# TERAH (The Epic Retirement AI Helper) - Feature Document

## 1.0 High Level Summary

This document outlines the core features and underlying technical architecture of the TERAH (The Epic Retirement AI Helper) project. Developed as a factual information platform, TERAH is meticulously engineered to provide Australians with precise, reliable information on retirement topics, strictly adhering to a "no financial advice" mandate. This report details the system's foundational data ingestion, intelligent context retrieval, a novel two-stage response validation process, and the final user-facing experience, including source attribution.

The project’s architecture is designed to manage the complexities of ingesting different data sources (text, spreadsheets, structured data) and transform them into a unified format for efficient AI processing. A key innovation is the two-stage model validation, which acts as a robust safeguard against misinformation and non-compliant advice, ensuring all outputs align with legal and ethical standards, particularly those set by Australian financial regulators.

## 2.0 Core Features and Technical Overview

### 2.1 Web Scraping and Data Ingestion

The cornerstone of TERAH's knowledge base is its ability to gather and process information from various authoritative sources. This feature leverages both automated web scraping and manual data gathering to create a comprehensive, fact-based corpus for the AI model.

#### **2.1.1 How It Works**

The data ingestion process is managed by the loadStaticData.js file, which contains specific logic for each data type. The system is designed to handle a diverse range of formats:

* **Text Files (.txt):** These are imported directly as raw strings, maintaining their original line breaks and formatting for full-text searchability.
* **CSV Files (.csv):** Raw CSV data is first imported and then parsed using the Papa Parse library. This transforms the raw text into a structured array of JavaScript objects, where each object represents a row in the original spreadsheet.
* **Excel Files (.xlsx):** The system fetches these files as binary data using an arrayBuffer, which is then processed by the XLSX library. This converts the Excel worksheets into structured JSON data, mirroring the spreadsheet's rows and columns as an array of objects.
* **JSON Files (.json):** These files are imported natively by JavaScript, preserving their original hierarchical structure.

This multi-faceted approach ensures that irrespective of the original file format, all data is transformed into a consistent, easily navigable JavaScript object. This structured format is critical for the AI model's subsequent processing, as it allows for efficient searching, filtering, and snippet extraction.

**Example Data Transformation:**

A raw text file like ATO.txt is ingested as a single, searchable string. A CSV file like dss-demographics-2021-sa2-june-2025.csv is transformed into an array of objects, with each object representing a row. This uniform object structure streamlines the models access to both textual and quantitative data.

### **2.2 Algorithmic Context Retrieval**

To ensure the AI's responses are accurate and efficient, TERAH employs a sophisticated algorithmic process to retrieve and present only the most relevant information to the model. This prevents the model from being overloaded with unnecessary data and significantly reduces the risk of factual "hallucinations."

#### **2.2.1 How It Works**

The system's intelligence lies in its ability to understand the user's query and match it to specific, relevant data points. This is not a simple keyword search but a multi-stage process:

1. **Pattern-Based Enhancement:** The application first analyzes the user's query against a set of predefined regular expressions. These patterns categorize the query by type (e.g., quantitative, temporal, procedural, financial). This initial classification provides a high-level understanding of the user's intent, helping the system narrow down the search.
2. **Intelligent Term Matching:** The system extracts key terms and phrases from the query. To account for typos and synonyms, a Levenshtein distance algorithm is used for fuzzy matching, ensuring that slight variations in spelling still lead to a match. This is crucial for matching user inputs to the specific, often jargon-filled, terms within government documents.
3. **Relevance Scoring and Snippet Selection:** Instead of feeding the AI entire documents, the system selects and sends only the most relevant "snippets." A scoring function evaluates each potential snippet based on the frequency of matching terms and phrases, with a higher weight given to phrases. This score is normalized by the snippet's length, similar to a simplified TF-IDF (Term Frequency-Inverse Document Frequency) approach, to ensure longer documents don't get an unfair advantage. The final output is a collection of the highest-scoring, non-overlapping text snippets, ready for ingestion.

This process is implemented across geminiChat.jsx and loadStaticData.js, ensuring that the AI is only fed the precise context it needs to formulate a factual and accurate response.

### **2.3 The Two-Stage Response Logic**

To enforce legal compliance and maintain the highest standard of accuracy, TERAH implements its two-stage AI response validation process. This is the project's most critical safeguard against the provision of personal financial advice.

#### **2.3.1 How It Works**

1. **Stage 1: Initial Response Generation:** The user's query, along with the curated snippets from the context retrieval process, is sent to the primary AI model. This model generates a comprehensive response based on the provided information.
2. **Stage 2: Independent Proofreading:** The initial response from Stage 1 is **not** immediately shown to the user. Instead, it is immediately routed to a **second, independent AI model instance**. This "proofreader" model operates with a strict set of system instructions, which are significantly different from the first. Its sole purpose is to audit the initial response for compliance.
   * **Strict Criteria:** The proofreader is explicitly instructed to only flag responses that contain personal financial advice (e.g., "you should invest in X") or clear factual errors.
   * **Conservative Approach:** It is instructed to be highly conservative in its corrections. It will not alter factual information, government rates, or dates, as these are considered safe and compliant.
   * **Decision:** The proofreader model returns one of two outcomes:
     + **"ACCEPTABLE":** If the response is compliant, it returns this keyword.
     + **Corrected Text:** If it finds a compliance issue, it will rewrite the response to omit the problematic parts while preserving the factual information.

This decision is made in the geminiChat.jsx file using a simple string comparison: if the proofreader's response is ACCEPTABLE, the original Stage 1 response is used. Otherwise, the corrected text from Stage 2 is used as the final output. This elegant and robust design provides a final layer of quality control and liability protection.

### **2.4 User-Facing Response Formatting**

The final output is designed for clarity, readability, and trustworthiness.

#### **2.4.1 How It Works**

1. **Dynamic Source Attribution:** The system uses referenceHighlighter.js to intelligently identify which parts of the final response are directly attributable to the ingested source data. This is not a manual process; the system compares the response's sentences and phrases against the original source text. A findMatchesInSentence algorithm identifies phrases that are also present in the source, and a lookup table (referenceMap.js) then maps the source filename to its original government URL (e.g., ATO, Services Australia). The final output is an HTML string where these phrases are wrapped in <a> tags with a special CSS class (terah-link-highlight) that makes them appear as orange hyperlinks.
2. **List and Paragraph Formatting:** The blockifyWithBullets function in referenceHighlighter.js processes the final text. It uses advanced regular expressions to automatically detect and reformat any dot points or numbered lists that may have been compressed into a single text block by the AI. It transforms these into properly spaced <ul> and <li> HTML elements, significantly enhancing readability. Other lines of text are formatted into <p> tags to create a clean, modern layout with appropriate spacing.

This feature ensures the final response is not only accurate but also visually appealing, scannable, and transparent by providing direct links back to the original source of the information.

## 3.0 User Story to Feature Mapping

The following table maps each feature to the specific user stories it addresses.

| **User Story ID** | **User Task** | **Features Supported** |
| --- | --- | --- |
| **1, 2, 4, 6, 9, 10** | Learn how much to save, understand how work affects savings, discover super options, etc. | **Algorithmic Context Retrieval:** This is the core engine that finds the specific financial information relevant to each user's query, whether it's about savings goals (US 1), part-time work (US 2), or super options (US 4). |
| **1, 2, 4, 6, 9, 10, 82, 83, 85** | All stories involving complex financial queries. | **Two-Stage Response Logic:** This feature is the primary safeguard for all user stories. It ensures that while the user is getting factual information on topics like savings, super options, and TTR strategies, the response never crosses into personal financial advice, addressing the core risk identified in the legal research. |
| **93** | Direct links to ATO/Services Australia pages. | **Source Highlighting and Hyperlink Creation:** This feature is a direct implementation of this user story, providing the user with direct, verifiable links back to the original source documents. This builds trust and allows users to self-verify the information. |
| **4, 6, 82** | Build a retirement plan, know if I can build a super, understand TTR strategies, etc. | **Data Ingestion and Transformation:** This is the foundational feature that makes the application possible. It is responsible for making all the disparate data sources (like the ATO.txt and ABS\_Retirement\_Comparison.xlsx) available and searchable for the algorithmic retrieval process. |
| **1, 2, 4, 6, 9, 10, 82, 83, 85, 93, 94, 95** | All stories requiring a clear, easy-to-read response. | **List and Paragraph Formatting:** This feature improves the user experience across all user stories by ensuring that complex topics, particularly those with lists or multiple points (like the TTR strategies in US 82), are presented clearly and legibly. |