

Hobby Web Application Project

BY HAARIS MOGHAL

Introduction

Created a Full stack web application, with utilization of supporting tools and methodologies, covered during training.

Create a full-stack web application following the Enterprise Architecture Model, using:

- Persistent layer - A managed database hosted locally or within the Cloud Provider (e.g. MySQL in GCP)
- Business Layer - An application back-end developed using the language from your Programming Fundamentals module (e.g Spring)
- Data Layer - A front-end developed using the language from your Front-End Web Technologies module (e.g. JavaScript)

Concept

Project objective:

- To create a functional Web application
- Focused on the MVP (Minimum Viable Product)

Project breakdown:

- Key deliverables were split into user stories on my Kanban board.
- Back-end application with full CRUD functionality
- 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 System: Git – Project management. Pushing and Reverting.
- Source Code Management: GitHub. See the progress of the project coming together.
- Database management system – local:H2 and Google Cloud Platform (GCP)
- Backend programming language – Spring
- Testing – Junit, Mockito, Selenium and SonarQube.
- Build tools – Maven. Using dependencies and building application.
- Project Management platform – Jira/Kanban board

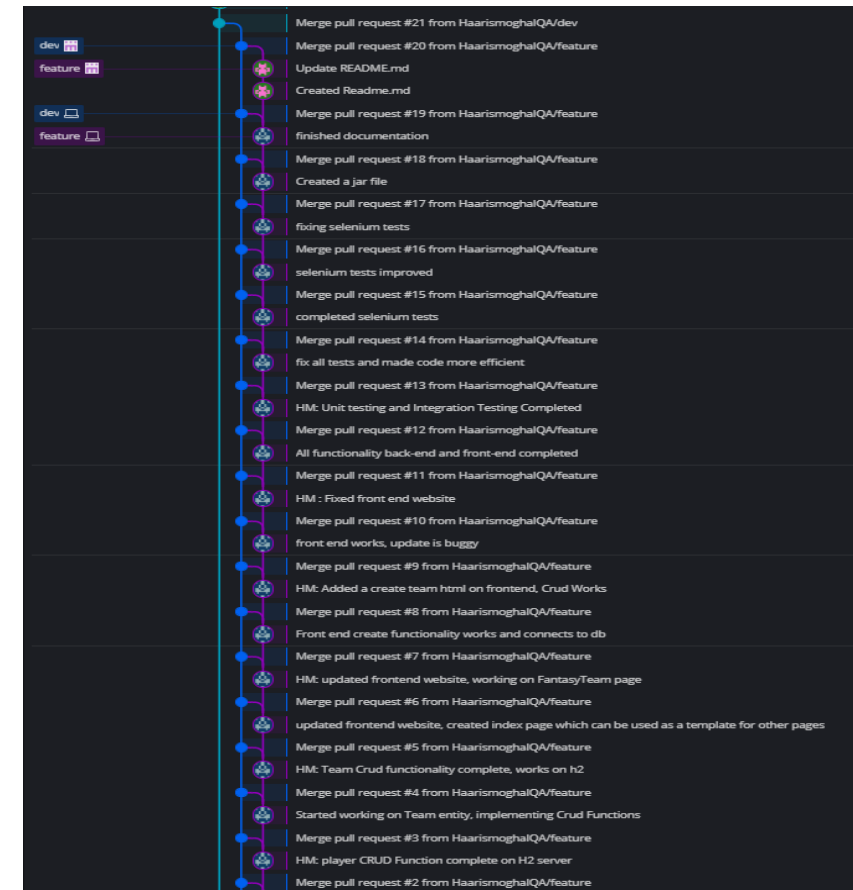
Continuous Integration

- Git for version control
- Used a Feature-branch model. (GitKraken)
- Regular commits to ensure project safety.
- Used Jira tags on most commits.

Continuous Integration




HM: Created a jar file	#18 by HaarismoghalQA was merged 9 minutes ago
HM: fixing selenium tests	#17 by HaarismoghalQA was merged 2 days ago
[HOB-15] HM: selenium tests improved	#16 by HaarismoghalQA was merged 2 days ago
HM: completed Selenium testing	#15 by HaarismoghalQA was merged 2 days ago
HM: fix all tests and made code more efficient	#14 by HaarismoghalQA was merged 3 days ago
[HOB-13,14] HM: Unit testing and Integration Testing Completed	#13 by HaarismoghalQA was merged 3 days ago
HM: All functionality back-end and front-end completed	#12 by HaarismoghalQA was merged 4 days ago
HM : Fixed front end website	#11 by HaarismoghalQA was merged 4 days ago
HM : front end works, update is buggy	#10 by HaarismoghalQA was merged 4 days ago
HM: Added a create team html on frontend, Crud Works	#9 by HaarismoghalQA was merged 5 days ago
HM : Front end create functionality works and connects to db	#8 by HaarismoghalQA was merged 5 days ago
HM: updated frontend website, working on FantasyTeam page	#7 by HaarismoghalQA was merged 9 days ago
HM: updated frontend website, created index page which can be used as a t...	#6 by HaarismoghalQA was merged 9 days ago
[HOB-7,8,9,10] HM: Team Crud functionality complete, works on h2	#5 by HaarismoghalQA was merged 9 days ago
HM : Started working on Team entity, implementing Crud Functions	#4 by HaarismoghalQA was merged 9 days ago
[HOB-3,4,5,6] HM: player CRUD Function complete on H2 server	#3 by HaarismoghalQA was merged 10 days ago
HM : Started my website and create Player entity, connected to H2 database	#2 by HaarismoghalQA was merged 10 days ago
[HOB-1] HM : Created the project and added first Diagram	#1 by HaarismoghalQA was merged 10 days ago

Added Jira Tags
On relevant
commits.



Testing unit

- Managed to get 84.3% test coverage on src/main/java folder
- Junit was used to test individual methods in each of the classes which were used. (Services and Controller)
- Mockito was used to mock objects so that I can use to test the methods.

HobbyProject (2) (20 Jun 2021 22:24:43)				
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▼ 📁 HobbyProject	 94.1 %	2,223	139	2,362
> 📁 src/main/java	 84.3 %	706	131	837
> 📁 src/test/java	 99.5 %	1,517	8	1,525

Testing- Mockito

- Mockito was used along with Junit which allows you to create and configure mock objects
- Mockito was used to mock objects so that I can use to test the methods.

```
@Test
void testCreate() throws Exception {

    Team testTeam = new Team("Arsenal");
    String testTeamAsJSON = this.mapper.writeValueAsString(testTeam);

    Team testSavedTeam = new Team(21, "Arsenal");
    String testSavedTeamAsJSON = this.mapper.writeValueAsString(testSavedTeam);

    RequestBuilder mockRequest = post("/team/create").content(testTeamAsJSON)
        .contentType(MediaType.APPLICATION_JSON);

    ResultMatcher checkStatus = status().isOk();

    ResultMatcher checkBody = content().json(testSavedTeamAsJSON);

    this.mvc.perform(mockRequest).andExpect(checkStatus).andExpect(checkBody);
}
```


Testing - SonarQube

The SonarQube dashboard for 'HobbyProject' (master branch) shows a 'Passed' quality gate status. The 'MEASURES' section displays the following data:

Measure	Value	Quality Gate
New Code	3 Bugs	Reliability (B)
Overall Code	4 Vulnerabilities	Security (I)
Security Hotspots	0	Security Review (A)
Debt	4h 45min	Maintainability (A)
Coverage on 256 Lines to cover	0.0%	Unit Tests
Duplications on 620 Lines	9.1%	Duplicated Blocks

The SonarQube dashboard for 'HobbyProject' (master branch) shows a list of code quality issues. The 'My Issues' tab is selected, displaying 1 / 63 issues with a total effort of 4h 45min.

Filters:

- Type: CODE SMELL (3)
- Severity: Blocker (4), Critical (0), Major (17), Minor (30), Info (12)

Issues:

- Remove this unnecessary import: same package classes are always implicitly imported. (Code Smell, Major, 2min effort)
- This block of commented-out lines of code should be removed. (Code Smell, Major, 5min effort)
- Replace this use of System.out or System.err by a logger. (Code Smell, Major, 10min effort)
- 1 duplicated blocks of code must be removed. (Code Smell, Major, 20min effort)
- Remove this unused import 'javax.persistence.Column'. (Code Smell, Minor, 2min effort)
- Rename this field 'PlayerId' to match the regular expression '[a-z][a-zA-Z0-9]*'. (Code Smell, Minor, 2min effort)
- Rename this field 'PlayerName' to match the regular expression '[a-z][a-zA-Z0-9]*'. (Code Smell, Minor, 2min effort)
- This block of commented-out lines of code should be removed. (Code Smell, Major, 5min effort)
- Replace this if-then-else statement by a single return statement. (Code Smell, Minor, 2min effort)

Testing - Selenium

```
@Test
public void testCreatePlayer() throws InterruptedException {

    //Opens the webpage of my index page
    driver.get("http://localhost:8080/index.html");

    Thread.sleep(2000);

    //clicks on the Create a player page
    targ = driver.findElement(By.xpath("/html/body/main/div/header/div[3]/ul/li[2]/a/p"));
    targ.click();
    Thread.sleep(2000);

    //click on the selection
    targ = driver.findElement(By.xpath("/html/body/div[2]/div/section[1]/div/form[1]/select"));
    targ.click();
    Thread.sleep(2000);
    //Chooses the first option on the selection
    targ = driver.findElement(By.xpath("/html/body/div[2]/div/section[1]/div/form[1]/select/option"));
    targ.click();
    Thread.sleep(2000);
    //The player name is entered
    WebElement u = driver.findElement(By.name("playerName"));

    u.sendKeys("Henry");

    Thread.sleep(2000);
    //The player age is entered
    WebElement a = driver.findElement(By.name("age"));

    a.sendKeys("25");
    Thread.sleep(2000);
    //Submit button is clicked
    u.submit();

    Thread.sleep(2000);

    // ASSERTIONS
    targ = driver.findElement(By.xpath("/html/body/div[2]/div/section[2]/div/div[2]/div/div[1]/p[4]"));
    assertEquals("playerName: Henry", targ.getText());

    targ = driver.findElement(By.xpath("/html/body/div[2]/div/section[2]/div/div[2]/div/div[1]/p[5]"));
    assertEquals("Age: 25", targ.getText());
}
```

Finding the target of each element

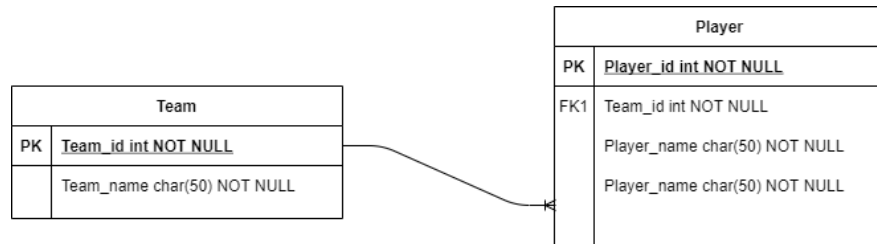
Able to submit a form using “submit()” method.

Used Assertion to double check if test worked/good practise.

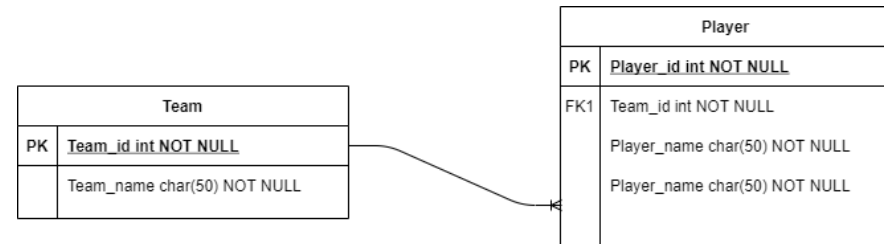
Demonstration

- User story 1: User is able to ADD a Team in the system SO THAT the User can create a team of players.
- User story 2: User is able to ADD an Player in the system SO THAT the user can add players to there Team.
- User story 3: User is able to Update a Team in the system SO THAT the User can change the name of the team.
- User story 4: User can Delete a Player in the system SO THAT they can remove players from there team.

ERD – Entity Relationship Diagram



Before



After

Jira – Kanban Board Week 1

Projects / HobbyProject

HOB board

HM Epic

TO DO 4 ISSUES

- User can UPDATE a Team
TEAM
HOB-8
- User can DELETE a Team
TEAM
HOB-9
- User can READALL Team
TEAM
HOB-10
- Complete Testing
HOB-11 HM

+ Create issue

IN PROGRESS 1 ISSUE

- User can CREATE a Team
TEAM
HOB-7

DONE 4 ISSUES

- User can CREATE a Player
PLAYER
HOB-3
- User can UPDATE a Player
PLAYER
HOB-4
- User can DELETE a Player
PLAYER
HOB-5
- User can READALL Players
PLAYER
HOB-6

Player / HOB-3

User can CREATE a Player

Done Done

Attach Add a child issue Link issue

Description
Add a description...

Acceptance Criteria
GIVEN THAT I have access to the system
WHEN I used the database
THEN I should be able to Create all customers

Activity
Show: Comments History Newest first

HM Add a comment...
Pro tip: press M to comment

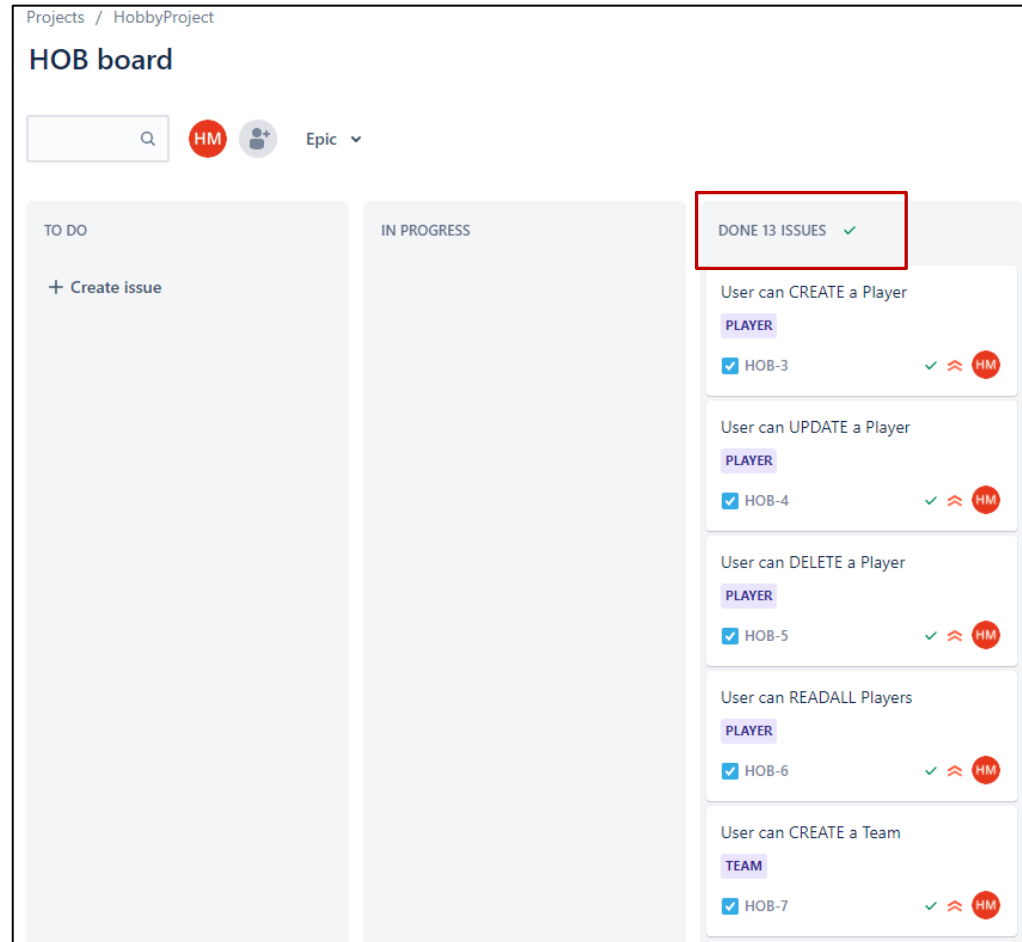
Priority Highest
Assignee Unassigned
Story Points 1
Reporter HM Haaris Moghal
Priority MOSCoW MUST HAVE
Labels None

Created yesterday
Updated 11 minutes ago
Resolved 6 hours ago

Configure

Example of one the user stories.

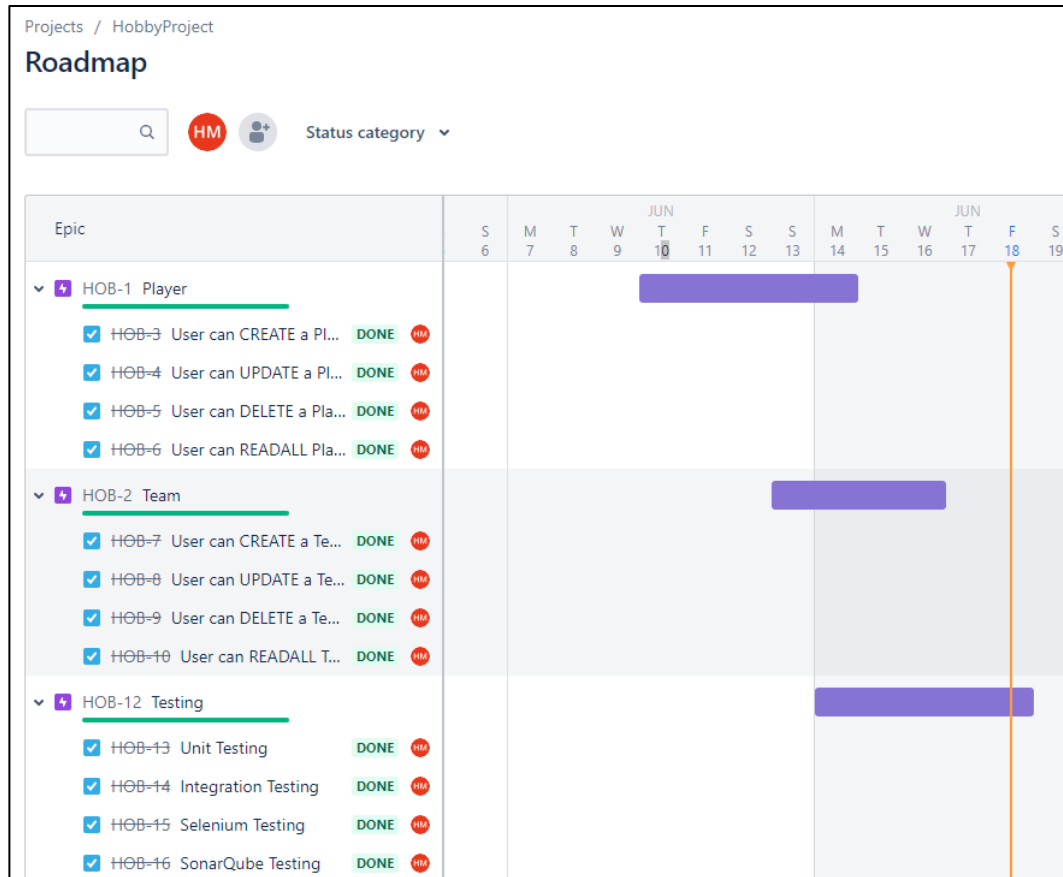
Jira – Kanban Board Week 2



All CRUD functions for Team and Players completed.

Unit, Integration and Selenium testing completed.

Jira Roadmap



All user stories completed.

Sprint review

I managed to complete the main functionality required according the deliverables:

- CRUD functionality on the Front-end
- CRUD functionality on the Back-end
- Testing <80%

What got left behind:

- All MVP has been met, nothing was left behind.
- On front-end, About page (extra Web page) is not complete
- On front-end, Login page (extra Web page) is not fully functional

Sprint retrospective

- What went well?
 - I was able to meet all the user requirements
 - Version Control
- Improvements?
 - SonarQube, fix all code smells and to be able to use unit coverage
 - Time management

Conclusion

Creating this Web application has improved my knowledge:

- Back-end programming : Spring
- Testing : Junit, Mockito, SonarQube and Selenium

Overall the aims project have been met:

- Crud functionality
- Testing
- Using a database

Improvements?

- More realistic with setting time frames of each sprints

Thank you for listening!

