Assignment 1

FIT5171- System Validation and Verification, Quality and Standards

Team -27

19th April 2020

Rebecca Christi (30824516)

Rashmitha Karkera (30438101)

Neel Shridharani (29882206)

Table of Contents

1. Testing Strategy

1.1 Introduction

This test strategy aims to describe the overall approach taken by the test team in the provision of test services rendered to ECM records. ECM, which stands for Edition of contemporary music, hails from Germany. We will create and test a web-based application that explores the marvellous world of ECM records.

This document details the project's high-level test plan. This test plan provides a framework for calculating the length and expense of the overall testing process, the nature and goals on which these calculations are based. Note that the test strategy is not a living document; it will provide the starting point for test planning. Upon completion and acceptance of the Test Plan, any improvements to, risks, problems, resource requirements, etc. will be handled by standard preparation and monitoring processes.

1.2. Test Scope and Objectives.

1.2.1. In Scope

The framework supports the following features:

- Presenting details about the album such as: Musicians, Genres, Year of publication, Songs etc...
- Modify the above entities.
- Analysis and exploration of above entities.

1.2.2. Out of Scope

Features related to security or Accessibility will not be developed or tested as part of this project.

1.3. Testing activities and deliverables

The following artefacts will be produced during the test phase:

- **Test Plan**: Used to define the scope, strategy, assets and schedule of the testing activities. The features to be checked, the people responsible for every task and the risks associated with this programme are to be described.
- **Test Cases**: Detail the pre-conditions, test steps and the expected test results. Positive and negative test cases may occur.
- **Bug Reporting:** Implement a bug reporting software that allows you to report, document, store, manage, assign, close and archive issues.

1.4. Testing Approach

- **Fully in Scope**: Unit, Functional, Integration, Regression and Automation.
- **Partially in Scope**: User Acceptance Testing.
- **Out of Scope**: Performance, Non-functional, Security, Compliance.

1.5. Test Environment, Infrastructure and Tools.

The ensuing specifies the environmental and infrastructural needs required for developing and testing the web application.

- **GitHub**: A modern VCS, a popular software hosting platform used by the team to collaborate in order to simplify the process of working amongst team members.
- **GitKaren**: A Git GUI Client, it helps developers to be more productive and efficient with Git.
- **SonarQube**: A web-based platform used to constantly measure and analyse the source code quality.
- **Intellij:** An IDE used here to develop the web-application, integrated with spark framework to develop web applications.
- Other tools such as: Neo4j, Freemarker etc...
- **Mantis Bug Tracker:** A free and open sourced bug tracking tool implemented to provision STLC in the project.

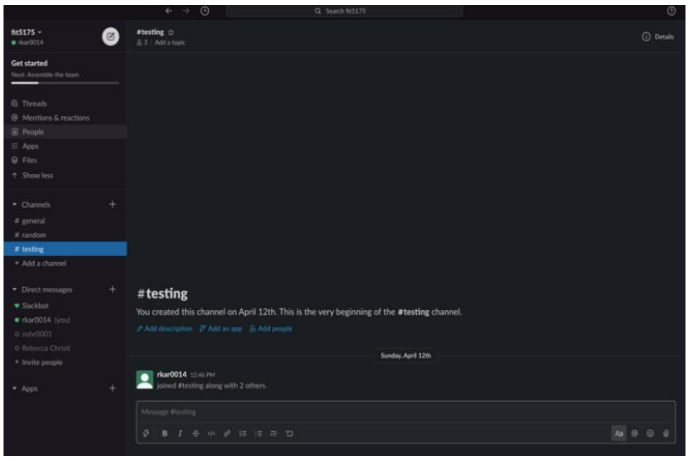
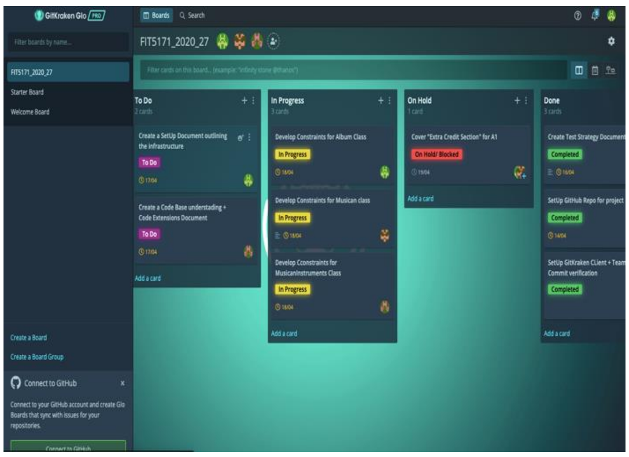
1.6. Risks and Contingencies

	Risk	**Mitigation Strategy**	**Impact**
1	Time delays to patch crucial bugs, which will need to be reviewed, may have an impact on project schedules.	In order to make sure bugs are patched and available for re-testing within the time limit, Development will require good bug resolution management.	High
3	Inter Team dependencies on acquiring domain knowledge, might cause a hurdle due to resource unavailability. The work on the project will be postponed.	It Is utmost necessary for the teams to be available at critical points or contactable during project deadlocks.	Medium
4	Certain features of Test Plan remain un-testable.	The test team tracks non-tested cases and asks the PM for business risks to be measured in favour of untested launches.	Low

2. Development & Testing Environment Setup

2.1. Communication Plan

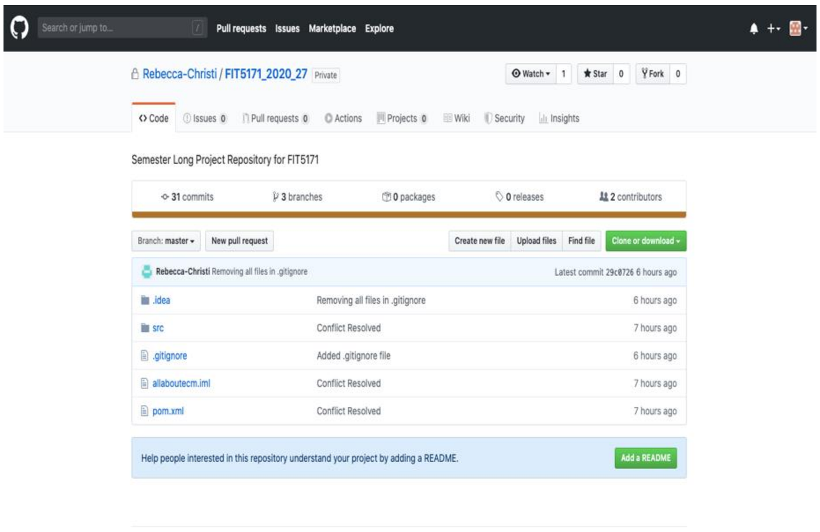
For the internal communication and collaboration among the team, “Slack” has been set up. Discussion on the progress has been shared across this platform. Similarly, for the task Management the team is utilizing the platform GLO BOARDS. A back up of all the files have also been stored in a shared drive in Google Drive.


#testing
You created this channel on April 12th. This is the very beginning of the #testing channel.


Slack and GitKarken Glo

GitHub

A remote repository is created in GitHub to commit the workflow of the Code Base and share them with the team. The repository is branched into 3 stages namely Master, Develop and Test. The master, by default is set as the Origin. Initial changes are updated onto the Develop Branch and later implemented in the Master in the final stage.



Rebecca-Christi / FIT5171_2020_27 Private

Semester Long Project Repository for FIT5171

31 commits | 3 branches | 0 packages | 0 releases | 2 contributors

Branch: master

Rebecca-Christi Removing all files in .gitignore — Latest commit 29c0726 6 hours ago

.idea	Removing all files in .gitignore	6 hours ago
src	Conflict Resolved	7 hours ago
.gitignore	Added .gitignore file	6 hours ago
allaboutecm.iml	Conflict Resolved	7 hours ago
pom.xml	Conflict Resolved	7 hours ago

Help people interested in this repository understand your project by adding a README.

Remote Repository

Using the URL https://github.com/Rebecca-Christi/FIT5171_2020_27.git . the repo can be opened in the local machine's Gitkraken. It is noticed that the GitHub and GitKraken will be in sync when they are cloned.

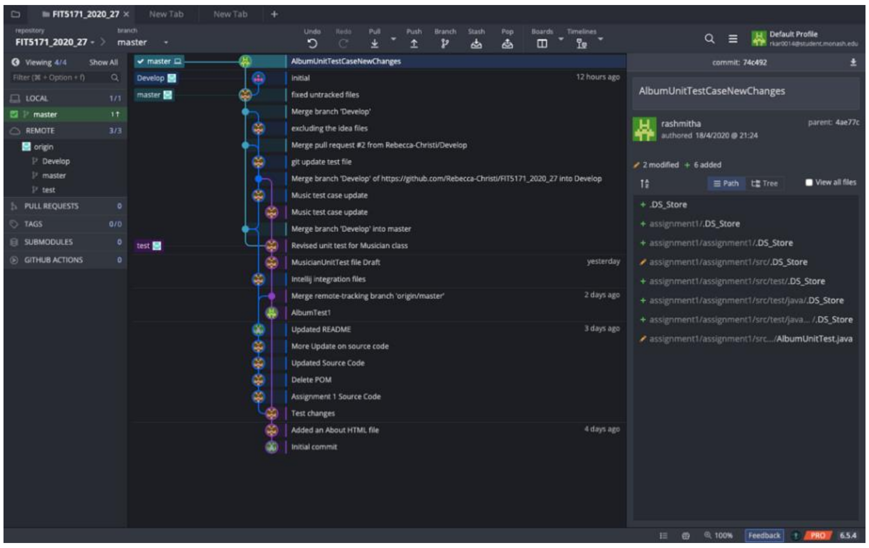
GitKraken

While cloning the URL of the remote repo in the GitKraken a certain conflict occurred by denying access to the remote repository since it was a private repository. This conflict has been resolved by upgrading to the GitKraken PRO version.

2.2. Resolving Conflicts

One major hurdle faced by the team was a conflict that arose while committing the changes into the develop branch. Since, Team Members used different Operating System , java versions in their local machine it created disruption while merging the files.. This issue was resolved by using gitignore . The gitignore ignores the configuration of Maven and JetBrains and only accepts the changes of the test files while pushing it into the Master. Also, The team commits code and assigns it to a representative. The representative reviews code before merging into the origin/master file.

The following snapshot depicts the successful setting up of the GitKraken in the local host.



```
AlbumUnitTestCaseNewChanges
Initial                                                    12 hours ago
fixed untracked files
Merge branch 'Develop'
excluding the idea files
Merge pull request #2 from Rebecca-Christi/Develop
git update test file
Merge branch 'Develop' of https://github.com/Rebecca-Christi/FIT5171_2020_27 into Develop
Music test case update
Music test case update
Merge branch 'Develop' into master
Revised unit test for Musician class
MusicianUnitTest file Draft                                yesterday
Intellij integration files
Merge remote-tracking branch 'origin/master'               2 days ago
AlbumTest1
Updated README                                             3 days ago
More Update on source code
Updated Source Code
Delete POM
Assignment 1 Source Code
Test changes
Added an About HTML file                                   4 days ago
Initial commit
```

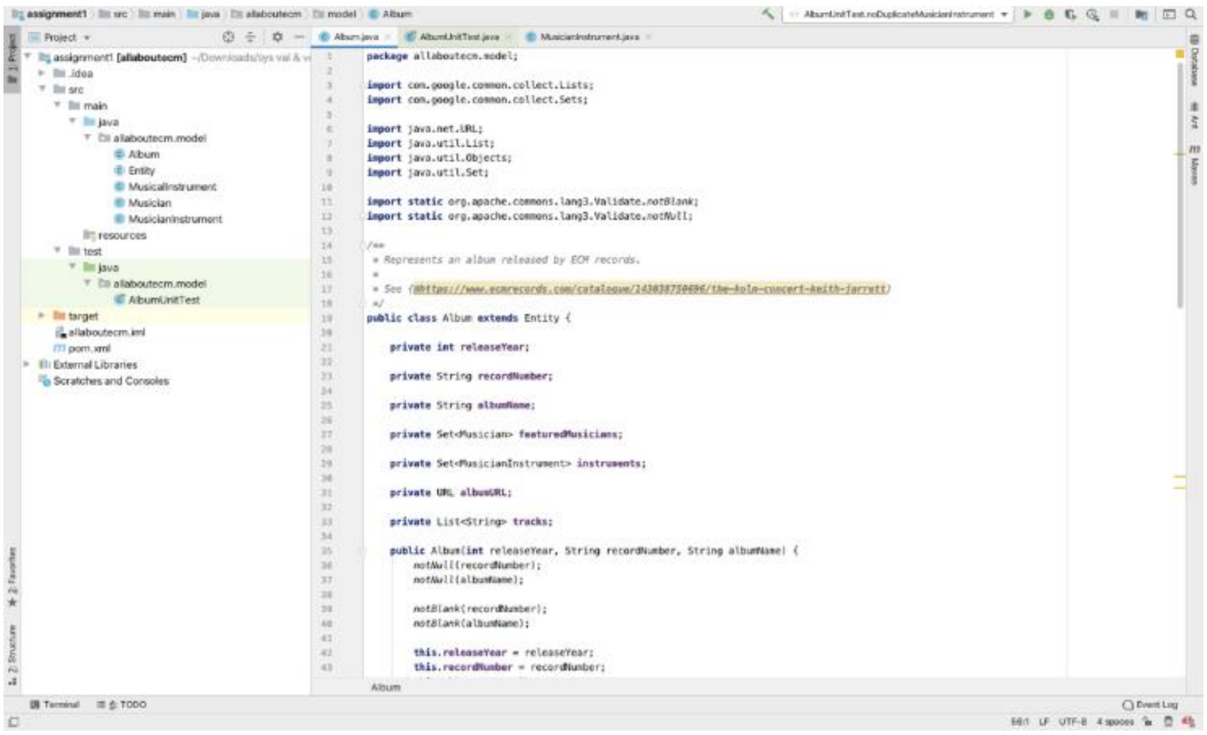
```
commit: 74c492
AlbumUnitTestCaseNewChanges
rashmitha                                     parent: 4ae77c
authored 18/4/2020 @ 21:24
2 modified  + 6 added
+ .DS_Store
+ assignment1/.DS_Store
+ assignment1/assignment1/.DS_Store
  assignment1/assignment1/src/.DS_Store
+ assignment1/assignment1/src/test/.DS_Store
+ assignment1/assignment1/src/test/java/.DS_Store
+ assignment1/assignment1/src/test/java... /.DS_Store
  assignment1/assignment1/src.../AlbumUnitTest.java
```

Gitkarken Pro Setup

2.3. Integrated Development Environment:

- The stage one is to install the **JDK (Java Development Kit version 8)** in the personal laptop/ computer.
- Having reviewed all software, the team came up with the unanimous decision to use IntelliJ Idea. The **IntelliJ IDEA** is downloaded from **https://www. jetbrains.com/idea/download**. The requirement for the project is to run the code in **Maven**. Since we are utilizing the IntelliJ as the IDE, the Maven is already built in the software.
- The next step is to import the file from the path of the remote repository saved in the local machine and open the code base in the **Maven**.

The snapshot below illustrates the successful set-up of the running code.



```java
package allaboutecm.model;

import com.google.common.collect.Lists;
import com.google.common.collect.Sets;

import java.net.URL;
import java.util.List;
import java.util.Objects;
import java.util.Set;

import static org.apache.commons.lang3.Validate.notBlank;
import static org.apache.commons.lang3.Validate.notNull;

/**
 * Represents an album released by ECM records.
 *
 * See {@https://www.ecmrecords.com/catalogue/143038750696/the-koln-concert-keith-jarrett}
 */
public class Album extends Entity {

    private int releaseYear;

    private String recordNumber;

    private String albumName;

    private Set<Musician> featuredMusicians;

    private Set<MusicianInstrument> instruments;

    private URL albumURL;

    private List<String> tracks;

    public Album(int releaseYear, String recordNumber, String albumName) {
        notNull(recordNumber);
        notNull(albumName);

        notBlank(recordNumber);
        notBlank(albumName);

        this.releaseYear = releaseYear;
        this.recordNumber = recordNumber;
```

3. Codebase Understanding and Extensions

3.1. Classes

The codebase contains class Entity which is extended by the classes: Album, Musical Instrument, Musician and MusicianInstrument class.

- **Entity class:** Is used to set and get unique dentification for each class that extends it.
- **Album class:** The class Album.java contains the following parameters: Name, Record number, Year, Instruments, Featured Musicians, Album URL and Track List.
- **MusicalInstrument class:** The class MusicalInstrument contains the parameter of Instrument name.
- **Musician Class:** The class contains parameters for: Musician name, musician url and object of Album.
- **MusicianInstrument Class:** This class passes Object of Musician and Musical Instrument as its parameters.

3.2. Test Classes

Described below are the variety of tests done for classes Album, Musician and MusicianInstrument class.

AlbumUnitTest Class

The class AlbumUnitTest.java contains test cases for Album class. It contains the following test cases:

- To check following parameters do not take ‘empty’, ‘null’ or ‘blank’ values: record number, album name

- Successful assignment of: Record Number, Release Year, Album name, Musicians, Tracks.
- Checking for duplicate in Album for: musician, track
- Checking if distinct instrument is assigned to musician.
- Parametrize test to check numeric values, alphanumeric and special character's for: record number, album name

MusicianUnitTest Class

The class MusicianUnitTest.java contains test cases for Musician class. It contains the following test cases:

- To check that following parameters do not take 'empty', 'null' or 'blank': Musician Name, Record Number, Release Year, Album Name.
- Parametrize test to check numeric values, alphanumeric and special character's for: Musician Name, Record Number, Album Name.
- Negative Test for: Release Year

MusicianInstrumentTest class

The class MusicianInstrumentTest.java contains test cases for MusicianInstrument class. It contains following test cases:

- To check that object being received is not 'null' or 'empty' for: Musician and MusicalInstrument
- To check duplicate is not added for: Musician and MusicalInstrument.

3.3 Assertions Used

In the test cases mentioned above the following assertions have been used:

- **assertsEqual**: Asserts that two objects are equal. If they are not, an AssertionError is thrown with the given message. If expected and actual are null, they are considered equal.
- **assertsThrow**: It throws an exception of the expected Type and returns the exception. If no exception is thrown, or if an exception of a different type is thrown, this method will fail.
- **assertsNotEqual**: Asserts that two objects are not equal. If they are not, an AssertionError is thrown with the given message. If expected and actual are different, they are considered unequal.
- **assertsTrue**: Asserts that a condition is true. If it isn't it throws an AssertionError with the given message.

4. Individual Contribution

Id	**Name**	**Contribution**
30824516	Rebecca Christ	33%
30438101	Rashmitha Karkera	33%
29882206	Neel Rahul Shridharani	33%