Hiring Task Documentation

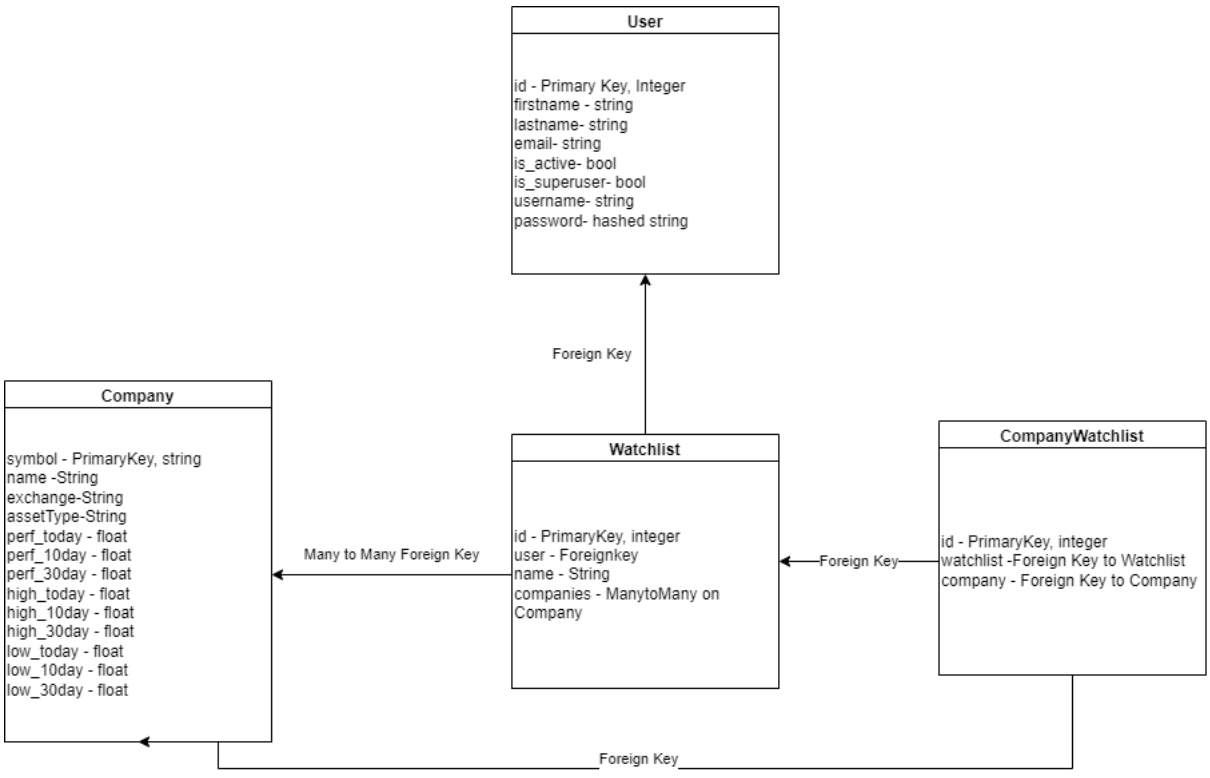
Project directory Structure:

```
Base_dir
    |-backend
    |       |-backend            – Contains Project Settings
    |       |- authentication    – Contains Authentication APIs
    |       |- user_api          – Contains User related APIs
    |
    |-frontend
    |       |-src
    |           |- Pages         - Contains folder for each page
    |           |- Component     - Contains folder for each component
```

Backend:

Models:

1. User: Default Django.auth.user
2. Company: Defined in .\backend\user_api\views.py
3. Watchlist: Defined in .\backend\user_api\views.py
4. WatchlistCompany: Defined in .\backend\user_api\views.py



APIs:

1. Login API

URL	http://127.0.0.1:8000/auth/signin/
File	.\backend\authentication\views.py
Request Type	POST

Body	{
	"username": username,
	"Password":password
	}

Input parameters	• username – string • password – string

Response	{
	"message": message
	"access": access token
	"refresh": refresh token
	}

Response parameters	message: string -login status access: string - JWT access token refresh: string - JWT refresh token

2. Sign Up API

URL	http://127.0.0.1:8000/auth/signup/
File	.\backend\authentication\views.py
Request Type	POST

Body	{
	firstName: firstName,
	lastName: lastName,
	email: email,
	username: username,
	password: password,
	confirmPassword: confirmPassword,
	}

Input parameters	• firstName - string • lastName - string • email – string - email • username - string • password - string • confirmPassword - string

Response	{
	"status": "success"
	}
	Or
	{
	"error": error message
	}

Response parameters	status: string

	error: string

3. Check Username Availability API

URL	http://127.0.0.1:8000/auth/check_username/<username>/
File	.\backend\authentication\views.py
Request Type	GET
Body	username
Input parameters	Username - String
Response	{'available': True or false}
Response parameters	'available': boolean

4. Logout API

URL	http://127.0.0.1:8000/auth/logout/
File	.\backend\authentication\views.py
Request Type	GET
Body	-
Input parameters	-

Response	{
	"message": logout status
	}

Response parameters	message: string – success or fail

5. Get CSRF token API

URL	http://127.0.0.1:8000/auth/get_csrf/
File	.\backend\authentication\views.py
Request Type	GET
Body	-
Input parameters	-
Response	{'csrfToken': token}
Response parameters	token: string – csrf token

6. Dashboard Data API

URL	http://127.0.0.1:8000/user/dashboard/
File	.\backend\user_api\views.py
Request Type	GET
Body	-
Input parameters	-
Response	{ 'user' : username, 'watchlists' : user watchlists, 'watchlist_companies': companies in watchlist, 'allCompanies': companies list, 'perf_metrics': performance metrics for overall companies

	}

Response parameters	User – string
	watchlists – List of Dictionaries of watchlist data
	watchlist_companies – dictionary-
	{
	watchlist name: [companies1, company2], ….
	}
	allCompanies – list of Dictionary
	perf_metrics – Dictionary
	{
	“Max_perf_today”: { },
	“Min_perf_today”: { },
	“Max_perf_10day”: { },
	“Min_perf_10day”: { },
	“Max_perf_30day”: { },
	“Min_perf_30day”: { }
	
	}

7. Create New Watchlist

URL	http://127.0.0.1:8000/user/addWatchlist/
File	.\backend\user_api\views.py
Request Type	POST
Body	{ name: new watchlist name}
Input parameters	name - string
Response	{'msg':success or failed} Or {'error': 'Method not allowed'}
Response parameters	message – string error - string

8. Get performance metrics of current watchlist API

URL	http://127.0.0.1:8000/user/getWatchlistPerf/
File	.\backend\user_api\views.py
Request Type	POST
Body	{'watchlist': id of watchlist}
Input parameters	watchlist- integer

Response	{
	‘max_perf_today': {},
	'min_perf_today': {},
	'max_perf_10day': {},
	'min_perf_10day': {},
	'max_perf_30day': {},
	'min_perf_30day': {}
	}

Response parameters	max_perf_today – Dictionary of company data min_perf_today – Dictionary of company data max_perf_10day– Dictionary of company data

	min_perf_10day– Dictionary of company data max_perf_30day– Dictionary of company data min_perf_30day– Dictionary of company data

9. Delete a Watchlist API

URL	http://127.0.0.1:8000/user/removeWatchlist/
File	.\backend\user_api\views.py
Request Type	POST
Body	{ selectedWatchlists: list of watchlist names }
Input parameters	selectedWatchlists – String[]
Response	{'msg':success or failed} Or {'error': 'Method not allowed'}
Response parameters	message – string error - string

10. Add Company to Watchlist API

URL	http://127.0.0.1:8000/user/addCompanyToWatchlist/
File	.\backend\user_api\views.py
Request Type	POST

Body	{
	'companydata': type WatchlistCompanies,
	'updatedwatchlist': name of watchlist to add company
	}

Input parameters	companydata - type WatchlistCompanies updatedwatchlist - string
Response	{'msg':success or failed} Or {'error': 'Method not allowed'}
Response parameters	message – string error - string

11. Get Data for Company Graph and Company performance Metrics

URL	http://127.0.0.1:8000/user/getCompanyGraph/
File	.\backend\user_api\views.py
Request Type	POST
Body	{ 'selectedCompany': symbol of selected company}
Input parameters	selectedCompany – string

Response	{
	'candlestick_data': data to plot Graph,
	'perf_metrics': company performance metrics
	}

Response parameters	candlestick_data – Dictionary
	{
	x: Timestamp
	y: [Open, high, low, close]
	}

	perf_metrics: type PerfMetricsCompany

Frontend:

Pages:

1. Home: Landing Page
    a. Functions
        i. -
2. Dashboard: User specific page reached after login
    a. Functions
        i. get_data(): Fetch dashboard API and set data
        ii. showCompanies(id): Expand / Collapse watchlist
        iii. displayCheckboxes(): Display/Hide checkbox when editing watchlist
        iv. updateSelectedList(): Add or remove watchlist from list sent to remove watchlist api
        v. getcsrftoken(): Fetch csrf token from server
        vi. update_watchlists(): Send list from updateSelectedList to server
        vii. handleaddwatchlist(): Handle create new watchist
        viii. isCompanyInWatchlist(symbol): Checks if symbol is in current selected watchlist
        ix. updatewatchlistcompanies(): add or remove company from current watchlist (on Frontend only)
        x. updatecompaniesinwatchlist(): Send updatewatchlistcompanies list to server
        xi. getWatchlistPerf(): Get performance metrics of current watchlist

3. Login: Page for users to log in
    a. Functions
        i. handleUsernameChange(): Update username state
        ii. handlePasswordChange(): Update Password State
        iii. handleSubmit(): Send username and password to server to login

4. SignUp: Page for Users to sign up
    a. Functions
        i. handleCheckAvailability – check from server if username is available
        ii. check_params – check if all parameters are filled
        iii. handleSubmit – submit signup form to server

Components

1. Navigation Bar: Top Navigation Bar
    a. Functions
        i. Logout(): sign out current user from server
    b. Called in – Base Page

2. Base Page: Template rendered by all pages
    a. Functions
        i. None

    b. Called in: All Pages
3. Default Performance Page: Performance Cards on default dashboard
    a. Functions
        i. None
    b. Called in: Dashboard Page

4. Watchlist Performance Page: Performance cards for specific watchlist
    a. Functions
        i. getCompanyGraph(): Fetch Compang graph data API
    b. Called in: Dashboard Page

5. Graph Plotter: Plots Candlestick chart from company data
    a. Funstions
        i. None
    b. Called in: Watchlist Performance Page