IT SOLUTION ARCHITECTURE

TheRuntimeTerrors

Final Presentation

AMOS LAM | GABRIEL KOH | NG RUI QIN | LIN HAN HUI | LIU ZUO LIN | TRUONG HAI BANG

Executive Summary

What we will sharing about today

Context	Architectural Overview	Dev Strategy	User Journey & Events
Background	Solution View	Dev view & Strategy	Login & Homepage
Business need	S3 + CloudFront	CI/CD	Main functionalities
Stakeholder	Microservices	Feature Branching	System Breakdown
Use Cases & Features	AWS Fargate	Merge Reviews	Security Testing
	Autoscaling		Maintenance

Key Architectural Decisions: Maintainability, Availability, Security, Performance, Portability

Executive Summary

What we will sharing about today

Context	Architectural Overview	Dev Strategy	User Journey & Events
Background	Solution View	Dev view & Strategy	Login & Homepage
Business need	S3 + CloudFront	CI/CD	Main functionalities
Stakeholder	Microservices	Feature Branching	System Breakdown
Use Cases & Features	AWS Fargate	Merge Reviews	Security Testing
	Autoscaling		Maintenance

Key Architectural Decisions: Maintainability, Availability, Security, Performance, Portability

Software Provider for Education Sector: Student Management System SMS

A **Student Management System** (SMS) **built by vendors** for universities.

Universities generally follow a standard, **GPA-based scoring model**, with each **calendar year divided into 2 terms**.

This application concerns **three main actors**: students, teachers and administrators.

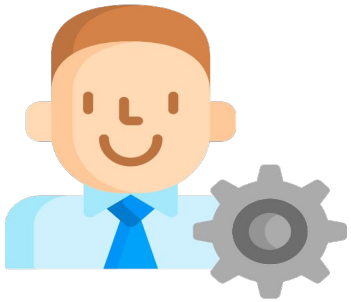
In scope: **Student System**

Project Context
Stakeholders

Software Provider for Education Sector: Student Management System SMS

Student

Teacher



Admin

Software Provider for Education Sector: Student Management System SMS

<u>Primary User: Teacher</u>

Disseminate announcements

Release results

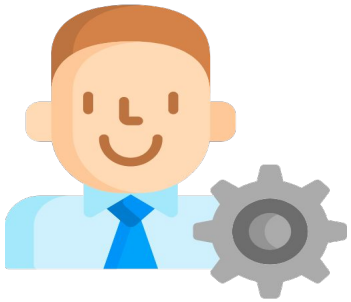
Teacher

Software Provider for Education Sector: Student Management System SMS

<u>Secondary User: Admin</u>

Provide admin support

Configure semesters, course, class, etc.



Admin

Project Context
Stakeholders

Software Provider for Education Sector: Student Management System SMS

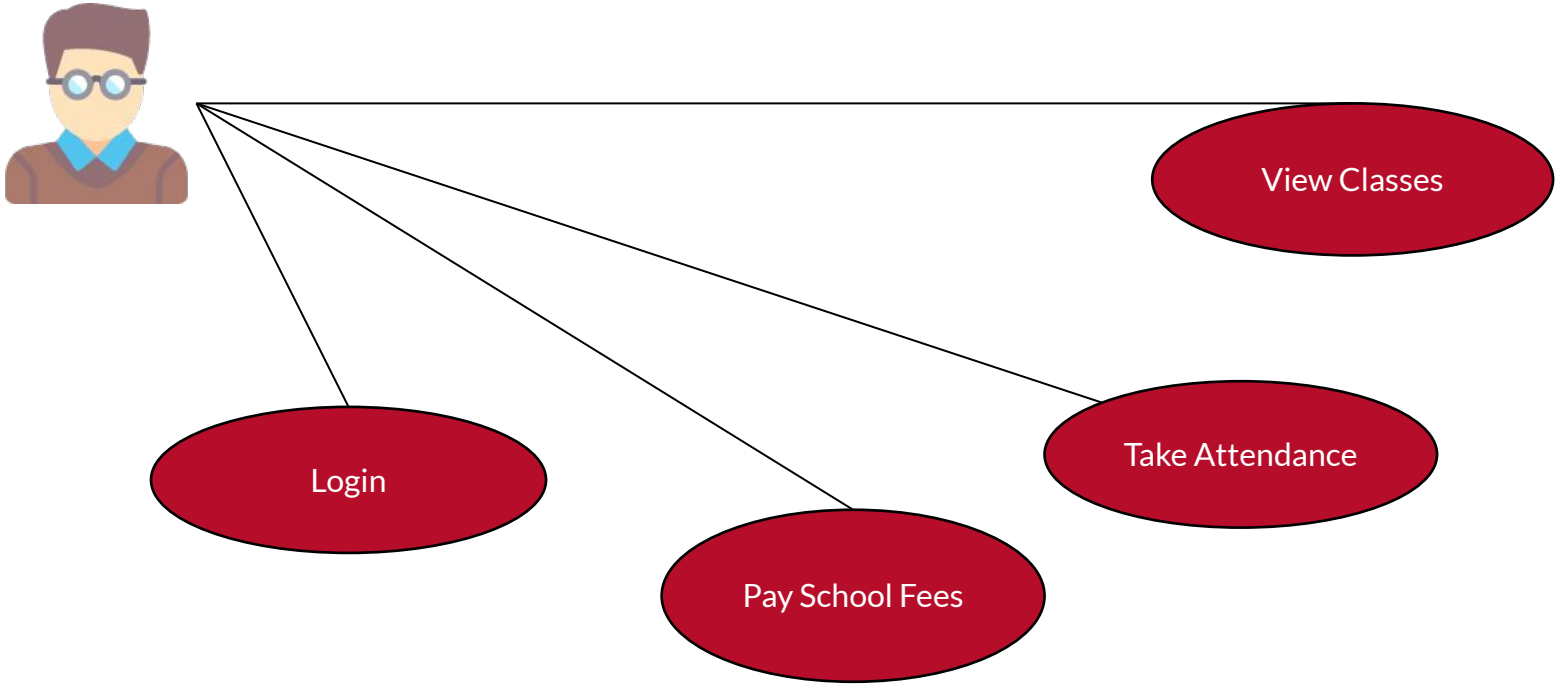
Student

<u>Primary User: Student</u>

Manage day to day activities

In Scope!

A Student using SMS



Modules in scope: **Module, Attendance, Payment, Student, Authentication**

Project Context
Business Needs

Due to the application's nature and context, students need to be able **to access SMS on a 24/7 basis.**

Student

As SMS contains **private and confidential information** such as students' payment details, users **must also be authenticated** in order to use its services.

Its services need to be **fast and well-performing**. This holds especially true during **peak periods** such as during periods where **payment deadline is approaching**.

Executive Summary

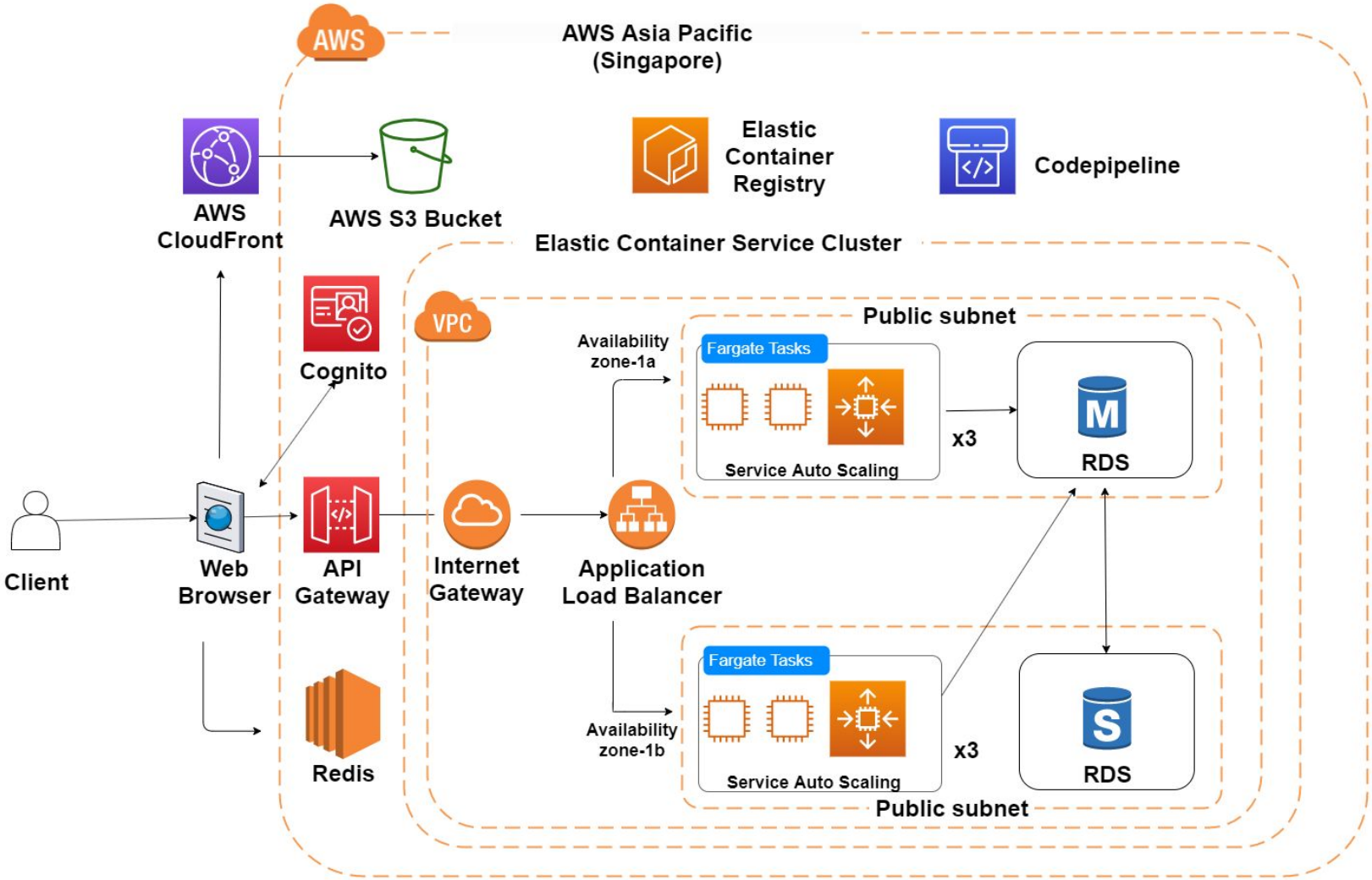
What we will sharing about today

Context	Architectural Overview	Dev Strategy	User Journey & Events
Background	Solution View	Dev view & Strategy	Login & Homepage
Business need	S3 + CloudFront	CI/CD	Main functionalities
Stakeholder	Microservices	Feature Branching	System Breakdown
Use Cases & Features	AWS Fargate	Merge Reviews	Security Testing
	Autoscaling		Maintenance

Key Architectural Decisions: Maintainability, Availability, Security, Performance, Portability

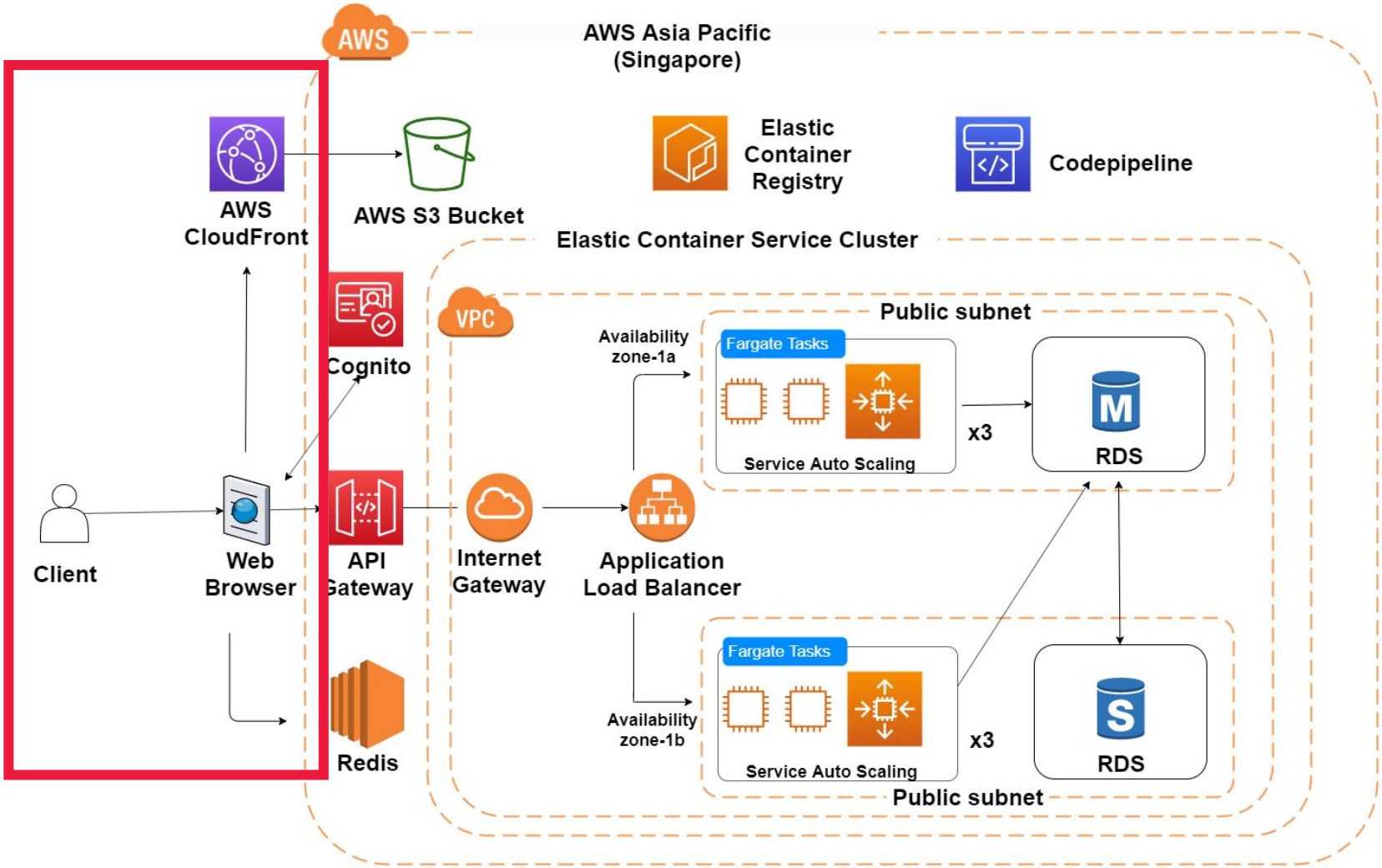
Architecture

Solution Overview – Architectural Diagram



Architecture

Solution Overview – Architectural Diagram



S3 + CloudFront CDN

Key Architectural Decisions

#1

S3 Bucket + CloudFront CDN

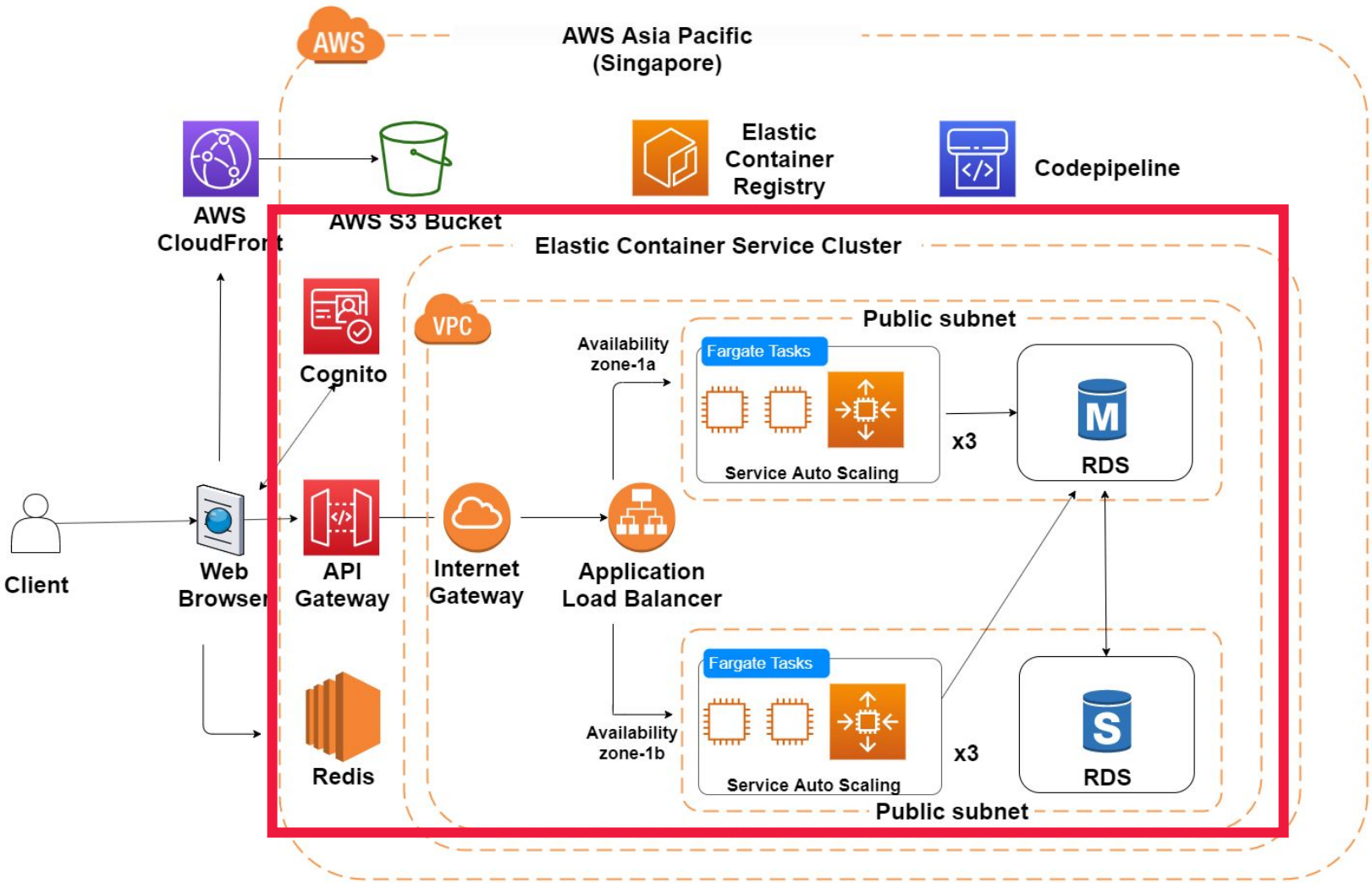
ISO25010:
Performance Efficiency, Security, Availability

Decision & Justifications

1	**S3 + CloudFront CDN Service offers caching, SSL encryption, DDOS mitigation**
2	**AWS guarantees 99.999999999% durability for S3 buckets**
3	**CDN Services provides caching and edge location speed optimisations**

Architecture

Solution Overview – Architectural Diagram



MICROSERVICE ARCHITECTURE

ISO25010:
Maintainability

Decision & Justifications

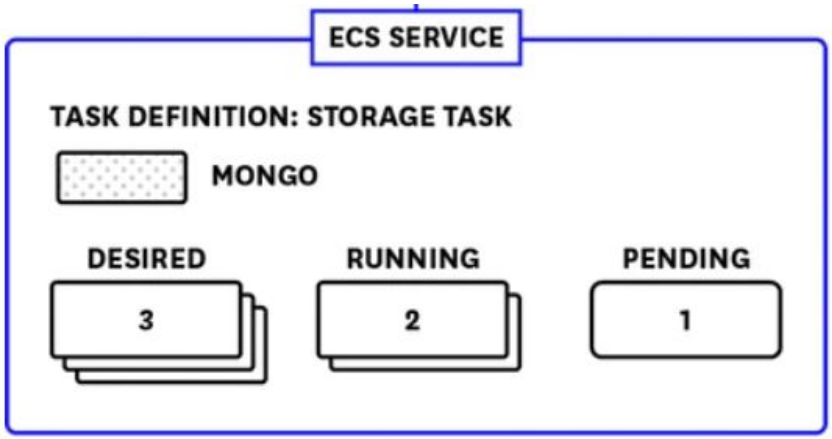
1	**Deploy three microservices (Student, Module and Payment)**
2	**Boosts maintainability (Modularity, Reusability, Testability)**
3	**Practical for SMS, easier to scale when there is increasing traffic**

AWS ECS (Fargate)

ISO25010: Availability
Others: Ease of deployment

Decision & Justifications

1. rvices
2. es clusters
3. Keeps a specified number of tasks up



Auto scaling with AWS Fargate

ISO25010:
Availability

Decision & Justifications

1	**Automatically adjust number of microservices deployed based on number of incoming requests**
2	**Easily manage availability issues, especially when network traffic is high**
3	**Cost management is ensured, dynamically increase and decrease resources that are deployed**

Integration Points

Solution Overview – Integration Points

Integration Endpoints

Source System	Destination System	Protocol	Format	Communication Mode
Student Management System	API gateway	HTTPS	JSON	Synchronous
API gateway	Application Load Balancer	HTTP	JSON	Synchronous
Application Load Balancer	Payment/Module/Student Microservice	HTTP	JSON	Synchronous
Payment/Module/Student Microservice	RDS database	MYSQL	SQL	Synchronous
Student Management System	Redis Cache	HTTPS	JSON	Synchronous

Executive Summary

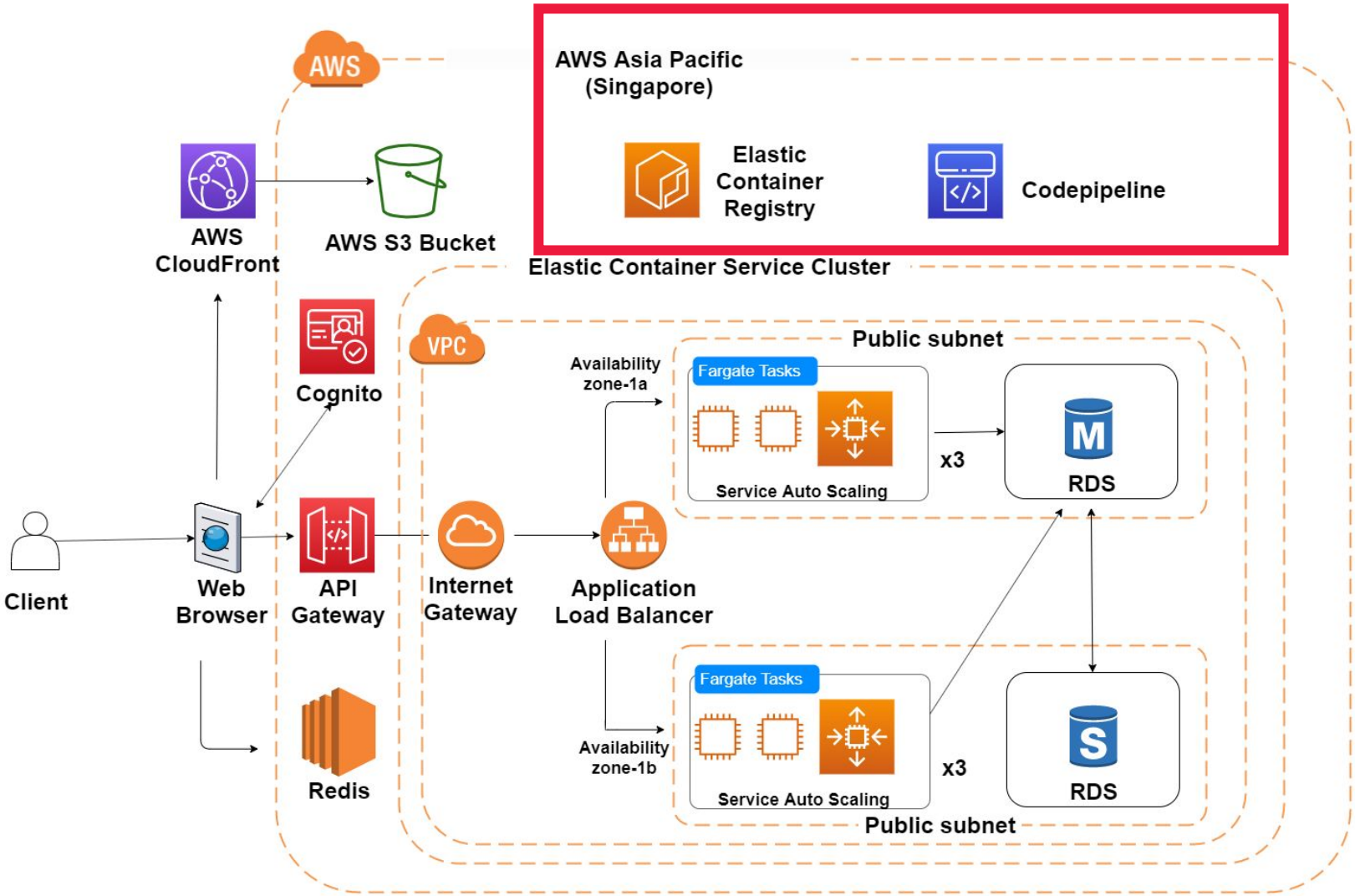
What we will sharing about today

Context	Architectural Overview	Dev Strategy	User Journey & Events
Background	Solution View	Dev view & Strategy	Login & Homepage
Business need	S3 + CloudFront	CI/CD	Main functionalities
Stakeholder	Microservices	Feature Branching	System Breakdown
Use Cases & Features	AWS Fargate	Merge Reviews	Security Testing
	Autoscaling		Maintenance

Key Architectural Decisions: Maintainability, Availability, Security, Performance, Portability

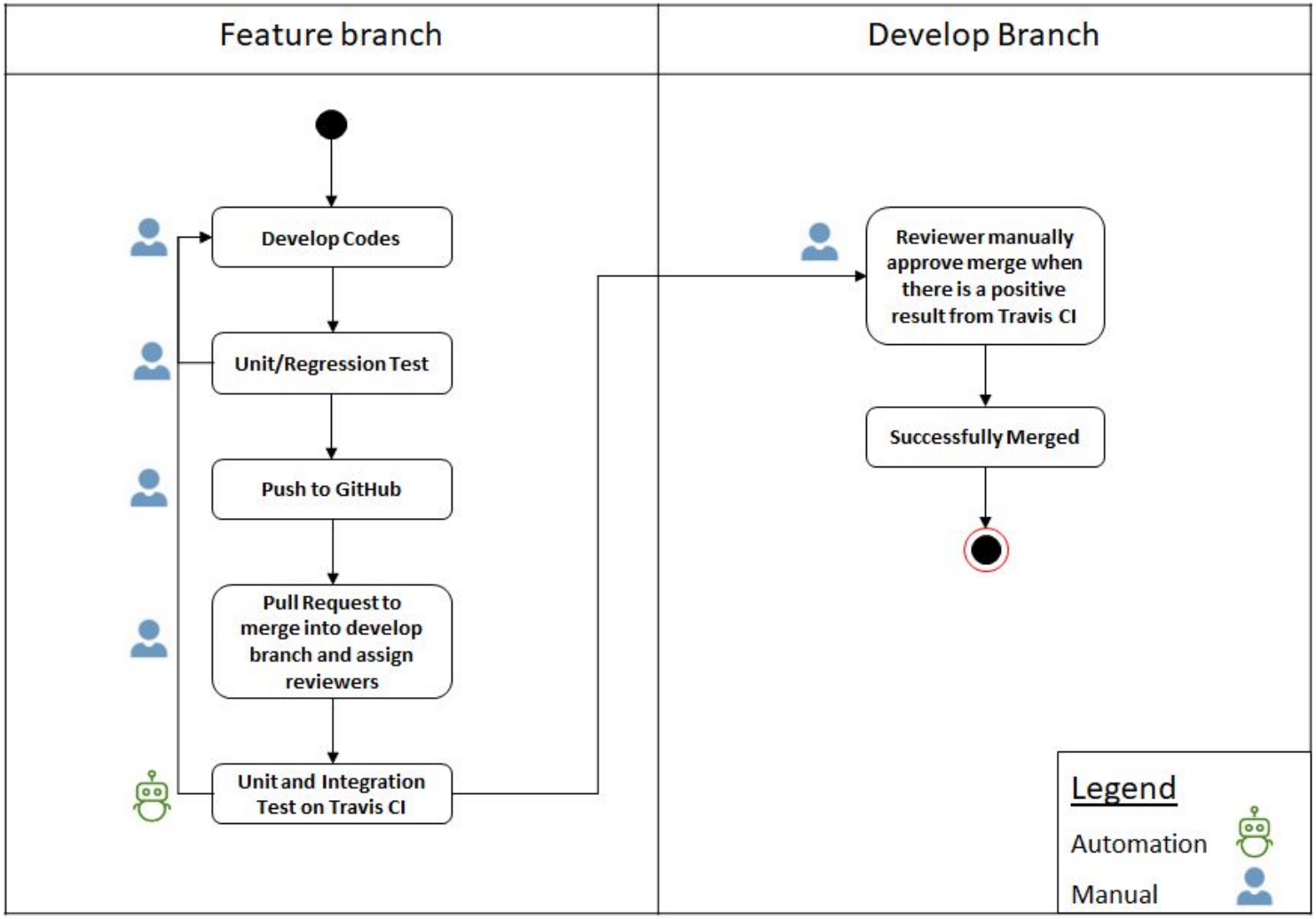
Architecture

Solution Overview – Architectural Diagram



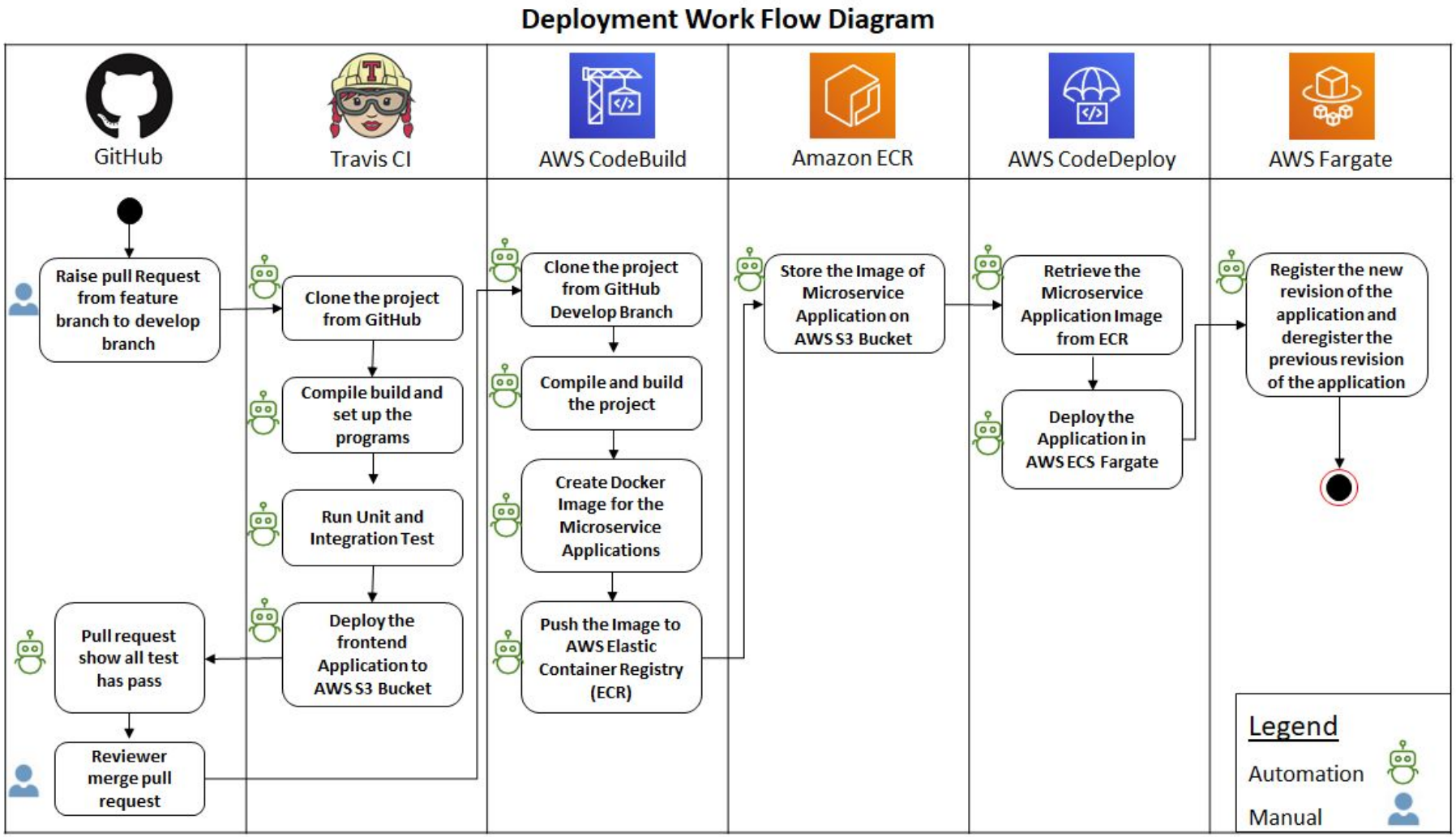
Development View

Our Development Strategy – GitHub Workflow



Development View

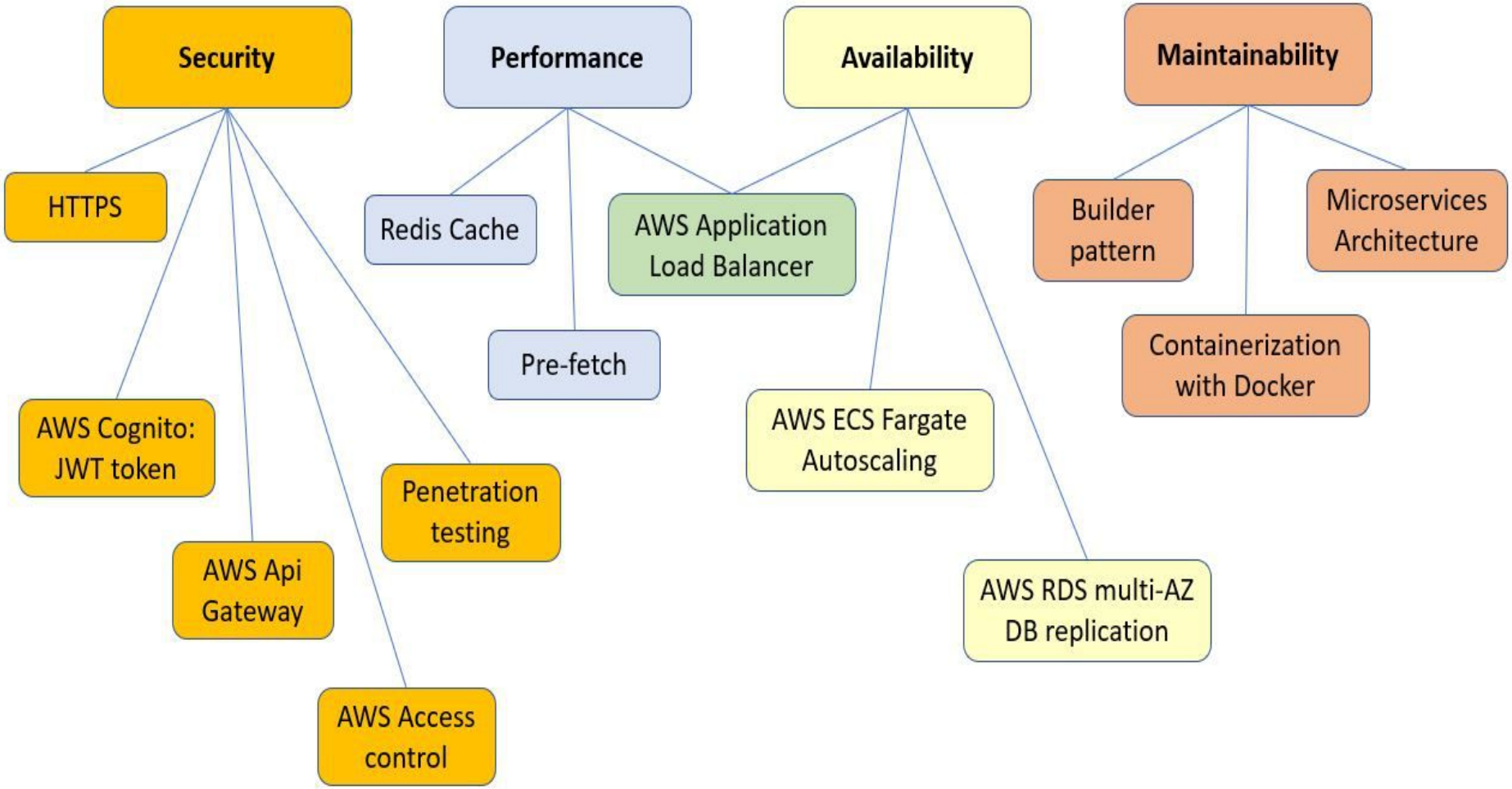
Our Development Strategy – Deployment Workflow with CI/CD



DEV STRATEGY DEMO

Summary of Architectural Principles

The SPAM Tree



Executive Summary

What we will sharing about today

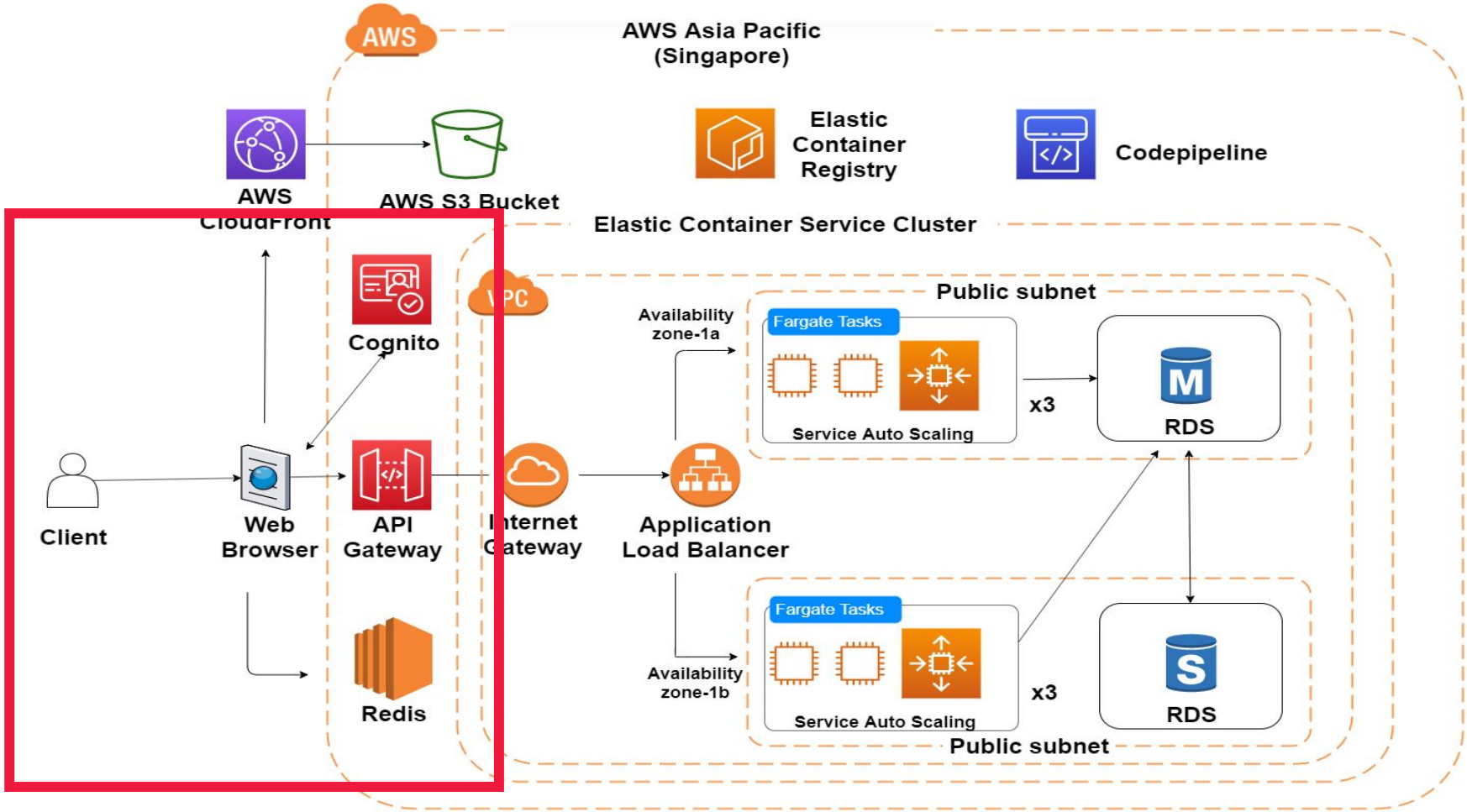
Context	Architectural Overview	Dev Strategy	User Journey & Events
Background	Solution View	Dev view & Strategy	Login & Homepage
Business need	S3 + CloudFront	CI/CD	Main functionalities
Stakeholder	Microservices	Feature Branching	System Breakdown
Use Cases & Features	AWS Fargate	Merge Reviews	Security Testing
	Autoscaling		Maintenance

Key Architectural Decisions: Maintainability, Availability, Security, Performance, Portability

USER LOGIN & HOMEPAGE DEMO

Architecture

Solution Overview – Architectural Diagram



AWS Cognito - Authentication Services and User Management

Key Architectural Decisions

AWS Cognito

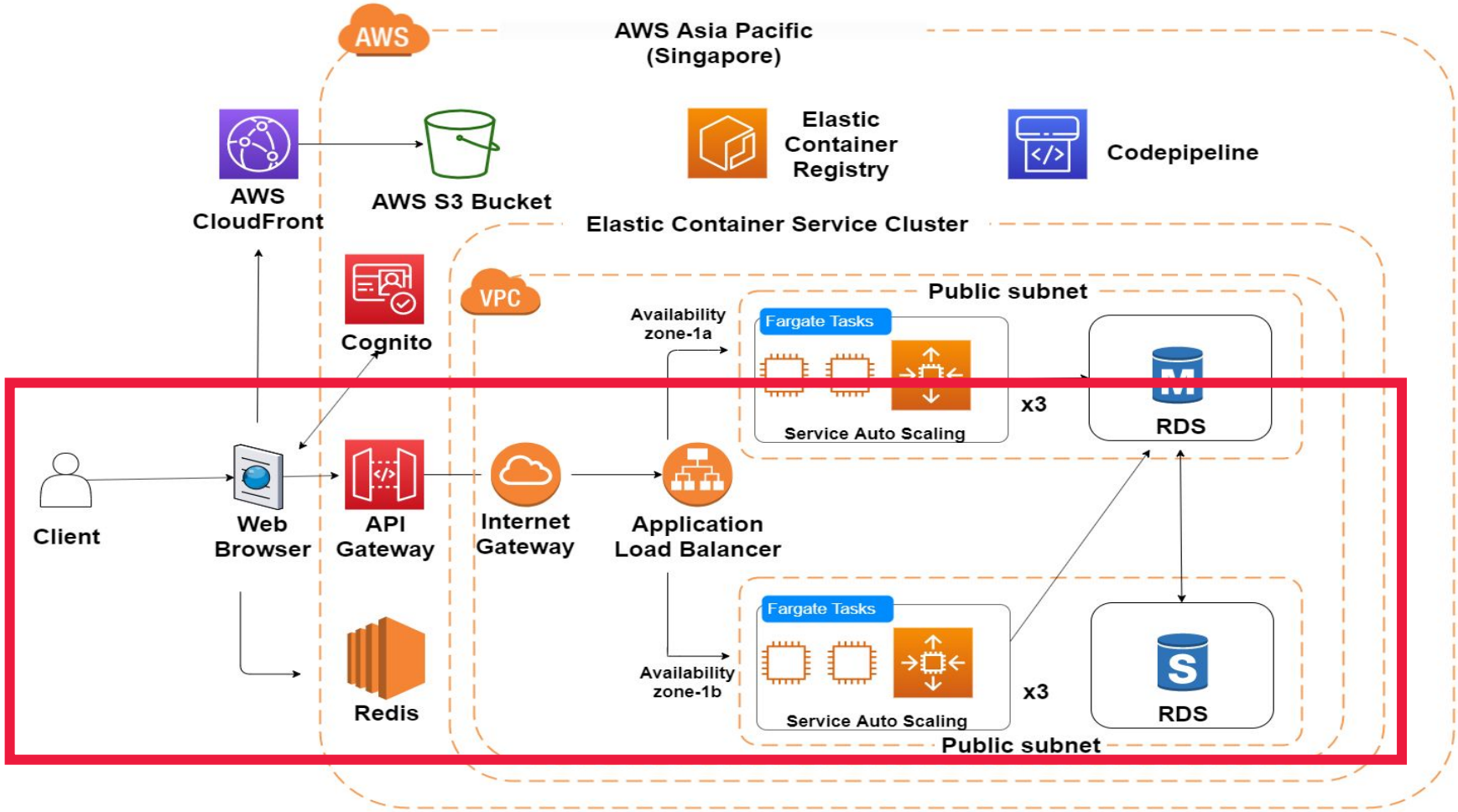
ISO25010:
Security, Compatibility, Maintenance

Decision & Justifications

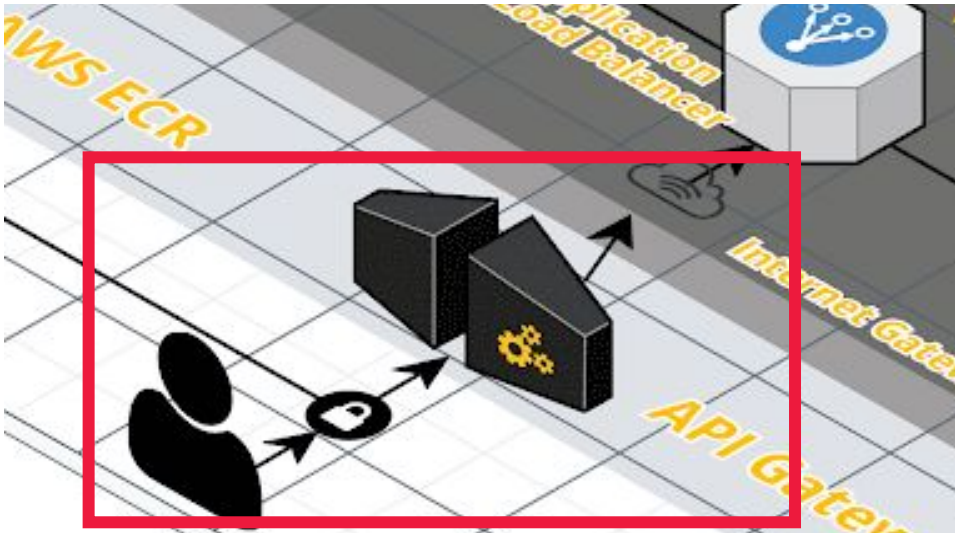
1	**AWS Cognito provides great customization, easy integration with AWS infrastructure**
2	**Role based access, easily configurable SSOs, including Facebook and Google**
3	**Supports Multi Factor Authentication, common security implementation in today's age**

Architecture

Solution Overview – Architectural Diagram



API Gateway
Key Architectural Decisions



API Gateway

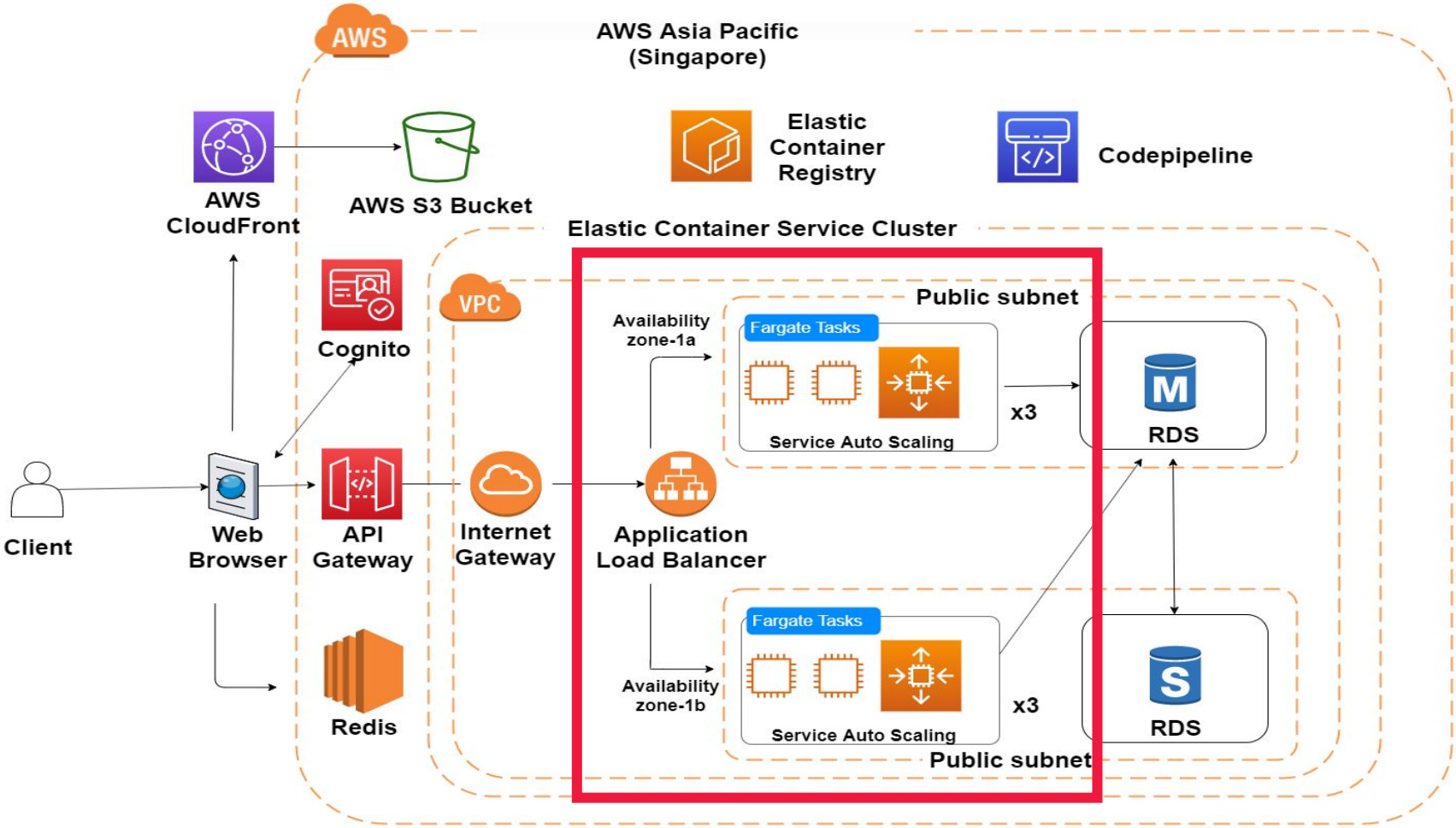
ISO25010:
Security, Modifiability,
Maintainability, Portability

Decision & Justifications

1	A single point of entry, to hide microservices. Provide a high level, unified, client facing interface
2	Façade Design Pattern, abstracts and encapsulates microservices, increasing modifiability, maintainability
3	Security ensured through HTTPS, well integrated with other AWS components, User Access configurations JWT validation,; restriction of access to different roles

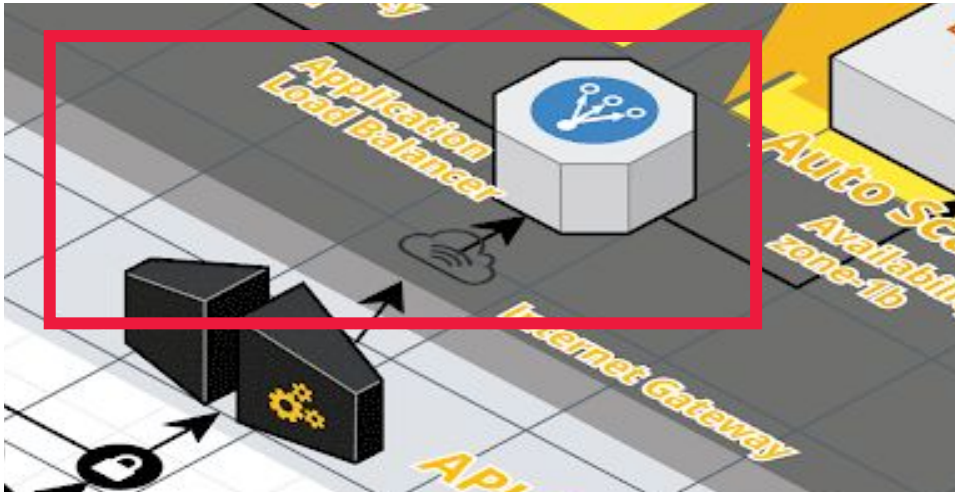
Architecture

Solution Overview – Architectural Diagram



Implementing a load balancer

Key Architectural Decisions



Application Load Balancer

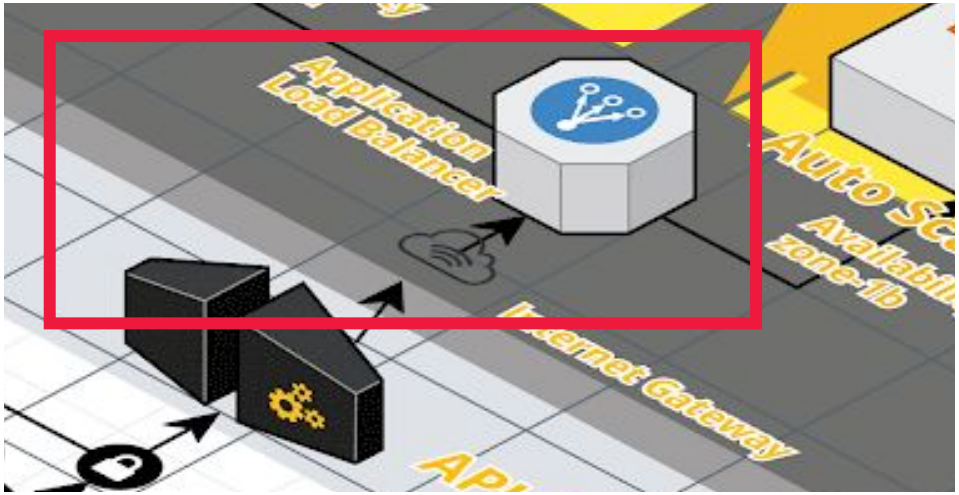
ISO25010:
Availability, Performance Efficiency

Decision & Justifications

1	Direct network traffic to microservices cluster, perform failover duties, thus ensuring high availability
2	Hide services from public by sitting in front of microservices, double up as a reverse proxy
3	Fewer requests handled in each instance after load-balancing, hence increasing performance.`

Implementing a load balancer

Key Architectural Decisions



Application Load Balancer

ISO25010:
Availability, Performance Efficiency

Baseline single instance of moduleservice

Requests	Executions			Response Times (ms)						Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	50	0	0.00%	4167.24	2303	5730	5437.00	5612.85	5730.00	8.31	2.82	1.38

Multiple instances of moduleservice

Requests	Executions			Response Times (ms)						Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	50	0	0.00%	578.10	35	1348	1066.00	1204.45	1348.00	29.53	10.73	6.06

Executive Summary

What we will sharing about today

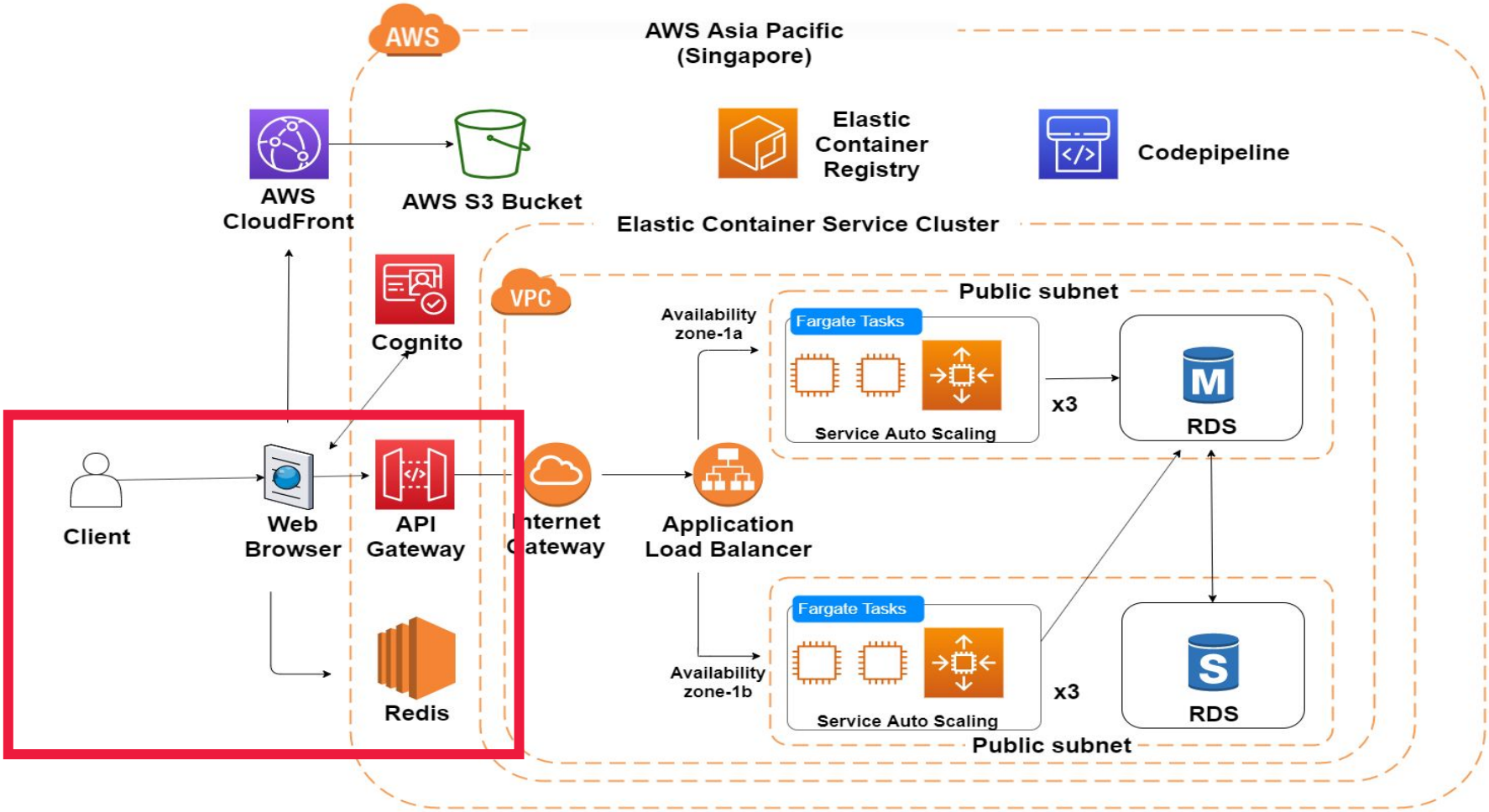
Context	Architectural Overview	Dev Strategy	User Journey & Events
Background	Solution View	Dev view & Strategy	Login & Homepage
Business need	S3 + CloudFront	CI/CD	**Main functionalities**
Stakeholder	Microservices	Feature Branching	System Breakdown
Use Cases & Features	AWS Fargate	Merge Reviews	Security Testing
	Autoscaling		Maintenance

Key Architectural Decisions: Maintainability, Availability, Security, Performance, Portability

VIEW MODULES DEMO

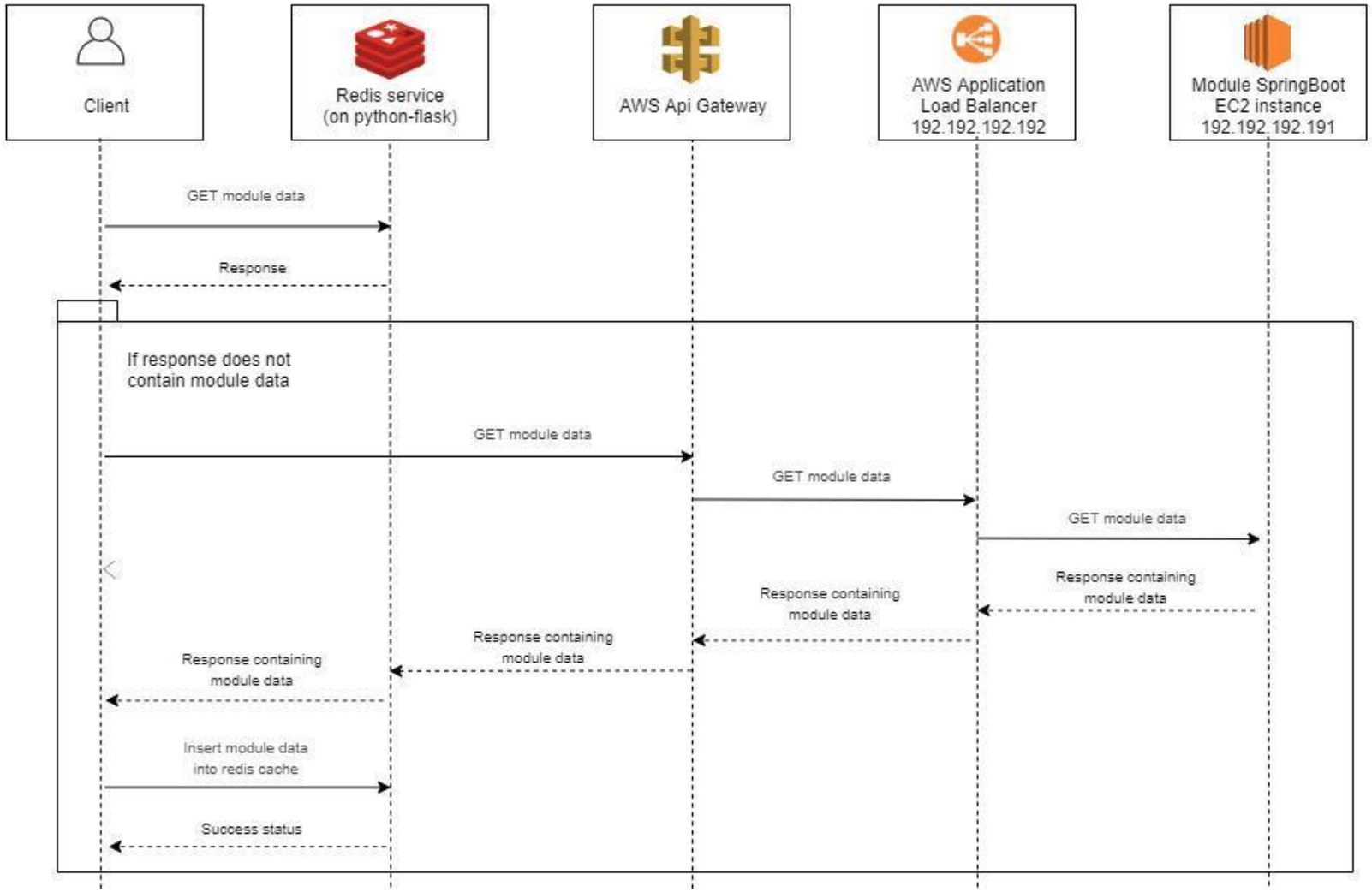
Architecture

Solution Overview – Architectural Diagram



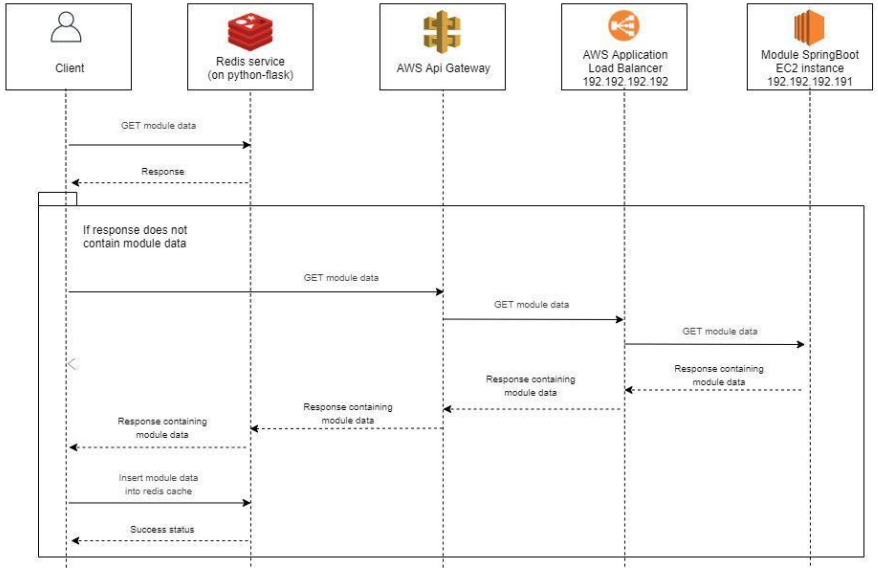
Sequence Diagram

Demonstrating caching with Redis



Implementing In Memory Caching with Redis

Key Architectural Decisions



Redis Cache

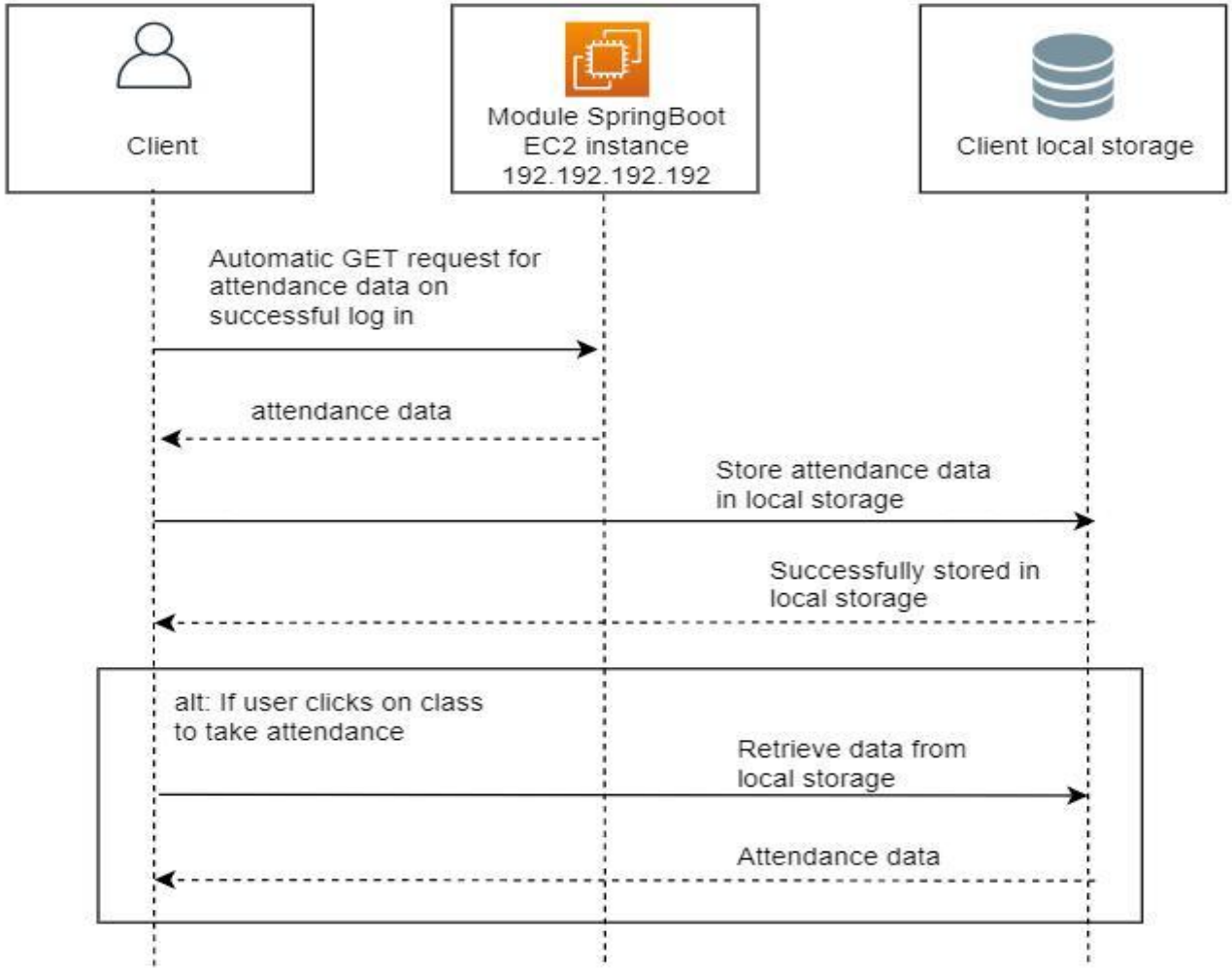
ISO25010:
Performance Efficiency

Decision & Justifications

1	**Reduce hits on the database, improve speed of page loads**
2	**As an in-memory database, Redis provides incredibly fast read and write speeds as compared to normal databases**

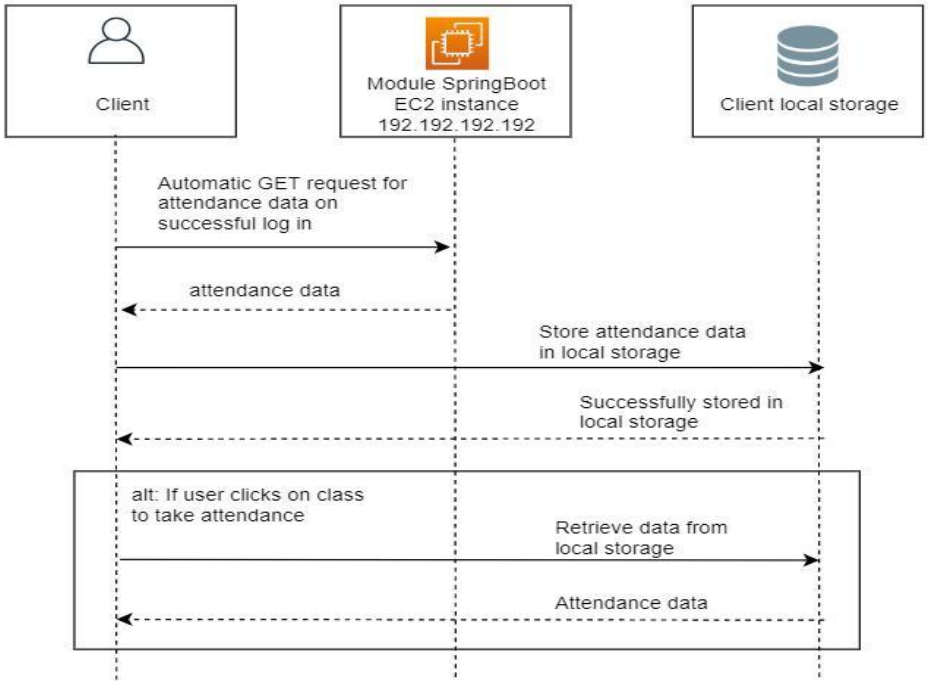
TAKING ATTENDANCE DEMO

Sequence Diagram
Demonstrating Pre-Fetch



Pre-fetch

Key Architectural Diagrams



Pre-fetch

ISO25010:
Performance Efficiency

Decision & Justifications

1	**Faster load time for commonly used functions such as take attendance and view classes**
2	**Reduce load time between pages**

PAY SCHOOL FEES DEMO

Payment Integration with Paypal

ISO25010:
Security, Usability

Decision & Justifications

1	**Established third party that can handle payments for an entity**
2	**PayPal provides great usability through its well design UI**

Executive Summary

What we will sharing about today

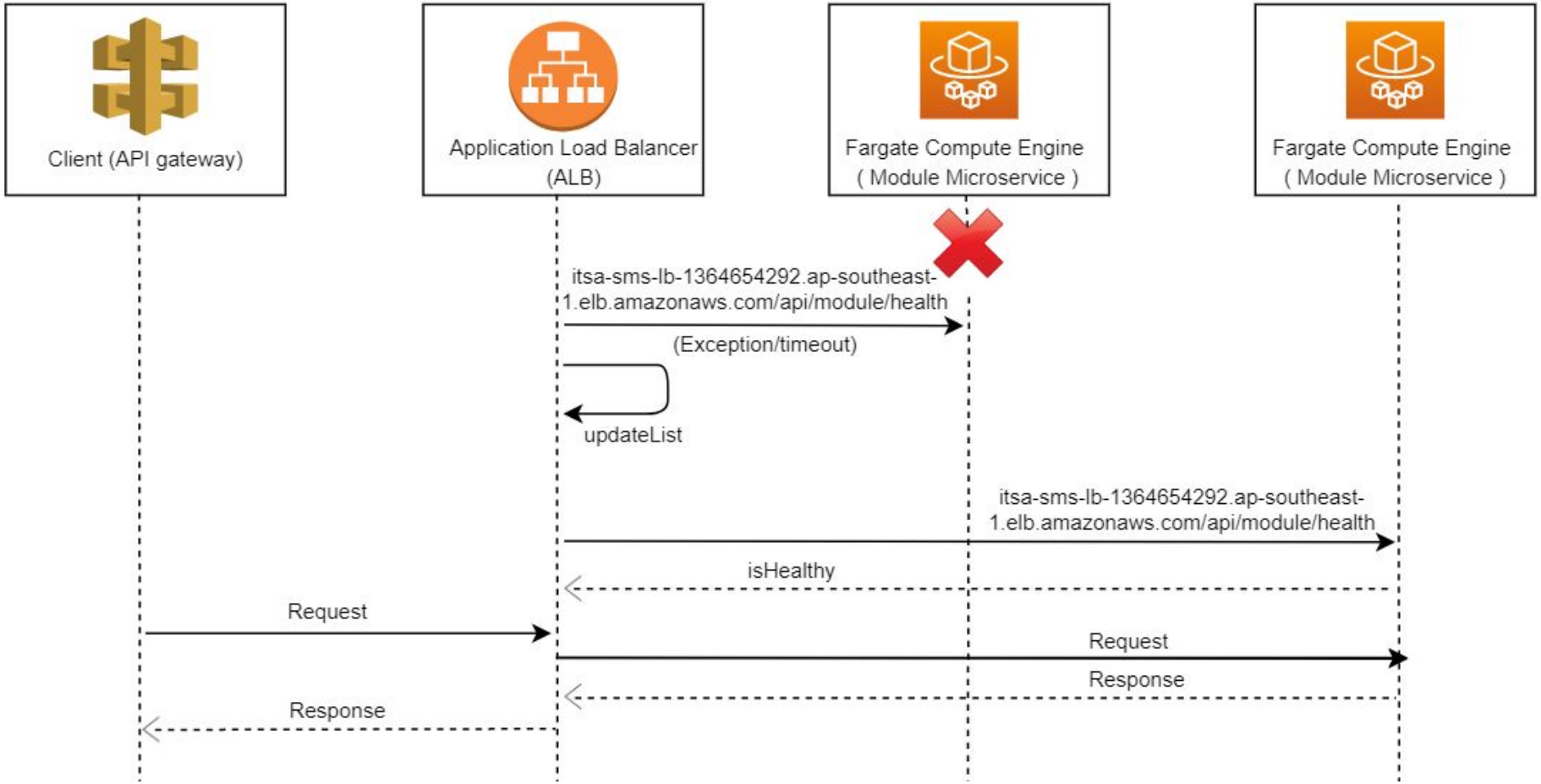
Context	Architectural Overview	Dev Strategy	User Journey & Events
Background	Solution View	Dev view & Strategy	Login & Homepage
Business need	S3 + CloudFront	CI/CD	Main functionalities
Stakeholder	Microservices	Feature Branching	**System Breakdown**
Use Cases & Features	AWS Fargate	Merge Reviews	Security Testing
	Autoscaling		Maintenance

Key Architectural Decisions: Maintainability, Availability, Security, Performance, Portability

SYSTEM BREAKDOWN DEMO: TURNING OFF MICROSERVICE

Sequence Diagram

Demonstrating Failover when an instance fails



SYSTEM BREAKDOWN DEMO: SHUTTING DOWN DATABASE

Database As A Service

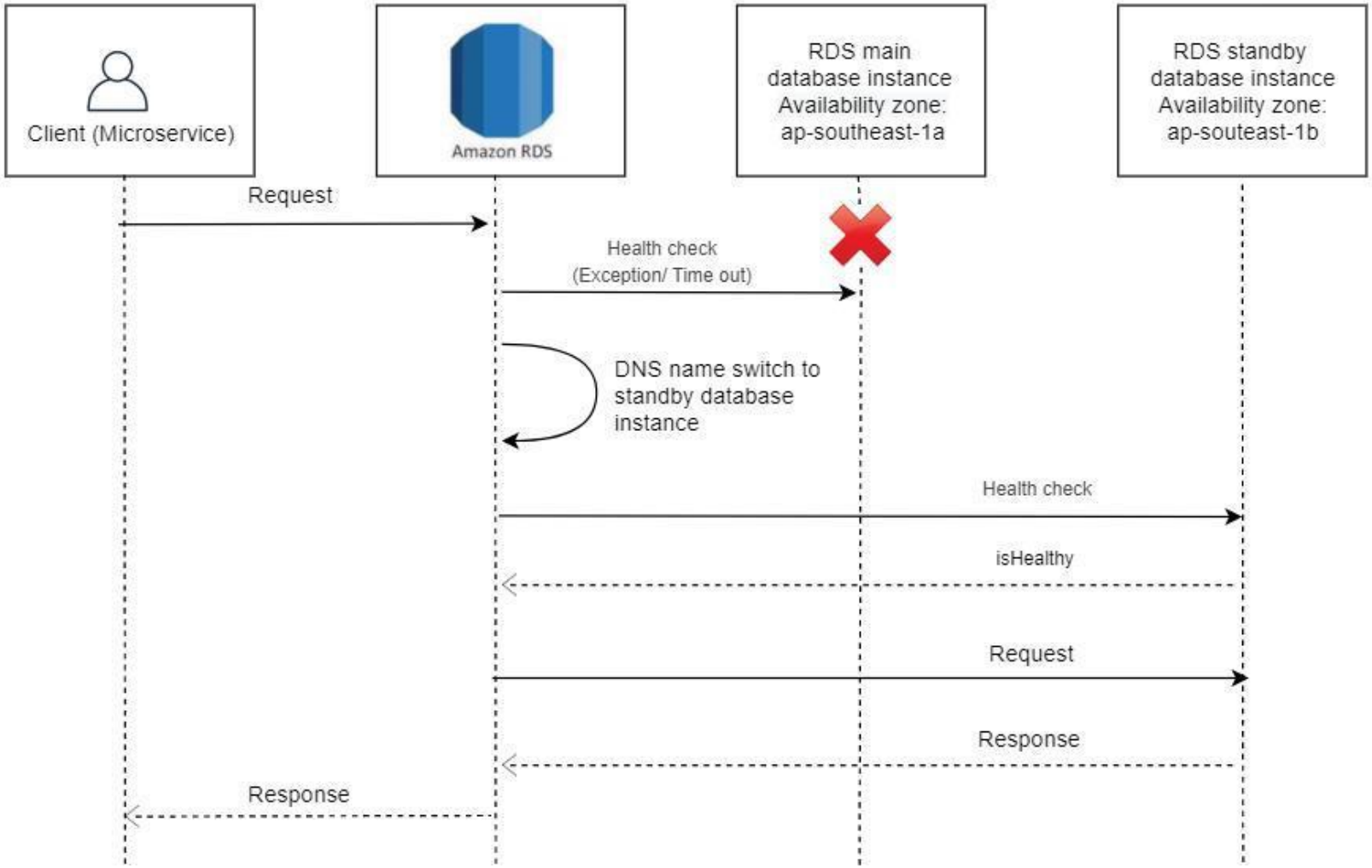
ISO25010:
Availability

Decision & Justifications

1	**Easier to manage - burden management and maintenance on DBaaS provider (no hardware)**
2	**Outsource support to experts with state-of-the-art servers and hardware**
3	**Database more suitable than caching due to requirement of persistent data**

Sequence Diagrams

Demonstration when one instance of the database fails



Executive Summary

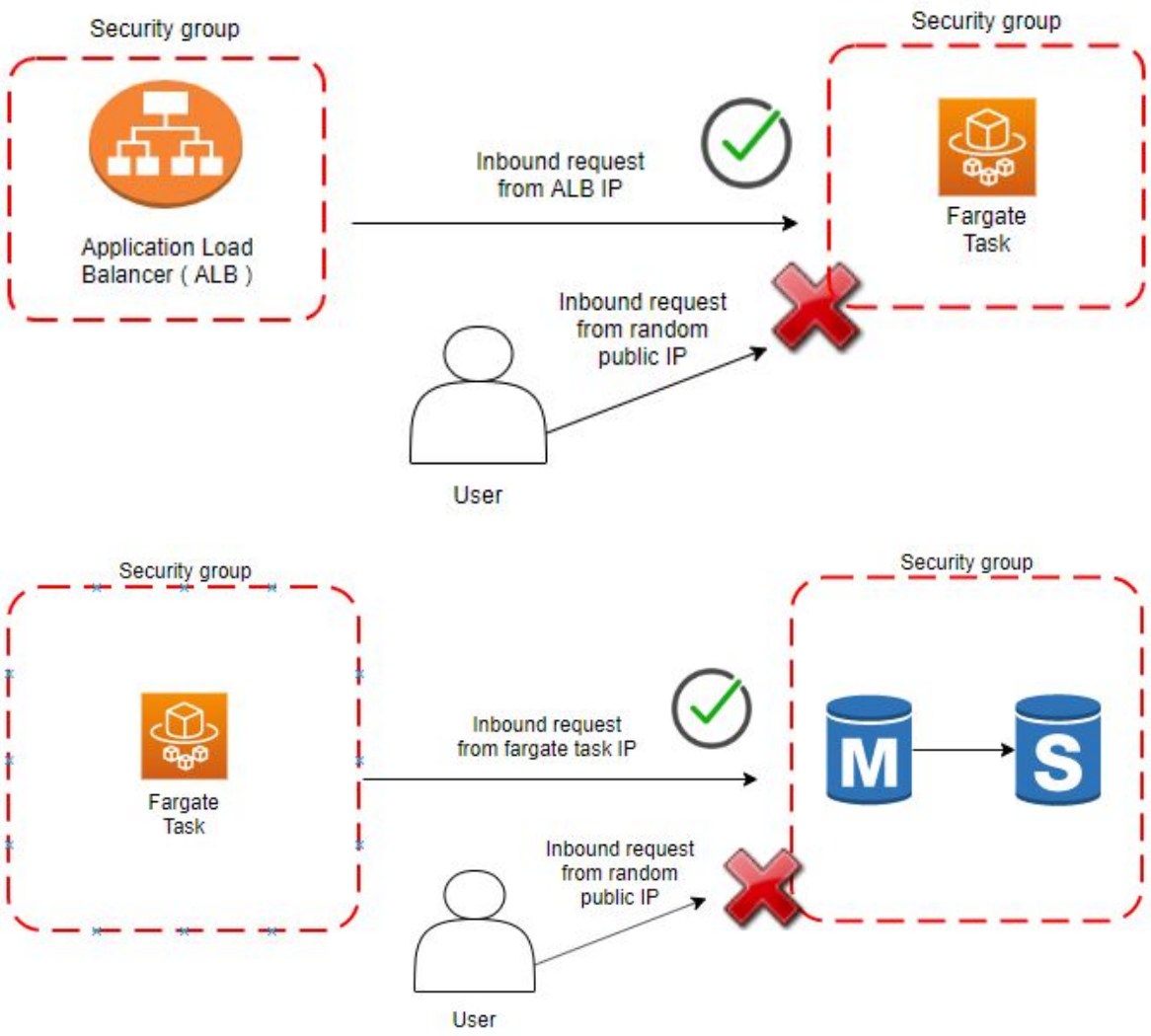
What we will sharing about today

Context	Architectural Overview	Dev Strategy	User Journey & Events
Background	Solution View	Dev view & Strategy	Login & Homepage
Business need	S3 + CloudFront	CI/CD	Main functionalities
Stakeholder	Microservices	Feature Branching	System Breakdown
Use Cases & Features	AWS Fargate	Merge Reviews	**Security Testing**
	Autoscaling		Maintenance

Key Architectural Decisions: Maintainability, Availability, Security, Performance, Portability

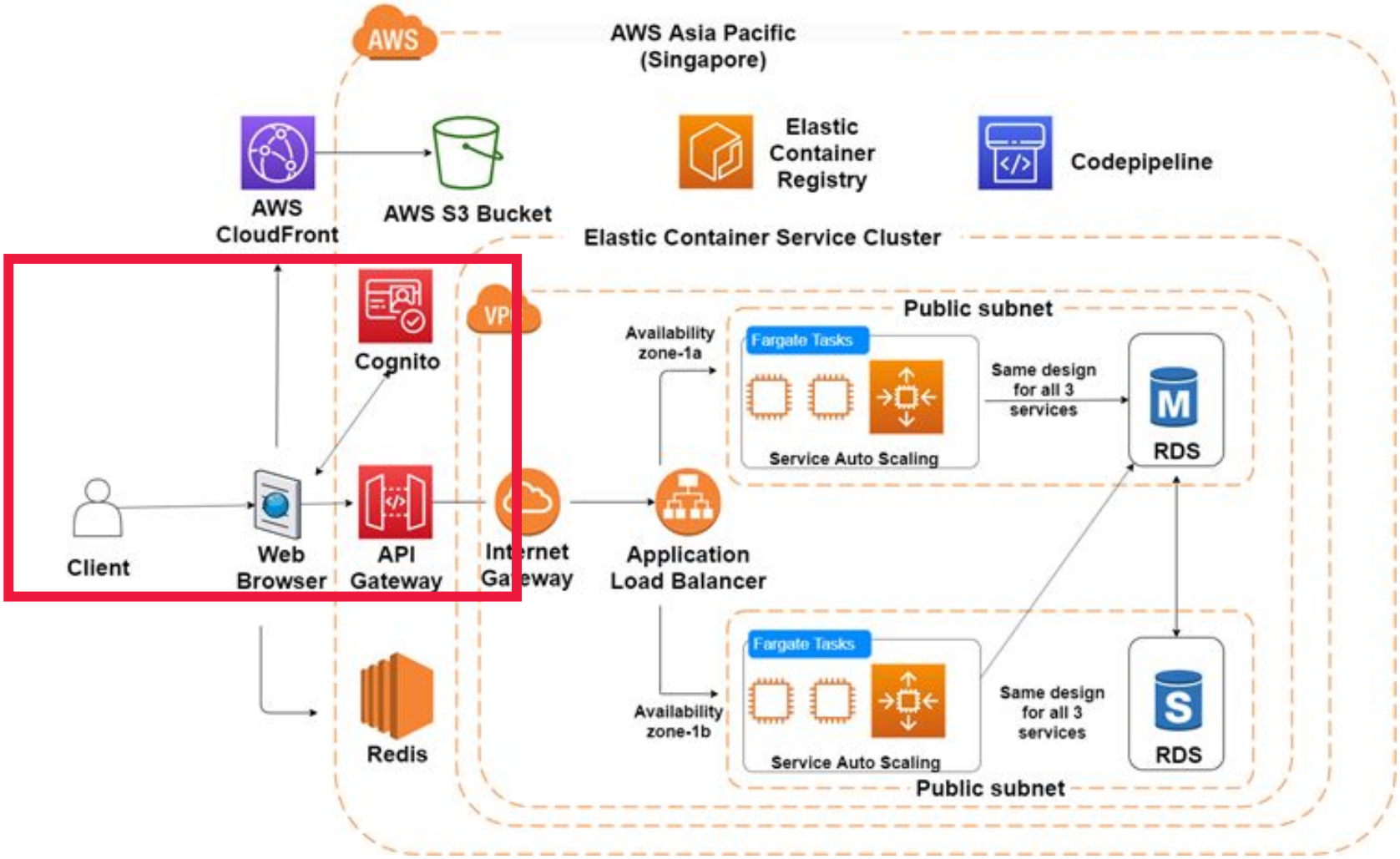
Architecture

Solution Overview – Architectural Diagram



Architecture

Solution Overview – Architectural Diagram



Security View

Vulnerabilities and Mitigations

Asset/Asset Group	Potential Threat/Vulnerability Pair	Possible Mitigation Controls
UI portal	Unauthorized URL Access: when an unauthorized user trying to access a page even though the permission is not granted to them (e.g trying to access admin URL)	Implement AuthGuard in Angular 7 to allow only authorized user to access the different URL endpoints.
HTTP Response for web services	Attacks based on MIME type confusion when browser MIME-sniffs response other than the declared content-type.	Set X-Content-Type-Options header to “nosniff” to reject responses with incorrect MIME types.
Student and module data	Unauthenticated access to web services that can lead to data leakage to unintended people, compromising confidentiality	Set JWT (token returned by AWS Cognito after uses logs in) in headers for each HTTP Request. Microservices can only be called when token has been successfully validated by our API Gateway.
Payment data	Unauthorised update to payment database with the web service in payment system, affecting integrity	Validate payment with payment provider before updating database

Security View
Spring Security

X-Content-Type-Options Header Missing

URL: http://localhost:8080/api/module/getResults?student_id=1
Risk: Low
Confidence: Medium
Parameter: X-Content-Type-Options
Attack:
Evidence:
CWE ID: 16
WASC ID: 15
Source: Passive (10021 - X-Content-Type-Options Header Missing)

Description:

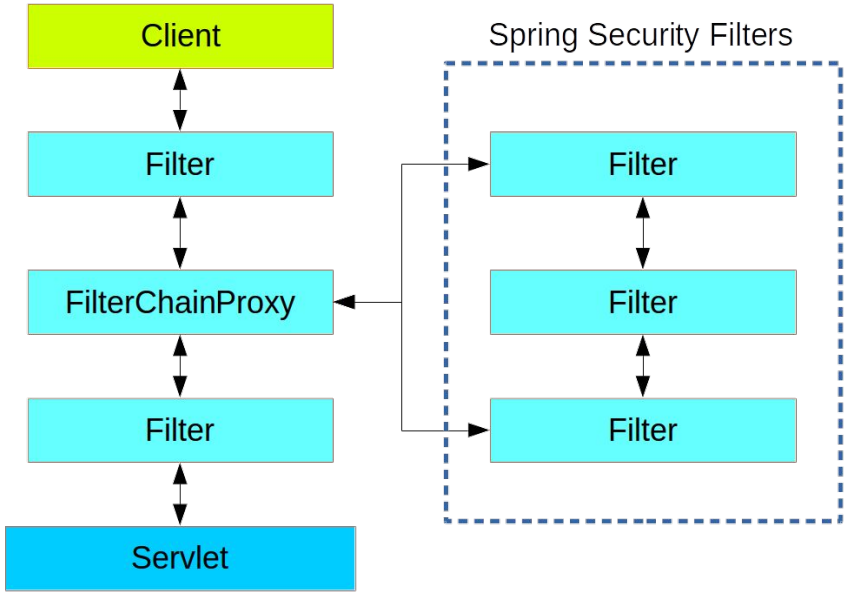
The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosni
content type other than the declared content type. Current (early 2014) and le

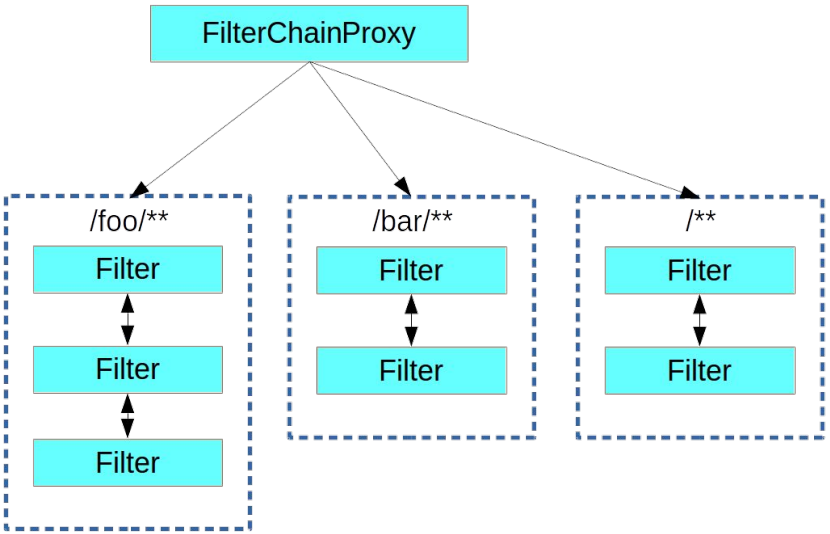
Other Info:

This issue still applies to error type pages (401, 403, 500, etc) as those pag
At "High" threshold this scanner will not alert on client or server error respon

spring
SECURITY

Security View
Spring Security





SECURITY DEMO: OWASP ZAP

Executive Summary

What we will sharing about today

Context	Architectural Overview	Dev Strategy	User Journey & Events
Background	Solution View	Dev view & Strategy	Login & Homepage
Business need	S3 + CloudFront	CI/CD	Main functionalities
Stakeholder	Microservices	Feature Branching	System Breakdown
Use Cases & Features	AWS Fargate	Merge Reviews	Security Testing
	Autoscaling		Maintenance

Key Architectural Decisions: Maintainability, Availability, Security, Performance, Portability

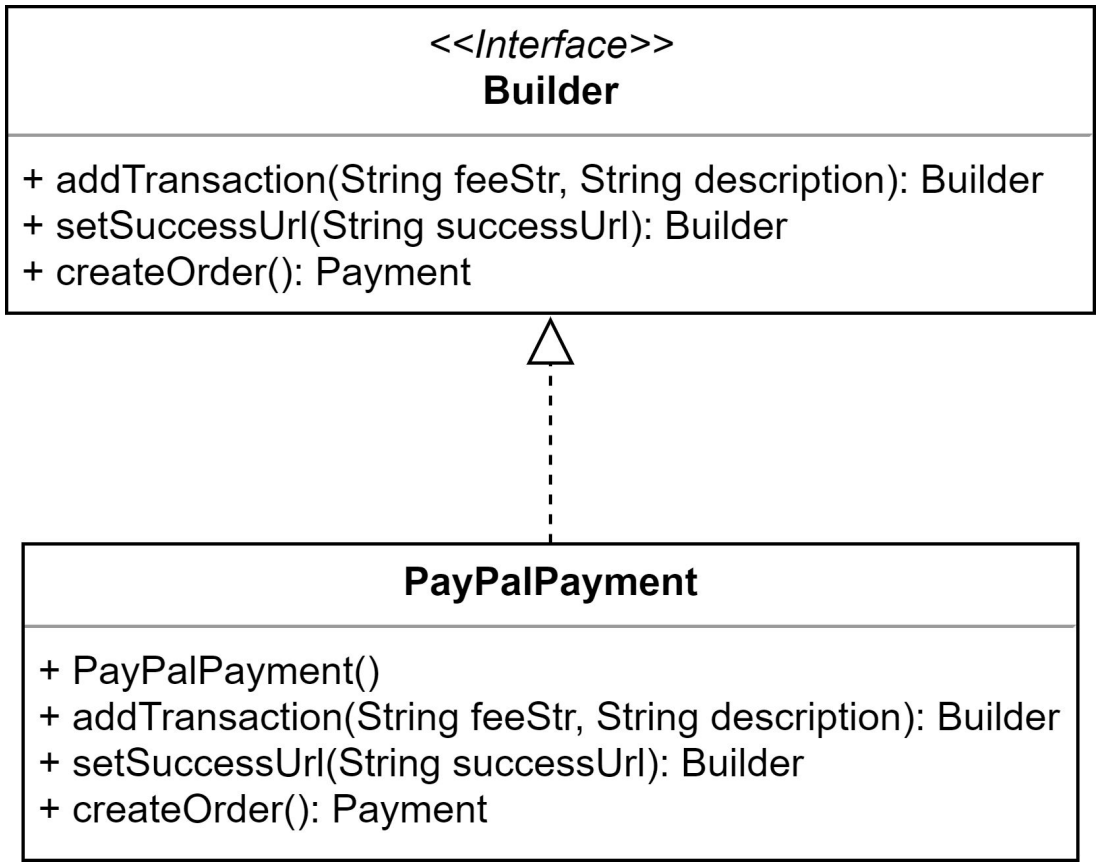
Builder Pattern

ISO25010:
Maintainability

Decision & Justifications

1	**Useful to create complex PayPal payment object with multiple parts (details, redirect urls, amount, etc.)**
2	**Encapsulate with Builder object to hide all details, making the design of the payment object easier**

Builder Pattern
Class Diagram



Builder Pattern

Build Object

```
Payment payment = new PayPalPayment()
                .setSuccessUrl(successUrl)
                .setFailureUrl(failureUrl)
                .addTransaction(amountStr, "School Fees")
                .createOrder();
```

THANK YOU!

Q&A

AMOS LAM | GABRIEL KOH | NG RUI QIN | LIN HAN HUI | LIU ZUO LIN | TRUONG HAI BANG