Summarizer Specs Sheet

Summary Logic:

- Handled By: Jayanth Kumar
- Role:
    - Develop an Algorithm to extract the summary of the discussed contents over the chatting section
    - The Algorithm should be well structured to take in user messages, perform various Natural Language Processing techniques to extract useful information and form the messages summary.

Summary Logic SubModule

Abstract:

1. Developing the application, where we support the end-user by adding the summarization feature, which sums up the conversations happened during the session in the chatting module.
2. The Summary Logic Module takes trigger from from the Session Manager to know when to generate summaries of happened discussions.

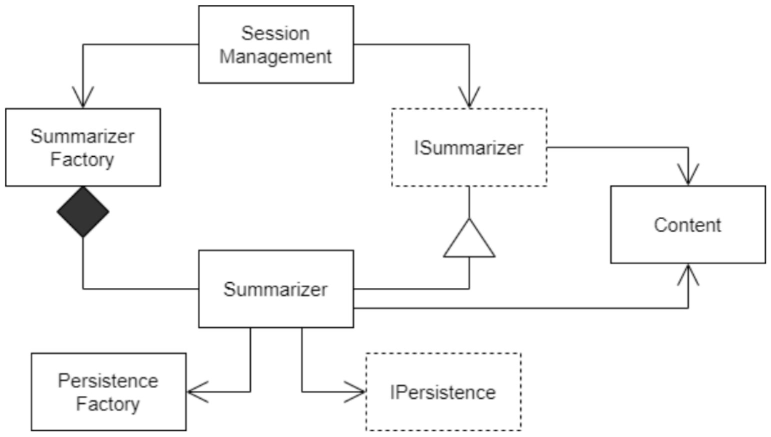
Objective:

1. One interface can be setup by session manager.
    a. This should be plain and extensible with convenient methods for session management.
    b. At the end of the session, there's only one instance of the summarizer module that takes out the job of going through and running the defined logic along side having a track of state of application.
2. The Logic should be running asynchronous in the server for the user to get seamless user experience.

a. If the logic of the summarizer is kept on the client side, the end-user may face any performance related issues and also, its useless to keep the logic on the client side.

b. Making the logic run asynchronously, makes the logic run in the background of the server and helps improving the performance.

Class Diagram :



Interface :

```
// Summary Logic Module Interface
public interface ISummarizer() {


    /*
     * Function to compute the summary of the chat/discussion to present
in the dashboard
     * Retuns : string
     * This function returns a considerably long string which contains the
summary of the chat without
     * containing any metadata and information about the request sender.
     */
    //
    string GetSummary(Chat[] chats);


    // Function to save the summary of the whole meeting after completion
via persistance
    int SaveSummary(Chat[] chats);


}
```

Design Approach :

Creating the instance of Summarizer Module :

1. We'll use the Summarizer Factory to create the instance of the summarizer which would thus persist at the end of the session.
2. The Session Management Module takes the Summarizer under it, because, the session manager is the one in charge of acquiring the chat messages and other information and re-directing the data to different modules.
3. This design approach of coding style would supply an abstraction of internal summarizer class by not exposing any functionality to other modules as only its interface functions are essential and other methods/functions arent required for functionality.

Summarizer :

1. The Summary logic algorithm is contained within the Summarizer class, which can be a data-driven or pre-determined based on the requirements after the end of the discussion.

2. As we are running the summarizer module asynchronously in the background of the server, it wouldn’t bother the performance of the application, as the algorithm wouldn’t interrupt or a take a long time to complete the execution, keeping the other executions on hold.
3. This clears the road for better summarization of the chats, because the algorithm would be exposed to the entire discussion/chat while giving the summary a bigger picture.
4. The first method in the summarizer module is the GetSummary which takes in a array of individual chats, which is supplied by the session manager and returns a crip depiction of the chat that gets displayed/updated on the dashboard.
5. Down below the roots, the summary logic algorithm is designed to be a probabilistic algorithm, so it doesn’t require any dataset or prior information for the summarization of the discussion.
6. The computed summary returned to the session manager is passed to the persistence module for saving purposes.
7. In this designing approach, the summary logic entirely runs only on the background of server and not at all on the client side. The end-user gets the updated summary after processing on the server through an update in the UX and Session Manager.