Prospectus - DQGUI

Team Description / Profiles

Our team consists out of five members, two students from the Texas A&M University and three from the University of Applied Sciences and Arts Hannover. The following part lists the individuals involved within the project and a brief summary about them.

German Side

Member: Julian Sender (31 years old)

Studies: Applied Computer Science
College: University of Applied Sciences and Arts Hannover
Technical strengths: I have experience with Java, small databases and have already written a small system for warehouse management in C#.
I also have basic knowledge in web development of the common technologies. My greatest interests and strengths include software architecture and administrative and organizational work. They also include statistics and number interpretation.
Anticipation of Growth Areas: I hope to gain some exciting experience in international software development from this project. I would like to improve through the project in the following areas: English in professional practice, team organization and communication as well as SCRUM.

Member: Marc Herschel (21 years old)

Studies: Applied Computer Science
College: University of Applied Sciences and Arts Hannover
Technical strengths: Java, JavaFX, GUI building with Java, Linux/Bash scripting, Python, Eclipse, SQL, Database interaction from within Java, some software engineering knowledge (Architecture, Patterns)
Anticipation of Growth Areas: Gaining practical experience from a real life setting, international experience and usage of English within a project, first time using an actual methodology to build software (SCRUM), Professional communication

Member: Daniel Diele (27 years old)

Studies: Applied Computer Science
College: University of Applied Sciences and Arts Hannover
Technical strengths: My technical strengths range from Java, JavaFX, web development, and to a lesser extent databases and their interaction with Java, basic knowledge of software engineering regarding Patterns and.
Anticipation of Growth Areas: I intend to improve in the areas of communication in English in a professional setting, teamwork and soft skills in an international context, agile software development (SCRUM) and Java GUI programming in general.

US-American Side

Member: Grant Singleton (27 years old)
Studies: Computer Engineering
College: Texas A&M University
Technical strengths: Object Oriented Programming in Java and C++. Web Development to include HTML, CSS, Javascript and ReactJS. Mobile App development in React Native.
Anticipation of Growth Areas: JavaFX and SQL.

Member: Kevin Duan (21 years old)
Studies: Computer Engineering
College: Texas A&M University
Technical strengths: Object Oriented Programming in C/C++ and Java. Web Development primarily in JavaScript/ReactJS. Extensive team leadership experience and professional experience leading in an Agile Scrum Environment
Anticipation of Growth Areas: A deeper understanding on Databases and Java, which I have less experience in.

Project Description

Quoted from the proposal:

"*The Trust@HsH research group has designed and implemented a data quality monitoring system for data warehouses in a research project concluded in 2018. The core of the system is a data quality monitoring language that enables DQ managers to specify rules for data quality.*"

The aim of this project is to create a cross-platform GUI that serves pretty much as an IDE for researchers of the IQM4HD project and Data Quality managers using the software in production. It will enable the researchers to work on parts of the Data Quality language without resorting to a complicated microservice based development environment. The IDE will allow for database connections to be managed and Data Quality rules to be defined, modified and executed via these databases. It will feature syntax highlighting and most likely also code completion. The creation and management of Data Quality language sources, checks and actions will be abstracted by the application, so users do not have to worry about placing the files in the correct folders. It will also feature a tabbed text editor that allows to work on multiple IQM4HD files at the same time. The reports that will be returned by the IQM4HD project will also be archived so analysis of historical data is possible as well.

Anticipated Platform and Tooling

Platform: Java 8 (Oracle JDK) with JavaFX 8 and JavaFXs FXML technology.
VCS: GitHub Private Repository
Database: SQLite for local storage.

Technologies: JSON for local user settings and export of saved data.
Tools: Eclipse and/or IntelliJ, Gluon SceneBuilder, ZenHub, TeamSpeak 3, TeamViewer, Slack, Trello

Ethical Considerations

Since the IQM4HD application analyzes the data for us we have no ethical concerns regarding violating the privacy of a company or their customers. We will only work with reports that the IQM4HD application sends us back, these reports contain a rating of the data quality but not the actual data.

We do however have concerns regarding the sensitive information that is required to access the databases that the IQM4HD application works with. Storing these information as clear text on the file system will allow other unauthorized persons to access the databases and thus the data that we have nothing to do with. Cases where this might happen are espionage and a compromised host system.

To circumvent this issues we have planned to add a voluntary encryption of the database credentials which guarantees that these sensitive information will never be stored in clear text. The user will have to enter their passphrase once they use the database credentials the first time to decrypt the local file but can then use all these connections until they close the application again. The passphrase will be kept inside memory until runtime termination so changes to the existing database connections can be saved to disk and also be encrypted.

Backlog (Not important after discussion with Mr. Heine / Important after discussion with Mr. Heine)

File system related:

- Prompt to greet user on first run to specify repo directory.
- Check on every start to verify if repo directory is valid.
    - On error initiate wizard to specify new directory.
- Once repo directory is established abstract file management of DQ rules (create/modify/delete) onto the categories sources, actions, checks.
    - The user only has to specify which category they want to create a new DQ rule in.
    - The DQ rule will be created inside the corresponding folder within the repo directory with the fitting extension.
- Abstract repo operations via an interface so we can implement it for the file system but the iqm4hd developers could also create a solution where the repo is within a database (Example RuleService interface in the iqm4hd application which we'll implement).
- On load of the application the tree menu will be populated with the existing DQ rules from within the repo directory and split accordingly into their corresponding categories.
    - Only files with the correct extensions will be loaded and treated as DQ rules.
- Ability to delete DQ rules and save changes to them.
- Backup export option where the repo dir is zipped, given a time stamp and stored in a backup folder.
- Local storage option (YAML, JSON, plain text??) in a hidden folder to save database connections in.
    - Modification/creation abstracted via database connection manager.
- Possibility to encrypt local file containing database connection credentials.
- Adding/deleting/searching logged iqm4hd actions in a local sqlite database.

- Simple global logger that can be used while developing the application so debugging is made easier and can be disabled quickly once run in production.
- R integration: Starting, monitoring and babysitting the rserve instance via the shell script.

Database connection related:

- Wizard to add a new database connection.
    - Specified databases will be supported.
    - Currently only supporting via direct IPs.
    - All necessary fields will be included (mostly address, username, password, database).
    - Option to test connection at the end of the wizard before saving it.
    - Wizard should be extendable for new databases.
- On load of the application the tree menu will be populated with the existing database connections.
- Right clicking an existing database connection offers the options of delete/modify/test/use.
- Local connection testing will either yield in a success or a detailed error message on why the connection might have failed.
- Solution to pass over database connections to IQM4HD
    - If the connection is invalid and not tested locally we can only deliver the error that iqm4hd delivers us.
- Possibility to voluntarily protect database file with a password so database related information like usernames, passwords and addresses are not in clear text on the local file system.
    - If chosen, the user will be prompted with a decryption wizard, as soon as they either test/modify/delete/use a database connection or run the tool.
    - Once decrypted the tool will not ask for the password again for the rest of the runtime.
    - Option to lock database related information again if a user wants to leave the tool open but not the database freely exposed.
        - Same procedure as above, unlock wizard will be started as soon as a database connection is needed.
- Grouping of database connections in environments.
    - If no environment exist all shall be put into a default environment.
    - Environments are a supersets of database connections.
    - Switching environments will exchange all database connections with iqm4hd.
    - Example:
        - Two environments (test, production) both contain a customer and contract database connection.
- Implementation of the DatabaseService interface that allows us to store the database credentials existing in the selected database environment. The DatabaseServiceTestImpl.java is an example of that interface implemented, that one however relies on hard coded values and is not feasible by our application.

IQM4HD action execution related:

- A user will be able to select a database environment and an action to execute a dropdown menu.
- Ability to favourite a combination of database connection and action and add to the tree menu for quick access.
- The gui will then try to pass the database information and action to run to iqm4hd.
    - If the iqm4hd path is not specified a wizard will open to specify the location.

    - If the action runs successfully and passes - a console window within the application will output the test results and the status bar will display success.
    - If the action fails because of an iqm4hd exception the same will happen, but the output will be colored red and the application status bar will display that an exception happened.
        - If the exception is syntax related we will try to mark the syntax error within the iqm4hd file IF it is opened currently using the position and error message given in that exception. Otherwise it will be displayed in the console as well.
    - If the test runs successfully, but iqm4hd marks the report as not passing the console output will be orange and the status bar will display that the action ran successfully but did not pass.
- Ability to log iqm4hd actions (reports created by the application) to a local database so historical data can be analysed by a third party if necessary.
    - Ability to export in JSON so data analytical systems can easier work with the data.
    - Ability to filter, so iqm4hd runs that failed duo an exception will not be logged.
    - Logging will be controlled via a checkbox that can be turned off/on and will remember the last choice.
- Execution of an action should be run in the background as this can take up a long time.
    - There should be one UI element where currently running actions can be stopped and their status can be viewed.
    - Once an action finished maybe we can add a notification?
    - The UI element should show finished and canceled actions.
        - Canceled/Finished actions should have the option to rerun.
        - Option to clean list of canceled/finished actions
    - It's not possible to implement a status as iqm4hd does not support this.
        - Valid states will be running, canceled, failure (iqm4hd crash), finished.
- Profiling (should be discussed with Mr. Heine later on)
- Implement a basic client via the Iqm4hd.java API that is useable by the application.

UI related:

- Lots of keyboard shortcuts so an efficient workflow can be established.
- Tabbed text editor so multiple instances of DQ rules can be opened and worked on at the same time.
    - Only one tab per rule of course.
- Ability to remember opened tabs after restarting the application.
- Syntax highlighting, if possible also simple auto code completion.
- Primitive undo/redo operation for text editor if possible.
- Clear, self explanatory UI that allows an efficient workflow.
- Settings page that allows for changing the iqm4hd location, the repo directory and some other application relevant settings.
- Repos can become pretty large.
    - Search, grouping should be implemented.
    - Trees maybe not the best way to deal with them.
        - One option: Tabbed Menus to split Sources, Actions and Checks into their own tab and include a filtering option.
        - Database Environments will still be managed by a Tree Menu in an extra tab.