

## Wyse SDE Intern Backend Task

### Problem Statement

Assignment: Django

In this assignment, you will create a small Django application.

Creating the project

1. Run your code with Python (3.9) and Django (3.2).
2. Setup a local mongodb database for storing the data.
3. Use Firebase to authorise (create and verify custom token) the User using Middleware

In all parts, you are allowed to use function-based or class-based views. You can also use any built-in tool that is already provided by Django

HTTP status codes

All requests should be responded to with an HTTP 200 OK status code unless otherwise specified. You must check for client errors in the following order: 401 UNAUTHORIZED and 404 NOT FOUND. That is, for example, at the create branch view, if the request is not authenticated and the bank ID is invalid as well, you should return 401 UNAUTHORIZED. Your server must never return a 500 INTERNAL SERVER ERROR.

Auth View:

Implement views for register and login.

Endpoint: /accounts/register/

Methods: POST

Payload: username,email, password, first\_name, last\_name

Validation errors: (copy and paste the exact error message)

A user with that username already exists

This password is too short. It must contain at least 8 characters

Email and password are required.

Only 100 characters are allowed for a field

Additional notes:

1. Email and password field are required. The other fields are optional. Moreover, all fields will be at most 100 characters long.
2. On registering, a unique username for the user should be generated, saved in the DB and returned in response.
  - a. Create a function to verify no other user exist with the generated Username before saving in a DB.
3. Return a JSON string with the following fields : username, email
  - a. Example response:

{ "username": "wishfulPanda23", "email": "[demon@gmail.com](mailto:demon@gmail.com)" }

Endpoint: /accounts/login/

Methods: GET, POST

Fields/payload: username, password

Invalid inputs:

Username or password is invalid

Additional notes:

1. If the credentials are valid, use Firebase auth to create a custom\_token and return in the response.

a. Apart from account login and register, all other Urls should be authorised with the custom\_token

2. Example response:

{ "username": "wishfulPanda23", "email": "[demon@gmail.com](mailto:demon@gmail.com)", "full\_name": "" }

Profile

In this question, you will implement the view and edit profile functionalities. Note that you should return HTTP 401 UNAUTHORIZED if the request is unauthenticated.

Endpoint: /accounts/profile/view/

Methods: GET

Fields: username

Authorization: custom\_token

Description:

1. Return a JSON string with the following fields : username, email and fullname (first\_name+last\_name).

a. Example response:

{ "username": "wishfulPanda23", "email": "[demon@gmail.com](mailto:demon@gmail.com)", "full\_name": "shubham-joshi" }

i. Here shubham is firstname, joshi is secondname

2. Verify custom\_token, if invalid custom\_token is used then return response status 401

3. Serializer to be used for creating full\_name field response. (Shouldn't be used in DB)

Endpoint: /accounts/profile/edit/

Methods: POST

Payload: first\_name, last\_name, username

Success URL: /accounts/profile/view/

Validation errors: (copy and paste the exact error message)

User already exist with the username \${entered\_username}

Additional Note:

1. User can edit one or more field.
2. Verify custom\_token, if invalid custom\_token is used then return response status 401.

Example response:

```
{"username": "wishfulPanda23", "email": "demon@gmail.com",  
"full_name": "shubham-joshi"}
```

a. Here shubham is firstname, joshi is secondname

3. If username field is edited, then username uniqueness should be checked.

**Implemented a React plus Django base Full Stack Application with using Firebase Authentication as Middleware and MongoDB for local storage of User Data.**

### **Flow of the App**

Initially we prompt users with a registration form on successful registration users are redirected to login page. After successful login in users can get their profile data or update their details.

### **Snippets of the App**

mayank4@gmail.com

.....

First Name

Last Name

Register

Already have an account? [Login](#)

Harshiv49@yahoo.com

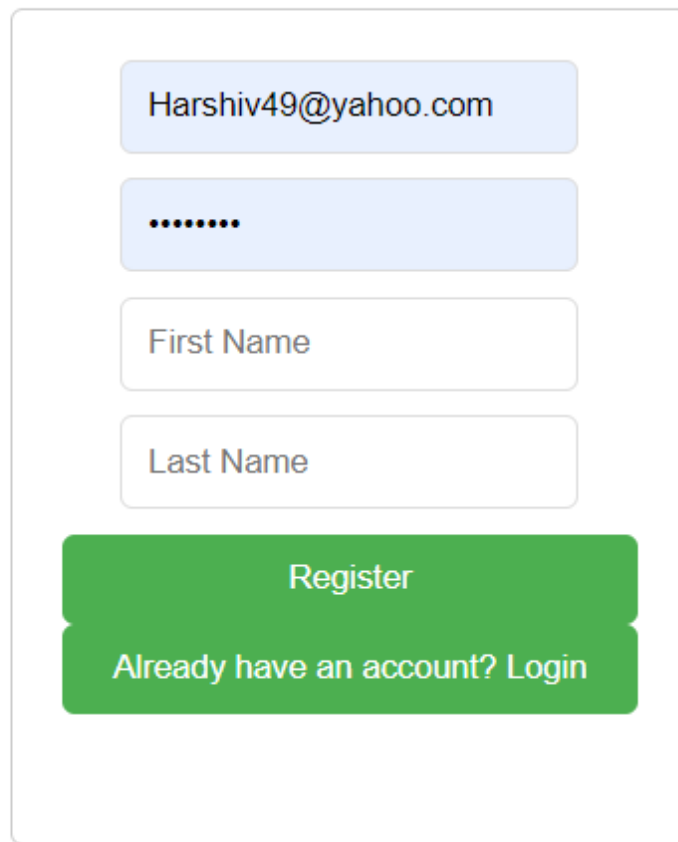
••••••••

First Name

Last Name

Register

Already have an account? [Login](#)



A registration form with a light gray background. It contains four input fields: an email field with 'Harshiv49@yahoo.com', a password field with masked characters '.....', a 'First Name' field, and a 'Last Name' field. Below these fields are two green buttons: 'Register' and 'Already have an account? Login'.

## Backend Response

Response from /accounts/profile/view

```
▼ {data: Array(1)} ⓘ namescreen.jsx:27
  ▼ data: Array(1)
    ▶ 0: {username: 'mayank4', email: 'mayank4@gmail.com', fullname: 'may, length: 1}
    ▶ [[Prototype]]: Array(0)
    ▶ [[Prototype]]: Object
```

Response from /accounts/profile/edit/

HomeScreen.jsx:46

```
{message: 'User details updated successfully', username: 'harshiv', email: 'Har49@yahoo.com', fullname: 'tha kkar'}  
  email: "Har49@yahoo.com"  
  fullname: "tha kkar"  
  message: "User details updated successfully"  
  username: "harshiv"  
  [[Prototype]]: Object
```

## Firestore Snippets

| Search by email address, phone number or user UID |           |            |            |                   | Add user |  |  |
|---|-----------|------------|------------|-------------------|----------|--|--|
| Identifier  | Providers | Created ↓  | Signed in  | User UID          |          |  |  |
| -   |           | 5 Nov 2023 | 5 Nov 2023 | mayank4@gmail.com |          |  |  |

## Mongo DB Snippets

```
{  
  _id: ObjectId('6547927301b7fec9f363fe6f')  
  id: 3  
  password: "pbkdf2_sha256$390000$JPK5Tu5xfN8dHxBS0u0A5X$Kk6KCObJ399M8pxMnk2nVz7l30..."  
  last_login: null  
  is_superuser: false  
  is_staff: false  
  is_active: true  
  date_joined: 2023-11-05T13:02:43.475+00:00  
  username: "mayank4"  
  first_name: "mayank"  
  last_name: "thakkar"  
  email: "mayank4@gmail.com"  
  mobile: "0"  
  is_verified: false  
  email_token: null  
  forget_password: null  
  last_login_time: null  
  last_logout_time: null  
  auth_provider: "email"
```

## Setup

Install the necessary packages of backend mentioned in requirements.txt

```
pip install requirements.txt
```

Run the command:

```
// inside the frontend directory install the dependencies mentioned in package.json
```

npm i