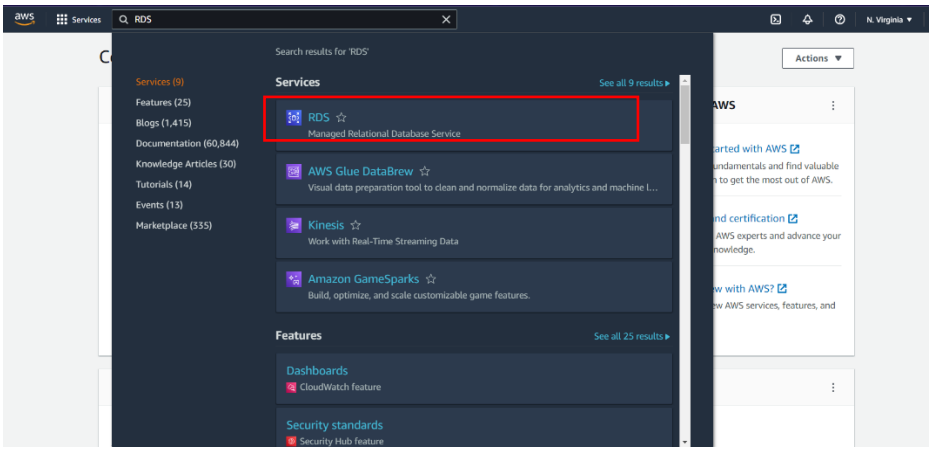
CEHD Aggie Educator Portal

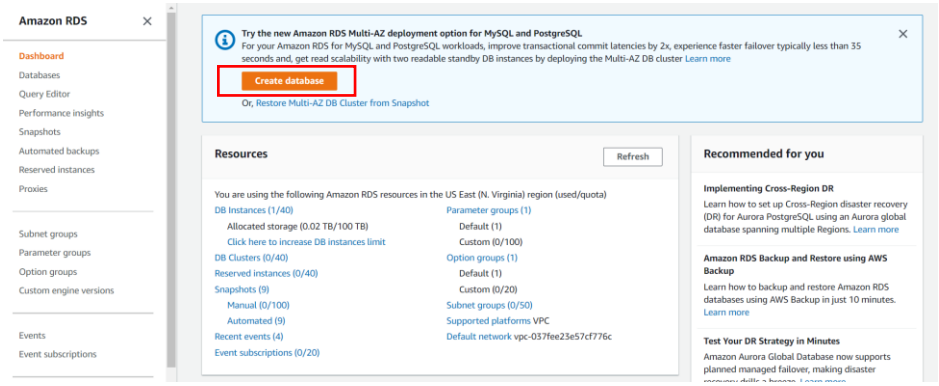
Database Setup

The application used database server from AWS cloud platform. Below steps shows how to create a database server in AWS.

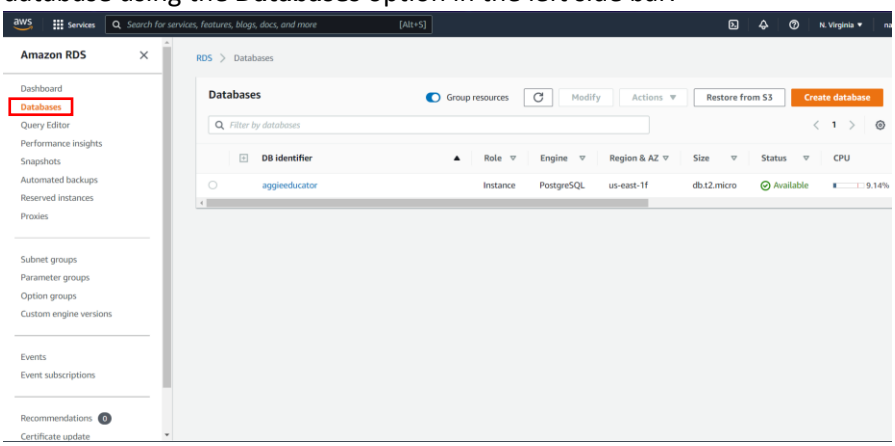
1. Create an AWS account (free tier) and login to the account. If you already have an AWS account, use the link https://console.aws.amazon.com/ to go to the console home page.
2. Search for RDS in the search bar and click on it as shown



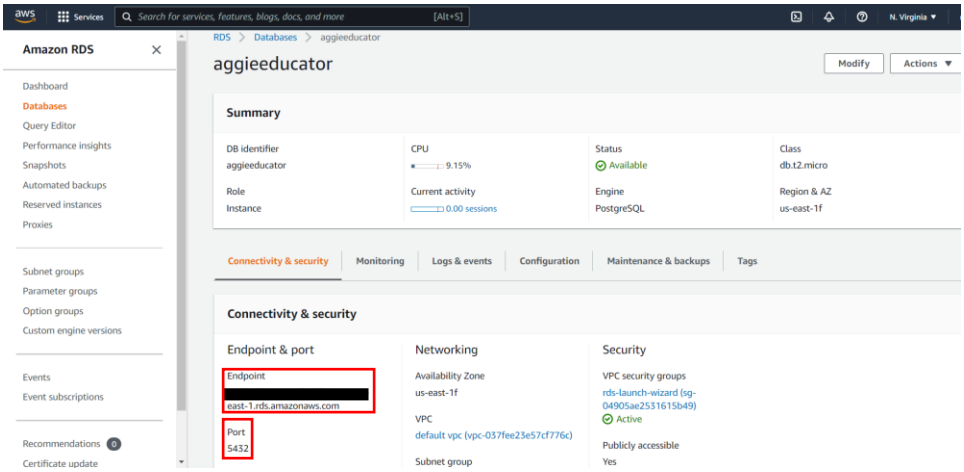
3. Click on Create database option to create a new database



4. Use the engine type as Postgres and fill up the details as per the requirements. You can keep everything as default, change the **DB cluster identifier** with your database name, create username and password for your database. You can keep the connectivity option as it is. Then click on **Create database**.
5. In few minutes, you will be able to see your database created. You can navigate to your new database using the **Databases** option in the left side bar.



6. Once the status of the database becomes **available**, you can start using your database.
7. Click on your created database to get the endpoint to connect with your database.

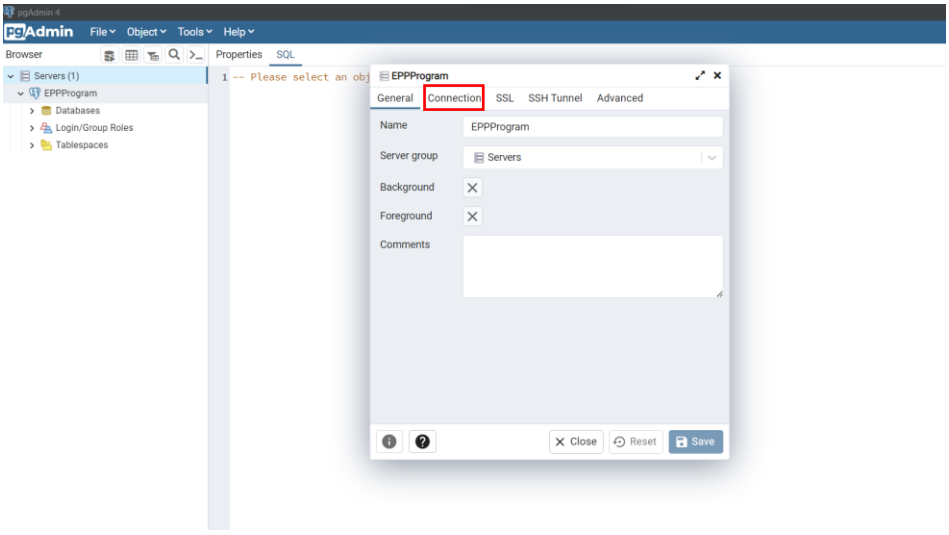


8. Use the endpoint provided as the database hostname, password used to setup the database as the database password, username set as the database username, port provided as the database port, and add a database name while connection.

Connect to Postgres database server

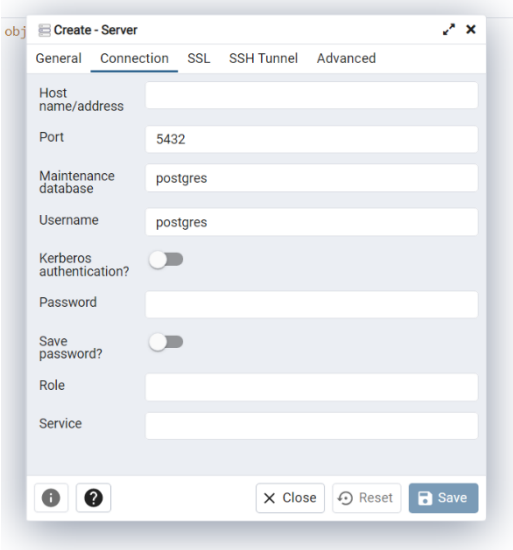
Follow the steps below to connect to postgres database created in AWS via pgadmin GUI based tool.

1. Download pgadmin tool by going to https://www.pgadmin.org/download/
2. Install pgadmin in your system. Once installed, open the pgadmin and follow the below steps
3. Use the **create server** option to create/connect to a new server
4. In the general section, add the name of the DB created in the previous section in AWS. It is a mandatory field.



5. In the **connection** tab above, fill in the details from the AWS postgres server created in the above section. Type in the endpoint provided in the hostname, username and password in the username and password section respectively and save the configuration.

CEHD Aggie Educator Portal



6. Using the configuration saved in the above steps, the postgres server created in AWS can be accessed. Using pgadmin, table creation, data addition, etc can be performed.

Development environment setup

- The code of the project, CEHD Aggie Educator Portal, can be accessed from the link: https://github.com/sanjaynayak1628/CEHD_Aggie_Educator_Portal. Use it to clone or fork from the github repository.
- https://github.com/sanjaynayak1628/CEHD_Aggie_Educator_Portal/tree/main/code/cehd contains the main code. The documentations of the project, including API documentation, getting started documents, and other reports can be found at https://github.com/sanjaynayak1628/CEHD_Aggie_Educator_Portal/tree/main/documentation/Spring2022
- The **.env** file contains the database connection details and SMTP connection details. Use this file to update with the postgres database and smtp server details being used.
- The **requirements.txt** file contains the python libraries being used for the development and test purposes.
- **psycopg2** library is essential for connection to postgres database. For installation of this library in MAC, please use the library **psycopg2-binary** or its equivalent library.
- The libraries **behave**, **behave-django**, **factory_boy**, **selenium** is used for the BDD test cases development. The library **coverage** is used for calculating the coverage of the test cases in the application code.
- The **Procfile** is used for Heroku deployment. Detailed explanation and steps for deployment of Django application in Heroku can be found at https://devcenter.heroku.com/articles/django-app-configuration
- The **config.json** contains configuration denoting the start and end dates for different terms – spring, fall, and summer. This can be edited to accommodate the start and end dates of different terms.
- PyCharm or VSCode can be used for development purposes. Connect the IDE used with your GitHub repo to push your changes. **Please don't push to the main original**

repository. Fork it into your repo and work on it. Steps on how to fork can be found in the git documentation or on the internet.

Database Migration

Once the development environment is setup, use the command line to execute some commands to create tables in the specified database. Here, we are using Django commands to create and migrate tables into the database.

Create migrations of the apps: `python manage.py makemigrations`
Migrate the migrations into the database: `python manage.py migrate`

Reference: https://docs.djangoproject.com/en/4.0/topics/migrations/

Heroku Deployment

Please follow the steps listed down in the link, https://devcenter.heroku.com/articles/git, to properly configure the Heroku CLI with git bash and use it to push your application to Heroku server. Some of the important steps are included below.

1. `heroku login -i` (Authenticate and connect to your heroku account)
2. Go to the folder where you have your github repo set
3. `heroku create`
4. If more than one apps present in your heroku account and you want to use a particular app, use the following command else directly go to step 5
5. `heroku git:remote -a \<app-name\>`
6. `heroku config:set DISABLE_COLLECTSTATIC=0` (To push to heroku from a non-master branch use the following command)
7. `git subtree push --prefix \<sub folder path to Django app\> heroku \<non-master branch-name\>:main`
   (This command is used to deploy code to Heroku from a branch other than main, where the code is in the sub-directory code/cehd)

BDD (Behavioral Driven Development) Setup

For running the BDD tests, the libraries **behave**, **behave-django**, **factory_boy**, and **selenium** needs to be installed on the system. Since the BDD of the application uses chrome as its web driver, it needs to be downloaded before running the BDD test cases. Use the link, https://chromedriver.chromium.org/downloads, to download the chrome web driver.

The path to the chrome web driver .exe file needs to be provided in the **environment.py** file in **/code/cehd/features** folder. It can also be found in the following link - (https://github.com/sanjaynayak1628/CEHD_Aggie_Educator_Portal/blob/main/code/cehd/features/environment.py)

The **code/cehd/features/** folder contains all BDD test cases with respective features for all the six different features provided in the application. The **steps/** folder inside it contains the steps required by the BDD test cases.

The BDD test cases can be run using the command: `python manage.py behave`

CEHD Aggie Educator Portal

To create a report of the BDD test cases, use command: `python manage.py behave > <file-name.txt>`

TDD (Test Driven Development) Setup

Individual tests for the APIs created for the applications can be found at **tests.py** files in each of the app folders in **/code/cehd/** directory. Test cases were written for the **cooperating**, **supervisor**, and **time_logs** apps which corresponds to the cooperating teacher views, supervisor views, and student time logs views respectively.

TDD can be run using the command: `python manage.py test`

Code Coverage Setup

Code coverage report of the application using the test cases written in the **tests.py** files can be created using the following commands:

1. `coverage run manage.py test`
2. `coverage report`
3. `coverage html`

.coveragerc file present in the **/code/cehd/** folder is used to store the coverage report generated using the above commands in the directory specified.

Coverage report: 93%

filter...

Module	statements	missing	excluded	coverage
D:\MS Admits\TAMU\Semester\second\Software Engineering\CEHD Aggie Educator Portal\CEHD_Aggie_Educator_Portal__init__.py	0	0	0	100%
D:\MS Admits\TAMU\Semester\second\Software Engineering\CEHD Aggie Educator Portal\CEHD_Aggie_Educator_Portal\code__init__.py	0	0	0	100%
__init__.py	0	0	0	100%
cehd__init__.py	0	0	0	100%
cehd\settings.py	31	0	0	100%
cehd\urls.py	3	0	0	100%
cooperating__init__.py	0	0	0	100%
cooperating\admin.py	1	0	0	100%
cooperating\apps.py	4	0	0	100%
cooperating\migrations__init__.py	0	0	0	100%
cooperating\models.py	0	0	0	100%
cooperating\tests.py	70	0	0	100%
cooperating\urls.py	3	0	0	100%
cooperating\views.py	165	14	0	92%
core__init__.py	0	0	0	100%
core\admin.py	1	0	0	100%
core\apps.py	3	0	0	100%
core\migrations\0001_initial.py	6	0	0	100%
core\migrations__init__.py	0	0	0	100%
core\models.py	34	0	0	100%
core\tests.py	1	0	0	100%
epp_program__init__.py	0	0	0	100%
epp_program\admin.py	1	0	0	100%
epp_program\apps.py	3	0	0	100%
epp_program\migrations\0001_initial.py	5	0	0	100%
epp_program\migrations__init__.py	0	0	0	100%
epp_program\models.py	14	0	0	100%
epp_program\tests.py	1	0	0	100%
epp_student__init__.py	0	0	0	100%
epp_student\admin.py	1	0	0	100%
epp_student\apps.py	3	0	0	100%
epp_student\migrations\0001_initial.py	6	0	0	100%
epp_student\migrations__init__.py	0	0	0	100%
epp_student\models.py	37	0	0	100%
epp_student\tests.py	1	0	0	100%
epp_student\urls.py	3	0	0	100%
epp_student\views.py	3	1	0	67%

4/25/22, 9:40 PM Coverage report

Module	statements	missing	excluded	coverage
student_placements\admin.py	1	0	0	100%
student_placements\apps.py	4	0	0	100%
student_placements\migrations\0001_initial.py	6	0	0	100%
student_placements\migrations\0002_alter_studentplacements_uin.py	5	0	0	100%
student_placements\migrations\0003_studentplacements_semester_year.py	4	0	0	100%
student_placements\migrations__init__.py	0	0	0	100%
student_placements\models.py	24	0	0	100%
student_placements\tests.py	24	0	0	100%
student_placements\urls.py	3	0	0	100%
student_placements\views.py	142	21	0	85%
supervisor__init__.py	0	0	0	100%
supervisor\admin.py	1	0	0	100%
supervisor\apps.py	4	0	0	100%
supervisor\migrations__init__.py	0	0	0	100%
supervisor\models.py	1	0	0	100%
supervisor\tests.py	44	0	0	100%
supervisor\urls.py	3	0	0	100%
supervisor\views.py	84	1	0	99%
test__init__.py	0	0	0	100%
test\factories__init__.py	0	0	0	100%
time_logs__init__.py	0	0	0	100%
time_logs\admin.py	1	0	0	100%
time_logs\apps.py	4	0	0	100%
time_logs\migrations\0001_initial.py	7	0	0	100%
time_logs\migrations\0002_alter_timelogs_notes.py	4	0	0	100%
time_logs\migrations\0003_alter_timelogs_hours_submitted.py	4	0	0	100%
time_logs\migrations\0004_timelogs_semester_year.py	4	0	0	100%
time_logs\migrations__init__.py	0	0	0	100%
time_logs\models.py	19	0	0	100%
time_logs\serializers.py	16	0	0	100%
time_logs\tests.py	76	0	0	100%
time_logs\urls.py	3	0	0	100%
time_logs\views.py	180	12	0	93%
utils__init__.py	0	0	0	100%
utils\emails.py	44	1	0	98%
utils\utility.py	35	0	0	100%
Total	1237	88	0	93%

coverage.py v6.3.2, created at 2022-04-25 18:49 -0500

CodeClimate Usage

CodeClimate is used to check the quality of the code. It is free to use and requires sign in with GitHub. Use the link, https://codeclimate.com/quality/pricing/, to sign up to codeclimate using github. Once you sign up, add the repository that needs to be connected to the codeclimate for code quality check.

The following link, https://codeclimate.com/github/sanjaynayak1628/CEHD_Aggie_Educator_Portal, can be used to check the code quality of this application.

CEHD Aggie Educator Portal