International Institute of Information Technology, Bangalore

Softwate Testing (CSE 731)

Project Report Mutation Testing

Submitted by :Abhay Bhadoriya (MT2024003)
: Naval Kishore Singh Bisht (MT2024099)

Under the guidance of Prof. Meenakshi D Souza

TABLE OF CONTENTS

TABLE OF CONTENTS **2**
CHAPTER 1: INTRODUCTION **4**
1.1 Project Overview 4
1.2 Motivation 4
1.3 Problem Statement 4
CHAPTER 2: THEORETICAL FRAMEWORK (MUTATION TESTING) **5**
2.1 Introduction to Software Testing Paradigms 5
2.2 Limitations of Traditional Coverage Metrics 5
2.3 Fundamental Concepts of Mutation Testing 6
2.4 The Mutation Testing Process 7
2.5 Mathematical Model: The Mutation Score 8
2.6 Types of Mutants & Operators 8
2.6.1 Mathematical Mutators 8
2.6.2 Conditional Mutators 8
2.6.3 Return Value Mutators 9
2.6.4 Void Method Call Mutator 9
2.7 The Equivalent Mutant Problem 9
2.8 Mutation Testing Tools: PIT (Pitest) 9
CHAPTER 3: SYSTEM DESIGN & IMPLEMENTATION **10**
3.1 Functional Requirements 10
3.2 Technology Stack 10
3.3 Backend Architecture 11
3.4 Database Schema 12
3.5 Key Logic Implementation 13
1. payFees Method 13
CHAPTER 4: TEST SUITE IMPLEMENTATION **15**
4.1 Unit Testing Strategy 15
4.2 Mocking with Mockito 15
4.3 Initial Coverage Metrics 16
5.1 PIT Configuration 17
5.2 Quantitative Results 17
5.3 Detailed Analysis: Killed Mutants 18
Case 1: Security Critical - Password Encryption 18
Case 2: Financial Logic - Remaining Balance 18
Case 3: Authentication - Login Logic 19
5.4 Detailed Analysis: Survived Mutants 19
Survivor 1: Mapper Conditional 19

Survivor 2: Lambda Exception Handling....19
CHAPTER 6:Unit and Integration Operators....26
1. Unit Level Operators....26
2. Integration Level Operators....27
CHAPTER 7: How to run....28
Command....28
2. Install Command....28
3. Test Command....28
4. Mutation Coverage Command....29
CHAPTER 8: CONCLUSION & FUTURE SCOPE....29
8.1 Conclusion....29
8.2 Future Scope....29
REFERENCES....30

CHAPTER 1: INTRODUCTION

1.1 Project Overview

The **Student Billing System** is a full-stack web application designed to streamline the financial operations of educational institutions. It provides a centralized platform for administrators to issue bills and for students to view dues, make partial payments, and track their transaction history. The project is built using **Spring Boot** for the backend and **ReactJS** for the frontend, ensuring a robust and scalable architecture.

However, the primary focus of this report is not just the development of the system, but the rigorous **verification and validation** of its backend logic using **Mutation Testing**.

1.2 Motivation

In financial software, a simple logic error can lead to significant monetary discrepancies. Traditional testing methods often rely on "Code Coverage" (e.g., checking if 90% of lines were executed). However, high code coverage does not guarantee correctness. A test can execute a line of code without actually checking if the output is correct (the "assertion-free" test problem).

We aimed to go beyond traditional metrics by employing **Mutation Testing**, a fault-based testing technique that mathematically proves the quality of a test suite by artificially "breaking" the code and checking if the tests detect the damage.

1.3 Problem Statement

The goal is to answer the question: "Does our test suite actually verify the business logic, or does it merely execute it?"

To answer this, we subjected our CustomerService and helper modules to thousands of artificial faults (mutants) using the PIT framework and analyzed the survival rate of these mutants.

CHAPTER 2: THEORETICAL FRAMEWORK (MUTATION TESTING)

2.1 Introduction to Software Testing Paradigms

Software testing is the process of executing a program with the intent of finding errors. It is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding.

Testing is generally classified into two main paradigms:

1. **Black Box Testing:** The tester has no knowledge of the inner workings of the application. Tests are derived from requirements and specifications (e.g., Functional Testing, System Testing).

2. **White Box Testing:** The tester has full knowledge of the internal logic and structure of the code. Tests are designed to exercise specific paths, branches, and conditions (e.g., Unit Testing, Integration Testing).

2.2 Limitations of Traditional Coverage Metrics

The most common metric for judging the quality of Unit Tests is **Code Coverage** (or Line Coverage). It measures the percentage of code lines executed during the test run.

The "Test Effectiveness Paradox":

It is possible to have 100% line coverage with 0% fault detection. Consider the following example:

Java

```java
public int add(int a, int b) {
    return a + b;
}
```

Bad Test:

Java

```java
@Test
public void testAdd() {
   calculator.add(2, 3); // Code is executed, but result is NOT checked.
}
```

In this scenario, Line Coverage is 100%, but if a developer changes `a + b` to `a - b`, the test will still pass. This proves that coverage measures *execution*, not *verification*.

2.3 Fundamental Concepts of Mutation Testing

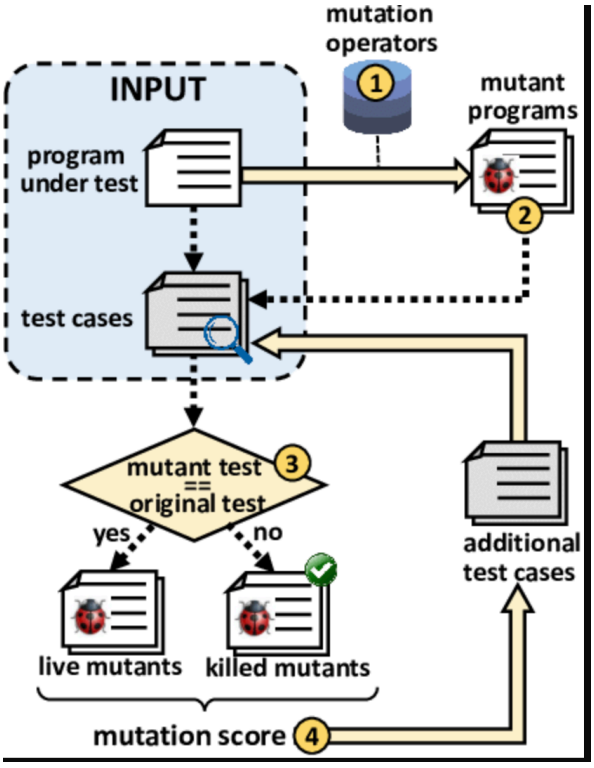
Mutation Testing is a white-box method that evaluates the quality of the *test suite* rather than the production code. It operates on two fundamental hypotheses:

1. **The Competent Programmer Hypothesis:** This states that programmers are generally competent and that most faults are "simple" syntactic errors (e.g., using `+` instead of `-`, or `<` instead of `<=`). Programmers rarely write code that is "wildly" incorrect; they mostly make small typographical or logical slips.

2. **The Coupling Effect:** This hypothesis asserts that complex faults are essentially coupled to simple faults. Therefore, a test data set that detects all simple faults is sensitive enough to detect most complex faults as well.

2.4 The Mutation Testing Process

The process of mutation testing involves the following steps



1. **Mutant Generation:** The mutation testing framework parses the source code and creates many modified versions called "Mutants." Each mutant contains exactly one syntactic change (a fault).

2. **Test Execution (Original):** The existing test suite is run against the original, un-mutated code to ensure it passes. If the tests fail on the original code, mutation testing cannot proceed.

3. **Test Execution (Mutants):** The test suite is run against each mutant.
    - **Killed:** If a test fails when running against a mutant, the mutant is considered "Killed." This is the desired outcome, as it proves the test suite noticed the change.
    - **Survived:** If all tests pass despite the mutation, the mutant "Survived." This indicates a gap in the test suite (the test failed to detect the fault).

4. **Analysis:** The tester analyzes the survived mutants to determine if they are "Equivalent Mutants" or true weaknesses in the test suite.

2.5 Mathematical Model: The Mutation Score

The effectiveness of a test suite is quantified by the **Mutation Score (MS)**. This metric represents the percentage of non-equivalent mutants that were detected by the test suite.

The formula is defined as:

$$MS = \frac{K}{(T - E)} \times 100$$

Where:

- K = Number of Killed Mutants
- T = Total Number of Mutants Generated
- E = Number of Equivalent Mutants

2.6 Types of Mutants & Operators

Mutation tools use specific rules called "Operators" to generate mutants. Understanding these is crucial for analysis.

2.6.1 Mathematical Mutators

These operators mimic errors in arithmetic logic.

- **Math Mutator:** Replaces binary arithmetic operations.
  - a + b $\rightarrow$ a - b
  - a * b $\rightarrow$ a / b
  - a % b $\rightarrow$ a * b
  - *Relevance:* Critical for financial applications like our Billing System where RemainingBalance calculations must be exact.

2.6.2 Conditional Mutators

These operators test boundary conditions, often the source of "off-by-one" errors.

- **Conditionals Boundary Mutator:**
  - if (i < 10) $\rightarrow$ if (i <= 10)
- **Negate Conditionals Mutator:**
  - if (i == j) $\rightarrow$ if (i != j)
  - *Relevance:* Essential for loop control and validation logic (e.g., verifying if a payment date is strictly *before* a deadline).
  -

2.6.3 Return Value Mutators

These simulate a method returning incorrect data, testing if the calling function verifies the response.

- **Empty Object Return:** Returns Collections.emptyList() instead of a populated list.
- **Null Return:** Returns null instead of an object.
- **Primitive Returns:** Returns 0 or false fixed values.

2.6.4 Void Method Call Mutator

This operator removes calls to methods that have no return value (void methods).

- *Example:* Removing database.save(user) or securityContext.setAuthentication(auth).
- *Relevance:* This is one of the most powerful operators. In our project, it helped verify that customer.setPassword() was actually being called.

2.7 The Equivalent Mutant Problem

An **Equivalent Mutant** is a modification that does not change the semantic behavior of the program.

- *Example:* Changing a loop for (int i=0; i<10; i++) to for (int i=0; i!=10; i++). Both loops execute exactly 10 times.
- *Challenge:* Tests cannot kill these mutants because the program output remains correct. Identifying equivalent mutants is theoretically undecidable (related to the Halting Problem), making 100% mutation coverage practically impossible to automate fully.

2.8 Mutation Testing Tools: PIT (Pitest)

For this project, we used **PIT**, the state-of-the-art mutation testing system for Java.

- **Bytecode Manipulation:** Unlike older tools that recompile source code for every mutant (which is slow), PIT modifies the compiled bytecode (.class files) in memory. This makes it significantly faster.
- **Integration:** It integrates seamlessly with Maven and JUnit 5.
- **Coverage Optimization:** PIT only runs tests that actually execute the mutated line of code (using code coverage data), saving massive amounts of time.

CHAPTER 3: SYSTEM DESIGN & IMPLEMENTATION

3.1 Functional Requirements

Based on the project scope, the system fulfills the following requirements:

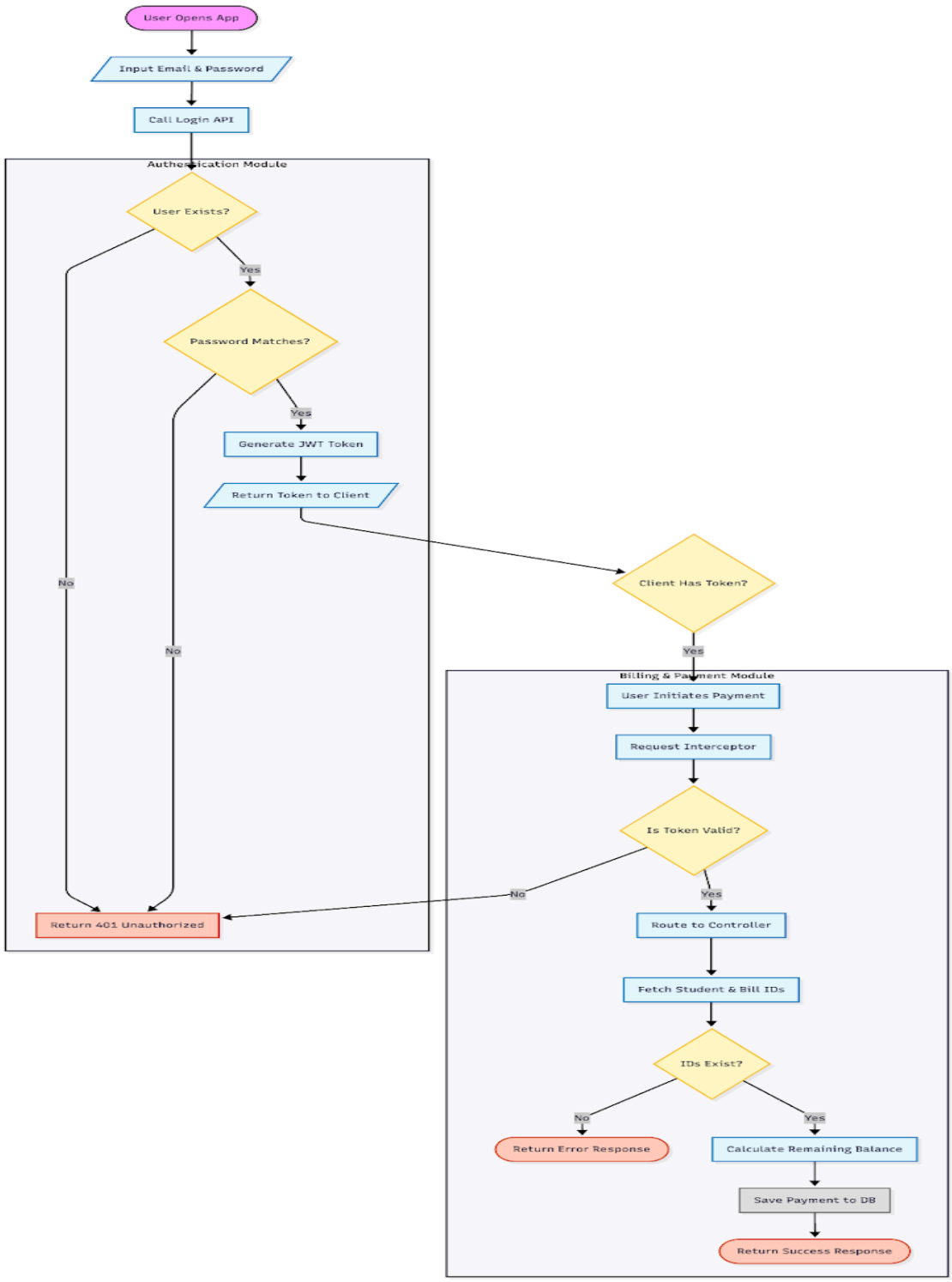
1. **Student Registration & Login:** Secure onboarding using Email and Password.
2. **View Due Bills:** Students can retrieve a list of unpaid obligations.
3. **Payment Processing:** The system supports partial payments. A student can pay a fraction of a bill, and the system updates the "Remaining Amount" accordingly.
4. **Payment History:** A complete log of all transactions is maintained.

3.2 Technology Stack

- **Language:** Java 17 (LTS)
- **Framework:** Spring Boot 3.x (Web, Data JPA, Security)
- **Database:** MySQL
- **Testing:** JUnit 5, Mockito, Pitest 1.15.2
- **Security:** JJWT (Java JWT) for token-based authentication.

3.3 Backend Architecture

The project follows the standard Controller-Service-Repository pattern.



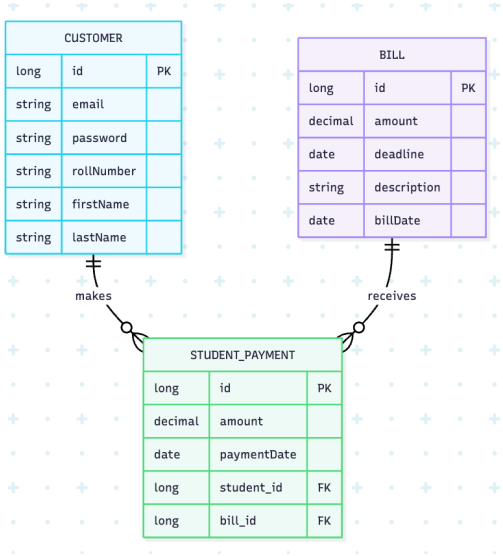
Key Classes:

- **CustomerService.java**: The core business logic layer. It handles student registration, login validation, bill retrieval, and fee processing.
- **JWTHelper.java**: Utility for generating and signing tokens.
- **RequestInterceptor.java**: A filter that intercepts HTTP requests to validate the Authorization: Bearer <token> header.

3.4 Database Schema

The data model consists of three primary entities:

1. **Customer:** (id, email, password, rollNumber, firstName, lastName)
2. **Bill:** (id, amount, deadline, description, billDate)
3. **StudentPayment:** (id, amount, paymentDate, student_id, bill_id)



3.5 Key Logic Implementation

The most complex logic lies in the payFees method, which ensures data integrity during payments.

1. payFees Method

```
public ResponseEntity<Object> payFees(FeesPayment feesPayment) {
  try {
      // Retrieve student and bill based on provided IDs
      Customer student = customerRepo.findById(feesPayment.studentId())
              .orElseThrow(() -> new CustomerNotFoundException(
                      format("Cannot update Fees:: No customer found with the provided ID:: %d", feesPayment.studentId())
              ));
      Bill bill = billRepo.findById(feesPayment.billId())
              .orElseThrow(() -> new BillNotFoundException(
                      format("Cannot update Fees:: No Bill found with the provided ID:: %d", feesPayment.studentId())
              ));
      // Map the DTO to the entity
        StudentPayment studentPayment = customerMapper.toEntity(feesPayment, student, bill);

        // Save the payment
        studentPaymentRepo.save(studentPayment);
        // Return success response
        return ResponseEntity.status(HttpStatus.CREATED)
                .body(Map.of(
                        "status", "success",
                        "message", "Fees Paid Successfully"
                ));
    } catch (DataIntegrityViolationException e) {
        // Handle database constraint violations, such as duplicate email

        return ResponseEntity.status(HttpStatus.CONFLICT)
                .body(Map.of(
                        "status", "error",
                        "message", "Fees already Paid"
                ));
    } catch (IllegalArgumentException e) {
        // Handle invalid arguments, such as null or improperly formatted fields
        return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                .body(Map.of(
                        "status", "error",
                        "message", "Invalid input: " + e.getMessage()
                ));
    } catch (Exception e) {
        // Handle any unexpected exceptions
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body(Map.of(
                        "status", "error",
                        "message", "An unexpected error occurred. Please try again later."
                ));
    }
}
```

2. `CustomerServiceTest` Setup

```
@ExtendWith(MockitoExtension.class)
class CustomerServiceTest {

    @Mock
    private CustomerRepo customerRepo;

    @Mock
    private CustomerMapper customerMapper;

    @Mock
    private EncryptionService encryptionService;

    @Mock
    private BillRepo billRepo;

    @Mock
    private StudentPaymentRepo studentPaymentRepo;

    @Mock
    private JWTHelper jwtHelper;

    @InjectMocks
    private CustomerService customerService;

    private Customer mockCustomerRaw;
    private CustomerRequest mockCustomerRequest;
    private Bill mockBill;
    @BeforeEach
    void setUp() {
        // Customer as returned by the Mapper (Raw Password)
        mockCustomerRaw = Customer.builder()
                .id(1L)
                .email("test@example.com")
                .password("rawPassword123") // Raw password initially
                .rollNumber("R101")
                .build();
        // Use POJO Constructor for Request
        mockCustomerRequest = new CustomerRequest(
                9.0f, "IT", "test@example.com", "Test", "User", 2025,
                "rawPassword123", null, 0, "R101", "Cyber", 4.0f
        );
        mockBill = Bill.builder()
                .id(10L)
                .amount(new BigDecimal("1000.00"))
                .build();
    }
```

CHAPTER 4: TEST SUITE IMPLEMENTATION

Our testing strategy focused on Unit Testing using **JUnit 5** and **Mockito**. We mocked the repository layer to isolate the service logic.

4.1 Unit Testing Strategy

We ensured that every service method had corresponding positive and negative test cases.

- **Positive:** Valid inputs, successful save, successful login.
- **Negative:** Invalid password, duplicate email, database connection failure.

4.2 Mocking with Mockito

To test CustomerService without a real database, we used @Mock.

Code Snippet: CustomerServiceTest.java Setup

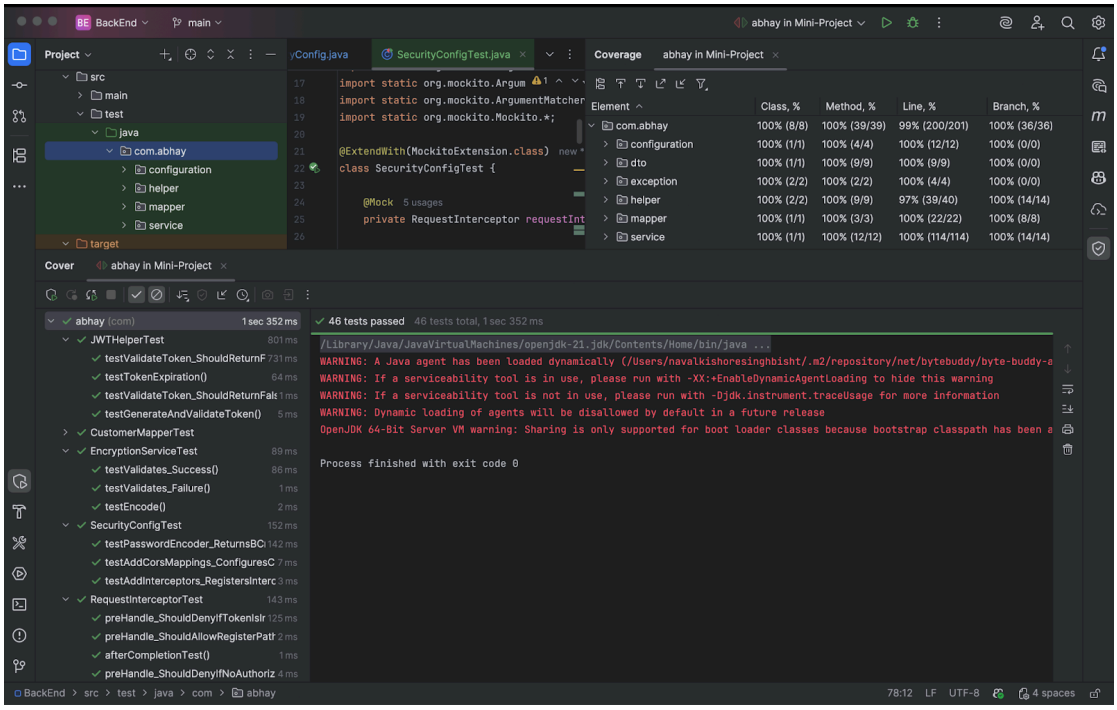
```
@ExtendWith(MockitoExtension.class)
class CustomerServiceTest {

    @Mock
    private CustomerRepo customerRepo;

    @Mock
    private CustomerMapper customerMapper;

    @Mock
    private EncryptionService encryptionService;

    @Mock
    private BillRepo billRepo;

    @Mock
    private StudentPaymentRepo studentPaymentRepo;

    @Mock
    private JWTHelper jwtHelper;

    @InjectMocks
    private CustomerService customerService;

    private Customer mockCustomerRaw;
    private CustomerRequest mockCustomerRequest;
    private Bill mockBill;

    @BeforeEach
    void setUp() {
        // Customer as returned by the Mapper (Raw Password)
        mockCustomerRaw = Customer.builder()
                .id(1L)
```

```
                .email("test@example.com")
                .password("rawPassword123") // Raw password initially
                .rollNumber("R101")
                .build();

        // Use POJO Constructor for Request
        mockCustomerRequest = new CustomerRequest(
                9.0f, "IT", "test@example.com", "Test", "User", 2025,
                "rawPassword123", null, 0, "R101", "Cyber", 4.0f
        );

        mockBill = Bill.builder()
                .id(10L)
                .amount(new BigDecimal("1000.00"))
                .build();
    }
```

4.3 Initial Coverage Metrics

Before running PIT, we verified our Line Coverage.

- **Class Coverage:** 100%
- **Method Coverage:** 100%
- **Line Coverage:** 99%



CHAPTER 5: ANALYSIS & RESULTS

This chapter presents the core findings of our project. We executed PIT against the com.abhay package structure.

5.1 PIT Configuration

The tool was configured in pom.xml to target our specific source classes and test classes.

Code Snippet: pom.xml configuration

```
<plugin>
   <groupId>org.pitest</groupId>
   <artifactId>pitest-maven</artifactId>
   <version>1.15.2</version>
   <configuration>
       <targetClasses>
           <param>com.abhay.*</param>
       </targetClasses>
       <targetTests>
           <param>com.abhay.*</param>
       </targetTests>
       <outputFormats>
           <param>HTML</param>
           <param>XML</param>
       </outputFormats>
   </configuration>
</plugin>
```

5.2 Quantitative Results

The overall results demonstrate a highly robust test suite.

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
6	99%	210/212	93%	70/75	95%	70/74

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
com.abhay.configuration	1	100%	15/15	100%	1/1	100%	1/1
com.abhay.helper	3	96%	47/49	94%	29/31	97%	29/30
com.abhay.mapper	1	100%	23/23	86%	6/7	86%	6/7
com.abhay.service	1	100%	125/125	94%	34/36	94%	34/36

Report generated by PIT 1.15.2

Enhanced functionality available at arcmutate.com

Project Summary Table:

Metric	Score	Detail
Line Coverage	**99%**	210/212 Lines Covered
Mutation Coverage	**93%**	70/75 Mutants Killed
Test Strength	**95%**	70/74 (Executed Mutants Killed)

Package Breakdown:

1. **com.abhay.service**: 94% Mutation Coverage (34/36 Killed).
2. **com.abhay.helper**: 94% Mutation Coverage (29/31 Killed).
3. **com.abhay.mapper**: 86% Mutation Coverage (6/7 Killed).
4. **com.abhay.configuration**: 100% Mutation Coverage (1/1 Killed).

5.3 Detailed Analysis: Killed Mutants

We analyzed specific mutants to understand *why* they were killed. This confirms the value of our tests.

Case 1: Security Critical - Password Encryption

- **Location:** CustomerService.java, Line 47
- **Mutation:** removed call to com/abhay/entity/Customer::setPassword
- **Status: KILLED**
- **Analysis:** The mutant deleted the line customer.setPassword(...). This mimics a developer forgetting to encrypt the password before saving.
- **Killing Test:** createCustomer_Success
- **Reason:** The test explicitly asserted: assertEquals(encodedPassword, savedCustomer.getPassword()). Since the mutant caused the saved password to be Raw (or null) instead of Encoded, the assertion failed.

Case 2: Financial Logic - Remaining Balance

- **Location:** CustomerService.java, Line 260
- **Mutation:** removed call to com/abhay/dto/BillWithPayments::setPaid
- **Status: KILLED**
- **Analysis:** The mutant prevented the setPaid method from being called when calculating bill history. This would result in the UI showing "0 Paid" for everyone.
- **Killing Test:** getBillsWithPayments_MixedScenarios
- **Reason:** The test verified the exact BigDecimal value of the paid amount. assertEquals(new BigDecimal("2000.00"), bill1.getPaid()). The mutant resulted in null or 0, causing the test to fail.

Case 3: Authentication - Login Logic

- **Location:** CustomerService.java, Line 102
- **Mutation:** removed conditional - replaced equality check with false
- **Status: KILLED**
- **Analysis:** The code if (!encryptionService.validates(...)) protects the login. The mutant removed this check, effectively allowing any password to work.
- **Killing Test:** login_InvalidPassword
- **Reason:** The test sent a wrong password and expected a 401 UNAUTHORIZED response. The mutant (which bypassed the check) returned 200 OK. The test correctly flagged this as a failure.

5.4 Detailed Analysis: Survived Mutants

Mutants that survived represent areas for improvement.

Survivor 1: Mapper Conditional

- **Location:** CustomerMapper.java, Line 24
- **Mutation:** removed conditional - replaced equality check with false
- **Status: SURVIVED**
- **Analysis:** This conditional likely checks if an optional field in the DTO is null before mapping.
- **Reason for Survival:** Our test suite CustomerMapperTest only included "Happy Path" scenarios where all fields were present. We lacked a test case where optional fields were missing.
- **Improvement Plan:** Add a test testToCustomer_WithNullFields() to ensure the mapper handles nulls gracefully without crashing.

Survivor 2: Lambda Exception Handling

- **Location:** CustomerService.java, Line 298
- **Mutation:** replaced return value with null (in lambda)
- **Status: SURVIVED**
- **Analysis:** Inside orElseThrow(() -> new CustomerNotFoundException(...)), the mutant returned null.
- **Reason for Survival:** The Optional.orElseThrow method throws a NullPointerException if the supplier returns null. If our test case was catching a generic Exception class instead of the specific CustomerNotFoundException, it might have treated the NPE as a "correctly caught exception."
- **Improvement Plan:** Refactor tests to use assertThrows(CustomerNotFoundException.class, ...) to ensure exact exception type matching.

Pit Test Coverage Report

Package Summary

com.abhay.configuration

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
1	100%	15/15	100%	1/1	100%	1/1

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
SecurityConfig.java	100%	15/15	100%	1/1	100%	1/1

Report generated by PIT 1.15.2

1

SecurityConfig.java

```
1  package com.abhay.configuration;
2
3  import com.abhay.helper.RequestInterceptor;
4  import lombok.RequiredArgsConstructor;
5  import org.springframework.context.annotation.Bean;
6  import org.springframework.context.annotation.Configuration;
7  import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
8  import org.springframework.security.crypto.password.PasswordEncoder;
9  import org.springframework.web.servlet.config.annotation.CorsRegistry;
10 import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
11 import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
12
13 @Configuration
14 // Removed @RequiredArgsConstructor to manually define the constructor below
15 public class SecurityConfig   implements WebMvcConfigurer  {
16
17     private final RequestInterceptor requestInterceptor;
18
19     // FIX: Manually defining the required constructor to initialize the final field.
20     public SecurityConfig(RequestInterceptor requestInterceptor) {
21         this.requestInterceptor = requestInterceptor;
22     }
23
24     @Override
25     public void addInterceptors(InterceptorRegistry registry) {
26
27         registry.addInterceptor(requestInterceptor)
28                 .addPathPatterns("/**")
29                 .excludePathPatterns("/api/auth/login","/api/students/register", "/api/students/register/");
30     }
31
32     @Override
33     public void addCorsMappings(CorsRegistry registry) {
34         registry.addMapping("/**")
35                 .allowedOrigins("http://localhost:3000") // Frontend URL
36                 .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS")
37                 .allowedHeaders("Content-Type", "Authorization")
38                 .exposedHeaders("Authorization") // Expose Authorization header for frontend
39                 .allowCredentials(true); // Allow credentials to be included
40     }
41
42     @Bean
43     public PasswordEncoder passwordEncoder() {
44 1       return new BCryptPasswordEncoder();
45     }
46 }
47
48
49
50
51
52
53
54 //package com.abhay.configuration;
55 //
56 //import com.abhay.helper.RequestInterceptor;
57 //import lombok.RequiredArgsConstructor;
58 //import org.springframework.context.annotation.Bean;
59 //import org.springframework.context.annotation.Configuration;
60 //import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
61 //import org.springframework.security.crypto.password.PasswordEncoder;
62 //import org.springframework.web.servlet.config.annotation.CorsRegistry;
63 //import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
64 //import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
65 //
```

Mutations

44 1. replaced return value with null for com/abhay/configuration/SecurityConfig::passwordEncoder → KILLED

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_ORDER_ELSE
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- com.abhay.configuration.SecurityConfigTest.[engine:junit-jupiter]/[class:com.abhay.configuration.SecurityConfigTest]/[method:testPasswordEncoder_ReturnsBCryptPasswordEncoder()] (447 ms)

Pit Test Coverage Report

Package Summary

com.abhay.helper

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
3	96%	47/49	94%	29/31	97%	29/30

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
EncryptionService.java	100%	3/3	100%	3/3	100%	3/3
JWTHelper.java	91%	21/23	90%	9/10	100%	9/9
RequestInterceptor.java	100%	23/23	94%	17/18	94%	17/18

Report generated by PIT 1.15.2

2

EncryptionService.java

```
1  package com.abhay.helper;
2
3  import lombok.RequiredArgsConstructor;
4  import org.springframework.security.crypto.password.PasswordEncoder;
5  import org.springframework.stereotype.Service;
6
7  @Service
8  @RequiredArgsConstructor
9  public class EncryptionService {
10     private final PasswordEncoder passwordEncoder;
11
12     public String encode(String password) {
13 1       return passwordEncoder.encode(password);
14     }
15
16     public boolean validates(String password, String encodedPassword) {
17 2       return passwordEncoder.matches(password, encodedPassword);
18     }
19 }
```

Mutations

```
13 1. replaced return value with "" for com/abhay/helper/EncryptionService::encode → KILLED
17 1. replaced boolean return with false for com/abhay/helper/EncryptionService::validates → KILLED
   2. replaced boolean return with true for com/abhay/helper/EncryptionService::validates → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_ORDER_ELSE
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- com.abhay.helper.EncryptionServiceTest.[engine:junit-jupiter]/[class:com.abhay.helper.EncryptionServiceTest]/[method:testValidates_Failure()] (0 ms)
- com.abhay.helper.EncryptionServiceTest.[engine:junit-jupiter]/[class:com.abhay.helper.EncryptionServiceTest]/[method:testValidates_Success()] (19 ms)
- com.abhay.helper.EncryptionServiceTest.[engine:junit-jupiter]/[class:com.abhay.helper.EncryptionServiceTest]/[method:testEncode()] (1 ms)

Report generated by PIT 1.15.2

JWTHelper.java

```
1  package com.abhay.helper;
2
3  import com.abhay.entity.Customer;
4  import io.jsonwebtoken.Claims;
5  import io.jsonwebtoken.ExpiredJwtException;
6  import io.jsonwebtoken.Jwts;
7  import io.jsonwebtoken.SignatureAlgorithm;
8  import org.springframework.stereotype.Component;
9
10 import java.util.Date;
11 import java.util.HashMap;
12 import java.util.Map;
13 import java.util.function.Function;
14
15 @Component
16 public class JWTHelper {
17     private String SECRET_KEY = "cr666N7wIV+KJ2xOQpWtcfAekL4YXd9gbnJMs8SJ9sI=";
18
19     // Extract username from the token
20     public String extractUsername(String token) {
21 1       return extractClaim(token, Claims::getSubject);
22     }
23
24     // Extract expiration date from the token
25     public Date extractExpiration(String token) {
26 1       return extractClaim(token, Claims::getExpiration);
27     }
28
29     // Extract claims
30     public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
31         final Claims claims = extractAllClaims(token);
32 1       return claimsResolver.apply(claims);
33     }
34
35     // Extract all claims
36     private Claims extractAllClaims(String token) {
37 1       return Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token).getBody();
38     }
39
40     // Generate token
41     public String generateToken(Customer customer) {
42         Map<String, Object> claims = new HashMap<>();
43 1       return createToken(claims, customer);
44     }
45
```

Mutations

```
21 1. replaced return value with "" for com/abhay/helper/JWTHelper::extractUsername → KILLED
26 1. replaced return value with null for com/abhay/helper/JWTHelper::extractExpiration → KILLED
32 1. replaced return value with null for com/abhay/helper/JWTHelper::extractClaim → KILLED
37 1. replaced return value with null for com/abhay/helper/JWTHelper::extractAllClaims → KILLED
43 1. replaced return value with "" for com/abhay/helper/JWTHelper::generateToken → KILLED
54 1. replaced return value with "" for com/abhay/helper/JWTHelper::createToken → KILLED
55 1. Replaced long addition with subtraction → KILLED
63 1. replaced Boolean return with False for com/abhay/helper/JWTHelper::validateToken → KILLED
66 1. replaced Boolean return with True for com/abhay/helper/JWTHelper::validateToken → NO_COVERAGE
69 1. replaced Boolean return with True for com/abhay/helper/JWTHelper::validateToken → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_ORDER_ELSE
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- com.abhay.helper.JWTHelperTest.[engine:junit-jupiter]/[class:com.abhay.helper.JWTHelperTest]/[method:testGenerateAndValidateToken()] (2 ms)
- com.abhay.helper.JWTHelperTest.[engine:junit-jupiter]/[class:com.abhay.helper.JWTHelperTest]/[method:testTokenExpiration()] (37 ms)
- com.abhay.helper.JWTHelperTest.[engine:junit-jupiter]/[class:com.abhay.helper.JWTHelperTest]/[method:testValidateToken_ShouldReturnFalseForTokenWithBadSignature()] (0 ms)
- com.abhay.helper.JWTHelperTest.[engine:junit-jupiter]/[class:com.abhay.helper.JWTHelperTest]/[method:testValidateToken_ShouldReturnFalseForInvalidFormat()] (57 ms)

RequestInterceptor.java

```
1  package com.abhay.helper;
2
3  import com.fasterxml.jackson.databind.ObjectMapper;
4  import jakarta.servlet.http.HttpServletRequest;
5  import jakarta.servlet.http.HttpServletResponse;
6  import lombok.RequiredArgsConstructor;
7  import org.springframework.http.HttpStatus;
8  import org.springframework.http.ResponseEntity;
9  import org.springframework.stereotype.Component;
10 import org.springframework.web.servlet.HandlerInterceptor;
11
12 import java.util.HashMap;
13 import java.util.Map;
14
15 @Component
16 @RequiredArgsConstructor
17 public class RequestInterceptor implements HandlerInterceptor {
18     private final JWTHelper jwtUtil;
19
20     @Override
21     public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
22 1       if (request.getMethod().equalsIgnoreCase("OPTIONS")) {
23 1           return true; // Allow preflight requests
24         }
25
26         String uri = request.getRequestURI();
27 2       if (uri.contains("/api/students/register") || uri.contains("/api/auth/login")) {
28 1           return true; // Skip authorization check for specific endpoints
29         }
30
31         String authorizationHeader = request.getHeader("Authorization");
32 2       if (authorizationHeader == null || !authorizationHeader.startsWith("Bearer ")) {
33 2           setResponse(response, HttpServletResponse.SC_UNAUTHORIZED, "You are not Authorized for this method");            return false;
34         }
35
36         String token = authorizationHeader.substring(7); // Extract token from "Bearer {token}"
37 //         System.out.println(token);
38
39 1       if (!jwtUtil.validateToken(token)) {
40 2           setResponse(response, HttpServletResponse.SC_UNAUTHORIZED, "Invalid or expired token");            return false;
41         }
42         String username = jwtUtil.extractUsername(token);
43
44 1       if (username == null) {
45 2           setResponse(response, HttpServletResponse.SC_UNAUTHORIZED, "Invalid or expired token");            return false;
46         }
47
48 1       return true;
49     }
50
51     private void setResponse(HttpServletResponse response, int status, String message) throws Exception {
```

Mutations

```
22 1. removed conditional - replaced equality check with false → KILLED
23 1. replaced boolean return with false for com/abhay/helper/RequestInterceptor::preHandle → KILLED
27 1. removed conditional - replaced equality check with false → KILLED
28 1. replaced boolean return with false for com/abhay/helper/RequestInterceptor::preHandle → KILLED
32 1. removed conditional - replaced equality check with false → KILLED
2. removed conditional - replaced equality check with false → KILLED
33 1. removed call to com/abhay/helper/RequestInterceptor::setResponse → KILLED
2. replaced boolean return with true for com/abhay/helper/RequestInterceptor::preHandle → KILLED
39 1. removed conditional - replaced equality check with false → SURVIVED
40 1. removed call to com/abhay/helper/RequestInterceptor::setResponse → KILLED
2. replaced boolean return with true for com/abhay/helper/RequestInterceptor::preHandle → KILLED
44 1. removed conditional - replaced equality check with false → KILLED
45 1. replaced boolean return with true for com/abhay/helper/RequestInterceptor::preHandle → KILLED
2. removed call to com/abhay/helper/RequestInterceptor::setResponse → KILLED
48 1. replaced boolean return with false for com/abhay/helper/RequestInterceptor::preHandle → KILLED
52 1. removed call to jakarta/servlet/http/HttpServletResponse::setStatus → KILLED
62 1. removed call to java/io/PrintWriter::write → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_ORDER_ELSE
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- com.abhay.helper.RequestInterceptorTest.[engine:junit-jupiter]/[class:com.abhay.helper.RequestInterceptorTest]/[method:preHandle_ShouldAllowIfTokenIsValid()] (1 ms)
- com.abhay.helper.RequestInterceptorTest.[engine:junit-jupiter]/[class:com.abhay.helper.RequestInterceptorTest]/[method:preHandle_ShouldDenyIfAuthorizationHeaderDoesNotStartWithBearer()] (1 ms)
- com.abhay.helper.RequestInterceptorTest.[engine:junit-jupiter]/[class:com.abhay.helper.RequestInterceptorTest]/[method:preHandle_ShouldDenyIfNoAuthorizationHeader()] (1 ms)
- com.abhay.helper.RequestInterceptorTest.[engine:junit-jupiter]/[class:com.abhay.helper.RequestInterceptorTest]/[method:preHandle_ShouldDenyIfUsernameIsNull()] (1 ms)
- com.abhay.helper.RequestInterceptorTest.[engine:junit-jupiter]/[class:com.abhay.helper.RequestInterceptorTest]/[method:preHandle_ShouldDenyIfTokenIsInvalid()] (73 ms)
- com.abhay.helper.RequestInterceptorTest.[engine:junit-jupiter]/[class:com.abhay.helper.RequestInterceptorTest]/[method:preHandle_ShouldAllowOptionsRequest()] (1 ms)
- com.abhay.helper.RequestInterceptorTest.[engine:junit-jupiter]/[class:com.abhay.helper.RequestInterceptorTest]/[method:preHandle_ShouldAllowRegisterPath()] (1 ms)
- com.abhay.helper.RequestInterceptorTest.[engine:junit-jupiter]/[class:com.abhay.helper.RequestInterceptorTest]/[method:preHandle_ShouldAllowLoginPath()] (1 ms)

Pit Test Coverage Report

Package Summary

com.abhay.mapper

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
1	100%	23/23	86%	6/7	86%	6/7

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
CustomerMapper.java	100%	23/23	86%	6/7	86%	6/7

Report generated by PIT 1.15.2

3

CustomerMapper.java

```
1  package com.abhay.mapper;
2
3  import com.abhay.dto.CustomerRequest;
4  import com.abhay.dto.CustomerResponse;
5  import com.abhay.dto.FeesPayment;
6  import com.abhay.entity.Bill;
7  import com.abhay.entity.Customer;
8  import com.abhay.entity.StudentPayment;
9  import org.springframework.stereotype.Service;
10
11 import java.time.LocalDate;
12
13 @Service
14 public class CustomerMapper {
15     public Customer toCustomer(CustomerRequest request) {
16 1       return Customer.builder()
17                 .firstName(request.firstName())
18                 .lastName(request.lastName())
19                 .email(request.email())
20                 .password(request.password())
21                 .rollNumber(request.rollNumber())
22 1               .cgpa(request.cgpa() != null ? request.cgpa() : 0.0f)
23 1               .graduationYear(request.graduationYear() != null ? request.graduationYear() : 1900)
24 1               .totalCredits(request.totalCredits() != null ? request.totalCredits() : 0.0f)
25                 .domain(request.domain())
26 1               .placementId(request.placementId() != null ? request.placementId() : 0)
27                 .photographPath(request.photographPath())
28                 .specialisation(request.specialisation())
29                 .build();
30
31     }
32
33     public CustomerResponse toCustomerResponse(Customer customer) {
34 1       return new CustomerResponse(customer.getFirstName(), customer.getLastName(), customer.getEmail());
35     }
36
37     public StudentPayment toEntity(FeesPayment feesPayment, Customer student, Bill bill) {
38 1       return StudentPayment.builder()
39                 .amount(feesPayment.amount())
40                 .student(student)
41                 .description(feesPayment.description())
42                 .bill(bill)
43                 .paymentDate(LocalDate.now())  // Assuming the payment date is the current date
44                 .build();
45     }
46 }
47
48
49
50
51
52
```

Mutations

```
16 1. replaced return value with null for com/abhay/mapper/CustomerMapper::toCustomer → KILLED
22 1. removed conditional - replaced equality check with false → KILLED
23 1. removed conditional - replaced equality check with false → KILLED
24 1. removed conditional - replaced equality check with false → SURVIVED
26 1. removed conditional - replaced equality check with false → KILLED
34 1. replaced return value with null for com/abhay/mapper/CustomerMapper::toCustomerResponse → KILLED
38 1. replaced return value with null for com/abhay/mapper/CustomerMapper::toEntity → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_ORDER_ELSE
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- com.abhay.mapper.CustomerMapperTest.[engine:junit-jupiter]/[class:com.abhay.mapper.CustomerMapperTest]/[method:testToCustomer_OptionalFieldsMissing()] (1 ms)
- com.abhay.mapper.CustomerMapperTest.[engine:junit-jupiter]/[class:com.abhay.mapper.CustomerMapperTest]/[method:testToCustomer_AllFieldsPresent()] (0 ms)
- com.abhay.mapper.CustomerMapperTest.[engine:junit-jupiter]/[class:com.abhay.mapper.CustomerMapperTest]/[method:testToEntity_FeesPaymentToStudentPayment()] (1 ms)
- com.abhay.mapper.CustomerMapperTest.[engine:junit-jupiter]/[class:com.abhay.mapper.CustomerMapperTest]/[method:testToCustomerResponse_ShouldMapCustomer()] (0 ms)

Pit Test Coverage Report

Package Summary

com.abhay.service

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
1	100%	125/125	94%	34/36	94%	34/36

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
CustomerService.java	100%	125/125	94%	34/36	94%	34/36

Report generated by PIT 1.15.2

4

CustomerService.java

```
1  package com.abhay.service;
2
3  import com.abhay.dto.*;
4  import com.abhay.entity.Bill;
5  import com.abhay.entity.Customer;
6  import com.abhay.entity.StudentPayment;
7  import com.abhay.exception.CustomerNotFoundException;
8  import com.abhay.exception.BillNotFoundException;
9  import com.abhay.helper.EncryptionService;
10 import com.abhay.helper.JWTHelper;
11 import com.abhay.mapper.CustomerMapper;
12 import com.abhay.repo.BillRepo;
13 import com.abhay.repo.CustomerRepo;
14 import com.abhay.repo.StudentPaymentRepo;
15 import lombok.RequiredArgsConstructor;
16 import org.springframework.dao.DataIntegrityViolationException;
17 import org.springframework.http.HttpStatus;
18 import org.springframework.http.ResponseEntity;
19 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
20 import org.springframework.stereotype.Service;
21
22 import java.math.BigDecimal;
23 import java.math.BigInteger;
24 import java.security.PublicKey;
25 import java.time.LocalDate;
26 import java.util.*;
27
28 import static java.lang.String.format;
29
30 @Service
31 @RequiredArgsConstructor
32 public class CustomerService {
33     private final CustomerRepo customerRepo;
34     private final CustomerMapper customerMapper;
35     private final EncryptionService encryptionService;
36     private final BillRepo billRepo;
37     private final StudentPaymentRepo studentPaymentRepo;
38     /// added
39 //    private BCryptPasswordEncoder passwordEncoder;
40     private final JWTHelper jwtHelper;
41     public ResponseEntity<Object> createCustomer(CustomerRequest request) {
42         try {
43             // Convert the request to a Customer entity
44             Customer customer = customerMapper.toCustomer(request);
45
46             // Encrypt the password
47 1           customer.setPassword(encryptionService.encode(customer.getPassword()));
48
49             // Save the customer to the repository
50             customerRepo.save(customer);
51
52             // Return success response
53 1           return ResponseEntity.status(HttpStatus.CREATED)
54                     .body(Map.of(
55                             "status", "success",
56                             "message", "Customer Created Successfully",
57                             "customerId", customer.getEmail()
58                     ));
59         } catch (DataIntegrityViolationException e) {
60             // Handle database constraint violations, such as duplicate email
61 //            System.out.println(e.getMessage());
62 1           return ResponseEntity.status(HttpStatus.CONFLICT)
63                     .body(Map.of(
64                             "status", "error",
65                             "message", "Email ID or RollNo already exists"
66                     ));
67         } catch (IllegalArgumentException e) {
68             // Handle invalid arguments, such as null or improperly formatted fields
69 1           return ResponseEntity.status(HttpStatus.BAD_REQUEST)
70                     .body(Map.of(
71                             "status", "error",
72                             "message", "Invalid input: " + e.getMessage()
73                     ));
74         } catch (Exception e) {
```

Mutations

47	1. removed call to com/abhay/entity/Customer::setPassword → KILLED
53	1. replaced return value with null for com/abhay/service/CustomerService::createCustomer → KILLED
62	1. replaced return value with null for com/abhay/service/CustomerService::createCustomer → KILLED
69	1. replaced return value with null for com/abhay/service/CustomerService::createCustomer → KILLED
76	1. replaced return value with null for com/abhay/service/CustomerService::createCustomer → KILLED
85	1. replaced return value with null for com/abhay/service/CustomerService::getCustomer → KILLED
86	1. replaced return value with null for com/abhay/service/CustomerService::lambda$getCustomer$0 → KILLED
93	1. replaced return value with null for com/abhay/service/CustomerService::retrieveCustomer → KILLED
102	1. removed conditional - replaced equality check with false → KILLED
103	1. replaced return value with null for com/abhay/service/CustomerService::login → KILLED
115	1. replaced return value with null for com/abhay/service/CustomerService::login → KILLED
122	1. replaced return value with null for com/abhay/service/CustomerService::login → KILLED
147	1. replaced return value with Collections.emptyList for com/abhay/service/CustomerService::GetunpaidBills → KILLED
169	1. replaced return value with Collections.emptyList for com/abhay/service/CustomerService::GetPaidBills → KILLED
252	1. removed conditional - replaced equality check with false → KILLED
253	1. removed call to com/abhay/dto/BillWithPayments::setBillId → KILLED
254	1. removed call to com/abhay/dto/BillWithPayments::setDescription → KILLED
255	1. removed call to com/abhay/dto/BillWithPayments::setAmount → KILLED
256	1. removed call to com/abhay/dto/BillWithPayments::setBillDate → KILLED
257	1. removed call to com/abhay/dto/BillWithPayments::setDeadline → KILLED
259	1. removed call to com/abhay/dto/BillWithPayments::setRemaining → KILLED
260	1. removed call to com/abhay/dto/BillWithPayments::setPaid → KILLED
264	1. removed conditional - replaced equality check with false → KILLED
266	1. removed call to com/abhay/dto/BillWithPayments$PaymentDetail::setPaymentId → KILLED
267	1. removed call to com/abhay/dto/BillWithPayments$PaymentDetail::setAmount → KILLED
268	1. removed call to com/abhay/dto/BillWithPayments$PaymentDetail::setPaymentDate → KILLED
269	1. removed call to com/abhay/dto/BillWithPayments$PaymentDetail::setDescription → KILLED
281	1. replaced return value with Collections.emptyList for com/abhay/service/CustomerService::getBillsWithPayments → KILLED
288	1. removed conditional - replaced equality check with false → KILLED
291	1. replaced return value with null for com/abhay/service/CustomerService::generateResponse → KILLED
298	1. replaced return value with null for com/abhay/service/CustomerService::lambda$payFees$1 → SURVIVED
302	1. replaced return value with null for com/abhay/service/CustomerService::lambda$payFees$2 → SURVIVED
314	1. replaced return value with null for com/abhay/service/CustomerService::payFees → KILLED
322	1. replaced return value with null for com/abhay/service/CustomerService::payFees → KILLED
329	1. replaced return value with null for com/abhay/service/CustomerService::payFees → KILLED
336	1. replaced return value with null for com/abhay/service/CustomerService::payFees → KILLED

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NULL_RETURNS
- PRIMITIVE_RETURNS
- REMOVE_CONDITIONALS_EQUAL_ELSE
- REMOVE_CONDITIONALS_ORDER_ELSE
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:getBillsWithPayments_MixedScenarios()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:retrieveCustomer_Success()] (2 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:payFees_Success()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:login_GenericException()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:login_UserNotFound()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:getPaidBills_MappingTest()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:getUnpaidBills_MappingTest()] (65 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:createCustomer_DataIntegrityViolation()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:login_InvalidPassword()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:login_Success()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:getCustomer_Success()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:createCustomer_Success()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:generateResponse_WithoutToken()] (0 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:generateResponse_WithToken()] (5 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:createCustomer_GenericException()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:payFees_CustomerNotFound()] (0 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:payFees_GenericException()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:payFees_BillNotFound()] (2 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:payFees_BadRequest()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:createCustomer_IllegalArgumentException()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:payFees_Conflict()] (1 ms)
- com.abhay.service.CustomerServiceTest.[engine:junit-jupiter]/[class:com.abhay.service.CustomerServiceTest]/[method:getCustomer_NotFound()] (1 ms)

CHAPTER 6:Unit and Integration Operators

1. Unit Level Operators

File link: BackEnd/target/pit-reports/com.abhay.helper/JWTHelper.java.html

- **MathMutator:** line 55

```
43 1. replaced return value with "" for com/abhay/helper/JWTHelper::generateToken → KILLED
54 1. replaced return value with "" for com/abhay/helper/JWTHelper::createToken → KILLED
55 1. Replaced long addition with subtraction → KILLED
63 1. replaced Boolean return with False for com/abhay/helper/JWTHelper::validateToken → KILLED
```

- **EmptyObjectReturnValsMutator:Line 54** or **Line 21**.

```
21 1. replaced return value with "" for com/abhay/helper/JWTHelper::extractUsername → KILLED
26 1. replaced return value with null for com/abhay/helper/JWTHelper::extractExpiration → KILLED
32 1. replaced return value with null for com/abhay/helper/JWTHelper::extractClaim → KILLED
37 1. replaced return value with null for com/abhay/helper/JWTHelper::extractAllClaims → KILLED
43 1. replaced return value with "" for com/abhay/helper/JWTHelper::generateToken → KILLED
54 1. replaced return value with "" for com/abhay/helper/JWTHelper::createToken → KILLED
```

- **NullReturnValsMutator: Line 37**

```
32 1. replaced return value with null for com/abhay/helper/JWTHelper::extractClaim → KILLED
37 1. replaced return value with null for com/abhay/helper/JWTHelper::extractAllClaims → KILLED
43 1. replaced return value with "" for com/abhay/helper/JWTHelper::generateToken → KILLED
```

File link:
BackEnd/target/pit-reports/com.abhay.helper/EncryptionService.java.html

- **BooleanFalseReturnValsMutator: Line 17**

```
17 1. replaced boolean return with false for com/abhay/helper/EncryptionService::validates → KILLED
   2. replaced boolean return with true for com/abhay/helper/EncryptionService::validates → KILLED
```

- **BooleanTrueReturnValsMutator: Line 17**

```
17 1. replaced boolean return with false for com/abhay/helper/EncryptionService::validates → KILLED
   2. replaced boolean return with true for com/abhay/helper/EncryptionService::validates → KILLED
```

2. Integration Level Operators

File link:
BackEnd/target/pit-reports/com.abhay.service/CustomerService.java.html

- **VoidMethodCallMutator: Line 47**

```
47  1. removed call to com/abhay/entity/Customer::setPassword → KILLED
53  1. replaced return value with null for com/abhay/service/CustomerService::createCustomer → KILLED
```

- **NullReturnValsMutator: Line 53** or **Line 62**.

```
53  1. replaced return value with null for com/abhay/service/CustomerService::createCustomer → KILLED
62  1. replaced return value with null for com/abhay/service/CustomerService::createCustomer → KILLED
```

- **EmptyObjectReturnValsMutator: Line 147** or **Line 169**

```
147  1. replaced return value with Collections.emptyList for com/abhay/service/CustomerService::GetunpaidBills → KILLED
169  1. replaced return value with Collections.emptyList for com/abhay/service/CustomerService::GetPaidBills → KILLED
252  1. removed conditional - replaced equality check with false → KILLED
```

File link :
BackEnd/target/pit-reports/com.abhay.helper/RequestInterceptor.java.html

- **RemoveConditionalMutator_EQUAL_ELSE: Line 22**, **27**, or **32**

```
22  1. removed conditional - replaced equality check with false → KILLED
23  1. replaced boolean return with false for com/abhay/helper/RequestInterceptor::preHandle → KILLED
27  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with false → KILLED
28  1. replaced boolean return with false for com/abhay/helper/RequestInterceptor::preHandle → KILLED
32  1. removed conditional - replaced equality check with false → KILLED
    2. removed conditional - replaced equality check with false → KILLED
```

CHAPTER 7: How to run

Command

1. Clean Command

`mvnw.cmd clean` / `./mvnw clean`

- **What it does:** This command deletes the `target/` directory in your project folder.
- **Why use it:** The `target/` folder holds all the temporary build artifacts (compiled `.class` files, JARs, test reports). Running `clean` ensures you are starting your build from scratch, removing any old or stale files that might cause errors.

2. Install Command

`mvnw.cmd install` / `./mvnw install`

- **What it does:** This triggers a full build lifecycle. It will sequentially:
    1. **Compile** your source code (`src/main/java`).
    2. **Run your unit tests** (like `CustomerServiceTest.java`).
    3. **Package** the application into a JAR file (e.g., `Mini-Project-0.0.1-SNAPSHOT.jar`).
    4. **Install** that JAR into your local Maven repository (usually located at `~/.m2/repository`).
- **Why use it:** This is the standard command to verify that your entire project builds correctly and passes all standard unit tests. If this fails, your code has compilation errors or standard bugs.

3. Test Command

`mvnw.cmd test` / `./mvnw test` *(Note: You had a typo `tes` in your Windows command; it must be `test`)*

- **What it does:** This runs **only** the unit tests located in `src/test/java`.
- **Why use it:** Use this when you are developing and want to quickly check if your changes broke any existing logic. It is faster than `install` because it doesn't package the JAR or install it.
- **Context:** It runs the tests defined in files like `CustomerServiceTest.java` using the JUnit 5 engine configured in your `pom.xml`.

4. Mutation Coverage Command

```
./mvnw org.pitest:pitest-maven:mutationCoverage
```

- **What it does:** This runs the **PIT Mutation Testing** tool.
- **How it works:**
    1. It reads the configuration from your `pom.xml` (where you specified target classes like `com.abhay.service.*`).
    2. It creates "mutants" (modified versions of your code) by introducing small bugs (like changing `+` to `-` or removing function calls).
    3. It runs your unit tests against these mutants.
    4. It reports whether your tests "Killed" the mutant (test failed, which is good) or if the mutant "Survived" (test passed, which is bad).
- **Output:** It generates the HTML report you see in `BackEnd/target/pit-reports/index.html`, which tells you your mutation coverage score.

CHAPTER 8: CONCLUSION & FUTURE SCOPE

8.1 Conclusion

The implementation of Mutation Testing on the **Student Billing System** has been a highly effective quality assurance exercise.

1. **Validation of Security:** We mathematically proved that our tests define a strict boundary for authentication and encryption. No password bypass or encryption skip can occur without breaking the build.
2. **Validation of Finances:** The "Math Mutators" confirmed that our billing calculations are precise and verified.
3. **High Assurance:** With a **93% Mutation Score**, the system is well above the typical industry standard (often 60-75%), indicating a mature and robust test suite.

8.2 Future Scope

- **Addressing Survivors:** We plan to implement the specific test cases identified in Section 5.4 to reach 100% mutation coverage.
- **Integration Testing:** Applying PIT to integration tests (where the real database is used) would be the next step to verify SQL queries and constraints.

REFERENCES

1. PIT Mutation Testing Documentation. *http://pitest.org/*
2. "Introduction to Software Testing", Paul Ammann and Jeff Offutt.
3. Spring Boot Documentation. *https://spring.io/projects/spring-boot*
4. JUnit 5 User Guide. *https://junit.org/junit5/*
5. *JUnit Testing in Spring Boot | @Test @ParameterizedTest @CsvSource @Argumen…*
6. *JUnit to Mutation Testing | PIT - Java Library | Mutation Testing | PIT + Spring Boot -…*
7. *Unit Testing in Spring Boot with JUnit 5 and Mockito | Part 1*

THANK YOU