This is a Django Full Stack application with a front-end and back-end component connected to a cloud database.

# Design

Backend: Python – Django framework

Database: MySQL DB

API endpoint: Python

Frontend: JavaScript – React Framework

# Setup

We first initialize a virtual environment for our python backend using the terminal command: python -m venv env. This is to create a virtual environment to work with and keep dependencies separate from our usual working environment. We then activate it using fullstack/Scripts/activate

We now need to create a requirements.txt file for our dependencies. We use this document for deployment to install the libraries used in our backend. To install the dependencies, we use pip install -r requirements.txt.

asgiref

Django

django-cors-headers

djangorestframework

djangorestframework-simplejwt

PyJWT

pytz

sqlparse

psycopg2-binary

python-dotenv

# Backend Development

## Django setup:

The **[makemigrations](https://docs.djangoproject.com/en/5.0/ref/django-admin/" \l "django-admin-makemigrations)** command looks at all your available models and creates migrations for whichever tables don’t already exist. [**migrate**](https://docs.djangoproject.com/en/5.0/ref/django-admin/#django-admin-migrate) runs the migrations and creates tables in your database, as well as optionally providing [much richer schema control](https://docs.djangoproject.com/en/5.0/topics/migrations/).

**makemigrations[¶](https://docs.djangoproject.com/en/5.0/ref/django-admin/" \l "makemigrations" \o "Permalink to this headline)**

django-admin makemigrations [app\_label [app\_label ...]][**¶**](https://docs.djangoproject.com/en/5.0/ref/django-admin/#django-admin-makemigrations)

Creates new migrations based on the changes detected to your models. Migrations, their relationship with apps and more are covered in depth in [the migrations documentation](https://docs.djangoproject.com/en/5.0/topics/migrations/).

Providing one or more app names as arguments will limit the migrations created to the app(s) specified and any dependencies needed (the table at the other end of a **ForeignKey**, for example).

**migrate**[**¶**](https://docs.djangoproject.com/en/5.0/ref/django-admin/#migrate)

django-admin migrate [app\_label] [migration\_name][**¶**](https://docs.djangoproject.com/en/5.0/ref/django-admin/#django-admin-migrate)

Synchronizes the database state with the current set of models and migrations. Migrations, their relationship with apps and more are covered in depth in [the migrations documentation](https://docs.djangoproject.com/en/5.0/topics/migrations/).

The behavior of this command changes depending on the arguments provided:

* No arguments: All apps have all of their migrations run.
* **<app\_label>**: The specified app has its migrations run, up to the most recent migration. This may involve running other apps’ migrations too, due to dependencies.
* **<app\_label> <migrationname>**: Brings the database schema to a state where the named migration is applied, but no later migrations in the same app are applied. This may involve unapplying migrations if you have previously migrated past the named migration. You can use a prefix of the migration name, e.g. **0001**, as long as it’s unique for the given app name. Use the name **zero** to migrate all the way back i.e. to revert all applied migrations for an app.

# Frontend Development