Guidelines for Deploying the Application

Prerequisites:

- Heroku Account
- GitHub Organization
- GitHub OAuth Credentials

This application will streamline the process of creating auto graded assignments by providing a GUI to generate new assignment GitHub repositories and interactively create new tests. All the Github repositories need to be created under the organization for the course.

Creating the Organization:

First you need to have a GitHub account. Go to your GitHub account, and follow the following steps-

1. In the upper-right corner of any page on GitHub, click your profile photo, then click Settings.
2. In the "Access" section of the sidebar, click Organizations.
3. Next to the "Organizations" header, click New organization.
4. Follow the prompts to create your organization.

In the Organization, you may want to give different roles to the different members (You don't want to give write permissions to everyone). There are some predefined roles in GitHub-

- **All-repository read**
- All-repository write
- All-repository triage
- All-repository maintain
- All-repository admin
- CI/CD admin

Lastly, your organization must contain a fork of the Autograder Core repository. This repository contains core autograder functionality that is common to all assignments. When a student submits their work to Gradescope, both the assignment-specific repository, as well as this repository are cloned into the autograder environment to run tests on the submission.

To fork this repository:

1. Navigate to the GitHub repository: https://github.com/AutograderFrontend/autograder-core

2. Click either the “Fork” button at the top of the window, or the “fork this repo” link in the README
3. Set the “Owner” field to the name of your newly-created GitHub Organization
4. You may optionally change the repository name, but you must then make sure that your GITHUB_AUTOGRADER_CORE_REPO environment variable (discussed below in the Heroku setup documentation) matches the new name.
5. Click “Create fork”

Optional Steps:

Assign an organization role:

1. In the upper-right corner of GitHub, select your profile photo, then click Your organizations.
2. Next to the organization, click Settings.
3. In the "Access" section of the sidebar, click Organization roles, then click Role assignments.
4. Click New role assignment.
5. Search for users or teams that you want to assign a role to, then select the role you want to give to these users and teams.
6. Click Add new assignment.

A user or team can have multiple organization roles. However, you can only assign one role at a time.

View an organization role assignments:

1. In the upper-right corner of GitHub, select your profile photo, then click Your organizations.
2. Next to the organization, click Settings.
3. In the "Access" section of the sidebar, click Organization roles, then click Role assignments.
4. Optionally, to filter by role assignments for users, click the Users tab. To filter by role assignments for teams, click the Teams tab.
5. To view role assignments, to the right of the user or team, click NUMBER roles.

Delete an organization role:

1. In the upper-right corner of GitHub, select your profile photo, then click Your organizations.
2. Next to the organization, click Settings.
3. In the "Access" section of the sidebar, click Organization roles, then click Role assignments.
4. Optionally, to filter by role assignments for users, click the Users tab. To filter by role assignments for teams, click the Teams tab.

5. To delete a role, to the right of the role, click NUMBER roles. Then click Remove.
6. In the pop-up window, click Remove.

Setting up Heroku App:

To create a Heroku application, we first need a Heroku account.

Creating a Heroku account

1. Visit Heroku's Signup Page.
2. Fill in the required details (name, email, password, etc.).
3. Verify your email address via the confirmation link sent to your inbox.
4. Log in to the Heroku dashboard at https://dashboard.heroku.com/.

Install the Heroku CLI

1. Download the Heroku CLI from the Heroku CLI Download Page.
   OR
2. Install it based on your operating system:
   macOS: Use Homebrew:
   ```
   brew tap heroku/brew && brew install heroku
   ```
   Ubuntu/Debian: Use Snap:
   ```
   sudo snap install --classic heroku
   ```
   Windows: Download and run the installer from the Heroku CLI page.
3. Verify the installation:
   ```
   heroku --version
   ```

Login to the Heroku CLI

- Log in to your Heroku account: `heroku login`
- This will open a browser window for you to log in.

Create a Heroku Application

In your Rails project directory, create a new Heroku app:

```
heroku create <app-name>
```

NOTE : If you don't provide `<app-name>`, Heroku will generate a random name for your app.

Following the above steps, you will have a new application created on Heroku.

Change the stack

- The application requires Git to be installed in the Heroku environment for functionality like cloning the repository to local.
- The Heroku application stack will automatically be set to heroku-24, but this stack does not have Git in it.
- To install Git, change the stack to heroku-22 where Git is already included using the command: `heroku stack:set heroku-22 --app <your-app-name>`
- To ensure that Git is installed, execute the command: `heroku run “git --version”`

Connecting the database to Heroku App

Once the application is created on Heroku, we need to connect the database to the application.

1. All apps on Heroku use the PostgreSQL database. For Ruby/Rails apps to do so, they must include the `pg gem`. However, we don't want to use this gem while developing, since we're using SQLite for that.
2. Make sure that the Gemfile has the following two configurations:
```
a. group :production do
        gem 'pg' # for Heroku deployment
   end
b. group :development, :test do
        gem 'sqlite3'
   end
```
3. Run `bundle install` to download the dependencies.
4. Once, the gem is installed the PostgresSQL database plan needs to be attached as an add-on to the application. The database can be added on through Heroku CLI or Heroku interface.
    a. **Heroku CLI:**
        i. For the current load, Heroku’s essential-0 plan is sufficient. The Essential-0 plan for Heroku Postgres is a production-grade database plan that offers 1GB of storage and a connection limit of 20.
        ii. Run the following command to add the essential-0 plan to your Heroku app: `heroku addons:create heroku-postgresql:essential-0`

        iii. This creates a more robust database suitable for production use with increased storage and performance.

    b. **Heroku interface**
        i. Log In to the Heroku Dashboard
            1. Go to Heroku Dashboard.
            2. Log in with your Heroku credentials.
        ii. Select your application that we created earlier
            1. Locate and click on your application in the list of apps on the dashboard.
        iii. Navigate to the Resources Tab
            1. Once on your app's page, click the Resources tab in the top menu.
            2. This tab shows all the add-ons currently attached to your app.
        iv. Search for the Add-on
            1. In the Add-ons section, type the name of the add-on you want to attach, such as Heroku Postgres.
            2. Select Heroku Postgres from the dropdown list.
        v. Select the Plan
            1. A pop-up will appear asking you to choose a plan. Select Essential-0 (or your desired plan).
            2. Confirm your choice by clicking the Submit or Provision button.
        vi. Verify the Add-on
            1. After provisioning, the new add-on will appear in the Add-ons section of the Resources tab.
            2. You can click on the add-on's name to open its management dashboard for further configuration and monitoring.
        vii. NOTE: The provisioning might take some time, so wait for a few minutes or check the progress on Heroku dashboard.

5. Heroku will automatically set the **DATABASE_URL** environment variable for your app once the add-on is attached.
6. To verify that the environment variable is setup, run the command `heroku config` and look for an entry like this:
   `DATABASE_URL: postgres://username:password@hostname:port/dbname`
7. To verify on Heroku interface, navigate to the **Settings** tab and click **Reveal Config Vars**. You should see the key **DATABASE_URL** with the correct URL value. (Check Edit Config Vars section to know more)
8. Once the database is attached to the application, we need to configure it for production. To do this, add the following part in config/database.yml

```
production:
    <<: *default
    database: <%= ENV['DATABASE_URL'] %>
```

Set Config variable in Heroku:

You can manage config variables via Heroku CLI or through the interface.
Using the Heroku CLI:
The heroku config commands of the Heroku CLI makes it easy to manage your app's config vars.
To view current config values:

```
$ heroku config
```

This command shows all current config values. You can also query specific values.

```
$ heroku config:get GITHUB_USERNAME
```

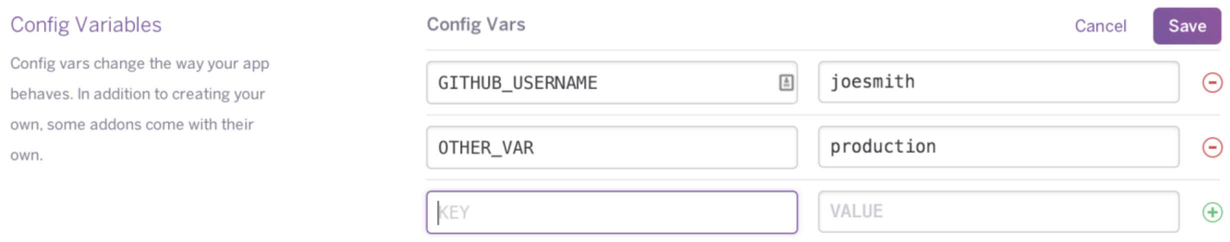
To set a config variable use:

```
$ heroku config:set GITHUB_USERNAME=joesmith
```

Remove a config variable:

```
$ heroku config:unset GITHUB_USERNAME
```

You can also access them through Heroku Dashboard interface:
You can edit config vars from your app's Settings tab in the Heroku Dashboard:


Config Variables

Config vars change the way your app behaves. In addition to creating your own, some addons come with their own.

Config Vars

KEY	VALUE
GITHUB_USERNAME	joesmith
OTHER_VAR	production

Currently we have the following variables:

- GITHUB_TEMPLATE_REPO_URL: 'philipritchey/autograded-assignment-template'
  [This is the template based on which the professor will create the assignment.]
- GITHUB_COURSE_ORGANIZATION: 'AutograderFrontend' (The way code is setup keep the same organization name even if you are creating a new one)
- ASSIGNMENTS_BASE_PATH: /app/'assignment-repos/'
  [This is the base path where the assignments will be cloned.]
- GITHUB_AUTOGRADER_CORE_REPO: autograder-core

Deploying the Application:

Before deploying the application, there are certain prerequisites:

1. A Procfile

2. Secret Key Base

Generate a Procfile

1. If the Procfile is not present, create a file named “Procfile” with no extension
2. Generate a puma config file if it does not exist using the command: `bundle exec puma --config config/puma.rb`
3. Ensure that the Procfile contains the following line that executes the Puma web server
   ```
   web: bundle exec puma -C config/puma.rb
   ```

Add a Secret Key Base

Generate and add the secret key base to the application:

```
heroku config:set SECRET_KEY_BASE=$(rails secret)
```

Deployment

Follow the steps to deploy the application:

1. In your project directory, execute the command `heroku login` to connect to Heroku.
2. Add heroku as a remote to your Git repository:
   ```
   heroku git:remote -a <heroku-app-name>
   ```
3. Verify that the heroku is added to remote by executing: `git remote`
4. Push your changes to Heroku: `git push heroku main`

Post Deployment:

Once the changes are pushed to heroku main, complete the following post-deployment tasks.

Migrations

1. Run database migrations: `heroku run rails db:migrate`
2. If needed, seed the database: `heroku run rails db:seed`
3. Restart heroku dynos: `heroku restart`
4. Verify that the app is running correctly: `heroku open`

Once the app is deployed, the application needs to be connected with Github OAuth to authenticate the users.

Get OAuth Credentials

Create an OAuth app: You can create and register an OAuth app under your personal account or under any organization you have administrative access to. Here, you should open it under the organization.

- In the upper-right corner of any page on GitHub, click your profile photo, then click Settings.
- In the left sidebar, click Developer settings.
- In the left sidebar, click OAuth apps.
- Click New OAuth App. (If you haven't created an app before, this button will say, Register a new application.)
- In "Application name", type the name of your app.
- In "Homepage URL", type the full URL to your app's website (If you have deployed in Heroku, provide that URL). For example:
    - https://csce-606-autograder-frontend-9219bed98016.herokuapp.com
- Optionally, in "Application description", type a description of your app that users will see.
- In "Authorization callback URL", type the callback URL of your app.
    - For example: https://csce-606-autograder-frontend-9219bed98016.herokuapp.com/auth/github/callback
    - NOTE: Ensure that the callback URL is configured in `routes.rb`
- If your OAuth app will use the device flow to identify and authorize users, click Enable Device Flow. For more information about the device flow, see "Authorizing OAuth apps."
- Click Register application.

After registering the OAuth you will get a `GITHUB_CLIENT_ID, and GITHUB_CLIENT_SECRET` which you need to use as a config variable.

Add GITHUB_CLIENT_ID, and GITHUB_CLIENT_SECRET to the environment variables in Heroku.

Troubleshooting:

1. If some functionality is not working correctly on deployed application then check the logs to find the root cause using the command: `heroku logs --tail`