**Developer Test Plan for ‘To Do List’ project**

| **File Name** |  |
| --- | --- |
| **Repository** | https://jaseincyberspace.github.io/BIT707\_documentation/ |

**Version Control**

| Version | Description | Date | Author |
| --- | --- | --- | --- |
| 1-0 | Initial Version | 30/05/2022 | Jason Norton |

**Test Data (Note: The supplied DDL file does not include entries for tasks 18-24. They will need to be entered manually as their DueDate values are relative to the current date.)**

| TaskNumber | TaskName | TaskDescription | DueDate | Status |
| --- | --- | --- | --- | --- |
| 1 | Task name 1 | Task description 1 | 2022-05-30 | FALSE |
| 2 | Task name 2 | Task description 2 | 2022-06-01 | TRUE |
| 3 | Task name 3 | Task description 3 | 2022-06-01 | FALSE |
| 4 | Task name 4 | Task description 4 | 2022-06-01 | FALSE |
| 5 | Task name 5 | Task description 5 | 2022-06-02 | TRUE |
| 6 | Task name 6 | Task description 6 | 2022-06-02 | TRUE |
| 7 | Task name 7 | Task description 7 | 2022-06-03 | FALSE |
| 8 | Task name 8 | Task description 8 | 2022-06-04 | TRUE |
| 9 | Task name 9 | Task description 9 | 2022-06-05 | FALSE |
| 10 | Task name 10 | Task description 10 | 2022-06-06 | TRUE |
| 11 | Task name 11 | Task description 11 | 2022-06-06 | FALSE |
| 12 | Task name 12 | Task description 12 | 2022-06-07 | TRUE |
| 13 | Task name 13 | Task description 13 | 2022-06-08 | FALSE |
| 14 | Task name 14 | Task description 14 | 2022-06-09 | TRUE |
| 15 | Task name 15 | Task description 15 | 2022-06-10 | FALSE |
| 16 | Task name 16 | Task description 16 | 2022-06-11 | TRUE |
| 17 | Task name 17 | Task description 17 | 2022-06-12 | FALSE |
| 18 | Task name 18 | Task description 18 | Monday this week | TRUE |
| 19 | Task name 19 | Task description 19 | Sunday this week | FALSE |
| 20 | Task name 20 | Task description 20 | Next Monday | TRUE |
| 21 | Task name 21 | Task description 21 | Next Sunday | FALSE |
| 22 | Task name 22 | Task description 22 | Yesterday | TRUE |
| 23 | Task name 23 | Task description 23 | Today | FALSE |
| 24 | Task name 24 | Task description 24 | Tomorrow | TRUE |

**Interface Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-IT.01 | | |
| **Test Case Description:** | At application start, display ListView to the user inside an MDI (multiple document interface) container | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application | | |
| **Post Conditions:** | The *List View* screen is displayed. JTable column headers for *Status, Task* and *Due Date* should be visible. *Add Task* button should also be visible on screen. | | |
| **Expected Result:** | The main JFrame displays the application header and navigation buttons. A JInternalFrame is displayed inside it with a scrollable ListView table inside that | **Actual Result:** | **ERROR:**  *Add Task* button is being cut off the bottom of the screen. JFrame needs to be resized.  **FIXED:**  JFrames now display all components correctly |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **01/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-IT.02 | | |
| **Test Case Description:** | Display CalendarView to the user inside an MDI container | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) | | |
| **Post Conditions:** | The *Calendar View* screen is displayed. *Add Task* button should be visible on screen. A date range is displayed along with a date reset button. | | |
| **Expected Result:** | The main JFrame displays the application header and navigation buttons. A JInternalFrame is displayed inside it with a scrollable CalendarView table inside that. The date range displayed should correspond to Monday – Sunday of the current week | **Actual Result:** | **ERROR:**  Date range buttons have excessive blank space above and below them.  **FIXED:**  Spacing around date range buttons is now consistent with the rest of the application |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **01/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-IT.03 | | |
| **Test Case Description:** | Successfully navigate from CalendarView to ListView | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Click the list icon (to the immediate left of calendar icon) | | |
| **Post Conditions:** | The *List View* screen is displayed. *Add Task* button should be visible on screen. | | |
| **Expected Result:** | The main JFrame displays the application header and navigation buttons. A JInternalFrame is displayed inside it with a scrollable ListView table inside that. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **01/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-IT.04 | | |
| **Test Case Description:** | ListView components resize to fill available space when application window is maximised or minimised | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the ‘Maximise’ button to make the application window to fill the computer screen 3. Click the ‘Restore Down’ button to return the application window to its original size | | |
| **Post Conditions:** | The *List View* screen is displayed. *Add Task* button should be visible on screen. | | |
| **Expected Result:** | The ListView table should dynamically resize it’s width and height to fill the available space as the screen size increases and decreases. The Add Task button should always remain in place at the bottom of the screen. | **Actual Result:** | **ERROR:**  Table is not resizing width or height. *Add Task* button is being pulled up from the bottom of the screen to sit just underneath the table  **FIXED:**  Table now resizes up and down to fill available window space. The Add Task button remains in place at the bottom of the screen |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-IT.05 | | |
| **Test Case Description:** | CalendarView components resize to fill available space when application window is maximised or minimised | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right) 3. Click the ‘Maximise’ button to make the application window to fill the computer screen 4. Click the ‘Restore Down’ button to return the application window to its original size | | |
| **Post Conditions:** | The *Calendar View* screen is displayed. *Add Task* button should be visible on screen. | | |
| **Expected Result:** | The CalendarView table should dynamically resize it’s width and height to fill the available space as the screen size increases and decreases. The Add Task button should always remain in place at the bottom of the screen. | **Actual Result:** | **ERROR:**  Table is not resizing width or height. *Add Task* button is being pulled up from the bottom of the screen to sit just underneath the table  **FIXED:**  Table now resizes up and down to fill available window space. The Add Task button remains in place at the bottom of the screen |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.06 | | |
| **Test Case Description:** | AddTask dialog is displayed when Add Task button is clicked from ListView | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the *Add Task* button (bottom left corner) | | |
| **Post Conditions:** | The *List View* screen is displayed with an *Add Task* modal dialog box on top. | | |
| **Expected Result:** | The *Add Task* dialog box pops up on screen in the centre of the *ListView* table.  It should be a modal dialog so that the user is unable to click on anything outside of the box. | **Actual Result:** | **ERROR:**  The dialog box is being displayed in the top left corner of the screen rather than the centre of the *ListView* table.  **FIXED:**  *Add Task* dialog now pops up in the centre of the JTable, even when the application window has been moved around on screen. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.07 | | |
| **Test Case Description:** | AddTask dialog is displayed when Add Task button is clicked from CalendarView | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Click the *Add Task* button (bottom left corner) | | |
| **Post Conditions:** | The *Calendar View* screen is displayed with an *Add Task* modal dialog box on top. | | |
| **Expected Result:** | The *Add Task* dialog box pops up on screen in the centre of the *CalendarView* table.  It should be a modal dialog so that the user is unable to click on anything outside of the box. | **Actual Result:** | **ERROR:**  The dialog box is being displayed in the top left corner of the screen rather than the centre of the *CalendarView* table.  **FIXED:**  *Add Task* dialog now pops up in the centre of the JTable, even when the application window has been moved around on screen. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.08 | | |
| **Test Case Description:** | AddTask dialog disappears from List View when save button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the *Add Task* button (bottom left corner) 3. Use test data relating to TaskNumber 23 to fill the respective fields 4. Click the *Save* button | | |
| **Post Conditions:** | The *Add Task* dialog box disappears and the *List View* screen is displayed. | | |
| **Expected Result:** | The *Add Task* dialog box is closed and the ListView table is displayed to the user. The newly added task should be showing in the list | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.09 | | |
| **Test Case Description:** | AddTask dialog disappears from Calendar View when save button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Click the *Add Task* button (bottom left corner) 4. Use test data relating to TaskNumber 24 to fill the respective fields 5. Click the *Save* button | | |
| **Post Conditions:** | The *Add Task* dialog box disappears and the *Calendar View* screen is displayed. | | |
| **Expected Result:** | The *Add Task* dialog box is closed and the CalendarView table is displayed to the user. The newly added task should be showing in the list | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.10 | | |
| **Test Case Description:** | AddTask dialog disappears from ListView when cancel button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the *Add Task* button (bottom left corner) 3. Click the *Cancel* button | | |
| **Post Conditions:** | The *Add Task* dialog box disappears and the *List View* screen is displayed. | | |
| **Expected Result:** | The *Add Task* dialog box is closed and the ListView table is displayed to the user. No new tasks should be showing in the list | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.11 | | |
| **Test Case Description:** | AddTask dialog disappears from Calendar View when cancel button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Click the *Add Task* button (bottom left corner) 4. Click the *Cancel* button | | |
| **Post Conditions:** | The *Add Task* dialog box disappears and the *Calendar View* screen is displayed. | | |
| **Expected Result:** | The *Add Task* dialog box is closed and the CalendarView table is displayed to the user. No new tasks should be showing in the list | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-IT.12 | | |
| **Test Case Description:** | Font colour and background colour changes when a ListView table row is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click on the third task from the top of the list 3. Click on the sixth task from the top of the list | | |
| **Post Conditions:** | The *sixth task in the list is showing as selected* | | |
| **Expected Result:** | When the third task is clicked on its text colour should turn white and its background colour should turn dark grey. When the sixth task is clicked on the third one should return to its original colour scheme while the sixth one should have white text and a dark grey background | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-IT.13 | | |
| **Test Case Description:** | Font colour and background colour changes when a CalendarView table row is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right) 3. Click on the first task from the top of the list 4. Click on the second task from the top of the list | | |
| **Post Conditions:** | The *second task in the list is showing as selected* | | |
| **Expected Result:** | When the first task is clicked its text colour should turn white and its background colour should turn dark grey. When the second task is clicked the first one should return to its original colour scheme while the second one should change to white text and a dark grey background | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-IT.14 | | |
| **Test Case Description:** | *Delete* button is displayed when a ListView table row is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click on the second task from the top of the list | | |
| **Post Conditions:** | The *second task in the list is showing as selected. The DeleteSelected* button is displayed. | | |
| **Expected Result:** | When the task is clicked its text colour should turn white and its background colour should turn dark grey. The ’Delete Selected’ button should be displayed in the bottom right corner of the JFrame. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-IT.15 | | |
| **Test Case Description:** | *Delete* button is displayed when a CalendarView table row is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right) 3. Click on the second task from the top of the list | | |
| **Post Conditions:** | The *second task in the list is showing as selected. The DeleteSelected* button is displayed. | | |
| **Expected Result:** | When the task is clicked its text colour should turn white and its background colour should turn dark grey. The ’Delete Selected’ button should be displayed in the bottom right corner of the JFrame. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.08-IT.16 | | |
| **Test Case Description:** | ViewTask dialog is displayed on ListView when table row item is double clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Double click on the second task from the top of the list (task 2) | | |
| **Post Conditions:** | The *View Task* dialog box appears on top of the *List View* screen. | | |
| **Expected Result:** | A modal dialog box is displayed in the centre of the table. Its field values for task name, description and date should be populated with the values of the task in the row that was double clicked | **Actual Result:** | **ERROR:**  The TaskName field contains HTML tags used for adding strikethrough as well as the task name.  **FIXED:**  HTML tags are now stripped from TaskName field in *View Task* dialog |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.08-IT.17 | | |
| **Test Case Description:** | ViewTask dialog is displayed on CalendarView when table row item is double clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Double click on the second task from the top of the list | | |
| **Post Conditions:** | The *View Task* dialog box appears on top of the *Calendar View* screen. | | |
| **Expected Result:** | A modal dialog box is displayed in the centre of the table. Its field values for task name, description and date should be populated with the values of the task in the row that was double clicked | **Actual Result:** | **ERROR:**  The TaskName field contains HTML tags used for adding strikethrough as well as the task name.  **FIXED:**  HTML tags are now stripped from TaskName field in *View Task* dialog |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.18 | | |
| **Test Case Description:** | AddTask dialog disappears from ListView when cancel button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the *Add Task* button (bottom left corner) 3. Click the *Cancel* button | | |
| **Post Conditions:** | The *Add Task* dialog box disappears and the *List View* screen is displayed. | | |
| **Expected Result:** | The *Add Task* dialog box is closed and the ListView table is displayed to the user. No new tasks should be showing in the list | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.19 | | |
| **Test Case Description:** | AddTask dialog disappears from CalendarView when cancel button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click on the calendar icon (top right corner) 3. Click the *Add Task* button (bottom left corner) 4. Click the *Cancel* button | | |
| **Post Conditions:** | The *Add Task* dialog box disappears and the *Calendar View* screen is displayed. | | |
| **Expected Result:** | The *Add Task* dialog box is closed and the CalendarView table is displayed to the user. No new tasks should be showing in the list | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-IT.20 | | |
| **Test Case Description:** | *EditTask* dialog is displayed on ListView when edit button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Double click on the first task at the top of the list to open the *ViewTask* dialog 3. Click the edit icon (top right corner of *ViewTask* dialog) | | |
| **Post Conditions:** | The *EditTask* dialog is displayed on top of the ListView. | | |
| **Expected Result:** | The *ViewTask* dialog box is closed and the *EditTask* modal dialog is displayed to the user on top of the ListView table. Its fields should be pre-populated as per the data in the *ViewTask* dialog fields | **Actual Result:** | **ERROR:**  The TaskName field contains HTML tags used for adding strikethrough as well as the task name.  **FIXED:**  HTML tags are now stripped from TaskName field in *EditTask* dialog |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-IT.21 | | |
| **Test Case Description:** | *EditTask* dialog is displayed on CalendarView when edit button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Double click on the first task at the top of the list to open the *ViewTask* dialog 4. Click the edit icon (top right corner of *ViewTask* dialog) | | |
| **Post Conditions:** | The *EditTask* dialog is displayed on top of the CalendarView. | | |
| **Expected Result:** | The *ViewTask* dialog box is closed and the *EditTask* modal dialog is displayed to the user on top of the CalendarView table. Its fields should be pre-populated as per the data in the *ViewTask* dialog fields | **Actual Result:** | **ERROR:**  The TaskName field contains HTML tags used for adding strikethrough as well as the task name.  **FIXED:**  HTML tags are now stripped from TaskName field in *EditTask* dialog |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-IT.22 | | |
| **Test Case Description:** | *EditTask* dialog disappears when save button is clicked. Edits are displayed on ListView | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Double click on the first task at the top of the list to open the *ViewTask* dialog 3. Click the edit icon (top right corner of *ViewTask* dialog) 4. Change the task name to “Task Name 200” 5. Change the task date to today’s date 6. Click the save button (bottom left corner of dialog) | | |
| **Post Conditions:** | The ListView table is displayed. | | |
| **Expected Result:** | The *EditTask* dialog box is closed and the the ListView table is displayed on screen. The first task in the list has task name “Task Name 200” and today’s date. | **Actual Result:** | **ERROR:**  The task is being updated in the task list and database but not displayed on the table.  **FIXED:**  Edits made to items in the ListView are now displayed to the user as per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-IT.23 | | |
| **Test Case Description:** | *EditTask* dialog disappears when save button is clicked. Edits are displayed on CalendarView | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Double click on the first task at the top of the list to open the *ViewTask* dialog 4. Click the edit icon (top right corner of *ViewTask* dialog) 5. Change the task name to “Task Name 400” 6. Change the task date to tomorrow’s date 7. Click the save button (bottom left corner of dialog) | | |
| **Post Conditions:** | The CalendarView table is displayed. | | |
| **Expected Result:** | The *EditTask* dialog box is closed and the the ListView table is displayed on screen. The first task in the list has task name “Task Name 400” and tomorrow’s date. | **Actual Result:** | **ERROR:**  The task is being updated in the task list and database but not displayed on the table.  **FIXED:**  Edits made to items in the ListView are now displayed to the user as per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-IT.24 | | |
| **Test Case Description:** | *EditTask* dialog disappears from ListView when cancel button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Double click on the first task at the top of the list to open the *ViewTask* dialog 3. Click the edit icon (top right corner of *ViewTask* dialog) 4. Click the cancel button (bottom left corner of dialog) | | |
| **Post Conditions:** | The ListView table is displayed. | | |
| **Expected Result:** | The *EditTask* dialog box is closed and the the ListView table is displayed on screen. No changes are made to the items displayed in the list. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-IT.25 | | |
| **Test Case Description:** | *EditTask* dialog disappears from CalendarView when cancel button is clicked | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Double click on the first task at the top of the list to open the *ViewTask* dialog 4. Click the edit icon (top right corner of *ViewTask* dialog) 5. Click the cancel button (bottom left corner of dialog) | | |
| **Post Conditions:** | The CalendarView table is displayed. | | |
| **Expected Result:** | The *EditTask* dialog box is closed and the the CalendarView table is displayed on screen. No changes are made to the items displayed in the list. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.26 | | |
| **Test Case Description:** | Confirm tasks added to ListView table are also added to CalendarView table | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | New Task:  TaskName: “Newly Added Task”, TaskDescription: “This is a new task”, TaskDate: today’s date | | |
| **Test Steps:** | 1. Run the application 2. Click *Add Task* button (bottom left) 3. Enter new task test data into the dialog fields 4. Click S*ave* 5. Click calendar icon | | |
| **Post Conditions:** | The CalendarView table is displayed. | | |
| **Expected Result:** | The new taskis visible in the calendarView table | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-IT.27 | | |
| **Test Case Description:** | Confirm tasks added to CalendarView table are also added to ListView table | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | New Task:  TaskName: “Another new task”, TaskDescription: “This is another new task”, TaskDate: today’s date | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Click *Add Task* button (bottom left) 4. Enter new task test data into the dialog fields 5. Click S*ave* 6. Click the list icon (top right corner beside calendar icon) | | |
| **Post Conditions:** | The ListView table is displayed. | | |
| **Expected Result:** | The new taskis visible in the listView table | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-IT.28 | | |
| **Test Case Description:** | Confirm tasks edited in the ListView table are also edited in the CalendarView table | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document  Edit Task Data:  TaskName: “Edited task name 1”, TaskDescription: “Edited task description 1”, TaskDate: tomorrow | | |
| **Test Steps:** | 1. Run the application 2. Double click on the first task in the list 3. Click the edit icon 4. Use the test data to update values in the *editTask* dialog fields 5. Click S*ave* 6. Click the calendar icon (top right corner) | | |
| **Post Conditions:** | The CalendarView table is displayed. | | |
| **Expected Result:** | The taskis visible in the CalendarView table and its values match the updated test data | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-IT.29 | | |
| **Test Case Description:** | Confirm tasks edited in the CalendarView table are also edited in the ListView table | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document  Edit Task Data:  TaskName: “Edited task name 2”, TaskDescription: “Edited task description 2”, TaskDate: today | | |
| **Test Steps:** | 1. Run the application 2. Click the calendar icon (top right corner) 3. Navigate to the calendar week showing Task number 2 4. Double click on task number 2 5. Click the edit icon 6. Use the test data to update values in the *editTask* dialog fields 7. Click S*ave* 8. Click the list icon (top right corner beside calendar icon) | | |
| **Post Conditions:** | The ListView table is displayed. | | |
| **Expected Result:** | The taskis visible in the ListView table and its values match the updated test data | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-IT.30 | | |
| **Test Case Description:** | Confirm tasks deleted from the ListView table are also deleted from the CalendarView table | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click on the first task in the list (task 1) 3. Click *Delete Selected* 4. Click the calendar icon (top right corner) 5. If required, change the selectedWeek date range until it displays the week that task 1’s date should show up in | | |
| **Post Conditions:** | The CalendarView table is displayed. | | |
| **Expected Result:** | Task1 has been deleted from the CalendarView table | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-IT.31 | | |
| **Test Case Description:** | Confirm tasks deleted from the CalendarView table are also deleted from the ListView table | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Run the application 2. Click on the calendar icon 3. If required, change the selectedWeek date range until it displays the week that task 1 shows up in 4. Click on Task 1 5. Click *Delete Selected* 6. Click the list icon (top right corner beside calendar icon) | | |
| **Post Conditions:** | The ListView table is displayed. | | |
| **Expected Result:** | Task1 has been deleted from the ListView table | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

**Unit Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.01 | | |
| **Test Case Description:** | Test method Controller.Run() | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | The MainForm is created with a ListView Form inside it. The ListView JTable model contains all of the tasks in taskList. | | |
| **Expected Result:** | A database connection is initiated. All records from database are added to the *taskList* as *Task* objects.  The MainForm is created and displayed on screen with a listView InternalForm displayed inside it. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.02 | | |
| **Test Case Description:** | Test method Controller.displayErrorMessage() | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | taskNumber = null , taskName = “errorTask”, taskDescription = “desc”, taskDate = today, taskStatus = false | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Programmatically add a task to *taskList* with a null value for *taskNumber* 3. Run the application in debug mode | | |
| **Post Conditions:** | The *errorDialog* dialog box is displayed on screen. | | |
| **Expected Result:** | *ErrorDialog* shows the message “Invalid task number retrieved from database…”. | **Actual Result:** | **ERROR:**  Dialog is showing “message”  **FIXED:**  Removed speech marks from “message” so it now displays the variable *message* instead*.* |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.03 | | |
| **Test Case Description:** | Test method Controller.getAllTasks() | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | *taskList* containsone task for each record in the database | | |
| **Expected Result:** | Each database row is retrieved as a string array and converted into a Task Object. *taskList* stores all of the Task the objects. | **Actual Result:** | **ERROR:**  s*tatus* value of “1” or “0” does not parse from string to Boolean.  **FIXED:**  *Status* is now stored in database as “true” or “false”. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.04 | | |
| **Test Case Description:** | Test method Controller.populateData(**listView**) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | The *listView* JTable is populated with data from *taskList*, and is sorted in ascending order based on the date column | | |
| **Expected Result:** | The listView JTable is assigned a table model that updates the view when it detects changes to table model data. The table model is updated to contain a table row for each Task in *taskList*. The table model sorts the data by the date column in ascending order. | **Actual Result:** | **ERROR:**  View is not being updated when table model data is added.  **FIRST ERROR FIXED:**  Added model listener to notify view when table model data is updated.  **ERROR:**  Dates are being displayed in ISO format (yyyy-MM-dd) and data is not being displayed in ascending date order  **FIXED:**  Added renderTableDateColumn() method and sortTable() method |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.05 | | |
| **Test Case Description:** | Test method Controller.populateData(**calendarView**) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Continue past the first breakpoint 4. Click the calendar icon (top right corner) | | |
| **Post Conditions:** | The *calendarView* JTable is populated with data from *taskList.* The data is arranged into days of the week (Monday – Sunday). | | |
| **Expected Result:** | The calendarView JTable is assigned a table model that updates the view when it detects changes to table model data. Tasks in taskList which have a taskDate that falls within the selected week (relevant tasks) are arranged into days of the week. The table model is updated to contain a table row for each relevant task. | **Actual Result:** | **ERROR:**  Tasks are being arranged into days of the week but all tasks are being displayed, not just those relevant to the selected week.  **FIXED:**  Wrapped dayOfWeek switch in a for loop that checks to see if the task falls inside the selected week. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.06 | | |
| **Test Case Description:** | Test method Controller.generateRowData(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | Task object is added to the table model | | |
| **Expected Result:** | The Task is parsed into an Object array. If its status is set as ‘true’ it has strikethrough formatting added to it.  The array is then added to the table model. | **Actual Result:** | **ERROR:**  Strikethrough formatting is not compatible with int and Boolean data types.  **FIXED:**  Strikethrough formatting is now only added to the taskName field. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.07 | | |
| **Test Case Description:** | Test method Controller.addTableModelListener(…) when fired by checkbox update | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the check box in the status column of the JTable at the third row from the top (Task 3) | | |
| **Post Conditions:** | The selected task is displayed in the view with strikethrough formatting on the task name. | | |
| **Expected Result:** | The selected task’s status value is updated in the taskList and database to show status as ‘true’. The taskName has HTML tags added to display strikethrough formatting. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.08 | | |
| **Test Case Description:** | Test method Controller.addTableModelListener(…) when fired by *viewTask* dialog button *‘Mark incomplete’* | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click in the listView JTable on the third row from the top (Task 3) 4. In the *ViewTask* dialog box, click the *‘Mark Incomplete’* button | | |
| **Post Conditions:** | The selected task is displayed in the view with no strikethrough formatting on the task name. | | |
| **Expected Result:** | The selected task is updated in the taskList and database to show status as ‘false. The taskName has HTML tags removed to display no strikethrough formatting. | **Actual Result:** | **ERROR:**  Strikethrough is being removed from a different row in the table.  **FIXED:**  The correct row index value is now targeted in the table model |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.09 | | |
| **Test Case Description:** | Test method Controller.getTaskIndex(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click in the listView JTable on the first row from the top (Task 1) 4. In the *ViewTask* dialog box, click the *‘Mark Complete’* button | | |
| **Post Conditions:** | The index value of the selected task in taskList is returned | | |
| **Expected Result:** | Checks the taskList to see if a task exists with a taskNumber matching the given argument and if so returns the taskList index value of that task. If no task is found it should return -1. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.10 | | |
| **Test Case Description:** | Test method Controller.addInitialStrikethrough(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | The string returned contains the taskName argument provided with HTML tags wrapped around it | | |
| **Expected Result:** | HTML tags are added to the given string argument. The HTML tags add strikethrough formatting to the string when it is displayed on screen to the user. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.11 | | |
| **Test Case Description:** | Test method Controller.addOrRemoveStrikethrough(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click in the listView JTable on the first row from the top (Task 1) 4. In the *ViewTask* dialog box, click the *‘Mark Complete’* button 5. Repeat steps 3 and 4 | | |
| **Post Conditions:** | The taskName string for the selected task is displayed without strikethrough formatting. | | |
| **Expected Result:** | At step 4, HTML tags are concatenated to the given string argument which adds strikethrough formatting to the string when it is displayed on screen. After step 5, the strikethrough formatting tags have been removed from the selected taskName. | **Actual Result:** | **ERROR:**  Strikethrough is being removed from a different row in the table.  **FIXED:**  The correct row index value is now targeted in the table model |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.07-UT.12 | | |
| **Test Case Description:** | Test method Controller.renderTableDateColumn(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | The listView table displays all date values in the format dd-MM-yyyy. | | |
| **Expected Result:** | Each row in the table model has the value in its date column changed from ISO format (yyyy-MM-dd) to NZ locale format (dd-MM-yyyy). | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.07-UT.13 | | |
| **Test Case Description:** | Test method Controller.sortTable(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | The listView table displays its data in ascending order based on which task in the list is due first | | |
| **Expected Result:** | The table model has sortKeys assigned to it based on the values in the date column so that it displays the earliest date at the top of the list and the latest date at the bottom | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.07-UT.14 | | |
| **Test Case Description:** | Test method Controller.formatDate(**LocalDate**) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | Returns a LocalDate object in the format dd-MM-YYYY | | |
| **Expected Result:** | Regardless of how the provided LocalDate argument is formatted, this method should re-format it into dd-MM-yyyy and return that value as a LocalDate object | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.07-UT.15 | | |
| **Test Case Description:** | Test method Controller.formatDate(**String**) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the task at the fourth row from the top (task 4) in the listView JTable. | | |
| **Post Conditions:** | Returns a String in the format dd-MM-YYYY | | |
| **Expected Result:** | The given argument is formatted from yyyy-MM-dd to dd-MM-yyyy. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.08-UT.16 | | |
| **Test Case Description:** | Test method Controller.addTableMouseClickListener(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the task at the fourth row from the top (task 4) in the listView JTable. | | |
| **Post Conditions:** | *ViewTask* dialog is displayed on screen | | |
| **Expected Result:** | The task number of the selected task is pulled from the table model and compared against taskNumber of tasks in taskList. Where the matching task is found the other attribute values of that task are obtained and stored in an ArrayList which is then passed to a dialog box to be displayed in the dialog fields. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.17 | | |
| **Test Case Description:** | Test method Controller.displayListView(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | A new *ListView* InternalForm is displayed on screen, any other InternalForms are closed | | |
| **Expected Result:** | A new ListView form is created and displayed on screen. Any other forms are disposed of. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.18 | | |
| **Test Case Description:** | Test method Controller.displayCalendarView(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon (top right corner) | | |
| **Post Conditions:** | A new *CalendarView* InternalForm is displayed on screen, any other InternalForms are closed | | |
| **Expected Result:** | A new CalendarView form is created and displayed on screen. Any other forms are disposed of. | **Actual Result:** | **ERROR:**  A new listView form is being created instead of a calendarView form.  **FIXED:**  Updated displayCalendarView method so it creates a CalendarView form rather than a ListView form |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.19 | | |
| **Test Case Description:** | Test method Controller.addTask(…) with valid input | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “Newly Added Task”, TaskDescription: “This is a new task”, TaskDate: yesterday’s date | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left corner) 4. Add test data in the appropriate fields 5. Click *Save* button | | |
| **Post Conditions:** | The new task is added to the taskList, the database and the table model. | | |
| **Expected Result:** | User input is validated before the task is added to the taskList, database and table model. The view is updated to show the new task | **Actual Result:** | **ERROR:**  View not being updated with new data  **FIXED:**  Added call to repopulate table data to ensure it is updated |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.20 | | |
| **Test Case Description:** | Test method Controller.addTask(…) with invalid input | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “”, TaskDescription: “”, TaskDate: “01/07/22” | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left corner) 4. Add test data in the appropriate fields 5. Click *Save* button | | |
| **Post Conditions:** | No new task is added to the taskList, the database or the table model. The user is shown a message explaining what went wrong | | |
| **Expected Result:** | User input is checked and the taskName and date are found to be invalid. The text fields in taskName and taskDate are bordered in red and updated to show the user why their input was invalid | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.21 | | |
| **Test Case Description:** | Test method Controller.addTask(…) with no input | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “”, TaskDescription: “”, TaskDate: “” | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left corner) 4. Add test data in the appropriate fields 5. Click *Save* button | | |
| **Post Conditions:** | No new task is added to the taskList, the database or the table model. The user is shown a message explaining what went wrong | | |
| **Expected Result:** | User input is checked and the taskName and date are found to be invalid. The text fields in taskName and taskDate are bordered in red and updated to show the user they are required fields | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.22 | | |
| **Test Case Description:** | Test method Controller.editTask(…) with valid input | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “200”, TaskDescription: “An edited task”, TaskDate: today | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click the second taskfrom the top(task 2) in the listView table. 4. Click the *edit* icon (top right corner) 5. Update the fields as per the test data 6. Click save | | |
| **Post Conditions:** | The method returns true. The second task from the top of the listView table is displayed with the test data as its values | | |
| **Expected Result:** | User input is checked and validated. The text fields in taskName, taskDescription and taskDate are updated to show the user’s changes | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.23 | | |
| **Test Case Description:** | Test method Controller.editTask(…) with invalid input | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “”, TaskDescription: “An edited task”, TaskDate: “01/05/2022” | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click the second taskfrom the top(task 2) in the listView table. 4. Click the *edit* icon (top right corner) 5. Update the fields as per the test data 6. Click save | | |
| **Post Conditions:** | The method returns false. No changes are made to the table data | | |
| **Expected Result:** | User input is checked and invalidated. The taskName and taskDate fields have a red border and display messages to the user explaining why they are invalid | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.24 | | |
| **Test Case Description:** | Test method Controller.editTask(…) with no input | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “”, TaskDescription: “”, TaskDate: “” | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click the second taskfrom the top(task 2) in the listView table. 4. Click the *edit* icon (top right corner) 5. Update the fields as per the test data 6. Click save | | |
| **Post Conditions:** | The method returns false. No changes are made to the table data | | |
| **Expected Result:** | User input is checked and invalidated. The taskName and taskDate fields have a red border and display messages to the user explaining that they are required fields | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.25 | | |
| **Test Case Description:** | Test method Controller.validateNameAndDate(…) with valid data | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “200”, TaskDescription: “An edited task”, TaskDate: today | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click the second taskfrom the top(task 2) in the listView table. 4. Click the *edit* icon (top right corner) 5. Update the fields as per the test data 6. Click save | | |
| **Post Conditions:** | The method returns true. | | |
| **Expected Result:** | Name and date fields are checked and found to be valid so the method returns true | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.26 | | |
| **Test Case Description:** | Test method Controller.validateNameAndDate(…) with invalid data | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “”, TaskDescription: “An edited task”, TaskDate: 06/06/2020. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click the second taskfrom the top(task 2) in the listView table. 4. Click the *edit* icon (top right corner) 5. Update the fields as per the test data 6. Click save | | |
| **Post Conditions:** | The method returns false. | | |
| **Expected Result:** | TaskName and date fields are checked and found to be invalid. The taskName and task date fields in editTask dialog box have a red border and explain to user why input was invalid. Method returns false. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.27 | | |
| **Test Case Description:** | Test method Controller.validateNameAndDate(…) with no data | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “”, TaskDescription: “”, TaskDate: null | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click the second taskfrom the top(task 2) in the listView table. 4. Click the *edit* icon (top right corner) 5. Update the fields as per the test data 6. Click save | | |
| **Post Conditions:** | The method returns false. | | |
| **Expected Result:** | TaskName and date fields are checked and found to be invalid. The taskName and task date fields in editTask dialog box have a red border and explain to user they are required fields. Method returns false. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.28 | | |
| **Test Case Description:** | Test method Controller.convertNZDateToISO (…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “200”, TaskDescription: “An edited task”, TaskDate: today | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click the second taskfrom the top(task 2) in the listView table. 4. Click the *edit* icon (top right corner) 5. Update the fields as per the test data 6. Click save | | |
| **Post Conditions:** | The method returns a string containing a date value formatted as dd-MM-yyyy. | | |
| **Expected Result:** | The given string argument is parsed into a LocalDate ISO format then re-foramtted into New Zealand locale (dd-MM-yyyy) before being re-parsed into a string | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-UT.29 | | |
| **Test Case Description:** | Test method Controller.deleteTask (…) from ListView | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click the fifth taskfrom the top(task 5) in the listView table. 4. Click the *Delete Selected* button (bottom right corner) | | |
| **Post Conditions:** | The method returns true and the task is no longer displayed on the listView Jtable. | | |
| **Expected Result:** | The selected task is removed from the database, the taskList and the table model. The table model data is refreshed. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-UT.30 | | |
| **Test Case Description:** | Test method Controller.deleteTask (…) from CalendarView | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon (top right corner) 4. Double click the second taskfrom the topin the calendarView table. 5. Click the *Delete Selected* button (bottom right corner) | | |
| **Post Conditions:** | The method returns true and the task is no longer displayed on the calendarView Jtable. | | |
| **Expected Result:** | The selected task is removed from the database, the taskList and the table model. The table model data is refreshed. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.31 | | |
| **Test Case Description:** | Test method Controller.getSelectedTask () | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click the second taskfrom the topin the listView table. 4. Click the ‘*Mark as completed’* button | | |
| **Post Conditions:** | The method returns a string array containing the values of the task in the selected table row | | |
| **Expected Result:** | The task in the selected row is found in the task list and it’s attributes are added to a string array which is returned. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.32 | | |
| **Test Case Description:** | Test method Controller.setSelectedWeek (…) to current week | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon (top right corner) | | |
| **Post Conditions:** | The selected week date range starts on Monday of the current week and ends on Sunday of the current week. | | |
| **Expected Result:** | The first date element in the selectedWeek array is calculated with a date value equivalent to Monday of the current week. The second array element is calculated with a date value equivalent to Sunday of the current week. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.33 | | |
| **Test Case Description:** | Test method Controller.setSelectedWeek (…) to previous week | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon (top right corner) 4. Click the previous arrow (top right corner) to change the selected date range to the previous week | | |
| **Post Conditions:** | The selected week date range is changed to display one week into the past, starting last Monday and ending last Sunday. | | |
| **Expected Result:** | The first date element in the *selectedWeek* array is calculated with a date value equivalent to Monday last week (7 days less than the original *selectedWeek*’s Monday date). The second array element is calculated with a date value equivalent to Sunday of last week. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.34 | | |
| **Test Case Description:** | Test method Controller.setSelectedWeek (…) to next week | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon (top right corner) 4. Click the next arrow (top right corner) to change the selected date range to the following week | | |
| **Post Conditions:** | The selected week date range is changed to display one week into the future, starting next Monday and ending next Sunday. | | |
| **Expected Result:** | The first date element in the *selectedWeek* array is calculated with a date value equivalent to Monday next week (7 days more than the original *selectedWeek*’s Monday date). The second array element is calculated with a date value equivalent to Sunday of next week. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.35 | | |
| **Test Case Description:** | Test method Controller.getSelectedWeek () | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon (top right corner) | | |
| **Post Conditions:** | A string is returned containing the dates for Monday and Sunday of the current week | | |
| **Expected Result:** | A string is returned in the format (dd-MM-YYYY to dd-MM-yyyy). The dates produced fall on the Monday and Sunday of the current week. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.36 | | |
| **Test Case Description:** | Test method Controller.parseStringsToTask(…) with valid input data | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “Newly Added Task”, TaskDescription: “This is a new task”, TaskDate: yesterday’s date | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *AddTask* button 4. Add a task as per the test data above | | |
| **Post Conditions:** | A Task object is returned. | | |
| **Expected Result:** | The string arguments are parsed into the data types needed to construct a Task object. The new Task object is returned. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.37 | | |
| **Test Case Description:** | Test method Controller.parseStringsToTask(…) with invalid input data | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | TaskName: “”, TaskDescription: “This is a new task”, TaskDate: “” | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *AddTask* button 4. Add a task as per the test data above | | |
| **Post Conditions:** | A Task object is returned with default values. | | |
| **Expected Result:** | The returned Task object should have a taskNumber value of -1. | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.38 | | |
| **Test Case Description:** | Test method DbConnection.initDatabase() with database already created | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | Database connection is established successfully | | |
| **Expected Result:** | The variable *DB* is instantiated as a valid database connection. No SQLExceptions are thrown | **Actual Result:** | As per expected result. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.39 | | |
| **Test Case Description:** | Test method DbConnection.initDatabase() when database does not exist | | |
| **Preconditions:** | Database file does not exist in the application file directory. This may require removal of *ToDo.db* from the application root folder | | |
| **Test Data:** | Nil. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Open console in application directory 4. Open database file with command: sqlite3 ToDo.db 5. View structure of Task table with command: PRAGMA table\_info(task); | | |
| **Post Conditions:** | A new database file *ToDo.db* is created in the application directory. A connection is established to the new database successfully. | | |
| **Expected Result:** | The variable *DB* is instantiated as a valid database connection. No SQLExceptions are thrown. Table *Task* is added to the new database with columns as follows: (int) taskNumber, (text) taskName, (text) taskDescription, (text) dueDate, (text) status. | **Actual Result:** | **ERROR:**  New database is created however *Task* table is not added to it.  **FIXED:**  Try/catch block now includes an executeUpdate statement to add *Task* table if it doesn’t already exist. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.40 | | |
| **Test Case Description:** | Test method DbConnection.getLargestTaskNumber() where task records exist in the database | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document.  New Task:  TaskName: “Task name 100”, taskDescription: “Task description 100”, taskDate: today’s date | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click *Add Task* button (bottom left corner) 4. Complete fields using ‘New Task’ test data 5. Click S*ave* button (bottom left of dialog) | | |
| **Post Conditions:** | Method returns a string containing the value of the largest integer in the taskNumber colum of database table *Task.* | | |
| **Expected Result:** | SQLite query is executed without any exceptions.  The string returned reflects the largest integer record stored in the taskNumber column of database table *Task.* | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.41 | | |
| **Test Case Description:** | Test method DbConnection.getLargestTaskNumber() where task records do not exist in the database | | |
| **Preconditions:** | Database has been created using DDL script  All task records have been deleted from the database | | |
| **Test Data:** | New Task:  TaskName: “Task name 100”, taskDescription: “Task description 100”, taskDate: today’s date | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click *Add Task* button (bottom left corner) 4. Complete fields using ‘New Task’ test data 5. Click S*ave* button (bottom left of dialog) | | |
| **Post Conditions:** | Method returns a string with a null value*.* | | |
| **Expected Result:** | SQLite query is executed without any exceptions.  There are no integers stored in the *task* table’s taskNumber column, so a null value is returned | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.42 | | |
| **Test Case Description:** | Test method DbConnection.createTask(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | New Task:  TaskName: “Task name 100”, taskDescription: “Task description 100”, taskDate: today’s date | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click *Add Task* button (bottom left corner) 4. Complete fields using ‘New Task’ test data 5. Click S*ave* button (bottom left of dialog) 6. Open database in console with command: sqlite3 ToDo.db 7. Query database in console: SELECT \* FROM Task; | | |
| **Post Conditions:** | The new task is stored as the last record in the database. Method returns true*.* | | |
| **Expected Result:** | SQLite query is executed without any exceptions.  The new task is saved as the last record in the database and its taskNumber is one integer larger than the preceding task. All of its fields contain the correct data as per the Test Data. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.43 | | |
| **Test Case Description:** | Test method DbConnection.updateTask(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document.  Updated Task Values:  TaskName: “Task name 100”, taskDescription: “Task description 100”, taskDate: today’s date | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the first task in the list (Task 1) 4. Click the edit icon (top right of dialog) 5. Change the field values to reflect the updated task values in test data 6. Click S*ave* button (bottom left of dialog) 7. Open database in console with command: sqlite3 ToDo.db 8. Query database in console: SELECT \* FROM Task; | | |
| **Post Conditions:** | The task record in the database with taskNumber 1 has the updated task values saved. Method returns true*.* | | |
| **Expected Result:** | SQLite query is executed without any exceptions.  The database task record with taskNumber 1 has the values of its fields updated to be consistent with the updated task test data values. | **Actual Result:** | **ERROR:**  SQL Exception – query string formatting is invalid  **FIXED:**  Reformatted SQL query string to contain escaped quotation marks and commas in thecorrectplaces. Typos in table column names are fixed |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.44 | | |
| **Test Case Description:** | Test method DbConnection.updateTaskStatus(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click on the checkbox in the third table row from the top (Task 3) | | |
| **Post Conditions:** | Task number 3 has a tick in its checkbox and strikethrough formatting added to its taskName value. | | |
| **Expected Result:** | SQLite query is executed without any exceptions.  The database record with taskNumber 3 has its status value updated to be ‘true’ instead of false | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.08-UT.45 | | |
| **Test Case Description:** | Test method DbConnection.readTask(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the task in the third table row from the top (Task 3) | | |
| **Post Conditions:** | *ViewTask* dialog is displayed. It’s fields are populated with data from the database record relating to task number 3. | | |
| **Expected Result:** | SQLite query is executed without any exceptions.  The method returns an arrayList of strings. The value of elements in the arrayList correspond to the database column values for the record with taskNumber 3 | **Actual Result:** | **ERROR:**  Wrong task values are being returned.  **FIXED:**  Added ‘WHERE taskNumber == …’  to the SQL query. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.46 | | |
| **Test Case Description:** | Test method DbConnection.readAllTasks() | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode | | |
| **Post Conditions:** | *ListView* table contains a row corresponding to each row of data in the database. | | |
| **Expected Result:** | SQLite query is executed without any exceptions.  The method returns an ArrayList. Each element in the list is another ArrayList whose elements contain the data from a single table row in the database. There is a list for every row of data in the database. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-UT.47 | | |
| **Test Case Description:** | Test method DbConnection.deleteTask(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click on the fourth task from the top in the *listView* table 4. Click *Delete Selected* button (bottom right corner) 5. Open database in console with command: sqlite3 ToDo.db 6. Query the database in console to get all records: SELECT \* FROM Task; | | |
| **Post Conditions:** | Task number 4 is deleted from *ListView* table and the corresponding record no longer exists in the database. | | |
| **Expected Result:** | SQLite query is executed without any exceptions.  The database record with taskNumber 4 no longer exists and the method returns true. | **Actual Result:** | **ERROR:**  SQL Exception encountered – query string is not valid  **FIXED:**  Reformatted sql query string with escaped quotation marks in the right places |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-UT.48 | | |
| **Test Case Description:** | Test method DbConnection.insertUpdateOrDelete(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click on the second task from the top in the *listView* table 4. Click *Delete Selected* button (bottom right corner) 5. Open database in console with command: sqlite3 ToDo.db 6. Query the database in console to get all records: SELECT \* FROM Task; | | |
| **Post Conditions:** | Task number 2 is deleted from *ListView* table and the corresponding record no longer exists in the database. | | |
| **Expected Result:** | SQLite query is executed without any exceptions.  The database record with taskNumber 2 no longer exists and the method returns true. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.49 | | |
| **Test Case Description:** | Test method MainForm.jButton\_listViewMouseClicked(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click on the calendar icon (top right corner) 4. Click on the list icon (to the immediate left of calendar icon) | | |
| **Post Conditions:** | ListView form is displayed on screen. | | |
| **Expected Result:** | The currently active form is closed. A new listView form is created and displayed on screen within the MainForm’s desktop pane. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.50 | | |
| **Test Case Description:** | Test method MainForm.jButton\_calendarViewMouseClicked(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click on the calendar icon (top right corner) | | |
| **Post Conditions:** | CalendarView form is displayed on screen. | | |
| **Expected Result:** | The currently active form is closed. A new CalendarView form is created and displayed on screen within the MainForm’s desktop pane. | **Actual Result:** | **ERROR:**  A new listView form is being created instead of a calendarView form.  **FIXED:**  Updated Controller’s displayCalendarView method so it creates a CalendarView form rather than a ListView form |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.51 | | |
| **Test Case Description:** | Test method ListViewForm.displayErrorMessage(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Open the database in console with command: sqlite3 ToDo.db 3. Use the console to update the date value of task1 with command:   UPDATE Task SET dueDate = “10-06-2022” WHERE taskNumber = 1;   1. Run the application in debug mode | | |
| **Post Conditions:** | ListView form is displayed on screen with an errorDialog box displayed on top. | | |
| **Expected Result:** | The error dialog is displayed in the centre of the listView form with an error message explaining that the database contains an invalid value. | **Actual Result:** | **ERROR:**  *ErrorDialog* is not being displayed because the listView form doesn’t exist yet.  **FIXED:**  Updated Controller’s *displayErrorMessage()* method so it now creates a ListViewForm before trying to create the errorDialog |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.52 | | |
| **Test Case Description:** | Test method ListViewForm.displayTask(…) when a completed task is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the fifth task from the top in the listView table (task number 5) | | |
| **Post Conditions:** | *ViewTask* dialog is displayed on top in the centre of the listView table. | | |
| **Expected Result:** | The *viewTask* dialog is displayed in the centre of the listView form. The dialog contains a button to update the selected task status with button text set to: ‘Mark as incomplete’ | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.53 | | |
| **Test Case Description:** | Test method ListViewForm.displayTask(…) when an incomplete task is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the fourth task from the top in the listView table (task number 4) | | |
| **Post Conditions:** | *ViewTask* dialog is displayed on top in the centre of the listView table. | | |
| **Expected Result:** | The error dialog is displayed in the centre of the listView form. The dialog contains a button to update the selected task status with button text set to: ‘Mark as completed’ | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.54 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_addTaskSaveActionPerformed(…) when valid input is entered | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | New Task:  TaskName: “Task name 100”, taskDescription: “Task description 100”, taskDate: today’s date | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Enter new task details into the dialog fields as per test data 5. Click *Save* button | | |
| **Post Conditions:** | The new task is displayed in the listView table. | | |
| **Expected Result:** | The *AddTask* dialog is closed and the new task is visible in the listView table with its values as per the test data | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.55 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_addTaskSaveActionPerformed(…) when invalid input is entered | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | New Task:  TaskName: “”, taskDescription: “Task description 100”, taskDate: “” | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Enter new task details into the dialog fields as per test data (leave task name and task date fields blank) 5. Click *Save* button | | |
| **Post Conditions:** | *AddTask* dialog is displayed on top in the centre of the listView table. | | |
| **Expected Result:** | The *AddTask* dialog remains visible and an error message is displayed at the bottom of the dialog box explaining what to do next | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.56 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskNameFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the “Task Name” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an empty Task Name field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Name’ field text changes from “Enter task name” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.57 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskNameFocusGained(…) after trying to save a new task with an empty task name | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the S*ave* button 5. Click the “Task Name” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an empty Task Name field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Name’ field text changes from “\*Task name is required” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.58 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskNameFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the “Task Name” field 5. Click the “Task Details” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with hint text showing in Task Name field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Name’ field text changes from an empty string to having the value “Enter task name” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.59 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskDetailsFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the “Task Details” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an empty Task Details field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Details’ field text changes from “Add task details” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.60 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskDetailsFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the “Task Details” field 5. Click the “Task Name” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with hint text showing in Task Details field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Details’ field text changes from an empty string to having the value “Add task details” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.61 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskDateFocusGained(…) when date field has not had focus before | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the Task Date field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an empty string value in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from “dd-mm-yyyy” to an empty string. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.62 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskDateFocusGained(…) after save button has been clicked with blank date field | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the Task Date field 5. Click the *Save* button 6. Click the Task Date field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an error message in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from “\*Date required” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.62.2 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskDateFocusGained(…) when date field has previously had invalid data | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the Task Date field 5. Click the *Save* button 6. Click the Task Date field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with a blank string in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from “\*Date required” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.63 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskDateFocusGained(…) after save button has been clicked with invalid data in date field | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | New Task:  TaskName: “”, taskDescription: “Task description 100”, taskDate: “10/06/22” | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Enter the test data into the respective dialog fields 5. Click the *Save* button 6. Click the Task Date field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with a blank string in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from “\*Must be dd-mm-yyyy” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.64 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_addTaskDateFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the Task Date field 5. Click the Task Name field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with hint text showing in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from an empty string to “dd-mm-yyyy” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.65 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_addTaskCancelActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button (bottom left) 4. Click the *Cancel* button (bottom right of dialog) | | |
| **Post Conditions:** | The ListViewtable is displayed on screen | | |
| **Expected Result:** | The *AddTask* dialog is closed and the listView table is displayed with no changes made to the table model, database or taskList | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.66 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_viewTaskMarkCompletedActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the ‘*Mark as completed’* button (bottom left of dialog) | | |
| **Post Conditions:** | The ListViewtable is displayed on screen and the third task from the top is marked as complete. | | |
| **Expected Result:** | The *AddTask* dialog is closed and the listView table is displayed. The third task from the top of the list (task 3) has a checked checkbox and strikethrough formatting on its taskName field | **Actual Result:** | **ERROR:**  The wrong task is being updated in the task list.  **FIXED:**  The correct index value is now found before the task status is updated |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.67 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_editTaskActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the *Edit* icon | | |
| **Post Conditions:** | The *EditTask* dialog is displayed on top of the listView. Its fields show data from the selected task | | |
| **Expected Result:** | The *ViewTask* dialog is closed and the *EditTask* dialog is displayed in the centre of the listView table. The *EditTask* dialog fields are populated with the same data that was previously displayed in the *ViewTask* dialog | **Actual Result:** | **ERROR:**  HTML tags are being displayed in the TaskName field when strikethrough formatting has previously been added to the task.  **FIXED:**  HTML tags are now stripped by Controller’s addOrRemoveStrikethrough() method. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-UT.68 | | |
| **Test Case Description:** | Test method ListViewForm.jTable\_listViewMouseClicked(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click on the third task from the top of the listView table (task 3) | | |
| **Post Conditions:** | The *Delete Selected* button is displayed below the listView table | | |
| **Expected Result:** | The *Delete Selected* button’s visibility is set to be ‘true’, causing it to be displayed in the bottom right corner of the listView form, below the listView table | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.69 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_editTaskSaveActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document  Updated Task:  TaskName: “Task name 300”, TaskDetails: “task description 300”, TaskDate: today | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the edit icon 5. Change the fields as per the ‘Updated Task’ test data 6. Click *Save* button | | |
| **Post Conditions:** | The listView table is displayed on screen. The values in task number 3 on the listView table reflect the updates from the test data | | |
| **Expected Result:** | The database record for task number 3 is updated to reflect the new test data values.  The *editTask* dialog is closed and the ListView table is displayed. The edits are displayed in the listView table entry for task number 3 | **Actual Result:** | **ERROR:**  The values in ListView table are not being updated  **FIXED:**  The table data is now refreshed after each update |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.70 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_editTaskNameFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the edit icon 5. Click in the Task Name field | | |
| **Post Conditions:** | The Task Name field in *editTask* dialog displays an empty string | | |
| **Expected Result:** | The Task Name field text changes from “Enter task name” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.71 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_editTaskNameFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the edit icon 5. Click in the Task Name field 6. Click in the Task Description field | | |
| **Post Conditions:** | The Task Name field in *editTask* dialog displays hint text | | |
| **Expected Result:** | The Task Name field text changes from an empty string to “Enter task name” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.72 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_editTaskDateFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the edit icon 5. Click in the Task Date field | | |
| **Post Conditions:** | Hint text disappears from the Task Date field in *editTask* dialog | | |
| **Expected Result:** | The Task Date field text changes from “dd-mm-yyyy” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.73 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_editTaskDateFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the edit icon 5. Click in the Task Date field 6. Click in the Task Name field | | |
| **Post Conditions:** | Hint text is displayed in the Task Date field in *editTask* dialog | | |
| **Expected Result:** | The Task Date field text changes from an empty string to “dd-mm-yyyy” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.74 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_editTaskDetailsFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the edit icon 5. Click in the Task Details field | | |
| **Post Conditions:** | Hint text disappears from the Task Details field in *editTask* dialog | | |
| **Expected Result:** | The Task Details field text changes from “Add task details” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.75 | | |
| **Test Case Description:** | Test method ListViewForm.jTextField\_editTaskDetailsFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the edit icon 5. Click in the Task Details field 6. Click in the Task Name field | | |
| **Post Conditions:** | Hint text is displayed in the Task Details field in *editTask* dialog | | |
| **Expected Result:** | The Task Details field text changes from an empty string to “Add task details” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.76 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_editTaskCancelActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Double click on the third task from the top of the listView table (task 3) 4. Click the edit icon 5. Click the *Cancel* button | | |
| **Post Conditions:** | *editTask* dialog is closed and ListView is displayed | | |
| **Expected Result:** | The *EditTask* dialog is closed without any changes being made to the table model, the taskList or the database. The ListView table is displayed on screen | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-UT.77 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_deleteSelectedTaskActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click on the second task from the top of the listView table (task 2) 4. Click the *Delete Selected* button | | |
| **Post Conditions:** | Task number 2 is no longer displayed in the listView table | | |
| **Expected Result:** | The database record with taskNumber 2 is deleted.  In the taskList, the task with taskNumber 2 is removed.  The table model is updated to reflect the deleted table row | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.78 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_addTaskActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the *Add Task* button | | |
| **Post Conditions:** | *AddTask* dialog is displayed on top of the listView table | | |
| **Expected Result:** | All fields in the *AddTask* dialog are reset to their default values and the dialog is displayed in the centre of the listView table. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.01-UT.79 | | |
| **Test Case Description:** | Test method ListViewForm.jButton\_errorDialogContinueActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Open the database in console using command: sqlite3 ToDo.db   Update the date value of task Number 1 using command: UPDATE Task SET dueDate = “10-06-2022” WHERE taskNumber = 1;   1. Run the application in debug mode 2. In the *errorDialog* click the *Continue* button | | |
| **Post Conditions:** | *ErrorDialog* is closed and the listView table is displayed. | | |
| **Expected Result:** | The *ErrorDialog* dialog box is disposed of and a ListView form is displayed on screen. The listView table model should include a row for each database record except task number 1. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.08-UT.80 | | |
| **Test Case Description:** | Test method CalendarViewForm.displayTask(…) when a completed task is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the second task from the top in the CalendarView table (task number 2) | | |
| **Post Conditions:** | *ViewTask* dialog is displayed on top in the centre of the CalendarView table. | | |
| **Expected Result:** | The *ViewTask* dialog is displayed in the centre of the CalendarView form. The dialog contains a button to update the selected task status with button text set to: ‘Mark as incomplete’ | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.08-UT.81 | | |
| **Test Case Description:** | Test method CalendarViewForm.displayTask(…) when an incomplete task is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the first task from the top in the CalendarView table (task number 1) | | |
| **Post Conditions:** | *ViewTask* dialog is displayed on top in the centre of the CalendarView table. | | |
| **Expected Result:** | The *ViewTask* dialog is displayed in the centre of the CalendarView form. The dialog contains a button to update the selected task status with button text set to: ‘Mark as completed’ | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.82 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_addTaskSaveActionPerformed(…) when valid input is entered | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | New Task:  TaskName: “Task name 100”, taskDescription: “Task description 100”, taskDate: today’s date | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Enter new task details into the dialog fields as per test data 6. Click *Save* button | | |
| **Post Conditions:** | The new task is displayed in the CalendarView table. | | |
| **Expected Result:** | The *AddTask* dialog is closed and the new task is visible in the CalendarView table with its values as per the test data | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.83 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_addTaskSaveActionPerformed(…) when invalid input is entered | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | New Task:  TaskName: “”, taskDescription: “Task description 100”, taskDate: “” | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Enter new task details into the dialog fields as per test data (leave task name and task date fields blank) 6. Click *Save* button | | |
| **Post Conditions:** | *AddTask* dialog is displayed on top in the centre of the CalendarView table. | | |
| **Expected Result:** | The *AddTask* dialog remains visible and an error message is displayed at the bottom of the dialog box explaining what to do next | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.84 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskNameFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the “Task Name” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an empty Task Name field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Name’ field text changes from “Enter task name” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.85 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskNameFocusGained(…) after trying to save a new task with an empty task name | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the S*ave* button 6. Click the “Task Name” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an empty Task Name field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Name’ field text changes from “\*Task name is required” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.86 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskNameFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the “Task Name” field 6. Click the “Task Details” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with hint text showing in Task Name field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Name’ field text changes from an empty string to having the value “Enter task name” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.87 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskDetailsFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the “Task Details” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an empty Task Details field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Details’ field text changes from “Add task details” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.88 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskDetailsFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the “Task Details” field 6. Click the “Task Name” field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with hint text showing in Task Details field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Details’ field text changes from an empty string to having the value “Add task details” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.89 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskDateFocusGained(…) when date field has not had focus before | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the Task Date field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an empty string value in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from “dd-mm-yyyy” to an empty string. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.90 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskDateFocusGained(…) after save button has been clicked with blank date field | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the Task Date field 6. Click the *Save* button 7. Click the Task Date field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with an error message in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from “\*Date required” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.91 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskDateFocusGained(…) when date field has previously had invalid data | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the Task Date field 6. Click the *Save* button 7. Click the Task Date field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with a blank string in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from “\*Date required” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.92 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskDateFocusGained(…) after save button has been clicked with invalid data in date field | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | New Task:  TaskName: “”, taskDescription: “Task description 100”, taskDate: “10/06/22” | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Enter the test data into the respective dialog fields 6. Click the *Save* button 7. Click the Task Date field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with a blank string in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from “\*Must be dd-mm-yyyy” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.93 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_addTaskDateFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the Task Date field 6. Click the Task Name field | | |
| **Post Conditions:** | The *AddTask* dialog is displayed with hint text showing in the Task Date field | | |
| **Expected Result:** | The *AddTask* dialog ‘Task Date’ field text changes from an empty string to “dd-mm-yyyy” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.94 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_addTaskCancelActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | Nil. | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button (bottom left) 5. Click the *Cancel* button (bottom right of dialog) | | |
| **Post Conditions:** | The CalendarView table is displayed on screen | | |
| **Expected Result:** | The *AddTask* dialog is closed and the CalendarView table is displayed with no changes made to the table model, database or taskList | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.09-UT.95 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_viewTaskMarkCompletedActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the CalendarView table (task 3) 5. Click the ‘*Mark as completed’* button (bottom left of dialog) | | |
| **Post Conditions:** | The CalendarView table is displayed on screen and the third task from the top is marked as complete. | | |
| **Expected Result:** | The *AddTask* dialog is closed and the CalendarView table is displayed. The third task from the top of the list (task 3) has a checked checkbox and strikethrough formatting on its taskName field | **Actual Result:** | **ERROR:**  The wrong task is being updated in the task list.  **FIXED:**  The correct index value is now found before the task status is updated |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.96 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_editTaskActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the CalendarView table (task 3) 5. Click the *Edit* icon | | |
| **Post Conditions:** | The *EditTask* dialog is displayed on top of the CalendarView. Its fields show data from the selected task | | |
| **Expected Result:** | The *ViewTask* dialog is closed and the *EditTask* dialog is displayed in the centre of the CalendarView table. The *EditTask* dialog fields are populated with the same data that was previously displayed in the *ViewTask* dialog | **Actual Result:** | **ERROR:**  HTML tags are being displayed in the TaskName field when strikethrough formatting has previously been added to the task.  **FIXED:**  HTML tags are now stripped by Controller’s addOrRemoveStrikethrough() method. |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-UT.97 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTable\_calendarViewMouseClicked(…) when a valid task is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click on the third task from the top of the CalendarView table (task 3) | | |
| **Post Conditions:** | The *Delete Selected* button is displayed below the CalendarView table | | |
| **Expected Result:** | The *Delete Selected* button’s visibility is set to be ‘true’, causing it to be displayed in the bottom right corner of the CalendarView form, below the CalendarView table | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.05-UT.98 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTable\_calendarViewMouseClicked(…) when a day of week heading task is selected | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click on the day of week table row with title “Monday” | | |
| **Post Conditions:** | The CalendarView table item is selected | | |
| **Expected Result:** | The *Delete Selected* button’s visibility remains ‘false’ and it is NOT displayed on the CalendarView form | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.99 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_editTaskSaveActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document  Updated Task:  TaskName: “Task name 300”, TaskDetails: “task description 300”, TaskDate: today | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the CalendarView table (task 3) 5. Click the edit icon 6. Change the fields as per the ‘Updated Task’ test data 7. Click *Save* button | | |
| **Post Conditions:** | The CalendarView table is displayed on screen. The values in task number 3 on the CalendarView table reflect the updates from the test data | | |
| **Expected Result:** | The database record for task number 3 is updated to reflect the new test data values.  The *editTask* dialog is closed and the CalendarView table is displayed. The edits are displayed in the CalendarView table entry for task number 3 | **Actual Result:** | **ERROR:**  The values in CalendarView table are not being updated  **FIXED:**  The table data is now refreshed after each update |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.100 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_editTaskNameFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the CalendarView table (task 3) 5. Click the edit icon 6. Click in the Task Name field | | |
| **Post Conditions:** | The Task Name field in *editTask* dialog displays an empty string | | |
| **Expected Result:** | The Task Name field text changes from “Enter task name” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.101 | | |
| **Test Case Description:** | Test method CalendarView.jTextField\_editTaskNameFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the CalendarView table (task 3) 5. Click the edit icon 6. Click in the Task Name field 7. Click in the Task Description field | | |
| **Post Conditions:** | The Task Name field in *editTask* dialog displays hint text | | |
| **Expected Result:** | The Task Name field text changes from an empty string to “Enter task name” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.102 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_editTaskDateFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the CalendarView table (task 3) 5. Click the edit icon 6. Click in the Task Date field | | |
| **Post Conditions:** | Hint text disappears from the Task Date field in *editTask* dialog | | |
| **Expected Result:** | The Task Date field text changes from “dd-mm-yyyy” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.103 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_editTaskDateFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the CalendarView table (task 3) 5. Click the edit icon 6. Click in the Task Date field 7. Click in the Task Name field | | |
| **Post Conditions:** | Hint text is displayed in the Task Date field in *editTask* dialog | | |
| **Expected Result:** | The Task Date field text changes from an empty string to “dd-mm-yyyy” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.104 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_editTaskDetailsFocusGained(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the CalendarView table (task 3) 5. Click the edit icon 6. Click in the Task Details field | | |
| **Post Conditions:** | Hint text disappears from the Task Details field in *editTask* dialog | | |
| **Expected Result:** | The Task Details field text changes from “Add task details” to an empty string | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.105 | | |
| **Test Case Description:** | Test method CalendarViewForm.jTextField\_editTaskDetailsFocusLost(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the calendarView table (task 3) 5. Click the edit icon 6. Click in the Task Details field 7. Click in the Task Name field | | |
| **Post Conditions:** | Hint text is displayed in the Task Details field in *editTask* dialog | | |
| **Expected Result:** | The Task Details field text changes from an empty string to “Add task details” | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.03-UT.106 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_editTaskCancelActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Double click on the third task from the top of the CalendarView table (task 3) 5. Click the edit icon 6. Click the *Cancel* button | | |
| **Post Conditions:** | *editTask* dialog is closed and CalendarView is displayed | | |
| **Expected Result:** | The *EditTask* dialog is closed without any changes being made to the table model, the taskList or the database. The CalendarView table is displayed on screen | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.04-UT.107 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_deleteSelectedTaskActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click on the second task from the top of the CalendarView table (task 2) 5. Click the *Delete Selected* button | | |
| **Post Conditions:** | Task number 2 is no longer displayed in the CalendarView table | | |
| **Expected Result:** | The database record with taskNumber 2 is deleted.  In the taskList, the task with taskNumber 2 is removed.  The table model is updated to reflect the deleted table row | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.02-UT.108 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_addTaskActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *Add Task* button | | |
| **Post Conditions:** | *AddTask* dialog is displayed on top of the CalendarView table | | |
| **Expected Result:** | All fields in the *AddTask* dialog are reset to their default values and the dialog is displayed in the centre of the CalendarView table. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.109 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_calendarView\_lastWeekMouseClicked(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *previous* arrow icon (top right corner beside date range) | | |
| **Post Conditions:** | The CalendarView table date range is set to one week in the past. Tasks are only displayed if their date value fits within the new date range | | |
| **Expected Result:** | 7 days are subtracted from both the Monday and Sunday dates displayed as the date range above the CalendarView table. Only tasks with a date value that falls within the new date range are displayed in the table | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.110 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_calendarView\_nextWeekActionPerformed(…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *next* arrow icon (top right corner beside date range) | | |
| **Post Conditions:** | The CalendarView table date range is set to one week in the future. Tasks are only displayed if their date value fits within the new date range | | |
| **Expected Result:** | 7 days are added to both the Monday and Sunday dates displayed as the date range above the CalendarView table. Only tasks with a date value that falls within the new date range are displayed in the table | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.06-UT.111 | | |
| **Test Case Description:** | Test method CalendarViewForm.jButton\_calendarView\_resetSelectedWeekActionPerformed (…) | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Place a breakpoint at the start of the method being tested 2. Run the application in debug mode 3. Click the calendar icon 4. Click the *reset* icon (top right corner beside date range) | | |
| **Post Conditions:** | The CalendarView table date range is set to the current week. Tasks are only displayed if their date value fits within the new date range | | |
| **Expected Result:** | Monday and Sunday dates for the current week are calculated and displayed as the date range above the CalendarView table. Only tasks with a date value that falls within the new date range are displayed in the table | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **08/06/2022** | **Comments (if any):** |  |

**Non-functional Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.10-NFT.01 | | |
| **Test Case Description:** | Check database size and type | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document | | |
| **Test Steps:** | 1. Locate the database file in the application directory 2. Review the file size 3. Open the database in console using command:   sqlite3 toDo.db; | | |
| **Post Conditions:** | The database file is opened in the console | | |
| **Expected Result:** | The database file is less than 256mb in size.  The database file is an SQLite database which can be opened using SQLite3 | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **09/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.11-NFT.02-04 | | |
| **Test Case Description:** | Confirm the application is written in the ‘write once run anywhere’ programming language Java | | |
| **Preconditions:** | Nil. | | |
| **Test Data:** | Nil. | | |
| **Test Steps:** | 1. Use an Integrated Development Environment (preferably Netbeans) to navigate to the application directory and open the project 2. Review the code to confirm whether or not it is java based. 3. Open the project properties and check the compile settings. | | |
| **Post Conditions:** | Nil. | | |
| **Expected Result:** | The application is built using the Java programming language and compiled using Java JDK 13. | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **09/06/2022** | **Comments (if any):** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID:** | TS.12-NFT.03 | | |
| **Test Case Description:** | Confirm each core function can be completed in 5 seconds or less | | |
| **Preconditions:** | Database has been created using DDL script | | |
| **Test Data:** | As per test data on page 1 of this document.  **NEW TASK:**  TaskName: “Newly Added Task”, TaskDescription: “This is a new task”, TaskDate: today’s date | | |
| **Test Steps:** | 1. Run the application 2. Using a stopwatch start a timer immediately prior to executing step 3 3. Click the *Add Task* button 4. Add test data in the dialog fields 5. Click *save* 6. Check the timer before resetting it 7. Using a stopwatch start a timer immediately prior to executing step 8 8. Double click on the task that was just added in the previous steps 9. Click the edit icon 10. Change the title to “Edited Task” 11. Click *save* 12. Check the timer before resetting it 13. Using a stopwatch start a timer immediately prior to executing step 14 14. Click on the task that was previously edited 15. Click *Delete Selected* 16. Check the timer before resetting it | | |
| **Post Conditions:** | Nil. | | |
| **Expected Result:** | It takes less than 5 seconds to add, edit or delete a task | **Actual Result:** | As per expected result |
| **Status (Pass/Fail):** | **Pass** | **Executed By:** | **Jason Norton** |
| **Executed Date:** | **09/06/2022** | **Comments (if any):** |  |