Report

CMPUT291 Mini-project 1

Jarrett Yu, Michael huang, nicholas serrano

2018

# Overview

## Introduction

## Setup and Operation Instructions

# Design

## Design Overview

We decided to split the code such that there was a starter class, which started the code; a manager class, which managed the other classes; a login class which dealt with initial connections to the database; a helper class which contained some frequently used functions; and screen classes, one for each functionality required. This allowed for easy concurrent code development as each group member could work on a module without interfering with other parts of the code. A variable that records states is used for the management of screens and will break the main loops of classes given specified states. Prepared statements are used to prevent SQL injection attacks to prevent the need to write our own classes for input sterilization.

## Helper Classes

JDBC\_Connection is the class where we store various shared functions that are involved with connections. The functions here are used for abstracting the creation of a connection as well as functions for sending and receiving messages. The method connect() is made so that if no database is specified in the command line a default database is used from the local directory.

## Driver and Manager Classes

These classes consist of the following: Main, Menu\_Main, and Menu\_Login. The main function contains the main function which accepts 0 or 1 arguments. The argument provided is the path to the database and if none is provided will default to the local directory. Main will handle initializing the connection to the database and the creation of a Scanner object which will be used for all inputs. It then calls and instance of Menu\_Login. A public static variable is initialized here for the program to track its state.

Menu\_Login manages user logins and registrations and operates under to modes, one for each of the previously mentioned functionalities. If a user successfully logs in or successfully registers the menu will call an instance of Menu\_Main which will be the main screen and manage each part of the program.

Menu\_Main manages each sub screen and allows the user to choose which part of the program they want to access.

## Sub Screen Classes

// do stuff for module 1

Menu\_Search handles the second point in the required functionalities described in the assignment specifications which is searching for existing rides and giving the option for a user to send a message requesting to be added to a ride. This class uses a finite state machine to track what should be printed by the program and what inputs from the user it should respond to. The class’s main loop is split into 2 primary functions which is the querying step and the printing step. There is a function dedicated to receiving user inputs, one to searching the database, one to printing a page (5 results), one to process post-query inputs, and one to send a message. If a function operates successfully it changes its state to the next function needed, else it reverts and requests the user tries again. The function links back to only the main menu by breaking out of its primary runtime loop.

# Testing

Stack bug <- ever increasing stack led to multiple main menu quits being required and potential overflow problems

SQL write locking <- stopping eclipse without properly exiting program left connection unclosed. This prevented writing to the database.

# Production Organization

We split the project so that one person had to work on the functionalities of one of the sub screens, as well as build the main menu system. They also worked more on the report since the implementation of the main menu was overall simpler than the sub screens. The other two group members instead worked on two sub screens each. Communication was through in-person speech or though a group-chat. Most of the time work was independent unless a particular member wanted to discuss a system or change part of the design.