Developer Documentation

For Frontend contributions:

How to obtain source code:

To obtain the source code, navigate to this repository and clone it: **Husky Hustlers Frontend Repository**

Directory layout:

Our directory structure follows the Next.js routing. So, our root file is /*app*, which directs to *fonts* containing all our style fonts, layout & global CSS files that define the layout of our pages, our default page *page.tsx*, and our *src* directory containing all of our pages, styles, and components.

With *src*, we have 3 folders: *pages, styles,* and *components*.

Pages:
This contains folders that are associated with the page. Currently, we have 5 pages (add business page, business page, login page, homepage, and profile page). We have some additional files like *index.tsx* and *pages.tsx*, which will act as the rendering hub to render our different pages.

Styles:
This contains all the styles/UI (CSS) of our custom components (eg. Navigation bar style *Navbar.module.css*)

Components:
This contains all the custom components we will be using within our project (eg. Navigation bar *Navbar.tsx*)

How to Build the Software:

Navigate to the **Husky Hustlers Frontend Repository** and open the *README.txt* file. From there, open the terminal and navigate to the project's directory (you should be in ../Husky-Hustlers-Frontend/frontend/) and run all the download commands listed in the README file.

How to test Software:

Because frontend is doing major restructuring by moving all our code to Next.js, our testing is currently not implemented within our Next.js version of our project. We will update this portion once we have testing set up in our new repo!

How to add new tests:

Because frontend is doing major restructuring by moving all our code to Next.js, our testing is currently not implemented within our Next.js version of our project. We will update this portion once we have testing set up in our new repo!

However, even without the testing set up in our new repo, to add tests, you will go into the testing file (will be added with what directory it's in once we set it up) to write the tests. When naming the tests, we will use camelCasing, mentioning the specific route or method it's testing.

Eg. `retrieveListOfBizzesTest`

If there are multiple versions of the same "test" (ie. they test the same route/method), including the specific case it's testing.

Eg. `retrieveEmptyListOfBizzesTest`, `retrieveNonEmptyListOfBizzesTest`

If it's testing the same route/method and the same input/output case, then make a note at the very end of the test name with the number of iterations that test is testing on.

Eg. `retrieveNonEmptyListOfBizzesTest1`, `retrieveNonEmptyListOfBizzesTest2`

Additionally, on top of all test headers, make sure to add a comment of what the test handles (route/method, case/special cases, and iteration) , any inputs it requires, and any additional information that could be important to understand what the specific case the test is testing.

How to Build release of the software:

Make sure to call the build code to run all tests (because we are moving our code from React.js to Next.js, this is not set up within our new Repo. This will be updated with the build code once we've finished setting everything up)

For backend contributions:

How to obtain source code:

Please navigate to this repository: aiillssa/Husky-Hustler-Backend: Backend for husky hustler and clone it to retrieve our source code.

Directory layout:

Our directory follows a model view controller structure. All of the relevant development directories are in the src directory.

- Config:
  - This directory contains our configurations. It includes our CORS configurations, and our TypeORM DataSource connection.
- Controller:
  - This directory contains controllers used to communicate with our database to receive and post data. We have 2 controllers for users and shops that support basic CRUD operations.
- Middleware
  - This directory contains our middleware. Currently, it consists mostly of validators we use to verify data sent from the frontend before we send it to our database.
  - We also have a logger to keep track of requests
- Models
  - This directory contains our TypeORM entity classes. This defines our database schema.
- Routes
  - This directory has all of our relevant API routes for accessing/interacting with each resource.
- Test
  - This directory contains our unit tests.
- Server.ts: This file is the setup of our Express server, specifying our database connections and middleware/routers.

Our database is hosted on Azure.

How to build:
Run \`npm run build\`.

How to test the software:
To test software, make sure to run \`npm run test\`. This will run tests written in server.test.ts. The tests will also run automatically when pushing branches to main.

How to write tests:

Please put tests inside server.test.ts. Use httpMocks to create mock requests to test routes. Make sure to call `AppDataSource.destroy()` after writing tests to clear input. Make sure each unit test is created using `it()`. Make sure the tests for a specific feature is created using `describe()`

Building:
Make sure to run npm install before building to ensure node modules are up to date.
Also make sure your node version is 20.18x.
You may have to run the following command if your machine does not recognize the type of jsonwebtoken:

```
npm i --save-dev @types/jsonwebtoken
```