The database used for this project is an AWS RDS database. The security for this database is managed so that it can only be accessed through our ec2 instance. This means that in order to connect to the database, the ec2 instance must also be running.

Accessing the Database

First, the database should be running. Check the RDS page in the AWS console to make sure that the database is running. To connect directly through the ec2 instance, you must ssh into the instance, which will require the .pem file. Once connected to the ec2 instance, you can use the command:

```
mysql -h
database-1.cluster-cf5kjev2ovc7.us-east-1.rds.amazonaws.com -u
admin -p IGA_DB
```

to connect to the database. (Although it appears on separate lines in this doc, this is one command and should be one line)

To connect through MySQL Workbench, first make sure the database and the ec2 instance are running. The first time you connect, you will need to add a new connection to MySQL workbench.

- Select “Standard TCP/IP over SSH” as the connection method
- The SSH Hostname is ec2-23-21-141-182.compute-1.amazonaws.com
- The SSH Username is “ec2-user”
- The SSH key file is the .pem file

Tables

The first three tables are TestEssays, TrainingEssays, and ValidSet. These tables exactly match the corresponding .tsv files that can be found in the data folder in the GitHub repository. As of right now, these tables are not being used, and instead the programs are accessing the .tsv files.

UserEssays

This table is used to store any text only essays. As of right now, the website only supports essays that are uploaded as word documents or pdfs, so this table is not currently used. If it is deemed necessary to allow users to submit only the text of their essay, this table should be used to store those essays. For that to happen, it would need to have attributes such as date and email added to it so that it can work with the most recent version of the website.

If necessary, this table can be recreated with the following command:

```
CREATE TABLE IF NOT EXISTS UserEssays (essay_id INT
AUTO_INCREMENT PRIMARY KEY, essay TEXT, grade FLOAT, feedback
TEXT);
```

essay_id: This is used as the primary key for the table. Since it has the auto increment property, this value will be automatically assigned to any rows that are added to this table. You do not need to manually assign this value.

essay: The text of the essay.

grade: The grade given to the essay.

feedback: The feedback given to the essay.

UserFiles

This table stores essays that are uploaded as files, which can either be .docx files or .pdf files. It stores the binary data of the files, which can be accessed later on. To see how to access the binary data stored in this table, look at how the results route in app.py retrieves the essay's text from the database. It's important that you delete any temp files that you create using os.remove(), so that the ec2 instance does not run out of storage.

If necessary, this table can be recreated with the following command:

```
CREATE TABLE IF NOT EXISTS UserFiles (file_id INT AUTO_INCREMENT
PRIMARY KEY, name CHAR(50), data LONGBLOB, grade FLOAT, feedback
TEXT, error TEXT, email CHAR(70), upload_date DATETIME);
```

file_id: This is used as the primary key for the table. Since it has the auto increment property, this value will be automatically assigned to any rows that are added to this table. You do not need to manually assign this value.

name: The name of the file that was uploaded.

data: The binary data of the file.

grade: The grade given to the essay.

feedback: The feedback given to the essay.

error: The error information that is displayed on the website. Provides some insight on why points were taken off.

email: The email address entered by the user when they uploaded this file. An email containing a link where the user can view their essay is sent to this email. This email is also used to retrieve the essay from the database when the user clicks on the link

upload_date: The date and time that the user uploaded their essay. This serves two purposes. The first is that every day at 5:25 AM UTC (12:25 AM CDT, 11:25 PM CST) the database checks for essays that are at least a week old, and deletes them. The second is that this is used to retrieve the essay from the database when the user clicks on their link, which means if they have multiple essays currently in the database, they will have a unique link for each one.

Events

In order to delete essays that are at least a week old every day, an event is used. For the event to be executed, the event scheduler must be on. For an RDS database, this is done by creating a parameter group in the AWS console that has the parameter event_scheduler set to 1, and then associating this parameter group with the database.

This is the event used to delete the essays:

```
CREATE EVENT delete_old_essays
  ON SCHEDULE
     EVERY 1 DAY
     STARTS str_to_date( date_format(now(), '%Y%m%d 0525'),
'%Y%m%d %H%i' ) ON COMPLETION PRESERVE ENABLE
  DO
     DELETE FROM UserFiles WHERE upload_date < DATE_SUB(NOW(),
INTERVAL 7 DAY);
```

EVERY 1 DAY is what makes this event executes every day.

STARTS is used to start the event on the day that it is created by using now(), and 0525 is the time that it will be executed each day.

ON COMPLETION PRESERVE ENABLE is used to ensure that this event stays active.

DATE_SUB(NOW() INTERVAL 7 DAY) subtracts 7 days from the current date. Any essays with an upload_date lower than this date will be deleted from the database.