DD2480 Report: Assignment 4

Group 16
Authors: Henrik Åkesson, Anton Sederlin, Robin Eggestig, Robert Skoglund, Tsz Ho Wat

Project

Name: JabRef
URL: https://github.com/JabRef/jabref

JabRef is an open-source, cross-platform citation and reference management tool.

1. Onboarding experience / Effort spent on JabRef

Henrik Åkesson	Description
Onboarding experience	Straight forward. Well documented and worked as expected. No problems with onboarding with Linux on Mac with duel core intel.
Contribution	Searched for viable projects. Searched for non-trivial issues. Primarily tasked with overseeing the report. Added test log. Wrote part of reflection in context of Essence Standards.

Anton Sederlin	Description
Onboarding experience	Straight forward. The jabRef documentation regarding setting up a development environment devDocs is comprehensive and easy to follow. I had no problems with onboarding on my Macbook Air with the M1 ship. It all worked as expected the first time I tried to build, run, and test the project.
Contribution	Searched for viable projects. Searched for non-trivial issues. Researched code documentation and functionality of the selected project JabRef. Implemented solution to the issue selected.

Robert Skoglund	Description
Onboarding experience	Okay. Although instructions were clear and worked as expected, software constantly froze/crashed due to limited computer resources. Since the program is performance heavy - development was de-facto impossible on my computer for the chosen project.
Contribution	Searching, building, testing viable projects - involving software tedious installations and complications. Report writing. Essence review.

Robin Eggestig	Description
Project/Issue searching	Probably the worst experience so far. Main issue for me was finding actual issues that seemed doable. Some of the difficulties I found myself in when trying to finding one or more issues:

	• A ton of issues required lengthy discussions with the repo team to even decide what to do. • Some had a ton of people who worked on and off on it, leaving the issue in a state where it's unclear what's needed or not even after reading the whole discussion. • Missing labels, such as 'good-first-issue', or 'help-wanted' to indicate that it's not an issue targeted to the internal team. Same with missing labels like "needs triage", where issues would need discussion with or clarifications from the internal team to continue. • Most projects have very few new 'good-first-issue' issues. I noticed a pattern that most issues with this label, that are at least a month old or older, have a ton of problems being completed for a wide array of reasons (which was the major majority of open 'good-first-issue' issues). • Some project just aren't beginner friendly where there are very infrequent issues created, taken, and completed. • A lot of issues are more of a discussion rather than an actual problem to solve with requirements and wants. After having found some issues that were doable, the next problem that arose was the development environment. Since I use wsl2 on windows, a lot of hidden issues arose from that. For an example, see our selected project's onboarding below
Onboarding	I had a ton of problems getting a development environment up and running with this project. Since I'm running wsl2 on windows, my gradle was stuck on java 17, even when neither windows nor my subsystem had java 17 installed. Which is the reason I believe that my gradle had problems running which lead to my intelliJ not finding the source modules to be able to run it. I also tried vscode integration with docker as per the documentation, but that does not work either, even after network mounting the project folder. Eventually I had to settle for just browsing the code, without running, and helping to pair program with it.
Pair programming	Pair programmed with Anton for the radio button and the directory browsing functionality
Tests	Wrote about the type of tests which could be used with our implemented code.
Report	Helped with the report.

Tsz Ho Wat	**Description**
Searching	Bad Experience. There were a lot of projects we found and some of them were not able to be built, and the environment was difficult to set up on the WSL platform. The documentation took a lot of time to read, and programs took a long time to build on my computer.
Onboarding	No problems with building the current project, the document is clear.
Contribution	Test, build and review viable projects, draw the UML diagram,help with the report.

Total Time spent	**Henrik Åkesson**	**Anton Sederlin**	**Robert Skoglund**	Robin Eggestig	**Tsz Ho Wat**
Searching for viable project/issues	14h	15h	15h	12h	8h
Time spent on JabRef					
plenary discussions/meetings	4h	4h	4h	4h	3h
discussions within parts of the group	6h	6h	6h	3h	2h
reading documentation	1h	2h	3h	2h)	4h
configuration and setup	1h	1h	5h	4h	3h
analyzing code/output	-	6h	-	4h	5h

writing documentation/report	5h	1h	4	2h	2h
writing code	-	6h	-	4h*	2h
running code	-	5h	-	4h*	2h
Total	30h	46h	37h	39h	31h

* During pair programming

2. Contribution

As can be seen in the tables for each group member, the work took well over 20 hours per group member, this was mainly because of the extensive time spent finding a viable project. Finding a project we could all build on our own computers was the initial plan, but as time went on it became apparent that this was not possible within our time frame. The goal pivoted towards finding a project which was relatively easy to build on at least some of our computers, and the idea was to then pair program on the computers where we could get a successful build. **Anton Sederlin** was the one putting the most time on this, having looked at numerous projects such as Pandas, ElasticSearch, JabRef, metasploit, jsonParser, among others. **Henrik Åkesson, Robin Eggestig, Robert Skoglund and Tsz Ho Wat** put a lot of work on this as well, looking at projects such as Mockito, Scarpy, Cake and Docker, among others. While some of these projects did have successful builds, another issue we struggled with was finding a project with non-trivial issues, as well as an active community. We had early on decided not to continue working on our previous project, JavaParser, given our experience with working on this project in the last assignment. We had a not so pleasant experience with the onboarding, where even though the build did at times work on all OS we used, it was highly unstable and did most of the time not work for reasons we didn't quite understand, receiving errors which was difficult to interpret and not related to the coding we had done our selves. This had greatly slowed down our progress and we decided that a safer option would be finding another project.

Having decided on the JabRef project, **Anton Sederlin** and **Robin Eggestig** and **Robert Skoglund** found some viable issues, Dynamic Group Mirroring the File System #10930, Markdown rendering should keep {} and Handle 504 in DOI Fetcher. **Issue #10930** was selected given that it was extensive enough not to be trivial, was not assigned to anyone, and was clearly described as to what the requested feature entailed. **Anton Sederlin** and **Robin Eggestig** did the initial code documentation research and identified the relevant functionality behind the issue. Together with **Robert Skoglund**, they then worked on formulating a viable plan for solving the issue. **Henrik Åkesson** made a test log, which was used to compare test success rate before and after code implementation, this can be seen later in the report under the section “**Tests are automated, their outcome is documented”**. **Henrik Åkesson** and **Robert Skoglund** made an evaluation of our work process in the context of the Essence Standards. **Anton Sederlin** and **Robin Eggestig** made the tests for the software implemented to fix the selected issue for the project JabRef.

Repository	Description
Docker	Onboarding was well-documented, however did not cover non-mentioned environment issues frequently occured with *buildx* and *bake* dependencies, especially on MacOS. MacOS users installed the Docker Desktop app in hopes of easier problem solving - but issues remained. MacOS users attempted to sign-up / log-in to KTH's Cloud platform since it runs Ubuntu Linux, but login authentication issues occurred - which weren't solvable without contacting IT support.
Elasticsearch	Building/testing was flaky on all operating systems upon cloning. Spent too much time trying to debug the issue - and decided to drop instead. Also we quiicly noticed that the issue we selected to move forward appeared to already be solved and implemented although the issue was still open and had no linked pull requests.
Pandas	Different issues occurred here depending on the operating system, such as frozen builds, Docker container issues, dependency issues inside containers (especially the *pandas._libs.pandas_parser* module). The Pandas development could be done with a virtual environment or a Docker container - both of which were tested. Tests passed on Linux Ubuntu occasionally but were generally very flaky.
Scrapy	Very hard to find how to create the development environment to contribute with. The documentation for development was practically non-existent, while the documentation for the actual
Cake	This project was suggested since it is primarily using C# following the .NET framework and syntactically similar to Java, which we all have experience in. Building/testing worked on Ubuntu - however not for our MacOS users. In fact, there was an open issue that tested could not build for VSCode for Mac.
Mockito	This project is suggested because it is written by java and it can be built in MacOS and windows, the issues are active. However, it does not have enough labels and documentation in the issue page, it is difficult to know if the issue is big or not, and how it would affect the whole project. It is hard to choose the issue.
JavaParser	• The project, at least in terms of issues, is not very active. Latest issue was opened 3 weeks ago, and had 0 comments, next one was opened last month. Most recently closed issue was also 3 weeks ago (without any code contribution). This project is not a good candidate for open source contribution, since it is so inactive in addition to no good candidate issues. An example of this would be issue/#4303, which is more of a question than an actual issue that someone could grab and work on (Most issues are just questions, even if the question label is not used). • As we used this project in assignment 3, we already know the onboarding experience, which is horrid. This is due to the inconsistent breaking of build/test/run - sometimes it refuses to build/test/run after simply cloning the project, sometimes it refuses after a couple of changes, and sometimes you can make a ton of changes and all of the sudden it it just permanently broken again, even if you remove all your changes.

3. Non-trivial issue, motivation

The issue chosen was Dynamic Group Mirroring the File System #10930. This was mainly since the issue was described well and had not been solved or assigned to anyone else. The issue was a requested feature, wanting the ability to add nested folders with a specific hierarchy, where the hierarchy was to be mirrored and persisted on the JabRef platform. An example can be viewed in figure 1 below. Implementing this feature involves making changes and adding code to multiple functions, affecting several classes and making changes to the code structure which does not simply involve adding functionality to one existing function, which further motivated the relevance of this issue. Jabref is an open-source, cross-platform citation and reference management tool. Jabref helps users organize their references, find papers needed and discover existing research. While it is possible adding folders with references, the current version does not provide the ability to

add nested folders following a specific hierarchy, and instead adds the folders in separate *groups*. Groups are entries on the user's JabRef platform containing their stored references and other subgroups. Thus the user *can* create nested subgroups in the mentioned hierarchical way, but this is not done seamlessly when adding new folders. Instead, the user has to manually create their subgroups in a way which mirrors their local structure. Adding the requested feature would therefore enhance the user experience and alleviate a lot of work, since many users already structure their folders in this manner.

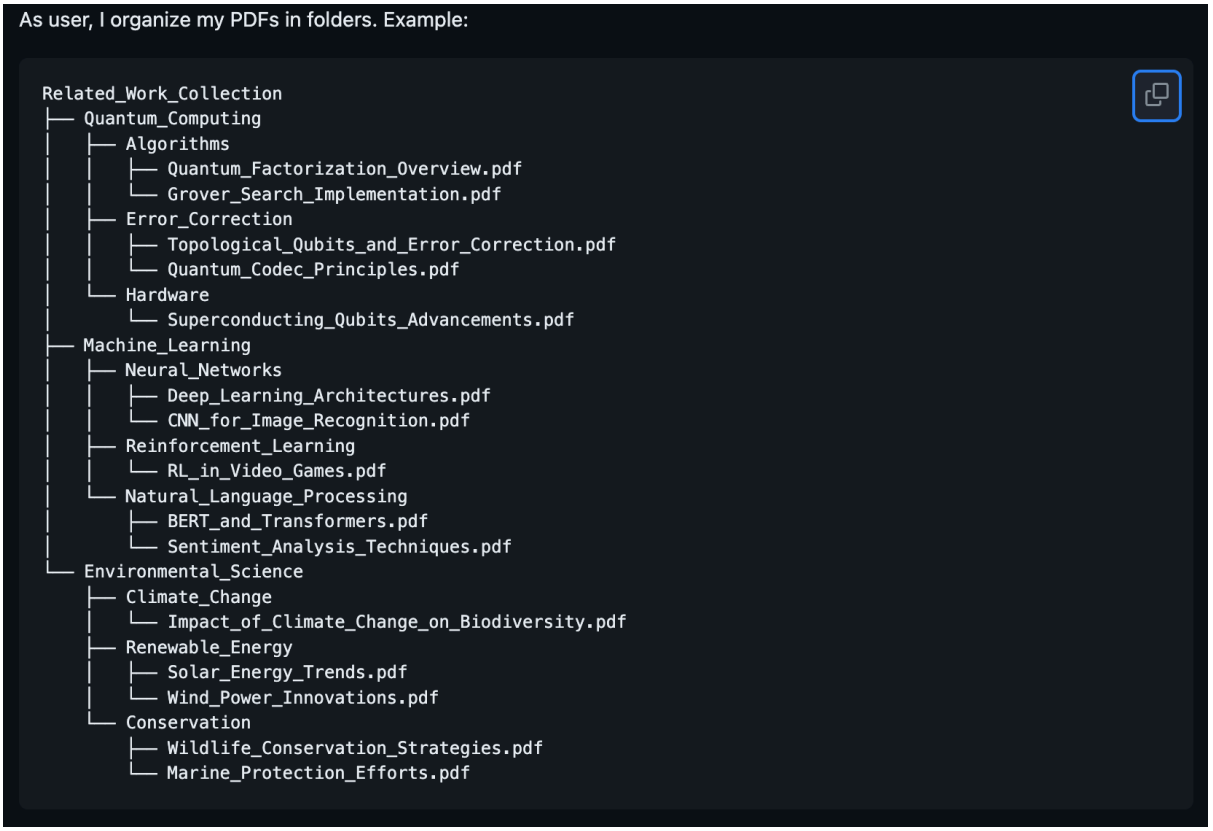


Figure 1. Example of a folder hierarchy, containing nested folders and pdf files. Figure taken as a screen-shot from issue on the issue tracker of the JabRef repository.

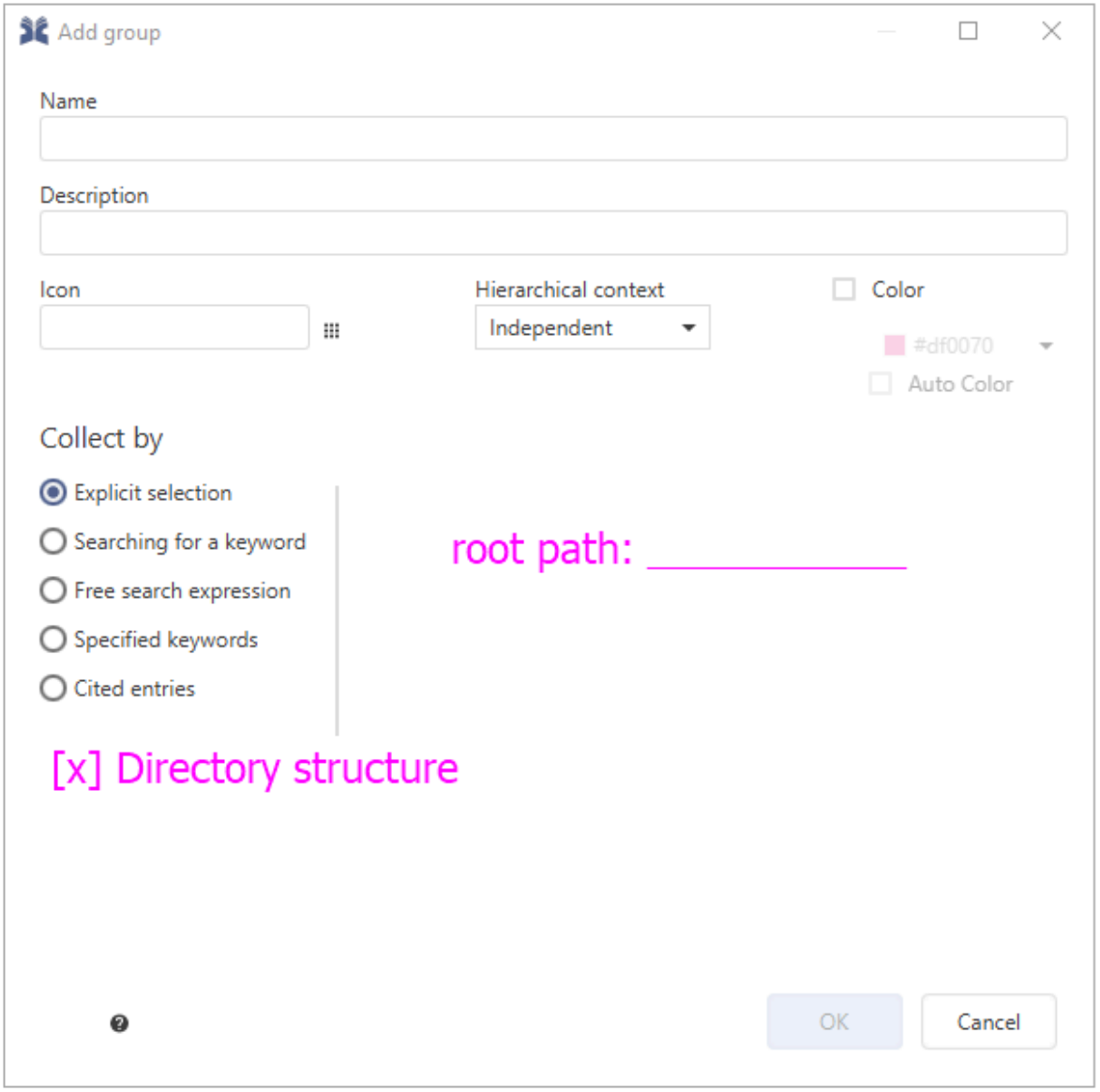


Figure 2: An overview of how the solution to the issue should be presented in the JabRef GUI according to the author of the issue.

4. Requirements related to the functionality

Requirements for the new feature or requirements affected by functionality being refactored

Name (ID)	Title	Description
issue#1	Browse folders	The user can browse folders in their local directory and add these to the JapRef platform
issue#2	Folder persistence	The hierarchical structure of the added folders are persisted and mirrored on the JabRef platform. The imported directory tree is translated into a corresponding group tree. Importing the directory home/user/softDev/covrage should result in softDev (group) -> coverage (subgroup) being created.
Issue#3	UI	There is a button for adding folders and browsing folders in the local directory. The user can select whether they want to persist the hierarchical structure (optional) with a radio button.
Issue#4	Continuity	The added functionality for adding folders that persist the hierarchical structure is in accordance with the current functionality of adding references and folders, adding to the existing structure. Reusing existing code and minimizing redundancy.
Issue#5	Documentation	The added functionality is well documented, letting future readers seamlessly work with and maintaining the added functionality.
Issue#6	Test suite	The added functionality is extensively tested with sufficient branch coverage. Building the project does not result in any errors as a result of the added functionality, effectively leading to a successful build.

5. Overview of issue(s) and work done.

Title: Dynamic group mirroring the file system.
URL: https://github.com/JabRef/jabref/issues/10930
Related issues:

- PDF file sync should create/use groups based on path koppor/jabref#501
- https://github.com/koppor/jabref/issues/502

In JabRef one can create groups and sub-groups for references. The new feature shall automatically create groups and sub-groups mirroring the directory structure chosen in the file system.

5.1 Trace tests to requirements.

There are a few things we need to test for. First when the button is selected, we want to test that the correct window is opened to the right. Since there are many different buttons with their own respective window, it's important to validate that the window that does open, is corresponding correctly with its respective button. Secondly, we want to test that only directories (i.e. no files) are able to be opened in the explorer - In our case this is simple because there are two different handlers for opening directories and opening files. Thirdly, we want to test that the validation of the directory works as intended, so that paths to non-directories or directories that don't exist return an error message. There are already tests in place to check the functionality of the actual validation in reference to allow/disallow continuing - That it won't allow you to continue if there are one or more error messages present. Remaining tests are out of scope for our current implementation, but what it would be to test that the groups and subgroups are created correctly compared to the directory chosen.

6. The architecture and purpose of the system

There are 8 packages in the main.java folder, each of them have different functionality. According to the documentation in the project[1], the `model` is the most important one, it is in the center. The `logic` is in the middle and `gui` is in the front. There are some utility packages for `preferences` and the `cli`. The dependency is only work between `logic`, `model`, and `gui` . There are JUnit tests to detect violations of the most crucial dependencies (between `logic`, `model`, and `gui`), and the build will fail automatically in these cases.

The `model` represents the most important data structures (`BibDatases`, `BibEntries`, `Events`, and related aspects) and has only a little bit of logic attached. The `logic` is responsible for reading/writing/importing/exporting and manipulating the `model`, and it is structured often as an API the `gui` can call and use. Only the `gui` knows the user and their preferences and can interact with them to help them solve tasks. For each layer, the project forms packages according to their responsibility, i.e., vertical structuring. The `model` should have no dependencies to other classes of JabRef and the `logic` should only depend on

[1] High-level documentation | Developer Documentation
https://devdocs.jabref.org/getting-into-the-code/high-level-documentation

`model` classes. The `cli` package bundles classes that are responsible for JabRef's command line interface. The `preferences` package represents all information customizable by a user for her personal needs.

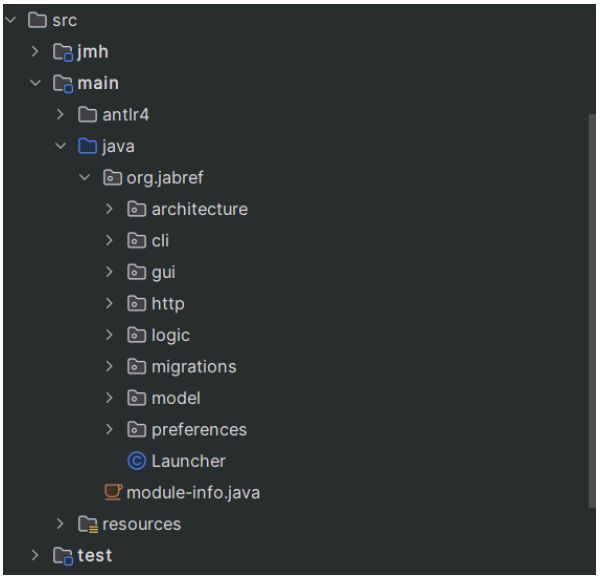


Figure 3: Screenshot of the dictionary of the source code

6.1 GUI package structure

Our issue is mainly focused on package `gui`. In the JabRef GUI package structure, the user interface stands at the forefront, interacting directly with the `logic` layer, which, in turn, communicates with the central model. This layered architecture ensures a clear separation of concerns. The GUI package not only encapsulates user-related functionalities but also orchestrates the flow of information between the `logic`, `model`, `preferences`, and `cli` packages. This modular design fosters maintainability and extensibility, allowing the GUI to evolve independently. The organization into vertical packages reflects a disciplined approach, with the GUI holding exclusive knowledge of user preferences and seamlessly orchestrating tasks through interactions with the underlying logic and model layers.

```
gui --> logic --> model
gui ------------> model
gui ------------> preferences
gui ------------> cli
gui ------------> global classes
```

Figure 4: diagram about how GUI package works

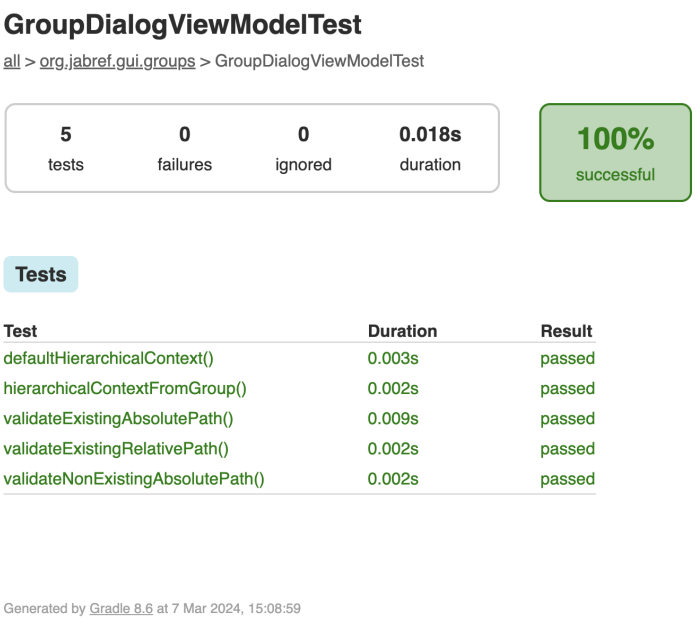
7. Code changes

7.1 Patch

To view our patch:

- Pull Request
- Our clone our Fork and run: git diff fix-for-issue-10930 main

7.2 Test results



GroupDialogViewModelTest

all > org.jabref.gui.groups > GroupDialogViewModelTest

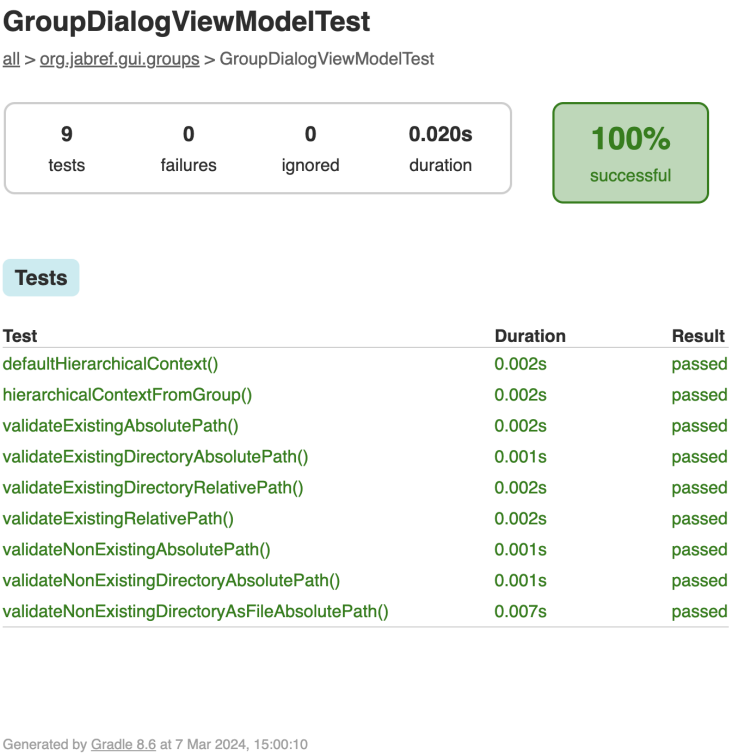
5 tests	0 failures	0 ignored	0.018s duration	100% successful

Tests

Test	Duration	Result
defaultHierarchicalContext()	0.003s	passed
hierarchicalContextFromGroup()	0.002s	passed
validateExistingAbsolutePath()	0.009s	passed
validateExistingRelativePath()	0.002s	passed
validateNonExistingAbsolutePath()	0.002s	passed

Generated by Gradle 8.6 at 7 Mar 2024, 15:08:59

Figure 5: Before adding new feature



GroupDialogViewModelTest

all > org.jabref.gui.groups > GroupDialogViewModelTest

9 tests	0 failures	0 ignored	0.020s duration	100% successful

Tests

Test	Duration	Result
defaultHierarchicalContext()	0.002s	passed
hierarchicalContextFromGroup()	0.002s	passed
validateExistingAbsolutePath()	0.002s	passed
validateExistingDirectoryAbsolutePath()	0.001s	passed
validateExistingDirectoryRelativePath()	0.002s	passed
validateExistingRelativePath()	0.002s	passed
validateNonExistingAbsolutePath()	0.001s	passed
validateNonExistingDirectoryAbsolutePath()	0.001s	passed
validateNonExistingDirectoryAsFileAbsolutePath()	0.007s	passed

Generated by Gradle 8.6 at 7 Mar 2024, 15:00:10

Figure 6: After adding new feature

7.3 UML class diagram and its description

https://drive.google.com/file/d/1Zb9Da4L9HCMWZxZ0ds5chl5iJs1QfmfR/view?usp=sharing

8. Experience evaluation

8.1 What are your main take-aways from this project? What did you learn?

This project has been a reality-hit of the nature of open-source. The issues we have been facing have been vast and real with varying problems such as flaky builds, inappropriate issues, environment configuration, etc. Excited by the idea of contributing to open-source projects, we wanted to find an interesting project in which we could make a difference. After a terrible experience with our previous project, JavaParser, we unanimously set forth to explore new projects - only to realize very similar issues. One large take away for this project, but in fact all previous projects, is the importance of environment standardization and smooth onboardings. Although we attempted to create standardized environments with virtualization via Docker containers, getting Docker working was incredibly time-consuming and in the end, unsuccessful. With a clean and comprehensive onboarding, the time from set-up to development can be drastically minimized. This decision-making and time-planning has also been a main take-away - should we continue spending time on debugging one project, or instead search elsewhere for another project? I think this course gives tangible insights on the importance of strong software engineering practices.

8.2 How did you grow as a team, using the Essence standard to evaluate yourself?

Essence team state: *performing*. Although our team has fulfilled the requirements to reach the state of performing on the Essence team scale, the way-of-working performs quite differently. We as a team have been able to continually adapt to our situation, meet our agreed commitments, work together with open communication and no outside help. Once we finally chose a project, we made efficient progress, however the onboarding of all the attempted projects was not a pleasant experience. Spending too much time on possible onboardings with poor environmental configuration manuals and flaky behavior was arguably a waste of time - although naturally we gave it our best. Throughout the projects, I think we have remained at this stage, however soon we may reach adjournment since the course is fishing.

9. Tests are automated, their outcome is documented

9.1 Test log in terminal on IntelliJ:

Test log.

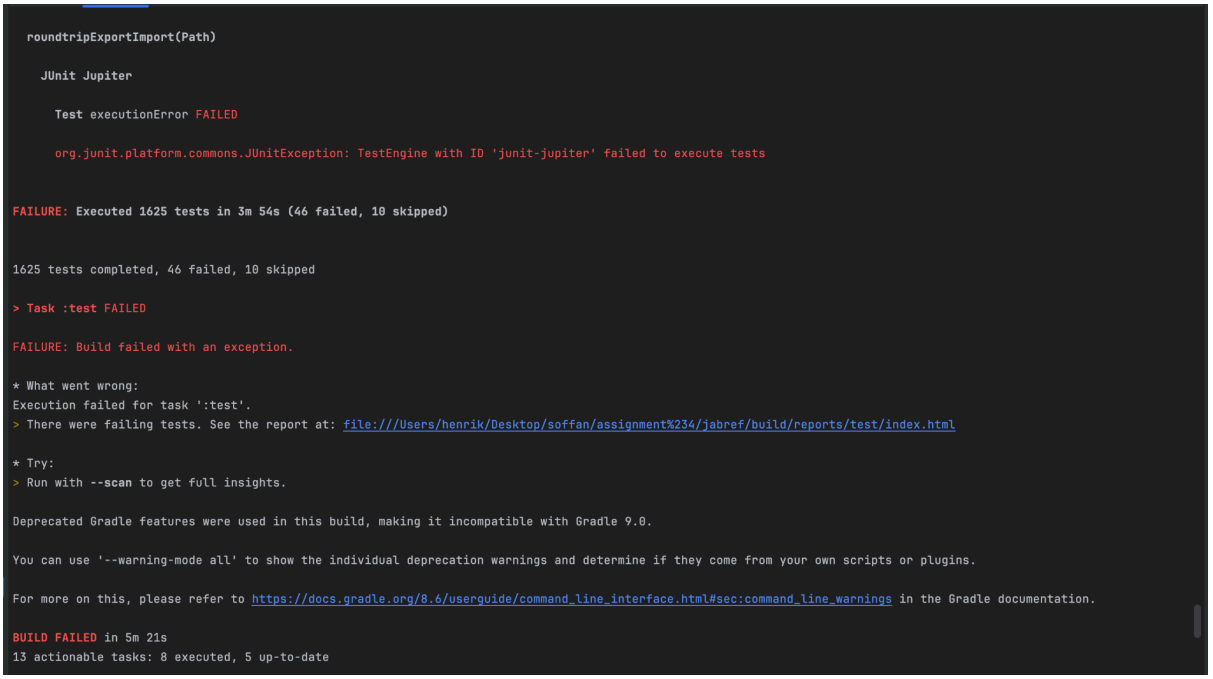


Figure 7: Build result, failed due to failed tests *before* group code implementation.

10. You can argue critically about the benefits, drawbacks, and limitations of your work carried out, in the context of current software engineering practice, such as the SEMAT kernel (covering alphas other than Team/Way of Working).

The following is a discussion about benefits, drawbacks and limitations in the context of the Essence Kernel standards, with respect to alphas.

Alpha: Software Systems

The essence alpha “Software systems” focuses on the software being built through the development process. States and checklists can be used to assess the development stages of the software, identifying areas which might need improvement and guiding the process. In our development process we did not explicitly adapt a checklist and states approach, but did this more in an abstract form. What this meant in the context of our work, was that we discussed the steps which needed to be performed to reach a viable solution to solving the issue at hand, and during the development assess how far we had come in this process, and what needed to be fixed. The benefits of this approach was mainly that we had an overview of what needed to be done, both currently and in the future. It also had the benefit of drawing a clearer picture of the time that would probably be needed to complete the software, and how far we had progressed. Later on, we did apply a more substantial checklist, which can be seen in the report listing the sub-issues related to solving the issue chosen for the project. Identifying these sub-issues had the benefit of giving guidance throughout the

development process, and also let us more easily divide work between the members of the group. Potential drawbacks might have been the time invested, as well as not having the complete knowledge of the project structure. This led to some difficulty in mapping a correct checklist for what needs to be done, and how this goal was to be reached. The limitations of our process was partly due to the limited projects we had to choose from, not because there were few projects, but because there were many projects which simply did not build or had other issues not making them viable projects to choose. This resulted in a more limited choice of architecture, where for example the language to program in was selected based not on our preferences, as much as on what language was used for the project.

Alpha: *requirements*

The alpha *requirements* define what the system must do - but not necessarily how - in order to fulfill the expectations of the stakeholders. Applying the requirements alpha to our specific issue requires a detail-oriented description due the high magnification - relative to the requirements set on the application/project level for JabRef. In relation, the issue we chose, we defined our requirements based on the functionality of the program as well as on code quality, i.e. good documentation, testing, etc. We believe we reached the requirement state of *Acceptable* - we detailed requirements demanded by the issue in the context of the project, but did not reach the next state *Addressed* since we did not implement the functionality to a usable level by the end-user. The benefits of our detailed requirements are that the community can understand how we addressed the stakeholder's need, the new mirroring button/functionality, in an explicit manner. From these requirements, the stakeholders understand how we approached our work at a lower level. The potential limitations/drawbacks of detailing the requirements in this issue is the niched, developer-level understanding of the code - irrelevant to the 'higher-level' stakeholders such as the end-users. In this context, who the stakeholders are is arbitrary as we are serving the end-users but also the developers. Another obvious limitation is that we did not fulfill all the requirements we specified. In a complete solution obtaining the state of *Fulfilled*, all requirements specified would be accurately addressed an officially accepted by stakeholders by means of review.