Proteomics Chatbot

The **Proteomics Chatbot** is an advanced AI-powered application developed to simplify access to detailed protein information. Built using the **Llama3.2:1b** language model, the UniProt database, and state-of-the-art retrieval techniques to provide contextual and accurate answers to user queries. By combining advanced natural language processing capabilities and protein-specific knowledge, this chatbot bridges the gap between complex protein data and user-friendly interaction.

System Architecture

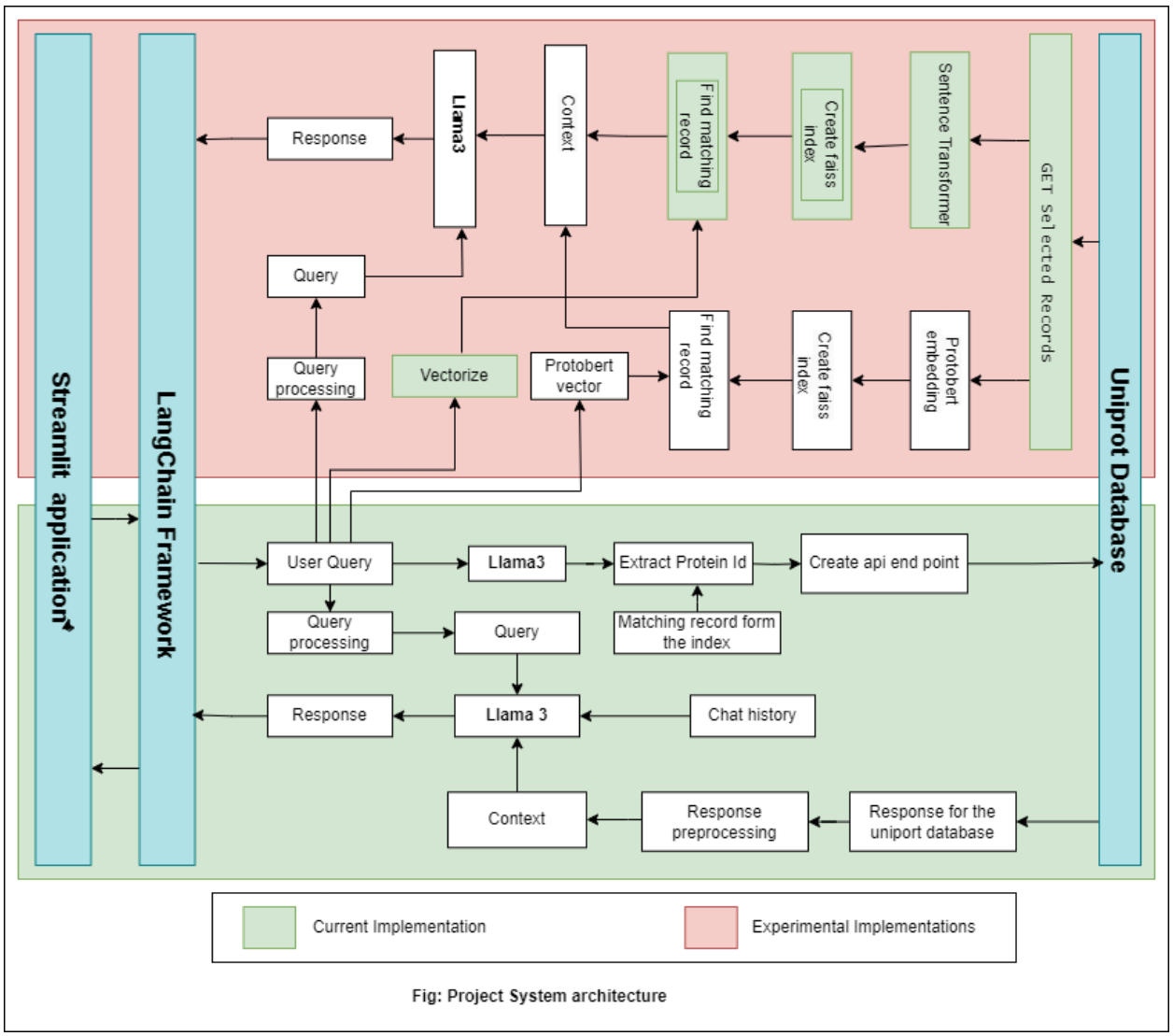


Fig: Project System architecture

1. **Frontend:** Built with **Streamlit**, enabling an intuitive user interface.

2. **Backend Processing:**
    - **Ollama Llama3.2:1b** for generating responses.
    - **FAISS** for efficient similarity search and retrieval.
    - **Sentence Transform**er for vector embedding
3. **Data Sources:** Uses the **UniProt REST API** and locally indexed CSV files.
4. **Agent Logic: LangChain** handles natural language queries.

Experimentations

Approach 1: SentenceTransformer for Embedding Entire Rows

We experimented with embedding all columns for each row using SentenceTransformer to create a FAISS index. Upon user input, the query was vectorized using the same model, and the most similar vector from the FAISS index was retrieved. The corresponding row served as the context for the LLM to answer the question. However, this approach often failed to match the correct protein entry, especially when using all columns for vectorization, as it struggled to capture the nuanced relationships between the fields.

Approach 2: ProtBERT Transformer for Protein-Specific Embedding

To address the shortcomings of generic embeddings, we utilized ProtBERT, a transformer model specialized for protein data, to embed entire rows and construct a FAISS index. While this improved embeddings for protein-specific fields, it still faced issues similar to the first approach, as it was unable to effectively handle multi-column relationships and retrieve the correct entry consistently.

Approach 3: Direct API Calls Using UniProt ID

We used Llama3.2 to extract the primaryAccession (UniProt ID) from the user's query. This ID was then used to fetch protein data directly from the UniProt API. Although this approach was highly accurate, it limited user flexibility, as queries without a specific UniProt ID were not supported.

Approach 4: Final Combined Approach: FAISS + API Retrieval

The final solution combines the strengths of the first and third approaches:

1. Extracted key fields (e.g., primaryAccession, uniProtkbId, proteinDescription) from the UniProt database and limited the dataset to 500 entries.
2. Embedded these fields using SentenceTransformer and built a FAISS index.
3. For user queries, the query is vectorized and matched against the FAISS index to retrieve the primaryAccession.
4. The retrieved primaryAccession is used to call the UniProt API, and the resulting data serves as the context for the LLM to generate a response.

This hybrid approach supports flexible querying with protein names, IDs, or descriptions while maintaining high accuracy in context retrieval.

Key Features

1. **Contextual Query Resolution**
    a. The chatbot processes natural language queries about proteins, including queries by name, UniProt ID (primaryAccession), or descriptions.
    b. It can retrieve information even for indirect or follow-up queries using a memory mechanism for conversational continuity.
2. **Flexible Data Retrieval**
    a. Integrates the UniProt REST API to fetch the latest protein data dynamically.
    b. Supports custom indexing for offline use, ensuring flexibility in data access.
3. **Enhanced User Interaction**
    a. Supports conversational context through a memory buffer, enabling fluid follow-up questions.
    b. Allows queries in both structured (with UniProt IDs) and unstructured formats (protein descriptions or names).
4. **Efficient Embedding and Retrieval**
    a. Incorporates FAISS indexing for fast and scalable retrieval of protein records.
    b. Employs pre-trained transformer models like ProtBERT and Sentence Transformers for embeddings tailored to protein data.

Technical Stack

- Language Model: Llama3.2:1b
- Embedding Models: SentenceTransformer (all-MiniLM-L6-v2), ProtBERT
- Database Integration: UniProt REST API
- Indexing: FAISS for fast similarity search
- Interface: Streamlit for the conversational chat interface

Project Setup

Prerequisites

- Python 3.8 or higher
- Ollama for running Llama3.2:1b
- Docker (optional)

Installation Guide

1. Clone the Repository

```
git clone https://github.com/Dipeshtripathi13/proteomics_chatbot.git
cd proteomics_chatbot
```

2. Set Up a Virtual Environment

```
python -m venv .venv
source .venv/bin/activate  # For Linux/MacOS
.venv\Scripts\activate     # For Windows
```

3. Install Dependencies

```
pip install -r requirements.txt
```

4. Install Ollama

Download Ollama from ollama.com and pull the Llama3.2 model:

```
ollama pull llama3.2:1b
ollama run llama3.2:1b
```

5. Run the Vectorization Script

Vectorize the UniProt data to build the FAISS index:

```
python vectorize_protoindex.py
```

Usage

1. Run the Streamlit Chatbot

```
streamlit run app.py
```

Navigate to `http://localhost:8501` to access the chatbot.

2. Example Query

- **Input:** `Give me the information about the protein:` Synaptonemal complex central element protein 3
- **Output:** Details about the specified protein, including its UniProt ID, primary accession, description and functions.

Project Structure

```
proteomics_chatbot/
│
├── app.py                          # Streamlit GUI for the chatbot
├── model_load.py                   # model loading
├── vectorize_protoindex.py         # encoding and indexing protein data
├── retrive_protoindex.py           # query-based retrieval from FAISS
├── requirements.txt                # List of dependencies
├── README.md                       # Project documentation
├── uniprot_data.csv                # Protein data file
├── response_process.py             # Preprocess logic for protein data
├── Dockerfile                     # Docker containerization instructions
├── get_selected_data.py            #extract records from the uniport
├── extract_index.py                   # To vectorize each records using
sentence Transformer
├── protein_info_vectors.index     # stored vector index of each of the
records
├── vectorizer.py                  # vector index of each of the records
├── documentation.pdf               # complete documentation
```

API Integration

The chatbot uses UniProt's REST API for live protein data retrieval. Example endpoint:

```
https://rest.uniprot.org/uniprotkb/{protein_id}
```

To get the records from the uniport database

```
https://rest.uniprot.org/uniprotkb/search?query=*&size=500&format=json
```

Containerization

1. Build Docker Image

```
docker build -t proteomics-chatbot .
```

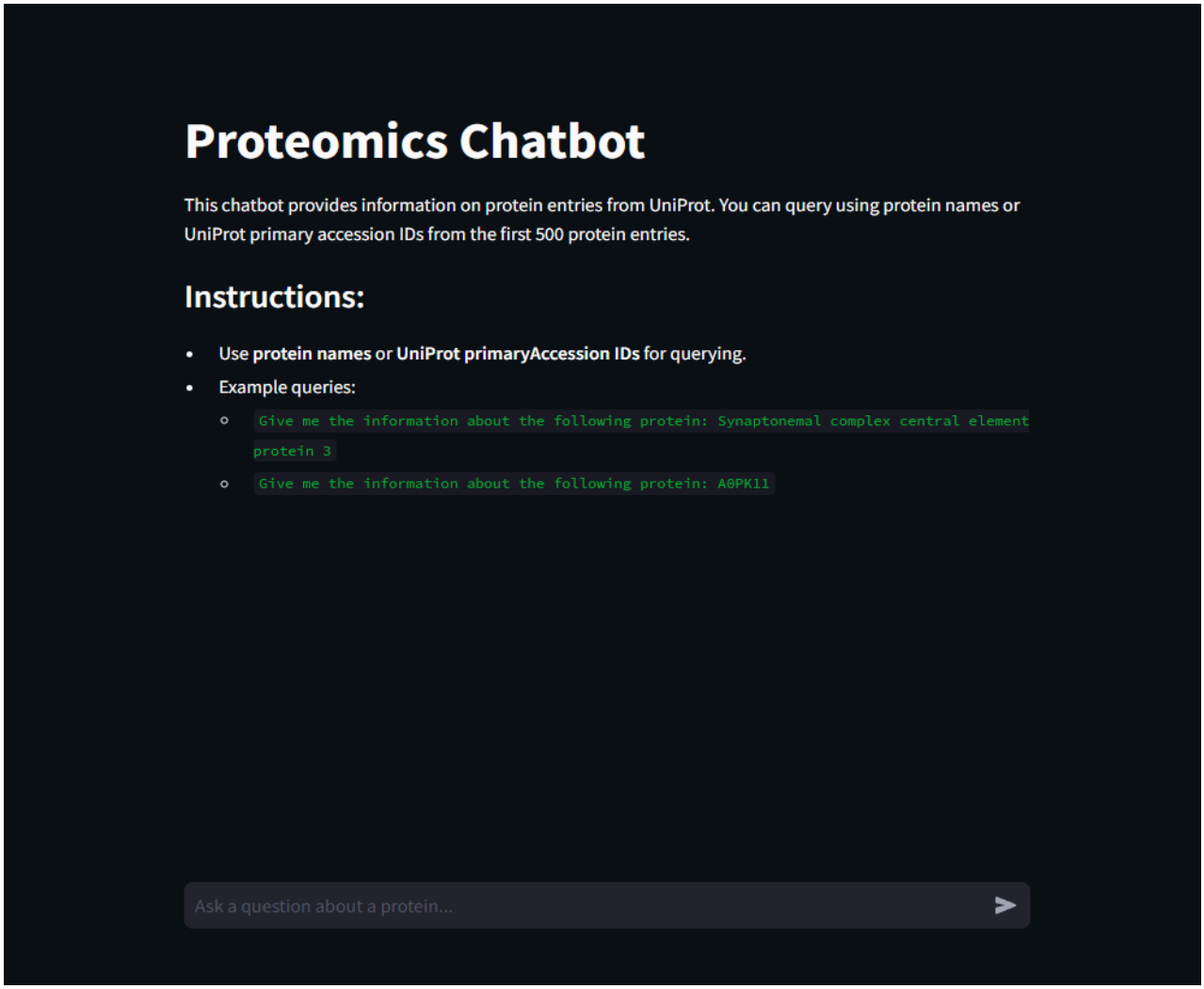
2. Run the Docker Container

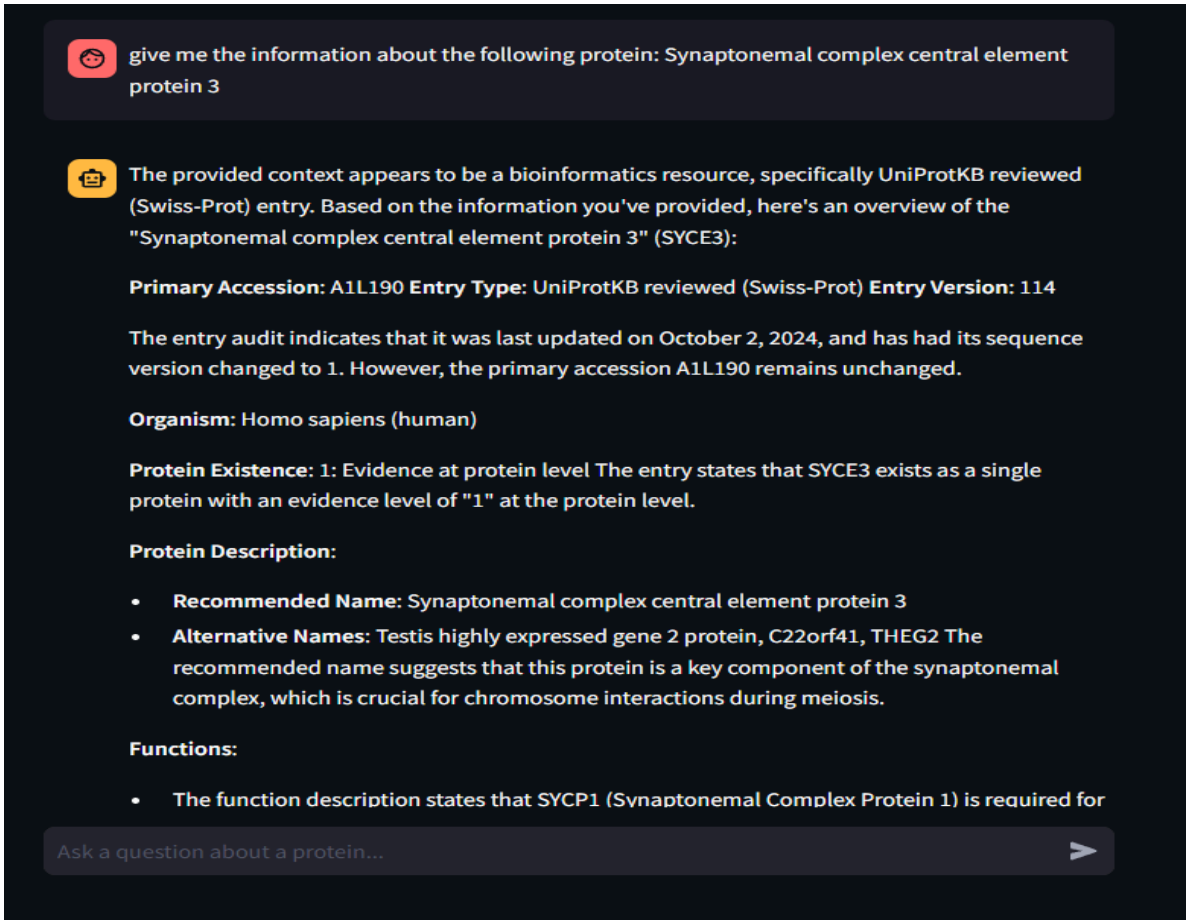
```
docker run -p 8501:8501 proteomics-chatbot
```

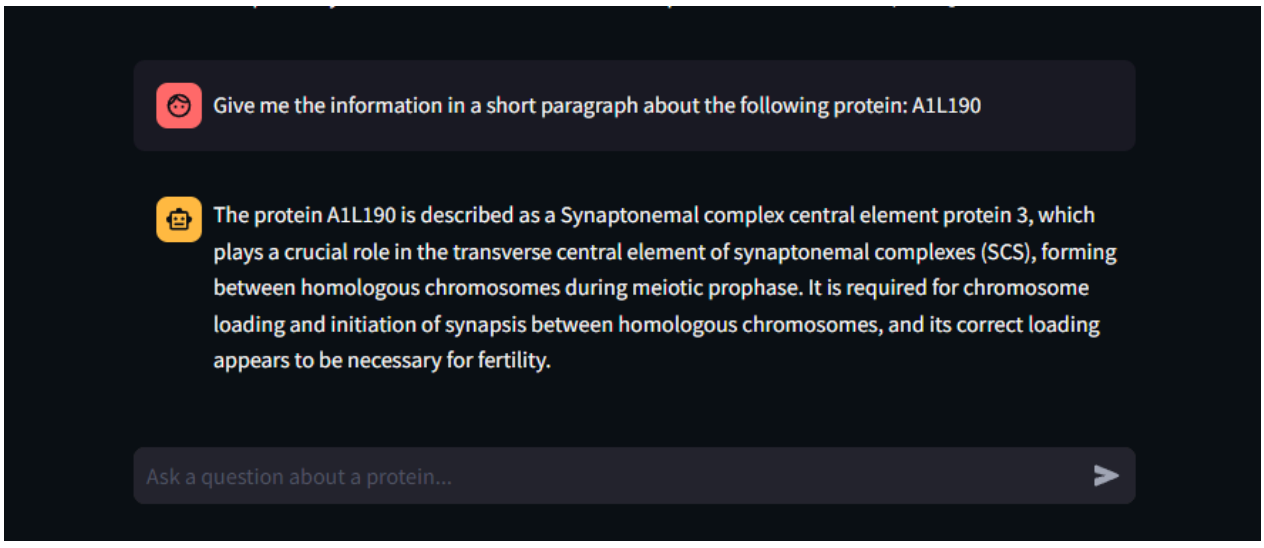
Examples

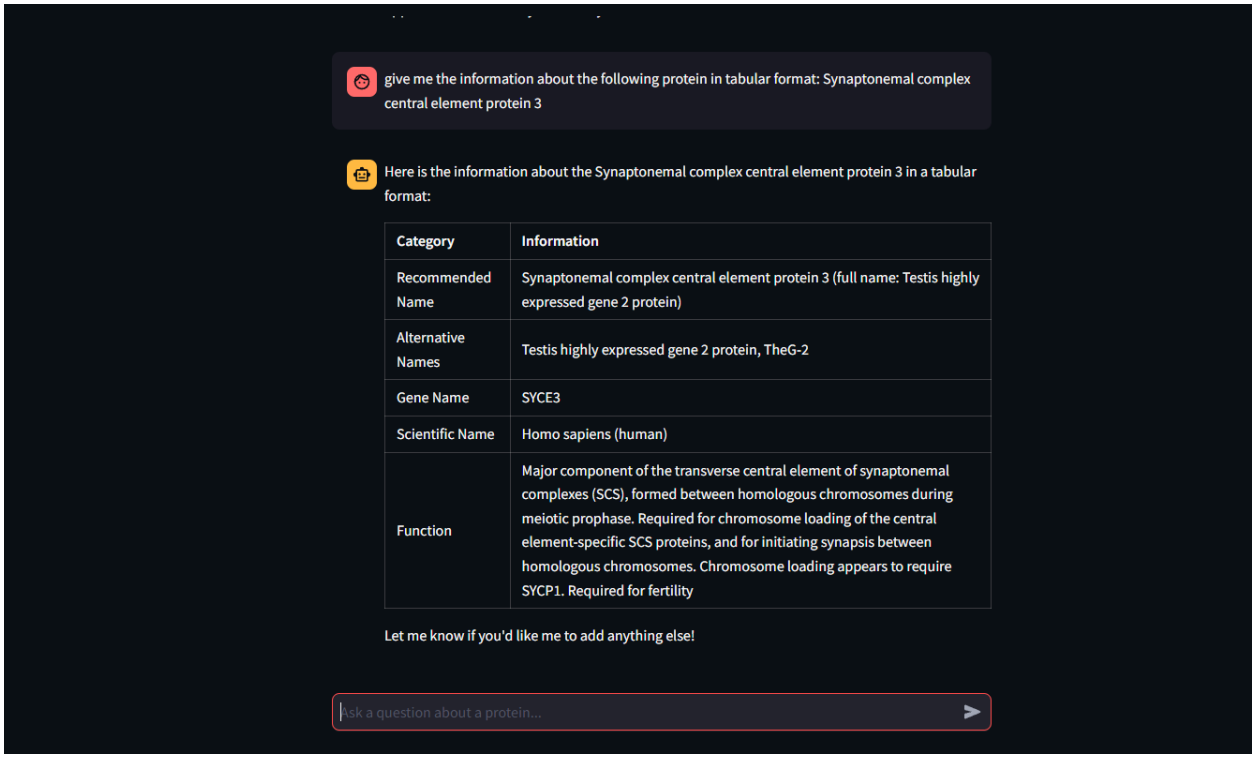
- **Query:** `Give me the information about the protein whose primaryAccession is A0JP26`
- **Result:** Protein details, including UniProt ID, primary accession, and description.

User Interface Screenshots




give me the information about the following protein: Synaptonemal complex central element protein 3
The provided context appears to be a bioinformatics resource, specifically UniProtKB reviewed (Swiss-Prot) entry. Based on the information you've provided, here's an overview of the "Synaptonemal complex central element protein 3" (SYCE3):
Primary Accession: A1L190 Entry Type: UniProtKB reviewed (Swiss-Prot) Entry Version: 114
The entry audit indicates that it was last updated on October 2, 2024, and has had its sequence version changed to 1. However, the primary accession A1L190 remains unchanged.
Organism: Homo sapiens (human)
Protein Existence: 1: Evidence at protein level The entry states that SYCE3 exists as a single protein with an evidence level of "1" at the protein level.
Protein Description:
Recommended Name: Synaptonemal complex central element protein 3
Alternative Names: Testis highly expressed gene 2 protein, C22orf41, THEG2 The recommended name suggests that this protein is a key component of the synaptonemal complex, which is crucial for chromosome interactions during meiosis.
Functions:
The function description states that SYCP1 (Synaptonemal Complex Protein 1) is required for
Ask a question about a protein...


Give me the information in a short paragraph about the following protein: A1L190
The protein A1L190 is described as a Synaptonemal complex central element protein 3, which plays a crucial role in the transverse central element of synaptonemal complexes (SCS), forming between homologous chromosomes during meiotic prophase. It is required for chromosome loading and initiation of synapsis between homologous chromosomes, and its correct loading appears to be necessary for fertility.
Ask a question about a protein...



Acknowledgments

- UniProt
- ProtTrans
- Streamlit
- FAISS
- Ollama

This documentation provides a comprehensive guide to understanding, setting up, and using the Proteomics Chatbot. For additional details, refer to the source code and accompanying comments.