



SERVICE ORIENTED ARCHITECTURES

ACIT3855 – FALL 2021



AGENDA

- Quick Review
- Quiz 2
- Sample Application – Services
- Topics:
 - Synchronous Communication
 - Database Per Service
- Lab 3
 - Data Storage Service Walk Thru

REVIEW QUESTIONS

- What are the benefits of Decentralized Data Management? What are the drawbacks?
- What is the Database Per Service pattern?
- What are the differences between a SQLite and MySQL database?
- What is an ORM and why would we use it versus direct SQL statements (what are the benefits/drawbacks)?

QUIZ 2

- Quiz is on the Learning Hub
- Closed book, do your own work
- You have 20 minutes to complete it

COURSE SCHEDULE – TUESDAY SET

Week	Topics	Notes
1	<ul style="list-style-type: none">• Services Based Architecture Overview• RESTful API Review	Lab 1
2	<ul style="list-style-type: none">• Microservices Overview• Edge Service	Lab 2, Quiz 1
3	<ul style="list-style-type: none">• Database Per Service• Storage Service (SQLite)	Lab 3, Quiz 2
4	<ul style="list-style-type: none">• Logging, Debugging and Configuration• Storage Service (MySQL)	Lab 4, Quiz 3
5	<ul style="list-style-type: none">• RESTful API Specification (OpenAPI)• Processing Service	Lab 5, Quiz 4, Assignment 1 Due
6	<ul style="list-style-type: none">• Synchronous vs Asynchronous Communication• Message Broker Setup	Lab 6A, Quiz 5
7	<ul style="list-style-type: none">• Messaging and Event Sourcing	Lab 6B, No Quiz
8	<ul style="list-style-type: none">• Midterm Review• Containerization of Services (Docker/Docker Compose)	Lab 7, Quiz 6, Assignment 2 Due (Midterm Review)
9	<ul style="list-style-type: none">• Dashboard UI and CORS	Lab 8, Quiz 7
10	<ul style="list-style-type: none">• Issues and Technical Debt	Lab 9, No Quiz
11	<ul style="list-style-type: none">• Deployment – Configuration and Logging	Lab 10, Quiz 8
12	<ul style="list-style-type: none">• Deployment – Reverse Proxy and Load Balancing• Deployment – Scaling (RESTful APIs)	Lab 11, Quiz 9
13	<ul style="list-style-type: none">• Final Exam Review• Assignment 3 In Class	Assignment 3 Due
14	<ul style="list-style-type: none">• Final Exam	

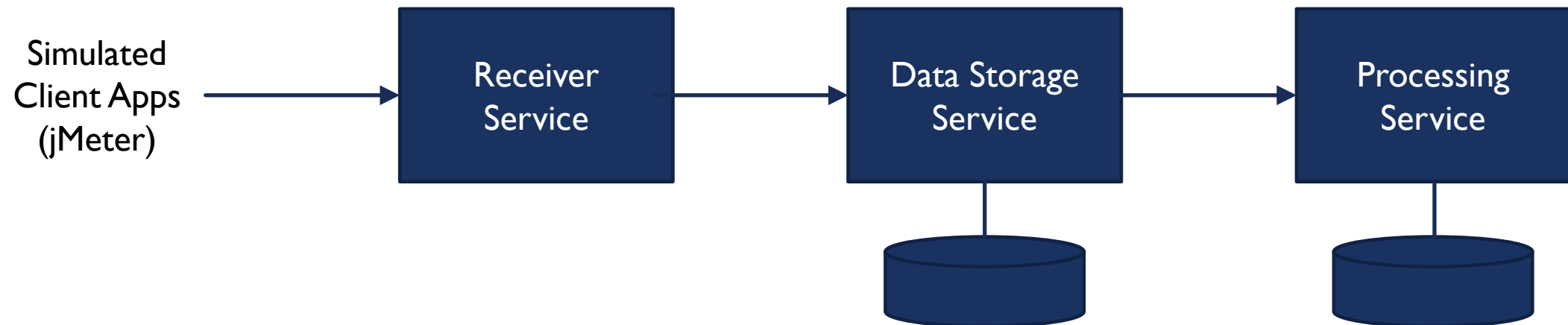
COURSE SCHEDULE – THURSDAY SETS

Week	Topics	Notes
1	<ul style="list-style-type: none"> Services Based Architecture Overview RESTful API Review 	Lab 1
2	<ul style="list-style-type: none"> Microservices Overview Edge Service 	Lab 2, Quiz 1
3	<ul style="list-style-type: none"> Database Per Service Storage Service (SQLite) 	Lab 3, Quiz 2
4	No Class - Holiday	
5	<ul style="list-style-type: none"> Logging, Debugging and Configuration Storage Service (MySQL) 	Lab 4, Quiz 3, Assignment 1 Due
6	<ul style="list-style-type: none"> RESTful API Specification (OpenAPI) Processing Service 	Lab 5, Quiz 4
7	<ul style="list-style-type: none"> Synchronous vs Asynchronous Communication Message Broker Setup Messaging and Event Sourcing 	Lab 6, Quiz 5
8	<ul style="list-style-type: none"> Midterm Review Containerization of Services (Docker/Docker Compose) 	Lab 7, Quiz 6, Assignment 2 Due (Midterm Review)
9	<ul style="list-style-type: none"> Dashboard UI and CORS 	Lab 8, Quiz 7
10	<ul style="list-style-type: none"> No Class – Holiday. Issues and Technical Debt (Take Home) 	Lab 9, No Quiz
11	<ul style="list-style-type: none"> Deployment – Configuration and Logging 	Lab 10, Quiz 8
12	<ul style="list-style-type: none"> Deployment – Reverse Proxy and Load Balancing Deployment – Scaling (RESTful APIs) 	Lab 11, Quiz 9
13	<ul style="list-style-type: none"> Final Exam Review Assignment 3 In Class 	Assignment 3 Due
14	<ul style="list-style-type: none"> Final Exam 	

OUR SAMPLE APPLICATION

Our sample application will have three initial services:

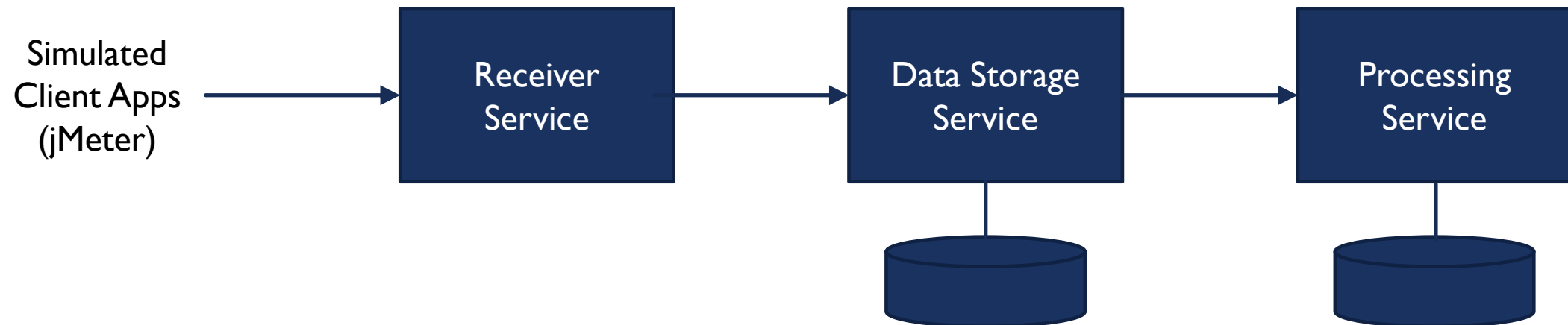
- Receiver Service (Lab 2)
- Storage Service (Lab 3)
- Processing Service (Lab 5)



OUR SAMPLE APPLICATION

Today you will be building the Data Storage Service and integrating it with the Receiver Service. The two concepts you will be using are:

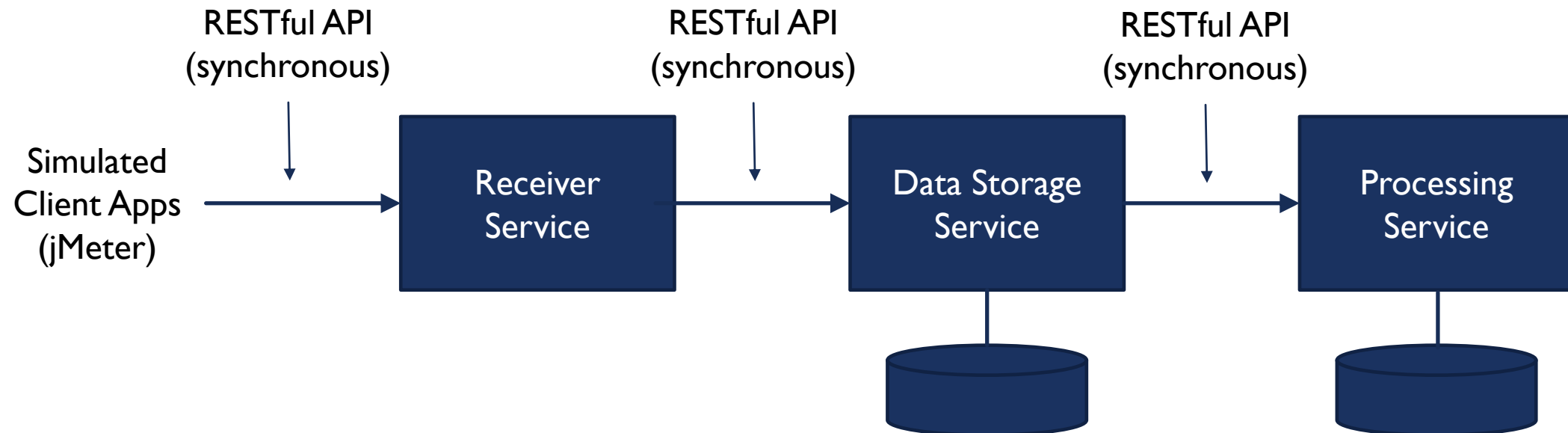
- Synchronous Communication
- Database Per Service



SYNCHRONOUS COMMUNICATION

Synchronous Communication – When a client makes a request and has to wait until the request fulfilled before receiving the response. The client thread generally is blocked until the server fulfills the request. The server thread has to complete all the activities required of the request before returning the response.

An HTTP call to a RESTful service is considered synchronous communication.



SYNCHRONOUS VS ASYNCHRONOUS COMMUNICATION

Asynchronous Communication – When a client makes a request and DOES NOT have to wait until the request completely fulfilled before receiving the response.

What is the advantage of
Synchronous
Communication?
What about a disadvantage?

What is the advantage of
Asynchronous
Communication?
What about a disadvantage?

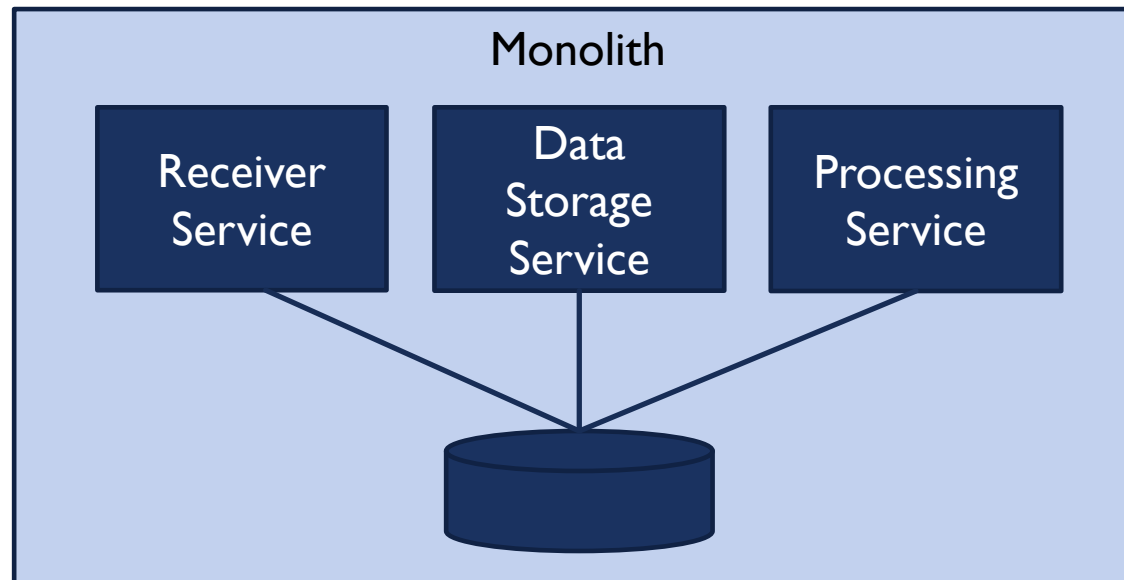
Why might Asynchronous
Communication be
important to a
Microservices Architecture?

We implement Asynchronous Communication in a later lab to help us address scalability of our Event Handling.

DATABASE PER SERVICE

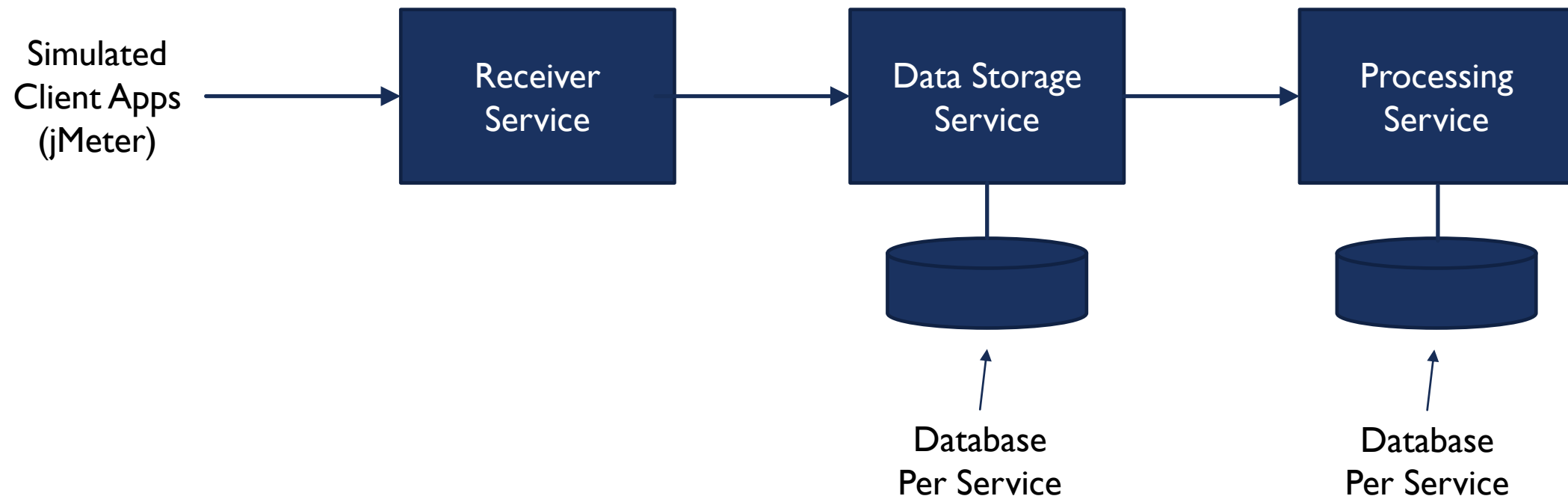
Traditional monolithic applications have one large database shared by multiple components in the application. This allowed for strong data integrity across tables because constraints can be added (i.e., foreign key references).

You could even use the database to share data between modules.



DATABASE PER SERVICE

Microservice based applications typically have a separate database per service (that actually requires persistent data storage). This provides looser coupling of services and different technology choices for data persistence for each service.



MONOLITHIC VS DATABASE PER SERVICE

What is the advantage of a single monolithic database used by all services?
What about a disadvantage?

What is the advantage separate databases per service?
What about a disadvantage?

Separate databases does add complexity to a Microservices Architecture. These include:

- Eventual consistency – Your system may have to accept that stored data will not always be consistent at a given time.
- Multiple Step Rollback – If you need to be consistent and a failure occurs, you may have to “manually” do rollback across multiple services.

REVIEW – ORM (SQLALCHEMY)

Example: We can map the person table to a Person class using a SQLAlchemy

person

id	first_name	last_name
----	------------	-----------

```
class Person(Base):
    __tablename__ = 'person'

    id = Column(Integer, primary_key = True)
    first_name = Column(String(250),
                        nullable=False)
    last_name = Column(String(250),
                      nullable=False)

    def full_name(self):
        """ Joins the first and last name to a
            full name """
        return self.first_name + ' ' + self.last_name
```

Columns are
Instance Variables

Table Mapping

Column Mapping

Custom Methods

Allows us to map
database tables to
Python classes

It also allows us to be
somewhat database
independent – so we will
be able to switch with
minimal code changes.

REVIEW – REQUESTS

```
# Example: POST /points
import requests

# POST request
point_data = { "x": 22, "y": 25 }
headers = { "content-type": "application/json" }
response = requests.post("http://localhost:8080/points",
                          json=point_data, headers=headers)

if response.status_code == 201: # May be 201 or 400 in our case
    # JSON response - has a built-in JSON decoder
    print(response.json())
```

requests – Python package to
call endpoints of RESTful APIs

TODAY'S TOOLS

RESTful API Specification: SwaggerHub and OpenAPI

- Define a RESTful API in a yaml format

RESTful API Implementation: Python connexion

- Built on top of Flask but allows integration with an OpenAPI specification

Database: SQLAlchemy and Sqlite, DB Browser for SQLite

- Both same as ACIT 2515
- We will be “upgrading” to MySQL for Lab 4

RESTful API Testing: PostMan and Apache jMeter

- Postman – same as ACIT 2515
- Apache jMeter – for load testing

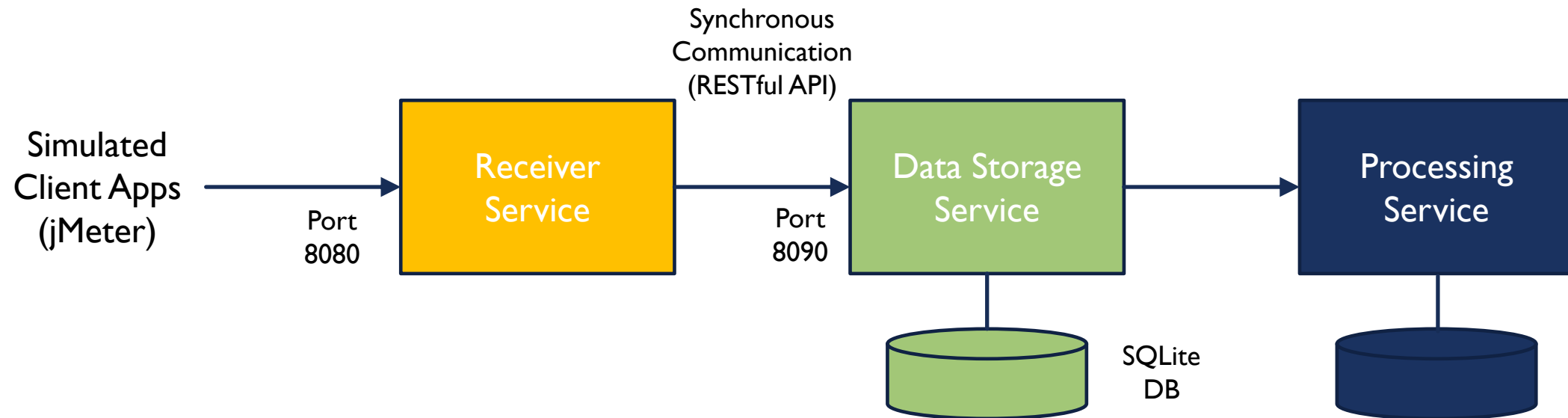
We will re-use the OpenAPI spec from the Edge Service.

We will use the Edge Service Implementation as a Starting Point.

Using an ORM will make it easy to switch to MySQL next lab (i.e., minimal code changes).

You'll add some randomness to your jMeter tests.

TODAY'S LAB



3. Add some randomness to your events and test to see what concurrent load you can support

2. Update the Receiver Service to pass the events through to the Storage Service

1. Implement the Storage Service with a SQLite DB

TODAY'S LAB – SAMPLE CODE

I've posted sample code for today's lab that you can use as a reference for the SQLAlchemy declaratives (i.e., the classes that are mapped to the database).

We'll go over the sample Storage application now and updates to your jMeter Test Case.

TODAY'S LAB

- Queue up for Lab 2 demos
- Start on Lab 3
 - You will create a new Storage service using a SQLite database
 - You will update the Receiver service to integrate with the new Storage service
 - You will update your jMeter Test Plan to have variability in the request messages
 - Demo is due by the end of the next class