Step – by – step everything

1.

Let’s start developing Mealy step by step while aligning with the structure we discussed. I'll guide you through every task to ensure it's organized and matches the structure.

---

### Step 1: Initial Project Setup

We’ll start by creating the base structure for the backend and frontend, then set up version control with GitHub.

---

#### Task 1.1: Create the Base Project Structure

1. Open a terminal or command prompt and navigate to the folder where you want the project.

2. Run the following commands to create the base folders:

```bash

mkdir Mealy

cd Mealy

mkdir backend\_django frontend\_mobile frontend\_web firebase

mkdir -p backend\_django/{mealy\_backend,static,media,community\_chat,migrations}

mkdir -p frontend\_mobile/{android,ios,assets/src}

mkdir -p frontend\_web/{public,src}

mkdir -p firebase

```

Folder Structure Created:

- `backend\_django`: Backend server.

- `frontend\_mobile`: React Native mobile app.

- `frontend\_web`: ReactJS web app.

- `firebase`: Firebase configurations.

---

#### Task 1.2: Set Up GitHub Repository

1. Navigate to the Mealy directory and initialize Git:

```bash

git init

```

2. Create a `.gitignore` file in the root of Mealy:

```bash

echo "node\_modules/" >> .gitignore

echo "\*.env" >> .gitignore

echo "\_\_pycache\_\_/" >> .gitignore

echo "db.sqlite3" >> .gitignore

```

3. Commit the base structure:

```bash

git add .

git commit -m "Initial project structure"

```

4. Create a new GitHub repository and push your code:

```bash

git remote add origin <your-github-repo-url>

git branch -M main

git push -u origin main

```

---

### Step 2: Backend Setup

#### Task 2.1: Set Up Django Project

1. Navigate to the `backend\_django` folder:

```bash

cd backend\_django

```

2. Create a virtual environment:

```bash

python -m venv venv

source venv/bin/activate # For Linux/Mac

venv\Scripts\activate # For Windows

```

3. Install Django and other dependencies:

```bash

pip install django djangorestframework psycopg2

pip freeze > requirements.txt

```

4. Start the Django project:

```bash

django-admin startproject mealy .

```

5. Create the main app for the backend:

```bash

python manage.py startapp mealy\_backend

```

6. Update the `INSTALLED\_APPS` in `mealy/settings.py`:

```python

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'rest\_framework', # Django REST Framework

'mealy\_backend', # Main backend app

]

```

7. Configure static and media files in `mealy/settings.py`:

```python

STATIC\_URL = '/static/'

STATIC\_ROOT = 'static/'

MEDIA\_URL = '/media/'

MEDIA\_ROOT = 'media/'

```

8. Apply migrations:

```bash

python manage.py makemigrations

python manage.py migrate

```

---

### Step 3: Frontend Setup

#### Task 3.1: Set Up React Native for Mobile

1. Navigate to the `frontend\_mobile` folder:

```bash

cd ../frontend\_mobile

```

2. Create a new React Native app:

```bash

npx react-native init MealyMobile

```

3. Organize the structure:

- Move React Native code into `src/`.

- Add `assets/` for images and fonts.

---

#### Task 3.2: Set Up ReactJS for Web

1. Navigate to the `frontend\_web` folder:

```bash

cd ../frontend\_web

```

2. Create a new React app:

```bash

npx create-react-app MealyWeb

```

3. Organize the structure:

- Move React components into `src/components`.

- Add `src/pages` for screens like Home, Login, etc.

---

### Step 4: Firebase Setup

1. Navigate to the `firebase` folder:

```bash

cd ../firebase

```

2. Add Firebase configuration files:

- `firebaseConfig.js`: For Firebase initialization.

- `auth.js`: For authentication logic.

- `database.js`: For database interaction.

---

### Summary of Tasks

1. Base Structure Creation: Completed using `mkdir` commands.

2. Backend Setup: Django project initialized with main app (`mealy\_backend`).

3. Frontend Setup:

- React Native app for mobile.

- ReactJS app for web.

4. Firebase Configuration: Placeholder setup for Firebase integration.

---

Let me know when you’ve completed these steps, and we’ll move on to building the first feature, like user authentication or meal planning API. 😊

S2;

Great! With the base structure and initial setup complete, let's move to the \*\*next step: building the first feature\*\*.

---

### \*\*Step 5: Implement User Authentication (Backend)\*\*

Authentication is the foundation for all other features, enabling secure access to personal dashboards, family collaboration, and more.

---

#### \*\*Task 5.1: Add User Authentication in Django\*\*

1. \*\*Install Django REST Framework Authentication Libraries\*\*:

- We’ll use \*\*Django REST Framework\*\* (DRF) and \*\*Simple JWT\*\* for token-based authentication.

```bash

pip install djangorestframework-simplejwt

pip install djoser

```

2. \*\*Update Backend Settings\*\*:

- Add the following to `mealy/settings.py`:

```python

REST\_FRAMEWORK = {

'DEFAULT\_AUTHENTICATION\_CLASSES': [

'rest\_framework\_simplejwt.authentication.JWTAuthentication',

],

'DEFAULT\_PERMISSION\_CLASSES': [

'rest\_framework.permissions.IsAuthenticated',

],

}

SIMPLE\_JWT = {

'ACCESS\_TOKEN\_LIFETIME': timedelta(minutes=30),

'REFRESH\_TOKEN\_LIFETIME': timedelta(days=7),

'AUTH\_HEADER\_TYPES': ('Bearer',),

}

```

3. \*\*Update Installed Apps\*\*:

```python

INSTALLED\_APPS += ['rest\_framework', 'rest\_framework\_simplejwt', 'djoser']

```

4. \*\*Create API Endpoints for Authentication\*\*:

- Update `backend\_django/mealy\_backend/urls.py`:

```python

from django.urls import path, include

urlpatterns = [

path('auth/', include('djoser.urls')), # For user registration

path('auth/', include('djoser.urls.jwt')), # For JWT login/logout

]

```

5. \*\*Test Authentication Endpoints\*\*:

- Run the Django server:

```bash

python manage.py runserver

```

- Use a tool like \*\*Postman\*\* to test:

- `POST /auth/users/` to register a user.

- `POST /auth/jwt/create/` to log in and get a token.

- `GET /auth/users/me/` to fetch the current user profile (requires authentication).

---

### \*\*Step 6: Set Up Authentication in the Frontend\*\*

#### \*\*Task 6.1: Connect React Native (Mobile) to the Backend\*\*

1. \*\*Install Axios\*\* for API requests:

```bash

cd ../frontend\_mobile

npm install axios

```

2. \*\*Create Authentication Service\*\*:

- In `src/services/api.js`:

```javascript

import axios from 'axios';

const API\_URL = 'http://127.0.0.1:8000/auth/';

export const registerUser = (userData) => {

return axios.post(`${API\_URL}users/`, userData);

};

export const loginUser = (credentials) => {

return axios.post(`${API\_URL}jwt/create/`, credentials);

};

export const getUserProfile = (token) => {

return axios.get(`${API\_URL}users/me/`, {

headers: { Authorization: `Bearer ${token}` },

});

};

```

3. \*\*Create Login and Register Screens\*\*:

- Add `src/screens/LoginScreen.js` and `src/screens/RegisterScreen.js`:

```javascript

import React, { useState } from 'react';

import { View, Text, TextInput, Button, Alert } from 'react-native';

import { loginUser } from '../services/api';

const LoginScreen = ({ navigation }) => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const handleLogin = async () => {

try {

const response = await loginUser({ email, password });

Alert.alert('Success', 'Logged in successfully!');

console.log(response.data); // Store token

} catch (error) {

Alert.alert('Error', 'Invalid credentials');

}

};

return (

<View>

<Text>Login</Text>

<TextInput placeholder="Email" value={email} onChangeText={setEmail} />

<TextInput

placeholder="Password"

value={password}

onChangeText={setPassword}

secureTextEntry

/>

<Button title="Login" onPress={handleLogin} />

</View>

);

};

export default LoginScreen;

```

---

#### \*\*Task 6.2: Connect ReactJS (Web) to the Backend\*\*

1. \*\*Install Axios\*\* for API requests:

```bash

cd ../frontend\_web

npm install axios

```

2. \*\*Create Authentication Service\*\*:

- In `src/services/api.js`:

```javascript

import axios from 'axios';

const API\_URL = 'http://127.0.0.1:8000/auth/';

export const registerUser = (userData) => {

return axios.post(`${API\_URL}users/`, userData);

};

export const loginUser = (credentials) => {

return axios.post(`${API\_URL}jwt/create/`, credentials);

};

export const getUserProfile = (token) => {

return axios.get(`${API\_URL}users/me/`, {

headers: { Authorization: `Bearer ${token}` },

});

};

```

3. \*\*Create Login and Register Pages\*\*:

- Add `src/pages/LoginPage.js` and `src/pages/RegisterPage.js`:

```javascript

import React, { useState } from 'react';

import { loginUser } from '../services/api';

const LoginPage = () => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const handleLogin = async () => {

try {

const response = await loginUser({ email, password });

alert('Logged in successfully!');

console.log(response.data); // Store token

} catch (error) {

alert('Invalid credentials');

}

};

return (

<div>

<h1>Login</h1>

<input

type="email"

placeholder="Email"

value={email}

onChange={(e) => setEmail(e.target.value)}

/>

<input

type="password"

placeholder="Password"

value={password}

onChange={(e) => setPassword(e.target.value)}

/>

<button onClick={handleLogin}>Login</button>

</div>

);

};

export default LoginPage;

```

---

### \*\*Step 7: Test Authentication\*\*

1. \*\*Backend:\*\* Ensure the API endpoints work correctly in Postman.

2. \*\*Frontend:\*\*

- Test the login flow in both mobile and web apps.

- Verify tokens are retrieved and used for subsequent requests.

---

Once this is working, we’ll move to the next step, \*\*building the meal planning API\*\*. Let me know when you're ready! 😊