Django Back-End REST API Documentation

- Starting the Django Back-End
    - Navigate to the back-end directory → cd application/backend/project
    - Initialise python virtual environment → python -m venv venv
    - Activate python virtual environment → venv/Scripts/activate
    - Install all dependencies → pip install -r requirements.txt
    - Create a new .env file with the following variables
        - ENVIRONMENT="DEVELOPMENT”
        - DJANGO_SECRET_KEY="yaRF6LbTNXxYsjZILFjPYhLINtPjaXYXhuhIG gvNCSFJ4j3Wunzfjq9ktVv71jSM"
        - DJANGO_JWT_SIGNING_KEY="eeM4FDLiN5jCcSVf4WoTKjgephBkR8x jwx5YYFgDEzmrJWNJwBh6txMN3XvZ8j7S"
        - DJANGO_JWT_ALGORITHM="HS512"
        - DJANGO_JWT_LIFETIME="14"
        - DJANGO_BACKEND_URL="http://127.0.0.1:8000"
        - DJANGO_DEBUG="True"
        - GMAP_SECRET="AIzaSyA8dLw5Ucv1Nyc5GMgzviVX-i8fxmjvu2M"
        - GEMINI_API="AIzaSyCiDY26tqvfy0ITtiImWoDKcMjiI4tjI2U"
    - Run the back-end server → python manage.py runserver

- Accessing the Back-End Documentation
    - If you have ran the django server
        - In the browser, navigate to http://127.0.0.1:8000/documentation/
    - If you are not running the django server
        - In the browser, navigate to our live deployed back-end documentation at https://studybuddybe.up.railway.app/documentation/
    - Enter the admin credentials
        - Username = ‘admin’
        - Password = ‘admin’
    - Main documentation created will be those for ‘models’ and ‘views’

- Accessing the Admin Console
    - If you have ran the django server
        - In the browser, navigate to http://127.0.0.1:8000/admin/
    - If you are not running the django server
        - In the browser, navigate to our live deployed back-end documentation at https://studybuddybe.up.railway.app/admin/
    - Enter the admin credentials
        - Username = ‘admin’
        - Password = ‘admin’

1. Users → /api/users/…
   - 1.1. POST /api/users/auth/login/ → Login a user, obtain access token
     - Body
       - “username”: “<string>”
       - “password”: “<string>”

   - 1.2. POST /api/user/auth/refresh/ → Obtain new access token
     - Header
       - “Authorization”: “Bearer <access token>”
     - Body
       - “refresh”: “<refresh token>

   - 1.3. POST /api/users/auth/register/ → Register a new user
     - Body
       - “username”: “<string>”
       - “password1”: “<string>”
       - “password2”: “<string>”
       - “email”: “<email>”

   - 1.4. GET /api/users/profile/ → Get the user’s profile
     - Header
       - “Authorization”: “Bearer <access token>”

   - 1.5. POST /api/users/profile/ → Update the user’s profile
     - Header
       - “Authorization”: “Bearer <access token>”
     - Body
       - “first_name”: “<string>”
       - “last_name”: “<string>”
       - “email”: “<email>”
       - “mobile_number”: “+<numbers>”

2. Bookings → /api/bookings/…
   - 2.1. GET /api/bookings/ → Get all the user’s bookings
     - Header
       - “Authorization”: “Bearer <access token>”

   - 2.2. POST /api/bookings/ → Create a new booking for the user
     - Header
       - “Authorization”: “Bearer <access token>”
     - Body
       - “date”: “<date in YYYY-MM-DD>”
       - “start_time”: “<time in HH:MM>”
       - “end_time”: “<time in HH:MM>”
       - “study_area”: <pk for study area>

2.3. GET /api/bookings/<int:pk>/ → Get a specific booking from the user

- Header
    - “Authorization”: “Bearer <access token>”
- Key
    - pk: <integer for booking pk>

2.4. PUT /api/bookings/<int:pk>/ → Update specific booking from the user

- Header
    - “Authorization”: “Bearer <access token>”
- Body
    - “date”: “<date in YYYY-MM-DD>”
    - “start_time”: “<time in HH:MM>”
    - “end_time”: “<time in HH:MM>”
- Key
    - pk: <integer for booking pk>

2.5. DELETE /api/bookings/<int:pk>/ → Delete a specific booking from the user

- Header
    - “Authorization”: “Bearer <access token>”
- Key
    - pk: <integer for booking pk>

3. Cart → /api/cart/…

3.1. GET /api/cart/ → Get the user’s cart

- Header
    - “Authorization”: “Bearer <access token>”

3.2. POST /api/cart/ → Add a product to the user’s cart

- Header
    - “Authorization”: “Bearer <access token>”
- Body
    - “product”: <integer for product pk>
    - “quantity”: <integer>

3.3. PUT /api/cart/ → Edit an item in the user’s cart

- Header
    - “Authorization”: “Bearer <access token>”
- Body
    - “product”: <integer for product pk>
    - “quantity”: <integer>

3.4. POST /api/cart/checkout/ → Checkout the user’s cart

- Header
    - “Authorization”: “Bearer <access token>”
- Body
    - “address”: “<string>”
    - “postal_code”: “<string of 6 characters>”

4. Orders → /api/orders/…
    4.1. GET /api/orders/ → Get the user’s confirmed orders
        - Header
            - “Authorization”: “Bearer <access token>”

    4.2. GET /api/orders/<int:pk>/ → Get a specific order from the user
        - Header
            - “Authorization”: “Bearer <access token>”
        - Key
            - pk: <integer for order pk>

5. Products → /api/products/…
    5.1. GET /api/products/ → Get all available products
    5.2. GET /api/products/<int:pk>/ → Get a specific product

6. Study Areas → /api/study_areas/…
    6.1. GET /api/study_areas/ → Get all available study areas
    6.2. GET /api/study_areas?lat=<float:lat>&lng=<float:lng> → Get all available study areas with distance field calculated using given coordinates
    6.3. GET /api/study_areas/<int:pk>/ → Get a specific study area
    6.4. GET /api/study_areas/<int:pk>/?lat=<float:lat>&lng=<float:lng> → Get a specific study area with distance field calculated using given coordinates

7. Buddy AI → /api/buddyai/
    7.1. POST /api/buddyai/ → Ask buddy AI a question
        - Header
            - “Authorization”: “Bearer <access token>”
    7.2. Body
        - “question”: “<string>”