

BACKEND DEVELOPMENT

2U - Tala Diab, Reina Najdi, Roy Yammine

PROJECT OVERVIEW

2U is a service matching platform connecting users with providers.

Backend responsibilities:

- User authentication and authorization
- Service request management
- Provider matching algorithm
- Database management

TECHNOLOGY STACK

Backend Technologies:

FastAPI - main web framework

SQLite - SQL database

SQLAlchemy - ORM for database operations and model definitions

Pydantic - data validation for request/response schemas

Uvicorn - ASGI server

WebSockets - for real time chat

JWT - authentication

Python-multipart - file uploads

Development Tools:

- GitHub
- Virtual Environment
- Docker

2U Backend API 1.0.0 OAS 3.1

/openapi.json

On-road assistance platform: users, providers, requests, and chat.

Authorize



Auth



POST

/auth/register Register



Parameters

Cancel

Reset

No parameters

Request body required

application/json



Edit Value | Schema

```
POST /auth/register
{
  "full_name": "John Doe",
  "email": "john@example.com",
  "password": "1234",
  "role": "client"
}
```

FastAPI Auto generated documentation

2U

DATABASE DESIGN

Data Models (SQLAlchemy):

- User: id, email , password, full_name, role, created_at
- ServiceRequest: id, description, status, user_id, created_at, updated_at
- Message: id, content, sender_id, request_id, timestamp
- FileAttachment: id, filename, file_path, request_id, uploaded_at

API Schemas (Pydantic):

- Login - authentication request
- MsgCreate/MsgOut - chat system
- RequestStatus - request state tracking
- SRCreate/SROut - service request flow
- SRUpdate
- Token, TokenData
- UserCreate/UserOut - user registration/responses
- UserRole - role management

Key Features:

- Role based access (client/provider)
- Request status tracking
- Real time messaging
- Secure data validation



Pydantic schemas

API ENDPOINTS

Authentication:

POST /auth/register - User registration

POST /auth/login - User login

Service Management:

POST /requests/ - Create service request

GET /requests/ - Get user requests

GET /requests/{id} - Get specific request

PUT /requests/{id} - Update request

POST /requests/{id}/accept - Provider accepts requests

POST /request/{id}/complete - Mark request as complete

Messages:

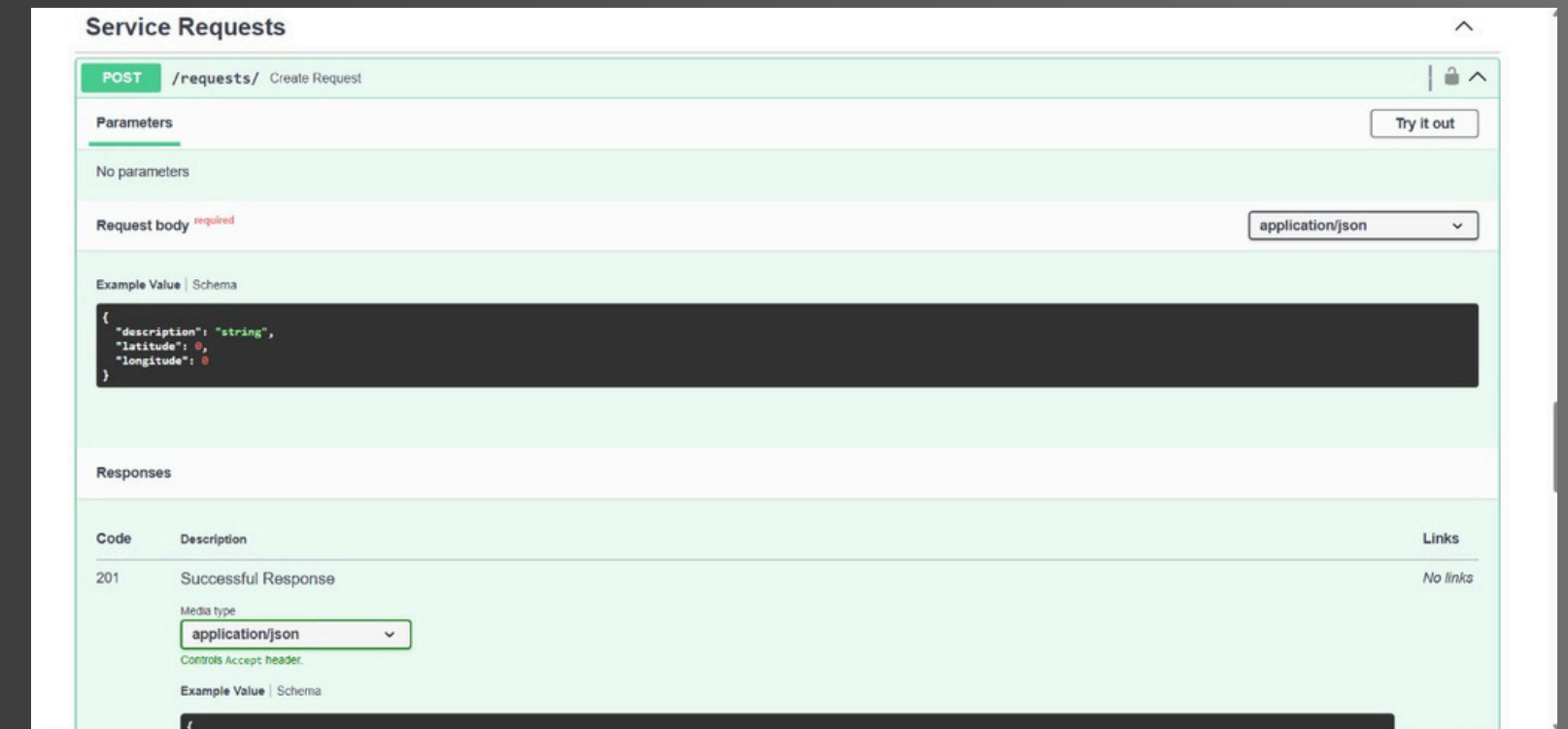
POST /messages/ - Send message

GET /messages/{request_id} - Get chat history

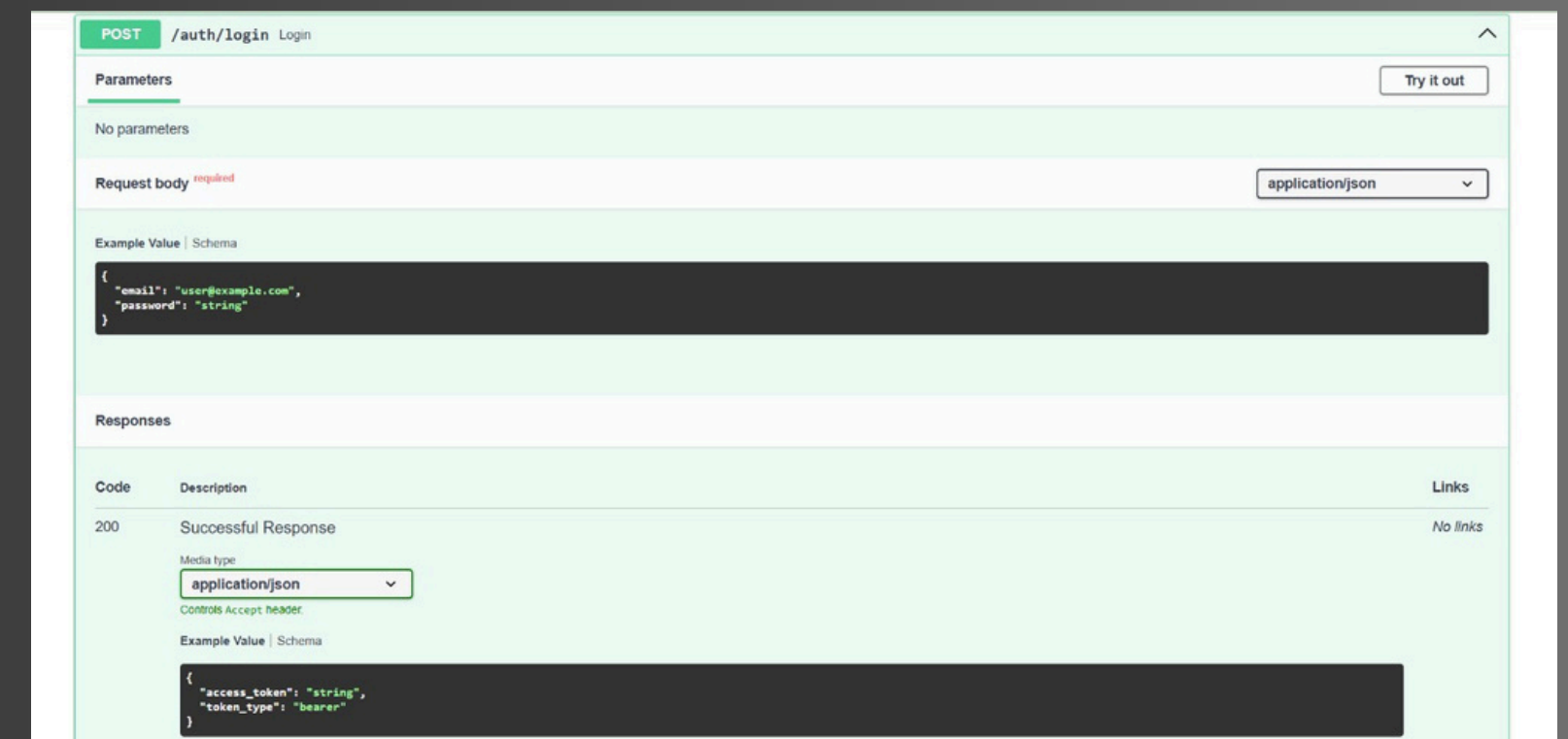
WebSocket /ws/{request_id} - Real time messaging

Files:

POST /requests/{id}/upload - Upload file attachment



API Endpoints - Service requests and authentication



API Access example

Request:

POST /auth/register

```
{  
  "email": "john@example.com",  
  "password": "1234",  
  "full_name": "John Doe",  
  "role": "client"  
}
```

Response:

```
{  
  "id": 1,  
  "full_name": "John Doe",  
  "email": "john@example.com",  
  "role": "client"  
}  
  
{  
  "access_token":  
  "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9....",  
  "token_type": "bearer"
```

Error Handling - Example: Invalid Login

```
{  
  "detail": "Invalid credentials"  
}
```

Features:

- HTTPException for meaningful error messages
- Pydantic automatic data validation
- Proper status codes (400, 401, 404, 500)
- Consistent error response format

Backend
running locally

```
Windows PowerShell
Requirement already satisfied: websockets>=10.4 in c:\users\royya\desktop\fall 2025-2026\cmps 279\project\2u-backend\.venv\lib\site-packages (from uvicorn[standard]) (15.0.1)
Requirement already satisfied: starlette<0.41.0,>=0.37.2 in c:\users\royya\desktop\fall 2025-2026\cmps 279\project\2u-backend\.venv\lib\site-packages (from fastapi) (0.40.0)
Requirement already satisfied: pydantic!=1.8,!1.8.1,!2.0.0,!2.0.1,!2.1.0,<3.0.0,>=1.7.4 in c:\users\royya\desktop\fall 2025-2026\cmps 279\project\2u-backend\.venv\lib\site-packages (from fastapi) (2.8.2)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\royya\desktop\fall 2025-2026\cmps 279\project\2u-backend\.venv\lib\site-packages (from fastapi) (4.15.0)
Requirement already satisfied: annotated-types>=0.4.0 in c:\users\royya\desktop\fall 2025-2026\cmps 279\project\2u-backend\.venv\lib\site-packages (from pydantic!=1.8,!1.8.1,!2.0.0,!2.0.1,!2.1.0,<3.0.0,>=1.7.4->fastapi) (0.7.0)
Requirement already satisfied: pydantic-core==2.20.1 in c:\users\royya\desktop\fall 2025-2026\cmps 279\project\2u-backend\.venv\lib\site-packages (from pydantic!=1.8,!1.8.1,!2.0.0,!2.0.1,!2.1.0,<3.0.0,>=1.7.4->fastapi) (2.20.1)
Requirement already satisfied: anyio<5,>=3.4.0 in c:\users\royya\desktop\fall 2025-2026\cmps 279\project\2u-backend\.venv\lib\site-packages (from starlette<0.41.0,>=0.37.2->fastapi) (4.11.0)
Requirement already satisfied: idna>=2.8 in c:\users\royya\desktop\fall 2025-2026\cmps 279\project\2u-backend\.venv\lib\site-packages (from anyio<5,>=3.4.0->starlette<0.41.0,>=0.37.2->fastapi) (3.11)
Requirement already satisfied: sniffio>=1.1 in c:\users\royya\desktop\fall 2025-2026\cmps 279\project\2u-backend\.venv\lib\site-packages (from anyio<5,>=3.4.0->starlette<0.41.0,>=0.37.2->fastapi) (1.3.1)
PS C:\Users\royya\Desktop\Fall 2025-2026\CMPS 279\Project\2u-backend> .\.venv\Scripts\python.exe -m uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
INFO:      Will watch for changes in these directories: ['C:\\Users\\royya\\Desktop\\Fall 2025-2026\\CMPS 279\\Project\\2u-backend']
INFO:      Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:      Started reloader process [10428] using WatchFiles
INFO:      Started server process [16688]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      127.0.0.1:58158 - "GET /docs HTTP/1.1" 200 OK
INFO:      127.0.0.1:58158 - "GET /openapi.json HTTP/1.1" 200 OK
```


Containerized with Docker

- Multi-stage Docker build for optimized images
- Docker Compose for easy local development
- Production ready for configuration

Deployment Setup

`docker-compose up --build` (single command to build & run)

`docker build -t 2u-backend .`

`docker run -d -p 8000:80 2u-backend`

Environment Configuration

`DATABASE_URL=sqlite:///./2u.db`

`JWT_SECRET=jwt-secret-key`

`JWT_REFRESH_SECRET=refresh_secret`

`ACCESS_TOKEN_EXPIRE_MINUTES=30`

Supported Platforms

- Render
- Railway
- Google Cloud Run
- Any Docker-supported platforms

DEPLOYMENT

Development Process

1. Database design - models & relationships
2. API Schema design - request/response format
3. Endpoint implementation
4. Testing and validation (error handling)

GitHub repository: github.com/reinanaajdi/2u-backend