                                 type = @type,
                                 modelAnswer = @model_answer,
                                 option1 = @option1,
                                 option2 = @option2,
                                 option3 = @option3,
                                 option4 = @option4,
                                 Course_id = @course_id
                      Where ID = @id
                      return 1
           END
```

2.2.33. Procedure: dbo.P_reGenerateExam

Input/Output

	Name	Data type	Description
→@	examID	int	
→@	TF_Count	int	
→@	MCQ_Count	int	
→@	Course_id	int	

Script

```
-- reGenerate exam and update it in exams table
-- input(ExamID, TF_Count, MCQ_Count)
Create Proc P_reGenerateExam @examID int, @TF_Count int, @MCQ_Count int, @Course_id int
as
           exec P_ExamQuestions_Insert @examID, @MCQ_Count, @TF_Count,@Course_id
           exec P_Exam_UpdateModelAnswer @examID
           select @examID as [Exam ID]
```

2.2.34. Procedure: dbo.P_Report_GetCourseTopics

Input/Output

	Name	Data type	Description
→@	courseID	int	

Script

```
create Proc  P_Report_GetCourseTopics @courseID int
as
           select c.name as [course name], t.name as [topic name] from Course.Courses c join Course.Topics t on t.Course_id =
c.ID where c.ID = @courseID
```

2.2.35. Procedure: dbo.P_Report_GetExam

Input/Output

	Name	Data type	Description
→@	examID	int	

Script

```
-- Get Exam
Create Proc P_Report_GetExam @examID int
as
           select q.[type], q.ID, q.question_Statment, q.option1, option2, option3, option4
           from Exam.Exam_Questions eq join Exam.Questions q on eq.Question_id = q.ID
           where eq.Exam_id = @examID
```

2.2.36. Procedure: dbo.P_Report_GetExamWithStudentAnswers

Input/Output

	Name	Data type	Description
→@	examID	int	
→@	studentID	int	

Script

```
-- Get Exam with Student Answers
Create Proc P_Report_GetExamWithStudentAnswers @examID int, @studentID int
as
	declare @studentAnswer varchar(100)
	set @studentAnswer = (select answers  from Student_Exam where Exam_id = @examID and Student_id = @studentID)

	select [type], main.ID, question_Statment, option1, option2, option3, option4, [Character] as [student answer]
from
	(
	select q.[type], q.ID, q.question_Statment, q.option1, option2, option3, option4, ROW_NUMBER() OVER(ORDER BY
(SELECT NULL)) AS rownum
	from Exam.Exam_Questions eq join Exam.Questions q on eq.Question_id = q.ID
	where eq.Exam_id = @examID
	) as main
	join split_string_to_rows(@studentAnswer) sa on sa.id = main.rownum
```

2.2.37. Procedure: dbo.P_Report_GetInstructorCourses

Input/Output

	Name	Data type	Description
→@	instructorID	int	

Script

```
create Proc  P_Report_GetInstructorCourses @instructorID int
as
          select i.name as [instructor name], c.name as [course name], count(sc.Student_id) as [student count] from
Course.Courses c join Student_Course sc on sc.Course_id = c.ID
          join School.Instructors i on c.Instructor_id = i.ID where i.ID = @instructorID group by c.ID,i.name,c.name
```

2.2.38. Procedure: dbo.P_Report_GetStudentGrades

Input/Output

	Name	Data type	Description
→@	studentID	int	

Script

```
create Proc  P_Report_GetStudentGrades @studentID int
as
           select s.name as [student name], c.name as [course name], se.results as grade from School.Students s join
Student_Exam se on se.Student_id = s.ID
           join Exam.Exams e on se.Exam_id = e.ID join Course.Courses c on e.Course_id = c.ID
           where se.Student_id = @studentID
```

2.2.39. Procedure: dbo.P_Report_GetStudentsInfo

Input/Output

	Name	Data type	Description
→@	departmentID	int	

Script

```
------------------------------------------------------------
create Proc P_Report_GetStudentsInfo @departmentID int
as
           select s.name, s.[level], d.name
           from School.Students s join School.Departments d on s.departmentID = d.ID where departmentID = @departmentID
```

2.2.40. Procedure: dbo.P_StudentExam_DeleteStudent

Input/Output

	Name	Data type	Description
→@	student_id	int	

Script

```
/****************** Student_Exam *****************/
-- Delete All student Exams
Create Proc P_StudentExam_DeleteStudent @student_id int
AS
	delete from Student_Exam where Student_id = @student_id
```

2.2.41. Procedure: dbo.P_StudentExam_InsertAnswer

Input/Output

	Name	Data type	Description
→@	examID	int	
→@	studentID	int	
→@	answers	nvarchar(100)	

Script

```
-- insert student answer
Create Proc P_StudentExam_InsertAnswer @examID int, @studentID int, @answers nvarchar(100)
AS
           insert into Student_Exam(Exam_id, Student_id, answers) values(@examID, @studentID, @answers)
```

2.2.42. Procedure: dbo.P_StudentExam_SelectAnswers

Input/Output

	Name	Data type	Description
→@	examID	int	
→@	studentID	int	
→@→	answers	varchar(50)	

Script

```
-- select student answers for Exam
Create Proc P_StudentExam_SelectAnswers @examID int, @studentID int, @answers varchar(50) out
AS
           select @answers = answers from Student_Exam where Exam_id = @examID and Student_id = @studentID
```

2.2.43. Procedure: dbo.P_StudentExam_UpdateResult

Input/Output

	Name	Data type	Description
→@	result	int	
→@	examID	int	
→@	studentID	int	

Script

```
-- update student result
Create proc P_StudentExam_UpdateResult @result int, @examID int, @studentID int
AS
           update Student_Exam set results = @result where Exam_id = @examID and Student_id = @studentID
```

2.2.44. Procedure: dbo.p_students_delete

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
-- delete Student
create proc p_students_delete @id int
as
           -- Student_Exam, Student_Course
           if Exists(select Student_id from Student_Exam)
                      exec P_StudentExam_DeleteStudent @id
           delete from School.Students
           where id=@id
```

2.2.45. Procedure: dbo.p_students_insert

Input/Output

	Name	Data type	Description
→@	name	varchar(50)	
→@	level	int	
→@	depID	int	

Script

```
--insert new student
create proc p_students_insert @name varchar(50),@level int ,@depID int
as
           insert into school.students values(@name,@level,@depID)
```

2.2.46. Procedure: dbo.p_students_select

Script

```
/************** stored procedure for student **********************/

--select the data from student table
create proc p_students_select
as
	select * from school.students
```

2.2.47. Procedure: dbo.p_stuents_update

Input/Output

	Name	Data type	Description
→@	id	int	
→@	name	varchar(50)	
→@	level	int	
→@	depID	int	

Script

```
--updatet name,level and departmentr_id in the students table
create proc p_stuents_update @id int,@name varchar(50),@level int ,@depID int
as
           begin try
                      update School.Students
                      set [name]=@name,
                                 [level]=@level,
                                 departmentID=@depID
                      where id=@id
                      return 1 -- success
           end try
           begin catch
                      return 0 --fail
           end catch
```

2.2.48. Procedure: dbo.P_submitAnswers

Input/Output

	Name	Data type	Description
⇢@	examID	int	
⇢@	studentID	int	
⇢@	answers	nvarchar(100)	

Script

```sql
-- save student exam answers
-- input(examID, studentID, answers)
Create Proc P_submitAnswers @examID int, @studentID int, @answers nvarchar(100)
as
           exec P_StudentExam_InsertAnswer @examID, @studentID, @answers
           exec P_CorrectExam @examID, @studentID
```

2.2.49. Procedure: dbo.P_Topic_DeleteCourseTopics

Input/Output

	Name	Data type	Description
→@	course_id	int	

Script

```
-- delete all topics related to a course
Create proc P_Topic_DeleteCourseTopics @course_id int
AS
           delete from Course.Topics where Course_id = @course_id
```

2.2.50. Procedure: dbo.p_topic_insert

Input/Output

	Name	Data type	Description
→@	id	int	
→@	name	varchar(50)	

Script

```
--insert new topic
create proc p_topic_insert @id int,@name varchar(50)
as
           begin try
                      insert into Course.Topics values(@id ,@name )
                      return 1 -- sucsess
           end try
           begin catch
                      return 0 -- failed
           end catch
```

2.2.51. Procedure: dbo.p_topic_select

Script

```
/***stored procedure for topic ***/

--select the data from topic table
create proc p_topic_select
as
           select * from course.Topics
```

2.2.52. Procedure: dbo.p_topic_selectCourseTopics

Input/Output

	Name	Data type	Description
→@	course_id	int	

Script

```
-- get all topics related to course
create proc p_topic_selectCourseTopics @course_id int
as
           select * from course.Topics where Course_id = @course_id
```

2.2.53. Procedure: dbo.p_topics_delete

Input/Output

	Name	Data type	Description
→@	id	int	
→@	name	varchar(50)	

Script

```
--delete topic depending on id and name
create proc p_topics_delete @id int,@name varchar(50)
as
	delete from course.topics where Course_id = @id and [name] = @name
```

2.2.54. Procedure: dbo.p_topics_update

Input/Output

	Name	Data type	Description
→@	oldId	int	
→@	newId	int	
→@	oldname	varchar(50)	
→@	newname	varchar(50)	

Script

```
-- update name of the topic
-- IMPORTANT:- Save the old Name and old ID in the UI before Updating
create proc p_topics_update @oldId int, @newId int,@oldname varchar(50),@newname varchar(50)
as
          update Course.Topics
          set [name] = @newname,
                    Course_id = @newId
          where course_id=@oldId and name = @oldname
```

2.3. Functions

2.3.1. Function: dbo.split_string_to_rows

Input/Output

	Name	Data type	Description
	Returns	table type	
	string	varchar(100)	

Script

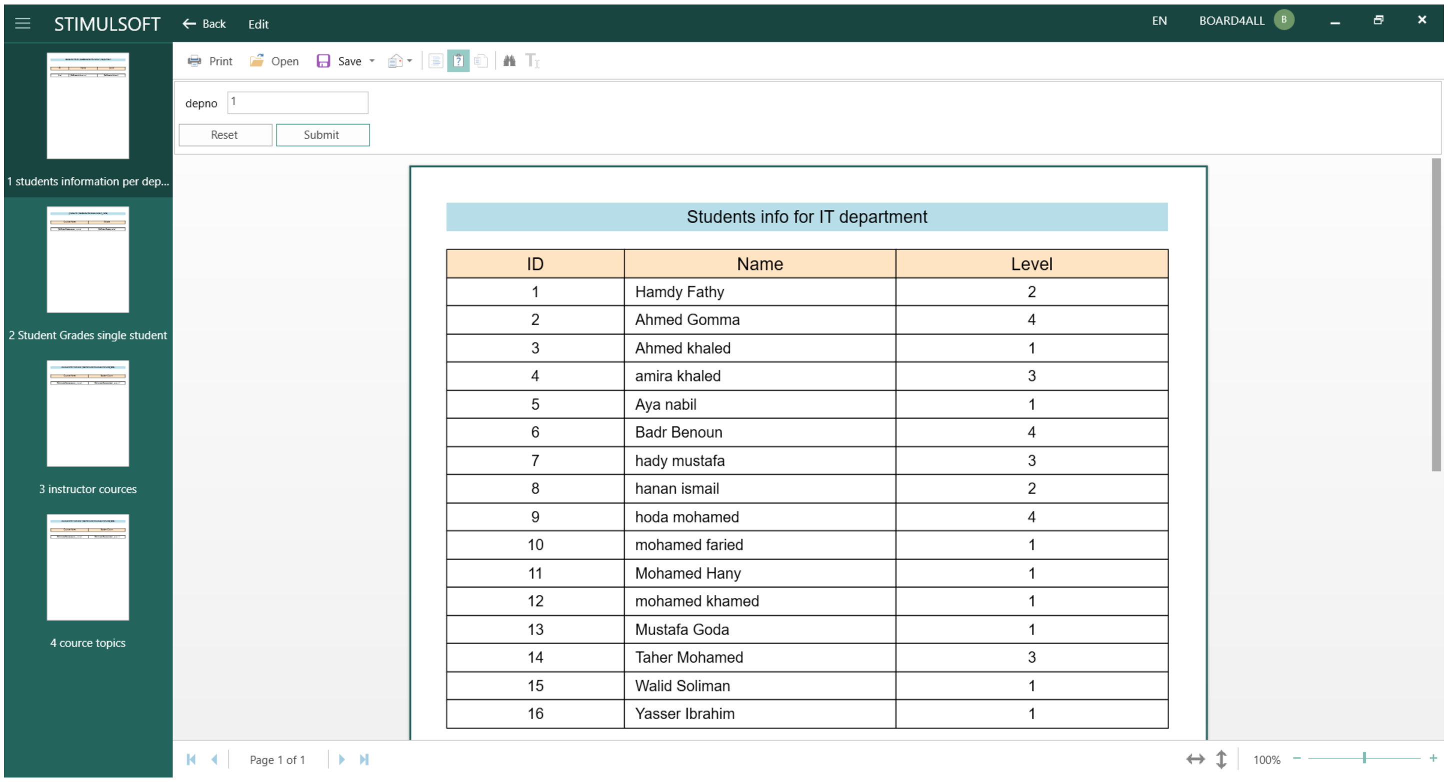
```
-- take string like 'ABCDE' and return 1 column with each char in row
Create function split_string_to_rows(@string varchar(100))
returns table
as
return
(
           WITH CTE AS
           (
                      SELECT
                                 1 as CharacterPosition,
                                 SUBSTRING(@string,1,1) as Character
                      UNION ALL
                      SELECT
                                 CharacterPosition + 1,
                                 SUBSTRING(@string,CharacterPosition + 1,1)
                      FROM
                                 CTE
                      WHERE CharacterPosition < LEN(@string)
           )
           SELECT CharacterPosition as id,Character FROM CTE
)
```

Reports:

We designed stored procedure for every report which take the required input and return the needed data

The report itself is designed by a tool from: https://www.stimulsoft.com/ , and the report can be saved to multiple formats (pdf, Word, Excel, images, HTML)

Report that returns the students information according to Department No parameter

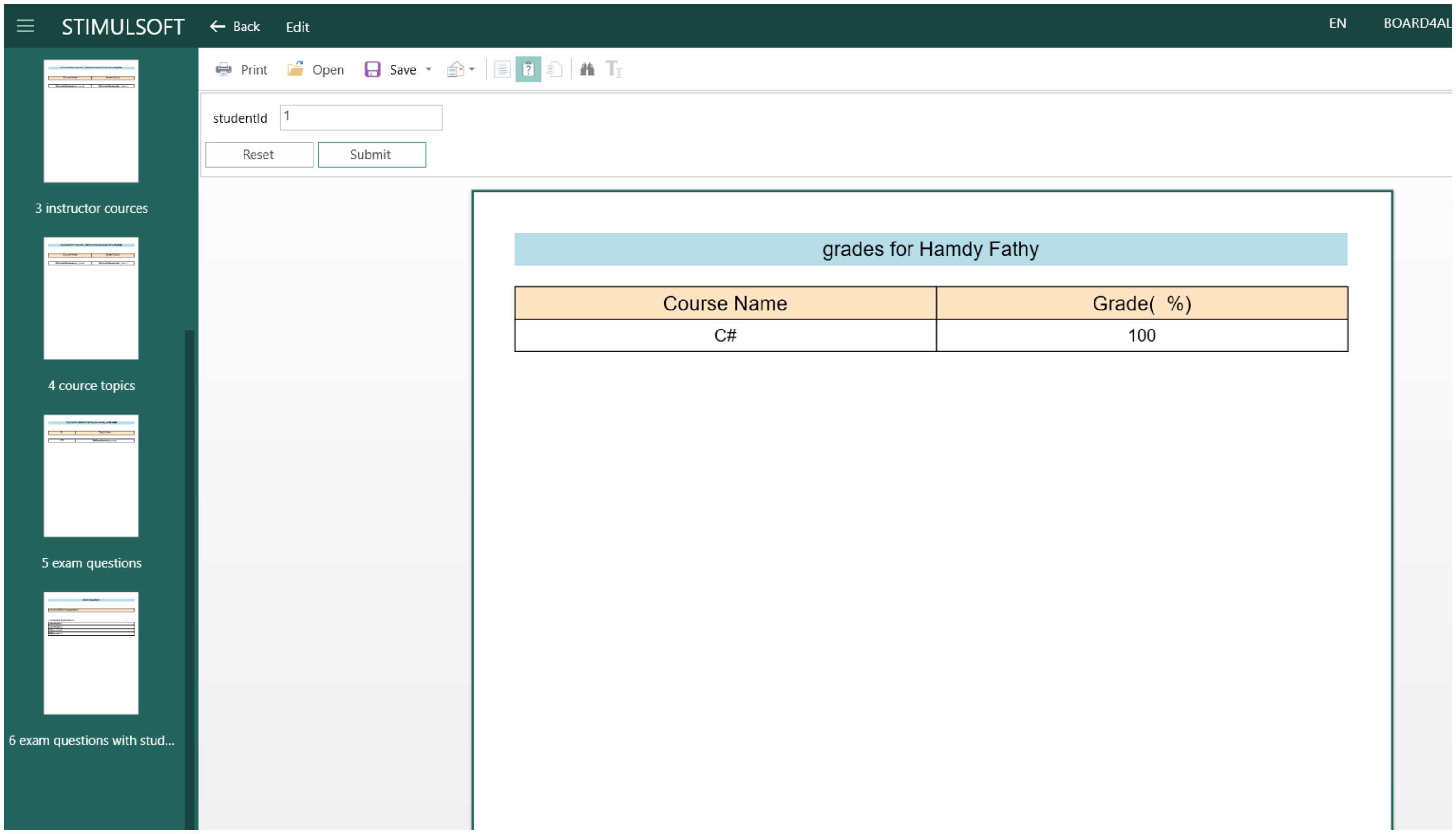


depno 1

Students info for IT department

ID	Name	Level
1	Hamdy Fathy	2
2	Ahmed Gomma	4
3	Ahmed khaled	1
4	amira khaled	3
5	Aya nabil	1
6	Badr Benoun	4
7	hady mustafa	3
8	hanan ismail	2
9	hoda mohamed	4
10	mohamed faried	1
11	Mohamed Hany	1
12	mohamed khamed	1
13	Mustafa Goda	1
14	Taher Mohamed	3
15	Walid Soliman	1
16	Yasser Ibrahim	1

Report that takes the student ID and returns the grades of the student in all courses.

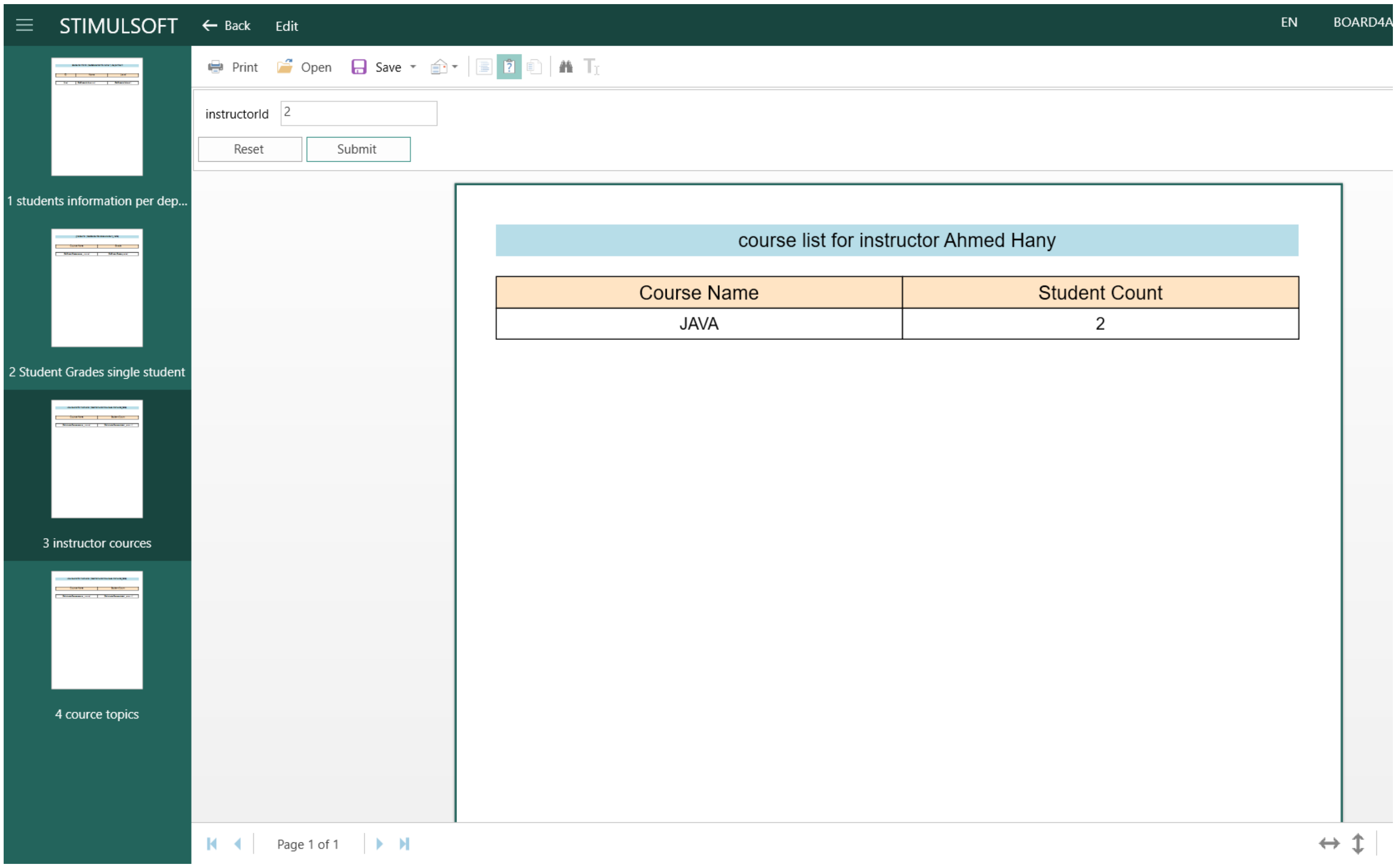


studentId 1

grades for Hamdy Fathy

Course Name	Grade(%)
C#	100

Report that takes the instructor ID and returns the name of the courses that he teaches and the number of student per course.

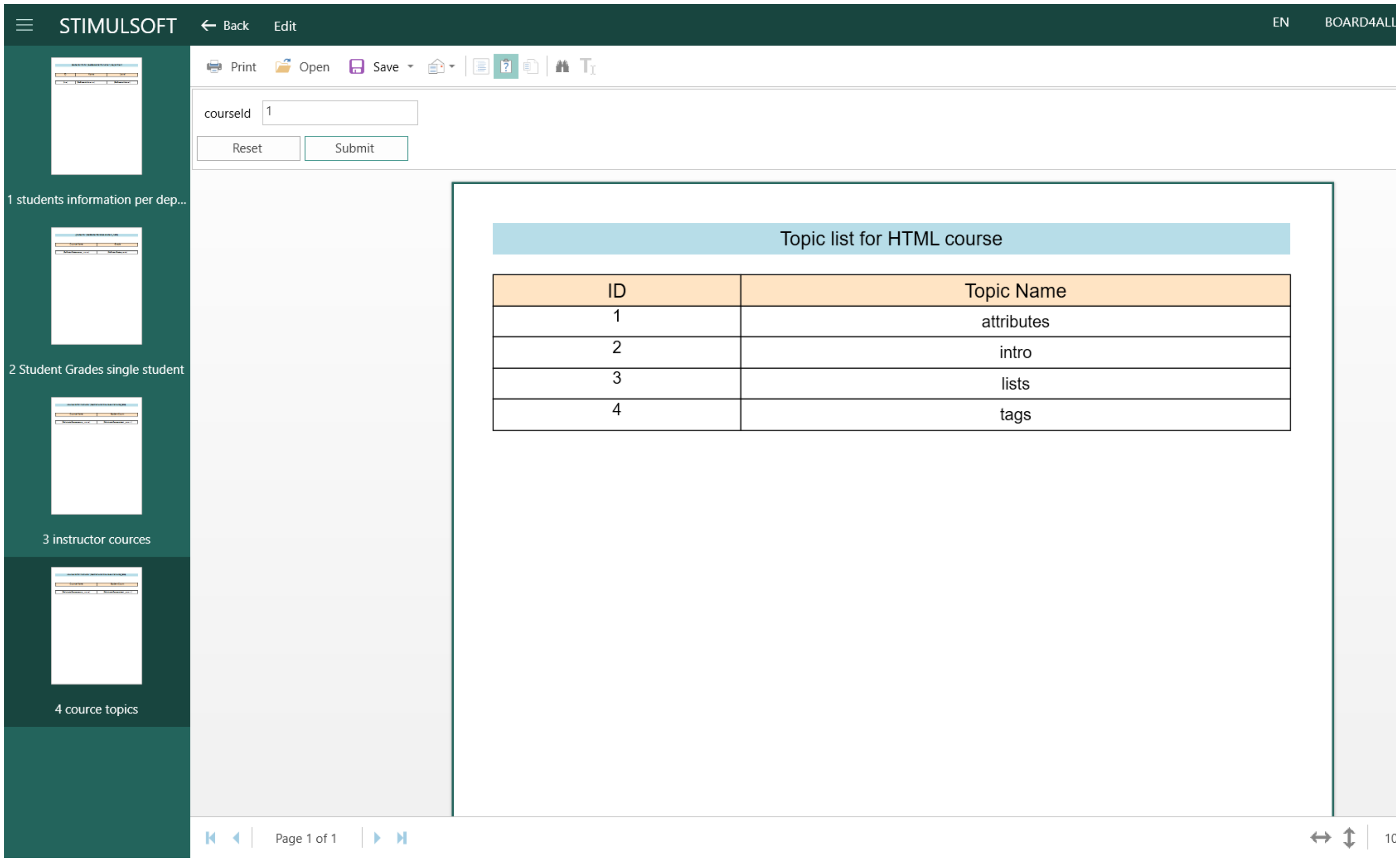


instructorId 2

course list for instructor Ahmed Hany

Course Name	Student Count
JAVA	2

Report that takes course ID and returns its topics

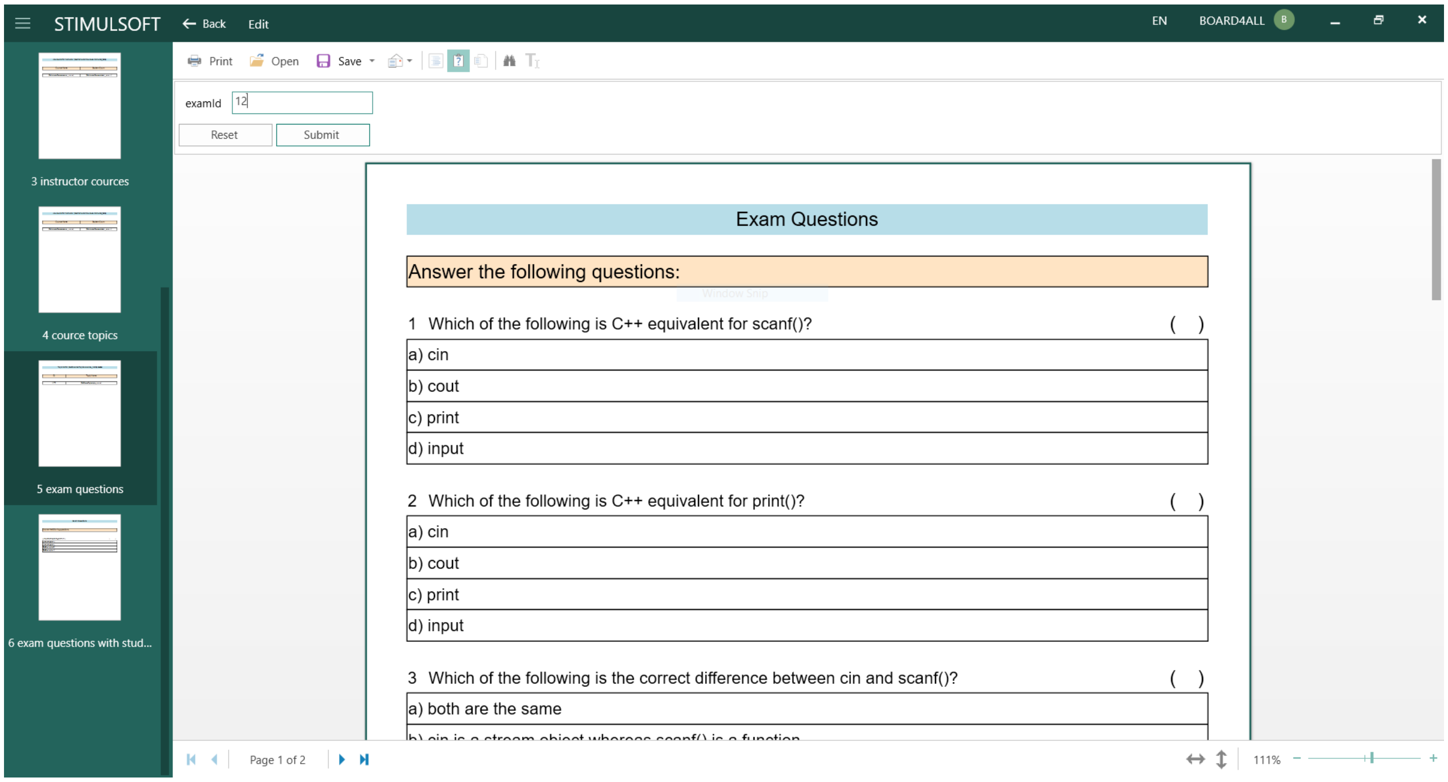


courseId: 1

Topic list for HTML course

ID	Topic Name
1	attributes
2	intro
3	lists
4	tags

Report that takes exam number and returns the Questions in it and chocies



Exam Questions

Answer the following questions:

1 Which of the following is C++ equivalent for scanf()? ( )

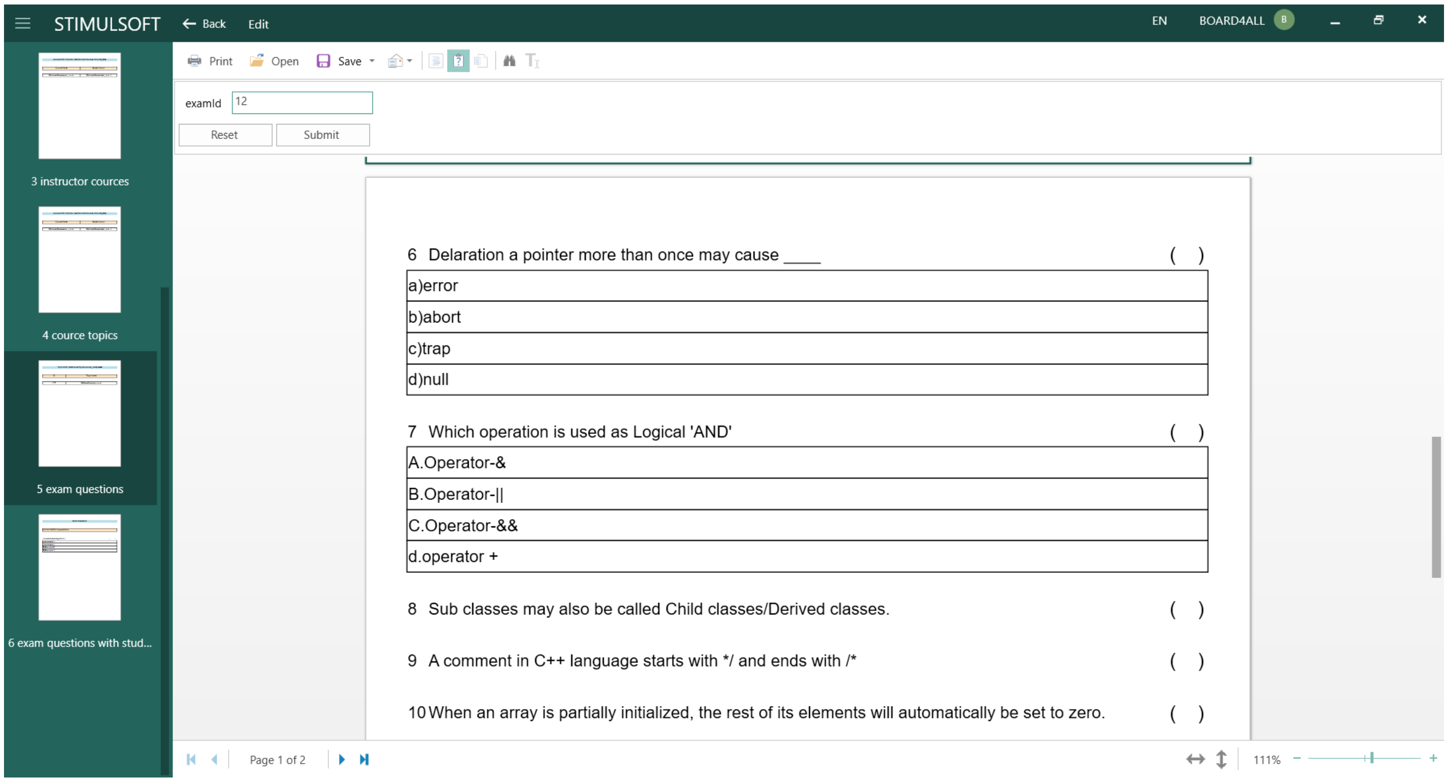
a) cin
b) cout
c) print
d) input

2 Which of the following is C++ equivalent for print()? ( )

a) cin
b) cout
c) print
d) input

3 Which of the following is the correct difference between cin and scanf()? ( )

a) both are the same



6 Delaration a pointer more than once may cause ____ ( )

a)error
b)abort
c)trap
d)null

7 Which operation is used as Logical 'AND' ( )

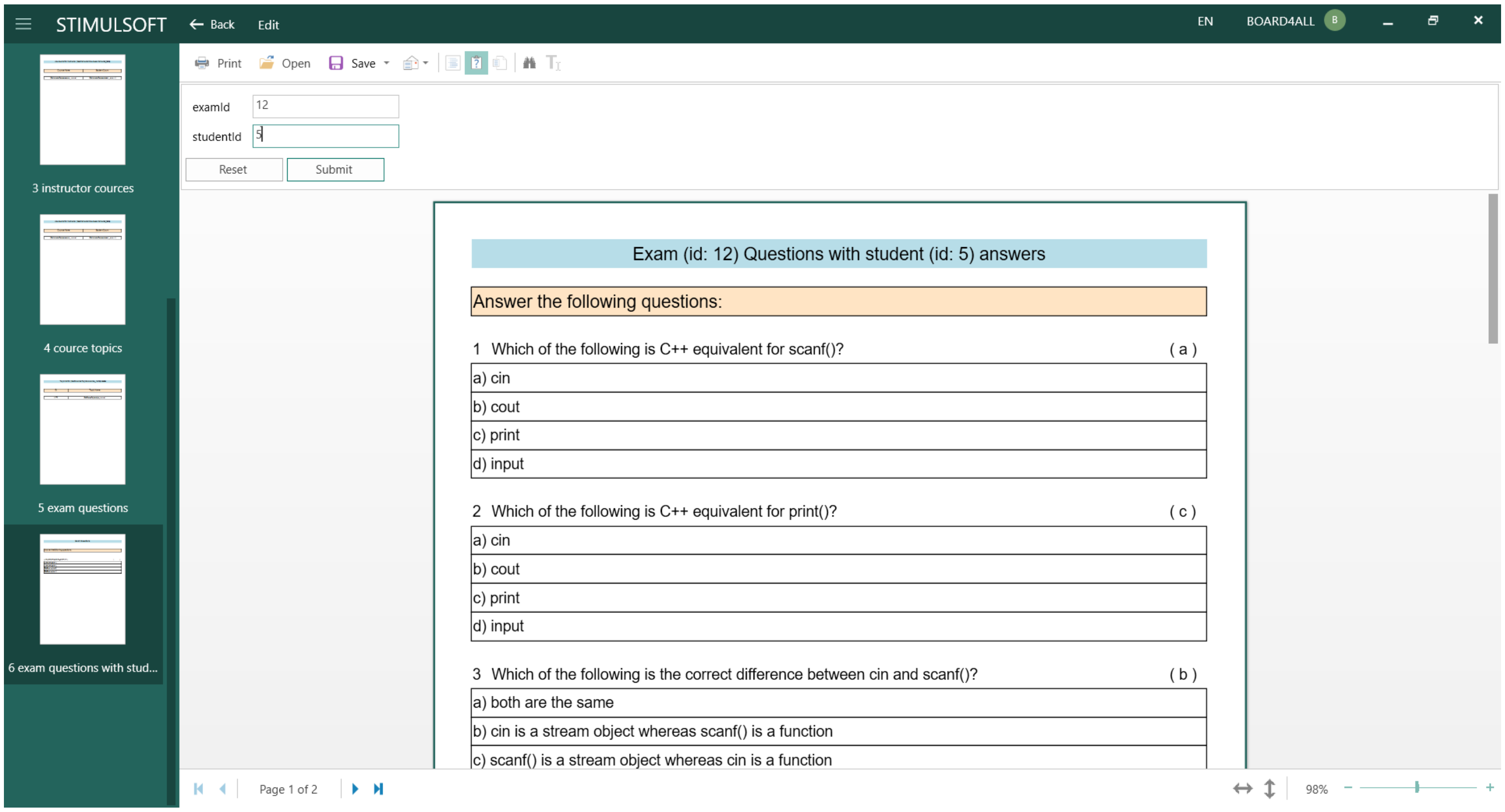
A.Operator-&
B.Operator-||
C.Operator-&&
d.operator +

8 Sub classes may also be called Child classes/Derived classes. ( )

9 A comment in C++ language starts with */ and ends with /* ( )

10 When an array is partially initialized, the rest of its elements will automatically be set to zero. ( )

Report that takes exam number and the student ID then returns the Questions in this exam with the student answers.



examId 12
studentId 5

Exam (id: 12) Questions with student (id: 5) answers

Answer the following questions:

1 Which of the following is C++ equivalent for scanf()? (a)

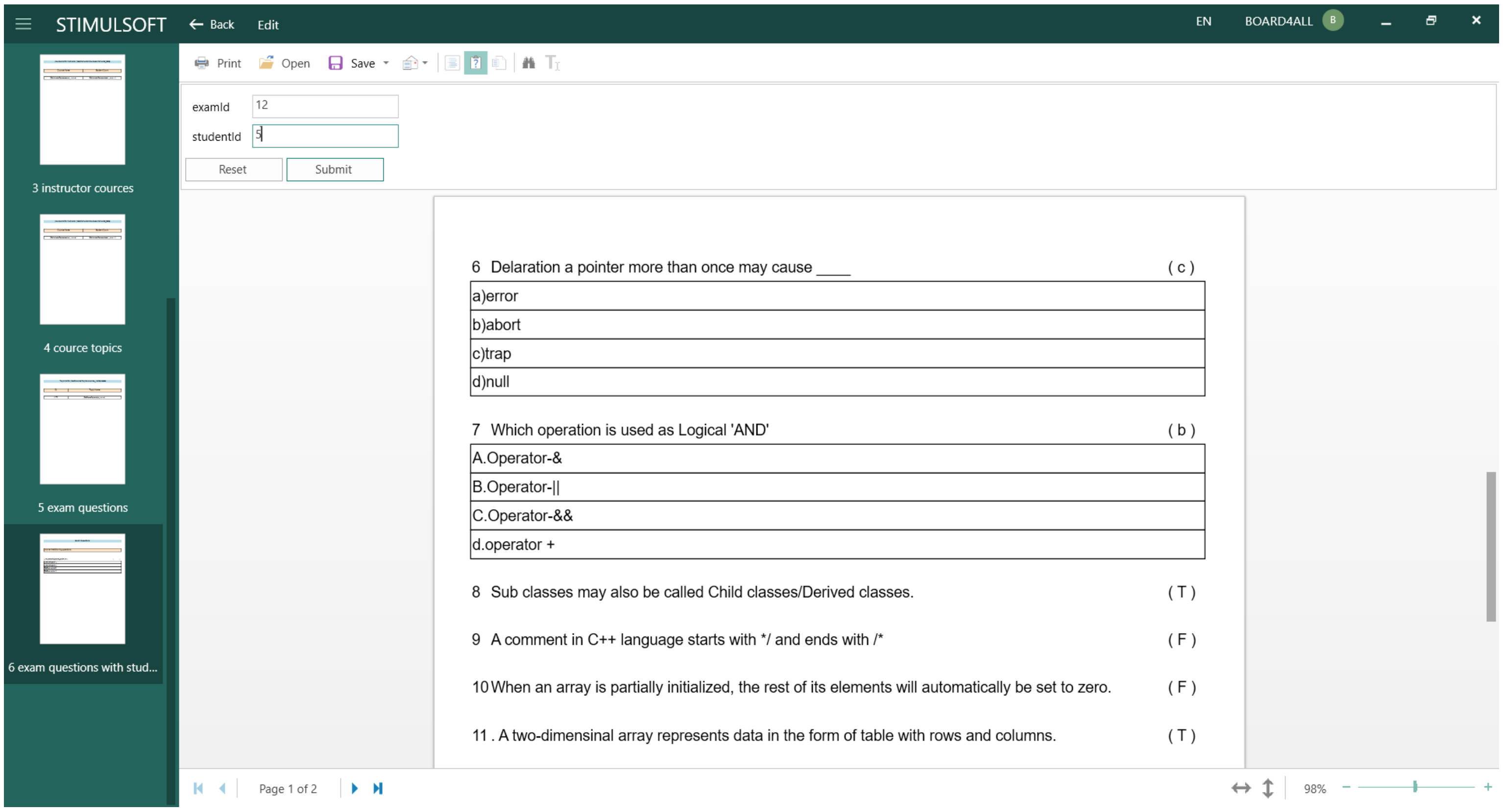
a) cin
b) cout
c) print
d) input

2 Which of the following is C++ equivalent for print()? (c)

a) cin
b) cout
c) print
d) input

3 Which of the following is the correct difference between cin and scanf()? (b)

a) both are the same
b) cin is a stream object whereas scanf() is a function
c) scanf() is a stream object whereas cin is a function



examId 12
studentId 5

6 Delaration a pointer more than once may cause ____ (c)

a)error
b)abort
c)trap
d)null

7 Which operation is used as Logical 'AND' (b)

A.Operator-&
B.Operator-||
C.Operator-&&
d.operator +

8 Sub classes may also be called Child classes/Derived classes. (T)

9 A comment in C++ language starts with */ and ends with /* (F)

10 When an array is partially initialized, the rest of its elements will automatically be set to zero. (F)

11 . A two-dimensinal array represents data in the form of table with rows and columns. (T)