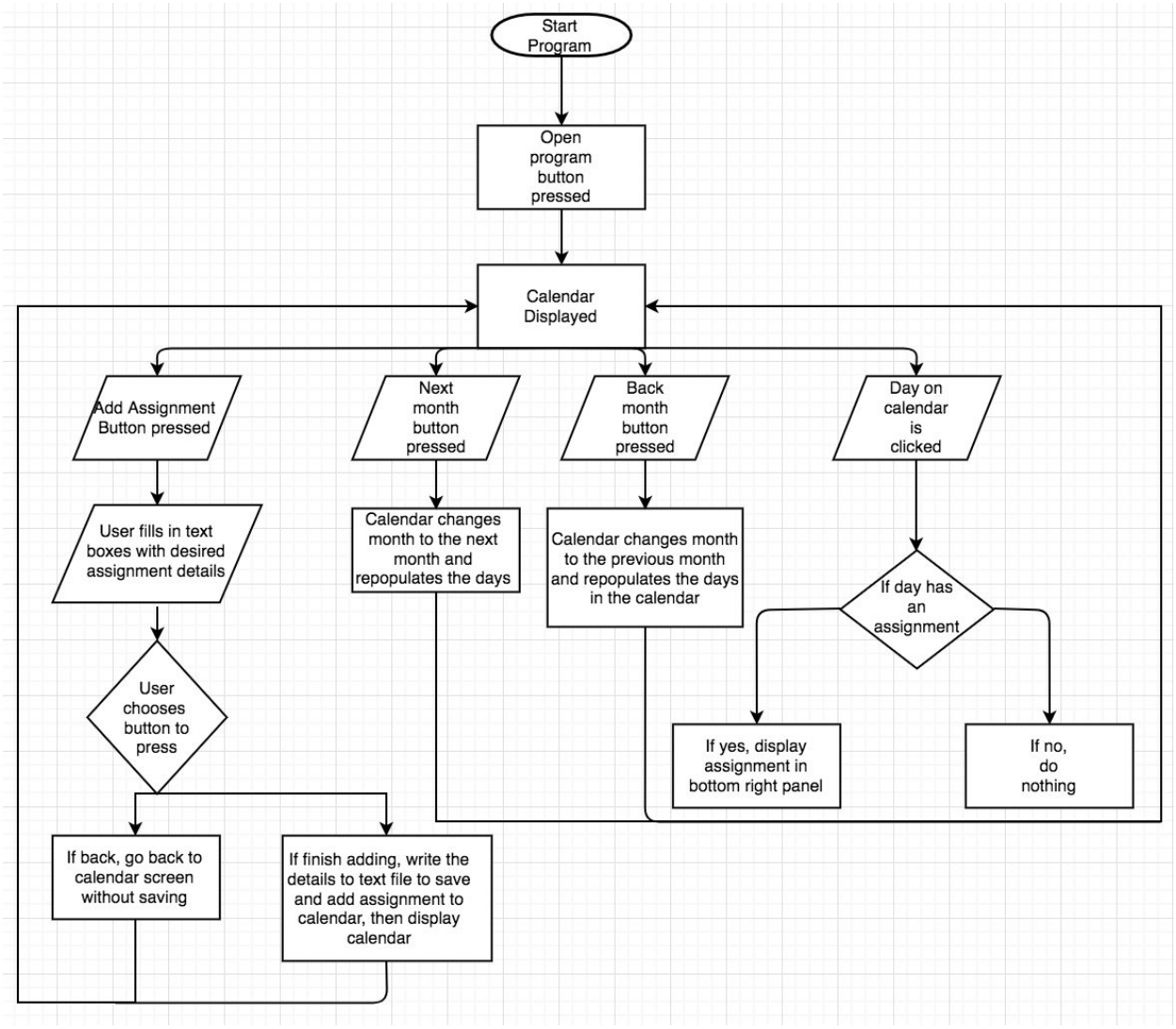
## Part B: Design Document

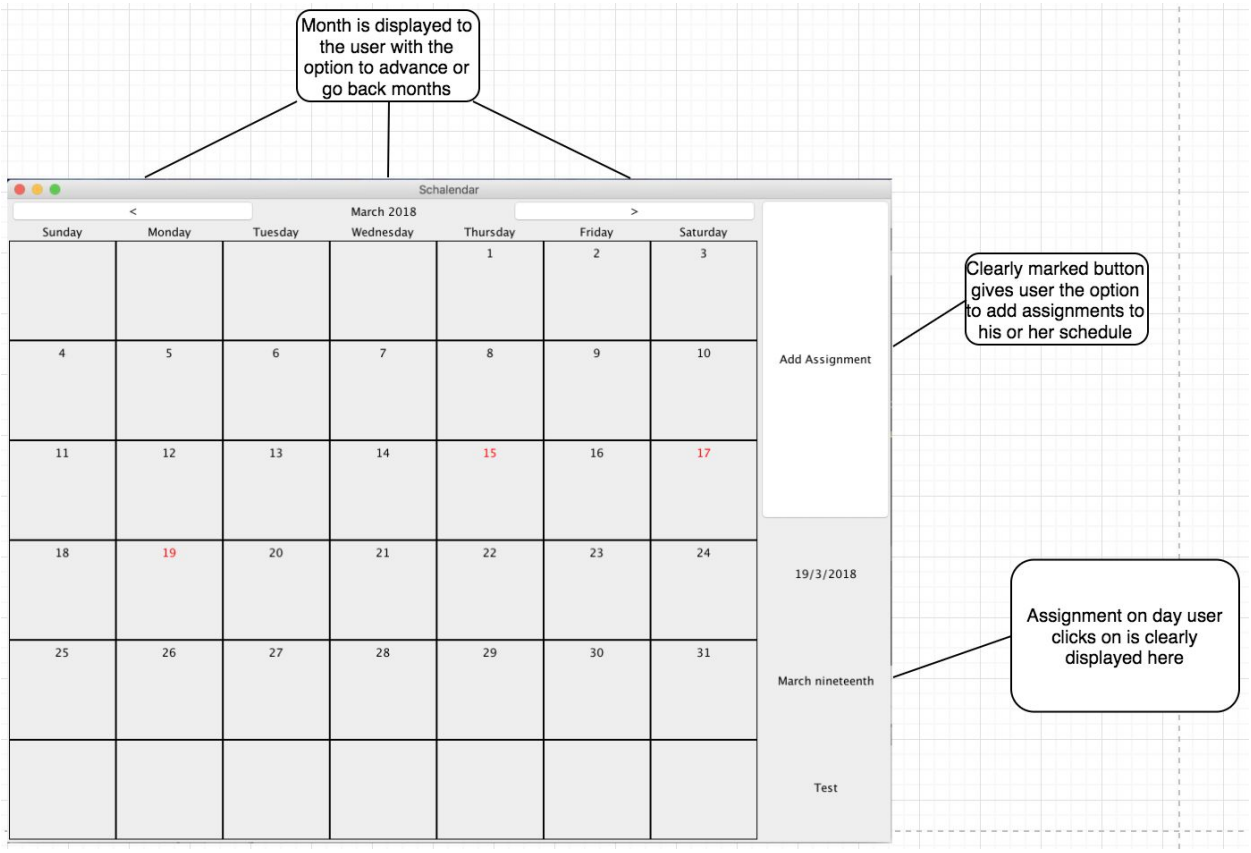
## Test Plan:

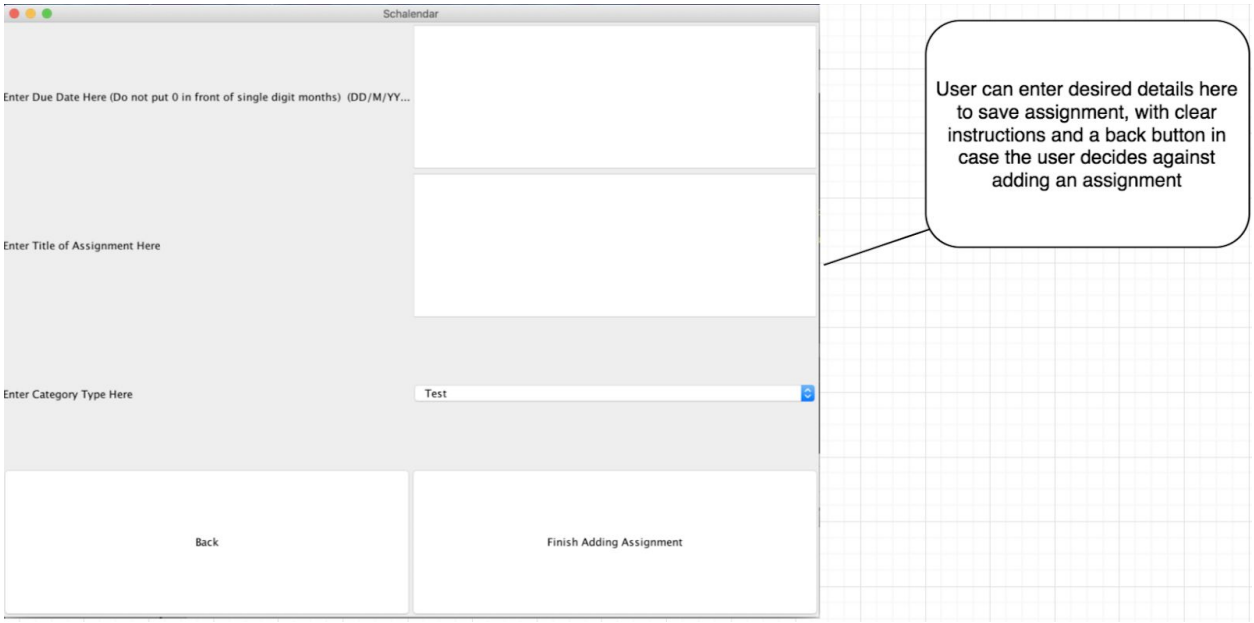
| Test Type | Nature of Test | Example |
|---|---|---|
| Upon starting the program the menu should come up | Check that menu opens properly | “Open Schalendar” |
| Upon pressing open button the program should open and display the main screen | Check that main screen displays | Visible calendar and main panels |
| Upon pressing the add assignment button, the add assignment screen should display | Check that add assignment screen displays | Visible add assignment panels |
| Upon typing in the parameters, an assignment should be saved with the user input | Check that saving assignment method works | “19/3/2018”<br>“Math homework”<br>“Homework” |
| Upon clicking the next month or back month button, the month should change to the next month or previous month and the calendar should change days to match | Check that changing the month works | User clicks “>” or “<” button |
| Upon clicking on day without assignment, nothing should happen | Check that false assignments aren’t displayed | User clicks day that is not highlighted |
| Upon clicking on day with an assignment, assignment on that day should display | Check that displaying assignment method works | User clicks highlighted day |

# Flowchart:


Start Program
Open program button pressed
Calendar Displayed
Add Assignment Button pressed
Next month button pressed
Back month button pressed
Day on calendar is clicked
User fills in text boxes with desired assignment details
Calendar changes month to the next month and repopulates the days
Calendar changes month to the previous month and repopulates the days in the calendar
If day has an assignment
User chooses button to press
If yes, display assignment in bottom right panel
If no, do nothing
If back, go back to calendar screen without saving
If finish adding, write the details to text file to save and add assignment to calendar, then display calendar

# Graphical Visualization:



## Key Algorithms:

```
    numCal = firstDayOfMonth;
    lastDay = Calendar.getInstance().getActualMaximum(Calendar.DAY_OF_MONTH);
    System.out.println("Last day of month: "+ lastDay);

    for(int i = 0; i<6; i++){
        for(int j = 0; j<7; j++){
            if(counter <= lastDay){
                if(j == numCal-1 && i== 0){
                    numbering[i][j].setText(Integer.toString(counter));
                    counter++;
                    if(numCal<7){
                        numCal++;
                    }else{
                        numCal = 0;
                    }
                }else if(i!=0){
                    numbering[i][j].setText(Integer.toString(counter));
                    counter++;
                }
            }
        }
    }
}
```

This algorithm is used to populate the calendar with the days of the month in the proper days. The way this works is by looping through the JLabels that contain the labels in charge of numbering the days, the “numbering” array. Then it checks that int “counter” is less than the last day of the month, checked by int “lastDay”. This is to make sure that the number of the day being added to the calendar is within the bounds of the last day of the month. Then it checks if the variables i and j meet the following two conditions: is i equal to zero, meaning it’s the first week of the month, and that j is equal to the first day of the month (numCal -1). If both of these conditions are met, then it sets the label array “numbering[i][j]” to the variable “counter” an integer that starts at 1. After it does this, it increases counter. As long as variable numCal is less than seven, meaning it still falls in the bounds of a week, it adds one to numCal so that the next day is labeled with counter + 1. If numCal is equal to or greater than 7, it resets numCal to 0, or the first day of the week. If i does not equal 0, meaning it has already passed the first week, than it loops through labeling each day with the proper counter until the last day of the month has been hit.

**Word Count:** 438