**Assignment 2a – Inbox app using Vue.js**

Using plain Javascript (no JQuery), this assignment requires you to recreate Assignment 1 using Vue.js but instead of using games, mock some people data. Your mark counts for 20% of your final grade.

Due Date:  **Friday,Dec. 8 @ 11:59 pm**.   
1 bonus mark awarded per week if assignment submitted early.

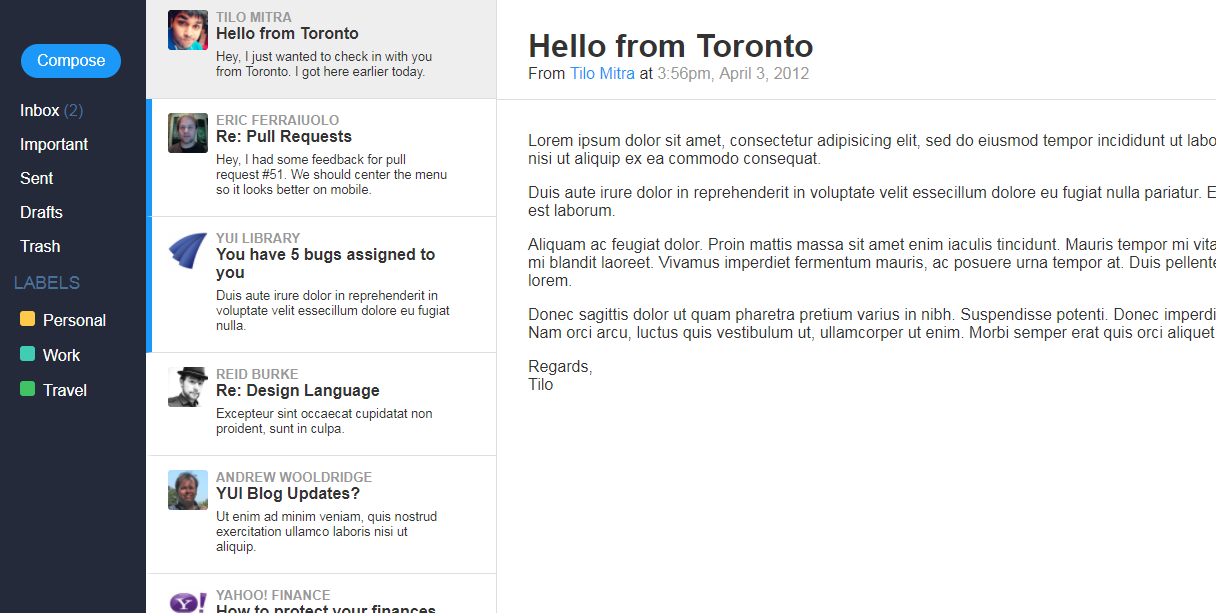
**Submission Requirements via Blackboard:**

* a single .zip file containing all your pages
* a link to your github repository where this assignment is also stored

If you wish to form groups of 2, then each member must individually submit the same zip file along with the other group member name and their contributions.

# Purpose:

In this assignment you will take an existing, static email-like HTML template, and make it dynamic via Vue.js.



# What you’ll need:

* HTML/CSS framework for creating webmail <https://purecss.io/layouts/email/>
* An array of objects taken from <https://www.mockaroo.com/>
* Vue.js - <https://vuejs.org/>

# Application Requirements:

1. Display a set of emails
   1. Your emails array of people data is obtained from mockaroo. See this youtube video on how to do this but remember to include the avatar. <https://www.youtube.com/watch?v=wk-phZTrwPk>
   2. Use Vue’s mustache templates to output each email entry. (this replaces ES6 template literals)
   3. Use Vue’s v-for to list all emails. (this replaces your .map / or other looping method)
2. Clicking a game in the inbox will add some kind of visual indicator to make it look selected and also, display the game in the main section
   1. Use Vue’s v-on:click to handle click events (this replaces addEventListener(‘click). So if an email is clicked, add appropriate CSS class via v-bind to make it look selected
   2. the upper section of main should show corresponding fields ‘from’, ‘subject’, ‘date’ and the main section, should show the email body.
3. Clicking on the Compose button will make a new email automatically appear in the inbox (just like the twitter app from Lab 2 where you added a new hero from a 2nd array of heros) So unlike Assignment 1, you don’t have to implement an input form.
4. Clicking delete button will remove entry from inbox. Clicking on the trash link will allow you to view all items that have been deleted. Clicking the inbox link again will return you back to the inbox view.
   1. v-on:click implemented for delete button, trash link, inbox link which updates its current view
   2. Deleting an entry moves game to the trash. While in trash view, clicking Deleted button moves the entry back to inbox.
   3. Deleting an entry changes the text on the Delete button to “Deleted”. While in trash view, clicking the “Deleted” button will change text to “Delete”
5. Whether you click inbox, or trash, it always displays an accurate, current state of the array of objects.
6. Use local storage such that it stores the current state of objects (assume you’ve deleted/added entries) so if the browser refreshes, it does not return back to the view of original state of objects, but rather will return you to the same view you had before refreshing.
   1. Use localstorage to set information of the current state of objects whenever a change happens
   2. Use localstorage to get information so that a browser refresh remembers current state of object and does not default back to the original view.
7. Document each section of your script with comments. You do not need to document every single line.

# Evaluation Criteria (max 20 marks + 2 possible bonus marks)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria** | **0** | **1** | **2** | **3** | **Marks** |
| **Use of Vue’s {{ mustache }} templates to output text content** | - no game entries shown in inbox | - .innerHTML or .innerText or textContent was found—don’t use these. | - code complete but doesn’t display properly | - All game entries showing | **3** |
| **Use of v-on:click to handle click events** | - No v-on:click is present inside HTML element for every email | - v-on:click refers to an existing function within Vue’s methods. | - game shows in main | - the appropriate CSS class is dynamically added to make email look selected via v-on:click and its corresponding function that it runs | **3** |
| Use of v-bind to handle attributes | - no v-bind is present | v-bind exists and successfully modifies attributes such as adding classes | v-bind’s corresponding variable is found in Vue’s data |  | **2** |
| Use of v-for which lists all emails | - no v-for exists | v-for exists to loop and list all emails |  |  | **1** |
| Compose button inserts new email into inbox | - No email inserted into inbox | - an object that represents a new person exists | - function to add new email is coded | - new email is inserted into inbox | **3** |
| Functional delete button, trash link and inbox link | - No code implemented for delete button, trash link, inbox link | addEventListeners implemented for delete button, trash link, inbox link | - Deleting moves entry to the trash. | - Clicking the delete button on an inbox entry will change text from “Delete” to “Deleted” and vice versa | **3** |
| **Clicking inbox or trash displays current state of objects for that view** | - Current state of objects not stored  - state is only reflected in DOM | - Current state of objects is accurate when inbox or trash is clicked | n/a | n/a | **1** |
| **Local storage used to remember state of objects** | - No local storage used | - used local storage to set information | - working functionality of local storage to set and get information | n/a | **2** |
| **Comments** | 1 to 3 comments | Comment every function to explain what it does | Comment every if/else structure | n/a | **2** |
| **Assignment submitted early?** | - no | - 1 week early | - 2 weeks early |  |  |