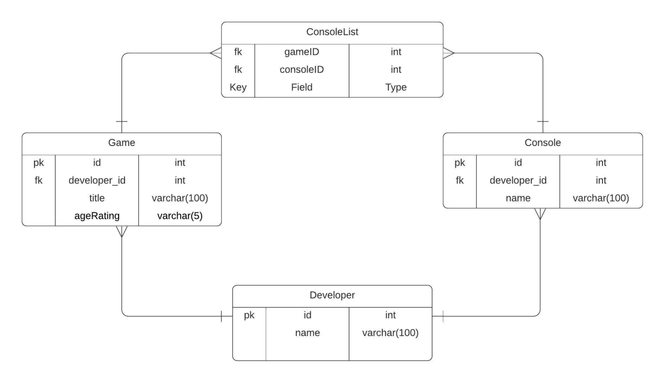
Hobby Website Application

Nathan Jackson

Technologies

- Front-End
  - HTML
  - Javascript
  - Bootstrap/CSS (Learned not Used)
- Back-end
  - Spring
  - Java
  - Maven
- Testing
  - J-Unit
  - Selenium
  - Mockito

Initial Planning & Approach



ConsoleList

fk	gameID	int
fk	consoleID	int
Key	Field	Type

Game

pk	id	int
fk	developer_id	int
	title	varchar(100)
	ageRating	varchar(5)

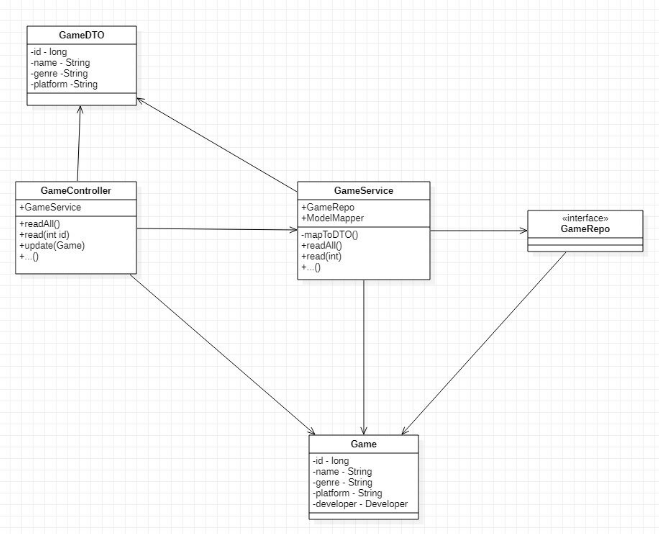
Console

pk	id	int
fk	developer_id	int
	name	varchar(100)

Developer

pk	id	int
	name	varchar(100)

- Initial Plan was for 3 Entity system based on Games
- Many-to-Many relationship between Console and Games



GameDTO
-id - long
-name - String
-genre -String
-platform -String

GameController
+GameService
+readAll()
+read(int id)
+update(Game)
+...()

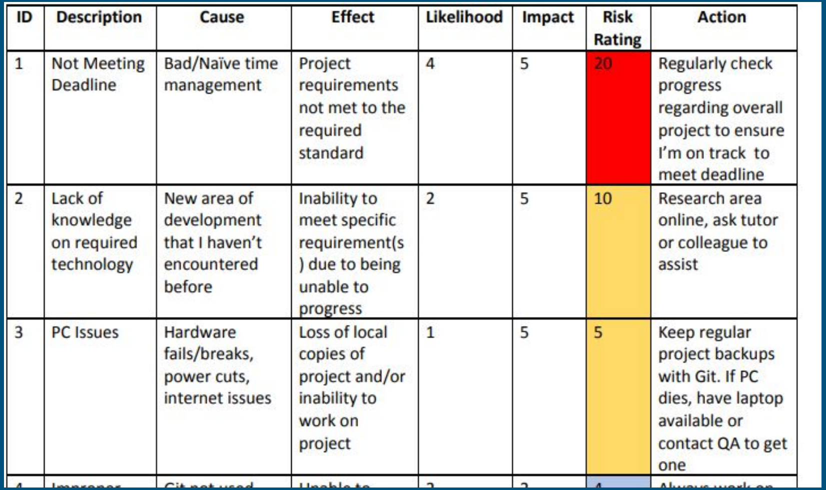
GameService
+GameRepo
+ModelMapper
-mapToDTO()
+readAll()
+read(int)
+...()

«interface»
GameRepo

Game
-id - long
-name - String
-genre - String
-platform - String
-developer - Developer

- Repository Design pattern
- Interaction between classes not modelled originally

Risk Assessment

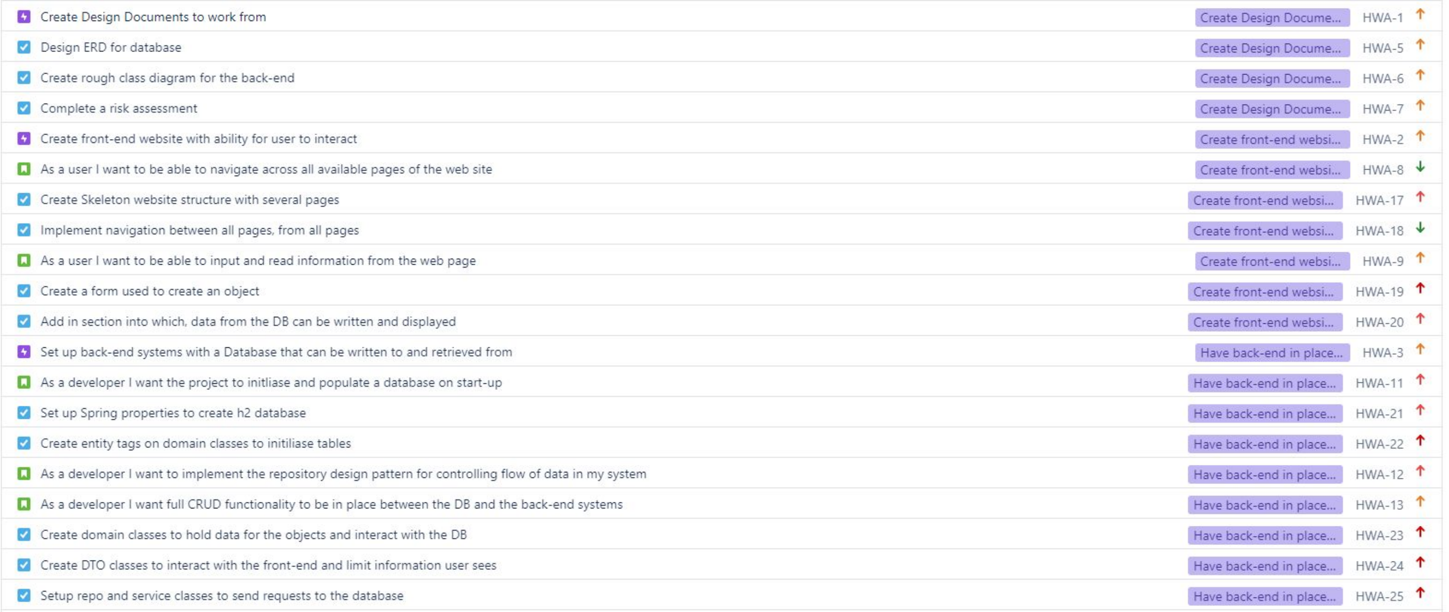


ID	Description	Cause	Effect	Likelihood	Impact	Risk Rating	Action
1	Not Meeting Deadline	Bad/Naïve time management	Project requirements not met to the required standard	4	5	20	Regularly check progress regarding overall project to ensure I'm on track to meet deadline
2	Lack of knowledge on required technology	New area of development that I haven't encountered before	Inability to meet specific requirement(s) due to being unable to progress	2	5	10	Research area online, ask tutor or colleague to assist
3	PC Issues	Hardware fails/breaks, power cuts, internet issues	Loss of local copies of project and/or inability to work on project	1	5	5	Keep regular project backups with Git. If PC dies, have laptop available or contact QA to get one
4	Improper	Git not used	Unable to	2	2	4	Always work on

- 7 total risks
- PC issues only risk to actualise
- Was able to recover with Git

Jira Board

- 4 Epics
- 8 Stories
- 18 Tasks



⚡	Create Design Documents to work from	Create Design Docume...	HWA-1	↑
☑	Design ERD for database	Create Design Docume...	HWA-5	↑
☑	Create rough class diagram for the back-end	Create Design Docume...	HWA-6	↑
☑	Complete a risk assessment	Create Design Docume...	HWA-7	↑
⚡	Create front-end website with ability for user to interact	Create front-end websi...	HWA-2	↑
🔖	As a user I want to be able to navigate across all available pages of the web site	Create front-end websi...	HWA-8	↓
☑	Create Skeleton website structure with several pages	Create front-end websi...	HWA-17	↑
☑	Implement navigation between all pages, from all pages	Create front-end websi...	HWA-18	↓
🔖	As a user I want to be able to input and read information from the web page	Create front-end websi...	HWA-9	↑
☑	Create a form used to create an object	Create front-end websi...	HWA-19	↑
☑	Add in section into which, data from the DB can be written and displayed	Create front-end websi...	HWA-20	↑
⚡	Set up back-end systems with a Database that can be written to and retrieved from	Have back-end in place...	HWA-3	↑
🔖	As a developer I want the project to initliase and populate a database on start-up	Have back-end in place...	HWA-11	↑
☑	Set up Spring properties to create h2 database	Have back-end in place...	HWA-21	↑
☑	Create entity tags on domain classes to initiliase tables	Have back-end in place...	HWA-22	↑
🔖	As a developer I want to implement the repository design pattern for controlling flow of data in my system	Have back-end in place...	HWA-12	↑
🔖	As a developer I want full CRUD functionality to be in place between the DB and the back-end systems	Have back-end in place...	HWA-13	↑
☑	Create domain classes to hold data for the objects and interact with the DB	Have back-end in place...	HWA-23	↑
☑	Create DTO classes to interact with the front-end and limit information user sees	Have back-end in place...	HWA-24	↑
☑	Setup repo and service classes to send requests to the database	Have back-end in place...	HWA-25	↑

Goals

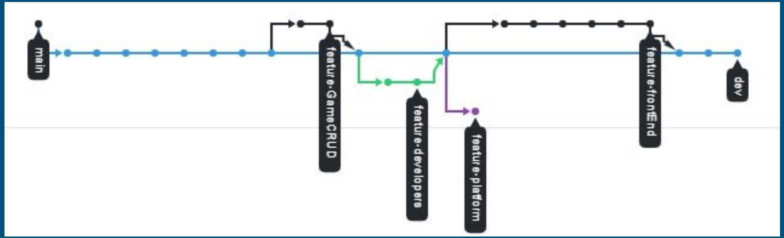
- Main Goals:
  - Front-End that enables user full CRUD functionality
  - Back-end that processes data going to and from the DB
  - Database created on started of program
  - Thorough testing of both website and back-end
- Stretch Goals:
  - Easy to use front-end
  - Feedback and data, output appropriately to web page
  - 3 Entities linked in the database

Version Control

Github used for version control

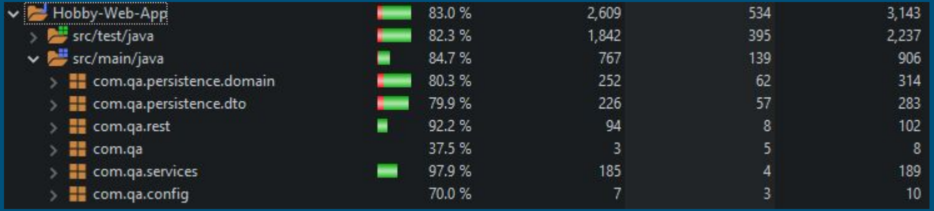
Branches used for each feature, platform entities in unfinished branch

Helped recover from PC reset



Testing

Back-end testing at 84.7% coverage



Hobby-Web-App	83.0 %	2,609	534	3,143
src/test/java	82.3 %	1,842	395	2,237
src/main/java	84.7 %	767	139	906
com.qa.persistence.domain	80.3 %	252	62	314
com.qa.persistence.dto	79.9 %	226	57	283
com.qa.rest	92.2 %	94	8	102
com.qa	37.5 %	3	5	8
com.qa.services	97.9 %	185	4	189
com.qa.config	70.0 %	7	3	10

Domain and DTO equals method not fully tested

J-Unit used with Mockito for Controllers

Front-End Testing

Selenium used to test website functionality

Very inconsistent tests with clickable buttons sometimes not registering

All functionality tested for both pages

Demo

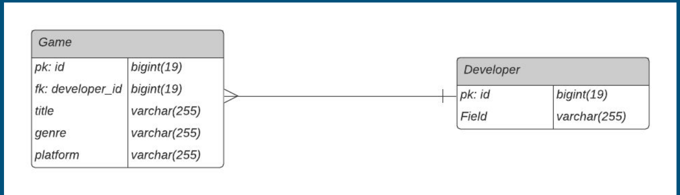
Review

What Got Done

- Full CRUD functionality for 2 linked entities
- Easy to use website for front-end
- Linked front end to access and modify DB through back-end

What Got Left Out

- Third Entity interaction
- Front End split through forms
- Complete Reliable Selenium Tests



Game	
pk: id	bigint(19)
fk: developer_id	bigint(19)
title	varchar(255)
genre	varchar(255)
platform	varchar(255)

Developer	
pk: id	bigint(19)
Field	varchar(255)

Retrospective

What Went Well

- All primary goals met in timely manner
- Version control helped revert and backup when needed
- Jira used to manage tasks well, linked with Git for smart commits

What Didn’t

- Errors in build path occured at some point
- Lack of thorough planning ahead of ERD
- Testing plan wasn’t thought through too well

Conclusion

- Generally Successful project
- Big improvements in process over last project
- More thorough planning for development and testing