# Sprint Completion Status Report

\*\*Student Name:\*\* Kusuma Jaipiam

\*\*Sprint Number:\*\* Sprint 0

\*\*Duration:\*\* Sep 7 – 14, 2025

\*\*Report Date:\*\* Sep 14, 2025

## 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.
   1. Copy the structure of Professor Ferguson’s repository
   2. Define two models.
   3. Implement “API first” definition by implementing placeholder routes for each resource:
      1. GET /<resource>
      2. POST /<resource>
      3. GET /<resource>/{id}
      4. PUT /<resource>/{id}
      5. DELETE /<resource>/{id}
   4. Annotate models and paths to autogenerate OpenAPI document.
   5. Tested OpenAPI document dispatching to methods.

\*\*Outcome:\*\* Achieved

## 2. Completed Work ✅

### Resource 1: Coursework

class CourseworkBase(BaseModel):

title: str = Field(

...,

description="Title of the coursework.",

json\_schema\_extra={"example": "Cloud Computing"},

)

semester: str = Field(

...,

description="Latest semester when the coursework is offered.",

json\_schema\_extra={"example": "Fall 2025"},

)

professor: PersonBase = Field(

...,

description="Professor overseeing the coursework (carries a persistent uni).",

json\_schema\_extra={

"example": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

}

}

)

people: List[PersonBase] = Field(

default\_factory=list,

description="People linked to this coursework (each carries a persistent uni).",

json\_schema\_extra={

"example": [

{

"uni": "abc1234",

"first\_name": "Ada",

"last\_name": "Lovelace",

"email": "ada@example.com",

"addresses": [

{

"id": "550e8400-e29b-41d4-a716-446655440000",

"street": "123 Main St",

"city": "London",

"state": None,

"postal\_code": "SW1A 1AA",

"country": "UK",

}

]

}

]

},

)

model\_config = {

"json\_schema\_extra": {

"examples": [

{

"title": "Cloud Computing",

"semester": "Fall 2025",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

},

"people": [

{

"uni": "abc1234",

"first\_name": "Ada",

"last\_name": "Lovelace",

"email": "ada.lovelace@example.com",

"addresses": [

{

"id": "550e8400-e29b-41d4-a716-446655440000",

"street": "123 Main St",

"city": "London",

"state": None,

"postal\_code": "SW1A 1AA",

"country": "UK",

}

]

}

]

}

]

}

}

class CourseworkCreate(CourseworkBase):

"""Creation payload for a Coursework."""

model\_config = {

"json\_schema\_extra": {

"examples": [

{

"title": "Cloud Computing",

"semester": "Fall 2025",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

},

"people": [

{

"uni": "abc1234",

"first\_name": "Ada",

"last\_name": "Lovelace",

"email": "ada.lovelace@example.com",

"addresses": [

{

"id": "550e8400-e29b-41d4-a716-446655440000",

"street": "123 Main St",

"city": "London",

"state": None,

"postal\_code": "SW1A 1AA",

"country": "UK",

}

]

}

]

}

]

}

}

class CourseworkUpdate(BaseModel):

"""Partial update for a Coursework; supply only fields to change."""

title: Optional[str] = Field(

None,

description="New title of the coursework.",

json\_schema\_extra={"example": "Advanced Cloud Computing"},

)

semester: Optional[str] = Field(

None,

description="New semester when the coursework is offered.",

json\_schema\_extra={"example": "Spring 2026"},

)

professor: Optional[PersonBase] = Field(

None,

description="New professor overseeing the coursework (carries a persistent uni).",

)

people: Optional[List[PersonBase]] = Field(

None,

description="Replace the entire set of people with this list.",

json\_schema\_extra={

"example": [

{

"uni": "kj2634",

"first\_name": "Kusuma",

"last\_name": "Jaipiam",

"email": "kj2634@columbia.edu",

}

]

},

)

created\_at: datetime = Field(

default\_factory=datetime.utcnow,

description="Creation timestamp (UTC).",

json\_schema\_extra={"example": "2025-01-15T10:20:30Z"},

)

model\_config = {

"json\_schema\_extra": {

"examples": [

{

"title": "Advanced Cloud Computing",

"semester": "Spring 2026",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

},

"people": [

{

"uni": "kj2634",

"first\_name": "Kusuma",

"last\_name": "Jaipiam",

"email": "kj2634@columbia.edu",

}

],

"created\_at": "2025-01-15T10:20:30Z",

}

]

}

}

class CourseworkRead(CourseworkBase):

"""Server representation returned to clients."""

id: UUID = Field(

default\_factory=uuid4,

description="Server-generated Coursework ID.",

json\_schema\_extra={"example": "99999999-9999-4999-8999-999999999999"},

)

updated\_at: datetime = Field(

default\_factory=datetime.utcnow,

description="Last update timestamp (UTC).",

json\_schema\_extra={"example": "2025-01-16T12:00:00Z"},

)

model\_config = {

"json\_schema\_extra": {

"examples": [

{

"id": "99999999-9999-4999-8999-999999999999",

"title": "Cloud Computing",

"semester": "Fall 2025",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

},

"people": [

{

"uni": "kj2634",

"first\_name": "Kusuma",

"last\_name": "Jaipiam",

"email": "kj2634@columbia.edu",

}

],

"updated\_at": "2025-01-16T12:00:00Z",

}

]

}

}

### Resource 2: Assignment

class AssignmentBase(BaseModel):

id: UUID = Field(

default\_factory=uuid4,

description="Persistent Assignment ID (server-generated).",

json\_schema\_extra={"example": "550e8400-e29b-41d4-a716-446655440000"},

)

title: str = Field(

...,

description="Title of the assignment.",

json\_schema\_extra={"example": "Assignment 1"},

)

description: str = Field(

...,

description="Description of the assignment.",

json\_schema\_extra={"example": "Complete the cloud computing project."},

)

due\_date: date = Field(

...,

description="Due date for the assignment.",

json\_schema\_extra={"example": "2025-12-31"},

)

coursework: CourseworkBase = Field(

...,

description="Coursework associated with this assignment.",

json\_schema\_extra={

"example": {

"id": "550e8400-e29b-41d4-a716-446655440000",

"title": "Cloud Computing",

"semester": "Fall 2025",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

}

}

}

)

created\_at: datetime = Field(

default\_factory=datetime.utcnow,

description="Creation timestamp (UTC).",

json\_schema\_extra={"example": "2025-01-15T10:20:30Z"},

)

model\_config = {

"json\_schema\_extra": {

"examples": [

{

"id": "550e8400-e29b-41d4-a716-446655440000",

"title": "Assignment 1",

"description": "Complete the cloud computing project.",

"due\_date": "2025-12-31",

"coursework": {

"id": "550e8400-e29b-41d4-a716-446655440001",

"title": "Cloud Computing",

"semester": "Fall 2025",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

}

},

"created\_at": "2025-01-15T10:20:30Z"

}

]

}

}

class AssignmentCreate(AssignmentBase):

"""Creation payload for an Assignment."""

model\_config = {

"json\_schema\_extra": {

"examples": [

{

"title": "Assignment 1",

"description": "Complete the cloud computing project.",

"due\_date": "2025-12-31",

"coursework": {

"id": "550e8400-e29b-41d4-a716-446655440001",

"title": "Cloud Computing",

"semester": "Fall 2025",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

}

}

}

]

}

}

class AssignmentUpdate(BaseModel):

"""Partial update for an Assignment; supply only fields to change."""

id: Optional[UUID] = Field(

default\_factory=uuid4,

description="Persistent Assignment ID (server-generated).",

json\_schema\_extra={"example": "550e8400-e29b-41d4-a716-446655440000"},

)

title: Optional[str] = Field(

None,

description="Title of the assignment.",

json\_schema\_extra={"example": "Assignment 1"},

)

description: Optional[str] = Field(

None,

description="Description of the assignment.",

json\_schema\_extra={"example": "Complete the cloud computing project."},

)

due\_date: Optional[date] = Field(

None,

description="Due date for the assignment.",

json\_schema\_extra={"example": "2025-12-31"},

)

coursework: Optional[CourseworkBase] = Field(

None,

description="Coursework associated with this assignment.",

json\_schema\_extra={

"example": {

"id": "550e8400-e29b-41d4-a716-446655440000",

"title": "Cloud Computing",

"semester": "Fall 2025",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

}

}

}

)

created\_at: Optional[datetime] = Field(

None,

description="Creation timestamp (UTC).",

json\_schema\_extra={"example": "2025-01-15T10:20:30Z"},

)

model\_config = {

"json\_schema\_extra": {

"examples": [

{

"title": "Assignment 1",

"description": "Complete the cloud computing project.",

"due\_date": "2025-12-31",

"coursework": {

"id": "550e8400-e29b-41d4-a716-446655440001",

"title": "Cloud Computing",

"semester": "Fall 2025",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

}

}

}

]

}

}

class AssignmentRead(AssignmentBase):

"""Server representation returned to clients."""

id: UUID = Field(

default\_factory=uuid4,

description="Server-generated Assignment ID.",

json\_schema\_extra={"example": "99999999-9999-4999-8999-999999999999"},

)

created\_at: datetime = Field(

default\_factory=datetime.utcnow,

description="Creation timestamp (UTC).",

json\_schema\_extra={"example": "2025-01-15T10:20:30Z"},

)

updated\_at: datetime = Field(

default\_factory=datetime.utcnow,

description="Last update timestamp (UTC).",

json\_schema\_extra={"example": "2025-01-16T12:00:00Z"},

)

model\_config = {

"json\_schema\_extra": {

"examples": [

{

"title": "Assignment 1",

"description": "Complete the cloud computing project.",

"due\_date": "2025-12-31",

"coursework": {

"id": "550e8400-e29b-41d4-a716-446655440001",

"title": "Cloud Computing",

"semester": "Fall 2025",

"professor": {

"uni": "df123",

"first\_name": "Donald",

"last\_name": "Ferguson",

"email": "donald.ferguson@cs.columbia.edu",

}

}

}

]

}

}

### main.py Routes

# -----------------------------------------------------------------------------

# Coursework endpoints

# -----------------------------------------------------------------------------

@app.post("/courseworks", response\_model=CourseworkRead, status\_code=201)

def create\_coursework(coursework: CourseworkCreate):

coursework\_read = CourseworkRead(\*\*coursework.model\_dump())

courseworks[coursework\_read.id] = coursework\_read

return coursework\_read

@app.get("/courseworks", response\_model=List[CourseworkRead])

def list\_courseworks(

title: Optional[str] = Query(None, description="Filter by coursework title"),

semester: Optional[str] = Query(None, description="Filter by semester"),

professor\_uni: Optional[str] = Query(None, description="Filter by professor's UNI"),

person\_uni: Optional[str] = Query(None, description="Filter by UNI of any enrolled person"),

):

results = list(courseworks.values())

if title is not None:

results = [c for c in results if c.title == title]

if semester is not None:

results = [c for c in results if c.semester == semester]

if professor\_uni is not None:

results = [c for c in results if c.professor.uni == professor\_uni]

if person\_uni is not None:

results = [c for c in results if any(p.uni == person\_uni for p in c.people)]

return results

@app.get("/courseworks/{coursework\_id}", response\_model=CourseworkRead)

def get\_coursework(coursework\_id: UUID):

if coursework\_id not in courseworks:

raise HTTPException(status\_code=404, detail="Coursework not found")

return courseworks[coursework\_id]

@app.patch("/courseworks/{coursework\_id}", response\_model=CourseworkRead)

def update\_coursework(coursework\_id: UUID, update: CourseworkUpdate):

if coursework\_id not in courseworks:

raise HTTPException(status\_code=404, detail="Coursework not found")

stored = courseworks[coursework\_id].model\_dump()

stored.update(update.model\_dump(exclude\_unset=True))

courseworks[coursework\_id] = CourseworkRead(\*\*stored)

return courseworks[coursework\_id]

@app.delete("/courseworks/{coursework\_id}", status\_code=204)

def delete\_coursework(coursework\_id: UUID):

if coursework\_id not in courseworks:

raise HTTPException(status\_code=404, detail="Coursework not found")

del courseworks[coursework\_id]

return

# ------------------------------------------------------------------------------

# Assignment endpoints

# -----------------------------------------------------------------------------

@app.post("/assignments", response\_model=AssignmentRead, status\_code=201)

def create\_assignment(assignment: AssignmentCreate):

assignment\_read = AssignmentRead(\*\*assignment.model\_dump())

assignments[assignment\_read.id] = assignment\_read

return assignment\_read

@app.get("/assignments", response\_model=List[AssignmentRead])

def list\_assignments(

title: Optional[str] = Query(None, description="Filter by assignment title"),

due\_date: Optional[str] = Query(None, description="Filter by due date (YYYY-MM-DD)"),

coursework\_id: Optional[UUID] = Query(None, description="Filter by associated coursework ID"),

):

results = list(assignments.values())

if title is not None:

results = [a for a in results if a.title == title]

if due\_date is not None:

results = [a for a in results if str(a.due\_date) == due\_date]

if coursework\_id is not None:

results = [a for a in results if a.coursework.id == coursework\_id]

return results

@app.get("/assignments/{assignment\_id}", response\_model=AssignmentRead)

def get\_assignment(assignment\_id: UUID):

if assignment\_id not in assignments:

raise HTTPException(status\_code=404, detail="Assignment not found")

return assignments[assignment\_id]

@app.patch("/assignments/{assignment\_id}", response\_model=AssignmentRead)

def update\_assignment(assignment\_id: UUID, update: AssignmentUpdate):

if assignment\_id not in assignments:

raise HTTPException(status\_code=404, detail="Assignment not found")

stored = assignments[assignment\_id].model\_dump()

stored.update(update.model\_dump(exclude\_unset=True))

assignments[assignment\_id] = AssignmentRead(\*\*stored)

return assignments[assignment\_id]

@app.delete("/assignments/{assignment\_id}", status\_code=204)

def delete\_assignment(assignment\_id: UUID):

if assignment\_id not in assignments:

raise HTTPException(status\_code=404, detail="Assignment not found")

del assignments[assignment\_id]

return

### OpenAPI Document (Partial)

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### Link to Recording of Demo <https://github.com/user-attachments/assets/ab4a693c-184c-49c3-ab24-2b12108baa35>

### Link to GitHub Repository <https://github.com/KusumaJ/SimpleMicroservices>

## 3. Incomplete Work ❌

All finished.

## 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]