A screenshot of a computer

Description automatically generated

Service that Connects Freelance Tutors with Individuals in Need

IST 659 Final Project by

James Du, Dale Deans, and Cynthia Scotton

**Overview**

TutorMatch is a service that connects clients with freelance tutors in hopes of getting assistance in their academics. Our goal is to create a database that allows clients to post jobs where tutors can view these available postings and have the opportunity to accept them and connect with these clients. First, we will focus on defining our entities and relationships and setting our business rules, producing our conceptual and logical models for our database. Then we will move towards implementation by building and testing our database to make sure it works and answering our business questions.

**Stakeholders**

* Software engineers/maintenance
* End-users (clients and freelance tutors)
* Potential business partners
* Potential investors

**Business Rules/Relationships**

* A client posts zero or more jobs.
* A job contains job details.
* An appointment is made for a job.
* An appointment contains appointment details.
* A tutor makes zero or more appointments.
* A client can make zero or more reviews.
* A review contains a rating.
* A review is made for a tutor.
* A tutor can have zero or more reviews.

**Glossary**

**Clients** are people who post a job for tutors to see.

**Jobs** are posted by clients. They contain details such as when the job was posted, the expiration/due date for the job, the type of assignment the job pertains to, and who posted the job.

**Appointments** are made by tutors who accept jobs. They contain information like the tutor who has made the appointment, the pay rate, the appointment date, and the job the appointment is for.

**Tutors** are the individuals who view the available jobs and accept the ones they are interested in, making an appointment with the client.

**Reviews** are made by clients for tutors. They tell us who made the review, who the review is for, and contain a rating value and comment.

**Rating\_astype** contains the rating value. It ranges from 1, which is considered really bad, to 5 , which is considered excellent.

**Conceptual Model**

A screenshot of a computer screen

Description automatically generated with medium confidence

**Logical Model**

A screenshot of a computer

Description automatically generated

**Business Questions**

* Which individuals posted the most jobs?
* Which tutor has taken the most jobs?
* Who are the inactive tutors on the platform? (no jobs posted/accepted)
* Create a table which includes the client information, tutor information, and appointment date.
* Which tutor has the highest pay rate?
* What are the average ratings for each tutor?
* Create a query that returns the jobs count grouped by assignment type

**Code**

create database tservice

GO

Use tservice -- replace this line with whatever database you named this as, is changeable

GO

--DROP

--drop procedures

drop procedure if exists p\_upsert\_client

GO

drop procedure if exists p\_upsert\_job

GO

drop procedure if exists p\_upsert\_tutor

GO

drop procedure if exists p\_upsert\_appointments

GO

drop procedure if exists p\_upsert\_reviews

GO

-- drop foreign key constraints

if exists(select \* from INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS

where CONSTRAINT\_NAME='fk\_jobs\_job\_client\_id')

alter table jobs drop constraint fk\_jobs\_job\_client\_id

if exists(select \* from INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS

where CONSTRAINT\_NAME='fk\_appointments\_appointment\_job\_id')

alter table appointments drop constraint fk\_appointments\_appointment\_job\_id

if exists(select \* from INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS

where CONSTRAINT\_NAME='fk\_appointments\_appointment\_tutor\_id')

alter table appointments drop constraint fk\_appointments\_appointment\_tutor\_id

if exists(select \* from INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS

where CONSTRAINT\_NAME='fk\_reviews\_review\_for\_user')

alter table Reviews drop constraint fk\_reviews\_review\_for\_user

if exists(select \* from INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS

where CONSTRAINT\_NAME='fk\_reviews\_review\_by\_user')

alter table Reviews drop constraint fk\_reviews\_review\_by\_user

if exists(select \* from INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS

where CONSTRAINT\_NAME='fk\_reviews\_rating')

alter table Reviews drop constraint fk\_reviews\_rating

GO

Drop table if exists Clients

Drop table if exists Jobs

Drop table if exists Rating\_Astype\_Lookup

Drop table if exists Reviews

DROP TABLE IF EXISTS Tutors

DROP TABLE IF EXISTS Appointments

GO

--UP Metadata

-- create tables

CREATE TABLE Clients (

    client\_id INT PRIMARY KEY IDENTITY (1, 1),

    firstname VARCHAR(100) not NULL,

    lastname varchar(100) not NULL,

    email varchar(100) not NULL,

    )

    -- Resetting surrogate key

    DBCC CHECKIDENT ('Clients', RESEED, 1);

GO

CREATE TABLE Jobs (

    job\_id INT PRIMARY KEY IDENTITY (1, 1),

    job\_client\_id INT,

    date\_posted date not NULL,

    due\_date date not NULL,

    assign\_type varchar(100) -- not sure if this is supposed to be yes or no thing but can change after

    constraint fk\_jobs\_job\_client\_id Foreign KEY (job\_client\_id) References CLIENTS(client\_id)

        )

    DBCC CHECKIDENT ('Jobs', RESEED, 1);

GO

create table Tutors (

tutor\_id int identity (1, 1) not null,

firstname char(50) not null,

lastname char(50) not null,

tutor\_email varchar(50) not null,

skills\_description varchar(50) not null,

constraint pk\_tutors primary key (tutor\_id),

constraint u\_tutors unique (tutor\_email),

)

GO

create table Appointments (

appointment\_id int identity (1, 1) not null,

appointment\_job\_id int not null,

appointment\_tutor\_id int not null,

pay\_rate money not null,

app\_date date not null,

constraint pk\_appointments primary key (appointment\_id),

constraint fk\_appointments\_appointment\_job\_id foreign key (appointment\_job\_id) references jobs(job\_id),

constraint fk\_appointments\_appointment\_tutor\_id foreign key (appointment\_tutor\_id) references Tutors(tutor\_id)

)

GO

CREATE TABLE Rating\_Astype\_Lookup(

    rating\_astype INT PRIMARY KEY not NULL

)

GO

CREATE TABLE Reviews(

    review\_id INT IDENTITY (1, 1) not NULL,

    review\_for\_user int not NULL,

    review\_by\_user int not NULL,

    rating int not NULL,

    comment varchar(150) not NULL

    constraint pk\_reviews primary key (review\_id),

    constraint fk\_reviews\_review\_for\_user foreign key (review\_for\_user) references Tutors(tutor\_id),

    constraint fk\_reviews\_review\_by\_user foreign key (review\_by\_user) references Clients(client\_id),

    constraint fk\_reviews\_rating foreign key (rating) references Rating\_astype\_lookup(rating\_astype)

)

GO

-- UP Data

    Insert into Clients (firstname, lastname, email)

    VALUES

    ('Stephen', 'King', 'skiing@gmail.com'),

    ('Andy', 'Harris', 'aharri@outlook.com'),

    ('Brynlee', 'Torres', 'Bryntorres123@gmail.com'),

    ('Angela', 'Simpson', 'A\_simp\_son@gmail.com'),

    ('Arabella', 'Holmes', 'homles93@willms.com'),

    ('Bentley', 'Horton', 'benton\_horley@gmail.com'),

    ('Ricardo', 'Reyes', 'rreyes@yahoo.com'),

    ('Christopher','Parker','cpaker34@gmail.com'),

    ('Joyce', 'Asta', 'astrojoyce@outlook.com'),

    ('Jorge', 'Parker', 'jayparker@gmail.com'),

    ('Angela', 'Smith', 'angelasmith333@gmail.com'),

    ('Axel', 'Reyes', 'axelotl@gmail.com'),

    ('Wynter', 'Foster', 'foster.wyn@foos.com'),

    ('Ian', 'Young', 'ian.old@gmail.com'),

    ('Alexa', 'Gutierrez', 'isporie@gmail.com'),

    ('Daniella', 'Walker', 'starwalker9@sky.net'),

    ('Annalise', 'Hermiston', 'Hermanna@gmail.com'),

    ('Simon', 'Ortiz', 'simonsaidso@gmail.com'),

    ('Kate','Douglas','katendoug@gmail.com'),

    ('Keith','Webster','Dictionarykeith@gmail.com')

GO

    Insert into Jobs (job\_client\_id, date\_posted, due\_date, assign\_type)

    VALUES

    ('1','2023-06-01', '2023-06-15', 'homework'),

    ('3','2023-05-29', '2023-06-05', 'project'),

    ('4','2023-04-30', '2023-07-25', 'test'),

    ('1', '2023-05-20', '2023-05-22', 'essay'),

    ('1','2023-06-03', '2023-06-04', 'misc'),

    ('10','2023-05-23', '2023-06-02', 'project'),

    ('5','2023-04-30', '2023-05-05', 'project'),

    ('13', '2023-05-22', '2023-05-24', 'quiz'),

    ('14','2023-06-01', '2023-06-14', 'homework'),

    ('13','2023-05-28', '2023-06-01', 'project'),

    ('17','2023-05-30', '2023-07-01', 'project'),

    ('15', '2023-05-20', '2023-05-22', 'test'),

    ('18','2023-06-01', '2023-06-01', 'homework'),

    ('7','2023-05-29', '2023-06-03', 'misc'),

    ('9','2023-05-31', '2023-06-25', 'misc'),

    ('6', '2023-05-20', '2023-05-25', 'essay')

GO

Insert into Tutors

    (firstname, lastname, tutor\_email, skills\_description) values

    ('John', 'Renner', 'jrenner24@gmail.com', 'research paper writing'),

    ('Sarah', 'Dunst', 'sd0408@gmail.com', 'history'),

    ('Darren', 'Sandusky', 'darrensan@aol.com', 'music history'),

    ('Oliver', 'Queen', 'oliqueen@yahoo.com', 'art history'),

    ('Jessica', 'Bernard', 'jb2707@aol.com', 'r language'),

    ('Claire', 'Johnson', 'cjohnson1@outlook.com', 'spanish'),

    ('Ben', 'Rivers', 'benriv@gmail.com', 'history'),

    ('Jaren', 'Jack', 'jjack99@yahoo.com', 'general biology'),

    ('Sonia', 'Kabelo', 'soniakab@aol.com', 'creative writing'),

    ('Nisha', 'Vimal', 'nvimal@outlook.com', 'statistics'),

    ('Chris', 'Mora', 'chrism34@outlook.com', 'business law'),

    ('Lindsay', 'Lamb', 'llcool20@aol.com', 'economics'),

    ('Kelly', 'Bradford', 'kb2408@gmail.com', 'calculus'),

    ('Ramiro', 'Mack', 'rmack@gmail.com', 'organic chemistry'),

    ('Henrietta', 'Roach', 'henrir1414@yahoo.com', 'SQL language'),

    ('Vince', 'Ken', 'vken@hotmail.com', 'music history'),

    ('Hunter', 'Saunders', 'hsaunders@aol.com', 'calculus'),

    ('Kristie', 'Graham', 'kg215@hotmail.com', 'film'),

    ('Bernice', 'Phillips', 'bephil@gmail.com', 'database management'),

    ('Heath', 'Reeves', 'hreev87@outlook.com', 'english literature')

GO

Insert into Appointments

    (appointment\_job\_id, appointment\_tutor\_id, pay\_rate, app\_date) values

    ('3', '17', '20', '2023-05-10'),

    ('7', '15', '25', '2023-05-01'),

    ('13', '10', '15', '2023-06-01'),

    ('12', '8', '30', '2023-05-20'),

    ('1', '20', '20', '2023-06-05'),

    ('6', '9', '18', '2023-05-24'),

    ('11', '12', '15', '2023-06-10'),

    ('16', '1', '25', '2023-05-21'),

    ('10', '10', '20', '2023-05-28'),

    ('2', '2', '18', '2023-05-31'),

    ('15', '14', '10', '2023-06-15'),

    ('4', '7', '30', '2023-05-20')

GO

INSERT into Rating\_astype\_lookup VALUES

 ('1'),

 ('2'),

 ('3'),

 ('4'),

 ('5')

 GO

insert into Reviews

(review\_for\_user, review\_by\_user, rating, comment) values

('17', '4', '5', 'great job'),

('15', '5', '4', 'good work'),

('10', '18','5', 'explained homework in a well-organized way'),

('8', '15', '5', 'test scores increased'),

('10', '13', '1', 'was not able to help with project, tutor was not well-versed on the subject'),

('2', '3', '3', 'tutor was very patient but still did not understand the concept')

GO

-- Verify

select \* from Clients

GO

select \* from Jobs

GO

select \* from Tutors

GO

select \* from Appointments

GO

select \* from Reviews

GO

select \* from Rating\_Astype\_Lookup

GO

-- create procedures

-- these stored procedures will allow users to create new client and tutor accounts, new jobs and appointments, and post new reviews

create PROCEDURE p\_upsert\_client (

    @firstname varchar(100),

    @lastname varchar(100),

    @email varchar(100))

AS

BEGIN

    if exists(select \* from "Clients" where firstname = @firstname)

        begin

            update "Clients"

            set firstname = @firstname, lastname = @lastname, email = @email

            where firstname = @firstname

        END

    ELSE

        BEGIN

            insert into "Clients"

            (firstname, lastname, email)

            VALUES

            (@firstname, @lastname, @email)

        END

        RETURN @@IDENTITY

    END;

GO

create PROCEDURE p\_upsert\_job(

    @job\_client\_id INT,

    @date\_posted date,

    @due\_date date,

    @assign\_type varchar(100)

)

AS

BEGIN

    if exists(select \* from "Jobs" where job\_client\_id = @job\_client\_id)

        begin

            update "Jobs"

            set job\_client\_id = @job\_client\_id, date\_posted = @date\_posted, due\_date = @due\_date,

            assign\_type = @assign\_type

            where job\_client\_id = @job\_client\_id

        END

    ELSE

        BEGIN

            insert into "Jobs"

            (job\_client\_id, date\_posted, due\_date, assign\_type)

            VALUES

            (@job\_client\_id, @date\_posted, @due\_date, @assign\_type)

        END

        RETURN @@IDENTITY

    END;

GO

create PROCEDURE p\_upsert\_tutor(

    @firstname char(50),

    @lastname char(50),

    @tutor\_email varchar(50),

    @skills\_description varchar(50)

)

AS

BEGIN

    if exists(select \* from "Tutors" where firstname = @firstname)

        begin

            update "Tutors"

            set firstname = @firstname, lastname = @lastname, tutor\_email = @tutor\_email, skills\_description = @skills\_description

            where firstname = @firstname

        END

    ELSE

        BEGIN

            insert into "Tutors"

            (firstname, lastname, tutor\_email, skills\_description)

            VALUES

            (@firstname, @lastname, @tutor\_email, @skills\_description)

        END

        RETURN @@IDENTITY

    END;

GO

create PROCEDURE p\_upsert\_appointments(

    @appointment\_job\_id int,

    @appointment\_tutor\_id int,

    @pay\_rate money,

    @app\_date date

)

AS

BEGIN

    if exists(select \* from "Appointments" where appointment\_job\_id = @appointment\_job\_id)

        begin

            update "Appointments"

            set appointment\_job\_id = @appointment\_job\_id, appointment\_tutor\_id = @appointment\_tutor\_id, pay\_rate = @pay\_rate, app\_date = @app\_date

            where appointment\_job\_id = @appointment\_job\_id

        END

    ELSE

        BEGIN

            insert into "Appointments"

            (appointment\_job\_id, appointment\_tutor\_id, pay\_rate, app\_date)

            VALUES

            (@appointment\_job\_id, @appointment\_tutor\_id, @pay\_rate, @app\_date)

        END

        RETURN @@IDENTITY

    END;

GO

create PROCEDURE p\_upsert\_reviews(

    @review\_for\_user int,

    @review\_by\_user int,

    @rating int,

    @comment varchar(150)

)

AS

BEGIN

    if exists(select \* from "Reviews" where review\_for\_user = @review\_for\_user)

        begin

            update "Reviews"

            set review\_for\_user = @review\_for\_user, review\_by\_user = @review\_by\_user, rating = @rating, comment = @comment

            where review\_for\_user = @review\_for\_user

        END

    ELSE

        BEGIN

            insert into "Reviews"

            (review\_for\_user, review\_by\_user, rating, comment)

            VALUES

            (@review\_for\_user, @review\_by\_user, @rating, @comment)

        END

        RETURN @@IDENTITY

    END;

GO

-- testing the stored procedure to make sure it works

exec p\_upsert\_client

@firstname = 'Bob',

@lastname = 'Truman',

@email = 'btruman80@aol.com'

GO

select \* from Clients

**A screenshot of a computer

Description automatically generated**

-- form created with WorkForms

A screenshot of a computer

Description automatically generated

-- views created to answer questions

create view v\_averageTutorRating as

select avg(rating) as 'average rating',

review\_for\_user as tutor\_id

from Reviews

group by review\_for\_user

GO

select \* from v\_averageTutorRating

GO

create view v\_jobsCount as

select count(job\_id) as '# of jobs',

assign\_type as 'assignmnet type'

from Jobs

group by assign\_type

GO

select \* from v\_jobsCount

GO

create view v\_clientTutorAppInfo as

select Clients.client\_id,

Clients.firstname+' '+Clients.lastname as 'client name',

Clients.email as 'client email',

Tutors.tutor\_id,

Tutors.firstname+' '+Tutors.lastname as 'tutors name',

Tutors.tutor\_email as 'tutor email',

Appointments.app\_date as 'appointment date'

from Appointments

join Tutors on Appointments.appointment\_tutor\_id = Tutors.tutor\_id

join Jobs on Appointments.appointment\_job\_id = Jobs.job\_id

join Clients on Jobs.job\_client\_id = Clients.client\_id

GO

select \* from v\_clientTutorAppInfo

GO

create view v\_tutorjobs as

select appointment\_tutor\_id,

count(appointment\_id) as job\_count

from Appointments

group by appointment\_tutor\_id

GO

select \* from v\_tutorjobs

GO

create view v\_maxpay as

select appointment\_tutor\_id, pay\_rate

from Appointments

GO

select \* from v\_maxpay

GO

drop view if exists v\_number\_jobs\_posted

GO

create view v\_number\_jobs\_posted as

select c.firstname + ' ' + c.lastname as Client\_Name, Count(job\_client\_id) as Jobs\_Posted

    from Jobs

        join clients c on c.client\_id = job\_client\_id

        group by job\_client\_id, c.firstname,c.lastname

GO

select \* from v\_number\_jobs\_posted

    order by Jobs\_Posted DESC

GO

drop view if exists inactive\_tutors

GO

create view inactive\_tutors as

select tutor\_id, t.firstname, t.lastname

    from Tutors t

        left outer join Appointments a on t.tutor\_id = a.appointment\_tutor\_id

        where appointment\_id is NULL

GO

select \* from inactive\_tutors

**Data Question Answers**

* Which individuals posted the most jobs?

A screenshot of a computer

Description automatically generated with medium confidence

* Which tutor has taken the most jobs?

A screenshot of a computer

Description automatically generated

* Who are the inactive tutors on the platform?

A screenshot of a computer

Description automatically generated with medium confidence

* Create a table which includes the client information, tutor information, and appointment date.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Which tutor has the highest pay rate?

A screenshot of a computer

Description automatically generated

* What are the average ratings for each tutor?

A screenshot of a computer

Description automatically generated

* Create a query that returns the jobs count grouped by assignment type.

A screenshot of a computer

Description automatically generated

**Summary**

Following the database design steps we learned in class made doing this final project a relatively smooth process. The use of our conceptual and logical models was vital in the creation of our database. Planning out how we wanted our application to work created a clear path when it came to implementing our idea. Using CRUD and the Up/Down script process allowed us to create our database in an organized manner as well as create stored procedures and use SELECT and VIEW functions to further update, explore, and analyze our data.

**Reflection**

Prior to taking this class, I took a similar database and management course that dealt with SQL as an undergraduate so I felt this class would serve as a bit of a refresher. However, I learned more new concepts than I thought I would, especially when it came to creating an efficient, working database. The most important thing I think I learned in this class is the use of conceptual and logical data models as well as entity relationships. Using these concepts made making a database so much easier and faster because before you start the implementation phase, you know what each of your tables will need so that they interconnect seamlessly, minimizing the amount of trial and error you may encounter. One thing I would do differently if I were to do this project over again is incorporate the use of excel, R, or access to provide more visualizations for our answers to our data questions. Another thing I would have done differently is include more data in our tables to get more variety in the answers to our data questions.

Meeting log

|  |  |  |
| --- | --- | --- |
| Date and meeting duration | Attendees | Purpose/What we achieved |
| 5/4/23 (30 mins) | James Du  Dale Deans | Got together to discuss viable topics and datasets for the project. |
| 5/8/23 (11 mins) | James Du  Cynthia Scotton  Dale Deans | Decided our topic |
| 5/18/23 (1 hr) | James Du  Cynthia Scotton  Dale Deans | Go Over Conceptual Data Model,  Logical Data Model  Assigning Table Responsibilities. |
| 5/25/23 (26 mins) | James Du  Dale Deans | Went over data types for the columns  Divided code work to be assigned |
| 6/1/23 (5 mins) | James Du  Cynthia Scotton  Dale Deans | Had a brief meeting to update each other on where we’re at with our assigned portions of code. |
| 6/8/23 (27 mins) | James Du  Cynthia Scotton  Dale Deans | Checked to see if our combined code works. Also discussed the powerpoint. |
| 6/15/23 (46 mins) | James Du  Cynthia Scotton  Dale Deans | Divided business questions to complete, moved on to completing paper and slide. |