For this feature we'll allow the user to remove a transaction from the application by selecting its corresponding row in the table and clicking a "Remove" button. Any time a transaction is removed, the application should update the Total Cost accordingly. If the user wishes to undo their action, they can click an "Undo" button to restore the most recently removed transaction, updating the Total Cost again to reflect that the transaction has been added back. In order to do this we must change the following:

- Model: We'll extend the ExpenseTrackerModel to track removals so they can be reversed. This can be done by adding a stack to hold the most recently removed transaction(s). Whenever a transaction is removed, we add it to the top of the stack. When "Undo" is triggered, we pop the saved transaction off the stack and reinsert it back into the main list of transactions.

- View: We need new UI elements such as an "Undo" button next to or near our "Remove" button. The "Remove" button stays responsible for determining which row is selected and communicating that to the Controller, while the "Undo" button signals that the last removal should be restored. Upon removing or undoing, the view will refresh itself showing the updated transactions and current total cost.

- Controller: We'll introduce methods such as removeSelectedTransaction() and undoLastRemoval(). The removeSelectedTransaction() method will identify which row is selected in the View, pull the relevant Transaction object from the Model, and call a model.removeTransaction() method, which handles logging that transaction for undo. The undoLastRemoval() method will call something like model.undoRemoval(), which re-adds the transaction to the model's list of transactions if there's an available "last removed" entry. Finally, the controller will have the view to refresh the display each time a removal or undo operation is executed.