Web Application Project Presentation

Jehad AbdelBaqi - IT Consultant

Presentation details

1. A brief introduction about myself and my journey in the world of tech. Also some information regarding what I have learned on this project.
2. Discussion regarding the project, including a brief outline of the project, version control, testing and final product outcome.
3. A demonstration of the application running through a few user stories - showing what the application can do with it's built in functionality.
4. Discussion of the sprint review and retrospective with a final reflection on the project - taking into consideration what more could have been done using MoSCoW methodology.
5. Designated time for questions and discussion.

Self Intro

Jehad AbdelBaqi - A journey into tech and development within the field.

- ➔ **Unexpected**
  A new journey into the world of tech with a previous sales background.
- ➔ **Emotional**
  Addicted to programming.
- ➔ **Simple**
  Fun and easy going individual.

Project details:

- Backend database creation using SpringBoot and Java.
- Testing - Unit and integration.
- Frontend web page creation with functionality
- Using HTML, CSS and JavaScript.
- Final build on Maven.

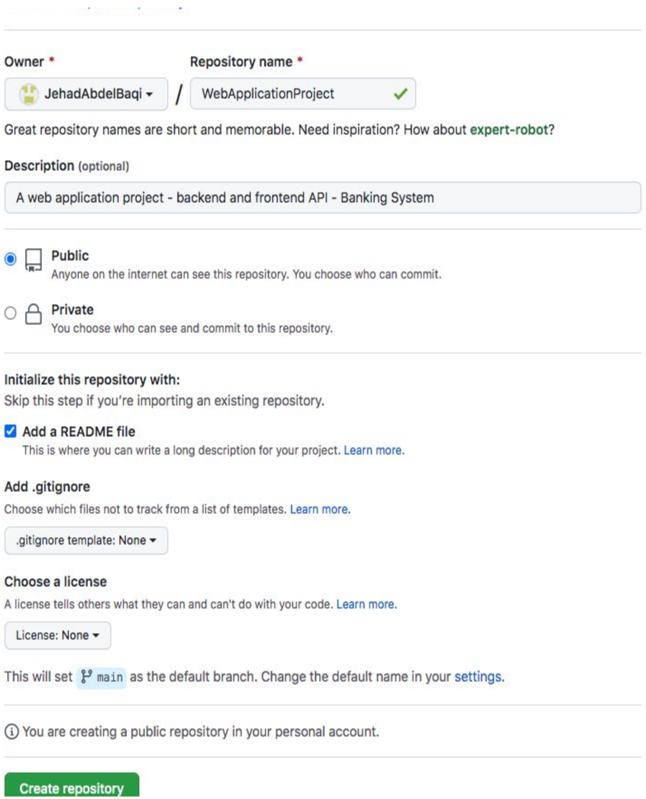
Project management

Source control using Git and GitHub.

Project management board using Jira software.

2. Version Control

- ➔ **What**
  Source control
- ➔ **Where**
  GitHub


Owner * JehadAbdelBaqi / Repository name * WebApplicationProject ✓

Great repository names are short and memorable. Need inspiration? How about expert-robot?

Description (optional)

A web application project - backend and frontend API - Banking System

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

✓ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.

License: None

This will set main as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

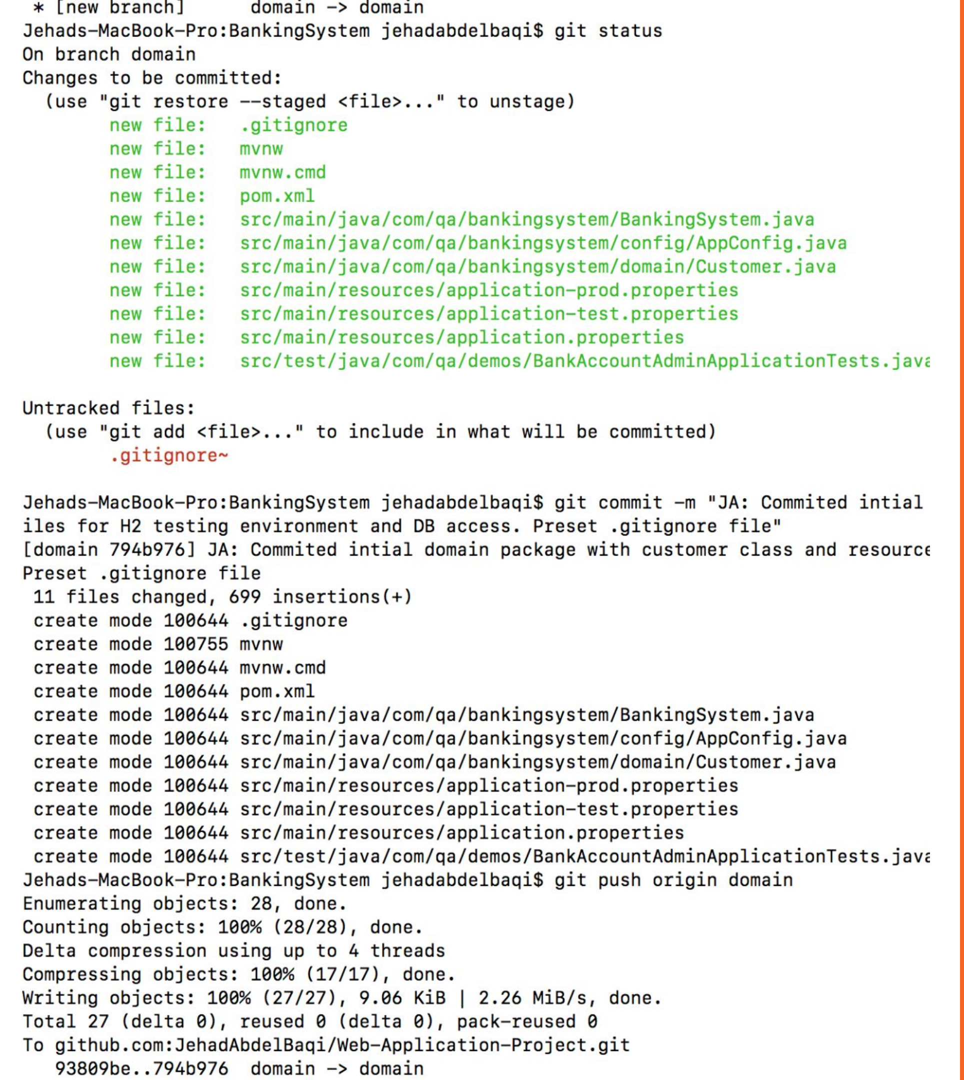
```
Last login: Mon May 30 12:13:31 on ttys000
Jehads-MacBook-Pro:~ jehadabdelbaqi$ cd /Users/jehadabdelbaqi/eclipse/BankingSystem/
Jehads-MacBook-Pro:BankingSystem jehadabdelbaqi$ git init
```

We create a GitHub repository and we git init our project folder.

Remember

Link your project to the GitHub Repository.

Create branches for each feature and push them into your development branch.



```
 * [new branch]      domain -> domain
Jehads-MacBook-Pro:BankingSystem jehadabdelbaqi$ git status
On branch domain
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        new file:   mvnw
        new file:   mvnw.cmd
        new file:   pom.xml
        new file:   src/main/java/com/qa/bankingsystem/BankingSystem.java
        new file:   src/main/java/com/qa/bankingsystem/config/AppConfig.java
        new file:   src/main/java/com/qa/bankingsystem/domain/Customer.java
        new file:   src/main/resources/application-prod.properties
        new file:   src/main/resources/application-test.properties
        new file:   src/main/resources/application.properties
        new file:   src/test/java/com/qa/demos/BankAccountAdminApplicationTests.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore~

Jehads-MacBook-Pro:BankingSystem jehadabdelbaqi$ git commit -m "JA: Commited intial
iles for H2 testing environment and DB access. Preset .gitignore file"
[domain 794b976] JA: Commited intial domain package with customer class and resource
Preset .gitignore file
 11 files changed, 699 insertions(+)
 create mode 100644 .gitignore
 create mode 100755 mvnw
 create mode 100644 mvnw.cmd
 create mode 100644 pom.xml
 create mode 100644 src/main/java/com/qa/bankingsystem/BankingSystem.java
 create mode 100644 src/main/java/com/qa/bankingsystem/config/AppConfig.java
 create mode 100644 src/main/java/com/qa/bankingsystem/domain/Customer.java
 create mode 100644 src/main/resources/application-prod.properties
 create mode 100644 src/main/resources/application-test.properties
 create mode 100644 src/main/resources/application.properties
 create mode 100644 src/test/java/com/qa/demos/BankAccountAdminApplicationTests.java
Jehads-MacBook-Pro:BankingSystem jehadabdelbaqi$ git push origin domain
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 4 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (27/27), 9.06 KiB | 2.26 MiB/s, done.
Total 27 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:JehadAbdelBaqi/Web-Application-Project.git
   93809be..794b976  domain -> domain
```

Add, commit, push.

Tip

Remember to always create a new branch.

Testing

➔ **What**

Testing

➔ **Where**

Eclipse

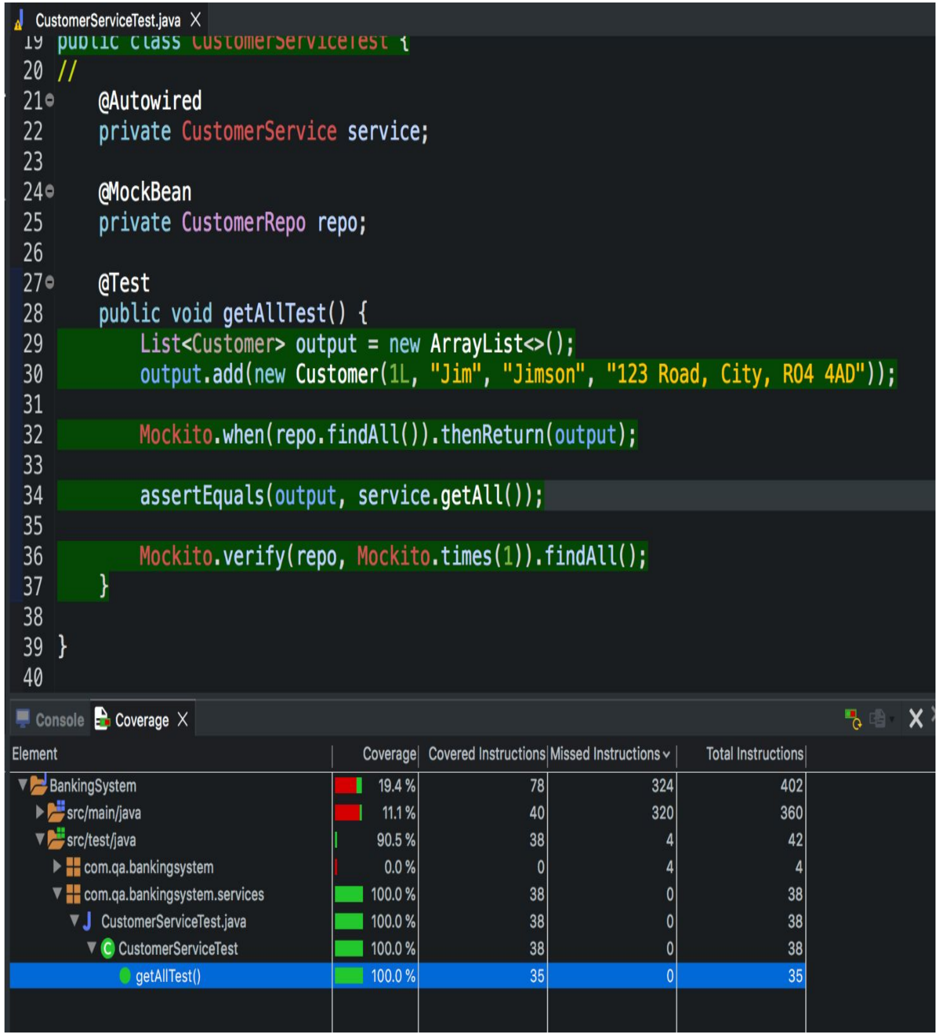
➔ **How**

Junit, SpringBoot and Mockito

Tip

Always run tests periodically as you are writing them.

Test each unit you write a test for before continuing to handle any issues.



```java
public class CustomerServiceTest {
//
    @Autowired
    private CustomerService service;

    @MockBean
    private CustomerRepo repo;

    @Test
    public void getAllTest() {
        List<Customer> output = new ArrayList<>();
        output.add(new Customer(1L, "Jim", "Jimson", "123 Road, City, R04 4AD"));

        Mockito.when(repo.findAll()).thenReturn(output);

        assertEquals(output, service.getAll());

        Mockito.verify(repo, Mockito.times(1)).findAll();
    }

}
```

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
BankingSystem	19.4 %	78	324	402
src/main/java	11.1 %	40	320	360
src/test/java	90.5 %	38	4	42
com.qa.bankingsystem	0.0 %	0	4	4
com.qa.bankingsystem.services	100.0 %	38	0	38
CustomerServiceTest.java	100.0 %	38	0	38
CustomerServiceTest	100.0 %	38	0	38
getAllTest()	100.0 %	35	0	35

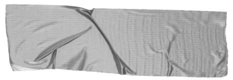
A simple Mockery

Create an entity

Mock the dependency

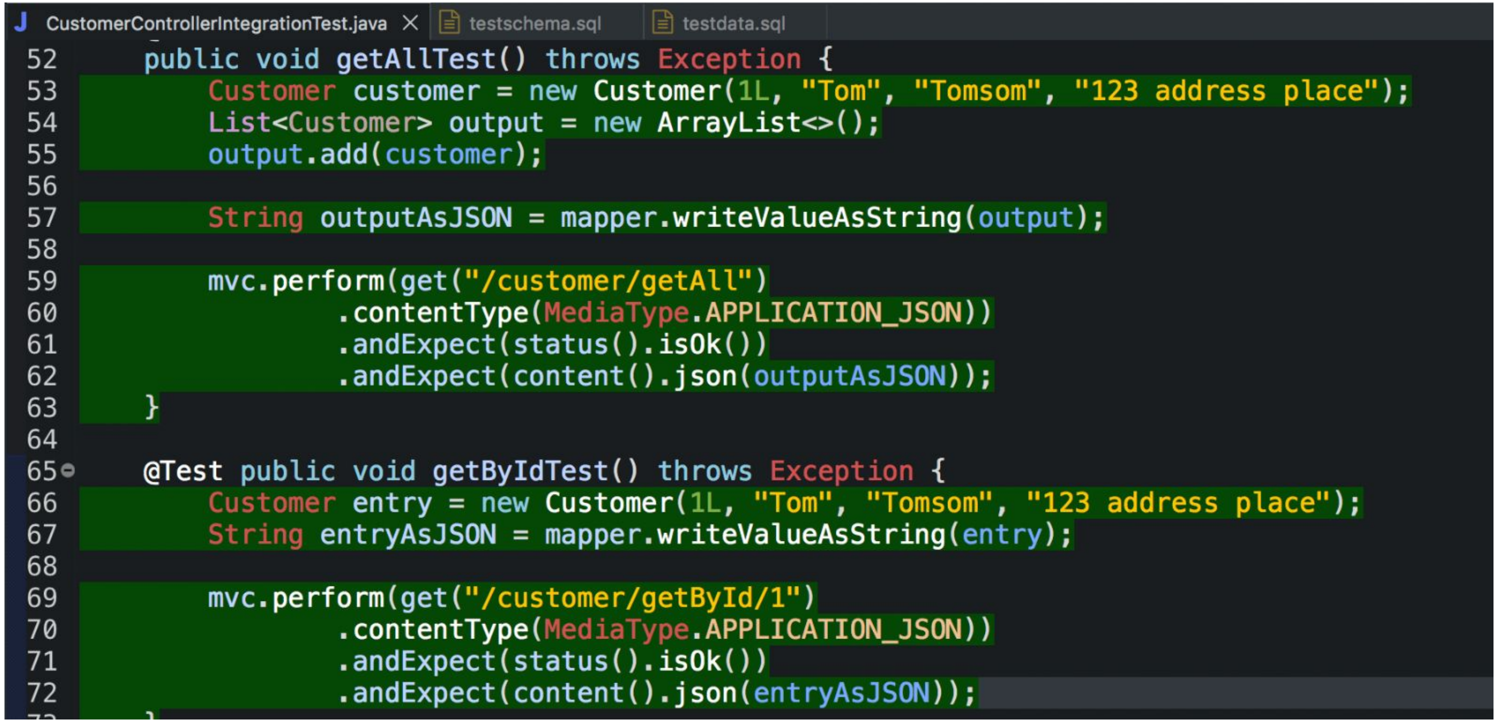
Test the method

Verify the Mock



Tip

Although the verify isn't needed it is still good practice to do it.


CustomerControllerIntegrationTest.java testschema.sql testdata.sql

```
    public void getAllTest() throws Exception {
        Customer customer = new Customer(1L, "Tom", "Tomsom", "123 address place");
        List<Customer> output = new ArrayList<>();
        output.add(customer);

        String outputAsJSON = mapper.writeValueAsString(output);

        mvc.perform(get("/customer/getAll")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(content().json(outputAsJSON));
    }

    @Test public void getByIdTest() throws Exception {
        Customer entry = new Customer(1L, "Tom", "Tomsom", "123 address place");
        String entryAsJSON = mapper.writeValueAsString(entry);

        mvc.perform(get("/customer/getById/1")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(content().json(entryAsJSON));
```

Let us integrate

Create an entity and change it to JSON. Return the outcome and expected status code.

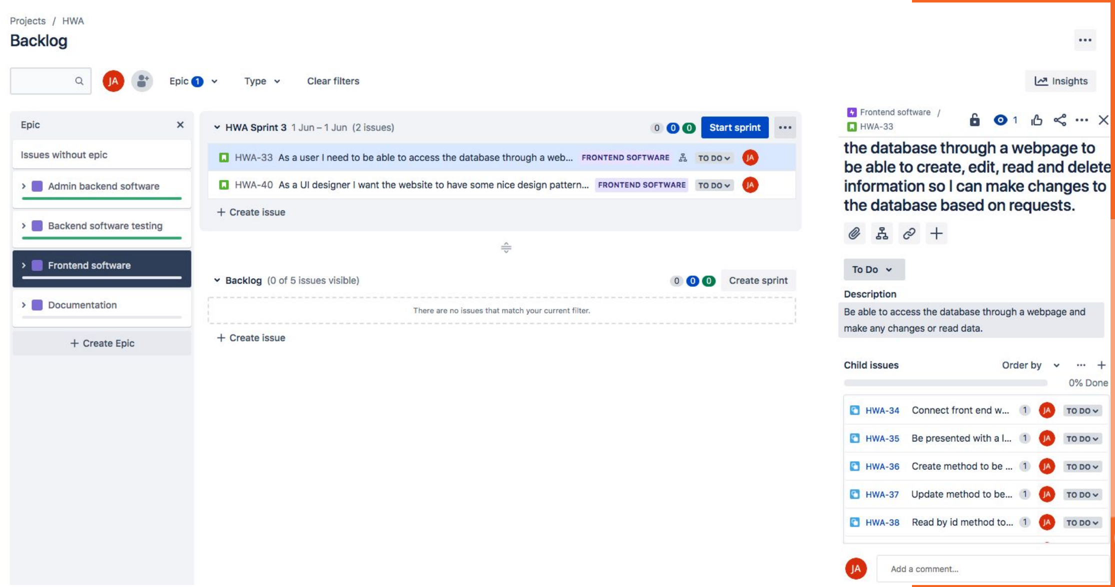
Demonstration

➔ **How**

Program built using Maven and run on terminal

Sprint Review

- ➔ **Where**
  Jira
- ➔ **What**
  Project management


Projects / HWA

Backlog

Epic 1 · Type · Clear filters

Epic

- Issues without epic
- Admin backend software
- Backend software testing
- Frontend software
- Documentation
- + Create Epic

HWA Sprint 3 1 Jun – 1 Jun (2 issues) 0 0 0 Start sprint

HWA-33	As a user I need to be able to access the database through a web...	FRONTEND SOFTWARE	TO DO	JA
HWA-40	As a UI designer I want the website to have some nice design pattern...	FRONTEND SOFTWARE	TO DO	JA

+ Create issue

Backlog (0 of 5 issues visible) 0 0 0 Create sprint

There are no issues that match your current filter.

+ Create issue

Insights

Frontend software / HWA-33

the database through a webpage to be able to create, edit, read and delete information so I can make changes to the database based on requests.

To Do

Description

Be able to access the database through a webpage and make any changes or read data.

Child issues Order by

0% Done

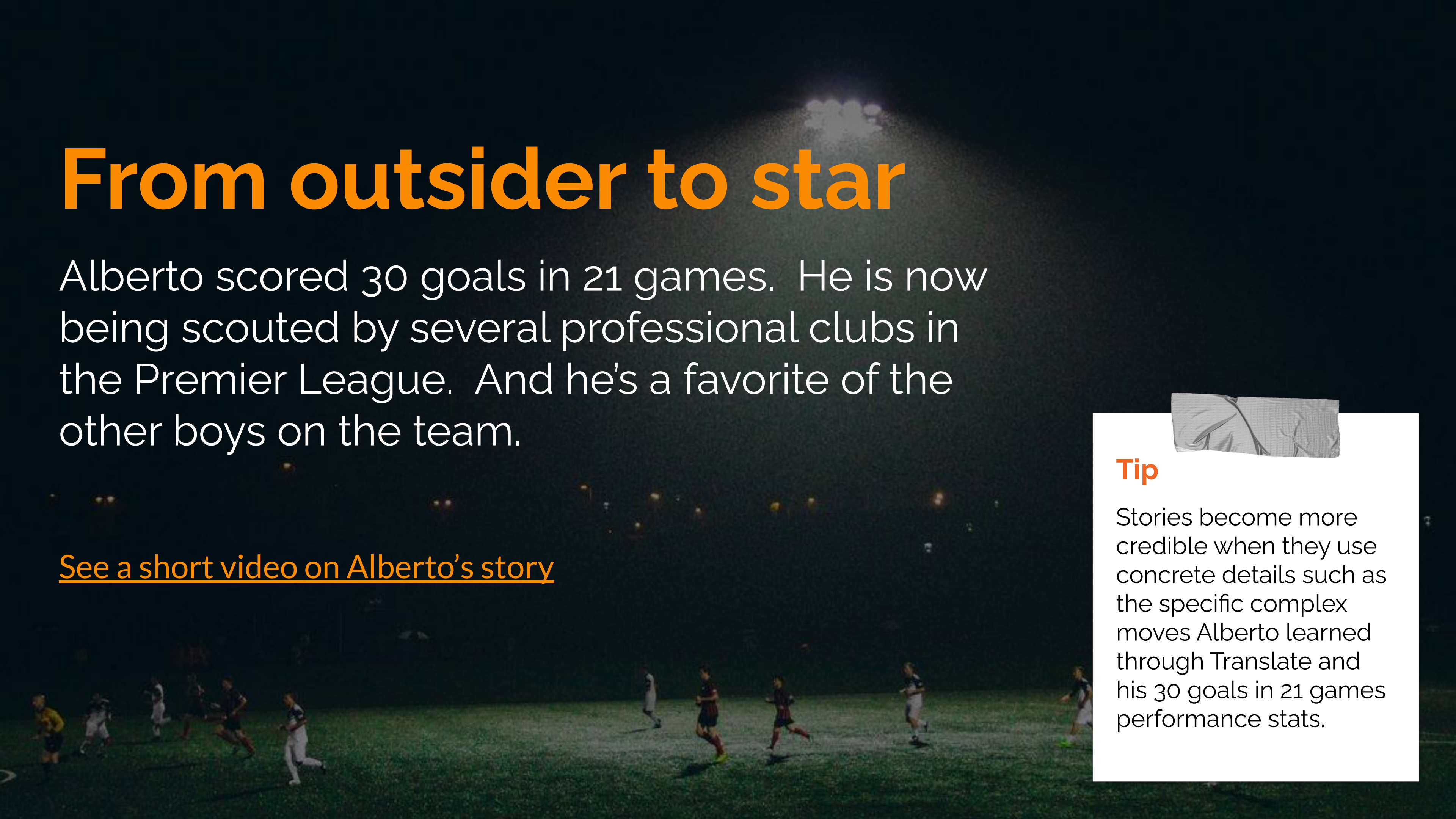
HWA-34	Connect front end w...	1	JA	TO DO
HWA-35	Be presented with a l...	1	JA	TO DO
HWA-36	Create method to be ...	1	JA	TO DO
HWA-37	Update method to be...	1	JA	TO DO
HWA-38	Read by id method to...	1	JA	TO DO

Add a comment...

Create tasks based off of user stories and manage your workflow using the Jira software

Tip

If not creating a user story - create a task instead. There tasks might include things which don't refer to a user story.



From outsider to star

Alberto scored 30 goals in 21 games. He is now being scouted by several professional clubs in the Premier League. And he's a favorite of the other boys on the team.

See a short video on Alberto's story

Tip

Stories become more credible when they use concrete details such as the specific complex moves Alberto learned through Translate and his 30 goals in 21 games performance stats.

Sprint Retrospective

- **What**
  MoSCoW to determine the end product needs and what is missing
- **Why**
  Agile development

A retrospective look

M The application MUST have a backend database which can be communicated to through a webpage. The basic functionality is to create, read, update and delete data into, from and out of the database.

o

S The application SHOULD have more data and table columns to be able to use the CRUD functions on. I.E customer account details and balance.

C The application COULD have user limitations and access to both customers and admin. Admin customers should be able to access all customer details and make changes whilst customer access is only for the specific customer using the page. Below this is discussed further by using password access.

o

W The application WON'T have password access and user limitations due to time constraints but this is definitely something which would be included should the application have customer access.

Thank you

QUESTION AND DISCUSSIONS