A Readme of

Automatic Evaluation of Block-based Coding Challenges

(blockJr)

Written by : Chan Siang Sheng (siangsheng.chan@gmail.com)

Created on : 18th July 2022

Table of Content

Getting Started

blockJr is using reactJs and Django as the frontend and backend. It is currently designed to host on a local server with sqlite3 as the database. There are a few prerequisites to get blockjr started.

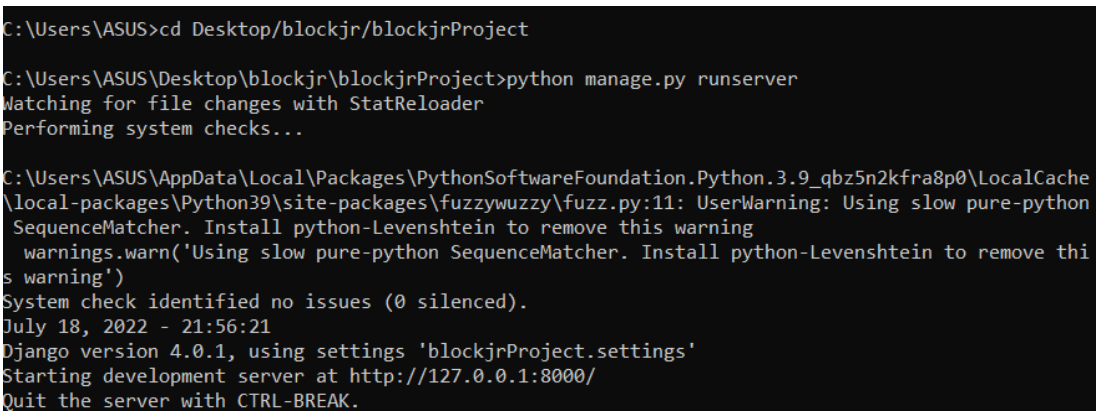
Prerequisites:

1. NodeJs (frontend)
2. Python (backend)

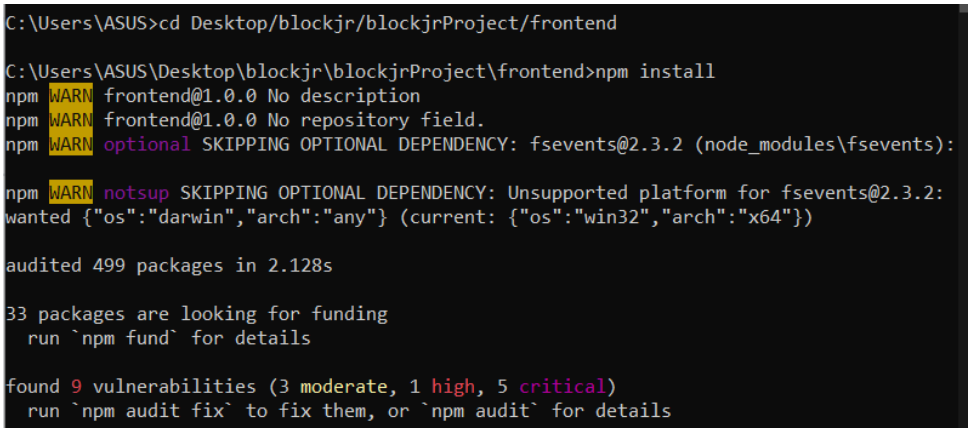
User needs to install the prerequisites in order to proceed. Once the prerequisites have been fulfilled and the source code is ready, we need to install the modules before we can start hosting blockJr.

1. Open up a command prompt and navigate to the project folder titled “blockjrProject” and run the command “pip install -r requirements” to install 4 backend libraries which are django, djangorestframework, fuzzywuzzy and ds4.

2. After backend libraries installation, we can run “python manage.py runserver”. If there is no error on the backend, you should be able to see a “Quit the server with CTRL-BREAK.” at the end of the line. An example is shown below.



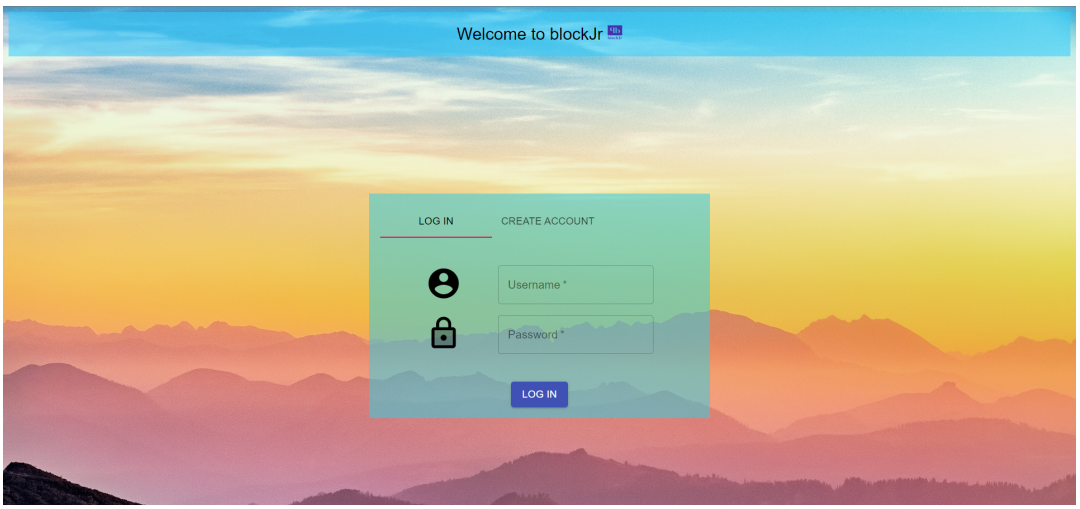
3. Next, open up another command prompt and navigate to a subfolder titled “frontend” in the project folder “blockjrProject” to run the command “npm install” to install every single module that is listed in package.json. The process might take some time as there are many libraries involved. An example is shown below.



4. Then, we can finally run another command “npm run dev”. If there is no error on the frontend, you should be able to see “webpack 5.70.0 compiled successfully in ... ms” at the end of the line. An example is shown below.

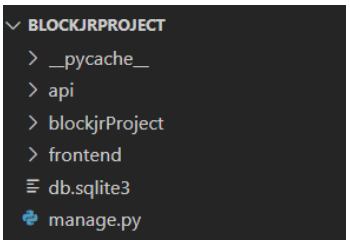
```
C:\Users\ASUS>cd Desktop/blockjr/blockjrProject/frontend

C:\Users\ASUS\Desktop\blockjr\blockjrProject\frontend>npm run dev

> frontend@1.0.0 dev C:\Users\ASUS\Desktop\blockjr\blockjrProject\frontend
> webpack --mode development --watch

asset main.js 2.74 MiB [compared for emit] [minimized] (name: main) 1 related asset
orphan modules 360 KiB [orphan] 250 modules
runtime modules 1.13 KiB 5 modules
cacheable modules 2.1 MiB
  modules by path ./node_modules/@material-ui/ 796 KiB 203 modules
  modules by path ./src/ 174 KiB 36 modules
  modules by path ./node_modules/@babel/runtime/helpers/ 8.86 KiB 24 modules
  modules by path ./node_modules/prop-types/ 34 KiB 7 modules
  modules by path ./node_modules/react-transition-group/esm/ 30.1 KiB 6 modules
  modules by path ./node_modules/scheduler/ 34 KiB 4 modules
  modules by path ./node_modules/hoist-non-react-statics/ 9.69 KiB 3 modules
  modules by path ./node_modules/react/ 62.2 KiB 2 modules
  modules by path ./node_modules/react-dom/ 748 KiB 2 modules
  modules by path ./node_modules/react-is/ 7.58 KiB 2 modules
  + 18 modules
webpack 5.70.0 compiled successfully in 5033 ms
assets by status 2.74 MiB [cached] 1 asset
cached modules 2.45 MiB (javascript) 1.13 KiB (runtime) [cached] 562 modules
webpack 5.70.0 compiled successfully in 701 ms
```

5. With that, we can open up a browser and navigate to “http://127.0.0.1:8000/login” to open up blockJr and expect to have a view of the login page as shown below.

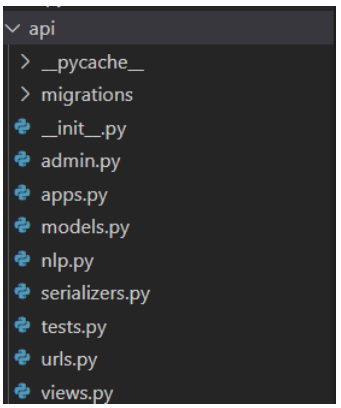


Directory Structure

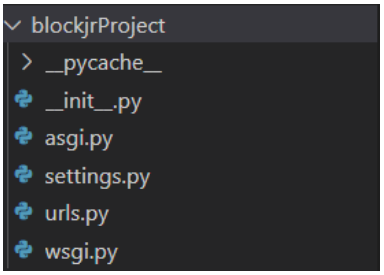
1. Main project folder
   The main project folder is titled as “blockjrProject”, it consists of “api”, “blockjrProject” and “frontend” folder which are the Django app directory. A Django app directory is not just an ordinary folder, it consists of multiple Django files with corresponding functionality. Next, “api” is the folder that is composed of api files, “blockjrProject” is the folder that acts as the entrypoint of the application, “frontend” is the folder that is composed of frontend files. On the contrary, “db.sqlite3” is the database while “manage.py” is the mandatory file for django to configure settings and run the server.



2. Backend/Api folder
   “api” is a Django app directory, every api or backend source code files are located in here. The only special thing in here is the nlp.py which is the NLP engine of blockjr and the serializers.py that handles the format of HTTP requests corresponding to the Django model (database).

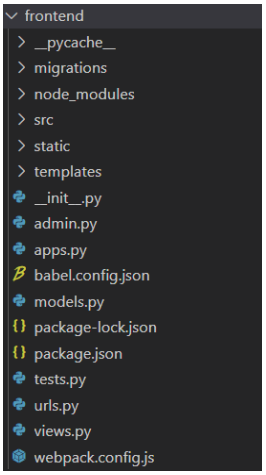


3. Entrypoint folder
   “blockjrProject” is the Django entrypoint directory, it is little different from the other Django app directory. urls.py is a Django file that handles the path of the url to a specific Django app directory. [1]



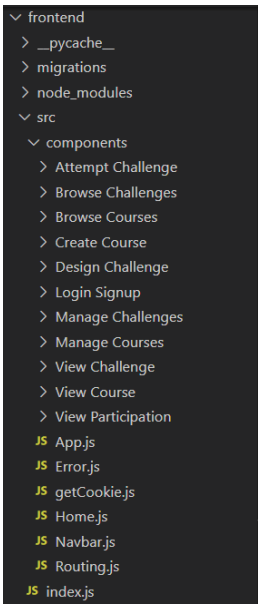
4. Frontend folder

   “frontend” is another Django app directory, every frontend source code files are located in here. “src” folder is composed of every reactJs source code, “static” folder consists of resources such as CSS, images, project bundles and lastly, “templates” folder consists of the HTML file which is used as the template for reactJS to render.



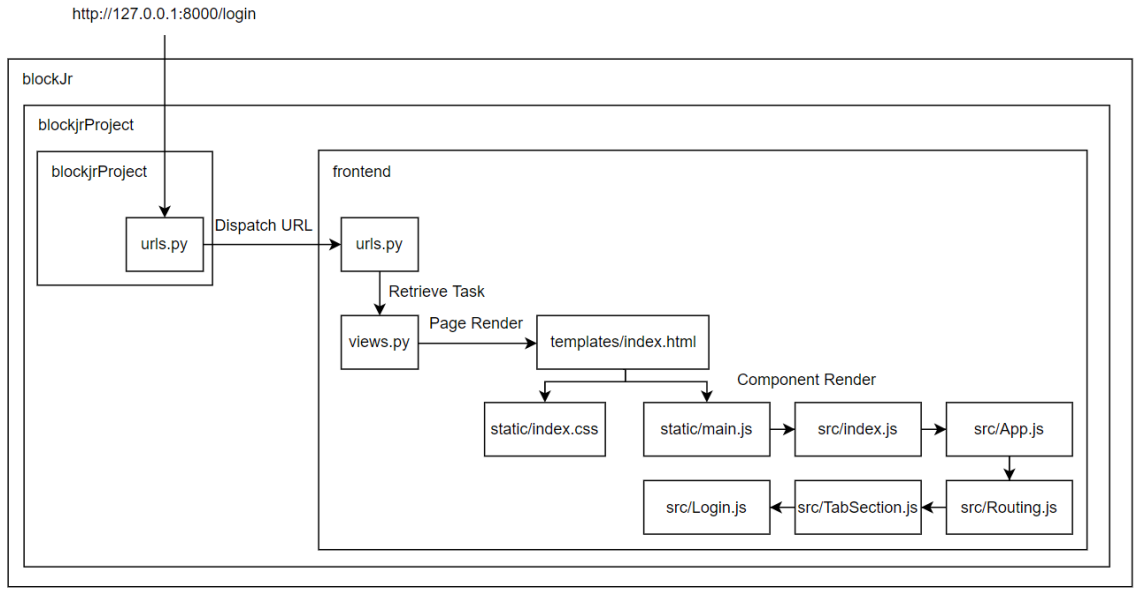
5. Component Render folder

   Component Render is a functionality from reactJs, it renders a certain element of the HTML page and the render is written in Javascript. Each page of blockJr is written and saved in a folder if there is more than one javascript file or else within the directory of “/src/component”. index.js is the file that links the App.js with main.js from “static/frontend” which is the javascript of project bundle. Appjs is the file that targets which specific element of HTML to render and the component to be rendered by will be decided by Routing.js. Routing.js will decide which component (Javascript file) to be rendered based on the URL. Only getCookie.js is not relevant to the component render but it is a javascript function to check on the cookie and it is used across most of the component render file.

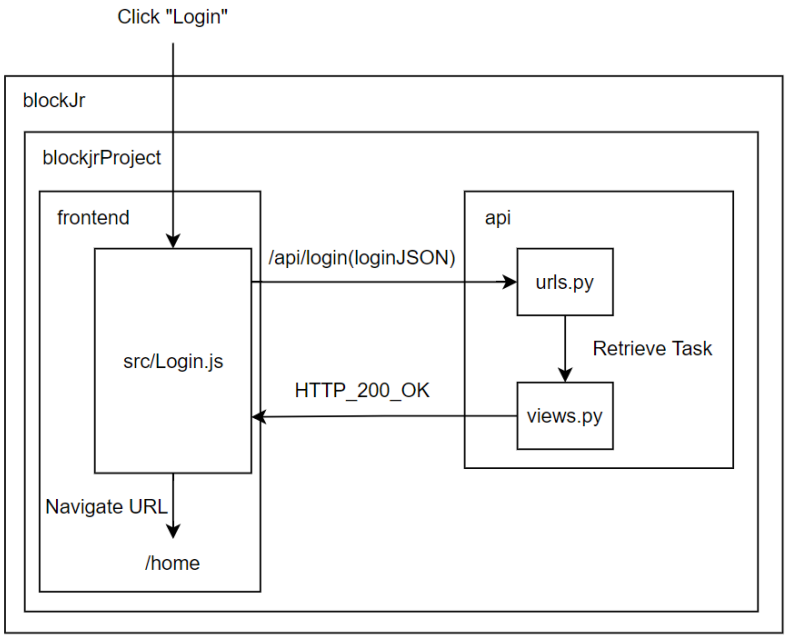


6. Django app directory
   “api” and “frontend” are the Django app directory, urls.py is a Django file that works almost the same as the urls.py in the entrypoint folder (blockjrProject), except that it handles the path of url to a specific view function that is declared in views.py. [1] views.py is another Django file that has multiple user-defined functions to do several tasks such as rendering page, database management, HTTP response, etc. [2] models.py is the declaration of a database in a table form. [3]

A workflow that illustrates how blockJr responds to a URL is shown below.



Another workflow that illustrates how blockJr responds to a HTTP request such as login with information is shown below.



References

[1] https://docs.djangoproject.com/en/4.0/topics/http/urls/
[2] https://docs.djangoproject.com/en/4.0/topics/http/views/
[3] https://docs.djangoproject.com/en/4.0/topics/db/models/