**Planner4Life Developer Guide**

Planner4Life is a standalone .exe file that performs the functions of a calendar planner. It allows users to input multiple tasks with their time, description and date, with which Planner4Life will create a running schedule. Planner4Life also contains functions that allows the user to perform CRUD processes on the schedule, as well as to organize the content into easily readable formats.

**Challenges**

* Implementing persistent memory for Planner4Life using a standalone .exe file
* Creating an architecture that is as modular as possible to facilitate future development and easy testing

**Principles**

* Ease of Use

We believe in creating a planner application that is easy for the user to learn and use efficiently.

* Clean code

Writing code that is concise as it can be, and easy to read at the same time

* Safe code

Application of defensive coding; comprehensive exception handling capabilities that provides consistent feedback to the user.

**Product Architecture**

<Diagram>

Given above is an overview of the main components.

Task: contains variables for storage and organization of the various information fields of a particular task entry. It also includes their getter and setter functions.

Planner: Organizes and arranges Task objects into their proper orders and categories. Contains functions to manipulate Task groups as well as getter and setter functions. Also converts task groups into readable data.

GUI: The GUI is seen by users. It is the platform of communication between the program and users. It takes in users’ commands, provide users with feedback on the outcome of the commands and displays different views of events depending on users’ preference.

Logic: Logic is the central dispatcher of the program which receives user input from GUI and requests relevant components including Planner and Task to carry out appropriate operations on the user input. The content to be displayed as well as outcome of operations are dispatched from Logic to GUI to be viewed by users.

**GUI**

(UI Diagram)

The UI is a CLR Windows form that takes in the user input and displays the output from Planner4Life. Its primary objectives are to display the information in a well-organized manner to the user. By maintaining semantic meaningfulness of the information displayed, the UI allows the user to understand his schedule.

All UI pages are displayed in the Display Window. The UI consists of two categories of pages:

* Schedule Content Display

Schedule Content Display windows are accessible through text commands or button clicks.

<Home> displays the tasks for the next 7 days.

<Upcoming> displays all the tasks past the next 7 days till the last entry.

<Missed> displays tasks that have concluded prior to the current date.

All pages are indicated by their respective highlighted button.

* Program information pages

Program Information pages are only accessible through text commands.

<Help> displays a help window.

<Search> displays the search results that match the search keyword.

<All> displays every task entered into Planner4Life, including past tasks.

Request flow

1. take in command-format String (User Input)
2. convert String to string
3. pass string to Logic
4. Logic processes the commands in the string
5. Logic return output-ready string
6. convert string to String
7. print output in Display Window
8. print output in feedback prompt

**Logic**

The Logic component handles the business logic of Planner4Life. It takes in command input from GUI and processes the information using the Planner and Task Classes. It then receives the feedback from the Planner and Task Classes and passes it back to GUI for the user to view. Logic also makes use of the Storage in order to save and load data.

In particular, Logic is responsible for these:

* Directing the user commands to the correct functions
* Passing the correct information to be displayed
* Activating save and load features

Logic API

Represented by 17 public functions:

* void processUserInput(string userInput, string currentView):

Takes in user input and the current view from GUI and classifies it into the appropriate type of operations based on the command word

Update the content to be displayed on GUI

* string extractCommand(std::string& userInput):

Takes out the first word in the original user input and identifies it as the command word

* processCommand functions (add/edit/delete/help/all/search/load)

Dispatches task details and current view to the relevant function depending on the command word

* displayOutcome() and displayContent() functions:

Provides GUI with outcomes of operations and content to be displayed

* void save(string fileName):

Allows all tasks to be saved to a .txt file

Preconditions/Postconditions

**Storage**

The Storage Class manages the save and load features in Planner4Life. It receives the data to be saved from the Logic Class and saves it in a text file. It is also able to receive, through the Logic class, a filename and/or save location entered by the user and hence give the user flexibility in what to save the date as and where to save it. Lastly, the Logic Class is able to obtain data from Storage on startup through the load functions.

In Particular, Storage is responsible for these:

* Save data from the major list into a text file
* Load data from the text file in to the major list
* Set filename and file locations

Storage API

* void save(string fileName, string fileLocation)

Saves the files to the selected location as a .txt file and with a desired filename.

* void load(string fileName, string fileLocation)

Loads the files from the selected location as a .txt file and with the requested filename.

**Planner**

The Planner class manages all tasks created by users, by categorising them into the relevant lists. It receives new tasks from Logic and adds them to a master list of all tasks in chronological order, and the master list aids in the formation of the other lists. The Planner class methods (delete, edit, search, undo, clear, save tasks) are called by Logic to perform appropriate operations on the master list depending on the commands given by users. It also provides Logic with the content to be displayed in the different views of GUI as well as the status of operations.

In particular, Planner is responsible for these:

* Perform operations on the master list of all tasks
* Generate content to be displayed in the different views of GUI
* Generate status of operations to be displayed GUI

Planner API

Important Attributes

List of Tasks: All, Home, Missed, Upcoming

The All List contains all the tasks inserted into the planner. The Home, Missed, Upcoming Lists contain the various tasks that apply to them depending on the date.

Important Functions

* string <process>Task(Task content) process: *add, delete, edit, undo, clear, save*

The above functions will manipulate the All List according to the command entered by the user.

* string toString(string nameOfList);

The toString function passes the display of the various Lists to the Logic Component on request.

* string statusToString(string command, Task theTask);

The statusToString function will return feedbacks and confirmations of the major functions being successful or unsuccessful.

Preconditions/Postconditions

**Task**

The Task class is in charge of creating new Task objects when prompted by Logic as well as providing Planner methods with information of Task objects specified.

In particular, Task is responsible for these:

* Create new Task object
* Retrieve information of existing Task object
* Search for target word

Task API

* void addDetails(string):

Receive task details from Logic, parse the details and store the relevant information in a newly created Task object.

* bool isSearchTargetPresent(string):

Search for keyword in the description attribute of the Task object

* Getter functions

Retrieve the required information from the Task object