

# Plant Care Manager

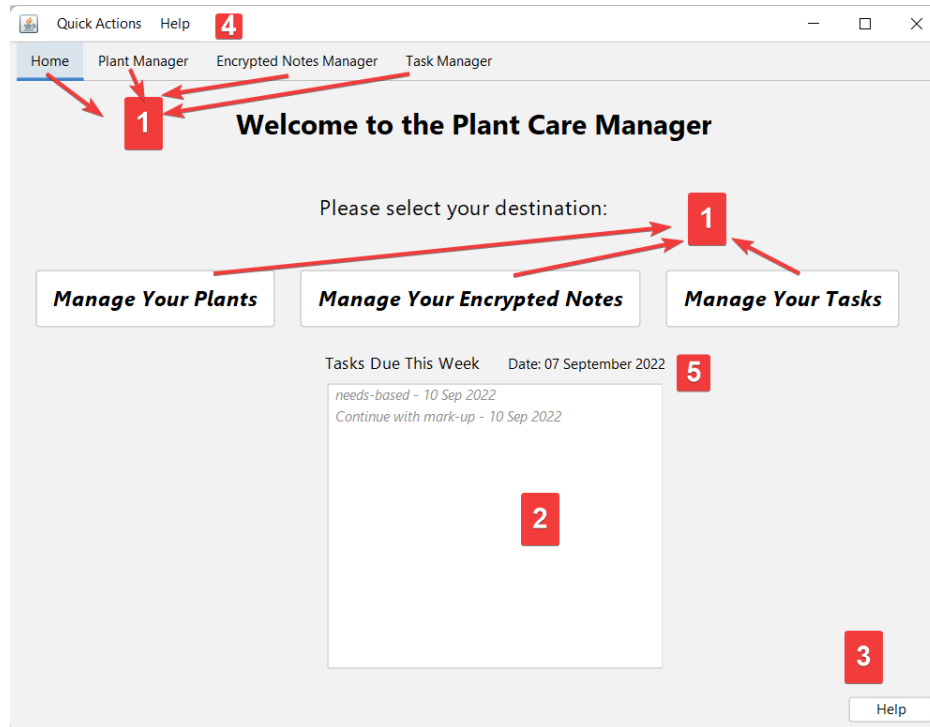
Kelvin Wei

Design document (Phase 2)

# Index

<b>Index</b>	<b>2</b>
2.1 Interface Design	3
2.2 Program Flow	14
2.3 Class Design and OOP Principles	15
2.4 Secondary Storage Design	17
2.5 Explanation of Secondary Storage Design	18
2.6 Explanation of Primary Data Structure related to Secondary Storage	19

## 2.1 Interface Design



### Description

This is the homepage screen in which the user first sees when the application runs. The user uses it to navigate to "Plant Manager", "Task Manager", "Encrypted Notes Manager"

#### 1) Navigation

The user can alternatively press these buttons to navigate them through the application. The corresponding button will navigate them to the desired tab/screen.

#### 2) List of Tasks Due Soon

Shows uncompleted tasks that are due within 7 days. The tasks would be displayed with its heading name and due date. The task's data would be pulled from a text file stored on the user's secondary storage.

#### 3) Help Button

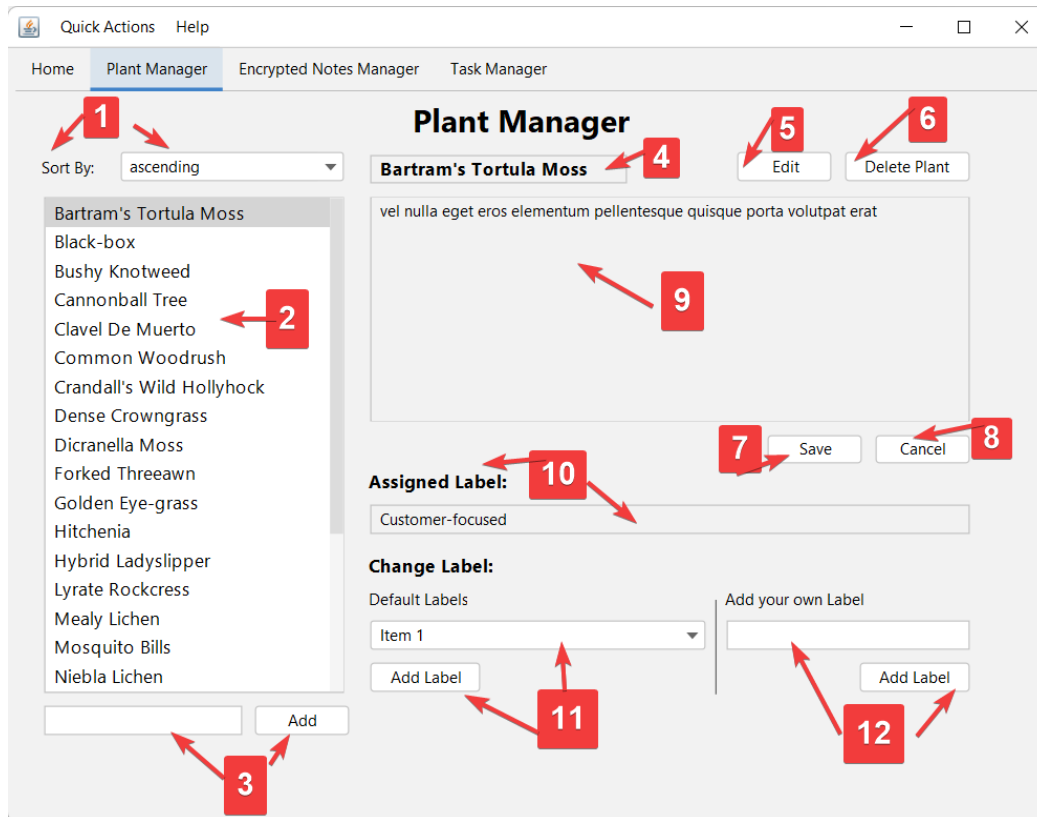
The user presses the button for a window to pop-up showing information to solve common problems

#### 4) Menu Bar

The Menu bar contains menu items that the user can click up to display actions that the user might want the application to do.

#### 5) Current Date

Displays the current date. The current date is taken from the local computer running the application.

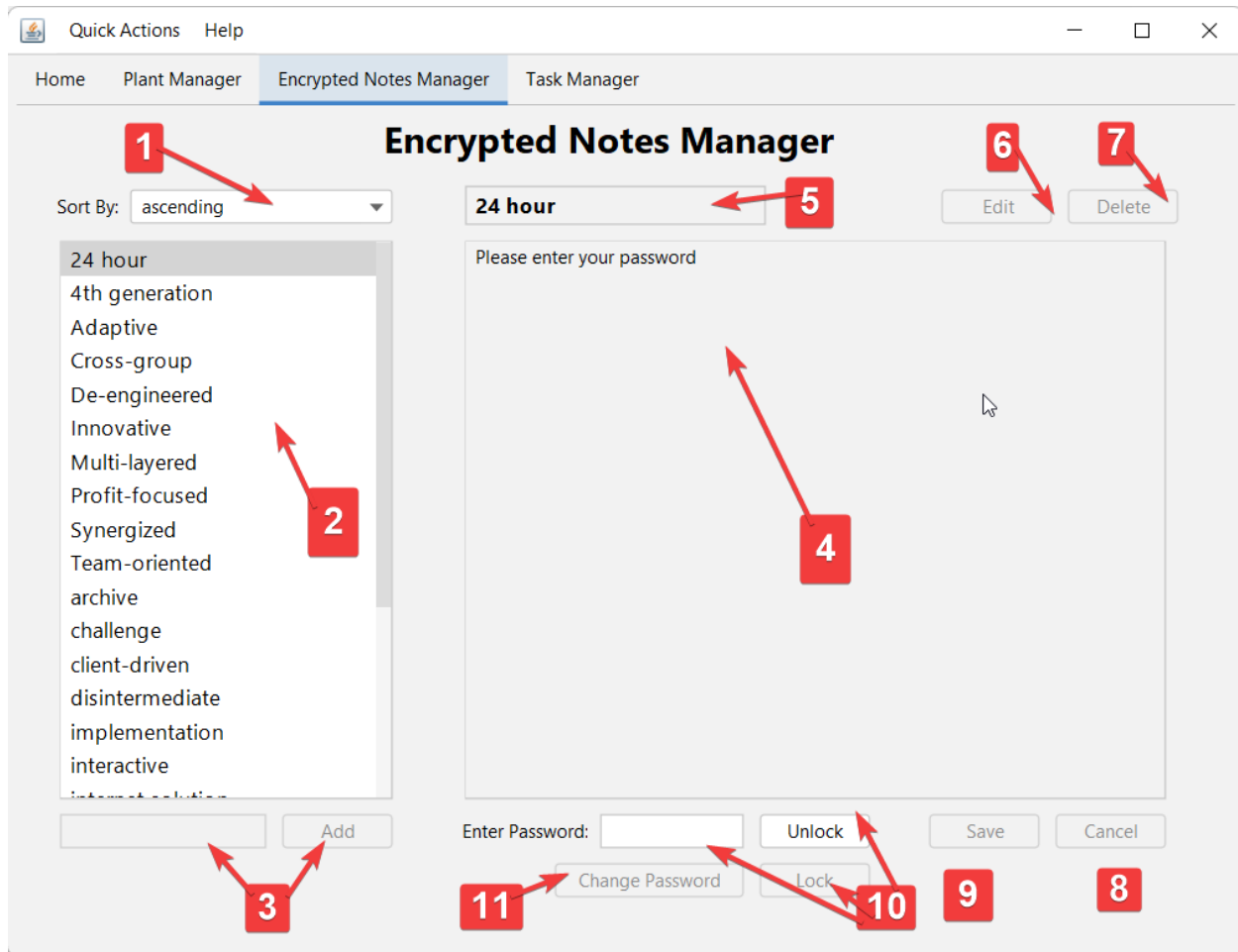


<u>Description</u>	
This screen is used by the user to manage and view their plants.	
1) Sort - Combo Box	The user can press the combo box to change how the list below(2) sorts
2) List of plants	Display the plants that are stored in the application in a list that are pulled from the text file stored. The list shows the name of the plants and the user can select them to display more information about them
3) Text Field and Add Button	The user can type in the text field and press the “Add” button to add a new plant. The text field stores the name of the plant to be added.
4) Plant Name - Text field	This is a text field which would display the name of the plant selected from the list(2). The Text field wouldn't be able to be edited until the edit button is pressed(5)
5) Edit Button	Pressing this button would enable editing of the plant details that are currently selected. Pressing this button would enable editing on Plant Name(4), Plant Note(9), and enable the save and cancel buttons (7-8).
6) Delete button	Deletes the selected plant from the list(2).

<b>7-8) Save and Cancel Button</b>	These buttons are normally disabled (unable to be clicked by the user). The buttons are enabled when the user presses the edit button. The buttons are used to confirm the changes or to cancel the changes.
<b>9) Plants Notes - Text Area</b>	Display the notes that the selected plant has from the list(2)
<b>10) Assigned Label</b>	Displays the assigned label that the selected plant has from the list(2).
<b>11) Default Label Button</b>	The user can select a label from the combo box. Selecting a label and pressing "Add Label" will change the assigned label to the selected label
<b>12) Custom Label</b>	The user can type their own label in the text field. When the data is entered in the text field, the user can press the "Add Label" button to change the assigned label to the custom label.



<b>6) Task Completed Button</b>	This button is a toggle button. This toggle button is used to indicate if a task is completed or not. The button's description would change based on if it is selected or not (Task Completed/Task Incomplete).
<b>7) Edit Button</b>	Pressing this button would enable editing of the task details that are currently selected. Pressing this button would enable editing on Task Name(5), Task Note(9), and enable the save and cancel buttons (10-11).
<b>8) Delete Button</b>	Deletes the selected task from the list (3)
<b>9) Task Notes - Text Area</b>	Displays the notes that the select task contains from the list(3)
<b>10-11) Save and Cancel Button</b>	These buttons are normally disabled (unable to be clicked by the user). The buttons are enabled when the user presses the edit button(6). The buttons are used to confirm the changes or to cancel the changes.
<b>12) Assigned List</b>	Displays the assigned label that the selected task has from the list(3).
<b>13) Default Label</b>	The user can select a label from the combo box. Selecting a label and pressing "Add Label" will change the assigned label to the selected label
<b>14) Custom Label</b>	The user can type their own label in the text field. When the data is entered in the text field, the user can press the "Add Label" button to change the assigned label to the custom label.
<b>15) Due Date</b>	Displays the due date of the selected task from list(3). The date can be edited until the edit button(16) is selected.
<b>16) Edit Button</b>	Pressing this button would enable editing of the date details of a task. Pressing the button would enable editing on the Due Date(15) and enable the save and edit buttons(17-18).
<b>17-18) Save and Cancel Buttons</b>	These buttons are normally disabled (unable to be clicked by the user). The buttons are enabled when the user presses the edit button(16). The buttons are used to confirm the changes or to cancel the changes.



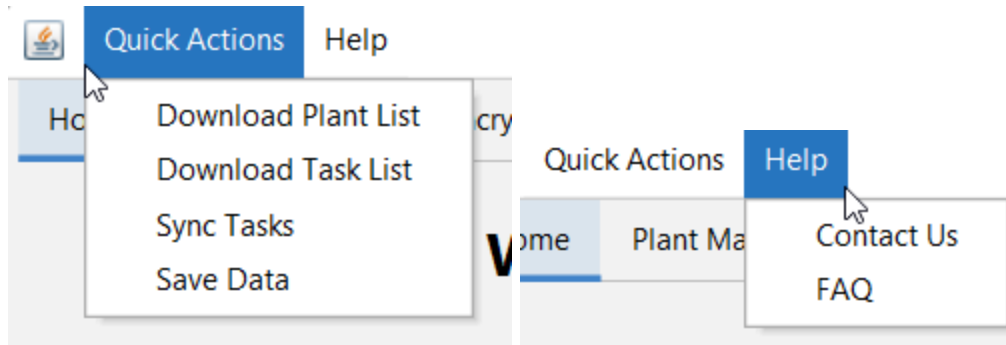
### Description

This is screen in which the user manages and views their encrypted notes

<b>1) Sort - Combo Box</b>	The user can press the combo box to change how the list below(11) sorts
<b>2) Note List</b>	Display the notes that are stored in the application in a list. The list shows the name of the notes and the user can select them to display more information about them
<b>3) Text Field and Add Button</b>	The user can type in the text field and press the “Add” button to add a new note. The text field stores the name of the note to be added.
<b>4) Note Contents - Text Area</b>	Displays the contents of the note that is selected from the list(11). The contents would be empty until the user unlocks the note(8) by entering the password to the note.
<b>5) Note Name - Text Field</b>	This is a text field which would display the name of the note selected from the list(11). The Text field wouldn't be able to be edited until the edit button is pressed(4)



<b>6) Edit button</b>	Pressing this button would enable editing of the note details that are currently selected. Pressing this button would enable editing on Note Name(3), Note Contents(6), and enable the save and cancel buttons (7).
<b>7) Delete Button</b>	Deletes the selected Note from the list (11)
<b>8-9) Save and Cancel Button</b>	These buttons are normally disabled (unable to be clicked by the user). The buttons are enabled when the user presses the edit button(4). The buttons are used to confirm the changes or to cancel the changes.
<b>10) Text Field and Unlock Button</b>	The user would need to enter the password into the text field and press unlock to verify the password and decrypt the note. The change password button(11), edit (6), delete(7) and add(3) would be disabled if the note is locked. If the note is decrypted the buttons then would be enabled
<b>11) Text Field and Change Password Button</b>	The "Change Password" button is normally disabled (unable to be clicked by the user). The button is only enabled after the selected note is unlocked. To change the password the user would need to type the new password into the text field above and press the "Change Password" button.



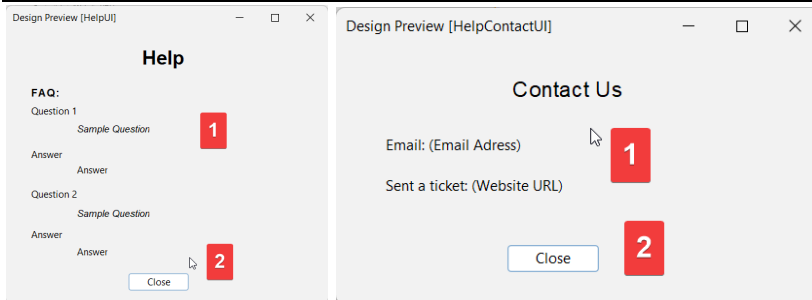
<b>Description</b>	
These are the menus that are present on the top of the windows. The user can press them to reveal a list of actions	
<b>Quick Actions</b>	
Download Plant List	The user presses it for a pop-up to appear. The pop-up allows the user to complete the process of downloading a plant list
Download Task List	The user presses it for a pop-up to appear. The pop-up allows the user to complete the process of downloading a task list
Sync Tasks	The user presses it for a pop-up to appear. The pop-up allows the user to complete the process syncing tasks
Save Data	The user presses it for the program to save the current data on the program to a text file
<b>Help</b>	
Help pop-up	The user presses it for a window to pop-up showing information to solve common problems
Contact Us	The user presses it for a pop-up containing the ways for a user to contact the developer



#### Description

This pop-up is prompted when the user encounters an error

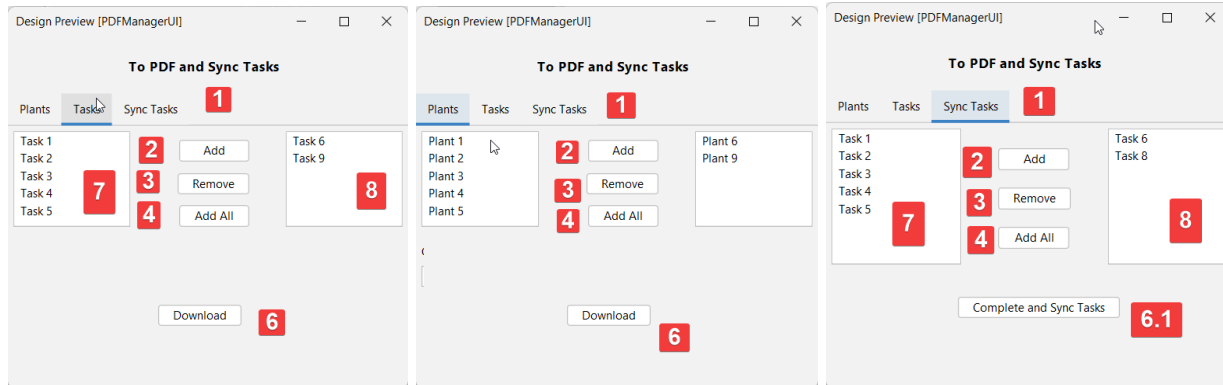
1) Message - "Sample Message"	A message will be displayed to the user based on what the error is
2) OK button	The user presses the button to close the window.



#### Description

The pop-up appears then the help button is pressed or selected from the menu

1) Content	Displays help information. FAQ information on help pop-up. Contact up pop-up contains email and ticket. The data pulled from secondary storage
2) Close Button	Closes the pop-up

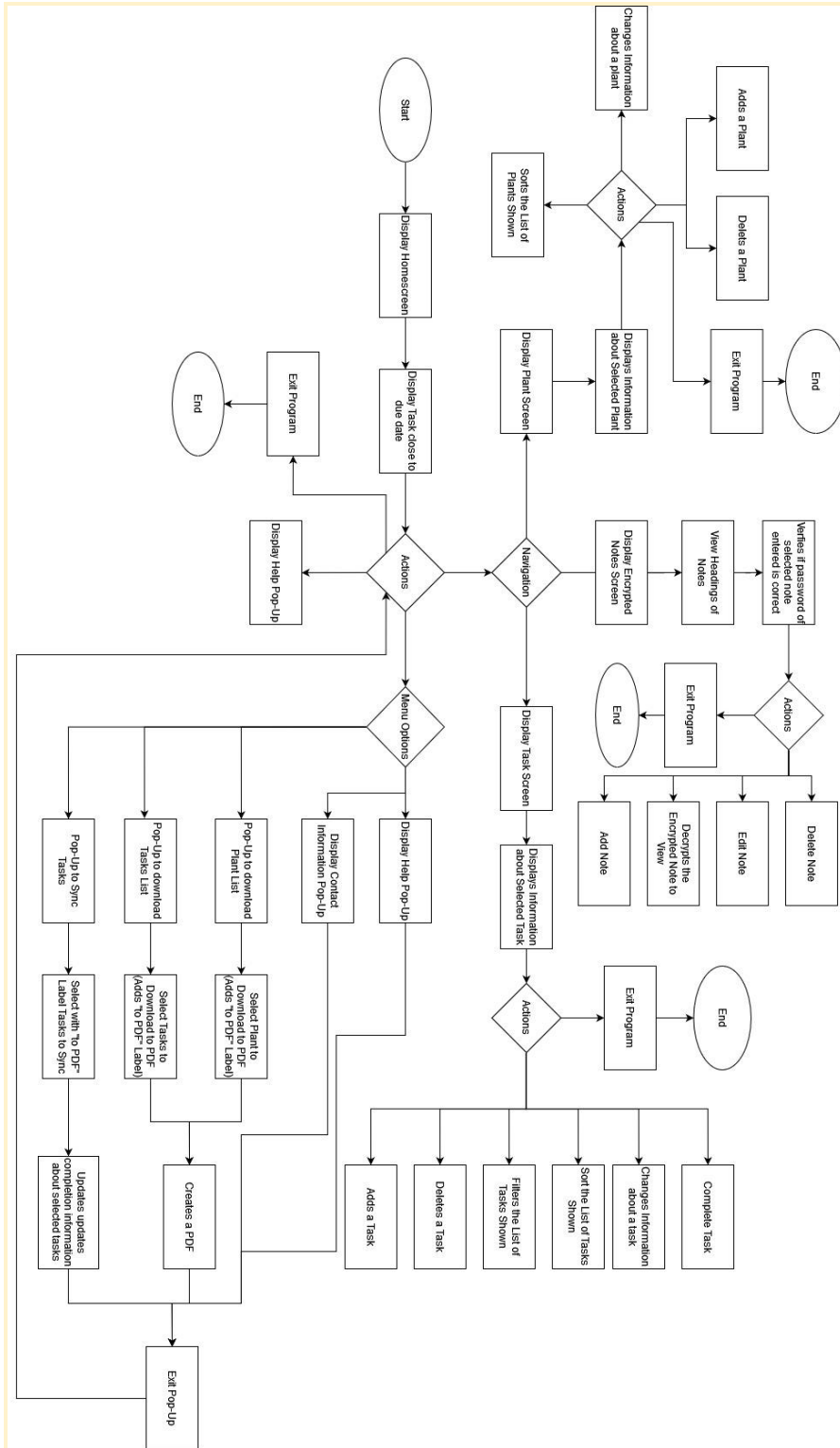


### Description

This pop-up appears when the user wants to download a plant/task list to pdf or sync tasks. This is access from the menu

1) Tabs	The user can click on the tabs of the tabbed pane to navigate through the plant, task, or sync tasks tabs.
7) Lists (left)	Displays the list objects(task/plant) that are present on the program. The user can select them. These tasks/plants are existing tasks/plants that the user has added and are present in secondary storage.  Sync tasks display tasks that have the label "to pdf"
8) Lists(right)	Displays a list of all the Objects (plants/tasks) that the user has selected and added from the left list.
2) Add Button	The user can select an object on the left list(7) and press the add button to add an object to the right list
3) Remove Button	The user can remove an object from the right list(8) by selecting the object on the right list and pressing the delete button.
4) Add all button	The user can all all objects from the left list(7) into the right list by pressing the add all button
6) Download Button	Proceeds to generate the pdf file from the user entered data and save the file in the user entered destination.
6.1) Sync Button	Will proceed to complete the selected tasks and remove the "to pdf" label from the task

<https://app.diagrams.net/#G1EutWt4BwAqJwybTJBcneZ1Yp0e3nzZWs>



## 2.3 Class Design and OOP Principles

Plant	Stores the data of a plant
<b>Properties:</b> - name : String - label : String - note : String	Stores the name of the plant Stores the label attached to the plant Stores notes attached to the plant
<b>Methods</b> + Constructor(name : String, label : String, notes: String) : void + Constructor(name: String) : void + getName() : String + getLabel() : String + getNotes() : String + setName(name : String) : void + setLabe(label : String) : void + setNotes(note : String) : void + toString() : String	Initialises variables  Initialises variables if only name is given. returns the name of the plant returns the label of the plant returns the notes of the plant changes the name of the plant change the label of the plant changes the notes of the plant returns a string containing all plant data













Task	Stores the data of a task
<b>Properties</b> - dueDate : LocalDate - doneOrNot : boolean	Stores the dueDate of the task Stores if the task has been completed or not
<b>Methods</b> + Constructor(name : String, label : String, notes: String, dueDate : LocalDate, doneOrNote : boolean) : void + Construtor(name : String) : void + getDueDate() : LocalDate + getDoneOrNot : Boolean + setDueDate(date : LocalDate) : void + changeDoneOrNot() : boolean   toString() : String	initialises the variables.  initialises the variables if only name is given returns the due date of the task returns if the task if done changes the due date of the task changes the value of the boolean. If true it would change it to false and return false converts a text date into a LocalDate variable   overridden method that returns all data of task

EncryptedNote	Stores data about an encrypted note
<p>Properties</p> <ul style="list-style-type: none"> <li>- heading : String</li> <li>- content : String</li> <li>- password : String</li> <li>- encryptedContent : String</li> <li>- encryptedPassword : String</li> <li>- decrypted : boolean</li> </ul>	<p>Stores the heading of the note</p> <p>Stores the decrypted content</p> <p>Stores the decrypted password</p> <p>Stores the content that has been encrypted</p> <p>Stores the password that has been encrypted</p> <p>Stores if the note has been unlocked by the user</p>
<p>Methods</p> <ul style="list-style-type: none"> <li>+ Constructor(heading : String, encryptedContent : String, encryptedPassword : String, masterPassword) : void</li> <li>+ Constructor(heading : String) : void</li> <li>+ getHeading() : String</li> <li>+ getContent() : String</li> <li>+ getPassword() : String</li> <li>+ getEncryptedContent() : String</li> <li>+ getEncryptedPassword() : String</li> <li>+ setHeading(heading : String) : void</li> </ul> <p>- static encryptString(plainText : String) : String</p> <p>- static decryptString(base64CiphertextAndNonceAndSalt : String) : String</p> <p>- static encrypt(plaintext : byte[ ], key : byte[ ]) : byte[ ]</p> <p>- static decrypt(ciphertextAndNonce : byte[ ], key : byte[ ]) : byte[ ]</p>	<p>Initialises the variables. decrypted will be initialised to a default value.</p> <p>Initialises the variables if only heading is given</p> <p>returns the heading of the note</p> <p>returns the content of the note</p> <p>returns the password of the note</p> <p>returns the encrypted content of the note</p> <p>returns the encrypted password of the note</p> <p>changes the heading of the note</p> <p>generates an encrypted String when text is inputted into the method. It is a base64 encoded string.</p> <p>decrypts the encrypted base64 encoded string text. The method returns the result.</p> <p>encrypts raw binary data and returns it</p> <p>decrypts raw binary data and returns it</p>

<b>UIManager</b>	Runs the UI Classes and stores methods related to the UI
<b>Methods</b> + constructor() : void  + static toListModel(array : ArrayList<String>) : DefaultListModel + static toArrayList(array : ListModel) : ArrayList + static switchModels(fromList : JList, toList : JList) : void + static removeListItem(model : DefaultListModel, index : int) : Object + static addListItem (model : DefaultListModel, element : Object) : void	Initialises the UI  converts an ArrayList to a ListModel  converts an ListModel to an ArrayList  swaps items in two DefaultListModel  removes an item from a ListModel  adds an item from a ListModel



## PlantCareManager

 - ArrayList<Task> tasks  
 - ArrayList<Plant> plants  
 - ArrayList<Note> notes  
 + final String PLANT  
 + final String TASK  
 + final String ENCRYPTED\_NOTE  
 + final String ASCENDING  
 + final String DESCENDING  
 + final String ALPH\_LABELS  
 + final String DUE\_DATE

Global ArrayList to store objects

Final static variables for identification

 + PlantCareManager()  
 + ArrayList<String> getTasksDueSoon()  
 + ArrayList getFilterList(String filterBy)  
 + ArrayList<String> getSortList(String sortBy, String type)  
 + int searchIndex(String type, String name)  
 + Plant displayPlant(String search)  
 + Task displayTask(String search)  
 + String getNoteHeading(String search)  
 + String getNoteContent(String search)  
 + void save()  
 + String add(String type, String name)  
 + boolean delete(String type, String name)  
 + String changeAssignedLabel(String type, String objectName, String labelName)  
 + String edit(String type, String name, String note)  
 + void changeTaskDate(String task, LocalDate date)  
 + String unlockNote(String note, String password)  
 + void lockNote(String note)  
 + void setTaskCompleted(String task)  
 + String changePassword(String note, String password)  
 + ArrayList<String> getSyncList()  
 + void toPDF(String type, ArrayList list)  
 + String toPDFFormat(String type, ArrayList list)  
 + void syncTasks(ArrayList list)

constructor to read text file  
 returns a list of tasks due soon  
 returns a list of filtered tasks  
 returns a sorted list  
 returns the index of a object

returns a plant object  
 returns a task object  
 returns a note heading  
 returns a note content

saves the data in program in text file  
 adds a new object to list  
 deletes an object from list  
 changes an label given

edits the name and note of task and plant  
 changes the task date

unlocks a note  
 locks a note  
 sets a task as complete  
 changes a password from task  
 gets a list of task with "in pdf" label  
 creates a pdf from list  
 formats a string for pdf

sync "in pdf" tasks to completed

## 2.4 Secondary Storage Design

*The data for the program will be stored in text files*

### **Plant Storage**

Format:

Plant Name;;label;;notetext

- Each variable will be separated by “;”.
- If there is no label/noteText. The program will write “empty”
- All elements in the text file are of type “String”

### **Format for Task Storage**

Format:

Task Heading;;label;;Notes;;DoneOrNote;;Due Date

- Each variable will be separated by “;”.
- If there is no label/noteText/Due Date. The program will write “empty”
- Date will be converted to String in the format [yyyy-mm-dd]
- boolean variables(DoneOrNote) will write true/false
- Task Heading, Label and Notes are of type “String”.

### **EncryptedNote Storage**

Format:

Note Heading;;password;;Note contents

- Each variable will be separated by “;”.
- The decrypted method will not be stored as the decrypt/encrypt process happens in the program and don't need to be stored
- password and note contents will be stored as string but will be encrypted
- All elements in the text file are of type “String”

## 2.5 Explanation of Secondary Storage Design

The data is stored in a text file because the data stored is simple and not expected to be in a large quantity. Databases are slower to access than a text file, therefore, the performance of the program will be impacted if it uses a database to access simple pieces of data. Although a database provides more security, the data on the encrypted notes text file isn't compromised as the data is encrypted.

## 2.6 Explanation of Primary Data Structure related to Secondary Storage

Each line in the text file represents a separate entity. Therefore, each line is stored in a class. The Plant, Task and Encrypted Notes Classes store values from the text files in Primary Storage. The plant class will store data about a plant's label, name and note. The task class will store data about a specific task. The task class inherits from the plant class, therefore, it would store what the plant class stores with extra details (LocalDate to store the due date of the task and Boolean to store if the task is completed). The EncryptedNote class will store data about a specific encrypted note (the note's heading, the note contents encrypted, password encrypted and if the note has been decrypted)