Task 1.2

Overview

This specification defines a FastAPI application for managing satellites, including CRUD operations. Satellites operate in circular orbits with arbitrary inclination and right-ascension-of-ascending-node (RAAN), represented by an Orbit resource.

Expected Format

Submit a single Python file that defines a FastAPI application with all required endpoints. The file must expose an `app` variable (the FastAPI instance). Use an in-memory database for all data storage; no external or persistent databases.

Endpoint Summary Task 1.1

Method	Endpoint	Description
GET	`/health`	Health check
POST	`/orbits/`	Create new orbit
GET	`/orbits/{id}`	Get orbit by ID
GET	`/orbits/`	List orbits with pagination
PUT	`/orbits/{id}`	Update orbit
DELETE	`/orbits/{id}`	Delete orbit
POST	`/satellites/`	Create new satellite
GET	`/satellites/{id}`	Get satellite by ID
GET	`/satellites/`	List satellites with pagination
PUT	`/satellites/{id}`	Update satellite
DELETE	`/satellites/{id}`	Delete satellite

Endpoint Summary Task 1.2

Method	Endpoint	Description
GET	`/satellites/{id}/position`	Get satellite position at given time

Environment

Python Version: `3.12`

Allowed Libraries:

- `fastapi >= 0.116.1`
- `uvicorn >= 0.32.1`
- `pydantic >= 2.11.7`
- `sqlalchemy >= 2.0.41`
- `python-dateutil >= 2.8.2`

- `scipy >= 1.16.0`
- `numpy >= 2.3.1`

Task 1.1

GET /health

Health check

Response (200):

```
{
  "status": "healthy"
}
```

GET /orbits/{id}

Retrieve orbit by ID

Path Parameters:

- `id`: integer, positive

Response (200):

```
{
  "id": 1,
  "name": "Starlink-Shell-1",
  "orbital_altitude": 550.0,
  "inclination": 53.0,
  "raan": 120.0
}
```

Response (400): `{ "detail": "Invalid ID format" }`

Response (404): `{ "detail": "Orbit not found" }`

POST /orbits/

Create a new orbit

Request Body:

```
{
  "name": "Starlink-Shell-1",
  "orbital_altitude": 550.0,
  "inclination": 53.0,
  "raan": 120.0
}
```

Response (201): Body identical to GET /orbits/{id}

Response (409): `{ "detail": "Orbit name already exists" }`

Validation:

- `name`: string, 1-100 chars, unique
- `orbital_altitude`: float, 160 < value ≤ 40000 (km)
- `inclination`: float, 0 ≤ value ≤ 180 (deg)
- `raan`: float, 0 ≤ value < 360 (deg)

GET /orbits/

List orbits with pagination

Query Parameters:

- `skip`: integer, default 0, min 0
- `limit`: integer, default 10, max 100, min 1
- `name`: optional string, case-insensitive contains filter

Response (200):


```
{
  "orbits": [
    {
      "id": 1,
      "name": "Starlink-Shell-1",
      "orbital_altitude": 550.0,
      "inclination": 53.0,
      "raan": 120.0,
    }
  ],
  "total": 1,
  "skip": 0,
  "limit": 10
}
```


Response (400): `{ "detail": "Invalid pagination parameters" }`

PUT /orbits/{id}

Update orbit

Path Parameters:

- `id`: integer, positive

Request Body: Same as POST, all fields required.

Response (200): Same as GET /orbits/{id}

Response (400): `{ "detail": "Invalid ID format or invalid data" }`

Response (404): `{ "detail": "Orbit not found" }`

Response (409): `{ "detail": "Orbit name already exists" }`

Notes:

- Full update; all fields must be provided.

DELETE /orbits/{id}

Delete orbit

Path Parameters:

- `id`: integer, positive

Response (204): No content

Response (400): `{ "detail": "Invalid ID format" }`

Response (404): `{ "detail": "Orbit not found" }`

Response (409): `{ "detail": "Orbit in use by satellites" }`

POST /satellites/

Create a new satellite

Request Body:

```
{
  "name": "Starlink-1234",
  "operator": "SpaceX",
  "launch_date": "2024-01-01T00:00:00Z",
  "status": "active",
  "initial_longitude": -74.0060,
  "orbit_id": 1
}
```

Response (201):

```
{
  "id": 1,
  "name": "Starlink-1234",
  "operator": "SpaceX",
  "launch_date": "2024-01-01T00:00:00Z",
  "status": "active",
  "initial_longitude": -74.0060,
  "orbit_id": 1
}
```

Response (409):

```
{ "detail": "Satellite name already exists" }
```

Validation:

- `name`: string, 1-100 chars, unique (checked via database constraint)
- `operator`: string, 1-50 chars
- `launch_date`: ISO-8601 UTC datetime, must be in the past
- `status`: optional, ["active", "inactive", "deorbited"], default "active"
- `orbit_id`: integer, must reference an existing Orbit
- `initial_longitude`: float, -180 to 180 (degrees)

GET /satellites/{id}

Retrieve satellite by ID

Path Parameters:

- `id`: integer, positive

Response (200):

```
{
  "id": 1,
  "name": "Starlink-1234",
  "operator": "SpaceX",
  "launch_date": "2024-01-01T00:00:00Z",
  "status": "active",
  "initial_longitude": -74.0060,
  "orbit_id": 1
}
```

Response (400):

```
{ "detail": "Invalid ID format" }
```

Response (404):

```
{ "detail": "Satellite not found" }
```

GET /satellites/

List satellites with pagination

Query Parameters:

- `skip`: integer, default 0, min 0
- `limit`: integer, default 10, max 100, min 1
- `operator`: optional string, case-insensitive

Response (200):

```
{
  "satellites": [
    {
      "id": 1,
      "name": "Starlink-1234",
      "operator": "SpaceX",
      "launch_date": "2024-01-01T00:00:00Z",
      "status": "active",
      "initial_longitude": -74.0060,
      "orbit_id": 1
    }
  ],
  "total": 1,
  "skip": 0,
  "limit": 100
}
```

Response (400):

```
{ "detail": "Invalid pagination parameters" }
```

PUT /satellites/{id}

Update satellite

Path Parameters:

- `id`: integer, positive

Request Body: Same as POST, all fields required.

Response (200): Same as GET /satellites/{id}

Response (400):

```
{ "detail": "Invalid ID format or invalid data" }
```

Response (404):

```
{ "detail": "Satellite not found" }
```

Response (409):

```
{ "detail": "Satellite name already exists" }
```

Notes:

- Full update; all fields must be provided.
- `created_at` cannot be updated.

DELETE /satellites/{id}

Delete satellite

Path Parameters:

- `id`: integer, positive

Response (204): No content

Response (400):

```
{ "detail": "Invalid ID format" }
```

Response (404):

```
{ "detail": "Satellite not found" }
```

Task 1.2

GET /satellites/{id}/position

Get satellite position at given time

Path Parameters:

- `id`: integer, positive

Query Parameters:

- `timestamp`: required, ISO-8601 UTC datetime

Behavior:

- If `timestamp` < `launch_date`, return 400:

```
{ "detail": "Timestamp before launch date" }
```

- If `timestamp` is malformed, return 400:

```
{ "detail": "Invalid timestamp format" }
```

- Position calculation (circular inclined orbit) (Wikipedia):
- **Simplified Model**: This simulation assumes the satellite instantly appears at `initial_longitude` when `launch_date` occurs. In reality, satellites launch from specific locations and follow complex injection orbits before reaching their target orbit.
  - Angles in radians (standard for orbital mechanics calculations)
  - Input parameters (inclination, raan, initial_longitude) are provided in degrees via API but converted to radians for calculations
  - ω = 2 * π / T, T = 2 * π * sqrt(a^3 / μ)
  - θ = ($\omega * \Delta t$ + initial_longitude_r) % (2 * π)
  - lat_r = asin(sin(inclination_r) * sin(θ))
  - wrap180(x) = ((x + 180) % 360) - 180
  - lon_r = atan2(cos(inclination_r) * sin(θ), cos(θ)) + raan_r
  - lat = lat_r * 180 / π
  - lon = wrap180(lon_r * 180 / π)
  - alt = orbital_altitude (km)
    - Δt = (timestamp - launch_date) in seconds
    - a = R_earth + orbital_altitude (km)
    - R_earth = 6371 km, μ = 398600.4418 km^3/s^2

Response (200):

```
{
  "lat": 0.0,
  "lon": -74.0060,
  "alt": 550.0
}
```

Response (400):

```
{ "detail": "Invalid ID format or timestamp" }
```

Response (404):

```
{ "detail": "Satellite not found" }
```

AI SPACE MARINE BOOTCAMP