TO DO LIST

OFER GREENBERG

QUAN NGUYEN

QUANG VU

ALLEN DAVID EL

CONTENTS

INTRODUCTION

- For this project, our team was tasked with designing a to-do-list program with the following functionalities:
  - Add/Remove/Rename to-do lists
  - Add/Remove/Rename tasks to the lists
  - Sort the lists based on due date/ creation date and alphabetically
  - Mark tasks as done/undone and important
  - Provide a visual interface
  - Include an additional feature of our own
  - Provide proper documentation of the process, testing code and final product

- As a team we wanted to deliver a finished product that is esthetically appealing and robust.

PROCESS

For the development of this product, our team worked through three platforms

- IDE – per member's preference
- Github – for sharing code
- Trello – for managing and documenting progress
- Draw.io - for visualization
- Discord – as the main line of communication

SPRINTS

The development cycle included 3 sprints and a last day crunch

- Sprint I – in the meeting we decided on program architecture and split the work for building the foundation of the program, the classes and core method headers without implementations.

- Sprint II – in the meeting we decided on methods inputs and outputs and split the work of providing implementations to all the methods except the view class

- Sprint III – in the meeting we discussed adjustment to the methods inputs/outputs, decided on a wow factor and the GUI implementation. In the sprint we finished and tested the implementation of the entire program.

SPRINT 1

- Formed a line of communication through emails and then discord.
- Agreed on a simple MVC model and created a preliminary class diagram (see in supported documents)
- Assigned classes or a combination of classes with overall comparable size, for each team member to create method headers based on the UML design.

- Lesson Learned:
  - Decide on inputs and outputs early on
  - Discuss basic style and formatting before sprint

SPRINT II

- In this scrum we went method by method and discussed the inputs and outputs.
- We randomly assigned an equal number of methods to each team member, for them to create an initial implementation
- We also discussed the proper way to document our progress

- Lesson Learned:
  - When assignment methods, we should in the future write peoples names as a comment, right on the file so we don't accidently work on the same methods or neglect a method.
  - Read the specs as a team again to make sure we didn't miss requirements

SPRINT III

- In this sprint we focused on the view model and test cases.
- We decided to work on the backend, model and control, and the frontend, view, separately so team members can work freely on a class without creating conflicts
- We alternated roles to make sure everyone touches every aspect of the program, from GUI through model and test cases.
- The goal was to get the GUI working, then in the last day crunch, sort out possible bugs.

- Lesson Learned:
  - When working on an interface, getting the aesthetic can be difficult when the class is cluttered, so planning using a diagram what object to use would've been useful.
  - preferably finish the design and then move to implememntation

CODE

VIEW PACKAGE

- The view package contains the GUI
- The code is organized, top to bottom, in a way that allowed us to collaborate most efficiently.
- The colors reflect code complexity

GUI Elements declared as fields and grouped by topic

Application start
-> calls set up methods
-> lambda handlers to each button calling methods

Setup methods
-> put elements into grids
-> call styling methods to configure looks and esthetics

Behaviore Implementation
-> Method for each program behavior
-> organized method by topic (list related, task related, sorting, updating, style)

CODE

MODEL PACKAGE

- The model package holds the ToDoModel class as well as the ToDoList and ToDoTask classes.
- All contain similar getters and setters with the model holding the data as ArrayList<ToDoList> which is serialized
- The ToDoList holds its data as ArrayList<ToDoTask>

CODE
CONTROL PACKAGE

- The control package contains the ToDoControl class which communicates between the view and the model
- The control also provide sorting support
- When a list is sorted a certain way, the control first update the model, then the view receives current lists through the regular display methods. This reduces number of ways the view is interacting with model through the control.

CODE

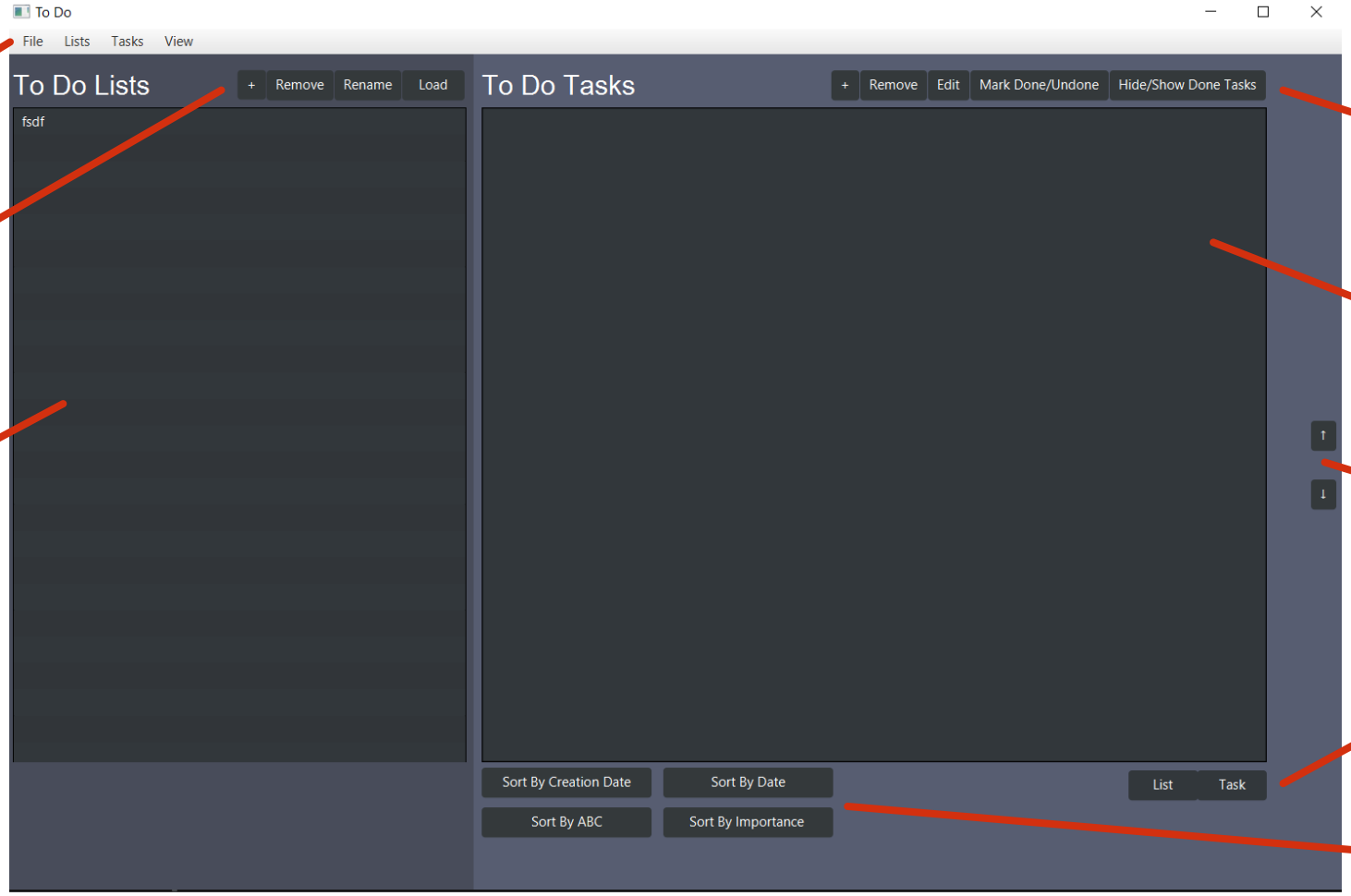
TEST PACKAGE

- The test cases were developed and updated to support all the features in the model and control classes
- We strived for 100% converge and got over 90% for both the model the control the ToDoTask and ToDoList
- The test cases helped us correct many issues with the program while we were working on it rather than at the end.

FINAL PRODUCT

- In the following slides you will see all the features in the GUI
- Using illustrations, we will try to explain how things are set up behind the scene (to be accurate, on the scene… JavaFX humor)

FINAL PRODUCT



menu bar

lists controls panel

ListView object holding all the ToDoList objects

tasks controls panel

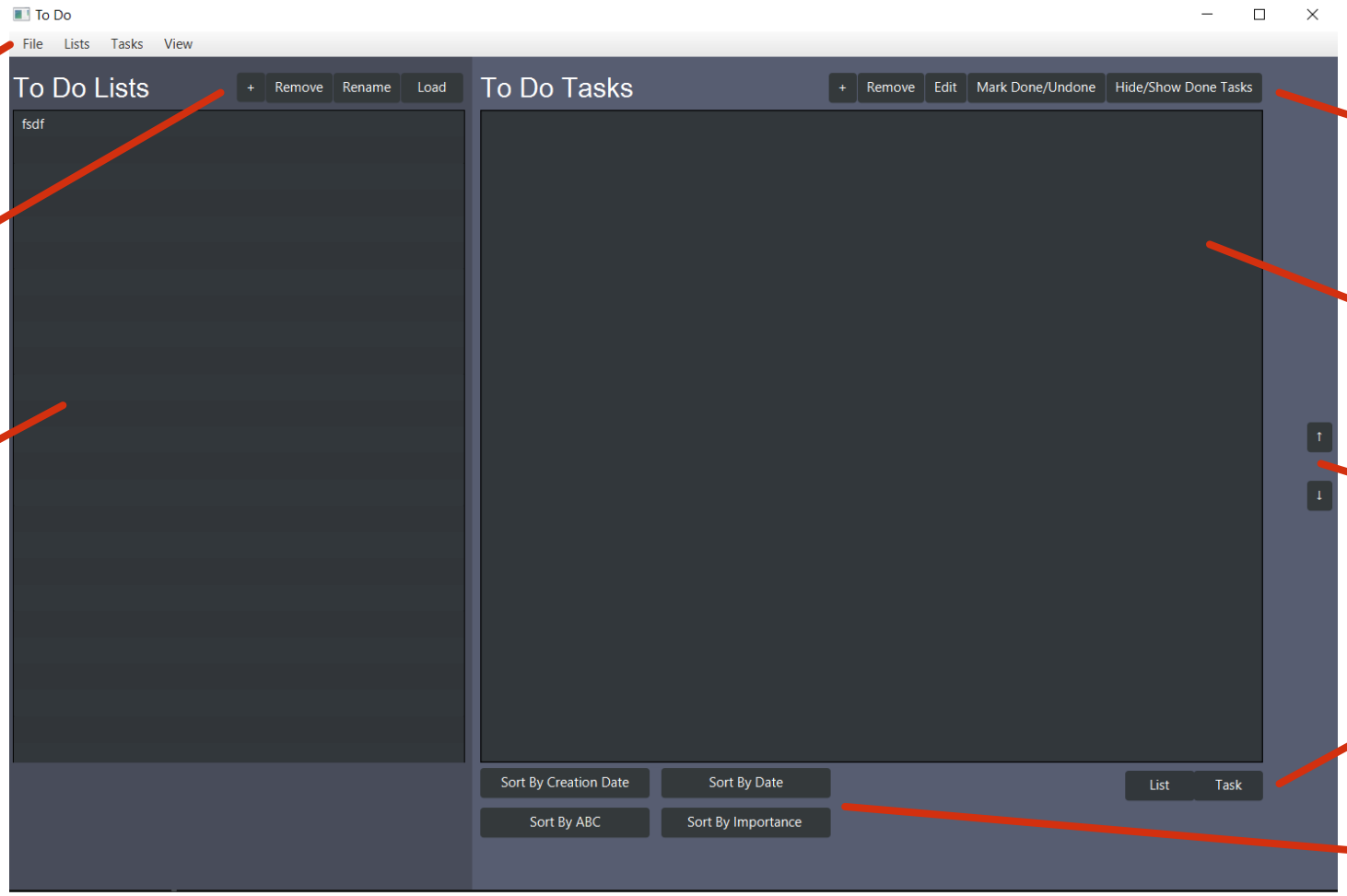
ListView object holding current ToDoTask objects

Moving task placement control, for manual ordering

Random color changer for both list and task panel, try to click it!

Sorting control panel

FINAL PRODUCT



menu bar

lists controls panel

ListView object holding all the ToDoList objects

tasks controls panel

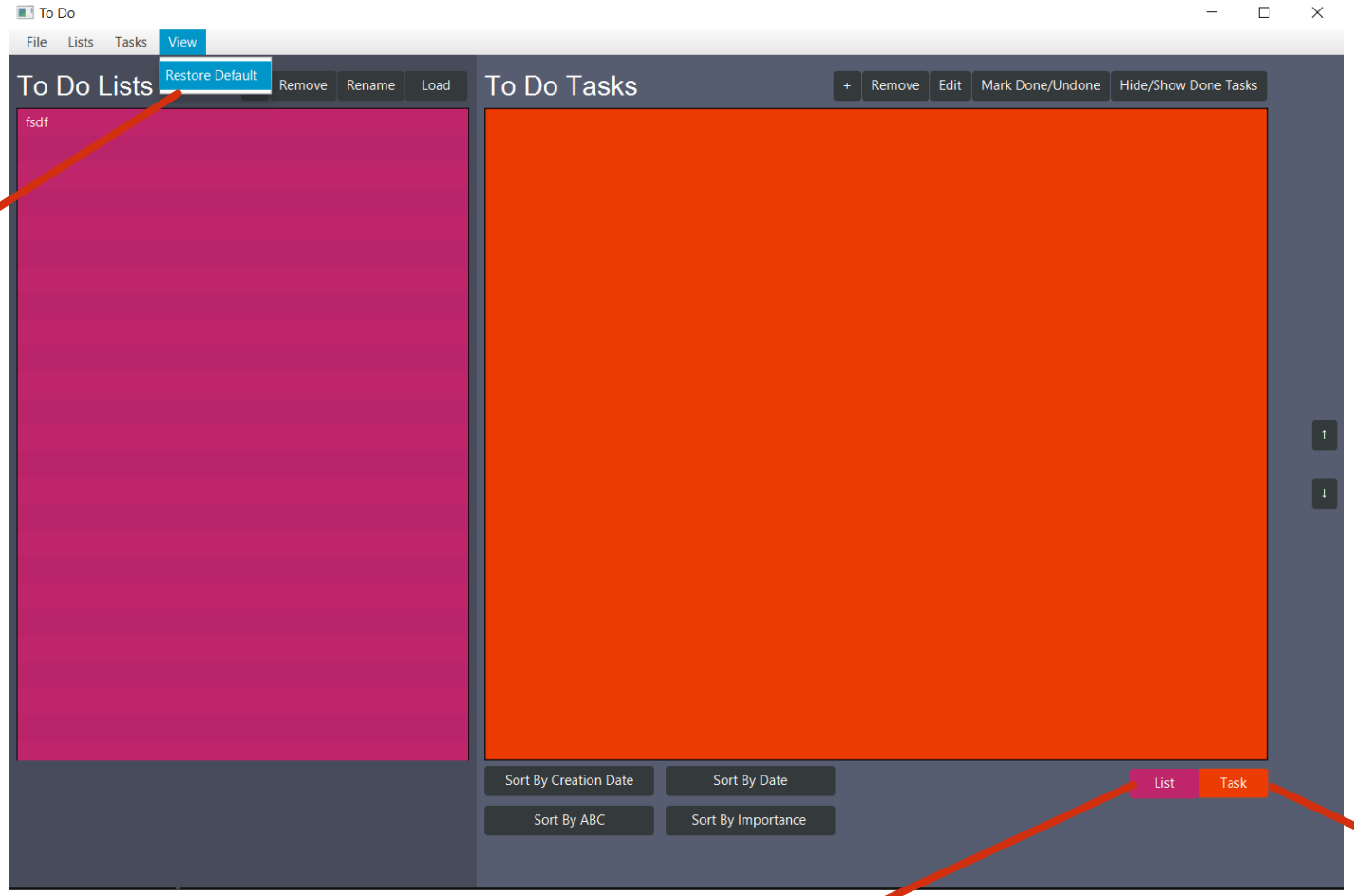
ListView object holding current ToDoTask objects

Moving task placement control, for manual ordering

Random color changer for both list and task panel, try to click it!

Sorting control panel

FINAL PRODUCT

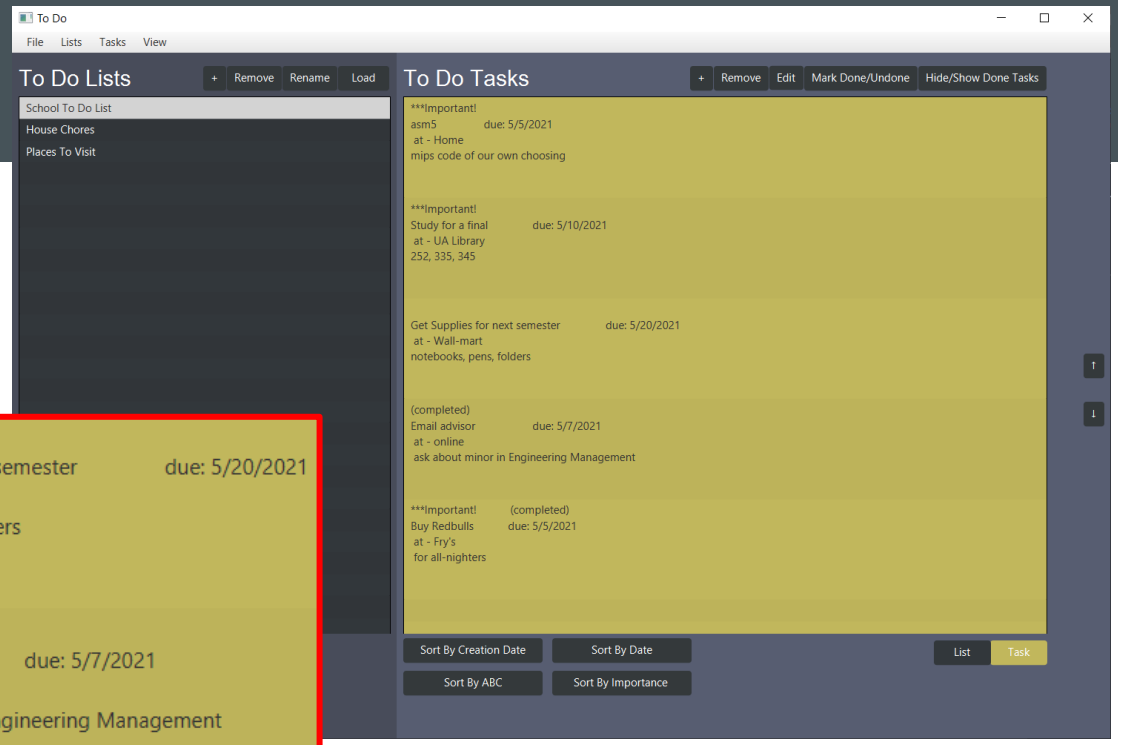


Restore default colors

Control color of right element

Control color of left element

FINAL PRODUCT



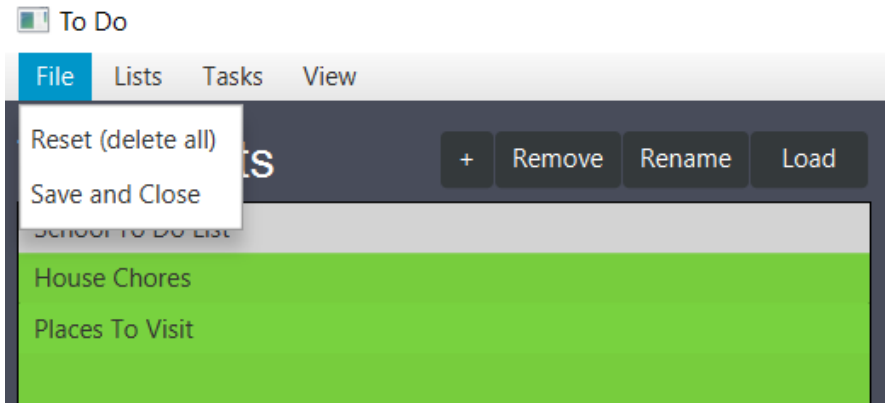
Status: important and/or completed

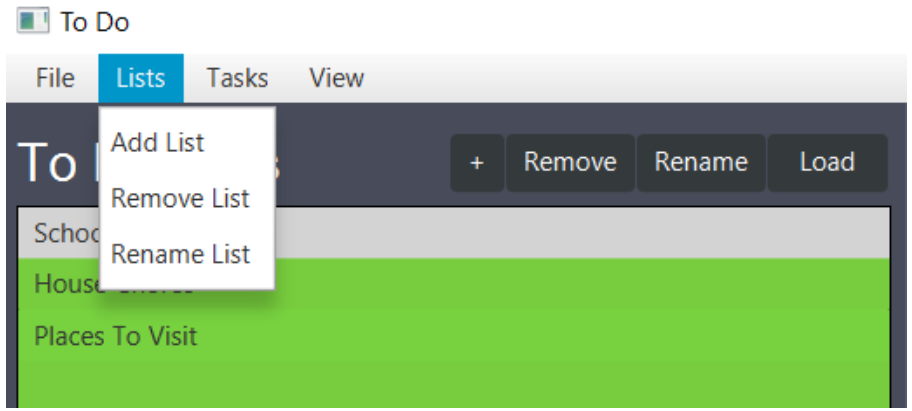
Task name - title

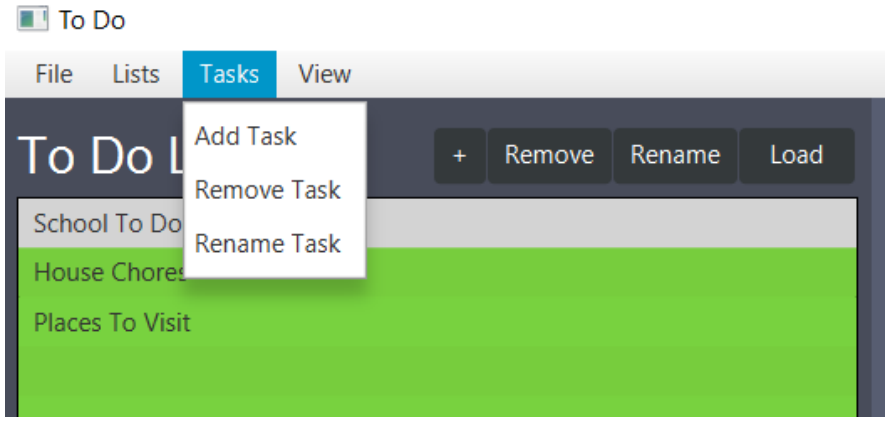
Task location

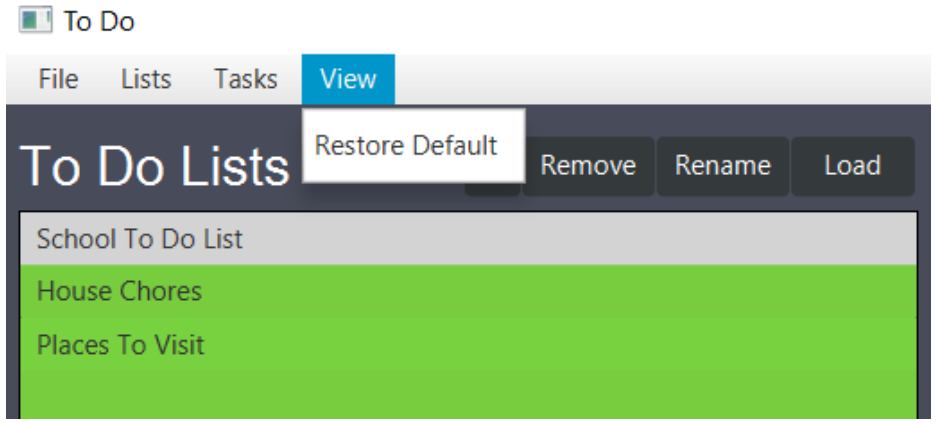
Task notes

FINAL PRODUCT – MENU ITEMS



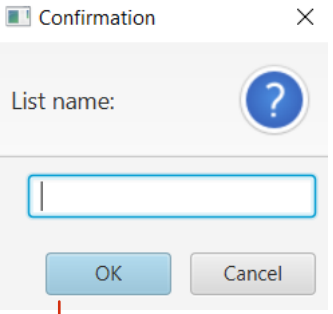




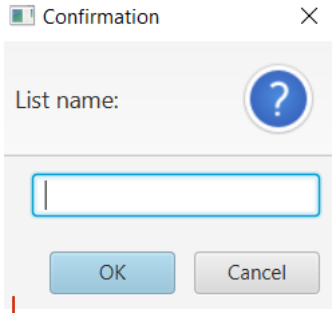


FINAL PRODUCT – INPUT PROMPTS

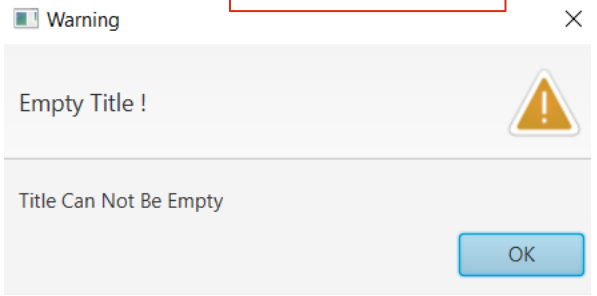
Add List



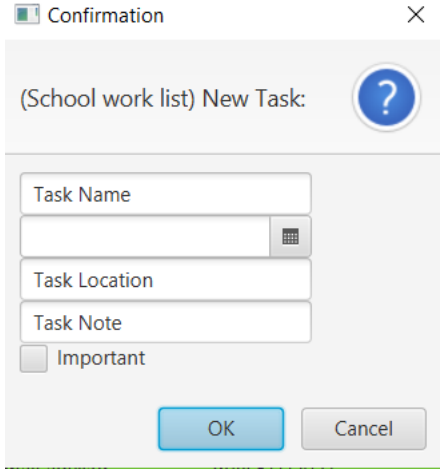
Rename List



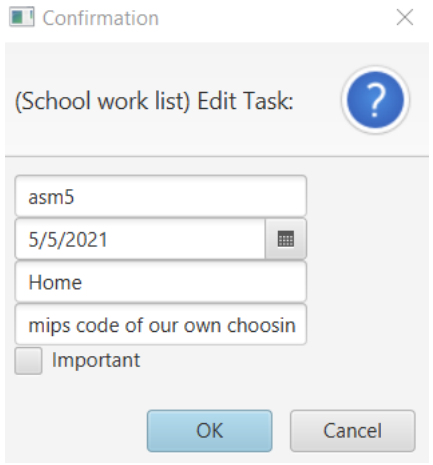
Appropriate error messages if empty



Add Task



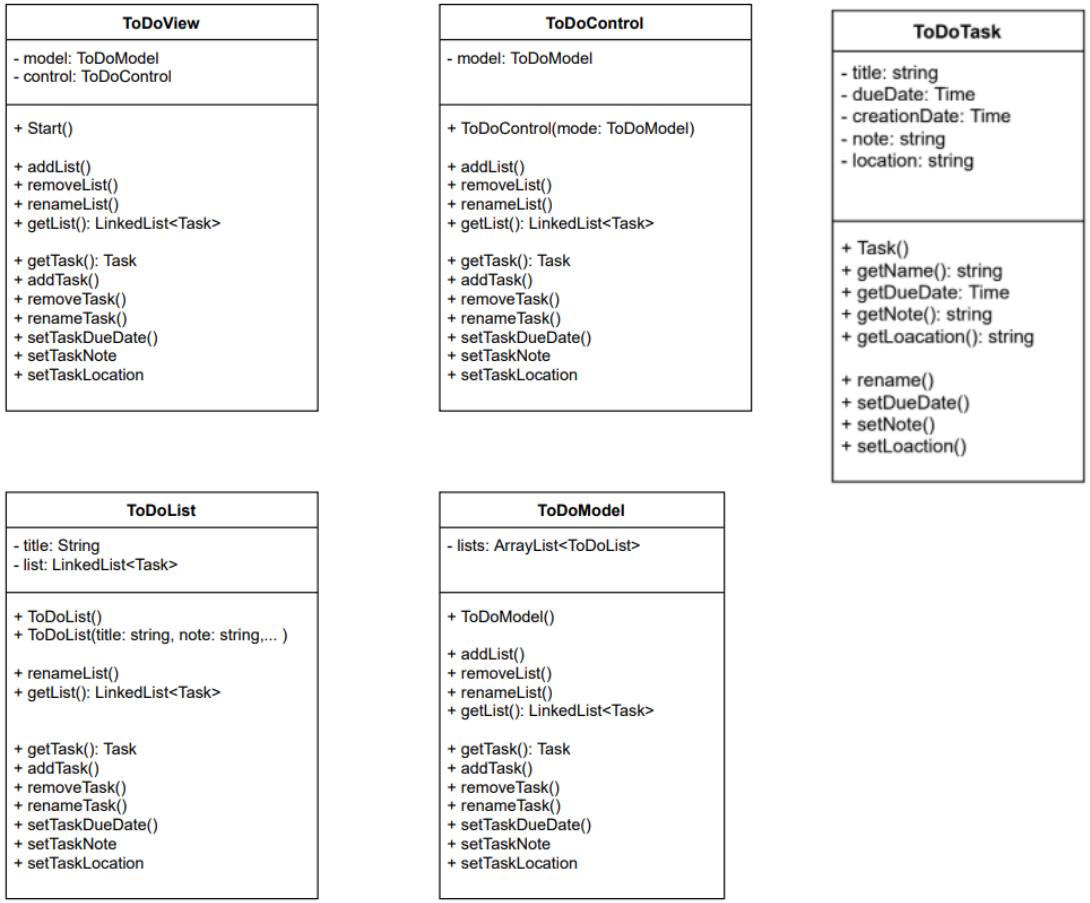
Edit Task



SUPPORTING DOCUMENTS

UML, NOTES AND OTHER DOCS

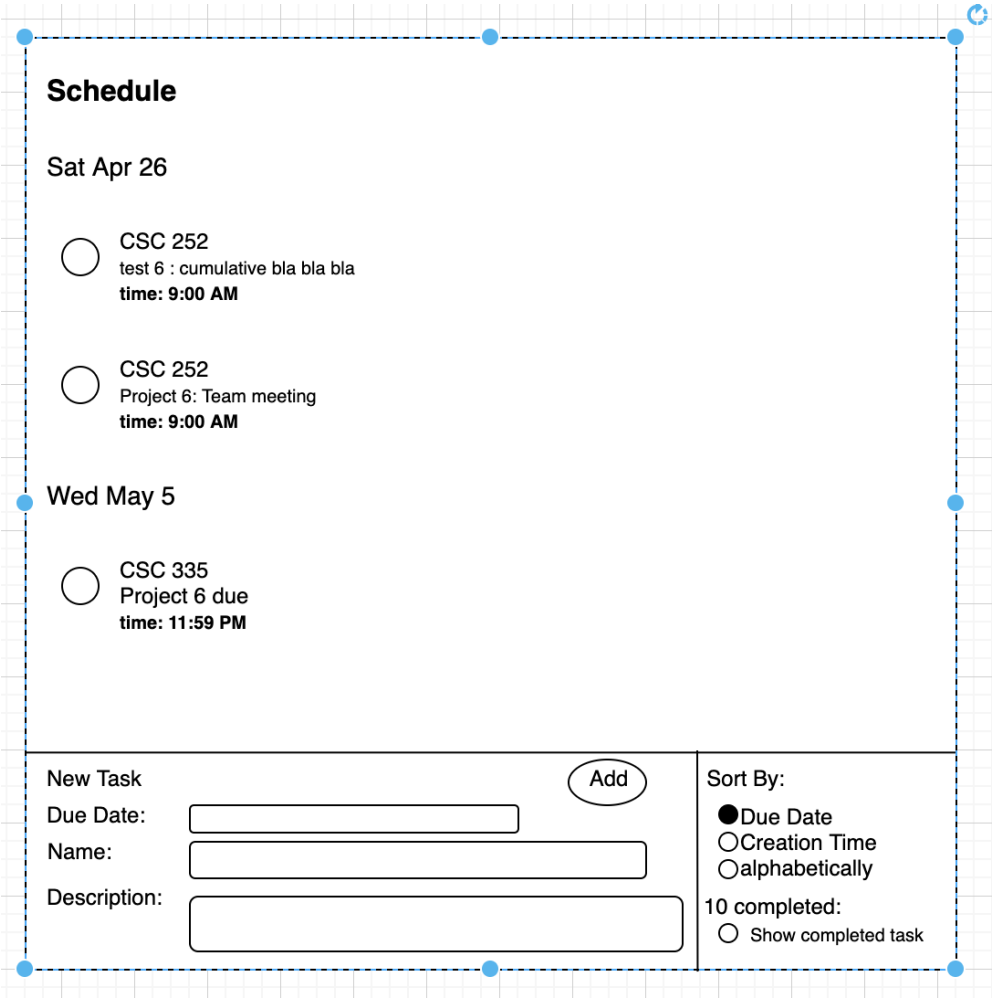
Initial class diagram



SUPPORTING DOCUMENTS

UML, NOTES AND OTHER DOCS

Initial UI design



SUPPORTING DOCUMENTS

UML, NOTES AND OTHER DOCS

Initial methods assignment

Model

void addList (String list) - add a list to the model lists - **Allen - complete**

ToDoList removeList(string id) - take name of the list as String and remove it from the lists, return list that is removed or null if it does not exist - **Kyelse**

void renameList(String oldName, String newName) - takes original id and replaced it with newName param **- Ofer**

ToDoList getList(String name) - searches through lists and returns the list with the param name - **quang**

ToDoTask getTask (String nameList, String nameTask) - return the task nameTask object found in the list nameList found in model private lists attribute - **Allen - completed**
-
boolean addTask (String nameList, ToDoTask newTask) - takes id of list, pass in a new task object. The methods adds it to the correct list - **Kyelse**

ToDoTask removeTask (String listName, taskName) - remove the task found in list param - **Ofer**

boolean renameTask (String nameList, String nameTask, String newName) - **Quang**

boolean setTaskDueDate (String listName, String taskName, *String dueDate) - change the due date found at list list name task task name to due date * java.util.Date explore as alternative - **Allen - complete with type error - need to discuss type error**

boolean setTaskNote (String listName, String taskName) - change the tasknote of the input params

- **Kyelse**

boolean setTaskLocation (String listName, taskName) - change the task location of the input params

- **Ofer**

Void saveAll() - Kyelse

SUPPORTING DOCUMENTS

UML, NOTES AND OTHER DOCS

Initial methods assignment

Controller:
void addList (String name)

- **Quang**

ToDoList removeList (String name)

- **Allen - completed**

boolean renameList (String oldName, String newName)

-Kyelse

ToDoList getList(String name)

Ofer

boolean addTask (String nameList, ToDoTask newTask)

quang

boolean sortTask(String nameList, String whatToSort)

Allen - completed - need to assign helper methods

ToDoTask removeTask (String listName, taskName) - remove the task found in list param

Kyelse

boolean renameTask (String nameList, String nameTask, String newName) -

Ofer

boolean setTaskDueDate (String listName, String taskName, *String dueDate) - change the due date found at list list name task task name to due date * java.util.Date explore as alternative

-quang

boolean setTaskNote (String listName, String taskName) - change the tasknote of the input params

-Allen - completed

boolean setTaskLocation (String listName, taskName) - change the task location of the input params

-Kyelse

SUPPORTING DOCUMENTS

UML, NOTES AND OTHER DOCS

Initial methods assignment

View:

List

void renameList(String oldName, String newName) - takes original id and replaced it with newName param **- Ofer**

ToDoTask getTask (String nameList, String nameTask) - return the task nameTask object found in the list nameList found in model private lists attribute **- Allen - completed**
-
boolean addTask (String nameList, ToDoTask newTask) - takes id of list, pass in a new task object. The methods adds it to the correct list - **Kyelse**

ToDoTask removeTask (String listName, taskName) - remove the task found in list param **- Ofer**

boolean renameTask (String nameList, String nameTask, String newName) - **Quang**

boolean setTaskDueDate (String listName, String taskName, *String dueDate) - change the due date found at list list name task task name to due date * java.util.Date explore as alternative - **Allen - not completed**

boolean setTaskNote (String listName, String taskName) - change the tasknote of the input params
Kyelse

boolean setTaskLocation (String listName, taskName) - change the task location of the input params

-Kyelse

THANK YOU