CS 5200 Database Management System

**Final Project Report**

**- War Drill Recording and Management System –**

**1. Purpose**

This project is based on the game, Total War: Warhammer II. The windows application will allow users to log in and manage their own troops, arms compositions, and war drill records.

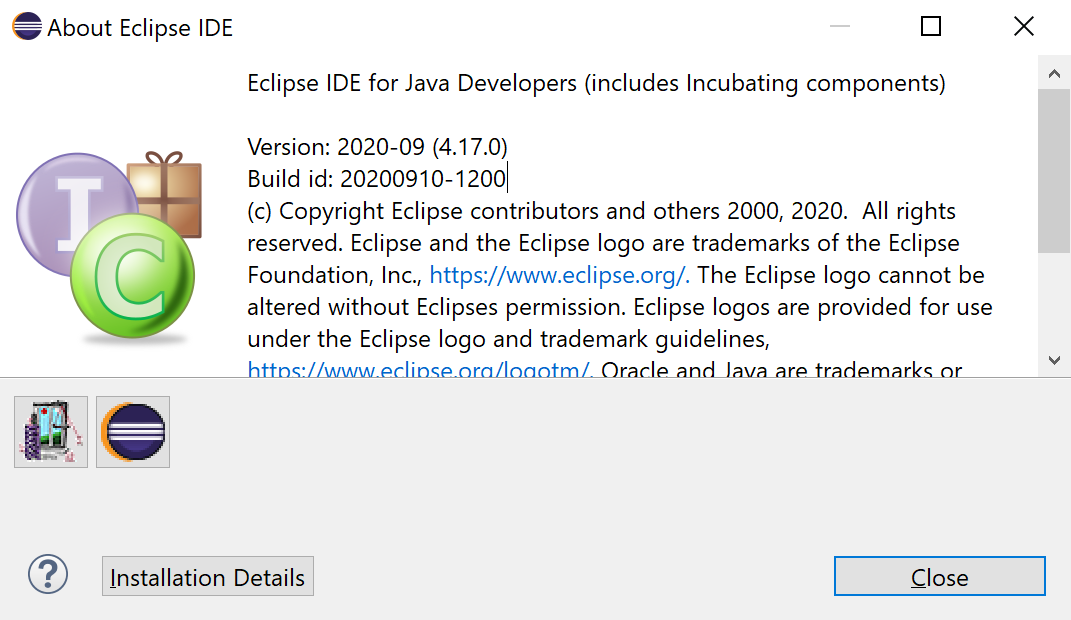
*Why does it interest us?* I was a great fan of Total War: Warhammer II, a war game based on a fantasy world. In this game, a player will play as a commander, exploring the continent, forming up his/her own troops, and selecting proper strategies to beat the enemies. However, since the core game mechanics are too complex, it is fairly unfriendly to beginners. For that reason, this application will be built for those players. They can record and manage the arms compositions they are about to try or have tried and will be able to find out the most suitable ones of their troops.

**2. Technical Supports**

This application used MySQL for database design and Java for programming. There are some preparations need to be done before running this application.

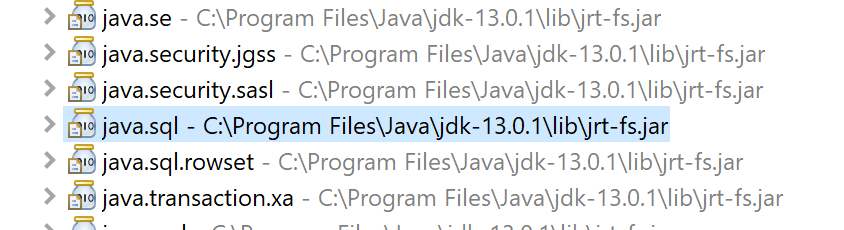
**2.1 Software**

This application was developed by *Eclipse IDE for Java Developers*, and the version is *2020-09 (4.17.0)*.



**2.2 Library**

A standard JRE System Library will serve, and it should contain *java.sql*.

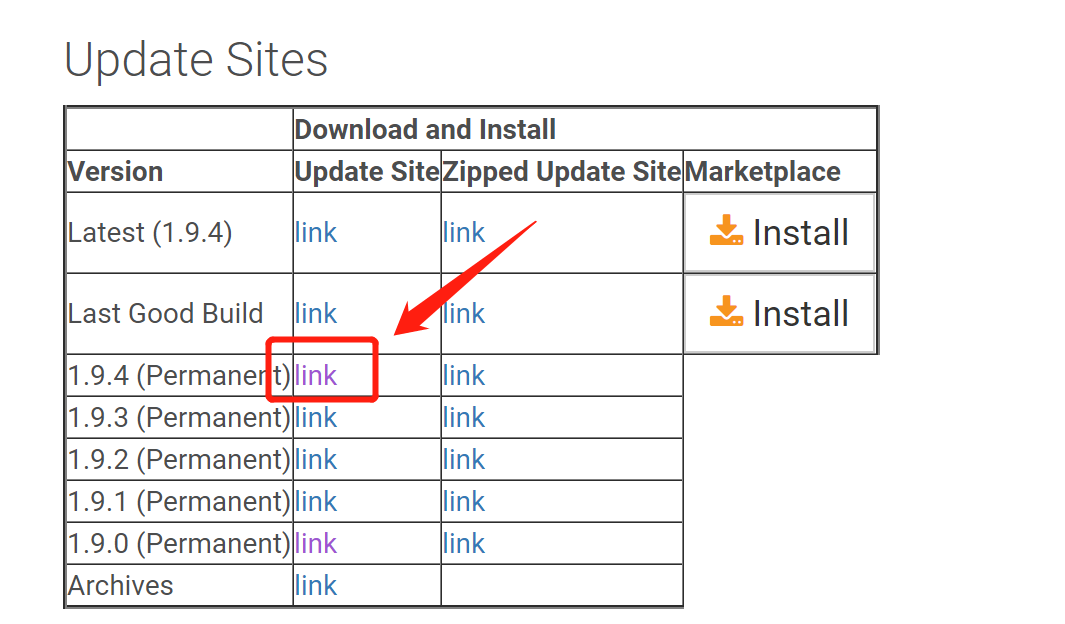


**2.3 Plugins**

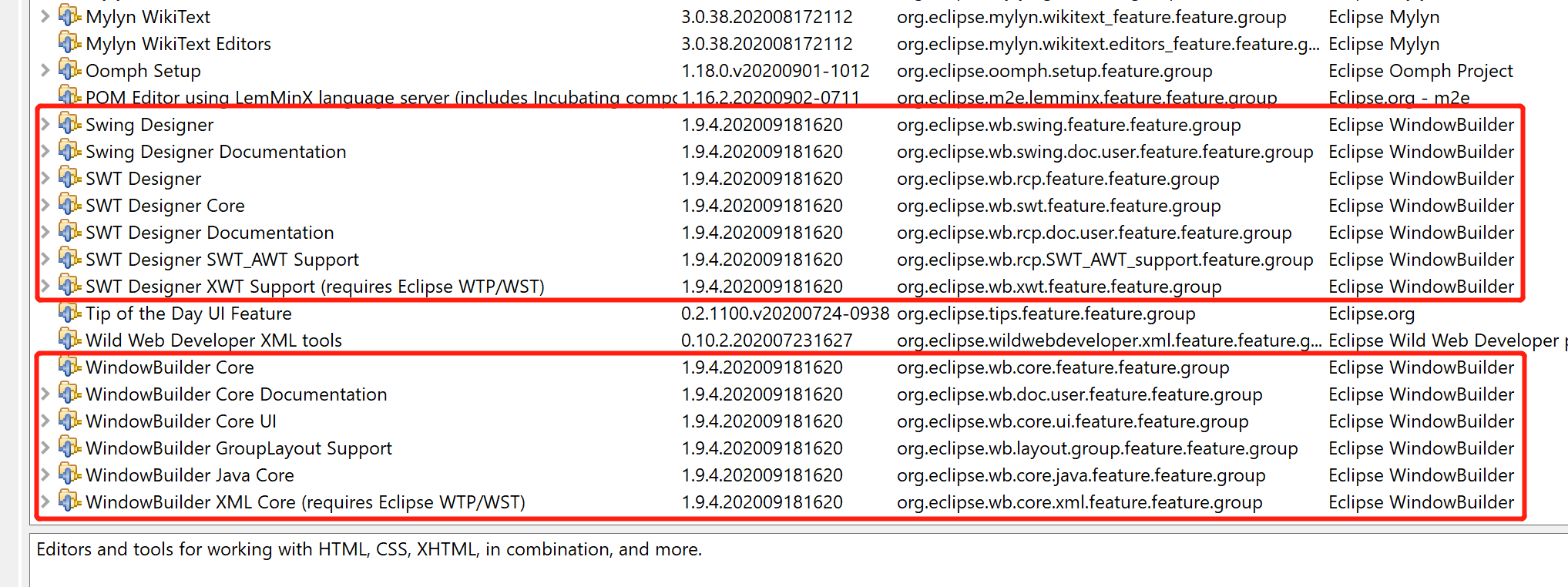
The front-end is based on *Swing* and *WindowBuilder*. Follow the links below:

<https://www.eclipse.org/windowbuilder/download.php>

Open the link and you will find a web page containing the *Update Sites* table below. Copy the link in the red rectangular below and paste it into Eclipse *Install New Software* window (in the navbar, click help and then click install new software). If you need more detailed information or instructions on this installation, click the link in the red rectangular below and you will get what you want.

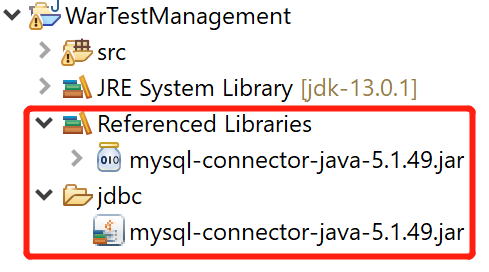


When Eclipse identify that installation link, it prompts you to select plugins. It is suggested to select all of them. When the installation finishes, restart Eclipse and check the installation details. Make sure you have those plugins in the red rectangular below installed.



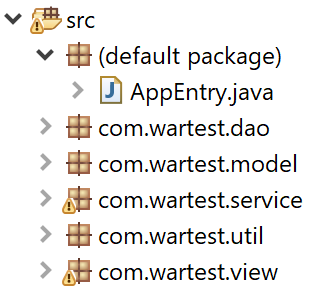
**2.4 JDBC**

Create a folder called jdbc and add mysql-connector-java into it. Remember to add path.

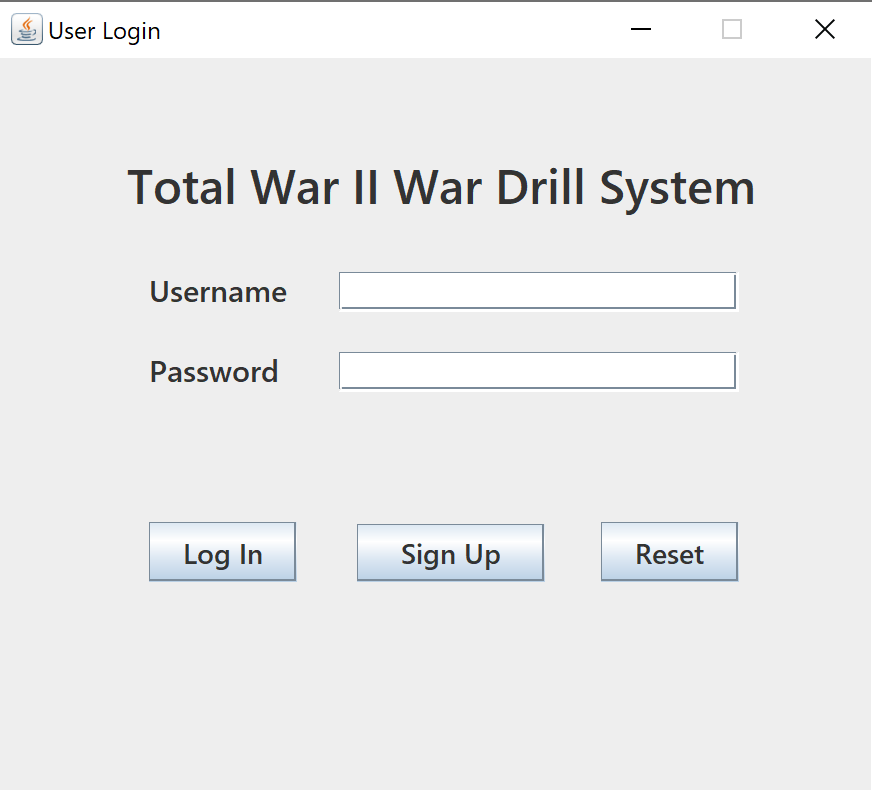


**2.5 Run Application**

Create a Java Project. Copy those packages and .java files into the *src* folder. There is only one .java file in the default package: AppEntry.java, which is the only entrance of the whole project. Run this file and the application starts.



When you can see the Log In form below, it means the application works well so far.



**3. Database Design**

The database UML diagram and the ER diagram generated by MySQL workbench are shown below.

The UML Diagram

The ER Diagram Generated by MySQL Workbench

The database for this application contains 6 main entities.

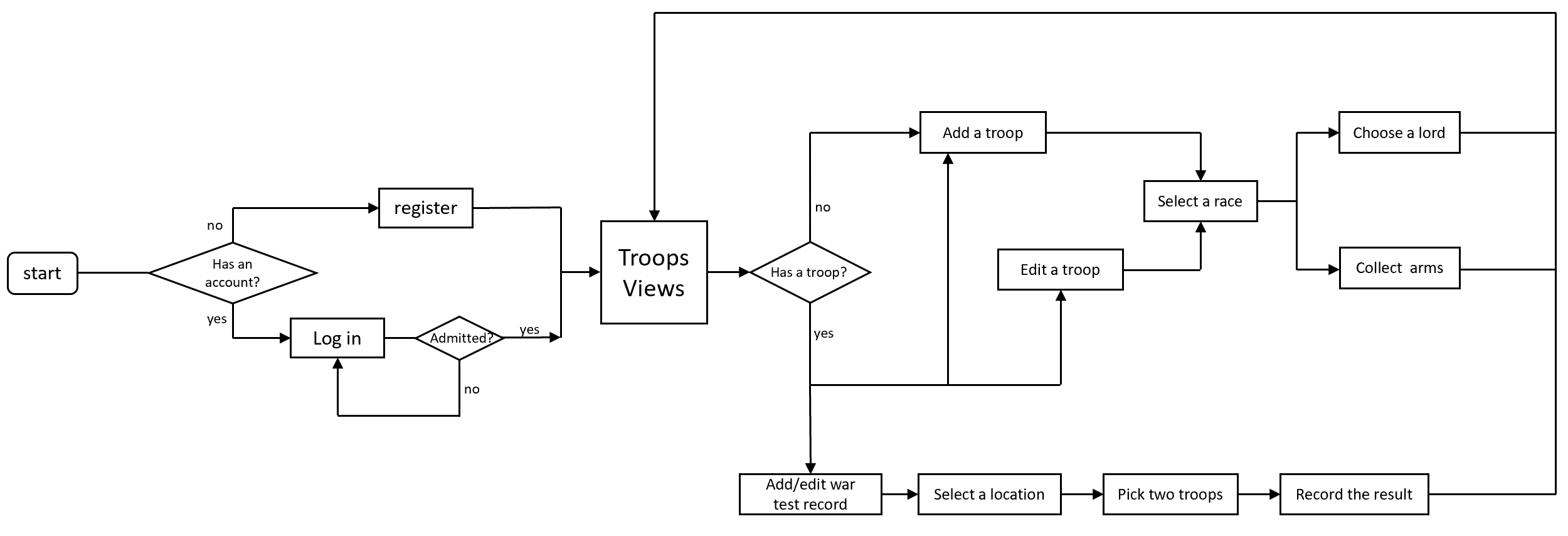
* **Users**. One user will have views only on his/her own troops and war-tests.
* **Troops**. Users can form up their own troops. A troop has a lord and several arms.
* **Lords**. According to the rules of Warhammer II, a troop must have a lord, the commander.
* **Arms**. An arm is a type of a phalanx. They are the main components of a troop.
* **Races**. There are several races in the fantasy world. Every single lord or phalanx belongs to exactly one race.
* **Wartests**. The record of war drills. A tuple of war-test will record which two troops were engaged, where was the combat, and what the result was.

**4. User Flow**

Below are step-by-step operations a user may execute.

* Log In. If a user has no account, then register for an account. This application requires a mandatory log-in.
* Troops Views, the main interface for this application. A user may see the troops that have been formed. One troop row shows simplified lord and arms(phalanx) information.
* Clicking one of the troops will lead to a troop edit interface, where the user can modify this troop by changing the race, the lord, or the arms. If the user has no troop, click the ADD button and also enter the edit interface. This interface contains full information on the lord and arms of the troop.
* When completing the modification/creation, click SUBMIT and back to the troop view. The information will be updated.
* If the user has at least one troop, the War Test interface will be accessible. In this view, the user may see the records of war drills. The edit/create form will also be integrated into this interface. Click BACK and go back to the troop view.

The flowchart is shown below.



The Flowchart

**5. Lessons Learned**

**5.1 Technical Expertise**

The knowledge of using Java to connect with the relational database has been consolidated. After this project, the basic connection works like getting connection, close connection, and catch and handle relevant exceptions can be implemented cleaner, faster, and smoother. The high volume of practice on manipulating data in the database through application code has improved the SQL writing skills.

For database design, this is also a wonderful practice. The abilities to get aware of a real-life issue, transform the abstract description into a database conceptual design, and further transform it into a logical design by using MySQL DDL have all been strengthened.

**5.2 Insights**

1) Practice makes perfect. One of the core requirements of gaining a skill is a large amount of practice, during which one will be able to get used to the specific working flow, accumulate experience with handling errors, strengthen the ability of accessing information, and consolidate the relevant concepts that support the work.

2) Keep your brain sharp. An inactive brain is very likely to seriously slow down the work progress, and even greatly increase the code error rate. Therefore, if you need to take a noon nap, take one; if you need to work out, work out. Relaxing your brain is not a waste of time; letting a bad-status brain ruin your work, however, is.

3) Constantly optimize. A runnable application is never enough. It should be fairly elegant and readable, respond to the user’s manipulations quickly, and has a considerable fault tolerance. There are some operations from the user side that a developer might never have thought about. Therefore, let your friends try your application and be humble to attend to their suggestions or complaints.

4) Do necessary redistribution. For example, when working on the front-end, the logic implements of all kinds of buttons could be quite cumbersome and tedious. If all of the business codes locate in the frontend, it will significantly reduce the readability and increase the difficulty of debugging. In this case, redistributing the business codes into an independent service layer is a very good choice.

**6. Future Work**