SERVICE ORIENTED ARCHITECTURES

ACIT3855 – WINTER 2024

COURSE OVERVIEW AND LEARNING OUTCOMES

This course focuses on the design of Enterprise Application Software using Service Based Architecture. Students use Service Based Architecture analysis and design techniques to identify business and systems services. The course will consider various techniques for implementing services including Web Services and Representational State Transfer. The course will review current and emerging standards that are relevant to Service Oriented Architecture, SaaS, and Micro-services.

Upon successful completion of this course, the student will be able to:

- Explain what Service Based Architectures are and their impact on modern Enterprise Application Software.
- Apply the concepts and principles of Service Based Architectures to software development projects.
- Compare and contrast Service Based Architecture analysis and design with object-oriented (OO) analysis and design.
- Design an interoperable Service Based Architecture application.
- Implement services using REST interfaces.

EVALUATION CRITERIA

Quizzes	20%	There will be an in-class quiz at the beginning of each class (except the first and last), including a midterm review quiz. Based on pre-reading materials posted to the Learning Hub.
Project Labs	35%	A microservices based application will be incrementally built and deployed through the project labs. Labs are done individually, typically require a demo and are due by the end of the next class.
Assignments	10%	There are two assignments – one written and one coding.
Final Exam	35%	Concepts and Coding

Quizzes will **only be done in-class only.** There are no late or make-up quizzes, but the lowest quiz of the term will be dropped.

There is a 20% penalty per week for late lab submissions, up to a maximum of 2 weeks.

Passing Criteria:

- Students must complete a minimum of 80% of the Project Labs.
- Students must achieve a minimum of 50% on the combination of the Quizzes and Final Exam (i.e., 27.5/55).
- Students must achieve an overall grade of at least 50% in the course.

COURSE SCHEDULE

Week	Topics	Notes
1	• Services Based Architecture Overview • RESTful APIs Review	Lab 1
2	• Microservices Overview • Edge Service	Lab 2, Quiz 1
3	• Database Per Service • Storage Service (SQLite)	Lab 3, Quiz 2
4	• Logging, Debugging and Configuration • Storage Service (MySQL)	Lab 4, Quiz 3
5	• RESTful API Specification (OpenAPI) • Processing Service	Lab 5, Quiz 4
6	• Synchronous vs Asynchronous Communication • Message Broker Setup, Messaging and Event Sourcing	Lab 6, Quiz 5, Assignment 1 Due
7	• Deployment - Containerization of Services *Note: At home lab for Monday Set*	Lab 7, Quiz 6 (Sets A and B)
8	• Midterm Week	Midterm Review Quiz
9	• Dashboard UI and CORS	Lab 8, Quiz 6 (Set C), Quiz 7
10	• Spring Break	No Class
11	• Issues and Technical Debt	Lab 9, Quiz 8
12	• Deployment – Centralized Configuration and Logging	Lab 10, Quiz 9
13	• Deployment – Load Balancing and Scaling *Note: At home lab for Monday Set*	Lab 11, Quiz 10 (Sets A and B)
14	• Final Exam Preview	Quiz 10 (Set C), Assignment 2 Due
15	• Final Exam	

You will need to use a Cloud VM (AWS or Azure or Equvalent) starting Week 6 as we will start using Docker for deploying 3rd party services and your own services.

INSTRUCTORS

- Set A – Thursday Afternoon – Mike Mulder
- Set B – Thursday Afternoon – Tim Guicherd
- Set C – Monday Morning – Mike Mulder

Please contact the instructor for your set for questions and to demo your labs. Each of us will monitor the Discord channels for our assigned set(s).

Our office hours is posted as a news item on the Learning Hub.

CLASS STRUCTURE

Pre-reading Before Class – On the week's tool/concept

In Classroom (30-60 minutes):

- Quick Review – On the reading
- Quiz – On the Learning Hub
- Discussion/demo on the week's lab and/or concept

Lab (remainder of class time)

- Demos of previous week's completed lab. This may also be done prior to class during office hours.
- Work on new lab.

DISCORD AND OFFICE HOURS

The invite for the Discord Server is posted in the course site on the Learning Hub. We will be using this for the following:

- Lab questions/discussions and screenshares outside of classtime
- Notifications (in addition to news items on the Learning Hub)
- Office Hours

CHECK-IN QUIZ

- Complete Quiz 0 on the Learning Hub now. It will take less than 5 minutes.
- It will not count for marks, but I will use it to check attendance.

REMEMBER THIS FROM ACIT 2515?

EntityManager
- _entities : Entity[]

+ add_entity(Entity entity) : int
+ get_entity(int id) : Entity
+ update_entity(Entity entity)
+ delete_entity(int id)

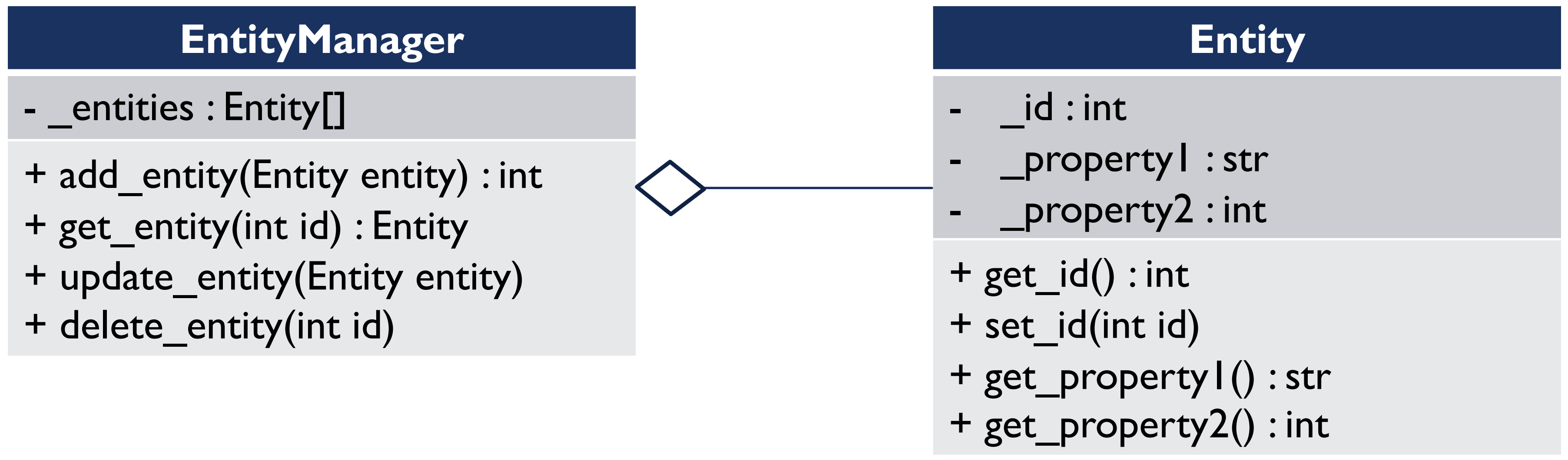
What do the methods in this class represent?

The Public Interface.
It can also referred to as an **API** – the Application Program Interface

Do you remember the Single Responsibility Principle (the S in SOLID)?

A class should only be responsible for a single part of the functionality of a software application. It can also be stated that the methods (the API) have <u>high-cohesion</u> – they are all related to the same task (i.e., they belong together)

REMEMBER THIS FROM ACIT 2515?



Different classes should be loosely coupled. They should only interact with each other through well-defined Public Interfaces (i.e., the API).

REMEMBER THIS FROM ACIT 2515?

EntityManager
- _entities : Entity[]

+ add_entity(Entity entity) : int
+ get_entity(int id) : Entity
+ update_entity(Entity entity)
+ delete_entity(int id)

POST /entities

GET /entities/<id>

PUT /entities/<id>

DELETE /entities/<id>

What was this?

The RESTful API we created on top of our EntityManager class

It provides an API to other software applications over a network using HTTP as the protocol.

SERVICE

Service

A service is a discrete unit of functionality that can be accessed remotely and can be acted upon and updated independently. *This is very much like our definition of a class but with RESTful API!*

A service typically has four properties:

- It logically represents a business activity with a specific outcome (i.e., single responsibility)
- It is self-contained
- It is a "black-box" for its consumers
- It may use other underlying services

SERVICE ORIENTED ARCHITECTURE

Service Oriented Architecture (SOA)

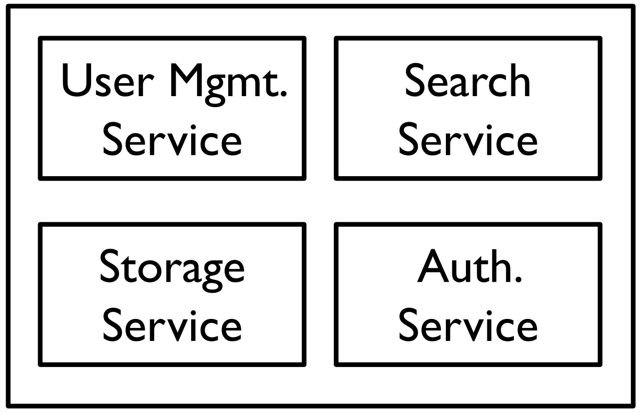
Style of software design where the software components are Services that interact through a communication protocol over a network.

Traditionally systems that use a SOA are decomposed into multiple software applications which are then further decomposed into services. Often these software applications are called **monoliths** because they consist of a single software deployment that is very large and complex.

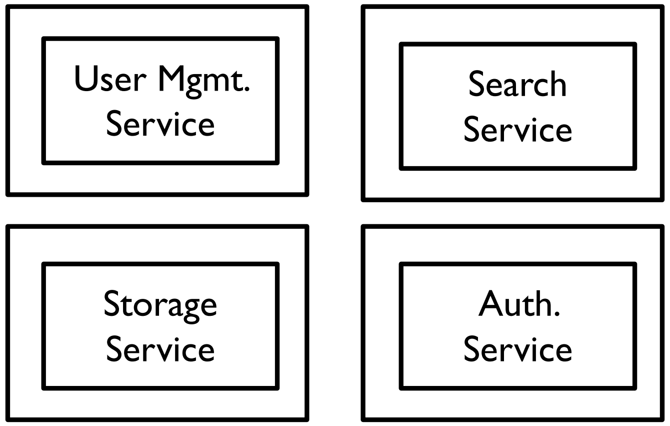
MICROSERVICES

Microservices

A form of SOA where the services are more fine-grained and independently deployable.



Monolith – Single Deployment



Microservices – Multiple Small Deployments

How Small is "Micro"?

MICROSERVICES

Advantages

- Breaks down the system into smaller pieces that can be worked on independently
- Enables ownership of individual services by specific developers or teams
- Allows for better scalability of the system – each service is deployed independently and can scale independently
- Allow for independent technology choices – each service could use a different programming language and related technologies

Disadvantages

- Allows for more diversity which can be harder to manage and make knowledge transfer more difficult
- It is much more difficult to deploy and operate multiple applications for a single system

RESTFUL APIS

Although microservices allow for diversity in technologies internally, generally they provide RESTful APIs to their clients externally.

What do you remember about RESTful APIs from ACIT 2515 (and other courses)?

RESTFUL APIS

- URI: Defines the endpoint
  - Example: POST http://api.example.com/users
- HTTP Resource (Noun): The data object your are operating on (i.e., user in the example above)
- HTTP Method (Verb): The operation you are performing on the resource (i.e., POST in the example above)
  - POST - create
  - PUT - update
  - GET – retrieve (no side effects)
  - DELETE – remove
- Request and Response Messages – Typically defined in a JSON format

RESTFUL API SPECIFICATION

- RESTful APIs are typically specified by the code. However, that means someone needs to read the code to understand how to use the API.
- Therefore, we want to document our API. However, if the documentation is separate from the code, it can get out of synch quickly.
- So strategies for specifying RESTful APIs that could be used to generate both documentation and code were developed. Two competing specifications were popular:
  - Swagger
  - OpenAPI
- Swagger and OpenAPI merged together a couple of years ago – now OpenAPI defines the specification for describing RESTful APIs and Swagger provides the tooling to generate documentation/code/etc. from the specification.

OPENAPI 3.0 AND SWAGGERHUB

- SwaggerHub is an online service that allows you to specify your RESTful API using OpenAPI (version 2 or 3).
  - Provides validation of your RESTful API specification and automatically generates corresponding documentation
  - https://swagger.io/tools/swaggerhub/
- OpenAPI allows you to specify the resources and supported HTTP methods for each resource in your API
  - https://app.swaggerhub.com/help/tutorials/openapi-3-tutorial
- We will be using SwaggerHub and OpenAPI 3.0 in this class to specify the RESTful APIs for our services.
- You will try these out in today's lab (Week 1) and we will go slightly deeper in Week 5.

COURSE PROJECT

Through the labs you are going to build a sample application using a Microservices Architecture.

You can define your application but it must have the following:

- Receives events from other systems. These events are received through the day but have times when the number of events can be very large such that the system may need to scale to handle the load.
- The events are stored for later processing and for users to view and possible manipulate them.
- Some analysis and/processing is performed on the stored events.

COURSE PROJECT – EXAMPLE 1 (THIS IS MINE – DON'T USE)

This is an IoT application for a system that receives and analyzes readings from medical devices at a patient's home (i.e., blood pressure, pulse, etc.).

The events are the blood pressure and pulse readings (i.e., heart rate). They are received throughout the day but the load increases dramatically in the evenings before bed and in the morning as the patients are taking their readings through their blood pressure and pulse oximeter devices. We expect to have a peak load of 5000 readings per second.

The readings are stored to allow the following:

- Caregivers to view trends in the readings for a patient over time.
- Analysis of the patient readings to do short-term anomaly detection and notification
- Aggregation of data across multiple patients to look for trends in the data

Users of the system are patients and caregivers.

COURSE PROJECT – EXAMPLE 2 (SAMPLE ONLY, DON'T USE)

This is a application for a new world-wide ride hailing service that allows for immediate requests for rides and scheduling of pooled rides for daily commutes.

The events are ride requests and schedule requests for a weekly pooled ride to work. The ride request events are expected to be very high on Friday and Saturday evenings with as many as 10000 requests per second while the schedule requests are expected to be very high on Sunday evenings as workers plan their commute for the week with as many as 5000 requests per second.

The requests are stored to allow the following:

- Automatic assignment of immediate ride requests to available drivers
- Manual viewing and planning of pooled rides by available drivers
- Analysis of the data to produce reporting of various metrics such as requests fulfilled vs. requests unfilled to help optimize the service

Users of the system are customers (i.e., those wanting a ride), drivers and management (i.e., reports)

COURSE PROJECT – EXAMPLE 3 (SAMPLE ONLY, DON'T USE)

This is a application for a business to business service for ordering widgets. Your business is selling a new widget that you expect frequent orders from other small and medium sized businesses.

The events two types of orders for each of two types of widgets. For both widget types, you expect a peak of orders on Monday mornings as business prepare for the week. This could be as many as 3000 orders per second for Widget 1 and 2000 orders per second for Widget 2.

The requests are stored to allow the following:

- Automatic fulfillment of orders for Widget 1
- Manual fulfillment of orders for Widget 2
- Analysis of the data to enable proactive alerting based on the available inventory of widgets to help the logistics and manufacturing teams better optimize workflows and staffing.

Users of the system are business customers (i.e., orders), fulfillment staff (i.e., Widget 2) and managers (i.e., optimization)

OPENAPI / SWAGGERHUB DEMO

- Let’s look at an OpenAPI specification for the events for my system for Example 1 that includes endpoints for reporting:
  - Blood Pressure Readings
  - Pulse (i.e., Heart Rate) Readings

LAB TOOLS – DEVELOPMENT OF SERVICES

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

RESTful API Testing: PostMan and Apache jMeter

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

Today you will be defining the RESTful API your first service in OpenAPI.

Next class you will be building and testing this service.

In subsequent classes you will be building more services following a similar pattern.

TODAY’S LAB (LAB 1)

The lab is to be submitted individually. Today you will:

- Define your project
- Design the RESTful API for the service that receives the events
    - Using OpenAPI and SwaggerHub
    - Must have two POST endpoints and at least 4 unique properties for each

Lab 1 is due midnight prior to next class (submission, no demo). It is an input for Lab 2, so try to get it done early so you can get feedback.

We will now transition to the lab. I will be available to answer questions and/or do an initial review of your project.