

Installation Guide

The following document describes in detail how set up the Tidbit system in a local environment.

Retrieving the project from Github

The most up-to-date, stable release of Tidbit can be found in its GitHub repository: <https://github.com/FIU-SCIS-Senior-Projects/Automated-Document-Summarization-1.0>

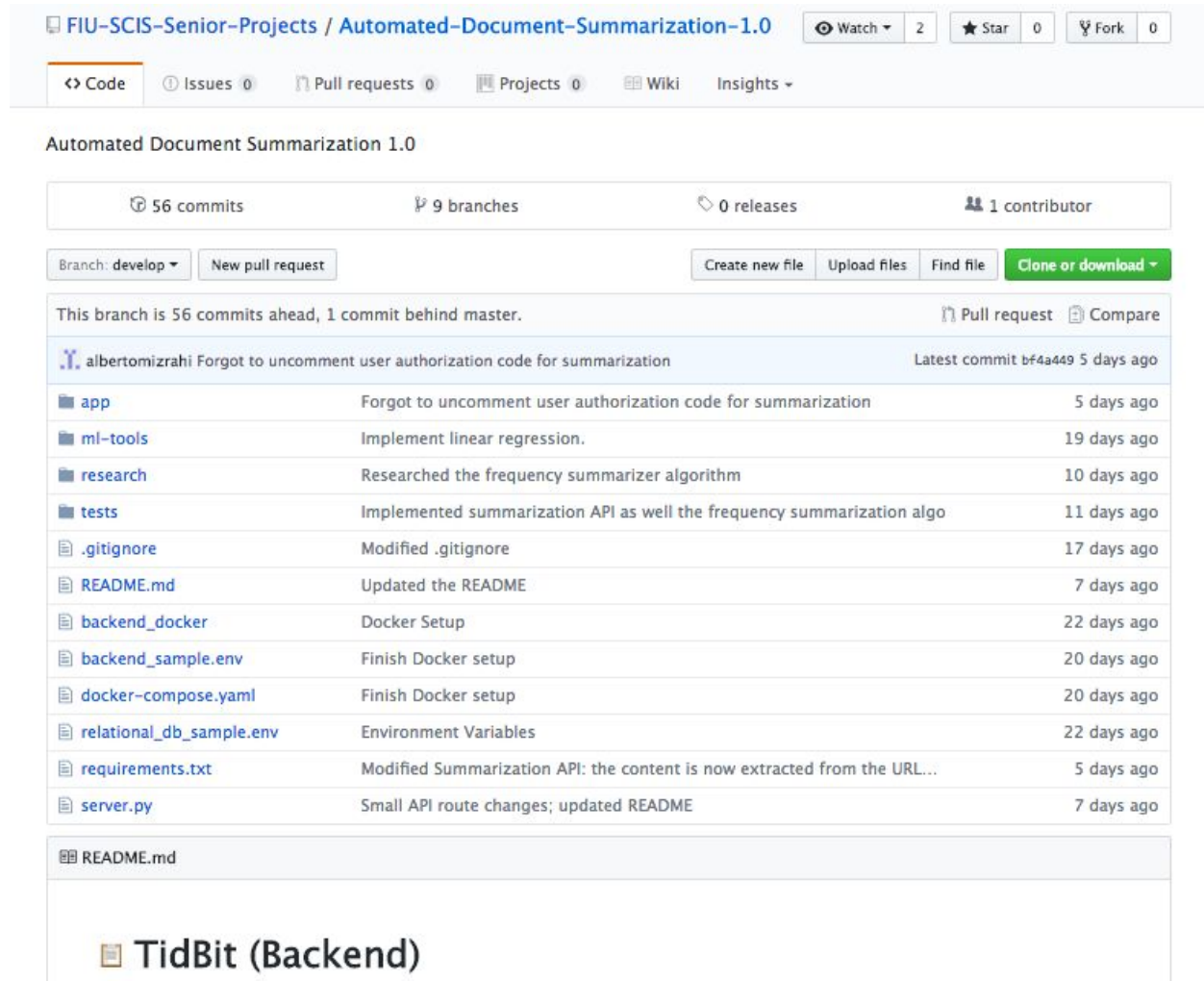


Figure 1. A picture of the Tidbit GitHub repo.

After installing Git in your local system, follow the following steps to clone the GitHub repository to your local environment:

1. In your terminal application of choice, travel to the directory where you wish to download the project.
2. Input the following command into the terminal: `git clone https://github.com/FIU-SCIS-Senior-Projects/Automated-Document-Summarization-1.0.git`

3. Immediately thereafter, input the following command: `cd Automated-Document-Summarization-1.0/Code`
4. You are now in the project's root directory.

Setting up the backend locally

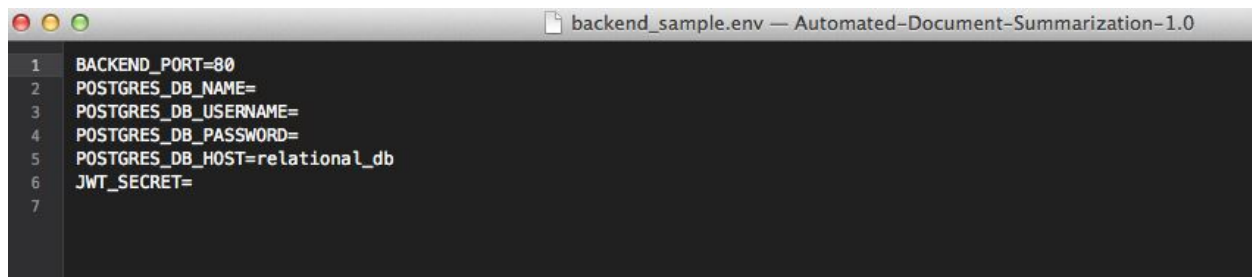
Our development workflow utilizes Docker and Docker Compose that allow us to compartmentalize our development environment so that it can be shared, set up and build in an easy and straightforward manner.

Before going it any further, you must install Docker and Docker Compose. Since the installation instructions are somewhat complex and vary between different computer system, operating systems and computer architectures, we provide links to the official guides on how to set them up:

- **Docker:** <https://docs.docker.com/engine/installation/>
- **Docker Compose:** <https://docs.docker.com/compose/install/>

Once they are installed, follow the following instruction to set up the local server:

1. Go into the project's Website directory and then go into the backend folder.
2. Rename `backend_sample.env` and `relational_db_sample.env` to `backend.env` and `relational_db.env`
3. Open the `backend.env` file. You will see the following:

A screenshot of a terminal window with a dark background. The title bar at the top reads "backend_sample.env — Automated-Document-Summarization-1.0". The terminal shows a list of environment variables with line numbers 1 through 7 on the left. The variables are: BACKEND_PORT=80, POSTGRES_DB_NAME=, POSTGRES_DB_USERNAME=, POSTGRES_DB_PASSWORD=, POSTGRES_DB_HOST=relational_db, and JWT_SECRET=. The last line (7) is empty.

```
1 BACKEND_PORT=80
2 POSTGRES_DB_NAME=
3 POSTGRES_DB_USERNAME=
4 POSTGRES_DB_PASSWORD=
5 POSTGRES_DB_HOST=relational_db
6 JWT_SECRET=
7
```

Figure 2. Contents of the `backend.env` file.

4. Modify the contents of `backend.env` as you prefer. In particular:
 - `BACKEND_PORT=80`: you should not change this.
 - `POSTGRES_DB_NAME`: the name that you want to give to the system's database.
 - `POSTGRES_DB_USERNAME`: username for the account to access the database.
 - `POSTGRES_DB_PASSWORD`: password for the account to access the database.
 - `POSTGRES_DB_HOST=relational_db`: you should not change this.
 - `JWT_SECRET`: the secret string that will be used to encrypt the database contents. It can be anything you want.
5. Open the `relational_db.env` file. You will see the following:



Figure 3. Contents of the relational_db.env file.

6. Fill in for each entry with the same information as you did for the backend.env file. Notice that in the relation_db.env file, it says POSTGRES_DB instead of POSTGRES_DB_NAME. They are, however, the same.
7. Run the following command in the terminal (note: if you receive an error when running this command or the ones that follow, add the word “*sudo*” before each command): *docker-compose build*
8. After it finishes, run the command: *docker-compose up*. If you get any errors (they appear as yellow messages) while this command is executing, press Ctrl-C. Then go to Step 10 and then repeat from this step.
9. After a few seconds, the server should be now up and running. You should be able to access the application’s backend by going to the following URL: *http://localhost:8000/*
10. After finishing, do not forget to run the following command to kill the server and destroy the Docker container: *docker-compose down*

Setting up the frontend locally

Before proceeding please make sure that the latest versions of Node.js and NPM are installed by following the official guide: <https://docs.npmjs.com/getting-started/installing-node>. Once done, continue with the following steps:

1. Go the project’s root directory and then go to the frontend folder.
2. Run the following command: *npm install*. This will install all the dependencies needed to compile the front end

Hosting the website in your local environment

In order to access the website locally, please follow the following steps:

1. In a terminal, go to the backend folder and execute the command: *sudo docker-compose up*.
2. In a separate terminal, go to the frontend folder and execute the command: *npm run dev*.
3. In a separate terminal, go to the frontend folder and execute the command: *sudo python -m SimpleHTTPServer <port>*, where you replace *<port>* by the port number where the website will be hosted (note that port 80 can not be used as it is already busy with handling the backend).

4. You can now access the website locally by following the URL: <http://localhost:<port>>, again replacing *<port>* by the port number you specified in step 3.