

**Singapore Polytechnic**

**School of EEE**

**ET0706: Object Oriented Programming**

**Project Report**

**Diploma in Computer Engineering**

**Semester 2, 2022/23**

**Prepared For:** Ms Yuek Wee Chan-Tan

**Prepared By:** Damien Law Yong Chung (P2019699)

**Class:** DCPE/FT/3A/21

Content Page

1. Project Overview 1
2. User Manual 1
3. Classes and Methods 19
4. UML Design 24

1. Database Design 25
2. Additional Resources Required 25

Project Overview

This project is a Task Management Application targeted at anyone who wants to improve their time management skills and reduce forgetfulness. It could be particularly useful for people with busy schedules, such as students, professionals, entrepreneurs, and small business owners. The app could also be beneficial for individuals who have trouble keeping track of multiple tasks or projects, or those who want to improve their productivity and efficiency.

The end users of a task management app would typically use it to create and organize tasks, set deadlines, and view their schedule. They can use the app to prioritize their tasks and manage their time more efficiently. They can also use the app to categorize and filter tasks. Users can also set view the task with the highest priority, notified when tasks are approaching the deadline, and are able to get an overview of their upcoming tasks via the bar graph and pie chart in the home page, helping them to stay on top of their responsibilities and avoid forgetfulness.

User Manual

As this is a local application, log in is not needed for this application.

**Creating a Task:**

1. Click on the “Add New Task” Button on the side panel, which will bring you to the page to fill in the details of your task.

Graphical user interface, chart, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. In this page, users are prompted to fill in the details of their task, namely: Task Name, Deadline, Importance Level and Description. To ensure appropriate inputs are being given, users will not be able to edit the DatePicker via typing and can only select one RadioButton in the Importance Level input. After filling in the details, they may click the “Done” Button to save the task. After this, using the inputs for the deadline and importance level, the program will use its algorithm to calculate the priority level of your task.
2. However, if the user left one or more input fields blank, or the app detects that only spacebar is being input, an alert will appear, informing the user of the presence of blank inputs. The task will also not be saved until the blank fields are filled in and the user clicks the “Done” Button again.

Graphical user interface, application

Description automatically generated

1. Additionally, if an invalid date given and the user clicks the “Done” Button, an alert will also show, informing the user that an invalid date is given. An example of invalid dates are the dates before the current date. (The date of when the screenshot below was taken is 12/2/2023) Graphical user interface, application

   Description automatically generated
2. If all inputs return no errors, a notification will appear, informing the user that their task have been saved successfully.

Graphical user interface, application

Description automatically generated

1. After creating a task, users may view all their tasks by clicking the “View All Tasks” Button. Additionally, users may get an overview of their tasks in the Homepage by clicking on the “Home” Button.

**Editing Tasks:**

1. To view your tasks, click on the “View All Tasks” Button on the side panel. The priority level of the tasks will change to “Overdue” if the task is past the deadline.

Graphical user interface, chart, application

Description automatically generated

Graphical user interface, application

Description automatically generated

1. In this page, users are able to edit their tasks by double-clicking the task they want to edit.

Table

Description automatically generated

1. In this example, I will use the task named “homeworks”. After double-clicking, the user will be presented with a page similar to that of the “Add New Task” page with the details of the task loaded into the inputs. Users may edit edit any inputs, however, the same error checking procedure as adding a new task applies, where user are unable to leave any blank inputs or invalid dates.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated Graphical user interface, text

Description automatically generated

1. After the user is satisfied, they may click the “Done” Button to update the task and may view it in the “View All Tasks” Page. If all inputs return no errors, a notification will appear, informing the user that their task have been updated successfully. In this example, I updated the name of “homeworks” to “study”.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

**Marking Tasks as Complete and Deleting Tasks**

1. To mark a task as complete, simply click on the checkbox on the far right of the table. The state of the checkbox will be saved.

Graphical user interface

Description automatically generated

1. To Delete a Task, select the task you want to delete by single-clicking the row of the task and click the “Delete Selected Rows” Button. If deletion of task is successful, a notification will appear, indicating that the selected task as been deleted.

Graphical user interface, table

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

**Search For Tasks**

1. To search for tasks, u can use the search bar on the top right of the “View All Tasks” Page. This will allow the user to search for task name, deadline, importance level. Priority level and description.

Graphical user interface

Description automatically generated

Graphical user interface, table

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, table

Description automatically generated

Table

Description automatically generated

Graphical user interface, table

Description automatically generated

**Export Tasks**

1.To export the task list, the user must first type in the name of their desired file name at the bottom left corner of the “View All Tasks” Page.

Graphical user interface, table

Description automatically generated

1. Next, click the “Export Tasks to .csv File” and the user will receive an alert to confirm that the file name they have entered is correct. Additional information such as the file location are also included in the alert. If cancel is pressed, the tasks will not be exported.

Graphical user interface, text, application

Description automatically generated

1. Users can then access the csv file in the location stated.

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application, table, Excel

Description automatically generated

**Task Summary**

1. If there are tasks present in the database, the homepage will show a summary of all tasks, namely: Distribution of tasks by importance, Number of tasks due on each date, number of overdue tasks, number of tasks and the task with the highest priority.

Graphical user interface, chart, application

Description automatically generated

1. Additionally, if there are tasks that are close to the due date (3 days or less) a reminder notification will pop up when the use opens the app.

Graphical user interface, application

Description automatically generated

Classes and Methods

**HomeController.java**

This class is the controller for the fxml file ,“Home.fxml” which is home screen of the application. It implements the Initializable interface, which means that it is initialized by the JavaFX framework before being used. It also implements the Tasks interface.

The initialize method connects to a MySQL database named "test" and retrieves data from a table named "tasks". It then performs some calculations on the data, such as computing the priority of tasks, counting the number of tasks with different levels of importance, and updating the priority of tasks in the database. The calculated data is then displayed on the UI, including a pie chart showing the distribution of tasks by importance, the date, and the day of the week.. Additionally, it will also trigger an alert when there are tasks approaching the deadline (3 days or less).

The class implements various methods, such as onAddBtnClicked, onHomeBtnClicked, and onTaskBtnClicked, that are called when the respective buttons are clicked they switch to the Add New Task page, Home page and View All Tasks page respectively. These methods use an instance of the SceneController class to switch between different scenes in the application.

**TaskController.java**

This class defines a controller class, TaskController, which serves as a bridge between the UI (defined in an FXML file, “Tasks.fxml”) and the data (stored in a MySQL). The purpose of this class is the allow the user to view all tasks in a tableview , search through the tasks using a filter method and perform sql delete operation on the tasks.

The class implements the Initializable interface, which allows the controller to be notified when the UI is ready to be populated with data. The class also implements the Tasks interface.

The initialize method connects to a MySQL database and retrieves task information stored in the "tasks" table. The retrieved information includes task ID, name, deadline, importance, and description and if the checkbox is checked. The priority of each task is calculated based on its importance and deadline via the PriorityCalculator class, and is stored back in the database if its priority is greater than 100. The task information is then populated into a tableview, “taskList” , with the ability to search for tasks based on name, importance, and description via the textfield named “searhField”. The table displays the task ID, name, deadline, importance, description, priority, and a checkbox indicating the task's completion status. The task information is stored in an observableArrayList, which is filtered and displayed in the table.

The class implements various methods, such as onAddBtnClicked, onHomeBtnClicked, and onTaskBtnClicked, that are called when the respective buttons are clicked they switch to the Add New Task page, Home page and View All Tasks page respectively. These methods use an instance of the SceneController class to switch between different scenes in the application.

The onDoubleClicked method is used to detect double click on the tableview. If double clicked is detected, another controller, EditTaskCtrl is opened. It creates an instance of SceneController and when a double-click is detected on a selected item in the taskList, it calls the switchToEditTasks method from the SceneController and passes the event and the selected task to it.

onExportClicked is a method that exports tasks to a CSV file. This method is triggered by a button click of “exportBtn”. When this method is called, it creates an instance of the "ExportCSV" class, which is used to export the tasks to a CSV file. Before exporting, the method checks if the user has entered a name for the CSV file. If no name is entered, an error message is displayed. If a name is entered, the user is prompted to confirm if they want to export the tasks as the specified name. If the user confirms the export, the "Export" method of the "ExportCSV" class is called, passing the name of the file and the tasks to be exported as arguments.

The deleteRows method is called by the button click of the delRows and starts by retrieving the selected task from the task list. Then it establishes a connection to a MySQL database, using the JDBC driver. Next, it prepares a SQL DELETE statement and sets the selected task's ID as the parameter. The statement is executed to delete the task from the database. Finally, the code removes the deleted task from the observableTasks list, closes the database connection, refreshes the page, and displays an information alert to the user, indicating that the task was successfully deleted.

**NewTaskCtrl.java**

This class is the controller of the fxml file, “New.fml” and main purpose of this class is to allow the user to create a new task with a specified name, deadline, importance level, description and checks if the inputs are valid. The task is then stored in a MySQL database. The class also extends a class called "TaskForm", inheriting the abstract method onDoneClicked() and implements an interface called "Tasks".

The class implements various methods, such as onAddBtnClicked, onHomeBtnClicked, and onTaskBtnClicked, that are called when the respective buttons are clicked they switch to the Add New Task page, Home page and View All Tasks page respectively. These methods use an instance of the SceneController class to switch between different scenes in the application.

The code defines a function called "onDoneClicked" in JavaFX that adds a new task to a MySQL database. The function first establishes a connection to the database and then creates a PreparedStatement with an INSERT SQL statement. The task's name, deadline, importance level, description, and checkbox status are obtained from user input. Before inserting the task into the database, the code checks if any of the fields are blank or if the deadline is in the past. If either of these conditions are true, an error message is displayed to the user. If all fields are filled and the deadline is in the future, the priority is calculated via the PriorityCalculator class and the task's data is converted into the appropriate data types and inserted into the database. If the task is successfully added, a notification is displayed to the user.

**EditTaskCtrl.java**

This class is the controller of the fxml file, “EditTasks.fml” and main purpose of this class is to allow the user to edit and update an existing task with a specified name, deadline, importance level, description and checks if the inputs are valid. The task is then updated in MySQL database. The class also extends a class called "TaskForm", inheriting the abstract method onDoneClicked() and implements an interface called "Tasks".

The class implements various methods, such as onAddBtnClicked, onHomeBtnClicked, and onTaskBtnClicked, that are called when the respective buttons are clicked they switch to the Add New Task page, Home page and View All Tasks page respectively. These methods use an instance of the SceneController class to switch between different scenes in the application.

The setTask method is used to set the values of various UI elements based on the Task object passed as an argument. The method retrieves the ID, name, deadline, importance and description of the task and sets these values to corresponding UI elements such as id, taskName, datePicker, radio buttons for importance, and taskDesc respectively. The deadline is first formatted to a LocalDate object before it is set to the datePicker element.

The onDoneClicked method updates the selected task in a MySQL database. The method connects to the database, updates the data for a task in the tasks table, and executes the update statement. If any of the input fields (task name, deadline, importance level, or description) are blank, an error message is displayed. If the deadline date is before the current date, an error message is displayed. If the inputs do not return any errors, the priority is calculated via the PriorityCalculator class and the task details are updated in the database, and a notification is displayed to the user. The method then navigates to the "View All Tasks" page.

**ExportCSV.java**

This Java class, named ExportCSV, serves the purpose of exporting data from an ObservableList of Task objects to a CSV file. The data of each Task object is written to a row in the CSV file, with columns for the task name, deadline, importance, priority, and description. The file is saved to a specific location on the file system specified in the FileWriter constructor (Downloads Folder). If an exception occurs during the export process, it is caught and printed to the console.

**Priority Calculator.java**

The purpose of PriorityCalculator is to calculate the priority level of a task based on its deadline and the level of importance assigned to it.

The class has a method priorityLvl that takes in two parameters, selectedOption and deadline. selectedOption represents the level of importance assigned to a task, which could be one of three options: "Not Important", "Important", or "Very Important". The deadline parameter represents the due date of the task.

The method first calculates the number of days between the current date and the due date using the ChronoUnit.DAYS.between() method. Based on the number of days, the priority level of the task is determined by adding a value to the importance level of the task. If the task is overdue (daysUntilDue < 0), the priority level is increased by 100. If the task is due in the next 3 days, the priority level is increased by 5. If the task is due in the next 7 days, the priority level is increased by 2. If the task is due in the next 14 days, the priority level is increased by 1. If the task is not due in the next 14 days, the priority level is not increased.

Finally, the method returns the calculated priority level of the task.

**SceneController.java**

The class SceneController handles switching between scenes in the application. The class has methods switchToAdd, switchToHome, switchToTasks, and switchToEditTasks, which switch the current scene to different FXML pages.

Each method first retrieves the current stage by casting the source of the ActionEvent or MouseEvent passed to it, and then uses an FXMLLoader to load an FXML file and create a new Scene with the loaded root node. Finally, it sets this new scene as the scene for the stage and shows the stage.

The switchToEditTasks method is slightly different in that it also sets the task to be edited by calling the setTask method on the controller of the EditTasks.fxml scene.

**Tasks.java**

The purpose of this class is to represent a task, which has several attributes such as a name, deadline, importance, description, and priority. The class creates a checkbox for each task in the tableview, which is used to mark the task as complete. When the state of the checkbox changes (i.e., it is selected or unselected), the change is persisted to a database by executing a SQL update statement.

**TaskForm.java**

This is class is an abstract class which contains an abstract method named “onDoneClicked” which takes a single argument of type "ActionEvent." This method is declared as "abstract," meaning that it has no implementation in the abstract class.

Any class that extends "TaskForm" must provide its own implementation of the "onDoneClicked" method. This allows different concrete classes to define their own behavior in response to a "Done" click event. For example, in NewTaskCtrl and EditTaskCtrl, both have the method onDoneClicked but have different implementations of it. In NewTaskCtrl, it is used to add a new task to the database. However, in EditTaskCtrl, it is being used to update the selected task.

**Task.java**

This is an interface defines a set of methods (without implementation). The interface defines three methods:

1. onAddBtnClicked(ActionEvent event)
2. onHomeBtnClicked(ActionEvent event)
3. onTaskBtnClicked(ActionEvent event)

Any class that implements this interface must provide an implementation for all of these methods. The methods could be used to handle events triggered by clicking buttons, such as the "Add New Task" button, "Home" button, and "View All Task" button.

**Main.java**

This code is the Main class of the JavaFX application. The class extends the Application class and overrides the start method. The purpose of this method is to load and display the Home.fxml file as the initial scene in a JavaFX stage.

The start method uses the FXMLLoader class to load the Home.fxml file, which is an XML file that defines the layout of a JavaFX scene. The loaded scene is then set as the scene of the primaryStage, and the stage is displayed on the screen by calling the show method.

The main method is the entry point of the application, and it launches the JavaFX application by calling the launch method and passing it the args argument.

This code sets up the foundation for the user interface of a JavaFX application.

**UML Diagram**

(fxml inputs are not included)

**Diagram

Description automatically generated**

**Database Design**

**Table

Description automatically generated**

\*note: tinyint type is Boolean

**Additional Libraries Needed**

Mysql-connector-j-8.032.jar

Download link: <https://dev.mysql.com/downloads/connector/j/>