Sprint Completion Status Report

Student Name: [Name]

Sprint Number: [Sprint 0]

Duration: [Start Date] – [End Date]

Report Date: [Date]

1. Sprint Goal 🎯

Defined Goal:

1. Clone Professor Ferguson's *Simple Microservices Repository.*
2. Create a project that is my version using two different resources.
   a. Copy the structure of Professor Ferguson's repository
   b. Define two models.
   c. Implement "API first" definition by implementing placeholder routes for each resource:
      i. GET /<resource>
      ii. POST /<resource>
      iii. GET /<resource>/{id}
      iv. PUT /<resource>/{id}
      v. DELETE /<resource>/{id}
   d. Annotate models and paths to autogenerate OpenAPI document.
   e. Tested OpenAPI document dispatching to methods.

Outcome: [Achieved / Partially Achieved / Not Achieved]

Notes: [Brief explanation if not fully achieved]

2. Completed Work ✅

Resource 1

Note: replace with your model.

```
class PersonBase(BaseModel):
    uni: UNIType = Field(
        ...,
        description="Columbia University UNI (2-3 lowercase letters
+ 1-4 digits).",
        json_schema_extra={"example": "abc1234"},
    )
    first_name: str = Field(
        ...,
        description="Given name.",
        json_schema_extra={"example": "Ada"},
    )
    last_name: str = Field(
        ...,
        description="Family name.",
        json_schema_extra={"example": "Lovelace"},
    )
    email: EmailStr = Field(
        ...,
        description="Primary email address.",
        json_schema_extra={"example": "ada@example.com"},
    )
    phone: Optional[str] = Field(
        None,
        description="Contact phone number in any reasonable
format.",
        json_schema_extra={"example": "+1-212-555-0199"},
    )
    birth_date: Optional[date] = Field(
        None,
        description="Date of birth (YYYY-MM-DD).",
        json_schema_extra={"example": "1815-12-10"},
    )
```

Resource 2

```
class AddressBase(BaseModel):
    id: UUID = Field(
        default_factory=uuid4,
        description="Persistent Address ID (server-generated).",
        json_schema_extra={"example": "550e8400-e29b-41d4-a716-
446655440000"},
    )
    street: str = Field(
        ...,
        description="Street address and number.",
        json_schema_extra={"example": "123 Main St"},
    )
    city: str = Field(
        ...,
        description="City or locality.",
        json_schema_extra={"example": "New York"},
    )
    state: Optional[str] = Field(
        None,
        description="State/region code if applicable.",
        json_schema_extra={"example": "NY"},
    )
    postal_code: Optional[str] = Field(
        None,
        description="Postal or ZIP code.",
        json_schema_extra={"example": "10001"},
    )
    country: str = Field(
        ...,
        description="Country name or ISO label.",
        json_schema_extra={"example": "USA"},
    )

    model_config = {
        "json_schema_extra": {
            "examples": [
                {
                    "id": "550e8400-e29b-41d4-a716-446655440000",
                    "street": "123 Main St",
                    "city": "New York",
                    "state": "NY",
                    "postal_code": "10001",
                    "country": "USA",
                }
            ]
        }
    }
```

main.py Routes

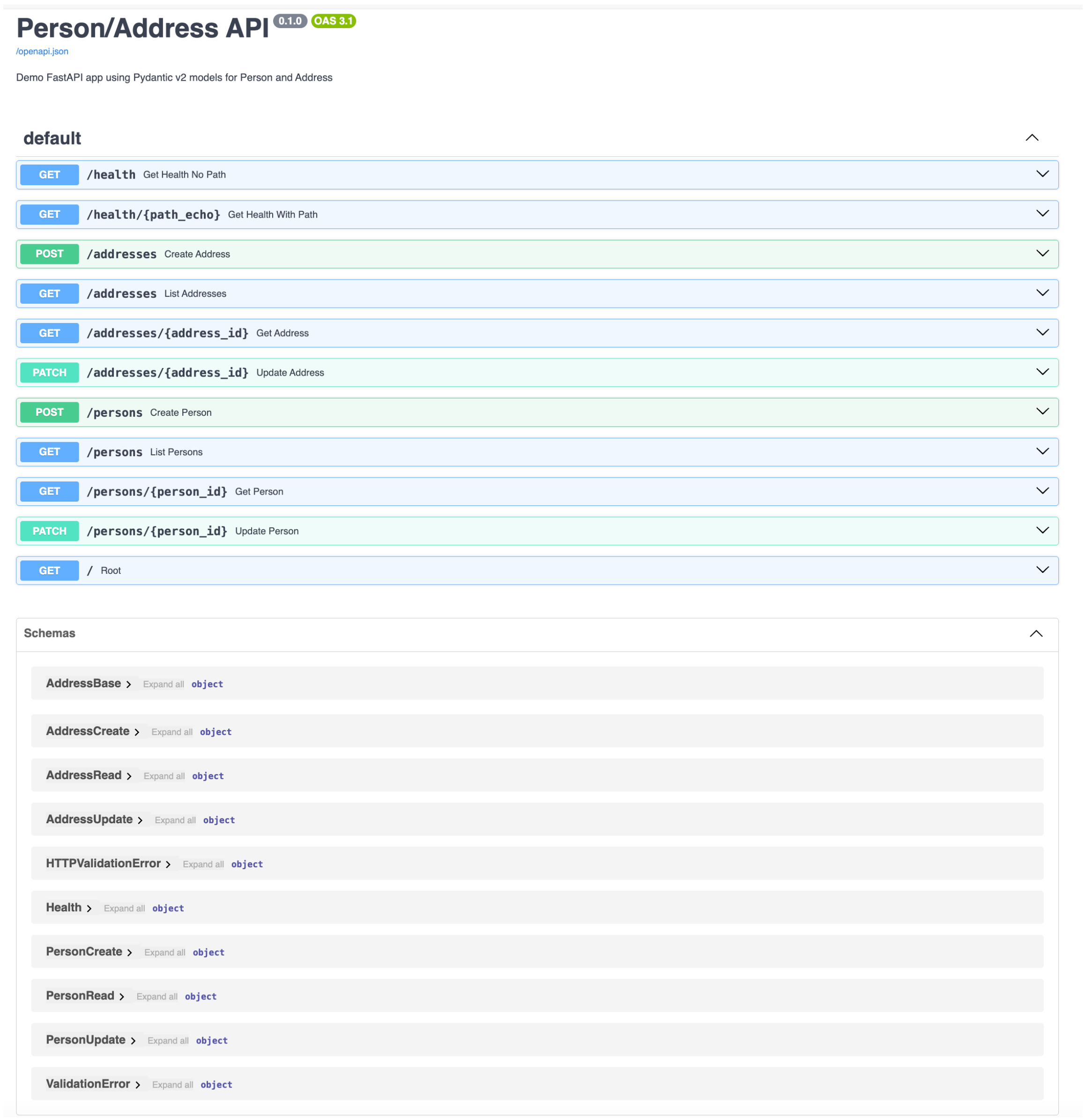
Note: Cut and paste your routes here.

```
@app.get("/health", response_model=Health)
def get_health_no_path(echo: str | None = Query(None,
description="Optional echo string")):
    # Works because path_echo is optional in the model
    return make_health(echo=echo, path_echo=None)

@app.get("/health/{path_echo}", response_model=Health)
def get_health_with_path(
    path_echo: str = Path(..., description="Required echo in the URL
path"),
    echo: str | None = Query(None, description="Optional echo
string"),
):
    return make_health(echo=echo, path_echo=path_echo)

@app.post("/addresses", response_model=AddressRead, status_code=201)
def create_address(address: AddressCreate):
    if address.id in addresses:
        raise HTTPException(status_code=400, detail="Address with
this ID already exists")
    addresses[address.id] = AddressRead(**address.model_dump())
    return addresses[address.id]
```

OpenAPI Document (Partial)



Person/Address API 0.1.0 OAS 3.1

/openapi.json

Demo FastAPI app using Pydantic v2 models for Person and Address

default

GET `/health` Get Health No Path

GET `/health/{path_echo}` Get Health With Path

POST `/addresses` Create Address

GET `/addresses` List Addresses

GET `/addresses/{address_id}` Get Address

PATCH `/addresses/{address_id}` Update Address

POST `/persons` Create Person

GET `/persons` List Persons

GET `/persons/{person_id}` Get Person

PATCH `/persons/{person_id}` Update Person

GET `/` Root

Schemas

AddressBase › Expand all `object`

AddressCreate › Expand all `object`

AddressRead › Expand all `object`

AddressUpdate › Expand all `object`

HTTPValidationError › Expand all `object`

Health › Expand all `object`

PersonCreate › Expand all `object`

PersonRead › Expand all `object`

PersonUpdate › Expand all `object`

ValidationError › Expand all `object`

Link to Recording of Demo

Note: A link to a publicly accessible screen recording that the TAs can view.

Link to GitHub Repository

Note: A link to the GitHub repo for your starter project.

3. Incomplete Work ❌

Items planned but not completed:

- [Story ID] – [Reason: e.g., dependency, scope creep, capacity issue]

- [Bug Fix ID] – [Reason]

Carryover to Next Sprint: [Yes/No, specify items]

4. Key Metrics 📊

Note: Ignore this section

Planned vs. Completed Points: [e.g., 40 planned / 35 completed]

Burndown Chart: [Attach image if available]

Defects Identified: [Number + Severity]

5. Risks & Blockers ⚠️

Note: Ignore this section

- [Risk/Issue] – [Impact] – [Mitigation/Resolution]

- [Dependency on X team] – [Impact on timeline]

6. Team Feedback 💬

Note: Ignore this section

What Went Well:

- [Positive note 1]

- [Positive note 2]

What Could Be Improved:

- [Improvement area 1]

- [Improvement area 2]

7. Next Steps 🔜

Note: Ignore this section

Upcoming Sprint Goal (Draft): [Proposed goal]

Focus Areas: [e.g., technical debt, new feature, stabilization]

Planned Dependencies: [Cross-team items, external blockers]