ASRC
Metrics Database
Design Document

Prepared by: Parth Darji

Version: 2.0

Date: 10/25/2019

Sponsors: Chris Bartley, ASRC Federal

Table of Contents

1. Introduction

1.1. Purpose

This document will go over in detail about the design and architecture that is currently present in the Metrics Database project.

1.2. Scope

The main goal of the Database Metrics project is to provide an automated system that will allow developers to run Unified Code Counter (UCC), CPPCheck, Gcov and Jenkins on their code and store given metrics into a database.

UCC: UCC is a comprehensive software lines of code counter produced by the USC Center for Systems and Software Engineering. This application uses UCC to get slocs, comments, and maintainability metrics of a program.

CPPCheck: CPPCheck is a static code naylysis tool for C and C++ programs. This application uses CPPCheck to get severity of the program written in C or C++.

Gcov: Gcov is a source code coverage analysis tool. It gives information how often a segment of code is executed.

Jenkins: Jenkins is an automation server that allows non-human part of a software development process. This application uses Jenkins to get built result of the program. Program compiles or it fails to compile.

1.3. System Environment

The system requires a Linux based operating systems, and a server that will store the database. The database is created in MySQL version 8.0.14.

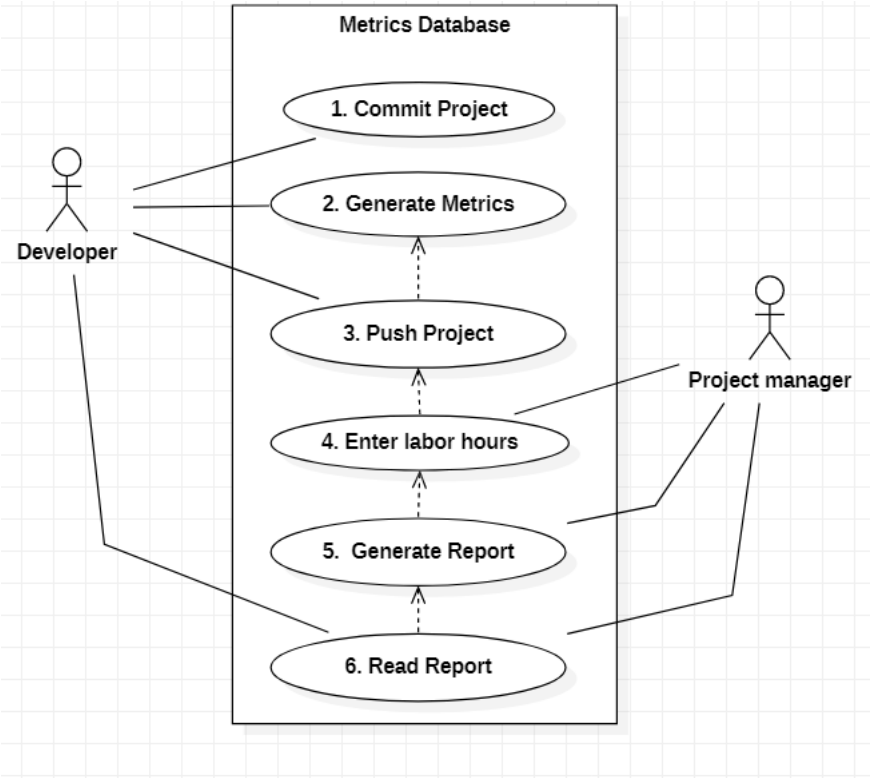
1.4. UI Design

The developer will be able use a Linux command line interface to use this tool.

2. Use Case Diagram

Actors: Developer, Project Manager (Admin)

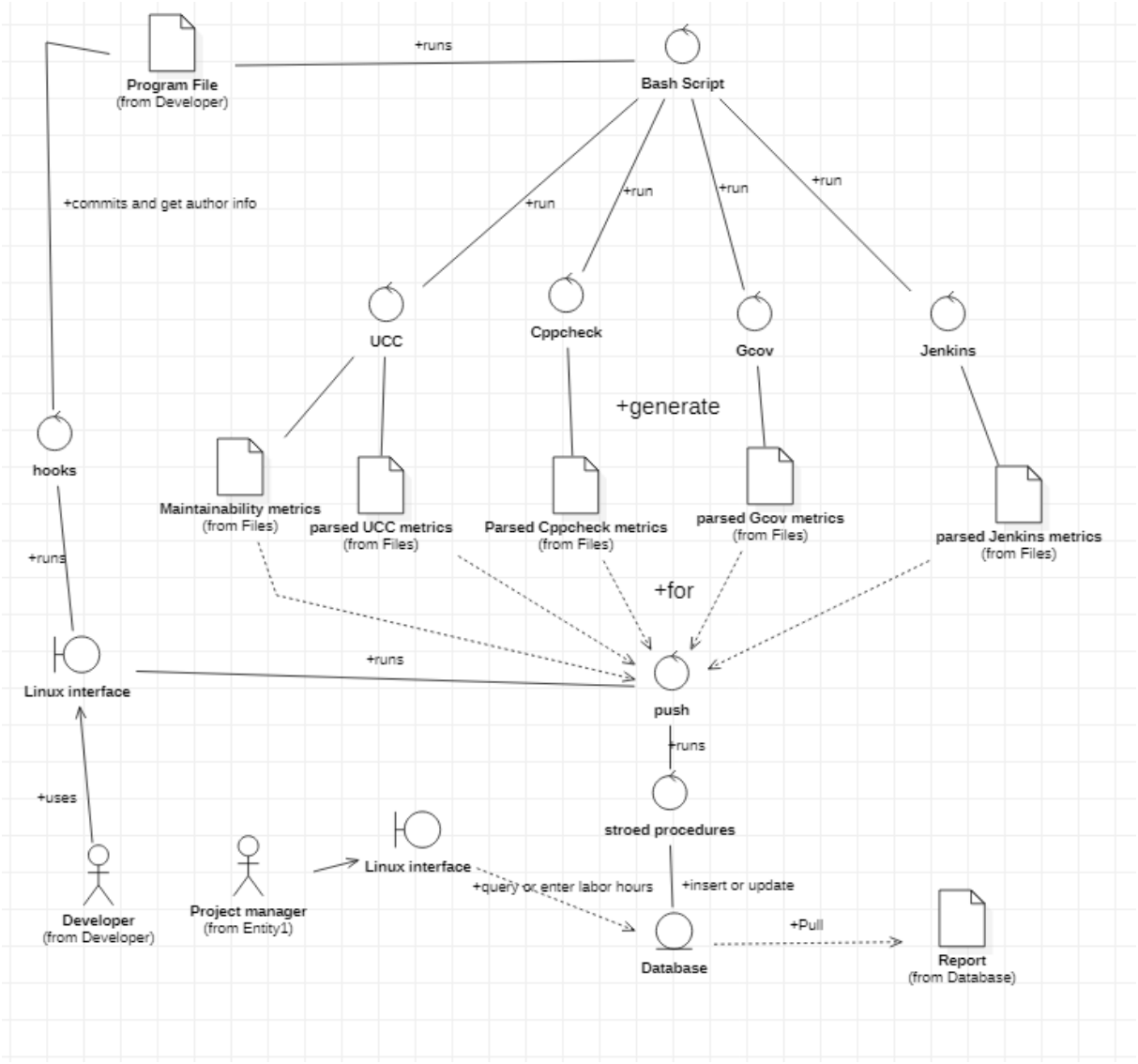
The following use Case Diagram shows all the actions that the Actors above can perform while using the Metrics Database application.



3. Analysis Class Diagrams

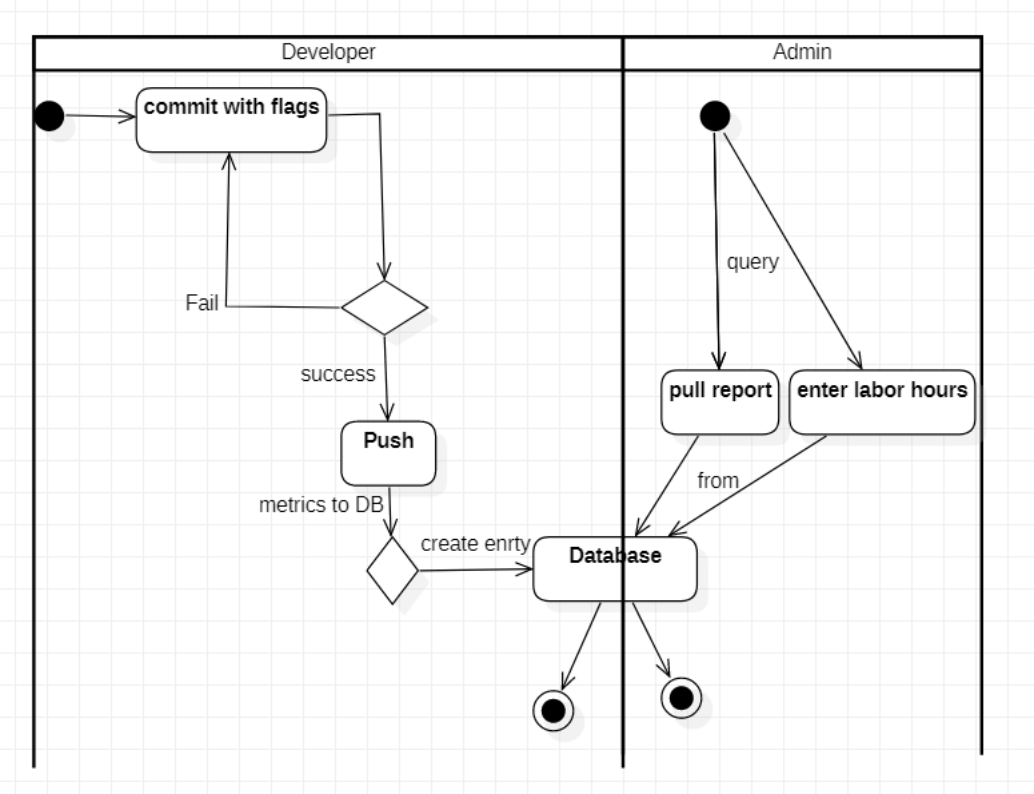
3.1. General User and Admin

The Following Analysis Class Diagram shows all the classes and functionalities a General User and group admin will interact with.



4. Activity Diagram

This Activity Diagram is used to model system behaviors throughout the Metrics Database application. Shows step-by-step activities a user can perform while using this Linux terminal. A developer can commit a program with ‘metrics’ flag to generate metrics. Developer than can push the program to get metrics stored into database. Admin can either add labor hours or pull report from the database.

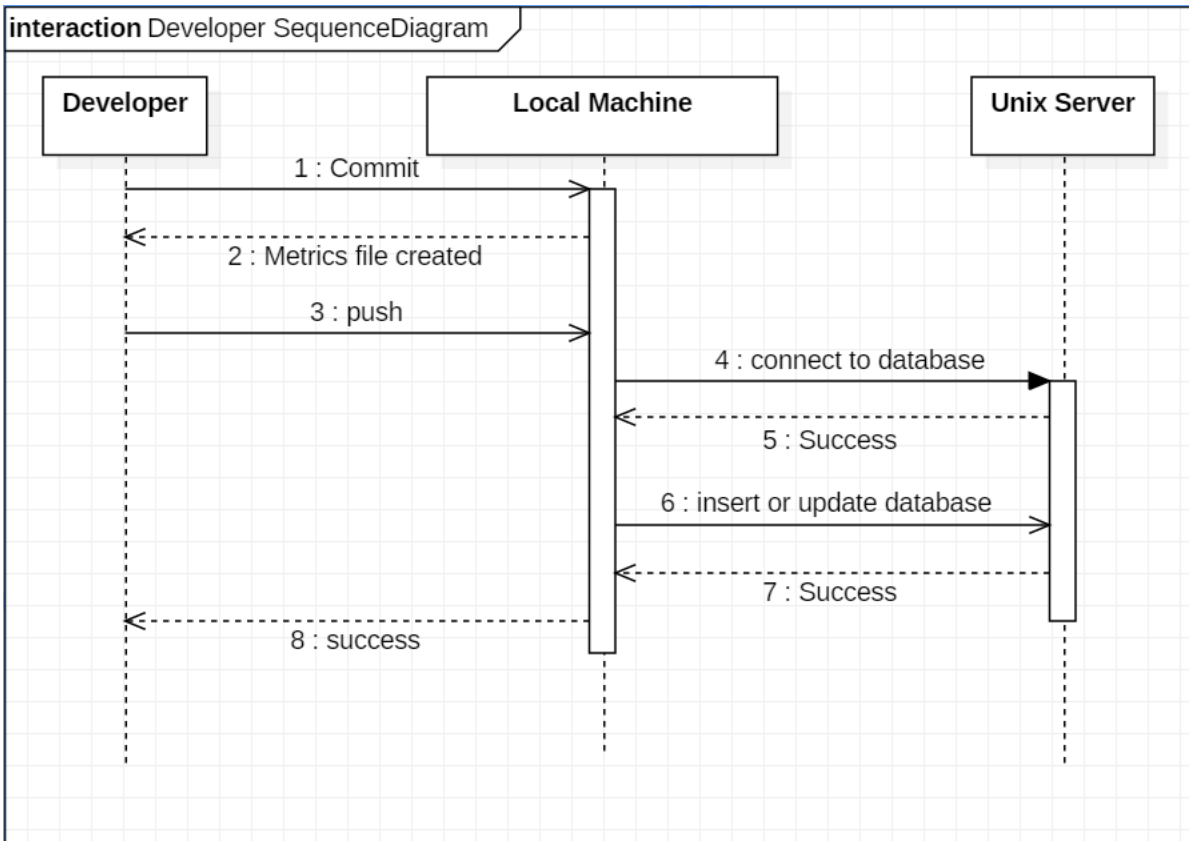


5. Sequence Diagrams

5.1. User actions

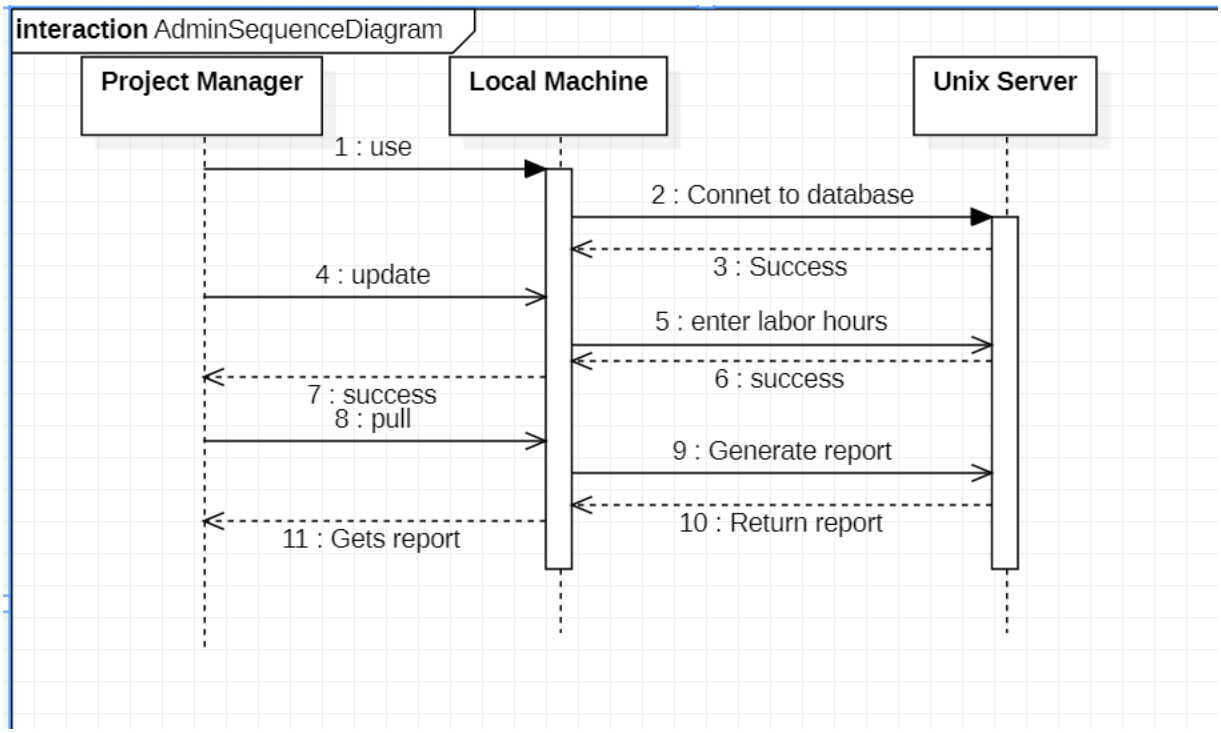
5.1.1. User action

The following Sequence Diagram shows the sequence diagram of all process a developer would use. Developer commits a program. Metrics file are generated given developer commits with ‘metrics’ tag. When developer pushes the program, it will automatically connect to database and store file that are generated from the commit.



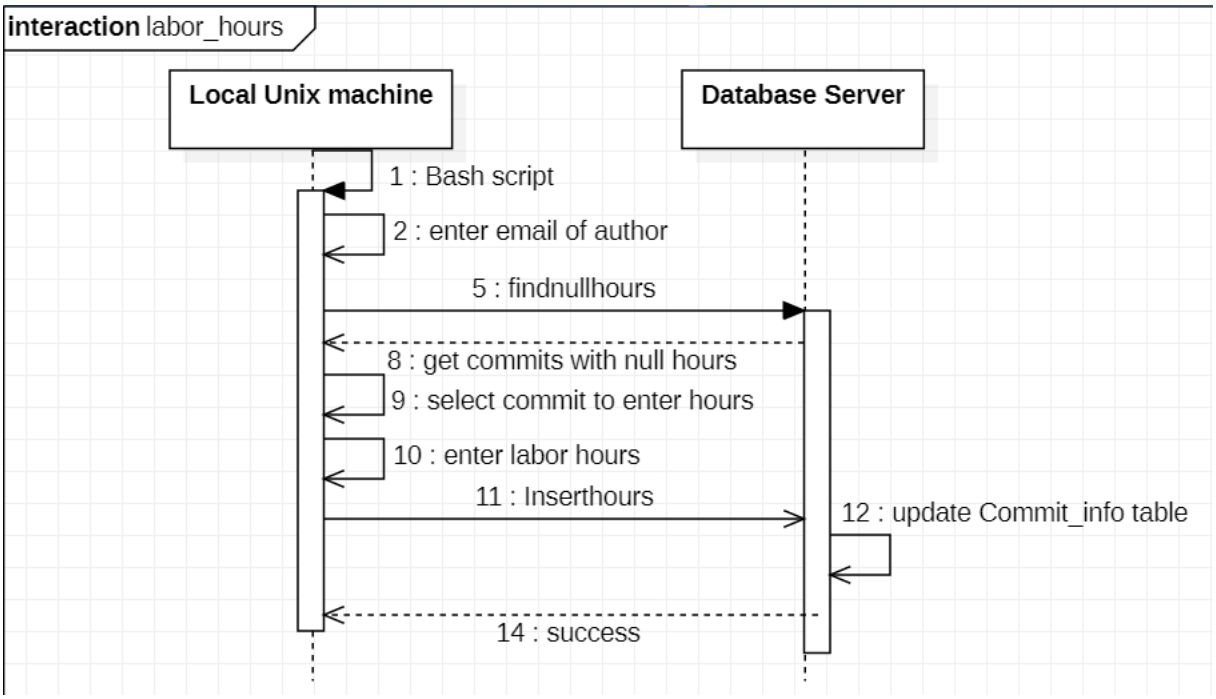
5.1.2. Admin action

The following Sequence Diagram shows the process Admin would use. Admin can enter labor hours or pull report from the database.



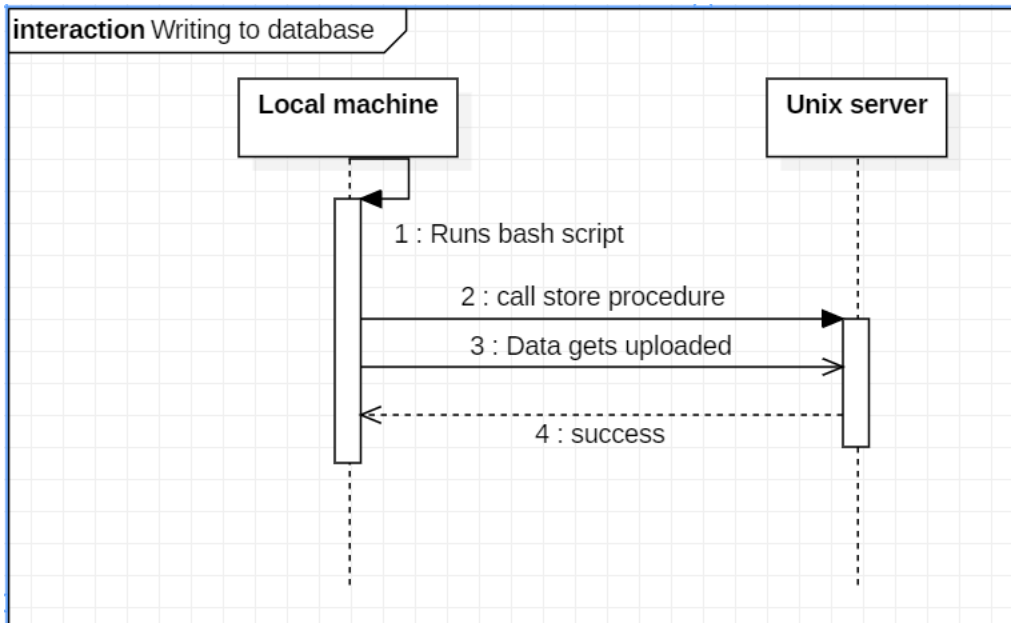
5.1.3. Labor Hour Entry

Admin executes labor_hour bash script. Email of the author is used to identify their commits and a table with all commits by the author with null labor hour is shown. Admin selects the commit_id to add labor hours. Store procedure Inserhours gets called that enters labor hour to related commits.



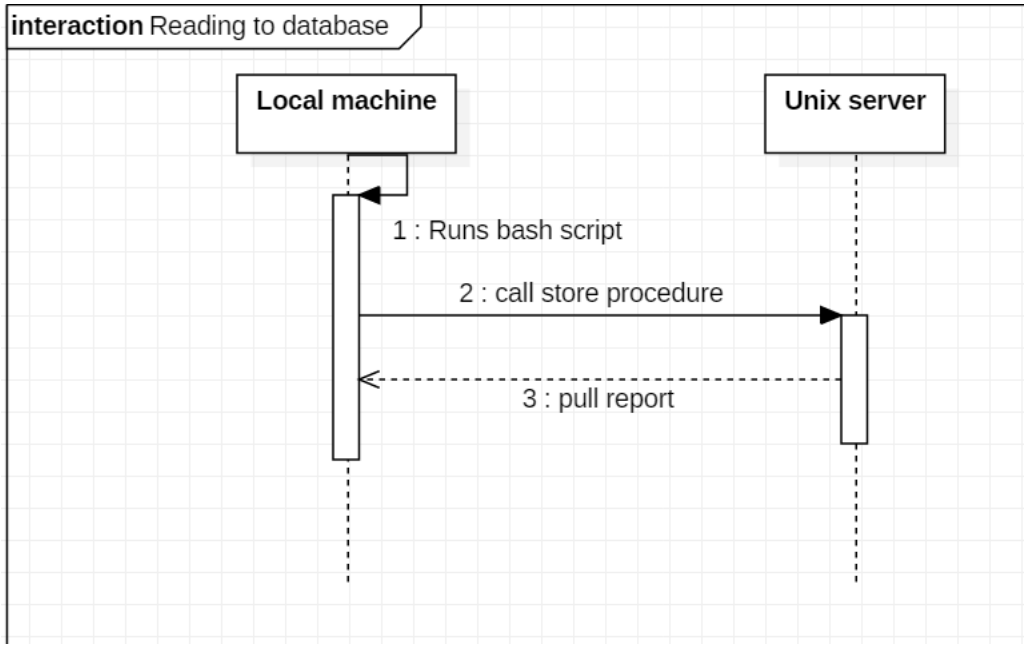
5.2.1. Writing Information to the Database

The following Sequence Diagram shows how information is written to the Database within the Metrics Database application.



5.2.2. Removing Information from the Database

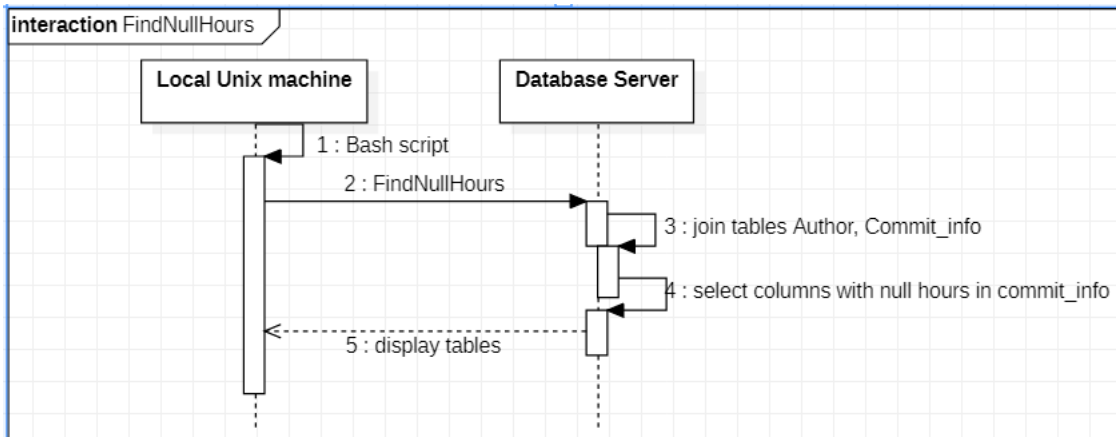
The following Sequence Diagram shows how information is removed from the Database within the Metrics Database application.



5.3. Stored procedures and functions from database

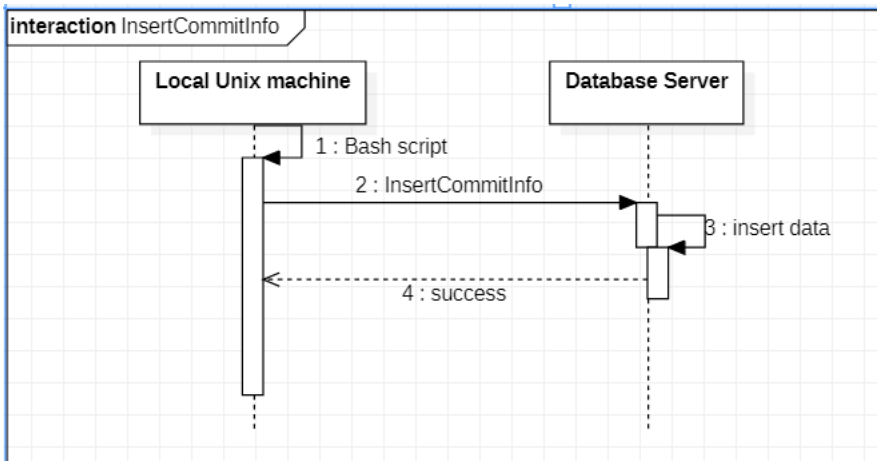
5.3.1. FindNullHours

FindNullHours(IN author_email varchar(250)) is a stored procedure used by an admin account. It allows Admin to see which commits have no labor hour entry.



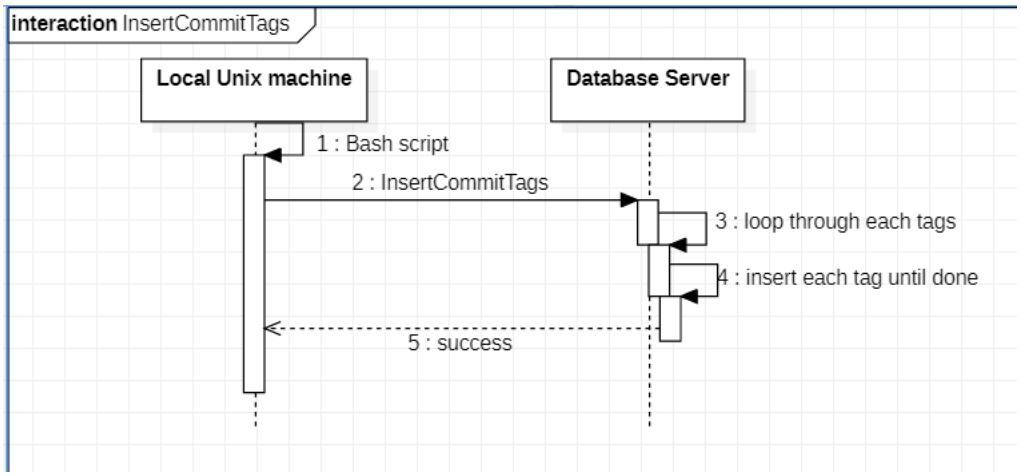
5.3.2 InsertCommitInfo

InsertCommitInfo(IN commit_date datetime, IN ticket_num int, IN author_ID int, IN commit_hash VARCHAR(250)) is a stores procedure used by a developer account. The procedure is called from bash script when a developer is pushing a commit. Database entry is created from parameters passed to the procedure.



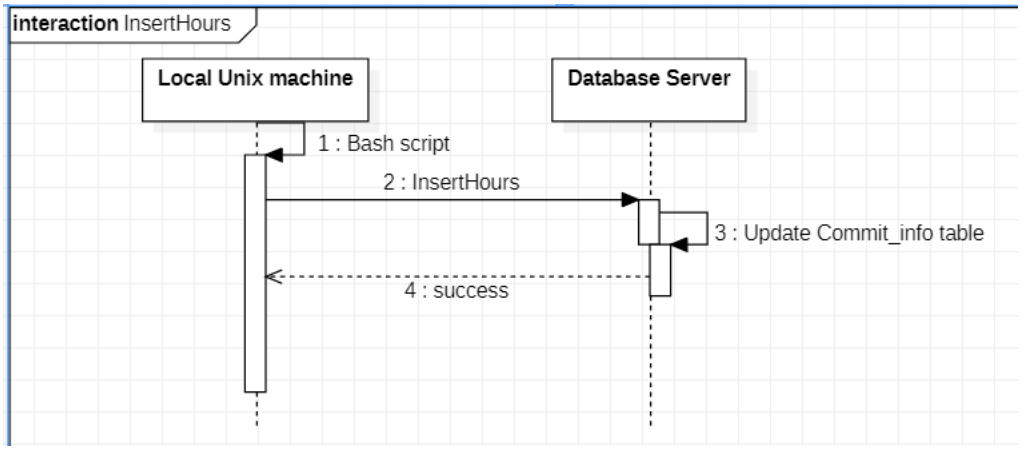
5.3.3 InsertCommitTags

InsertCommitTags(IN commit_hash VARCHAR(250), IN tag_string TEXT, IN repo_url VARCHAR(250), IN num_tag int) is a store procedure used by developer account. It takes repo_url and commit_hash to identify a commit. A tag_string is string of tags separated by commas. Num_tag is used to loop through string to get individual commit tag.



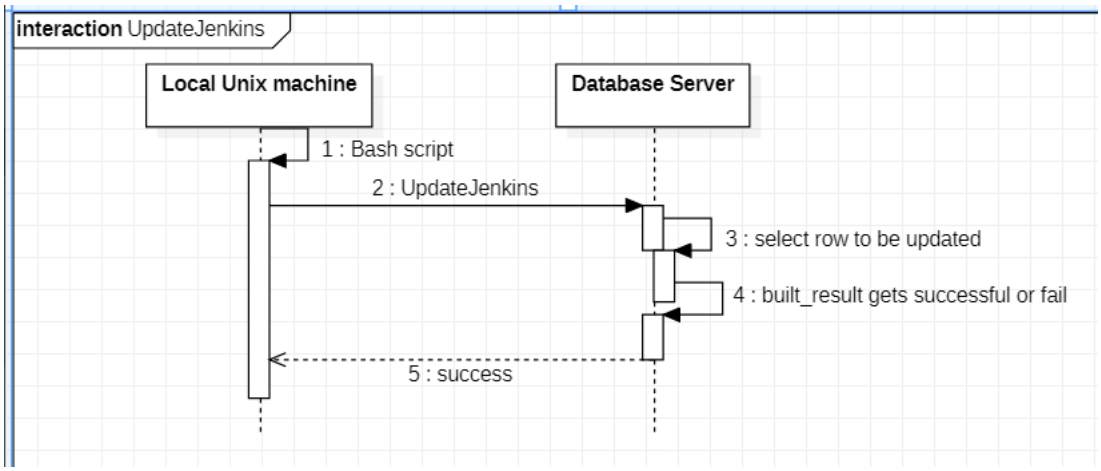
5.3.4 InsertHours

InsertHours(IN commit_ID int, IN hours double) takes two arguments, the hours related to the commit _id. Stored procedure runs update query to insert hours into the commit_info table.



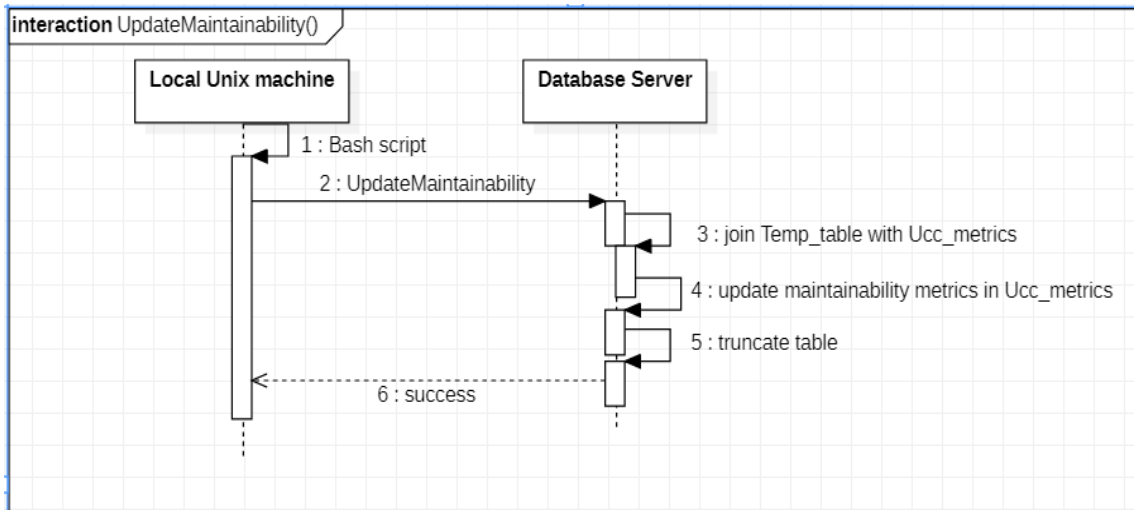
5.3.5 UpdateJenkins

UpdateJenkins(IN commit_hash VARCHAR(250), IN build_result VARCHAR(250)) is stored procedure that uses commit_hash to verify a commit. It updates the row of built_result in Commit_info as success or fail.



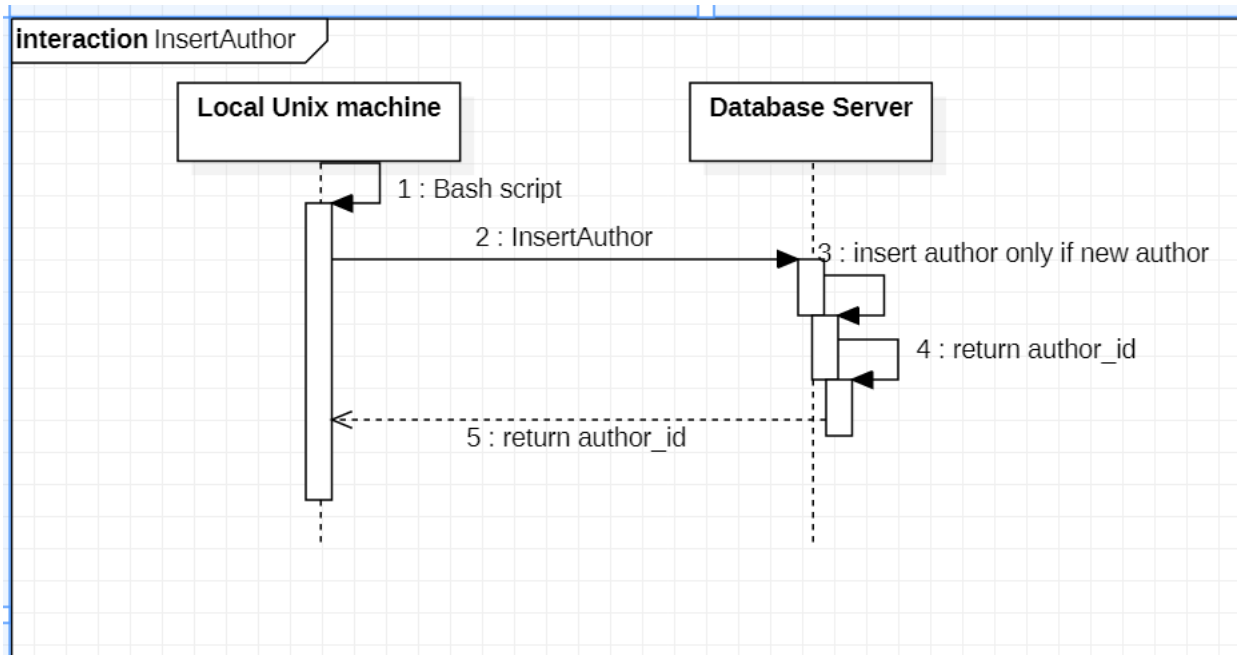
5.3.6 UpdateMaintainability

UpdateMaintainability() is a stored procedure that updates columns metrics_maintainability3 and metrics_maintainability4 in Ucc_metrics table, by joining Temp_maintainability with Ucc_metrics table. At the end the Temp_table is dropped to use again for other commits.



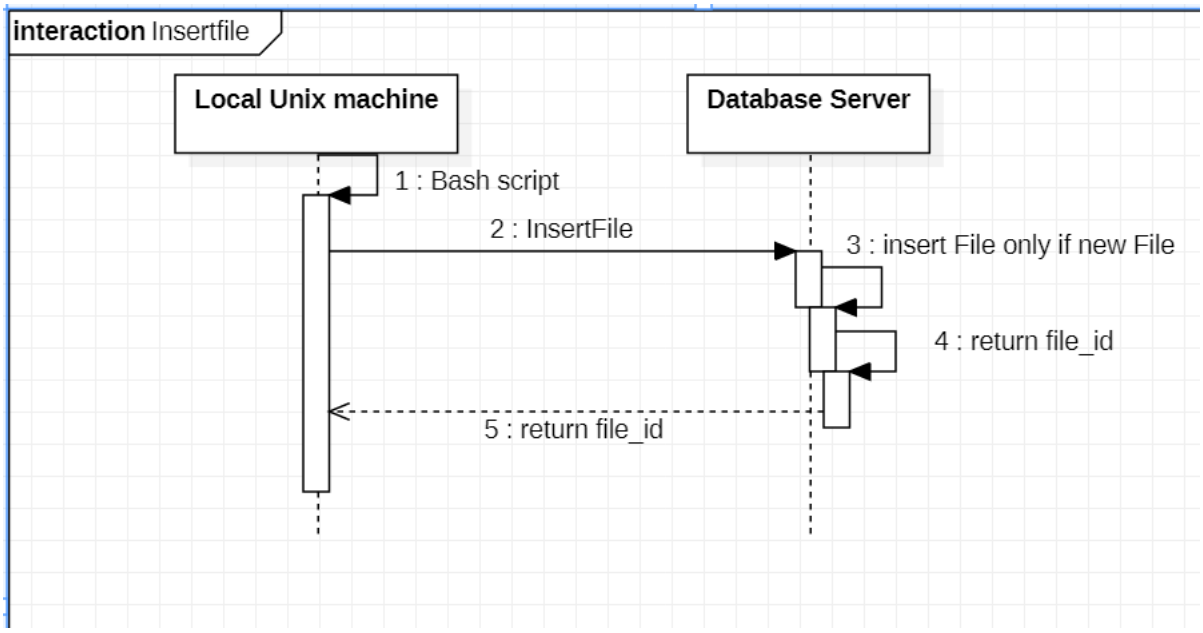
5.3.7 InsertAuthor

InsertAuthor(IN author_email VARCHAR(100), IN author_name VARCHAR(100)), is a function that gets author name and email from hook and calls store procedure to enter that information into to Author table.

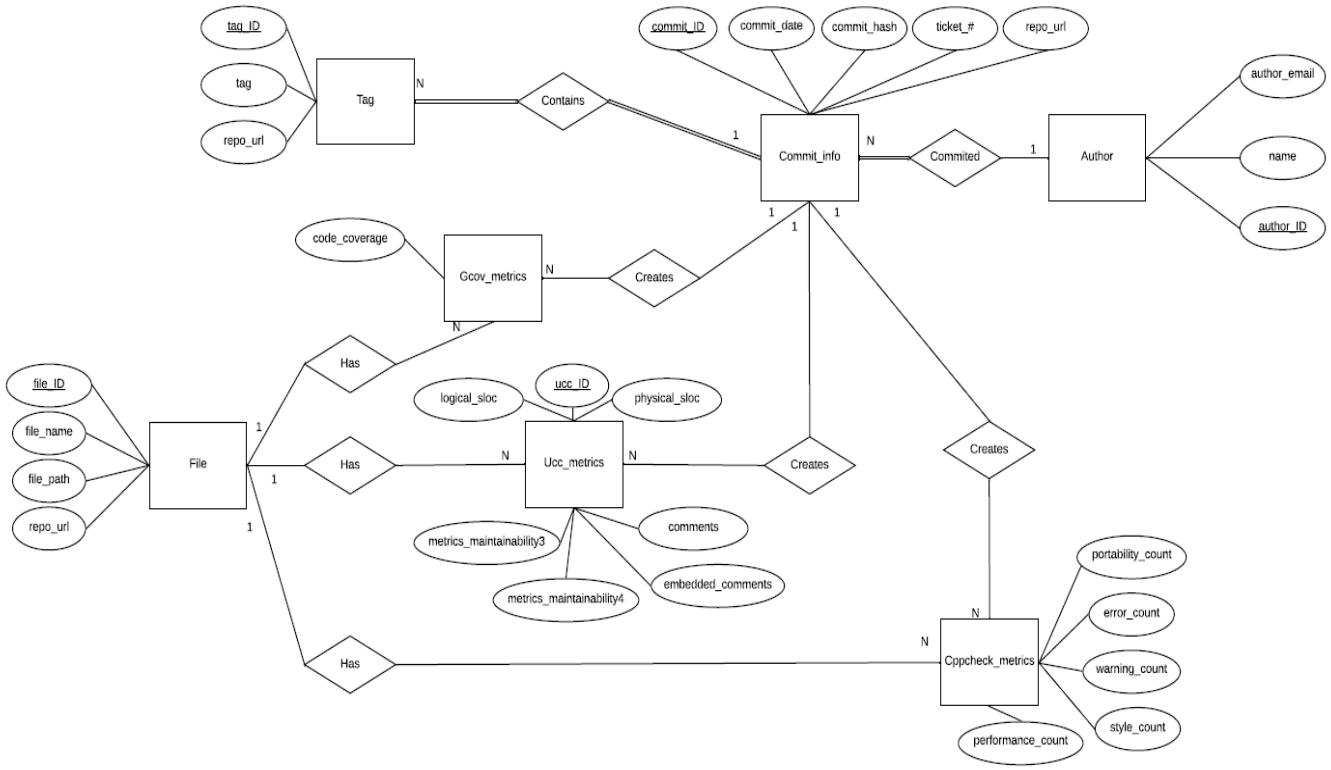


5.3.8 InsertFile

InsertFile(IN file_path VARCHAR(250), IN repo_url VARCHAR(250)) is a function that returns file_id. File are only inserted if they do not exist in the database.



6. EER Diagrams


tag_ID
tag
repo_url
Tag
N
Contains
1
commit_ID
commit_date
commit_hash
ticket_#
repo_url
Commit_info
N
Commited
1
Author
author_email
name
author_ID
1
1
1
code_coverage
Gcov_metrics
N
Creates
N
Has
file_ID
file_name
file_path
repo_url
File
1
1
1
Has
logical_sloc
ucc_ID
physical_sloc
Ucc_metrics
N
N
Creates
Creates
metrics_maintainability3
comments
embedded_comments
metrics_maintainability4
N
N
Has
Cppcheck_metrics
portability_count
error_count
warning_count
style_count
performance_count