3XA3 Development Plan

Group 7 - Databased
Eesha Qureshi, qureshe
Faiq Ahmed, ahmedf46
Kevin Kannammalil, kannammk

February 4th 2022

Contents

Table 1: **Revision History**

Date	**Version**	**Notes**
February 4, 2022	1.0	Initial Document
February 4, 2022	1.1	Final Document
April 12, 2022	1.2	Changes added for rev1

1 Introduction

This is the Development Plan for the project Databased

2 Team Meeting Plan

Where	When	Length
Microsoft Teams/ITB 236	Every Tuesday & Thursday at 9:30 am	2 hours
Discord	Every Friday at 5 pm	2 hours

Table 2: Team Meetings

2.1 Rules for Agenda

An agenda will be used for all team meetings that will be held. The rules for this agenda will be:

1. The agenda must be prepared and distributed before a meeting begins
2. There should be a reasonable amount of topics in an agenda
3. There will always be a chair overlooking the meeting
4. The chair must go over the topics of the agenda before proceeding to the first topic
5. The topics of the agenda should all be addressed by the end or postponed to the next meeting

2.2 Meeting Roles

The meeting roles of each member are:

Member	Roles
Faiq Ahmed	Chair, Participant
Eesha Qureshi	Timekeeper, Participant
Kevin Kannammalil	Recorder, Participant

Table 3: Meeting Roles

2.3 Meeting Role Descriptions

2.3.1 Chair

The role of the chair is to

1. create an agenda for the meeting and plans logistics

2. guides the meeting corresponding to the agenda
3. be in charge of the meeting environment and participants
4. assigns responsibilities and tasks for each member

2.3.2 Timekeeper

The role of a timekeeper is to

1. be in charge of time limits for each topic in the agenda

2.3.3 Recorder

The role of a recorder is to

1. coordinate with the chair to create an agenda
2. distribute the agenda before the meeting begins
3. create minutes for the meeting and distribute it to everyone

2.3.4 Participant

The role of a participant is to

1. understand the agenda and objective of the meeting
2. participates in discussion and contributes to topics of the agenda

3 Team Communication Plan

Team communication will take place predominantly during the team meetings. These meetings will take place on Microsoft Teams every Tuesday and Thursday, and will take place on Discord every Friday. The meetings on Discord will take place in a group voice call in the 3XA3_L01GRP07 Discord channel with all members expected to be present and on time. The meetings on Teams will take place in a group call in the 3XA3_L01GRP07 Teams channel with all members expected to be present and on time. Additional communication outside of regular meeting times will take place in the corresponding Discord text channel.

4 Team Member Roles

Member	Roles
Faiq Ahmed	Team Leader, Developer, Code Reviewer, LaTeX Expert, Tester
Eesha Qureshi	Developer, Code Reviewer, Git Expert, Tester
Kevin Kannammalil	Developer, Code Reviewer, Scribe, Documentation Expert

Table 4: Team Member Roles

5 Git Workflow Plan

The Git repository where all the project code will be sourced is named "GoDBMS L01 GRP07". Edits will be reflected in this repository with descriptive commit messages to provide an accurate version history. A branch-feature workflow will be implemented. For each feature, a new branch is created and after validation and verification, it is merged into the main branch, thus ensuring that the main branch is always functional. Labels are represented by git tags, where a tag displays the version.

6 Proof of Concept Demonstration Plan

6.1 Significant Risks

6.1.1 Implementation

As this project contains both an SQL parser and the implementation of the actual database with the addition of adding in concurrency, the scope of this project may be too large to finish by the end of the term. Though the SQL parser would be a relatively large part of the project, it would not be difficult to implement due to its straight forward nature and having various resources include the original project source code to reference. The DBMS on the other hand would need to be modified significantly and would require a broad understanding of the theory of databases to implement efficiently.

6.1.2 Testing

Due to the concurrent system design aspect of this project certain robust testing methods such as statement coverage based testing may be difficult to implement. Similarly exploratory testing would also be difficult due to the large number of scenarios that can be created by combining different SQL queries.

6.1.3 Libraries

Library installation is not a concern as Go has a package manager that can be used to easily install all necessary libraries for this project. The original projects also does not have any Java specific libraries that are not also available on Go.

6.1.4 Portability

Portability is not a concern as Go can be compiled to a binary for all modern operating systems.

6.2 Plan to Overcome Risks

6.2.1 Implementation

The scope of this project would be significantly reduced to only support a few primitive data types such as strings and integers as well a single simple storage method such as a B+

Tree. To further reduce the complexity of this project an SQL parsing library can be used instead of creating our own SQL parser if the scope needs to be further reduced. All team members are familiar with Database theory and how transactions are handled concurrently in a DBMS. For the proof of concept demo, the functionality for a single SQL instruction such as creating a table in the database can be implemented.

6.2.2 Testing

To ensure that testing can be done efficiently and within the allocated time frame of the course, unit testing will be used instead of exploratory testing. Additionally the majority of the unit tests would only loosely follow a path coverage model. Additional testing methods such as exploratory testing may also be used after unit testing is complete if possible within the time frame. For the proof of concept demonstration, some basic unit tests would be implemented for the database and SQL parser for the specific SQL instruction that would be implemented. Additionally due to the difficulty of testing concurrency and locking with unit testing, the tests to ensure concurrency will be done via manual tests.

6.2.3 Libraries

The official go package manager would be used to install and maintain the libraries. The modules file would be pushed onto the repository to keep track of all installed libraries between team members.

6.2.4 Portability

All members will use the latest version of Go and compile and build the project on their own operating systems for testing and the demonstration. Docker will also be used to allow for easy installation of the project in a docker container.

7 Technology

7.1 Programming Language

This project will be implemented using the latest version of the Go programming language. This language was chosen due to high performance and its simplistic approach to creating concurrent systems using message passing. All team members are familiar with written concurrent systems in Go.

7.2 IDE

This project will be using the Visual Studio Code IDE. This IDE is easy to use and set up and has additional support for Go using addons to the IDE. All team members are already fimiliar with this IDE as well.

7.3 Testing Framework

This project will use the Go standard testing library for writing unit tests. These tests can easily be ran over the command line by the Go package manager.

7.4 Document Generation

This project will be documented in LaTeX and the documentation would be pushed to the repository. All source code will also be properly documented with comments and the Godocs library will also be used to automatically generate documentation from the comments.

8 Coding Style

The coding standard that we will be going off of is the Golang style described in `https://go.dev/doc/effective_go` to stay consistent to one style. ~~We will also be utilizing the *camelCase* naming convention for variables and parameters and the *PascalCase* naming convention for methods and classes~~. In accordance with Golang's naming convention, private methods and structs will be using *camelCase* while public methods and structs will be using *PascalCase*. Furthermore, all local variables and parameters will be using *pascalCase* and file names will follow *snake_case*.

9 Project Schedule

The project schedule for our group is located in our git repository which can be found at `https://gitlab.cas.mcmaster.ca/ahmedf46/GoDBMS_L01_GRP07/-/blob/main/ProjectSchedule/GanttChart.pdf`. Any updates to our project timeline and tasks will be reflected in the gantt chart. The gantt chart focuses on the progress of the project tasks and the resources.

10 Project Review

The re-implementation of the SimpleDB project by group 7 was successful in achieving its goals. The purpose of this project was to re-implement the SimpleDB project to make it easier to learn from by reducing complexity in terms of both the source code, and the architecture, as well as improving its usability. This was achieved by using Go's message passing concurrency model and libraries to be easily be able to run transaction concurrently. Since the http library in Go automatically runs all http request concurrently, this allowed us to easily be able to establish concurrency in our project while also increasing its usability through the http server. Additionally, modularization of the project into various different modules and adhering to the MVC architecture also helped with reducing the complexity of this project. To improve the usability of the project, a SQL interpreter and a command line interface were also added which allowed users to directly interact with the system in a controller manner.

Though we were able to successfully implement this project, there are still a lot of additional improvements that can be added. One of the most important improvements to this project that we did not have the time to implement, would be having the ability to commit and abort transactions. In our current project, it is very hard to demonstrate and test the concurrency of the implementation, but if abort and commit were to be added, it would significantly help with allowing the users to learn more about concurrency in a DBMS. Additionally, the current implementation of the project uses a heap file to store the tuples in the database instead of a B+ tree. Since a B+ tree is a much more common data structure that is used in database management system, it would have also helped with increasing the understanding of DBMS for the users if it was implemented.

The team for this project was able to contribute and work together effectively. Communication through discord and in person meetings during tutorials allowed the team to be able to finish the project on time and adhere to our schedule as planned.

In conclusion, the GoDBMS project was a success in meeting its original objective of being easier to learn about database management systems from and having reduced complexity. Though there are several additional features that can be added to improve this implementation even further, these improvements will be easy to add due to the modularization of the code.