Software Design Document

Pranav Patil 16010421076
Darshit Shah 16010421094

April 24, 2024

Aim

To prepare Software Design Document (SDD)

1 Introduction

1.1 Design Overview

The Chrome Sidebar Extension is designed to provide a convenient way to interact with and understand the Website/PDF, thus increasing the excitement to learn something with ease. It acts like a one on one tutor to explain something to the user in case they do not understand what is written on the Webpage/PDF.

1.2 Requirements Traceability Matrix

The requirements traceability matrix will show how each feature identified in the SRS is supported by the design components.

	CF	QG	SG
CFW: Chat Functionality with Website	X		
QGW: Question Generation for Website		X	
SGW: Summary Generation for Website		X	
CFP: Chat Functionality with uploaded PDFs	X		

2 System Architectural Design

2.1 Chosen System Architecture

The chosen system architecture will be a client-server architecture, with the client being the Chrome Sidebar Extension and the server being a web-based backend service. The significant component groupings will include the following:

- User Interface Component: This component will handle the user interface elements of the Chrome Sidebar Extension, including chat interfaces, summary generation interfaces, and settings interfaces.
- Natural Language Processing (NLP) Component: This component will utilize NLP algorithms and models to process user queries, extract relevant information from websites and PDFs, and generate summaries.
- Backend Service Component: This component will consist of the backend server responsible for processing requests from the Chrome Sidebar Extension, interacting with databases, and managing user sessions and authentication.
- Database Component: This component will store user preferences, chat histories, and any other relevant data required by the extension.

2.2 Discussion of Alternative Designs

Other design options explored include a full fledged website with the option to upload the URL or PDF on that website but was later on turned down since we wanted to make the application to be convenient for the user without them needing to open an entire website.

2.3 System Interface Description

The interfaces will include internal and external interfaces, such as:

- Internal Interfaces
  - Interfaces between the User Interface Component and the NLP Component for passing user queries and receiving processed data.
  - nterfaces between the NLP Component and the Backend Service Component for handling data processing and retrieval.
  - Interfaces between the Backend Service Component and the Database Component for storing and retrieving data.
- External Interfaces
  - The Chrome Sidebar Extension will interact with external websites and PDF documents through web scraping and API calls.
  - The Backend Service Component will interact with external services for authentication, payment processing, and any other required functionalities.

3 Detailed Description of Components

3.1 Chat Functionality

The chat functionality component will be responsible for allowing the users to chat with the Website/PDF in order to clear their doubts or find answers to their questions without needing to go through the entire document.

3.2 Question Generation

The question generation component will be responsible for generating the questions related to the content present in the website or PDF uploaded by the user to get them started.

3.3 Summary Generation

The summary generation component will be responsible for generating the summary of the entire website/PDF content in order to give the gist of what is present in that resource(website/PDF).

4 User Interface Design

4.1 Description of User Interface

The user interface is designed using a responsive web design approach, with a clean and user-friendly interface. The interface includes screens for user registration, Movies menu display and ticket booking.

4.1.1 Screen Images

4.1.2 Objects and Actions

- **Sidebar Extension:**
  - **Object:** Chat interface, Sidebar navigation
  - **Action:** Initiate conversation, navigate through sidebar options
  - **Registration screen:**
    - **Object:** Username, email, password fields
    - **Action:** Enter username, email, and password to create an account
  - **Login screen:**
    - **Object:** Username and password fields
    - **Action:** Enter username and password to log in
- **Conversation screen:**
  - **Object:** Chat window, Message input field
  - **Action:** Send and receive messages in real-time

5 System Architecture

The use case specification is provided separately in a use case document, using a standard use case template. The use cases include user registration, Movie menu display, Ticket Booking and user management.

Use Case Specification

Use Case ID	1		
Use Case Name	Chat with Website		
Created By	Darshit Shah	Last Updated By	Pranav Patil
Date Created	11th March 2024	Date Last Updated	17th March 2024

Primary Actors	Vishal Patil, Dheer Sheth
Secondary Actors	None
Description	This use case involves chatting with the website the user is currently on.
Trigger	User clicks on Read Website button
Preconditions	User needs to be on the website they want to chat with.
Postconditions	The website content is parsed and ready for question-answer
Normal Flows	• User clicks on Read Website button. • The website content is parsed and ready for question-answer. • The user asks questions to the website through this chrome sidebar extension.
Alternative Flows	None
Exceptions	Website has blocked html parsing
Includes	None
Priority	High
Frequency of Use	High
Business Rules	Should not answer anything that might be hurting or insensitive
Special Requirements	None
Open Issues	None
Assumptions	User has proper internet connection
Notes and Issues	None

Table 1: Caption

Use Case ID	2		
Use Case Name	Chat with PDF		
Created By	Pranav Patil	Last Updated By	Darshit Shah
Date Created	10th April 2024	Date Last Updated	19th April 2024

Primary Actors	Purav Sanghavi,Sakshaat Raut
Secondary Actors	None
Description	This use case involves chatting with the PDF.
Trigger	User clicks on Uplaod PDF button
Preconditions	User needs to have the PDF on their PC
Postconditions	The PDF content is parsed and ready for question-answer
Normal Flows	• User clicks on PDF upload button. • The PDF content is parsed and ready for question-answer. • The user asks questions to the PDF through this chrome sidebar extension.
Alternative Flows	None
Exceptions	PDF file too big to be parsed
Includes	None
Priority	High
Frequency of Use	High
Business Rules	Should not answer anything that might be hurting or insensitive
Special Requirements	None
Open Issues	None
Assumptions	User has proper internet connection
Notes and Issues	None

Table 2: Caption

6 Data Flow Specifications

6.1 Level 0 DFD with Description

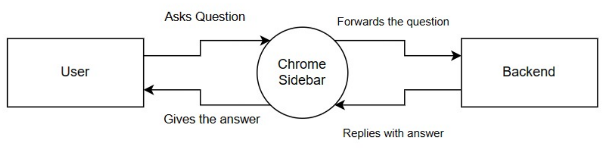

Figure 1: Level 0 DFD

6.2 Level 1 DFD with Description

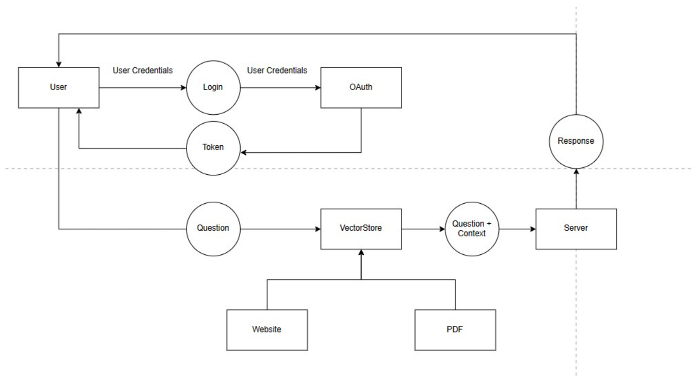

Figure 2: Enter Caption

Questions:

1.Explain Architecture design patterns and styles with examples.

Answer:

Architecture design patterns and styles are established principles and guidelines used to solve common architectural problems in software development. They provide a structured approach to design and organize software systems, promoting scalability, maintainability, and reusability. Here are some key architecture design patterns and styles along with examples:

1. Layered Architecture : - Description : In this style, the system is divided into multiple layers, with each layer responsible for a specific aspect of functionality. Communication typically occurs only between adjacent layers. - Example : A typical example is a web application with three layers: presentation layer (UI), business logic layer (backend logic), and data access layer (database interaction).

2. Client-Server Architecture : - Description : This pattern separates the client and server into two separate logical entities. Clients send requests to servers for data or services, and servers respond to these requests. - Example : Web applications follow the client-server architecture, where the web browser acts as the client, and the web server serves requested web pages and resources.

3. Model-View-Controller (MVC) : - Description : MVC separates an application into three interconnected components: Model (data and business logic), View (presentation layer), and Controller (handles user input and updates the model). - Example : Web frameworks like Ruby on Rails, Django, and ASP.NET MVC follow the MVC pattern to organize web applications.

4. Microservices Architecture : - Description : In this style, an application is composed of small, independently deployable services, each responsible for a specific business function. Communication between services typically occurs via APIs. - Example : Netflix, Amazon, and eBay have adopted microservices architecture to build large-scale, highly scalable systems.

5. Event-Driven Architecture (EDA) : - Description : EDA relies on the production, detection, and consumption of events. Components communicate asynchronously through events, enabling loose coupling and scalability. - Example : Message brokers like Apache Kafka and RabbitMQ are used in event-driven architectures to decouple components and enable communication through events.

6. Service-Oriented Architecture (SOA) : - Description : SOA is an architectural style where software components (services) communicate with each other over a network. Services are designed to be self-contained and have well-defined interfaces. - Example : Enterprise systems often adopt SOA to integrate disparate systems and expose functionalities as reusable services.

7. Hexagonal Architecture (Ports and Adapters) : - Description : Hexagonal Architecture decouples the application core from external concerns by placing ports (interfaces) at the boundary of the system. Adapters implement these ports, allowing the core to remain independent of external implementations. - Example : A banking system might have adapters for interacting with external payment gateways or messaging systems, while the core logic remains unchanged.

These architecture design patterns and styles provide a foundation for designing robust, scalable, and maintainable software systems. Choosing the right pattern depends on factors like the system's requirements, scalability needs, and development team's expertise.

Outcomes: CO3: Demonstrate requirements, modeling and design of a system.

Conclusion: After completion of this experiment we have succesdully prepared the Software Design Document for our project - The Chrome Sidebar.

References:

Books: Roger S. Pressman, Software Engineering: A practitioners Approach, 7th Edition, McGraw Hill, 2010.
Technical report on Guidelines for Documents Produced by Student Projects In Software Engineering based on IEEE standards
Timothy C. Lethbridge, Robert Laganiere " Object-Oriented Software Engineering – A practical software development using UML and Java", Second Edition, Tata McGraw-Hill, New Delhi,2004