

Project structure and API routes

Introduction

This reading is an overview of the scope of the project, all the necessary endpoints, and notes that you will have to implement in the final project. This reading will help you to successfully complete the project so read it carefully and reference it while developing your API project to help you keep on track.

Scope

You will create a fully functioning API project for the Little Lemon restaurant so that the client application developers can use the APIs to develop web and mobile applications. People with different roles will be able to browse, add and edit menu items, place orders, browse orders, assign delivery crew to orders and finally deliver the orders.

The next section will walk you through the required endpoints with an authorization level and other helpful notes. Your task is to create these endpoints by following the instructions.

Structure

You will create one single Django app called LittleLemonAPI and implement all API endpoints in it. Use pipenv to manage the dependencies in the virtual environment. Review the video about [Creating a Django Project using pipenv](#).

Function or class-based views

You can use function- or class-based views or both in this project. Follow the proper API naming convention throughout the project. Review the video about [Function- and class-based views](#) as well as the video about [Naming conventions](#).

User groups

Create the following two user groups and then create some random users and assign them to these groups from the Django admin panel.

- Manager
- Delivery crew

Users not assigned to a group will be considered customers. Review the video about [User roles](#).

Error check and proper status codes

You are required to display error messages with appropriate HTTP status codes for specific errors. These include when someone requests a non-existing item, makes unauthorized API requests, or sends invalid data in a *POST*, *PUT* or *PATCH* request. Here is a full list.

HTTP Status code	Reason
<i>200 - Ok</i>	For all successful <i>GET</i> , <i>PUT</i> , <i>PATCH</i> and <i>DELETE</i> calls
<i>201 - Created</i>	For all successful <i>POST</i> requests
<i>403 - Unauthorized</i>	If authorization fails for the current user token
<i>401 – Forbidden</i>	If user authentication fails
<i>400 – Bad request</i>	If validation fails for <i>POST</i> , <i>PUT</i> , <i>PATCH</i> and <i>DELETE</i> calls
<i>404 – Not found</i>	If the request was made for a non-existing resource

API endpoints


Here are all the required API routes for this project grouped into several categories.

User registration and token generation endpoints

You can use Djoser in your project to automatically create the following endpoints and functionalities for you.

Endpoint	Role	Method	Purpose
<i>/api/users</i>	No role required	<i>POST</i>	Creates a new user with name, email and password
<i>/api/users/users/me/</i>	Anyone with a valid user token	<i>GET</i>	Displays only the current user

Endpoint	Role	Method	Purpose
<i>/token/login/</i>	Anyone with a valid username and password	<i>POST</i>	Generates access tokens that can be used in other API calls in this project

When you include Djoser endpoints, Djoser will create other useful endpoints as discussed in the [Introduction to Djoser library for better authentication](#)  video.

Menu-items endpoints

Endpoint	Role	Method	Purpose
<i>/api/menu-items</i>	Customer, delivery crew	<i>GET</i>	Lists all menu items. Return a <i>200 – Ok</i> HTTP status code
<i>/api/menu-items</i>	Customer, delivery crew	<i>POST, PUT, PATCH, DELETE</i>	Denies access and returns <i>403 – Unauthorized</i> HTTP status code
<i>/api/menu-items/{menuItem}</i>	Customer, delivery crew	<i>GET</i>	Lists single menu item
<i>/api/menu-items/{menuItem}</i>	Customer, delivery crew	<i>POST, PUT, PATCH, DELETE</i>	Returns <i>403 - Unauthorized</i>
<i>/api/menu-items</i>	Manager	<i>GET</i>	Lists all menu items
<i>/api/menu-items</i>	Manager	<i>POST</i>	Creates a new menu item and returns <i>201 - Created</i>

Endpoint	Role	Method	Purpose
<i>/api/menu-items/{menuItem}</i>	Manager	<i>GET</i>	Lists single menu item
<i>/api/menu-items/{menuItem}</i>	Manager	<i>PUT, PATCH</i>	Updates single menu item
<i>/api/menu-items/{menuItem}</i>	Manager	<i>DELETE</i>	Deletes menu item

User group management endpoints

Endpoint	Role	Method	Purpose
<i>/api/groups/manager/users</i>	Manager	<i>GET</i>	Returns all managers
<i>/api/groups/manager/users</i>	Manager	<i>POST</i>	Assigns the user in the payload to the manager group and returns <i>201-Created</i>
<i>/api/groups/manager/users/{userId}</i>	Manager	<i>DELETE</i>	Removes this particular user from the manager group and returns <i>200 – Success</i> if everything is okay. If the user is not found, returns <i>404 – Not found</i>
<i>/api/groups/delivery-crew/users</i>	Manager	<i>GET</i>	Returns all delivery crew
<i>/api/groups/delivery-crew/users</i>	Manager	<i>POST</i>	Assigns the user in the payload to delivery crew group and returns <i>201-Created HTTP</i>

Endpoint	Role	Method	Purpose
<i>/api/groups/delivery-crew/users/{userId}</i>	Manager	<i>DELETE</i>	Removes this user from the manager group and returns <i>200 – Success</i> if everything is okay. If the user is not found, returns <i>404 – Not found</i>

Cart management endpoints

Endpoint	Role	Method	Purpose
<i>/api/cart/menu-items</i>	Customer	<i>GET</i>	Returns current items in the cart for the current user token
<i>/api/cart/menu-items</i>	Customer	<i>POST</i>	Adds the menu item to the cart. Sets the authenticated user as the user id for these cart items
<i>/api/cart/menu-items</i>	Customer	<i>DELETE</i>	Deletes all menu items created by the current user token

Order management endpoints

Endpoint	Role	Method	Purpose
<i>/api/orders</i>	Customer	<i>GET</i>	Returns all orders with order items created by this user
<i>/api/orders</i>	Customer	<i>POST</i>	Creates a new order item for the current user. Gets current cart items from the cart endpoints

Endpoint	Role	Method	Purpose
			and adds those items to the order items table. Then deletes all items from the cart for this user.
<i>/api/orders/{orderId}</i>	Customer	<i>GET</i>	Returns all items for this order id. If the order ID doesn't belong to the current user, it displays an appropriate HTTP error status code.
<i>/api/orders</i>	Manager	<i>GET</i>	Returns all orders with order items by all users
<i>/api/orders/{orderId}</i>	Customer	<i>PUT, PATCH</i>	<p>Updates the order. A manager can use this endpoint to set a delivery crew to this order, and also update the order status to 0 or 1.</p> <p>If a delivery crew is assigned to this order and the <i>status</i> = 0, it means the order is out for delivery.</p> <p>If a delivery crew is assigned to this order and the <i>status</i> = 1, it means the order has been delivered.</p>
<i>/api/orders/{orderId}</i>	Manager	<i>DELETE</i>	Deletes this order
<i>/api/orders</i>	Delivery crew	<i>GET</i>	Returns all orders with order items assigned to the delivery crew
<i>/api/orders/{orderId}</i>	Delivery crew	<i>PATCH</i>	A delivery crew can use this endpoint to update the order status to 0 or 1. The delivery crew will not be able to update anything else in this order.

Additional step

Implement proper filtering, pagination and sorting capabilities for */api/menu-items* and */api/orders* endpoints. Review the videos about [Filtering and searching](#) and [Pagination](#) as well as the reading [More on filtering and pagination](#).

Throttling

Finally, apply some throttling for the authenticated users and anonymous or unauthenticated users. Review the video [Setting up API throttling](#) and the reading [API throttling for class-based views](#) for guidance.

Conclusion

Now that you have a better idea of the scope of this project with the essential API endpoints, it's time to start coding. Good luck!