

TDL Project – To-Do-List Application

Waleed Tahir

20DecSoftware1

Introduction

Name Waleed Tahir

QA trainee

The overall objective of the project was:

To create an OOP-based web application, with utilisation of supporting tools, methodologies, and technologies, that encapsulates all fundamental and practical modules covered during training.

Concept

To Breakdown the project I looked at the MVP deliverable features which and broke them down into individual user stories and tasks for my Kanban Board and then went about completing the required tasks before looking at any further function that could be added to my application.

Sprint Plan

- Back-end application with full CRUD functionality
- Unit and Integration Testing
- Git repository and management
- Full supporting documentation including UML and ERD diagrams
- Build Tool Maven
- Project Management platform – Jira-Kanban Board
- Fully integrated front end

Consultant Journey

- Version control –Git(Git Hub and Git Bash)
- Programming language – java, javascript, html and css
- Database – local:H2
- Testing- JUnit, Integration and Selenium
- Static analysis – Sonarqube
- Project management – Jira

Continues Integration

The Version Control System that I used was Git;


- Use a mix of GitBash console and Github to manage iterations of my project working within the Master->Dev->features model to ensure full clarity and safety of my code.
- Regular commits to ensure Code is update operating using swift design principles by ensuring that a copy of my latest working code was always available in my repository.

CI

```
pc@DESKTOP-K4GMEJU MINGW64 ~/Documents/Project 2 TDL/TDL-Project (HTML)
$ git branch
  CRUD
  Dev
* HTML
  Testing
  features
  integration_testing
  main
  service_testing
```


Branch models

Front End and Backend now working in tandem


 awatahirqa committed 22 hours ago

Commits on Feb 12, 2021

Copy of HTML and JS from visual studios inside static folder of project


 awatahirqa committed 4 days ago

Built Integration tests and run sonar cube initially on project

 awatahirqa committed 4 days ago


Commits on Feb 10, 2021

Got my tests working by reorganising my package structures so I have ...


 awatahirqa committed 6 days ago

Commits on Feb 9, 2021


Created my services tests and made some adjustments to my service cla...

 awatahirqa committed 7 days ago

Finished testing for Domain and DTO and adjusted mistake in DTO class

 awatahirqa committed 7 days ago

Domain and DTO tests created

 awatahirqa committed 7 days ago

Commit history ensuring regular updates

Testing

Testing was completed using a combination of both JUnit Integration and Selenium

- JUnit was used to test the individual methods within each of my classes by initialising the relevant objects in my test setup and then calling the method to test it.
- Integration testing with JUnit 5 allows us to see if the HTTP endpoints in our application work with various CRUD-based HTTP requests. This makes use of the Controller layer, which can be 'mocked' so that we can test the outcomes of sending data to each HTTP endpoint.

src/main/java	71.8 %	491	193	684
> com.qa.springtdl.persistai	60.1 %	92	61	153
> com.qa.springtdl.persistai	69.4 %	129	57	186
> com.qa.springtdl.utilities	0.0 %	0	49	49
> com.qa.springtdl.restcont	85.7 %	102	17	119
> com.qa.springtdl	37.5 %	3	5	8
> com.qa.springtdl.services	97.5 %	158	4	162
> com.qa.springtdl.config	100.0 %	7	0	7
coverage at 71.8%				

JUnit

```
@Test
public void setterGetterTest() {
    task = new TasksDomain(1L, "Simple task to test my domain", 2, "01-01-2021", null, "Ongoing");

    Assertions.assertNotNull(task);
    task.setTaskId(1L);
    task.setSummary("Simple task to test my domain");
    task.setPriority(2);
    task.setDeadline("01-01-2021");
    task.setMyList(null);
    task.setStatus("ongoing");
    Assertions.assertNotNull(task.getTaskId());
    Assertions.assertNotNull(task.getSummary());
    Assertions.assertNotNull(task.getPriority());
    Assertions.assertNotNull(task.getDeadline());
    Assertions.assertNull(task.getMyList());
    Assertions.assertNotNull(task.getStatus());
}
```

Integration

```
@Test
public void create() throws Exception{

    TasksDomain contentBody = new TasksDomain(1L, "need to pick up veg",3,"07/03/21",null,"Ongoing");
    TasksDTO expectedResult = mapToDTO(contentBody);

    MockHttpServletRequestBuilder mockRequest = MockMvcRequestBuilders.request(HttpMethod.POST, "http://localhost:8080",
        .contentType(MediaType.APPLICATION_JSON).content(jsonifier.writeValueAsString(contentBody)).accept(MediaType

    ResultMatcher matchStatus = MockMvcResultMatchers.status().isCreated();
    ResultMatcher matchContent = MockMvcResultMatchers.content().json(jsonifier.writeValueAsString(expectedResult));

    this.mock.perform(mockRequest).andExpect(matchStatus).andExpect(matchContent);
}
```

```

est
public void createTask() throws InterruptedException{
    test = report.startTest("Create Task Selenium Test");
    //Spin up URL
    driver.get(URL);
    //Input Vales
    targ.findElement(By.xpath("//*[@id=\"summary\"]"))
    targ.sendKeys("selenium test case");
    targ.findElement(By.xpath("//*[@id=\"priority\"]"))
    targ.sendKeys("2");
    targ.findElement(By.xpath("//*[@id=\"deadline\"]"))
    targ.sendKeys("20/02/2021");
    targ.findElement(By.xpath("//*[@id=\"myList\"]"))
    targ.sendKeys("1");
    targ.findElement(By.xpath("//*[@id=\"status\"]"))
    targ.sendKeys("Ongoing");
    //execute create
    targ.findElement(By.xpath("/html/body/div[1]/div/"))
    targ.click();
    //then check it has worked
    targ = new WebDriverWait(driver, 5).until

    //targ = driver.findElement(By.xpath("//*[@id=\"summary\"]"))
    String result = targ.getText();
    String expected = "was created, use read

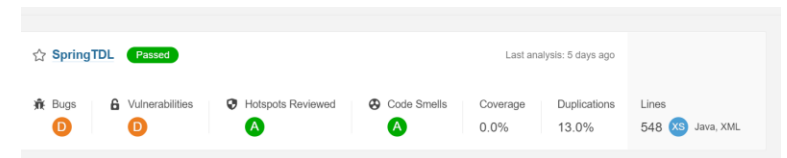
```

Selenium

Selenium is a collection of tools and libraries used to automate web browsers, and is used for User acceptance testing. It can be broken down into three main components Given, when and then

Static analysis - SonarQube

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities



Demonstration

User story : As a user I want to be able to create tasks so that I can keep track of my outstanding work and organise them within my ToDoList.

User Story: As a user I want to be able to delete a task so that I can keep my ToDoList clear of any unnecessary information.

Sprint Review

- I managed to complete the core functionality of my code;
 - CRUD services for Back end
 - Fully operation Front end
 - Junit and Integration Tests
- What got left behind;
 - Testing coverage not able to reach 80%
 - Additional functionality such as Users access control.
 - Optimized my code by reducing smells identified by sonarqube.

Sprint Retrospective

What went well?

- Back end functionality
- Front End Integration
- Unit and Integration Testing

Improvements

- Adding further functionality to the code
- More efficient/cleaner code e.g. Lombok
- Higher test coverage
- Cleaner Selenium Tests using POM design model

Conclusion

- Core Aim of the Brief met
 - I have deepened my understanding of the technologies learnt over the course of the project
 - Moving forward continue to improve and as I gain experience complete my project with more functionality.
 - Complete Optimization and improve code making use of technologies such as sonarqube to improve weaknesses in my code
- Thank you for Listning are there any Question?