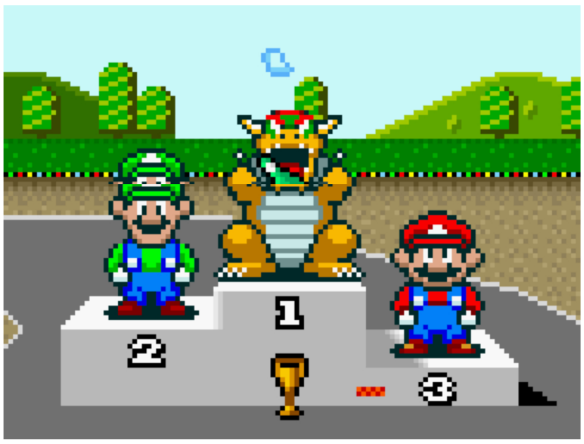
Coffee Shop Full Stack

REVIEW | CODE REVIEW 8 | HISTORY

Meets Specifications

You made it!! Congratulations!! I'm very proud of you, and you should be as well!



I wish you the best in your career and once again congratulations! 🎉

- If you feel like I can be more helpful to you and other students in any manner, please use the feedback session to drop me a hint about it.

PS: The reviewers are assigned automatically, so I guess I'm not the same one you interacted with before, just to let you know that I'm not simply ignoring your message.

Regards,
Daniel

Flask server setup

✓ **All project code has been included in a single zip file.**

The virtual env directory, pycache, and other local files are included in `.gitignore`.

The `.gitignore` file is present and excluding environment files.

✓ **The code adheres to the PEP 8 style guide and follows common best practices, including:**

- **Variable and function names are clear.**
- **Endpoints are logically named.**
- **Code is commented appropriately.**
- **The README file includes detailed instructions for scripts to install any project dependencies, and to run the development server.**
- **Secrets are stored as environment variables.**

Running the `pycodestyle` tool (which is the new name to pep-8) gives some warnings, but they are not critical and totally understandable. The code documentation is very extensive, though. 👍

PS2: Here is a link that may help understand a little bit better why using raw `except` is not a good practice.

```
dcerag@Daniels-MBP-2 ~/Downloads/fsnd-project-3-coffee-shop/backend/src
```

```
> $ pycodestyle api.py
api.py:37:5: E722 do not use bare 'except'
api.py:57:5: E722 do not use bare 'except'
api.py:134:5: E722 do not use bare 'except'
api.py:165:5: E722 do not use bare 'except'
```

✓ All `@TODO` flags in the *./backend/src/api.py* file have been completed.

The endpoints follow flask design principles, including `@app.route` decorators and request types.

The routes perform CRUD methods on the SQLite database using the simplified interface provided.

Best efforts should be made to catch common errors with `@app.errorhandler` decorated functions.

The following endpoints are implemented:

- `GET /drinks`
- `GET /drinks-detail`
- `POST /drinks`
- `PATCH /drinks/<id>`
- `DELETE /drinks/<id>`

All `@TODO` flags are fulfilled as expected; the required endpoints and error handlers are implemented.

✓ The backend can be run with `flask run` and responds to all required REST requests.

No issues found while starting the python backend app

```
dcerag@Daniels-MBP-2 ~/Downloads/fsnd-project-3-coffee-shop/backend/src
> $ flask run
 * Serving Flask app "api.py"
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [28/Apr/2020 07:27:18] "GET /drinks HTTP/1.1" 200 -
({'code': 'invalid_header', 'description': 'Authorization header is missing'}, 401)
```

Secure a REST API for applications

✓ Auth0 is set up and running at the time of submission.

All required configuration settings are included in the `auth.py` file:

- The Auth0
- Domain Name
- The Auth0 Client ID

`Auth0` information is valid and implemented through the code.

✓ A custom `@requires_auth` decorator is completed in `./backend/src/auth/auth.py`

The `@requres_auth` decorator should:

- Get the Authorization header from the request.
- Decode and verify the JWT using the Auth0 secret.
- Take an argument to describe the action (i.e., `@require_auth('create:drink')`).
- Raise an error if:
  - The token is expired.
  - The claims are invalid.
  - The token is invalid.
  - The JWT doesn't contain the proper action (i.e. `create: drink`).

The `@requires_auth` decorator is implemented 👍🏽

✓ Roles and permission tables are configured in Auth0. The JWT includes the RBAC permission claims.

Barista access is limited:

- can get drinks
- can get drink-details

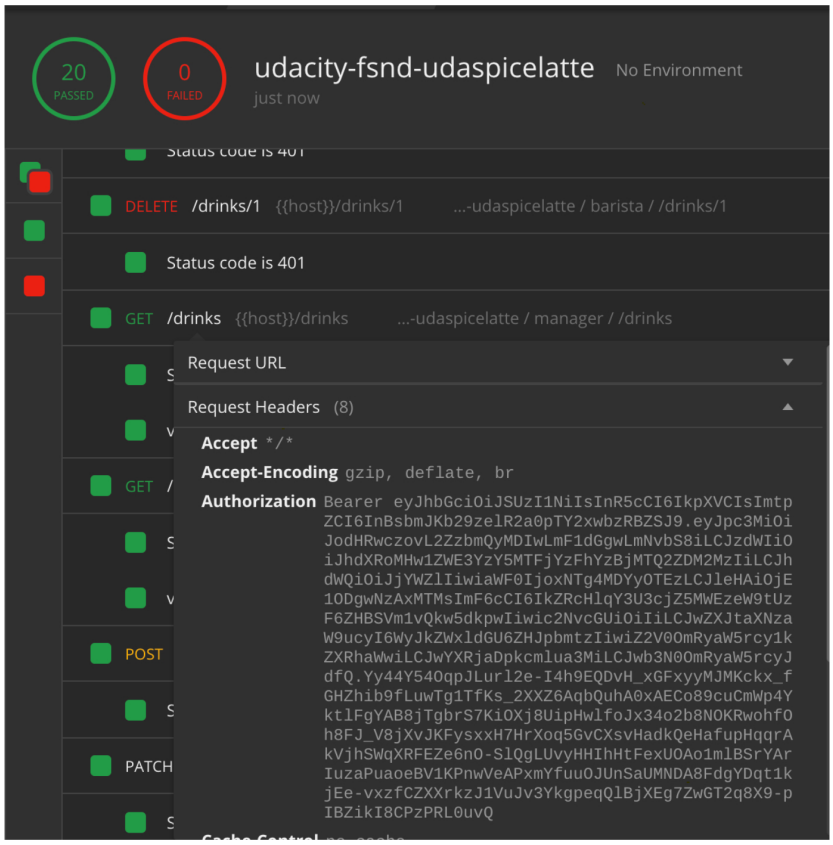
Manager access is limited

- can get drinks
- can get drink details
- can post drinks
- can patch drinks
- can delete drinks

The provided postman collection passes all tests when configured with valid JWT tokens.

You must export the postman collection to `./starter_code/backend/udacity-fsnd-udaspicelatte.postman_collection.json` with your JWTs configured by right-clicking the collection folder for barista and manager, navigating to the authorization tab, and including the JWT in the token field.

The POSTman test suit runs without any exception! It is very important to be able to create and manage REST API documentation; this way you can share your integrations more easily and document the functionalities while making sure it works as expected.



Front end

✓ The frontend has been configured with Auth0 variables and backend configuration.

The `./frontend/src/environment/environment.ts` file has been modified to include the student's variables.

All required variables are present in the frontend environment

```
};
```

✓ The frontend can be run locally with no errors with `ionic serve` and displays the expected results.

No issues while running the `npm start` command

DOWNLOAD PROJECT

8 CODE REVIEW COMMENTS ›

RETURN TO PATH