# Technology Stack

* Java Spring Boot Framework 3.4.1
* Java 17
* Hibernate (h2) as database
* Apache Maven as dependency management
* GitHub
* 3rd parties
  + Spring Actuator
  + Spring Data JPA (ORM)
  + Spring Security
  + Spring Web (REST)
  + Spring Validation
  + Lombok
  + Spring Test (Mockito used)
  + Swagger (documentation)
  + Mapstruct
  + Passay
  + JsonWebToken

# System Requirements (how to run)

* Java 17 (developed in amazon-coretto w/ 17.0.11 distribution)
* Maven (for getting dependencies)
  + mvn clean install (Downloads 3rd parties and prepare a JAR file)

Alternatively,

* Docker (Possible to build an image without others)
  + docker build -t cmc:1 .
  + docker run -d -p 8080:8080 cmc:1

Note that – When system is started first time, administrator user has been automatically created by ExampleDataInitializer class.

* Admin email: [admin@email.com](mailto:admin@email.com)
* Admin username: admin
* Admin password: password

Other users are created with standard role. So, authorization is limited.

Client API documentation can be found in <http://localhost:8080/swagger-ui.html> after project starts.

# Project Hierarchy

Project hierarchy is given below. It can be also organized as common, config and modules (consists of service, controller, entity, repository, mapper for each capability such as user, customer and loan).

A screenshot of a computer

Description automatically generated

In this project, the following modules are implemented out:

* User
  + User entity
  + User repository
  + User service
  + User controller
* Customer
  + Customer entity
  + Customer repository
  + Customer service
  + Customer controller
* Loan
  + Loan and Loan Installment entities
  + Loan and Loan Installment repositories
  + Loan service
  + Loan controller

In this layered architecture, endpoints are defined in controller. They take an object input, convert object into DTO (as also known as data transfer object) and send to service layer. After the service method has been called, it returns DTO to controller and it’s automatically converted to object from DTO once again. Mapper layer is for conversion of DTO to object or object to DTO.

The service layer interacts with repository directly. Each repository is associated with an entity. The service layer is responsible for applying/performing business logic.

# Logging

Logging is provided with logback implementation.

## Client logging

Interceptor and filter are developed for client logging for incoming and outgoing. There is a trade off for getting pay load from HTTP request while using interceptor. So, filter implementation is also used for this purpose.

## Exception logging

Exception handlers run for specific service methods, and they’re thrown from controllers. They are also recorded by logback.

# Authentication and Authorization

All endpoints need that user authenticates first. From authorization perspective, the endpoints are limited based on role.

JWT Token is used. Token includes username and role information as well.

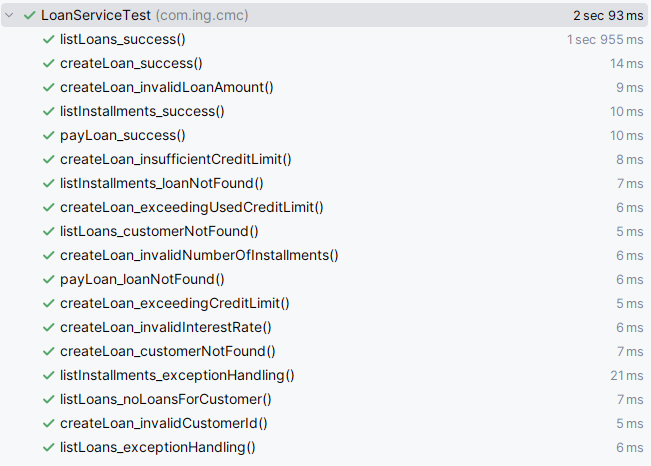
# Response

HTTP responses are designed to generic response pattern, and HTTP status code such as 403 can be seen.

# Testing

## Service test

Cases for unit testing are prepared by using Mockito. Test results can be seen in the following picture.



## Controller test

Postman collection can be found in test folder for API testing.

A screenshot of a computer

Description automatically generated

Requests and responses are given below.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated