**PM2**

1. Provide an example of five hypothetical non-functional requirements for this system. Be sure to include the specific type of requirement discussed in class, with each requirement coming from a unique category.

**1. Unlimited items in TODO list**

**2. Response time less than 2 seconds**

**3. Takes less than 1 TB**

**4. Readable font size and color**

**5. Must use python**

2. Provide an example of five hypothetical functional requirements for this system.

1. **Modifiable font size and color**
2. **Autosaving TODO lists**
3. **Allow adding and deleting items**
4. **Change order of items**
5. **Order items based on name or time added**

3. Think of a specific task required to complete each of the functional requirements and non-functional requirements mentioned above (10 total). Estimate the amount of effort needed to complete this task using function points (i.e., using the values here). Briefly explain your answer.

**Non-Functional**

**1. Creation of an array that can infinitely expand (1 day)**

**2. A sort function that makes the array easy to sort through (2 hours)**

**3. clears the memory of items when not using them (2 hours)**

**4. A function that sets a default text size and color (1 hour)**

**5. Use python IDE (10 minutes)**

**Functional**

**1. A function that allows the user to change the font size and color (2 days)**

**2. Create a pointer to the item so that it still exist when memory is cleared (3 days)**

**3. Create an array that supports adding and removing items (1 week)**

**4. Allows the user to edit the contents of an item so the sort function will move it (3 days)**

**5. Create a struct that keeps track of time that an item is added and other properties (1 hour)**

4. Write three user stories from the perspective of at least two different actors. Provide the acceptance criteria for these stories.

1. **John Doe, 32-year old male looking to get his life together by finding some way to organize his days. John wants an app that will plan the time he wakes up, sleeps, his meal times, and the times for his recreational soccer league games.**

**Acceptance Criteria:**

* **User can submit as many todo list items**
* **User can use group similar items together in their own tabs or boxes**

1. **Jane Doe, is a 19-year old female looking to plan her days in college as a student. Jane needs to track every single one of her homework assignments along with their due dates. The app should also send her notifications when a due date is creeping up.**

**Acceptance Criteria:**

* **User gets a notification when due date for assignment is coming up**
* **User can set due dates for individual assignments and groups of assignments**

1. **LeBron James, is a 39-year old basketball player who needs to plan out his recovery sessions after games. He needs an app that will help him keep track of where his recovery sessions are, what time they are, and after the appointments he wants to check off/remove the items from the list.**

**Acceptance Criteria:**

* **User wants to check off/remove items from the list**
* **User wants to add location and time informations for list items**
* **User wants to be able to have a clickable link that redirects him to the website of the recovery company website.**

5. Provide two examples of risk that could potentially impact this project. Explain how you would mitigate these risks if you were implementing your project as a software system.

1. **One potential risk or limitation of this project would be collaboration. For example, if there is a team that wants to utilize our to-do list, they would all need to be able to access and edit the list. To solve this issue, when implementing our project, we would allow for public access to a specific list through a generated custom link.**
2. **Another potential issue is losing the list made due to a point of failure in saving or data corruption. To mitigate this, we would create a backup for each list which can be used to restore the data.**

6. Describe which process your team would use for requirements elicitation from clients or customers, and explain why.

1. **To discover the requirements of our system, we would gather feedback from one of our primary stakeholders: college students. We would allow access to our program and ask students to plan their week or day using our to-do list. At the end of their experience, we would conduct a mini-interview collecting feedback, ideas, and potential areas of improvement in order to finetune our product.**
2. **This process will help us gain insight into building a robust product that will fit the needs of our clients and by going back to asking them about how they felt with the prototype, we can improve it and fix any problems to better fit their needs.**

### 

### **Requirements Analysis (5%)**

Requirements analysis is the process of *understanding* the requirements for a software application. This deliverable will include *5*:

* Fully dressed use cases
* Model (use case or sequence diagrams) for representing each use case

### **Process Deliverable (2%)**

<https://github.com/KoolBushido/todolist>

**Use Case 1 (Philip):**

Actor:

* User

Scenario:

* User is trying to add a new task to his to-do list

Preconditions:

* User has installed the to-do list application and has created an account
* User is logged into the application

Main flow:

* User selects option to add a new task to the list. Application will ask user to enter task details.. User enters information and application updates todo list with the new task

Sub flows:

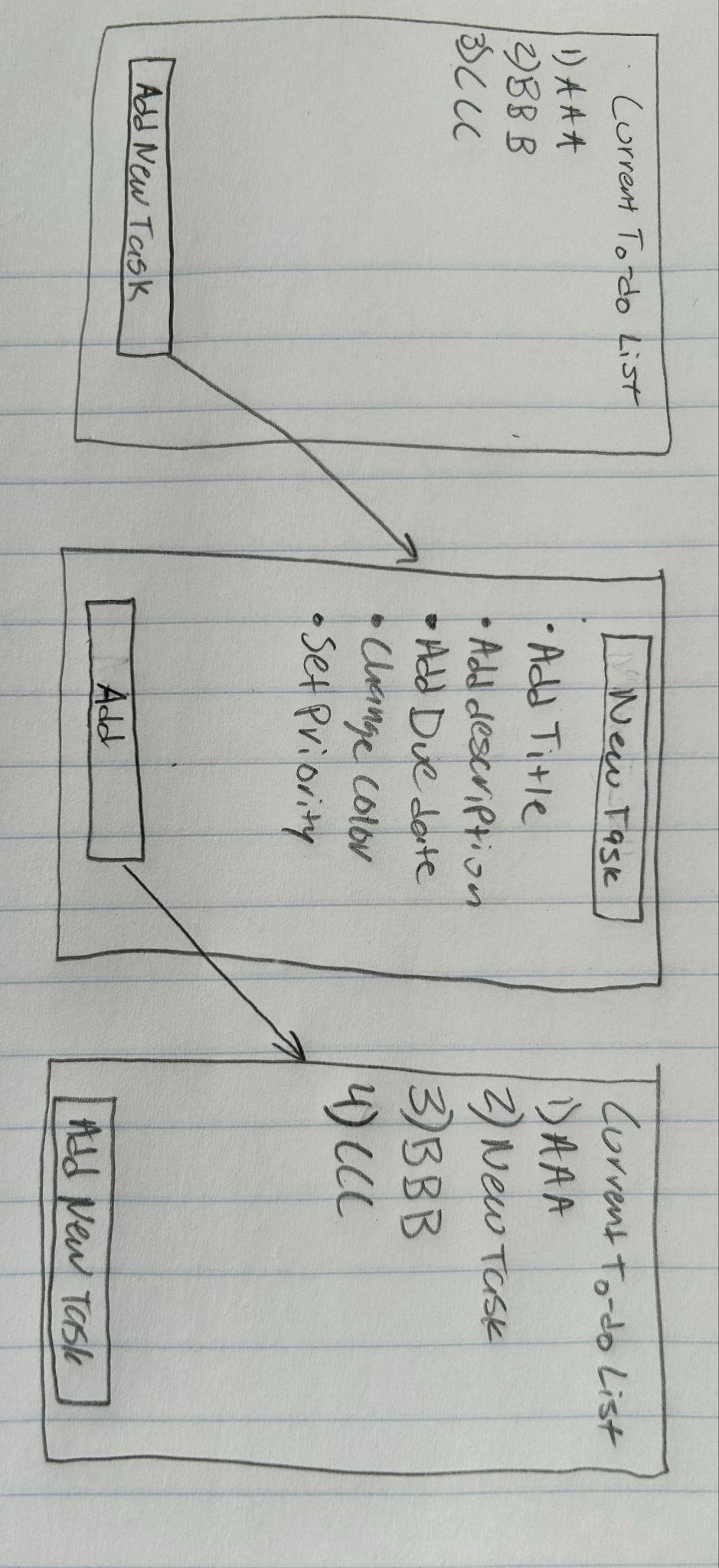
* User clicks “add new task” button on application
* Application prompts user to enter task details such as title, description, due date, color, and priority
* User types information and clicks “add”
* Application returns updated list with the new task added

Alternative flows:

* User wants to edit an existing task
* User wants to delete an existing task
* User changes the priority of a task

Postconditions:

* User successfully adds new task to list, and application returns updated list with new task

****

**Use Case 2 (Jonathan):**

Use case: Change order of list

Precondition:

* User must to authorized to edit the list
* There are 2 or more items on the list

Main flow:

* User requests to swap the order of tasks on their todo list. System prompts user to select the 2 tasks to swap. User selects the 2 task numbers. System swaps the order of the two tasks at the specified task numbers

Subflow:

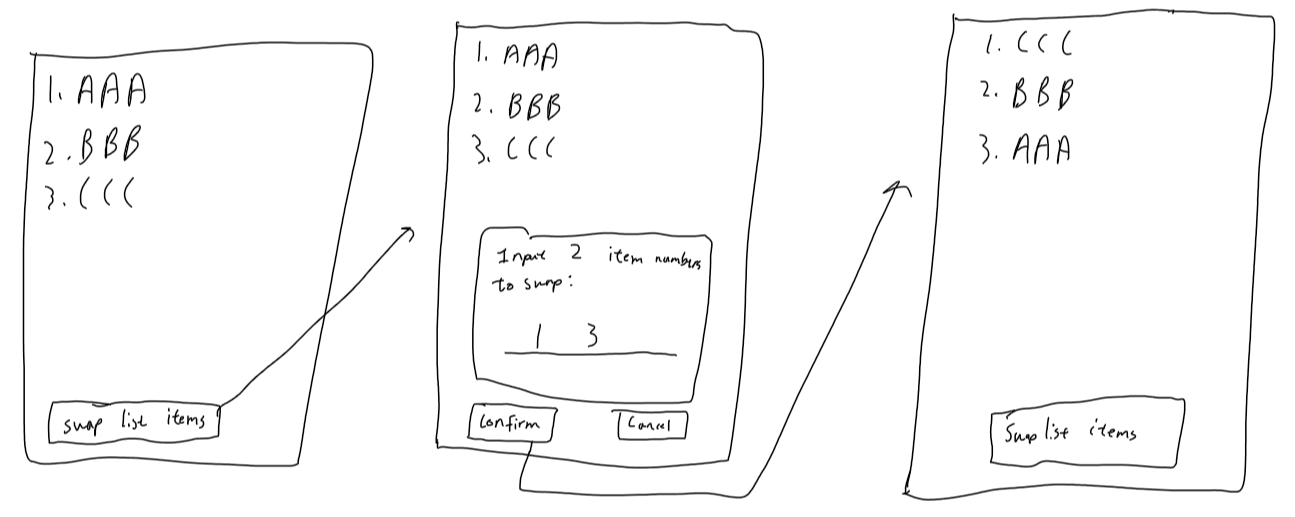
* User selects the “Swap List Items” option
* System will display the list and ask the user to input the 2 task numbers
* User enters the 2 task numbers that they want the order to swap at
* System swaps the two tasks then displays the new list with the two tasks swapped to confirm that it was completed correctly

Postcondition:

* The order of the two specified tasks are swapped on the list

Alternative flows:

* User cancels the task swapping process.
* User enters the same 2 tasks to be swapped so nothing happens
* There are less than 2 tasks in the list so no swapping can be completed
* User is not authorised to edit the list, so no swapping will be completed
* User inputs invalid task numbers such as negative, decimals, or out of range.



**Use Case 3 (Tanay):**

Use Case: Mark Task as Complete

Preconditions:

* Users must be authenticated and have access to the todo list functionality.

Main Flow:

* User selects a task from their todo list to mark as complete [S1]. System updates the task status to "Completed" [S2]. System confirms the task has been marked as complete [S3].

3 Subflows

* [S1] User navigates to their todo list and selects a specific task to mark as complete.
* [S2] System updates the status of the selected task in the database from "Pending" to "Completed".
* [S3] System displays a confirmation message indicating that the task has been successfully marked as complete.

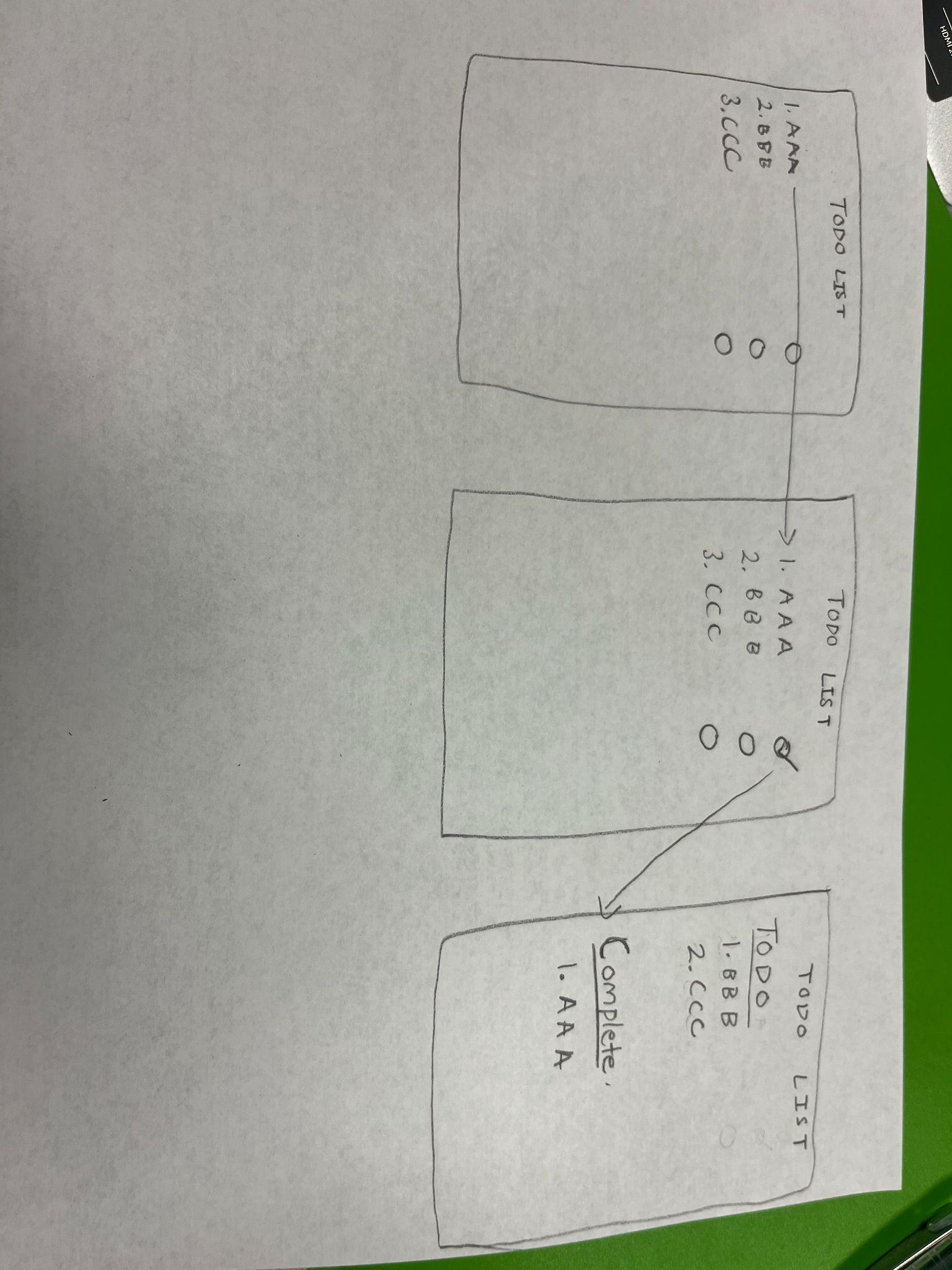
Alternative Flows:

* User attempts to mark a task as complete that is already marked as complete.
* System displays an error message indicating that the task is already completed and prompts the user to choose another task.
* User cancels the task completion process.

- System cancels the task completion action and returns the user to the main menu or previous screen.

- System encounters an error while updating the task status.

- System displays an error message indicating that the task could not be marked as complete and advises the user to try again later or contact support if the issue persists.



**Use Case 4 (Camden):**

Use case: Remove an item from the list

Precondition:

* User must to authorized to edit the list and have logged onto the application

Main flow:

* User requests to delete a specified item(s) on their to-do list. System prompts user to select the task(s) to remove. User selects the task(s). System removes the specified task(s) and updates the list accordingly.

Subflow:

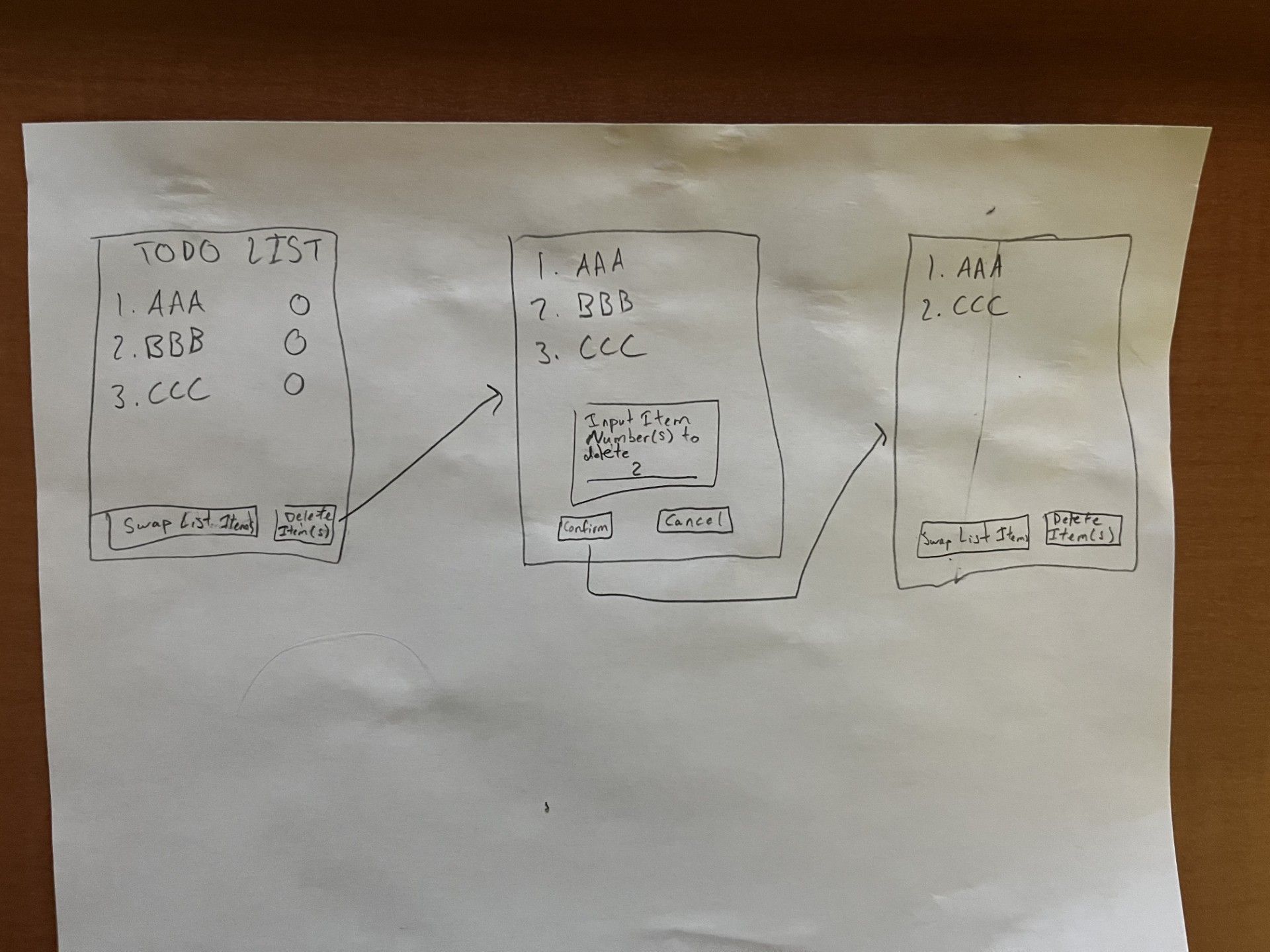
* User selects the “Delete Item(s)” option
* System will display the list and ask the user to input the task number(s)
* User enters the task number(s) that they want to remove
* System deletes task(s) then displays the new list with the task(s) deleted to confirm that it was completed correctly

Postcondition:

* The items are removed from the list

Alternative flows:

* User cancels the delete task process
* User deletes a task and then undo’s the delete
* User tries to delete a task on an empty list so nothing happens
* User is not authorised to edit the list, so no deleting will be completed
* User inputs invalid task numbers such as negative, decimals, or out of range

****

**Use Case 5 (Ahman):**

Use case: Change font of the list.

Precondition:

* User is authorized to edit the list

Main flow:

* User requests to change the current font of the list. System prompts the user to choose a suitable font from the available options. User selects the font, the system changes the list font from that point forward.

Subflow:

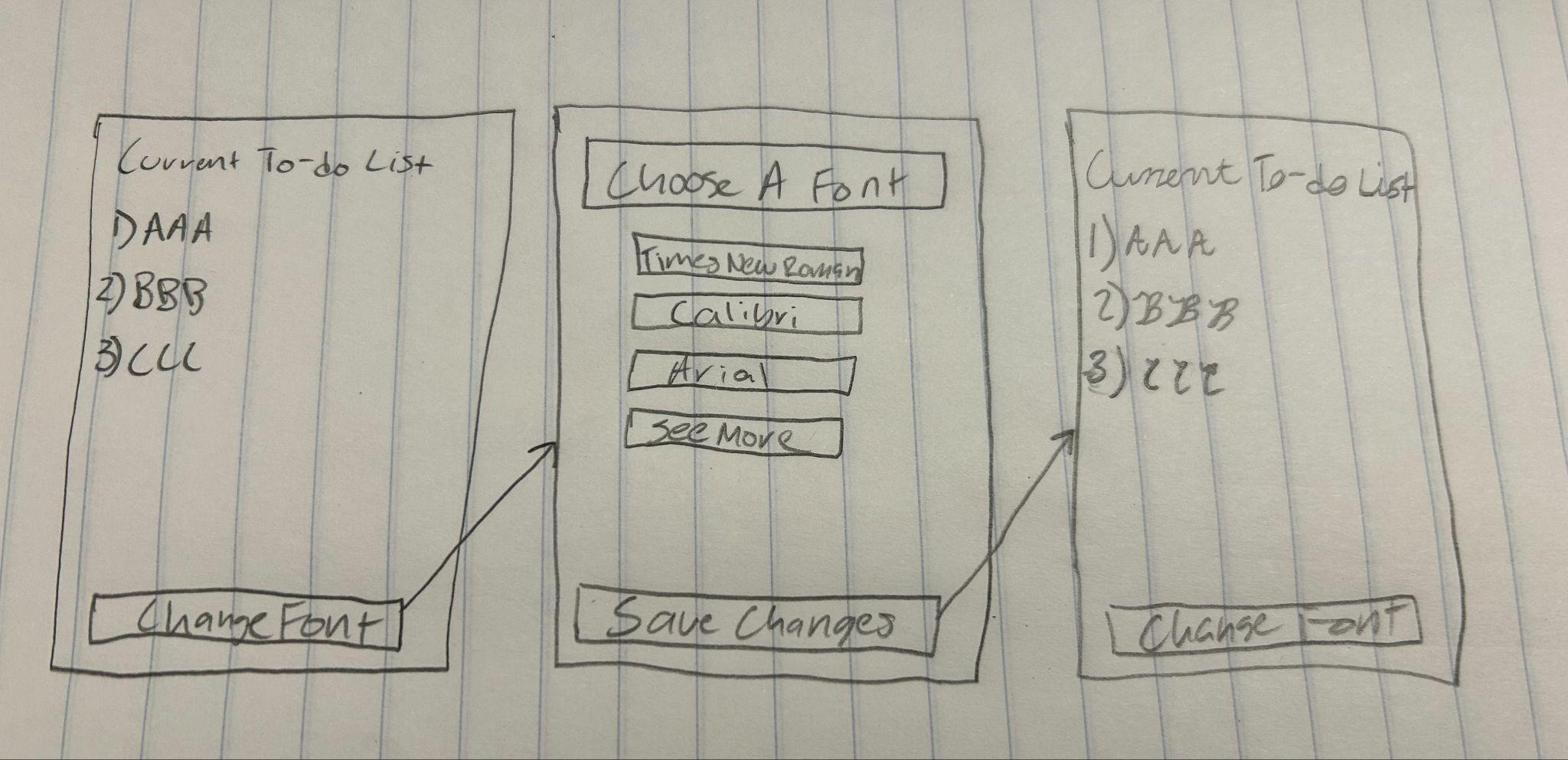
* [S1] User selects the “font” option from the taskbar options.
* [S2] User selects the desired font.
* [S3] The system changes the font.

Postcondition:

* The new font is in effect from then onward.

Alternative flows:

* User attempts to select the same font so nothing happens
* User doesn’t select a font so nothing happens.
* User isn’t authorized to edit the list, so the font won’t be changed

****