TEAM 14

E-Learning Management Systems

Dinesh Murugesan - 002197301
Prachi Kunjir - 002197614
Sizhe Gu - 002198472
Sudarshan Waydande - 001563532
Yu Zhang - 001051940

Github Link: https://github.com/SudWay/DMDD-E-Learning-Management-Systems

Mission Statement:

- With the rapidly increasing rate of the demand for e-learning, the number of online courses is increasing. This project's aim is to make the management of education system flexible with easy accessibility, effectiveness, and reasonable cost.

- In this project, we'll be storing information of user's, the course providers, course, etc., manage and provide better services

Objectives:

- Database of courses that can be accessed by the user under one place, here users will be able to enroll for one or more course and complete certifications.

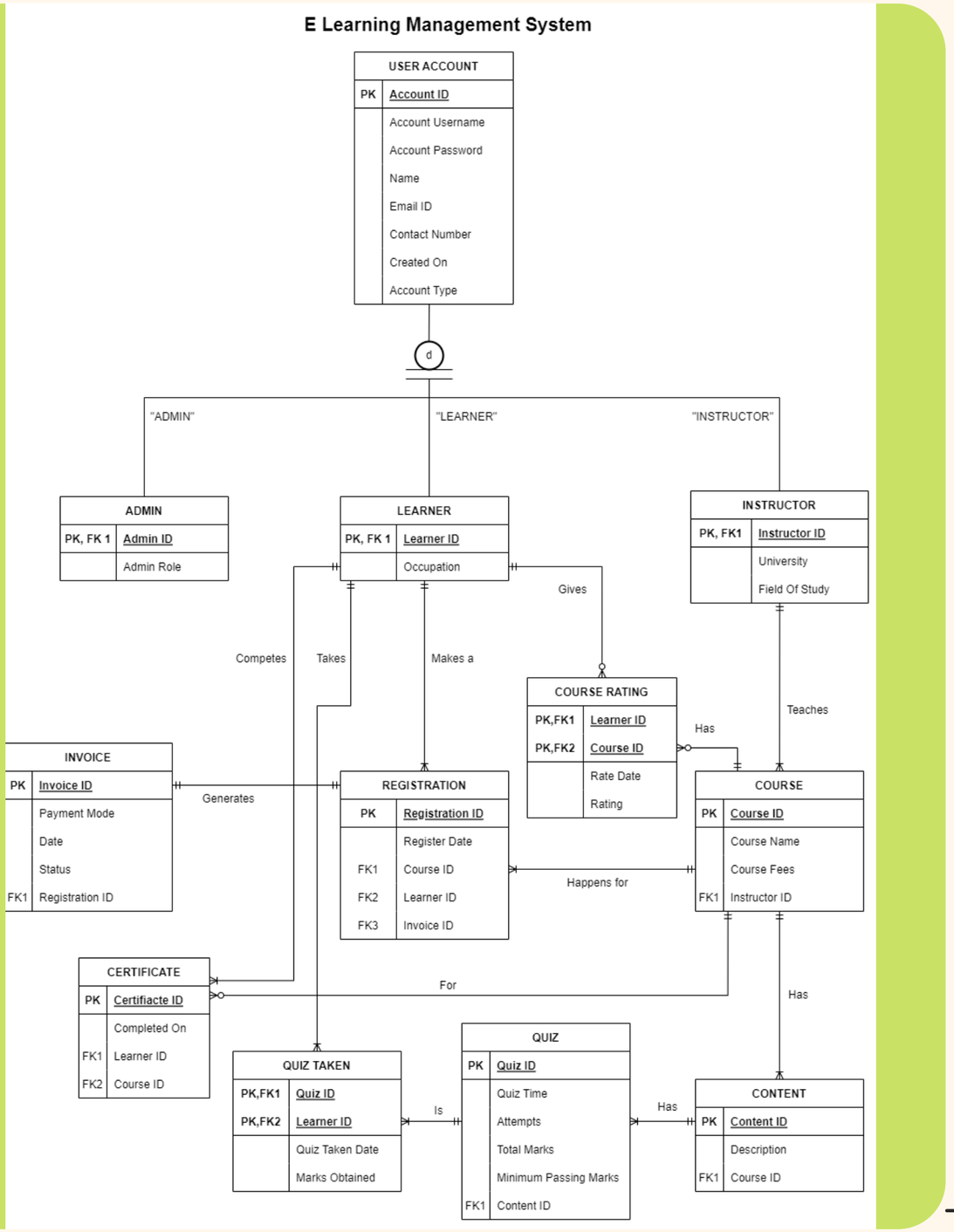
- Providing data to determine the popularity for the course depending upon how many users have taken that course.

- Providing data insights from the dataset that will help the platform owners to make an informed decision based on analyzed data.

Layout

- E-R Diagram
- Database Objects
  (DDl's, UDF's, Triggers,
  Stored Procedure, View's)
- Visualization
- GUI

E-R Diagram


E Learning Management System
USER ACCOUNT
PK Account ID
Account Username
Account Password
Name
Email ID
Contact Number
Created On
Account Type
d
"ADMIN"
"LEARNER"
"INSTRUCTOR"
ADMIN
PK, FK 1 Admin ID
Admin Role
LEARNER
PK, FK 1 Learner ID
Occupation
INSTRUCTOR
PK, FK1 Instructor ID
University
Field Of Study
Gives
Competes
Takes
Makes a
Teaches
COURSE RATING
PK,FK1 Learner ID
PK,FK2 Course ID
Rate Date
Rating
Has
INVOICE
PK Invoice ID
Payment Mode
Date
Status
FK1 Registration ID
Generates
REGISTRATION
PK Registration ID
Register Date
FK1 Course ID
FK2 Learner ID
FK3 Invoice ID
Happens for
COURSE
PK Course ID
Course Name
Course Fees
FK1 Instructor ID
CERTIFICATE
PK Certifiacte ID
Completed On
FK1 Learner ID
FK2 Course ID
For
Has
QUIZ TAKEN
PK,FK1 Quiz ID
PK,FK2 Learner ID
Quiz Taken Date
Marks Obtained
Is
QUIZ
PK Quiz ID
Quiz Time
Attempts
Total Marks
Minimum Passing Marks
FK1 Content ID
Has
CONTENT
PK Content ID
Description
FK1 Course ID

DDl's

We have created Database name 'Elearning' and created different tables with respective attributes. Following are the tables:

1. **USER_ACCOUNT**
2. **ADMIN**
3. **LEARNER**
4. **INSTRUCTOR**
5. **COURSE**
6. **CONTENT**
7. **QUIZ**
8. **REGISTRATION**
9. **QUIZ TAKEN**
10. **CERTIFICATE**
11. **COURSE RATING**
12. **INVOICE**

```
--Creating ELearning DataBase.
CREATE DATABASE [ELearning];

GO
--Using the ELearning Database.
USE [ELearning];

GO
--Table USER_ACCOUNT
CREATE TABLE [USER_ACCOUNT](
    [AccountId]          int     identity(1000000,1)   NOT NULL  PRIMARY KEY,
    [AccountUsername]    varchar(100)    NOT NULL,
    [AccountPassword]    varchar(100)    NOT NULL,
    [FirstName]          varchar(100)    NOT NULL,
    [LastName]           varchar(100)    NULL,
    [EmailId]            varchar(100)    NOT NULL,
    [ContactNo]          bigint          NOT NULL,
    [CreatedOn]          datetime        NOT NULL,
    [AccountType]        varchar(20)     CONSTRAINT CHK_ACCT_TYPE CHECK(AccountType IN ('ADMIN','LEARNER','INSTRUCTOR')),
)

GO

CREATE UNIQUE INDEX [USER_ACCOUNT_INDEX]
ON [USER_ACCOUNT]([AccountId]);

GO
--Table For USER PASSWORD ENCRYPTED.
CREATE TABLE [USER_ACCOUNT_ENCRYPTED](
    [AccountId]          int     identity(1000000,1)   NOT NULL  PRIMARY KEY,
    [AccountUsername]    varchar(100)    NOT NULL,
    [EncryptedPassword]    varbinary(400)    NOT NULL,
    [FirstName]          varchar(100)    NOT NULL,
    [LastName]           varchar(100)    NULL,
    [EmailId]            varchar(100)    NOT NULL,
    [ContactNo]          bigint          NOT NULL,
    [CreatedOn]          datetime        NOT NULL,
    [AccountType]        varchar(20)     NOT NULL,
)
```

UDF's and Encryption

We have created 3 User Defined Function's. First is for Encrypting the User Account Password, Second is to Decrypting that Password and Third to get the Admin Role given the AdminId.

```
--For Encryption please run in the following Steps:
--Step 1 :Run below commands:
create MASTER KEY
ENCRYPTION BY PASSWORD = 'DAMGTeam14^';

CREATE CERTIFICATE AccountPass
   WITH SUBJECT = 'Account Password';
GO

CREATE SYMMETRIC KEY AccountPass_SM
    WITH ALGORITHM = AES_256
    ENCRYPTION BY CERTIFICATE AccountPass;
GO
--Step 2: Run Below command:
OPEN SYMMETRIC KEY AccountPass_SM
   DECRYPTION BY CERTIFICATE AccountPass;

--Step 3: Run below command and create a  UDF
----UDF for ENCRYPTING---------------
create function ENCRYPT_PASSWORD
(
        @pwd varchar(100)
)
returns varbinary(400)
as
begin
        declare @encryptedPassword varbinary(400)
        set @encryptedPassword = EncryptByKey(Key_GUID('AccountPass_SM'),  convert(varbinary, @pwd))
        return @encryptedPassword
end
go
```

```
--Step 4: Run below command and create a UDF
-----UDF for DECRYPTING--------------
create function DECRYPT_PASSWORD
(
        @encryptpwd varbinary(400)
)
returns varchar(100)
as
begin
        declare @decryptedPassword varchar(100)
        set @decryptedPassword = CONVERT(varchar, DecryptByKey(@encryptpwd))
        return @decryptedPassword
end
go

--Step 5: Run below command
CLOSE SYMMETRIC KEY AccountPass_SM;

GO
-------------------------------------------------------------------------------------------------
----------------UDF for getting Admin Role ---------------------------------------

CREATE FUNCTION [AD_Role](@Admin_Id int)
RETURNS varchar(100)
AS
BEGIN
        DECLARE @res varchar(100)
        IF exists (SELECT AdminId FROM [ADMIN] WHERE [AdminId] = @Admin_Id)
        BEGIN

                SET @res = (SELECT AdminRole FROM [ADMIN] WHERE [AdminId] = @Admin_Id)
                END

        Return @res

END
GO
```

Triggers's

We have created 2 Trigger's. First trigger, when a new User is Added into the USER ACCOUNT table we'll use Encryption UDF to encrypt the password and store the user information in new Table USER_ACCOUNT_ENCRYPTED. And second trigger for when USER updates their ContactNo we'll store the record in the USER_ACCOUNT_AUDIT table.

```
--Trigger to Encrypt user password and added User in USER_ACCOUNT_ENCRYPTED Table
CREATE TRIGGER [INSERT_USER_AND_ENCRYPT]
ON USER_ACCOUNT
AFTER INSERT
AS
BEGIN

	SET NOCOUNT ON;
	OPEN SYMMETRIC KEY AccountPass_SM
	   DECRYPTION BY CERTIFICATE AccountPass;

	INSERT INTO [dbo].[USER_ACCOUNT_ENCRYPTED]
			([AccountUsername]
			,[EncryptedPassword]
			,[FirstName]
			,[LastName]
			,[EmailId]
			,[ContactNo]
			,[CreatedOn]
			,[AccountType])

		SELECT
		i.AccountUsername,
		dbo.ENCRYPT_PASSWORD(i.AccountPassword),
		i.FirstName,
		i.LastName,
		i.EmailId,
		i.ContactNo,
		i.CreatedOn,
		i.AccountType
		FROM
			inserted i


	CLOSE SYMMETRIC KEY AccountPass_SM;
END

GO
```

```
------Trigger for when contact number is updated

Create TRIGGER [UPDATE_USER_ACCOUNT_CONTACT] ON [USER_ACCOUNT]
After UPDATE
AS
BEGIN

	SET NOCOUNT ON;
	DECLARE @ID INT, @NewContactNo BIGINT, @OldContactNo BIGINT

	Select @ID = i.AccountId from inserted i

	Select * into #TempTable from inserted

	Select Top 1 @NewContactNo = ContactNo from #TempTable

	Select @OldContactNo = ContactNo from deleted where AccountId = @ID


	IF UPDATE(ContactNo)
		INSERT INTO [USER_ACCOUNT_AUDIT]
		VALUES(@ID, @OldContactNo,@NewContactNo)


END
GO
```

Stored Procedure's

We have created 5 Stored Procedure. They are as follows:

1. **Stored Procedure to INSERT USER.**
2. **Stored Procedure for Given the Instructor get the Course and Learner details.**
3. **Stored Procedure for Getting Registration Details of a Learner.**
4. **Check whether given the LearnerId if Learner hs unpass a Quiz.**
5. **Find the Max Score obtained given the QuizId**

```
CREATE PROCEDURE [FIND_LEARNER] @Instructor_id INT
AS
BEGIN
        SELECT CO.InstructorId, CO.CourseId, RGTR.LearnerId, UA.FirstName, UA.LastName
        FROM
        (
                COURSE CO
                JOIN REGISTRATION AS RGTR
                ON CO.CourseId = RGTR.CourseId and CO.InstructorId = @Instructor_id
        )
        JOIN USER_ACCOUNT AS UA
        ON RGTR.LearnerId = UA.AccountId
        ORDER BY CourseId ASC
END
GO
```

View's

We have created 4 Views. They are as follows:

1. VIEW the top 3 students performed in the quiz for each contents or any specific content.
2. View the average rating of all courses.
3. View the Number of Registration and Number of Certificate completed for that Course.
4. View the Number of contents for the Course.

```
CREATE VIEW [TOP3LEARNER]
AS
SELECT [ContentId], [LearnerId], [Ranking]
FROM
(
    SELECT Q.ContentId, QT.LearnerId,
    ROW_NUMBER() OVER (PARTITION BY q.ContentId ORDER BY Q.ContentId, MAX(QT.MarksObtained) DESC, MIN(QT.MarksObtained) DESC, QT.LearnerId DESC) AS Ranking
    FROM QUIZ_TAKEN QT JOIN QUIZ Q
    ON Q.QuizId = QT.QuizId
    GROUP BY Q.ContentId, QT.LearnerId
)ranktable
WHERE Ranking <= 3
GO
```

Visualization (PowerBI)

Field of Study

Select all	Data Analytics
(Blank)	Data Science
Application Development	Database Design
Big Data	SCRUM
Cloud Computing	User Interface
Cyber Security	Web Design

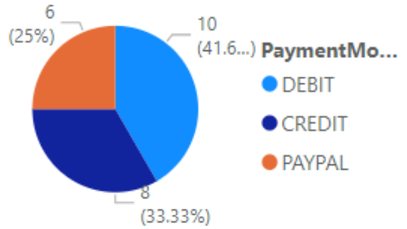
Course Name

Application Engineering and Development

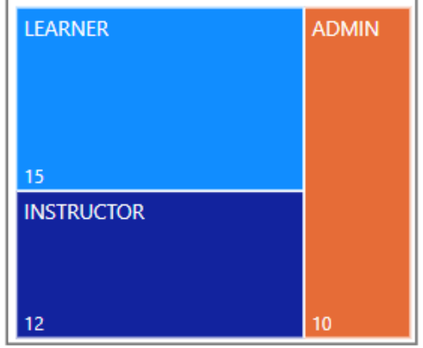
Course Rating

4.08

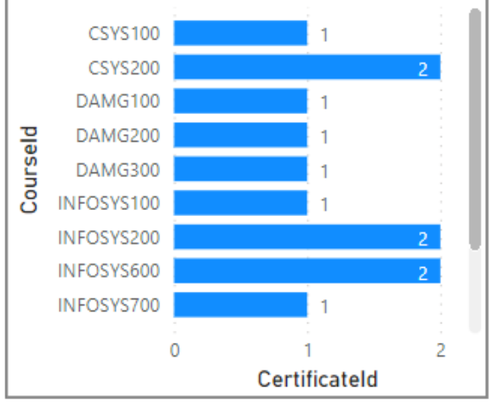
Payment Mode Insights



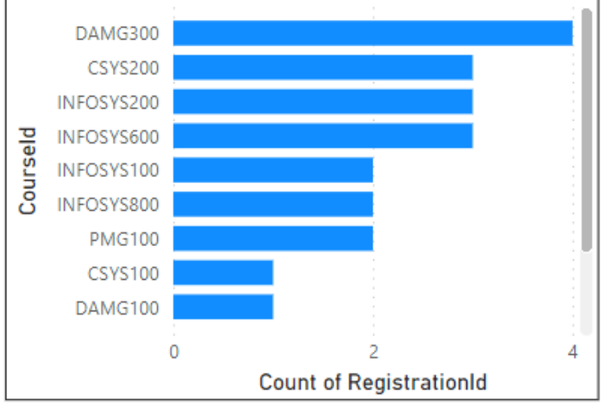
Account Type



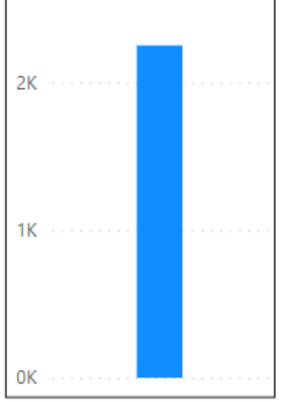
Certificates



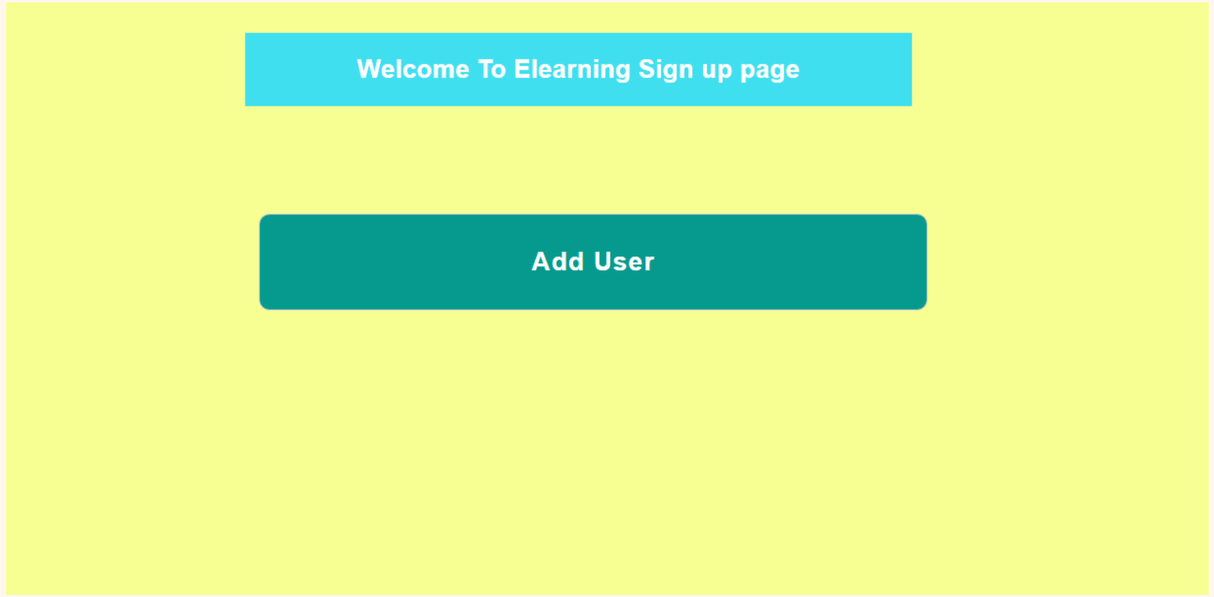
Registrations

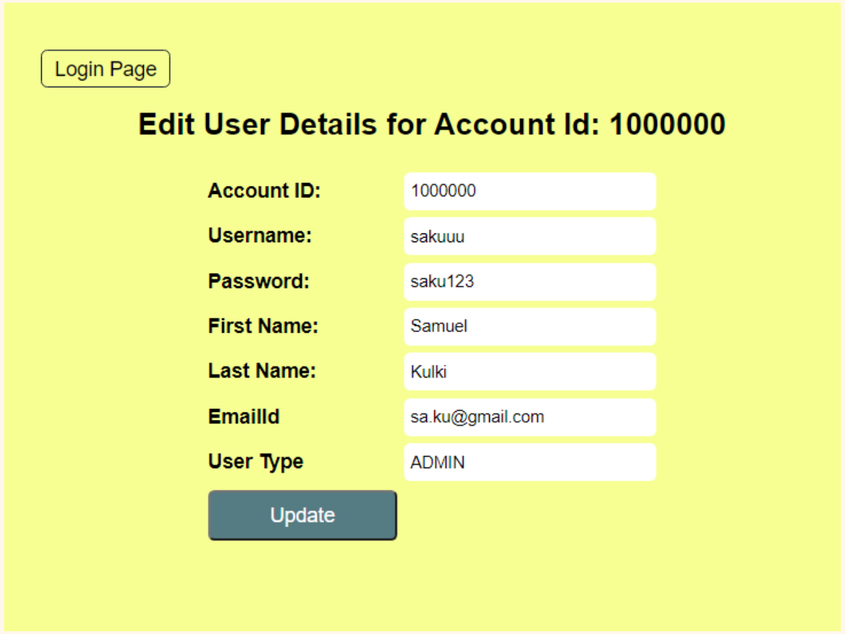


Course Fee



GUI Screenshots





Thank You For Listening