

APIT – Prompt-Friendly API Specification (v1)

1. User Service

1.1 Register

Endpoint: POST /auth/register

Request Body

```
{
  "user_name": "user name",
  "email": "example@email.com",
  "password": "password"
}
```

Response

```
{
  "result": {
    "code": "0000",
    "description": "Success"
  },
  "data": {
    "id": "01J8...",
    "user_name": "user name",
    "email": "example@email.com",
    "created_at": "2025-08-16T03:21:34Z"
  }
}
```

Request Specification	Field Name	Required	Type	Max Length	Description
	user_name	M	string	50	User name
	email	M	string	100	Valid email
	password	M	string	100	Must meet policy rules

Response Specification (data only)	Field Name	Type	Description
	id	string	UUID/ULID unique ID
	user_name	string	User name
	email	string	Registered email address
	created_at	string	ISO 8601 timestamp of creation

1.2 Login

Endpoint: POST /auth/login

Request Body

```
{
  "user_name_or_email": "example@email.com",
  "password": "password"
}
```

Response

```
{
  "result": {
    "code": "0000",
    "description": "Success"
  },
  "data": {
    "access_token": "eyJ...",
    "token_type": "Bearer",
    "expires_in": 3600,
    "scope": "tester:read tester:write"
  }
}
```

Request Specification	Field Name	Required	Type	Max Length	Description
----- ----- ----- ----- -----	user_name_or_email	M	string	100	Username or email login
	password	M	string	100	User password

Response Specification (data only)	Field Name	Type	Description
----- ----- -----	access_token	string	JWT bearer token
	token_type	string	Always Bearer
	expires_in	integer	Lifetime in seconds
	scope	string	Granted scopes (space separated)

1.3 Logout

Endpoint: POST /auth/logout

Response

```
{
  "result": {
    "code": "0000",
    "description": "Logged out"
  },
  "data": {}
}
```

1.4 Refresh Token

Endpoint: POST /auth/refresh

Response

```
{
  "result": {
    "code": "0000",
    "description": "Success"
  },
  "data": {
    "access_token": "eyJ...",
    "expires_in": 3600
  }
}
```

1.5 Get Current User

Endpoint: GET /auth/me

Response

```
{
  "result": {
    "code": "0000",
    "description": "Success"
  },
  "data": {
    "id": "01J8...",
    "user_name": "namtran",
    "roles": ["tester"]
  }
}
```

1.6 Change Password

Endpoint: POST /auth/change-password

Request Body

```
{
  "old_password": "oldpass",
```

```
"new_password": "NewPass123"
}
```

Response

```
{
  "result": {
    "code": "0000",
    "description": "Password changed"
  },
  "data": {}
}
```

2. Payment Service

2.1 Create Payment Intent

Endpoint: POST /payments/intents

Request Body

```
{
  "amount": 100000,
  "currency": "VND",
  "method": "card"
}
```

Response

```
{
  "result": {
    "code": "0000",
    "description": "Created"
  },
  "data": {
    "payment_id": "pay_123",
    "status": "pending"
  }
}
```

2.2 Capture Payment

Endpoint: POST /payments/{id}/capture

Response

```
{
  "result": {"code": "0000", "description": "Captured"},
  "data": {"payment_id": "pay_123", "status": "succeeded"}
}
```

2.2.1 Idempotency Support

- Header: Idempotency-Key: <UUID>
-

2.3 Refund Payment

Endpoint: POST /payments/{id}/refund

Request Body

```
{
  "amount": 50000
}
```

Response

```
{
  "result": {"code": "0000", "description": "Refunded"},
  "data": {"refund_id": "ref_456", "status": "succeeded"}
}
```

2.4 Get Payment

Endpoint: GET /payments/{id}

Response

```
{
  "result": {"code": "0000", "description": "Success"},
  "data": {"id": "pay_123", "amount": 100000, "status": "succeeded"}
}
```

3. Reporting

3.1 Summary

Endpoint: GET /reports/summary?from=2025-01-01&to=2025-01-31&group_by=user

Response

```
{
  "result": {"code": "0000", "description": "Success"},
  "data": [{"user": "namtran", "count": 12, "amount": 1200000}]
}
```

3.2 Failure Details

Endpoint: GET /reports/failures

Response

```
{
  "result": {"code": "0000", "description": "Success"},
  "data": [{"id": "pay_456", "reason": "insufficient_funds"}]
}
```

4. Business Codes

Code	HTTP	Description
0000	200	Success
1001	400	Missing required field
1002	400	Invalid email format
1003	400	Password policy failed
2001	401	Invalid credentials
2002	401	Token expired
3001	409	Email already exists
3003	409	Username already exists
4003	403	User blocked

5. Prompt Testing Examples

- **Task:** Extract all `User Service` endpoints.
- **Task:** Generate test cases for `Register` API including missing field scenarios.
- **Task:** Summarize Payment endpoints and their required fields.
- **Task:** Return only Business Codes with HTTP mapping.