

SPELLTRACKER V2

NSAC15 PROJECT

WHAT IS THIS PROJECT?

A Software Development exercise

- Agile & Project Management
- Databases
- Java/Spring Boot API
- Front-End
- Automated Testing

Agile - Scrums and Kanban boards

Databases - Testing in h2 and production in SQL

Java – Maven, Spring Starter project written in Eclipse

Front-end – Written in VSCode. HTML, JavaScript, CSS, Bootstrap

Testing – JUnit, Mockito

WHAT IS SPELLTRACKER?

- Interactive, updatable database for use with D&D 5e
- Stores and manipulates basic information about game elements
- Can be used via a simple but effective front-end
- Niche project but with a lot of scope for expansion



Spell entity – id (auto-increment), name (unique), level (0-9), school

13 methods in SpellController

11 implemented on front-end so far

Talk about expansion later in presentation

HOW WAS IT DONE?

- Jira project management
- Risk Assessment
- Version control with GitHub
- API built in Java
- JUnit automated testing
- HTML front-end with JavaScript integration
- Bootstrap/CSS styling
- Compiled into a .jar file

Week-long sprint, burndown chart

RA – matrix

GH – feature-branch model

API – Spring Starter project, written in Eclipse. Using Postman for debugging

Testing – JUnit and Mockito

SPRINT PLAN

- Broken down into Epics, Stories and Tasks
 - 6 Epics
 - 7 Stories
 - 38 Tasks
- Added User Stories and Acceptance Criteria
- MoSCoW priority
- Planning Point assignment

STV2-30 / STV2-1

As a developer, I want to create a functional API, so that it can be used to interact with the database

Done ✓ Done

Description

Acceptance criteria:

Given a user is interacting with the API,

When they try to use CRUD functionality,

Then the correct processes and database interactions occur

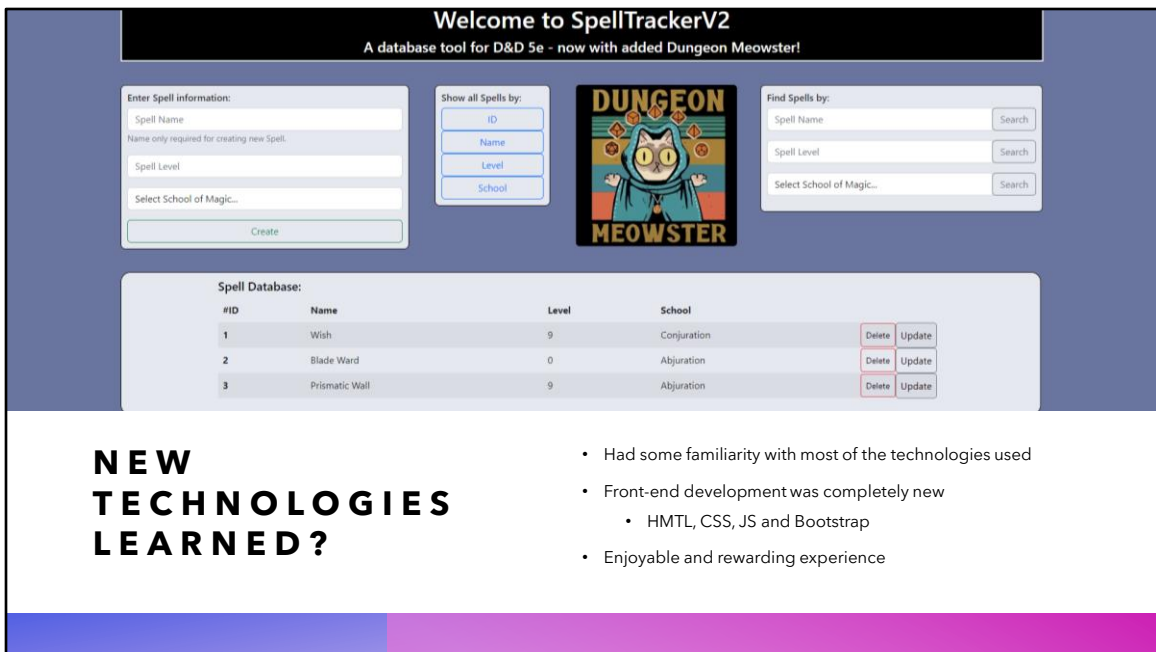
Linked issues

relates to

✓ SFW2-21	Create	⬆️	👤	DONE
✓ SFW2-23	Update	⬆️	👤	DONE
✓ SFW2-22	Read	⬆️	👤	DONE
✓ SFW2-24	Advanced CRUD	⬆️	👤	DONE

Epics:

- API
- Documentation
- Front-End
- Testing
- Version Control
- Jira



As this was building on previous work, I had used a lot of the technology used before to some degree

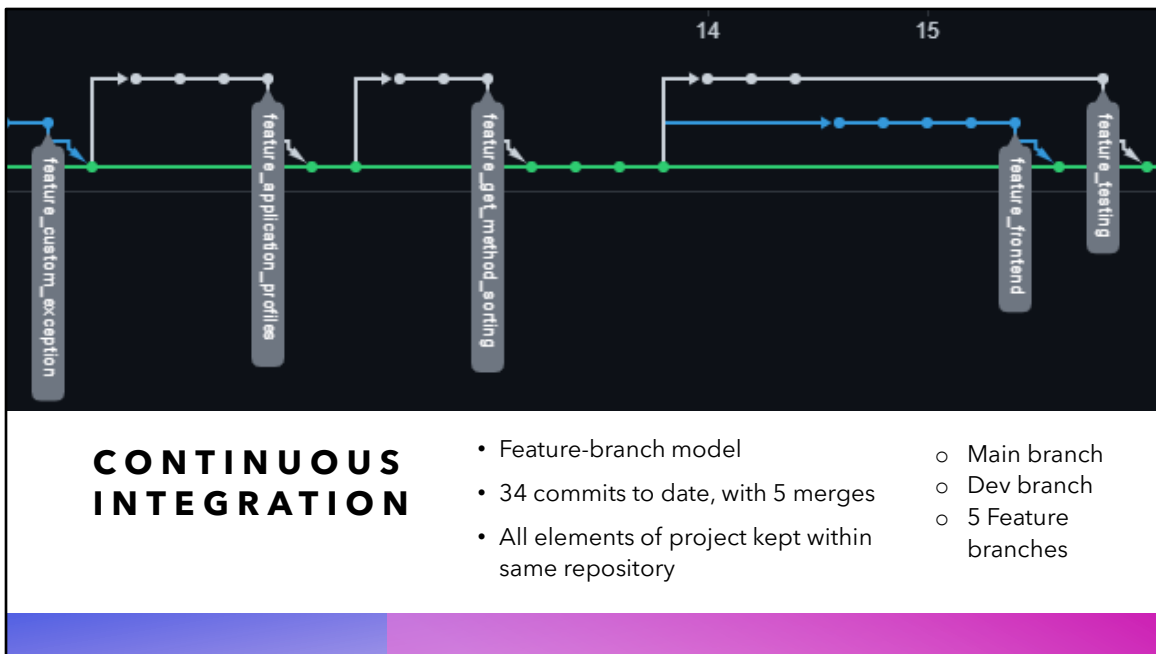
Picture of front-end

That's not to say that there weren't still plenty of smaller issues adapting it to the new project and getting everything to work well together!

Spent a whole day on getting the front-end functional and trying to improve the UI and UX

Challenging and rewarding to be learning something completely for scratch

Completing the project and getting it to all function has been very satisfying and enjoyable



All went to plan overall

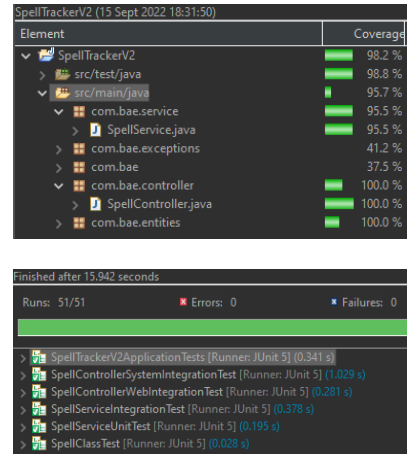
Picture of network insight diagram

Minor hiccup trying to merge with multiple active feature branches but was fixed relatively simply

Front-end files saved directly into Spring Starter project resources to keep everything together and backed up

TESTING

- Had to adapt previous testing
 - New auto-incrementing PK led to different behaviour
- 51 Tests
- Achieved 95.7% test coverage of src/main/java



The screenshot displays two panels from an IDE. The top panel, titled 'SpellTrackerV2 (15 Sept 2022, 18:31:50)', shows a tree view of code elements with their corresponding test coverage percentages. The bottom panel shows the results of a test run, indicating that 51 tests were executed successfully with zero errors or failures.

Element	Coverage
SpellTrackerV2	98.2 %
src/test/java	98.8 %
src/main/java	95.7 %
com.bae.service	95.5 %
SpellService.java	95.5 %
com.bae.exceptions	41.2 %
com.bae	37.5 %
com.bae.controller	100.0 %
SpellController.java	100.0 %
com.bae.entities	100.0 %

Finished after 15.942 seconds		
Runs: 51/51	Errors: 0	Failures: 0
[Progress Bar]		
SpellTrackerV2ApplicationTests [Runner: JUnit 5] (0.341 s)		
SpellControllerSystemIntegrationTest [Runner: JUnit 5] (1.029 s)		
SpellControllerWebIntegrationTest [Runner: JUnit 5] (0.281 s)		
SpellServiceIntegrationTest [Runner: JUnit 5] (0.378 s)		
SpellServiceUnitTest [Runner: JUnit 5] (0.195 s)		
SpellClassTest [Runner: JUnit 5] (0.028 s)		

Biggest issue with testing was adding a new id attribute to the Spell.class and using it as an auto-incrementing PK


Once testing methodology was adapted to this and new tests were added to cover interacting with objects by ID then it all worked out and very high coverage was achieved


A rectangular graphic with a purple-to-blue gradient background. The text "FRONT-END DEMO" is centered in white, bold, uppercase letters. The background features a subtle circular gradient pattern.

FRONT-END DEMO

Demonstrate interacting with database through front-end

SPRINT REVIEW

All Must-Have Tasks and Stories are on track to be completed

Of the Should-Have and Could-Have priorities, all are set to be completed

Went well overall

Room for some extra granularity with some stories.

Could have made more tasks to map progress of testing, rather than the majority having to be finished before checking it off

PROJECT RETROSPECTIVE

- MVP met and then exceeded
- All technologies worked with only minor issues
- Jira board and feature-branch model used well
- Room for expansion of ideas
 - Class attributes
 - Other entities
 - New methods
- Increased front-end functionality

Other classes such as user/character with one-to-many relationships

Expanded CRUD functionality to cover this

Additional front-end work to improve on UX as well as covering any other expanded ideas



Thank you to Anoush Lowton for always being quick to answer any questions I had and offer guidance and advice