Deployment Documentation

To access the Google Cloud SQL database locally:

- Download the Cloud SQL Auth proxy here (https://cloud.google.com/sql/docs/mysql/quickstart-proxy-test). Download until your base directory (don't do it in middleground, do it at least one level below)
- Get '*middleground-314106-b99091a90bd9.json*' from project owner
- In the base directory, run this command: *./cloud_sql_proxy -instances=middleground-314106:us-central1:final-instance-id=tcp:3306 -credential_file=middleground-314106-b99091a90bd9.json*
- In mgflask/db.py, uncomment the line that says "Use this for to access the Google Cloud SQL db"
- Get 'creds.json' file from project owner and place it at the same directory level as 'server'
- Start up the virtual environment and do '*flask run*'. Check out localhost:5000/news for articles

To deploy the Flask app onto Google Cloud:

- For an overview, follow the instructions here (https://medium.com/@dmahugh_70618/deploying-a-flask-app-to-google-app-engine-faa883b5ffab)
- Install and configure the GCloudSDK here (https://medium.com/google-cloud/cloud-iot-step-by-step-quality-of-life-tip-the-command-line-ce23046867d4#016c). You only need to do the steps that are located at the top of the article
- Install the App Engine extension: `gcloud components install app-engine-python`
- Create an app.yaml file that is on the same directory level as 'server'. This is to let App Engine some settings to run
    - Add an entrypoint to locate the Flask 'app'. On default, App Engine searches for 'main.py' so if you don't have one, make sure to specify
- Set the project as your gcloud default: `*gcloud config set project middleground-314106*`
- Enable the Cloud Build API: `gcloud services enable cloudbuild.googleapis.com`
- Deploy to App Engine: `gcloud app deploy`

To create a Google Cloud SQL database instance:

- Follow the steps here (https://www.geeksforgeeks.org/setting-up-google-cloud-sql-with-flask/)
- When it comes to creating the engine, follow this to create the SqlAlchemy Database URI: (https://github.com/mtdefelice/flask-gcp-sql)

Troubleshooting the deployed Flask app:

- Go to the Google Cloud Console
- Find Logging and look at Logs Explorer
- If there are errors with initially deploying, make sure to double check the app.yaml
- If the logs are saying "cannot connect the MySQL" or something along those lines, make sure you have the right database URI inside db.py (LOCAL_SQLALCHEMY_DATABASE_URI vs LIVE_SQLALCHEMY_DATABASE_URI)
- If there are issues about the Google Cloud SQL database, refer below to "**Populating the Google Cloud SQL database"**

Updating the database schema (models):

- If you are testing with the local database (SQLite), simply running 'flask run' will recreate the database structure

- **Option 1:** If you are accessing the Google Cloud SQL database locally or deployed
    - Go to the Google Cloud Console
    - In the ‘middle ground’ project, go to ‘SQL’
    - Find the ‘final-instance-id’ instance and navigate to ‘Databases’
    - Delete the database that needs to be updated
    - If local: have the Cloud SQL Auth proxy running and do `*flask run*`. This will automatically recreate the database structure based on models.py
    - If deployed: Run `gcloud app deploy`
- **Option 2:** Edit the existing table directly by using MySQL terminal
    - Run: `*gcloud sql connect flask-demo --user=<USERNAME>*`
    - Show all databases: `*show databases;*`
    - Select the database that needs to be updated: `*use <DB_NAME>;*`
    - Show all tables: `*show tables;*`
    - Example of altering the ‘user’ table: `*alter table user modify username varchar(64) NOT NULL;*`
    - IMPORTANT: run `describe <TABLE_NAME>;` to make sure the table has been updated properly

Populating the Google Cloud SQL database:

- Have the Cloud SQL Auth proxy running
- Follow the steps listed on the README.md
- Common issues:
    - Data for column ‘whatever_column’ is too long
        - Edit the database schema to accommodate for the extra length. MySQL is very strict on their rules