Features & REST API Requirement List – Foodie App (Flask + Pytest + Robot Framework)

Technology Stack

- **Backend API**: Python Flask (REST API)
- **Testing**: Pytest, Robot Framework
- **Manual Testing**: Postman
- **Automation Support**: requests library, JSON parsing

1. Restaurant Module (Flask REST APIs)

#	Requirement	HTTP Method	URI	Request Data	Response Data	Status Codes
1	Register Restaurant	POST	/api/v1/restaurants	{name, category, location, images, contact}	Restaurant JSON	201 Created, 400 Bad Request, 409 Conflict
2	Update Restaurant Details	PUT	/api/v1/restaurants/{restaurant_id}	Updated fields	Updated restaurant	200 OK, 404 Not Found
3	Disable Restaurant	PUT	/api/v1/restaurants/{restaurant_id}/dis	-	{“message”:“Restaurant disabled”}	200 OK, 404 Not Found

#	Requirement	HTTP Method	URI	Request Data	Response Data	Status Codes
			able			
4	View Restaurant Profile	GET	/api/v1/restaurants/{restaurant_id}	-	Restaurant JSON	200 OK, 404 Not Found

---

2. Dish Module

#	Requirement	HTTP Method	URI	Request Data	Response Data	Status Codes
5	Add Dish	POST	/api/v1/restaurants/{restaurant_id}/dishes	{name,type,price,available_time,image}	Dish JSON	201 Created, 400 Bad Request
6	Update Dish	PUT	/api/v1/dishes/{dish_id}	Updated fields	Updated dish	200 OK, 404 Not Found
7	Enable / Disable Dish	PUT	/api/v1/dishes/{di	{enabled:true/false}	Message	200 OK, 404 Not Found

#	Requirement	HTTP Method	URI	Request Data	Response Data	Status Codes
			sh_id}/status			
8	Delete Dish	DELETE	/api/v1/dishes/{dish_id}	-	{“message”:“Dish deleted”}	200 OK, 404 Not Found

---

3. Admin Module (Admin Routes)

#	Requirement	HTTP Method	URI	Request Data	Response Data	Status Codes
9	Approve Restaurant	PUT	/api/v1/admin/restaurants/{restaurant_id}/approve	-	Message	200 OK, 404 Not Found
10	Disable Restaurant	PUT	/api/v1/admin/restaurants/{restaur	-	Message	200 OK, 404 Not Found

#	Requirement	HTTP Method	URI	Request Data	Response Data	Status Codes
			ant_id}/disable			
11	View Customer Feedback	GET	/api/v1/admin/feedback	-	Feedback list	200 OK
12	View Order Status	GET	/api/v1/admin/orders	-	Order list	200 OK

---

4. User / Customer Module

#	Requirement	HTTP Method	URI	Request Data	Response Data	Status Codes
13	User Registration	POST	/api/v1/users/register	{name,email,password}	User JSON	201 Created, 409 Conflict
14	Search Restaurants	GET	/api/v1/restaurants/search?name=&loca	-	Restaurant list	200 OK

#	Requirement	HTTP Method	URI	Request Data	Response Data	Status Codes
			tion=&dish=&rating=			
15	Place Order	POST	/api/v1/orders	{user_id,restaurant_id,dishes}	Order JSON	201 Created, 400 Bad Request
16	Give Rating	POST	/api/v1/ratings	{order_id,rating,comment}	Rating JSON	201 Created, 400 Bad Request

---

5. Order Module

#	Requirement	HTTP Method	URI	Request Data	Response Data	Status Codes
17	View Orders by Restaurant	GET	/api/v1/restaurants/{restaurant_id}/orders	-	Order list	200 OK
18	View Orders by User	GET	/api/v1/users/{user_id}/orders	-	Order list	200 OK

API Testing Approach

Manual Testing (Postman)

- Validate request/response
- Verify status codes
- Test positive and negative scenarios

Pytest Automation

- Use `requests` library
- Validate status codes, response body, schema
- Use fixtures and parameterization

Robot Framework Automation

- Use **RequestsLibrary**
- Keyword-driven and data-driven test cases
- Separate test setup and teardown

Sample Pytest Test Case (Reference)

```
import requests

def test_add_restaurant():
    response = requests.post("http://localhost:5000/api/v1/restaurants",
json={"name":"Food Hub"})
    assert response.status_code == 201
```

Sample Robot Framework Test Case (Reference)

```
*** Settings ***
Library    RequestsLibrary

*** Test Cases ***
Add Restaurant
    Create Session    foodie    http://localhost:5000
    ${response}=    POST On Session    foodie    /api/v1/restaurants
json={"name":"Food Hub"}
    Status Should Be    201    ${response}
```

Notes for Students

- Follow REST principles
- Validate all user inputs
- Maintain proper folder structure (routes, services, models)

- Each team member must develop and test at least **4 CRUD APIs**

This document is ready to be shared with the mentor for review.