FINAL REPORT

The application that we have been working on is an event scheduler with a GUI that the user will interact with it. This Agenda application would be useful to the users to add new events or delete them, so that they will be able to schedule their day easily. Some functionality details found in this app are the calendar to select days, generalized view of the events on a day. In addition, the calendar will also display the months and the years with and without events. To add more events or to view the events in detail the user must select a day in the calendar. In this app the user will also have buttons to add or delete events.

The target industry will include everyone who needs an easy day planner that helps them to create personal reminders throughout the day. This app would be perfect for casual users, businesses and students as well because it has all the agenda properties. Furthermore, the users can find this scheduler app in different app stores for a small and affordable fee comparing to other agenda apps. While for student users this app is for free.

In this project we build four UML Diagrams: 1) Class Diagram 2) State Chart Diagram 3) Sequence Diagram 4) Activity Diagram.

The UML Class Diagram shows the project’s classes, their attributes methods and the relationships among them. In this project the UML Class Diagram has 9 different classes. This diagram also shows the cardinality between classes. For example, there exists only one main class and that shows one LoginController, the event for this project is connected to the event controller which means that many events have one event controller. Also, the Event class is connected to the DayPaneController. The cardinality between these two classes is zero or many to one which means that zero or more events have only one DayPaneController.

MyValidation Class is connected to LoginController class and Database class because it checks if the user’s information is correct or not and if they are then the information goes to the database. The cardinality of these two classes is one to one.

CalendarPaneController Class and DayPaneController Class have a one to one relationship. An obvious thing in the UML Class Diagram is that every class is connected to the database class because each one of them uses a database connection either read or write or both except the CalendarPaneController.

State Chart Diagram

Sequence Diagram

The Activity Diagram describes the dynamic aspects of the system. In this app the activity diagram is very similar to a flowchart with decisions that represent the flow from one activity to another. This diagram starts with the Login Screen where two things can occur. If user’s information is correct than the login is successful which means that the user will see next page which is the calendar pane. If the login is not successful the user will get an error message and he can either try again or exit the program.

From the calendar screen the user can switch it to the day pane where he can either create a new event or view event. If the user wants to add the event than he will be switched to day pane again and the same thing will happen if the user wants to close the event page. In addition, user can exit the program.

Regarding to the agenda implementation there were difficult and easy things to implement. In coding perspective, the most difficult implementation was the database class because it was difficult to make it compatible and useful for each member of the group to use in their part of the code. Also, the method for assigning a number to specific boxes on the calendar was more difficult than initially planned because a lot of defaults libraries don’t store the day of the week they only store the day of the month, the month and the year but not the day of the week. On the other hand, the easiest to implement is the NewEventController class because it only takes user input. Regarding the other classes, each of them had easy and difficult components but we tried to manage them in the best way possible.

If we could go back and start the project all over again we would change the way how we organized everything and be more able and flexible to meet with the group members and to work more as a group than individually because working individually it cost us a little bit more time than expected.

In conclusion, we learned a lot in this project. The most important thing that we learned is that this project was similar to real life projects that maybe one day we will be working for, using UML Diagrams to decide about the classes and also using Git to share the code with the group members. Another thing that we learned in this project is that UML Diagrams are very helpful before writing any code because they help you brainstorm everything. We also learned how to utilize JavaFx and SQLite within our Java application and we got our first taste of version control.