Your project Logo here if any

CS673 Software Engineering
Team 3 - EMS
Software Test Document

Team Member	Role(s)	Signature Date
Jianle Xie	Team Leader	*Jianle Xie* 9/9/2023
Yuxuan Wang	QA leader	*Yuxuan Wang* 9/9/2023
Abigya Devkota	Design and Implementation Leader	*Abigya Devkota* 9/9/2023
Jingjing Tang	Configuration Leader	*Jingjing Tang* 9/10/2023
HaoLun Li	Requirement Leader	*HaoLun Li* 9/11/2023
WeiHao Mai	Security Leader	*Weihao Mai* 9/11/2023
Zhiwei Lin	Backup Leader	*Zhiwei Lin* 9/11/2023

Revision history

Version	**Author**	**Date**	**Change**
v2	**Weihao**	**10/9/23**	**Unit testing**
v3	**Jianle Xie**	**10/09/23**	**Frontend Testing** **Acceptance Testing** **Manual Testing**

● Testing Summary

In this section, you will summarize what was tested, who is involved in testing, testing techniques used, and testing result. You may have the following tests

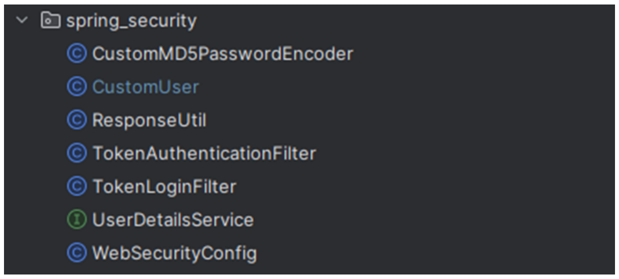
○ Unit Testing
○ Integration testing
○ System Testing
○ Acceptance Testing
○ Regression Testing

Unit Testing

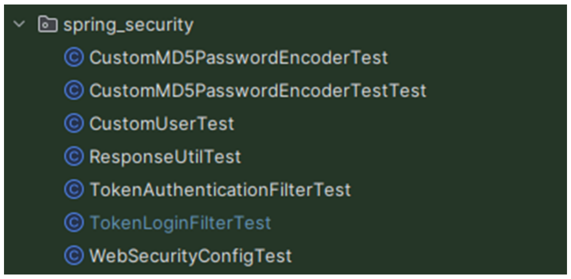
Unit testing is the process of isolating a section of code and then testing that the internals of that section work as you expect.

Mock code structure: The following is an introduction to my code structure, this unit test mainly tests the security part, for the security of each class file, write corresponding

For example, the user's class: In the CustomUser.java file, write CustomUserTest.java to test the class.

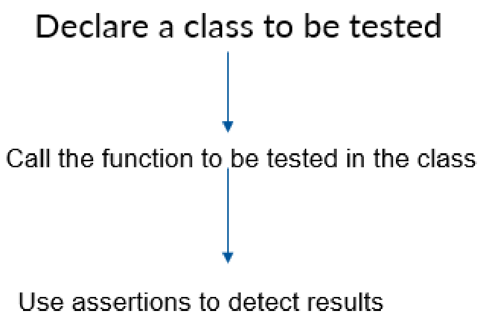


```
spring_security
    CustomMD5PasswordEncoder
    CustomUser
    ResponseUtil
    TokenAuthenticationFilter
    TokenLoginFilter
    UserDetailsService
    WebSecurityConfig
```



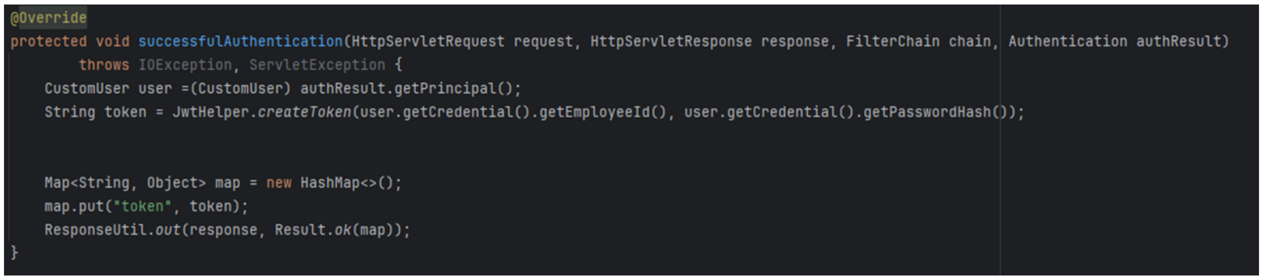
```
spring_security
    CustomMD5PasswordEncoderTest
    CustomMD5PasswordEncoderTestTest
    CustomUserTest
    ResponseUtilTest
    TokenAuthenticationFilterTest
    TokenLoginFilterTest
    WebSecurityConfigTest
```

Test flow:

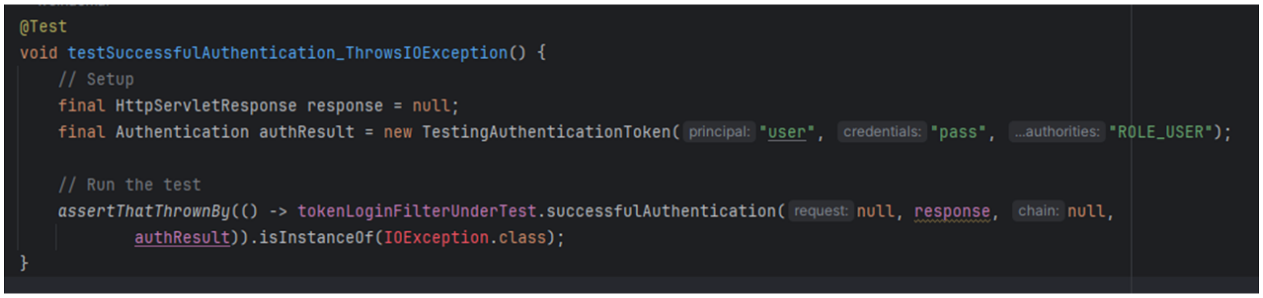

Declare a class to be tested
Call the function to be tested in the class
Use assertions to detect results

For each public function in each class, write the function to test as follows: First, declare a test class, and then in the Class calls the function to be tested, and finally uses the assertion as the test result, when the condition is true, the program will continue to execute; When the conditions are false, The JVM will throw an AssertionError.

Mock code example:



```
@Override
protected void successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain, Authentication authResult)
        throws IOException, ServletException {
    CustomUser user =(CustomUser) authResult.getPrincipal();
    String token = JwtHelper.createToken(user.getCredential().getEmployeeId(), user.getCredential().getPasswordHash());


    Map<String, Object> map = new HashMap<>();
    map.put("token", token);
    ResponseUtil.out(response, Result.ok(map));
}
```



```
@Test
void testSuccessfulAuthentication_ThrowsIOException() {
    // Setup
    final HttpServletResponse response = null;
    final Authentication authResult = new TestingAuthenticationToken( principal: "user",  credentials: "pass",  ...authorities: "ROLE_USER");

    // Run the test
    assertThatThrownBy(() -> tokenLoginFilterUnderTest.successfulAuthentication( request: null, response,  chain: null,
            authResult)).isInstanceOf(IOException.class);
}
```

Let's take a look at a code example: there is a function called successfulAuthentication, the method (pictured above)
The purpose is to generate a JWT token after the user successfully authenticates and return it to the client as part of the response.
I wrote the test code (the image below), which consists of three steps: Setup, Run the test, and verify the results. We simulate a successful authentication where the user's username is "user" and password value is "pass" and the role is "ROLE_USER". In this test, we used an empty HttpServletResponse object and applied it passed to successfulAuthentication method.
Finally, we verified the results. When validating the result, if no exception is thrown
Consider the test passed.

● Manual Testing Report

In this section, you will give a detailed description of each manual test case performed and the result. If this is a previous You shall list what are existing tests developed in the previous semester and what are new tests developed currently.

Here is a sample template that can be used for each test case. For system tests or acceptance tests, you may also include some screenshots.

- Test case ID, name
- New or old:
- Test items: (what do you test)
- Test priority (high/medium/low)
- Dependencies (to other test case/requirement if any):
- Preconditions: (if any)
- input data:
- Test steps:
- Postconditions:
- Expected output:
- Actual output:
- Pass or Fail:
- Bug id/link: (this should link to your github issue id)
- Additional notes

See sheet:

Manual Testing

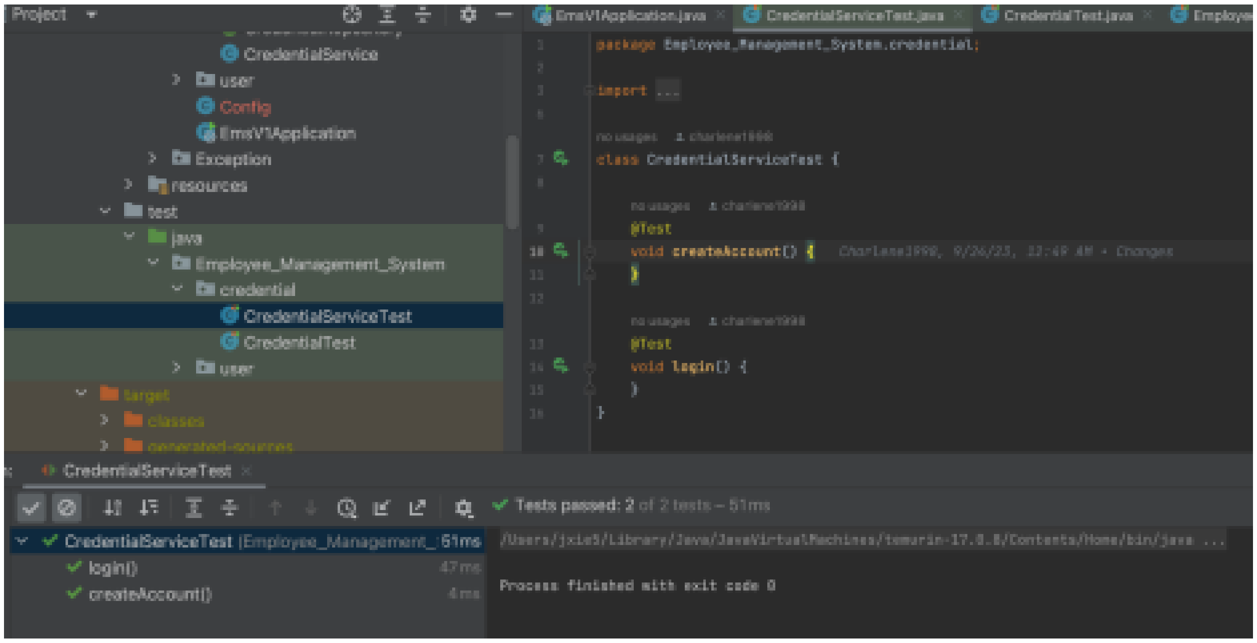
● Pass or Fail: pass ● Automated Testing Report

Describe briefly the automated testing you have done, including where the test code resides in your code repository, what test frameworks are used, and the screen shots or generated testing report.

System testing (Backend-testing)

These tests reside in project-team-3/code/Employee_management_system/src/test/java/Employee_Management_system according to Intellij and under the database-frontend-backend branch. There are currently 2 folders of tests, credential and user.

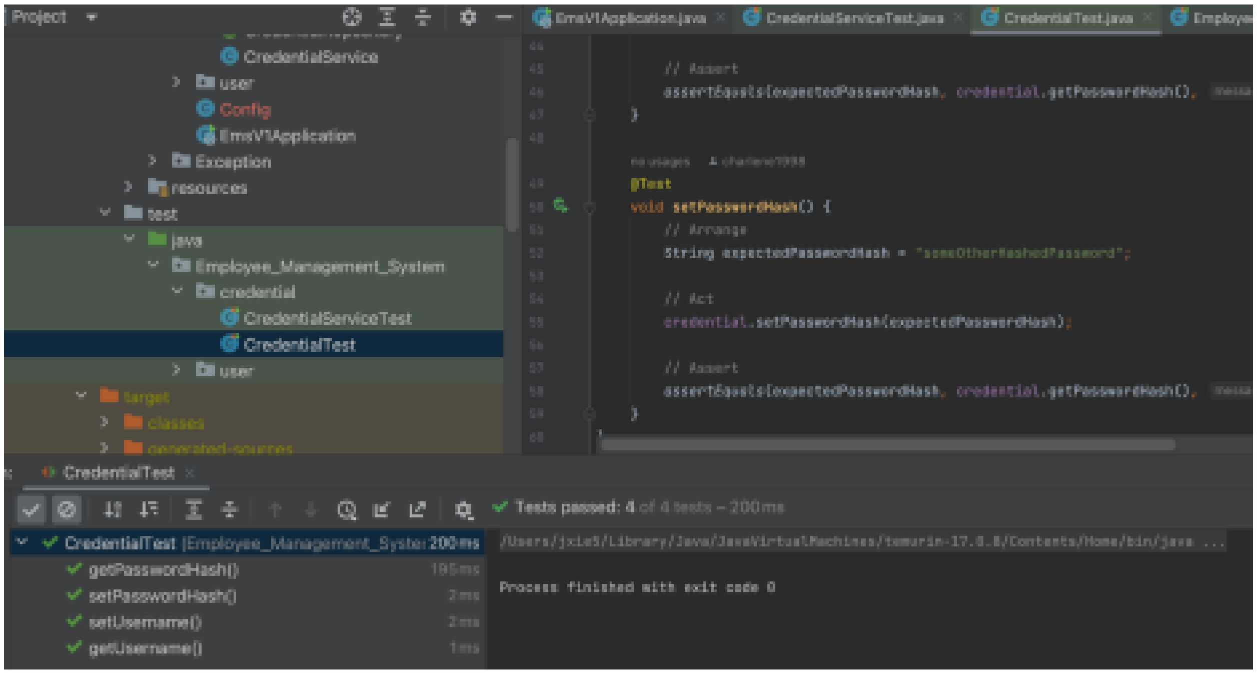
CredentialServiceTest: We are testing the methods of the CredentialService entity.



```
package Employee_Management_System.credential;

import ...

no usages  charlene1998
class CredentialServiceTest {

    no usages  charlene1998
    @Test
    void createAccount() {   Charlene1998, 9/26/23, 12:49 AM • Changes
    }

    no usages  charlene1998
    @Test
    void login() {
    }
}
```

```
Tests passed: 2 of 2 tests – 51ms

CredentialServiceTest (Employee_Management_ 51ms
  login()          47ms
  createAccount()   4ms

/Users/jxie5/Library/Java/JavaVirtualMachines/temurin-17.0.8/Contents/Home/bin/java ...

Process finished with exit code 0
```

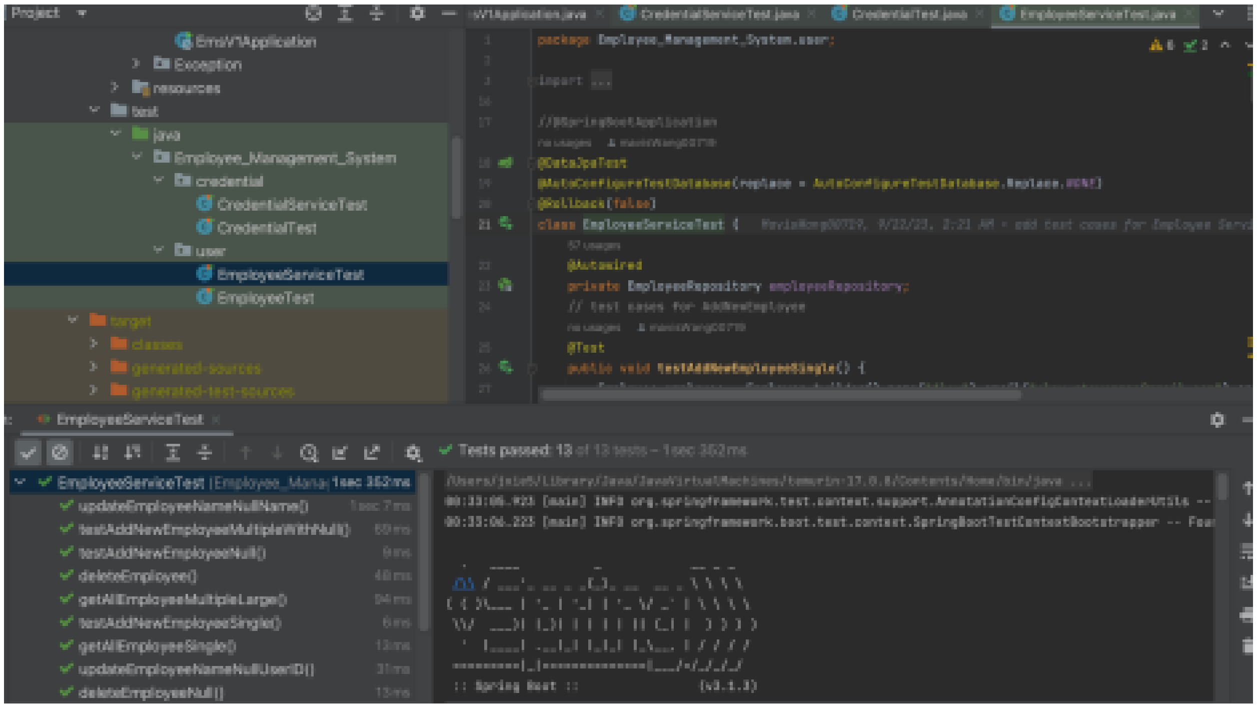
CredentialTest: Testing Credential entity



```
        // Assert
        assertEquals(expectedPasswordHash, credential.getPasswordHash(),
    }

    no usages  charlene1998
    @Test
    void setPasswordHash() {
        // Arrange
        String expectedPasswordHash = "someOtherHashedPassword";

        // Act
        credential.setPasswordHash(expectedPasswordHash);

        // Assert
        assertEquals(expectedPasswordHash, credential.getPasswordHash(),
    }
}
```

```
Tests passed: 4 of 4 tests – 200ms

CredentialTest (Employee_Management_Syster 200ms
  getPasswordHash()  195ms
  setPasswordHash()    2ms
  setUsername()        2ms
  getUsername()        1ms

/Users/jxie5/Library/Java/JavaVirtualMachines/temurin-17.0.8/Contents/Home/bin/java ...

Process finished with exit code 0
```

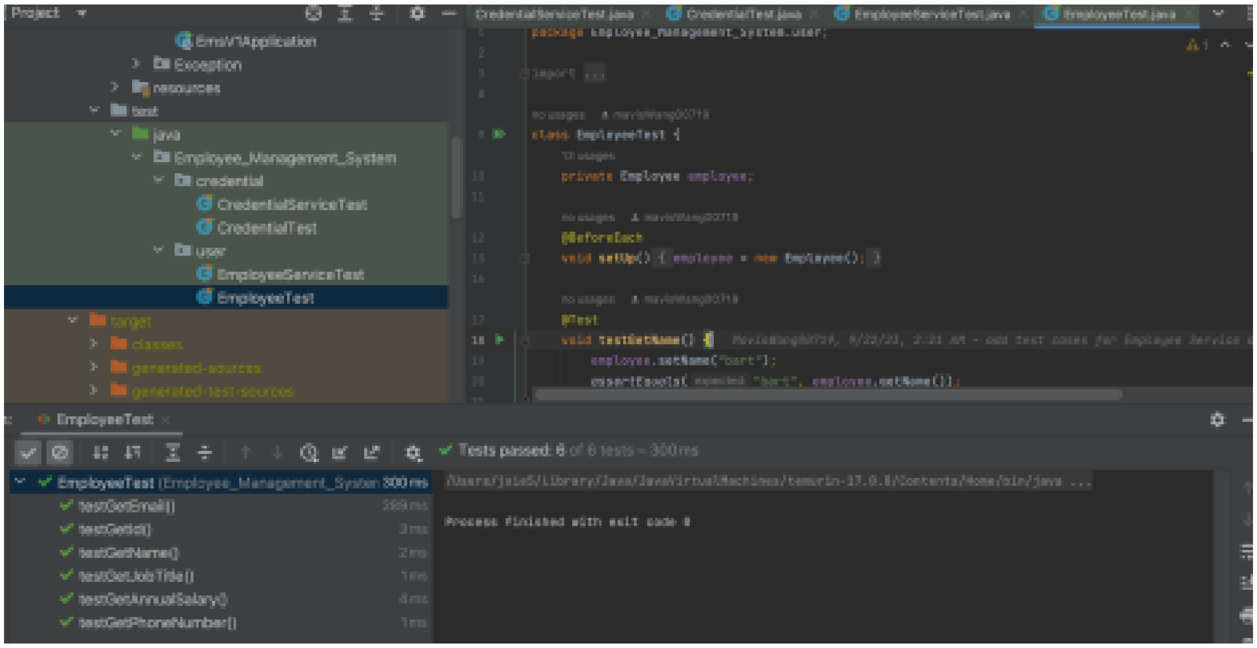
EmployeeServiceTest: Testing for SpringBoot initialization and EmployeeSerivce class entity methods.



```
package Employee_Management_System.user;

import ...

//@SpringBootApplication
@DataJpaTest
@AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
@Rollback(false)
class EmployeeServiceTest {
    @Autowired
    private EmployeeRepository employeeRepository;
    // test cases for AddNewEmployee
    @Test
    public void testAddNewEmployeeSingle() {
```

```
Tests passed: 13 of 13 tests – 1 sec 352 ms

EmployeeServiceTest (Employee_Mana  1 sec 352 ms
  updateEmployeeNameNullName()             1 sec 7 ms
  testAddNewEmployeeMultipleWithNull()     60 ms
  testAddNewEmployeeNull()                 9 ms
  deleteEmployee()                         48 ms
  getAllEmployeeMultipleLarge()            94 ms
  testAddNewEmployeeSingle()               6 ms
  getAllEmployeeSingle()                   13 ms
  updateEmployeeNameNullUserID()           31 ms
  deleteEmployeeNull()                     13 ms

/Users/jxie5/Library/Java/JavaVirtualMachines/temurin-17.0.8/Contents/Home/bin/java ...
00:33:05.923 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils --
00:33:06.223 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper -- Fou

 :: Spring Boot ::                (v3.1.3)
```

EmployeeTest: Testing for Employee entity’s methods.



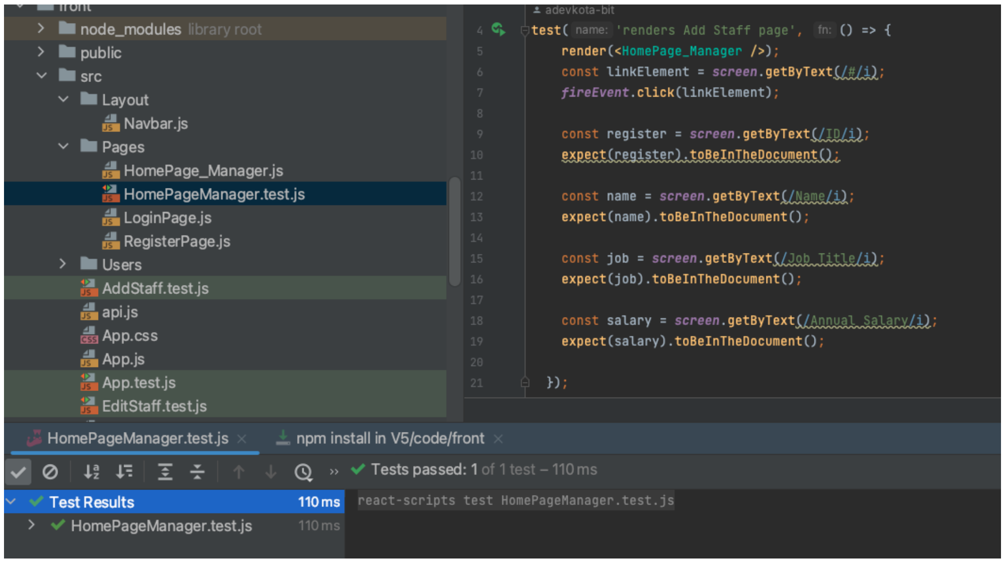
```
package Employee_Management_System.user;

import ...

class EmployeeTest {
    private Employee employee;

    @BeforeEach
    void setUp() { employee = new Employee(); }

    @Test
    void testGetName() {
        employee.setName("bart");
        assertEquals( expected: "bart", employee.getName());
```

```
Tests passed: 6 of 6 tests – 300 ms

EmployeeTest (Employee_Management_System  300 ms
  testGetEmail()            289 ms
  testGetId()               3 ms
  testGetName()             2 ms
  testGetJobTitle()         1 ms
  testGetAnnualSalary()     4 ms
  testGetPhoneNumber()      1 ms

/Users/jxie5/Library/Java/JavaVirtualMachines/temurin-17.0.8/Contents/Home/bin/java ...

Process finished with exit code 0
```

System testing (FrontEnd):

Frontend testing reflects the components that are being tested. All frontend test are located within /code/front/src

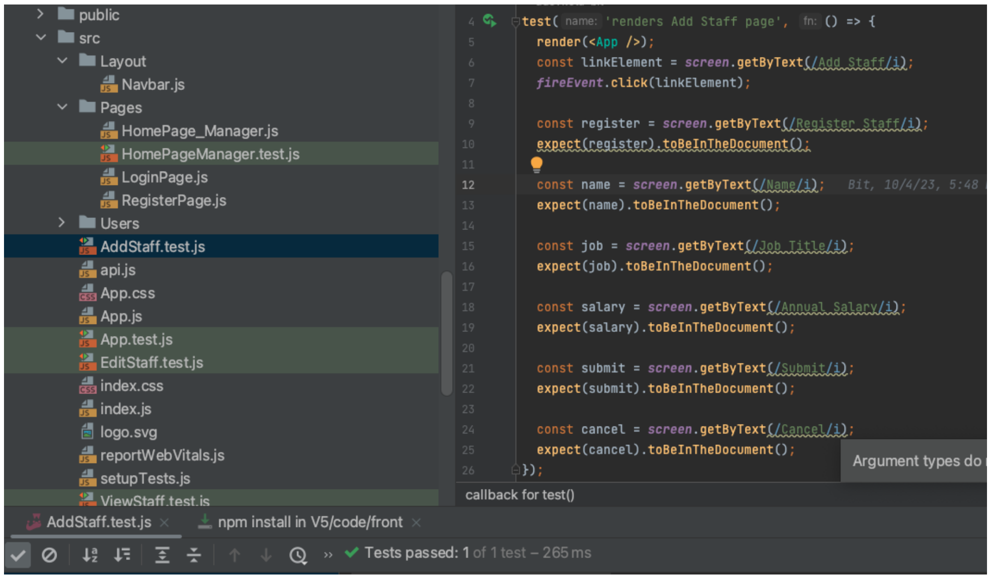
HomePage_Manager tests



```
test('renders Add Staff page', () => {
    render(<HomePage_Manager />);
    const linkElement = screen.getByText(/#/i);
    fireEvent.click(linkElement);

    const register = screen.getByText(/ID/i);
    expect(register).toBeInTheDocument();

    const name = screen.getByText(/Name/i);
    expect(name).toBeInTheDocument();

    const job = screen.getByText(/Job Title/i);
    expect(job).toBeInTheDocument();

    const salary = screen.getByText(/Annual Salary/i);
    expect(salary).toBeInTheDocument();

  });
```

```
Tests passed: 1 of 1 test – 110 ms
react-scripts test HomePageManager.test.js
```

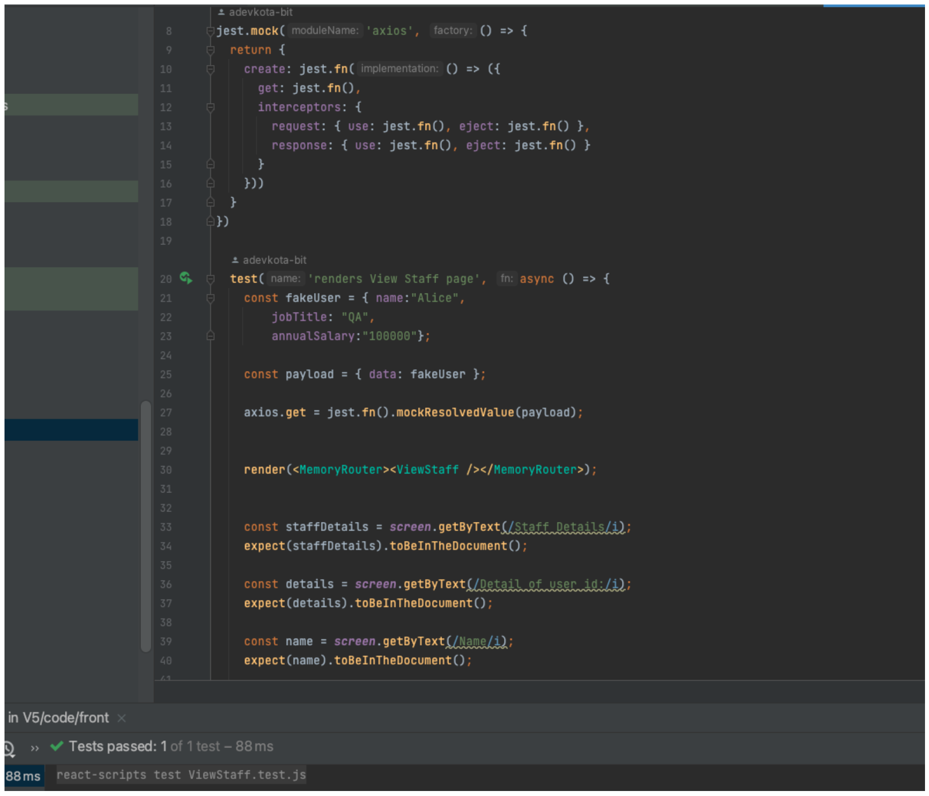
AddStaff test



```
test('renders Add Staff page', () => {
  render(<App />);
  const linkElement = screen.getByText(/Add Staff/i);
  fireEvent.click(linkElement);

  const register = screen.getByText(/Register Staff/i);
  expect(register).toBeInTheDocument();

  const name = screen.getByText(/Name/i);    Bit, 10/4/23, 5:48
  expect(name).toBeInTheDocument();

  const job = screen.getByText(/Job Title/i);
  expect(job).toBeInTheDocument();

  const salary = screen.getByText(/Annual Salary/i);
  expect(salary).toBeInTheDocument();

  const submit = screen.getByText(/Submit/i);
  expect(submit).toBeInTheDocument();

  const cancel = screen.getByText(/Cancel/i);
  expect(cancel).toBeInTheDocument();
});
```

```
Tests passed: 1 of 1 test – 265 ms
```

Viewstaff testing

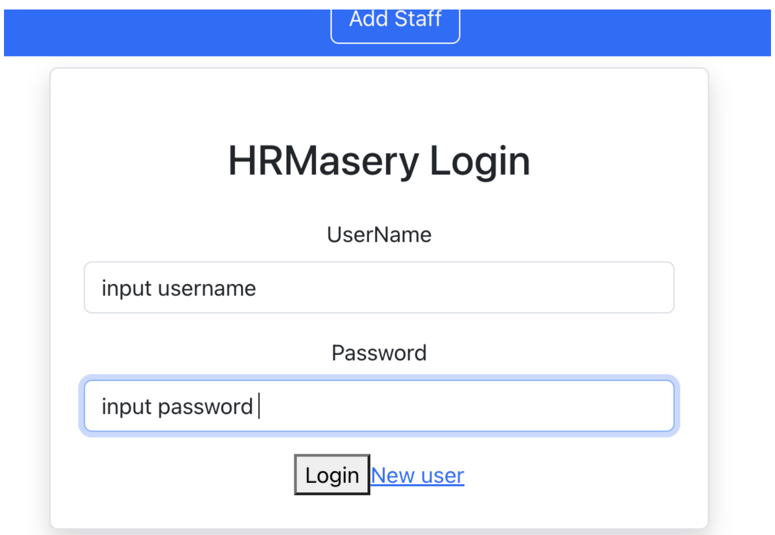


```
adevkota-bit
jest.mock('axios', () => {
  return {
    create: jest.fn(() => ({
      get: jest.fn(),
      interceptors: {
        request: { use: jest.fn(), eject: jest.fn() },
        response: { use: jest.fn(), eject: jest.fn() }
      }
    }))
  }
})

  adevkota-bit
  test('renders View Staff page', async () => {
    const fakeUser = { name:"Alice",
        jobTitle: "QA",
        annualSalary:"100000"};

    const payload = { data: fakeUser };

    axios.get = jest.fn().mockResolvedValue(payload);


    render(<MemoryRouter><ViewStaff /></MemoryRouter>);


    const staffDetails = screen.getByText(/Staff Details/i);
    expect(staffDetails).toBeInTheDocument();

    const details = screen.getByText(/Detail of user id:/i);
    expect(details).toBeInTheDocument();

    const name = screen.getByText(/Name/i);
    expect(name).toBeInTheDocument();
```

```
in V5/code/front
Tests passed: 1 of 1 test – 88 ms
88 ms  react-scripts test ViewStaff.test.js
```

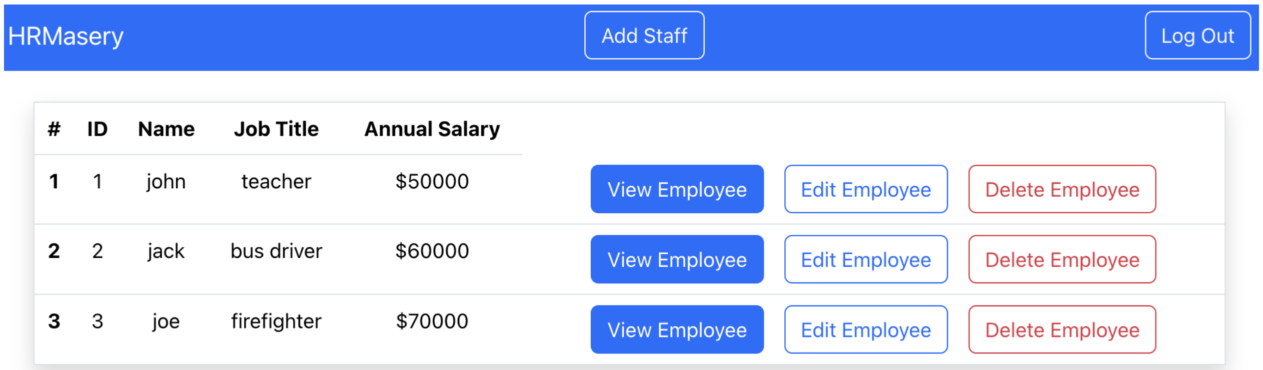
Acceptance Testing

User can login with username and password and then click on Login button



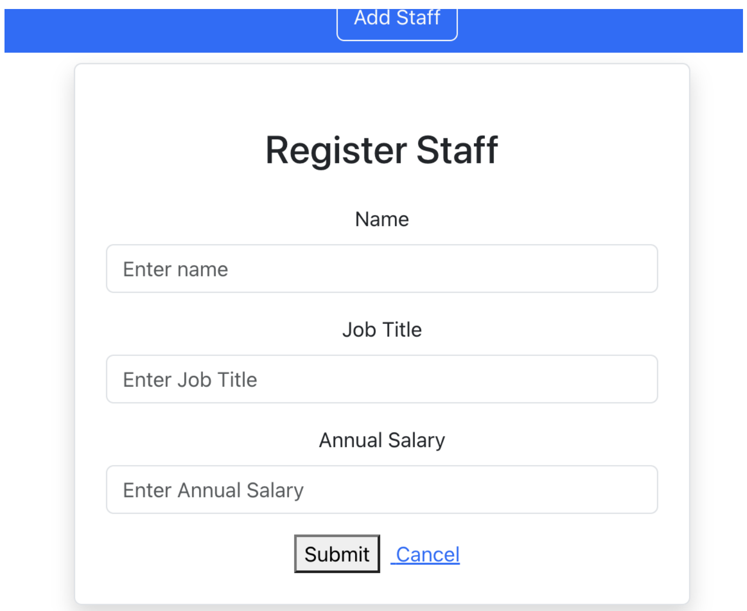
Admin user’s home page can view all staff information with feature buttons to view, edit, and delete employees.

Admin user’s home page also includes a Logout button to logout of the application.



#	ID	Name	Job Title	Annual Salary			
1	1	john	teacher	$50000	View Employee	Edit Employee	Delete Employee
2	2	jack	bus driver	$60000	View Employee	Edit Employee	Delete Employee
3	3	joe	firefighter	$70000	View Employee	Edit Employee	Delete Employee

Admin user can add additional stuff via the Add Staff button and add staff via submit after entering staff information



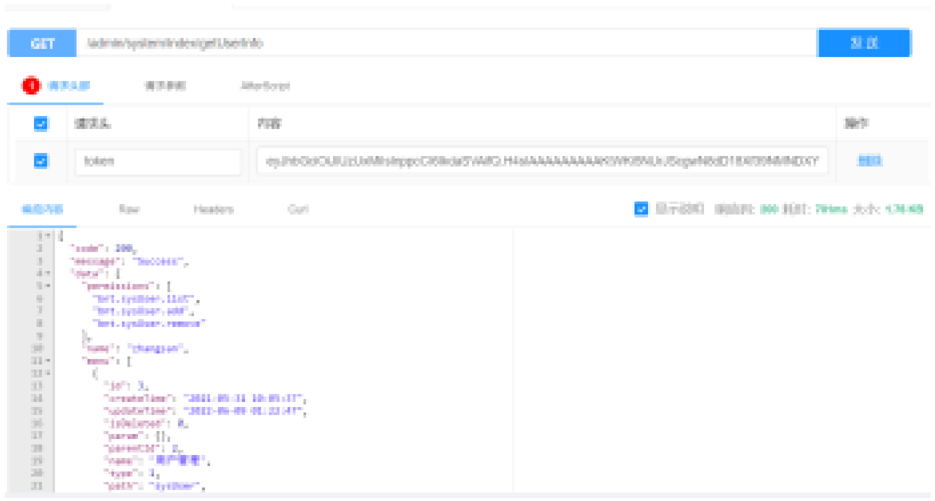
Register Staff

Name	Enter name
Job Title	Enter Job Title
Annual Salary	Enter Annual Salary

SpringSecurity:
After adding SpringSecurity dependencies, even if you bypass the front-end login page and go directly to use the method url like “ http://localhost:8800/admin/system/sysRole/findAll” , you will not get the information. It will show that

```
{"code":209,"message":"No permission","data":null}
```

After adding the global variable token, prove that you are logged in. All the methods will work just as before

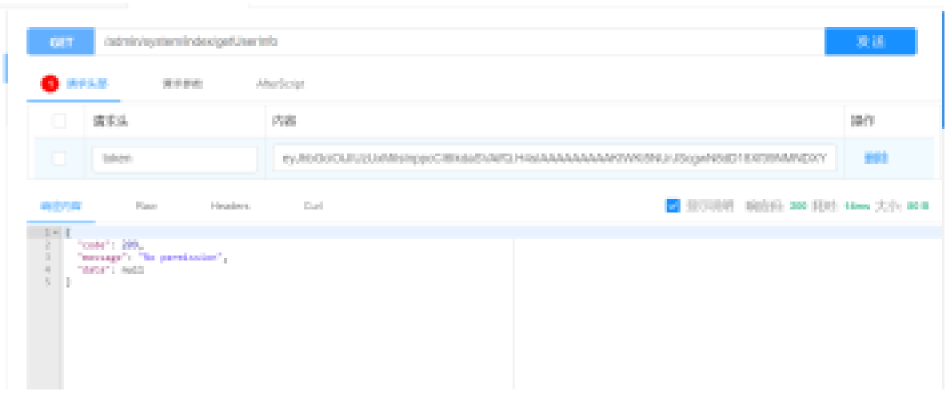


GET /admin/system/index/getUserInfo

token eyJhbGciOiJIUzUxMiIsInppcCI6IkdaSVAifQ.H4sIAAAAAAAAAK[illegible]

```
{
  "code": 200,
  "message": "Success",
  "data": {
    "permissions": [
      "bnt.sysUser.list",
      "bnt.sysUser.add",
      "bnt.sysUser.remove"
    ],
    "name": "zhangsan",
    "menus": [
      {
        "id": 3,
        "createTime": "[illegible]",
        "updateTime": "[illegible]",
        "isDeleted": 0,
        "param": {},
        "parentId": 2,
        "name": "用户管理",
        "type": 1,
        "path": "sysUser",
```

Without Token:



GET /admin/system/index/getUserInfo

token eyJhbGciOiJIUzUxMiIsInppcCI6IkdaSVAifQ.H4sIAAAAAAAAAK[illegible]

```
{
  "code": 209,
  "message": "No permission",
  "data": null
}
```

● Testing Metrics

In this section, you shall report any metrics used for the evaluation, e.g. # of test cases, test coverage, defects rate, etc.

● References

● Glossary