Group MCQJ Project Document

Group Members:

Conor Keegan, Joshua Arquette, Max Gerber, and Quoc Nhan.

Project Management Methodology:

To best suit a group software development project, our team will be using the Scrum methodology for the upcoming tasks throughout the semester.

Standup Meetings:

Our team will try and meet every day / every other day, mostly through discord, but possibly in-person after class time as well. This will be mostly to catch up on any work we've completed, and will only last short periods of time.

Product Backlog:

Our product backlog will be created early on in the semester. Since we already have a list of milestones, we will simply create a list of needed stories and epics on jira.

Sprint Planning:

Sprint planning will be done on discord, where our group will discuss requirements for upcoming tasks / milestones, and will come up with at least a minimum set of tasks / stories to be added to the sprint backlog on jira. Tasks will probably be estimated based on previous assignments in the course, and will also take into account how long the assigned developer feels they will take. We will use the point system in jira to officially estimate each task.

Sprint Review:

Sprint reviews will take place on discord, where our team will discuss what we've completed / were not able to complete. We will also take this time to view changes / statuses of stories on our jira project, and make sure they are updated according to our current progress.

Sprint Retrospective:

Our sprint retrospectives will take place on discord, where we will discuss any difficulties the team ran into during our last completed milestone, and we will try and solve them. Communication issues, hard-to-read code, misunderstanding the current milestone's objective, all of these problems will be discussed.

Meeting Times and Communication:

Our team will communicate almost entirely through discord, sometimes communicating in-person after class if needed. Weekends are our best fit timewise, as most of us will be available then. The same goes for Monday-Wednesday, where the whole team will be available around 7 - 8 P.M. Our communication will mostly be loosely defined, and realistically, we will probably communicate at chosen times throughout the day, at least every other day, when we find the time to do so.

Project Management Software:

Our group will be using jira to manage Scrum-related processes. This is where we will keep track of tasks / stories that we are required to complete for the upcoming milestones. We've set up WebHooks for discord, so it will be easy to see who has changed what on our jira project, and who is assigned to which task.

Upcoming Tasks:

Logging:

- Add https://logging.apache.org/log4j/2.x/ logging to the project
- Implement 4 different logging levels
- Log files at debug level
- Use console logging at either info or warn level
- Almost all important changes to data (properties in classes or local variables in methods) in the program should be logged in some sort of way, especially for debugging purposes
- Implement functionality to allow developers to limit logging levels, or shut off logging entirely
- Possibly implement a system which clears log files
- Any and all errors / warnings should be logged
- Reading and writing from any files will be where logging needs to be the heaviest
    - Log any data read from a file, and log any data written as well
- Logging should be sufficient enough to make it easy to understand where control is being passed to / which method has control at any time. This will make it easy to understand where errors / bugs originate from
    - Log messages which announce where code is being executed e.g. "Reading from file 'example.sav' in method 'readFile()'", etc.
- We want to make sure each application "session" is followed from start to end.
    - Make sure we are logging when each file is loaded into the program
    - Make sure we are logging when each file is created by the program
    - Make sure we log when the program is about to exit
- Implement logging for tests as well, making it easier to track where tests fail / succeed

Tests and Plan:

- Add https://logging.apache.org/log4j/2.x/ logging to the project
- Fix bugs and warnings generated by SQL files (when they are loaded into a database)

- Create 4 of 5 of these tests: CreateDDLMySQLTest, EdgeConvertCreateDDLTest, EdeConvertFileParserTest, EdgeFieldTest, and EdgeTableTest
- Any calculated and non-hard-coded values in the application should be tested
- Create mock edg and sav files
    - When the user reads a file in, any strings used from the file should be tested against these files
- Before writing to any files, we should be testing for expected output / string creation from any methods that create database tables
- Every time the user changes a table or updates a column / row in a table, we should be testing for expected output
- Any time the user chooses an output file, we should be testing to make sure that it exists
- Any text / data presented to the user should be tested against a series of expected strings, so we are not displaying incorrect / corrupted data

Refactored Code:

- Introduce greater decoupling inside EdgeConvertFileParser class
- Limit calls to System.exit() for some error conditions
- Make sure each class defines appropriate class properties, and that each method defines local variables in-scope, instead of on the class itself
- Make sure each *processing* method which is called upon startup of the application is passing data through to other processing methods appropriately (e.g. by passing data through each method's parameters), and that we're not just setting each piece of data as a property of the class
- Refactor EdgeTable and EdgeField to simplify the code
- Move EdgeConvertFileParser.parseEdgeFile() and EdgeConvertFileParser.parseSaveFile() to their own classes
- Possibly convert EdgeConvertFileParser to an abstract class
- Create abstract classes and interfaces if needed
- Allow EdgeConvertFileParser to read in additional file formats
- Possibly separate the constants defined in EdgeConvertFileParser into their own constants class
- Possibly move ParseSaveFile and parse EdgeFile into their own classes, outside of EdgeConvertFileParser
- Modify the UI code to allow for additional file formats