Bilkent University
Department of Computer Engineering

Senior Design Project
Group T2503
VeriFact

Detailed Design Report

Orhun Ege Çelik - 22202321
Egehan Yıldız - 22203014
Alhassan Raad Jassim Al-Badri - 22201170
İrem Damla Karagöz - 22203691
Eray İşçi - 22201686

Supervisor: Sinem Sav
Innovation Expert: Mustafa Sakalsız
Instructors: İlker Burak Kurt, Mert Bıçakçı

11/03/2026

Software Architecture Document

1. Introduction

1.1 Purpose of the System

VeriFact is a real-time claim verification platform designed to support fact-based decision making during online meetings. The system integrates directly into Zoom meetings and analyzes spoken statements in real time. It transcribes speech, detects factual claims, retrieves relevant evidence from authorized documents using a Retrieval-Augmented Generation (RAG) pipeline, and evaluates whether the claim is supported, refuted, or cannot be verified. By providing real-time verification indicators and evidence citations, VeriFact helps reduce misinformation risks and encourages evidence-driven discussions during meetings.

1.2 Design Goals

The system is designed according to several key engineering goals:

- **Usability**
  The interface must integrate seamlessly into Zoom meetings and present verification results in a clear and non-intrusive way.
- **Performance**
  Since the system operates during live meetings, verification must occur within a few seconds to maintain meeting flow.
- **Reliability**
  The system must maintain stable performance during long meetings and continue operating even if certain components temporarily fail.
- **Marketability**
  VeriFact addresses a gap in current meeting tools by providing real-time factual verification instead of only transcription or summarization.
- **Extendibility**
  The architecture is modular so new verification models, retrieval mechanisms, or meeting platforms can be added easily.
- **Security**
  Documents and verification results must follow strict permission controls and encryption policies.
- **Scalability**
  The system must support multiple organizations and meetings simultaneously without performance degradation.
- **Maintainability**
  Components are separated into independent modules to simplify debugging, updates, and testing.

- **Flexibility**
  The architecture supports both permanent organizational documents and temporary meeting-specific documents.
- **Modularity**
  Key components such as transcription, claim detection, document retrieval, and verification operate as separate services.
- **Aesthetics**
  The UI follows Zoom design conventions to ensure a consistent and professional appearance.

1.3 Definitions, Acronyms, and Abbreviations

RAG (Retrieval-Augmented Generation)
A technique where retrieved documents are used as context for language model reasoning.

STT (Speech-to-Text)
Technology used to convert spoken audio into text.

NEI (Not Enough Information)
Verification category used when available documents cannot confirm or refute a claim.

Claim Detection
Process of identifying factual statements in transcribed speech that require verification.

Verification Pipeline
The system pipeline that retrieves evidence and determines whether a claim is supported or refuted.

Permanent Documents
Organization-level documents stored in the knowledge base for long-term verification.

Temporary Documents
Documents uploaded for a specific meeting and automatically deleted after the meeting ends.

1.4 Overview

The proposed system **VeriFact** is a modular, real-time claim verification infrastructure designed as a pipeline-oriented system. The system integrates directly into meetings using the Zoom SDK [1] . The architecture captures live audio for segmented transcription. The system subsequently identifies check-worthy factual claims, retrieves supporting evidence from both static and session-specific corpora using Retrieval-Augmented Generation (RAG), and cross-references these results to evaluate claim veracity [2]. This modular design ensures that each domain-specific component can scale independently while maintaining high system reliability.

In this document we describe the architecture of VeriFact, including the system structure, subsystem decomposition, and interactions between major components. It explains how the system captures meeting audio, detects claims, retrieves evidence from authorized document sources, and produces verification results in real time. Additionally, the document describes security mechanisms, data management strategies, subsystem services, and test procedures used to validate system functionality.

2. Current Software Architecture

Today, a significant portion of organizational communication and decision-making takes place in online meetings. Teams discuss project progress, financial results, strategic plans, and technical decisions through virtual collaboration platforms. Despite this shift toward digital meetings, organizations still lack a systematic way to verify factual claims while the discussion is happening. As a result, the verification process is mostly informal and delayed. Participants typically rely on the speaker's credibility, make personal notes for later checking, or verify information after the meeting has ended. By that point, however, incorrect statements may already have shaped the direction of the discussion or influenced decisions made during the meeting.

Most existing meeting-support tools are designed to improve meeting productivity rather than ensure factual accuracy. Platforms such as Otter.ai, Fireflies.ai, MeetGeek, Granola AI, and Webex AI Companion focus on features like automatic transcription, meeting summaries, action-item extraction, and conversation analytics [3] . Architecturally, these systems are centered around speech-to-text pipelines followed by summarization or insight generation modules. While these capabilities make it easier to document discussions and review them afterward, they do not attempt to evaluate whether statements made during the meeting are factually correct or supported by reliable sources.

Some other tools offer capabilities that partially intersect with this problem space but still fail to address the core challenge of real-time verification. For example, Cluely provides AI-driven suggestions and coaching to help users respond more effectively during conversations. However, its design prioritizes improving individual communication performance rather than validating the truthfulness of claims. In addition, such systems may raise ethical concerns because they can influence how participants present information rather than ensuring that the information itself is accurate. Similarly, enterprise knowledge platforms like Glean enable users to search across internal company documents and data repositories. Although this makes it easier to locate information, the system operates primarily as a search interface and does not actively monitor discussions or verify claims as they occur.

Because of these limitations, the current ecosystem of meeting technologies lacks an architecture that integrates real-time speech processing with factual verification mechanisms. Existing solutions either document conversations or assist participants in responding more effectively, but none function as a real-time fact-checking layer embedded within meetings. This leaves a clear gap in the technological landscape. Systems like VeriFact aim to address this gap by combining transcription, claim detection, document retrieval, and automated verification within a single pipeline that operates during live meetings. By verifying statements against trusted organizational documents and presenting evidence-backed results in real time, such systems can help reduce misinformation and support more informed decision-making during collaborative discussions.

3. Proposed Software Architecture

3.1 Overview

The proposed system presents VeriFact as a real-time claim verification infrastructure that operates directly inside Zoom meetings. The system is integrated into the Zoom environment through a meeting bot that joins the meeting as a participant and continuously monitors the conversation. This bot serves as the connection point between the meeting platform and the verification system, allowing the software to capture audio, analyze spoken content, and deliver verification feedback to meeting participants while the discussion is taking place.

The overall architecture follows a pipeline-based processing approach in which several components work together sequentially to transform raw meeting audio into verified claim outputs. The process begins with the Zoom bot capturing the live audio stream of the meeting. This audio stream is then forwarded to a speech processing component that converts spoken language into structured transcript segments. Each segment of text is associated with a timestamp and speaker identity so that the system maintains a coherent and traceable representation of the conversation as it unfolds.

After the speech is converted into text, the transcript is analyzed by a claim detection module. The goal of this component is to identify sentences that contain factual statements that may require verification. Not every spoken sentence needs to be analyzed for correctness, since many statements in meetings consist of opinions, informal discussion, or conversational filler. Therefore, the claim detection component uses a trained classification model to filter the transcript and select only those statements that are likely to represent verifiable claims. This filtering step allows the system to focus its verification resources on meaningful statements while avoiding unnecessary computation.

Once a claim has been detected, the verification pipeline begins the process of retrieving relevant evidence from the document repository. This step relies on retrieval augmented reasoning in which the system searches for semantically similar passages in stored documents. The search is performed across two categories of documents. The first category consists of permanent organizational documents that remain available across meetings and represent long-term knowledge for the organization. The second category consists of temporary documents that are uploaded specifically for the current meeting session. These temporary materials may include drafts, presentations, reports, or other documents that are relevant only for the ongoing discussion.

The system retrieves the most relevant document passages and evaluates how they relate to the detected claim. The verification component analyzes the relationship between the claim and the retrieved evidence and determines whether the evidence supports the claim, contradicts it, or does not provide enough information to reach a clear conclusion. The result of this evaluation is then communicated back to meeting participants through the Zoom interface. Claims are displayed together with visual indicators and evidence excerpts so that users can quickly understand the verification outcome and inspect the supporting documents if necessary.

3.2 Subsystem Decomposition

3.2.1 Diagram Overview

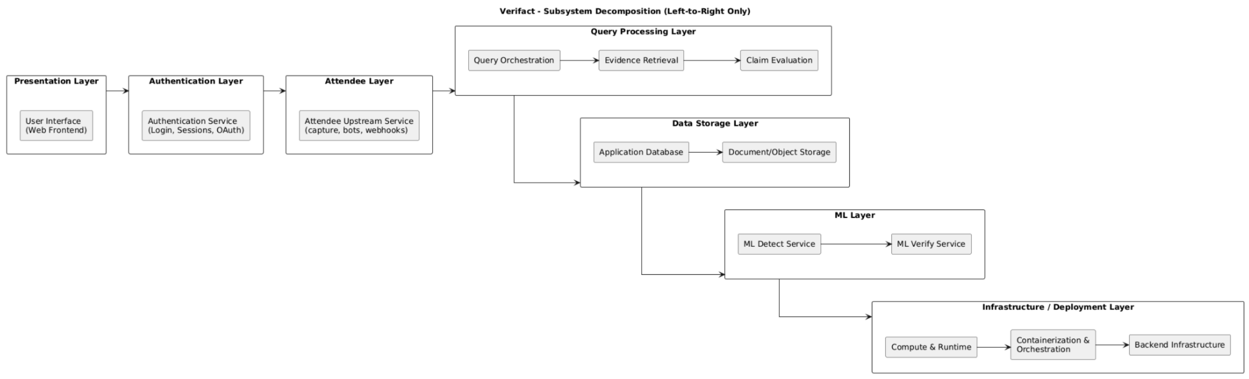

Figure 1: Subsystem Decomposition Diagram of Verifact.

3.2.2 Presentation Layer (Web Frontend)

- **Main Responsibilities:** Runs as a Zoom App UI, initializes the Zoom SDK, establishes a secure backend session, and renders the user experience (either a lobby or the in-meeting view).
- **Key Components:** The core application logic handles the bootstrap of the Zoom SDK and session setup. It renders the main views, for in-meeting and lobby view. A bot status client manages real-time UI updates via WebSocket.
- **Interactions:** Communicates with the Backend API over HTTPS (for authentication, meetings, and documents) and consumes real-time updates via WebSocket events.

3.2.3 Authentication Layer (Backend API Auth)

- **Main Responsibilities:** Converts Zoom app context tokens into VeriFact session JSON Web Tokens (JWTs) and enforces authentication across all API routes.
- **Key Components:** Authentication routes validate the Zoom token, persist user data, and issue session JWTs. Authentication middleware enforces token validity on subsequent requests.
- **Interactions:** The Presentation Layer calls this layer first to receive a session JWT,

which is then used to authenticate all subsequent API calls.

3.2.4 Attendee Layer (Attendee Upstream)

- **Main Responsibilities:** A full-featured Django service responsible for managing "bots" that join meetings to stream transcripts and events.
- **Key Components:** The Django project includes applications for bots. Backend API client wrappers facilitate interaction for creating, getting, and commanding the bot.
- **Interactions:** The Query Processing Layer calls the Attendee's REST API to manage bots and retrieve normalized transcripts and participant events.

3.2.5 Query Processing Layer (Backend API Business Logic + Worker)

- **Main Responsibilities:** Manages high-level operations for meetings, documents, and organizations. It also queues and orchestrates machine learning jobs for claim detection, verification, and document indexing.
- **Key Components:** The HTTP API surface defines routes for meetings, documents, and organizations. Services orchestrate repositories, storage, and message queues. A dedicated Worker configures and executes background jobs using BullMQ for claim processing and indexing. Worker libraries manage communication with the ML services and database updates.
- **Interactions:** The Backend API enqueues jobs to Redis queues, which are consumed by the Worker. The Worker then interacts with the ML Layer and Data Storage Layer before updating the database and notifying the Presentation Layer (e.g., via WebSocket).

3.2.6 Data Storage Layer (Database and Object Storage)

- **Main Responsibilities:** Persists all essential data, including users, organizations, meetings, claims, documents, and transcripts. It also stores large binary objects (blobs) like recordings.
- **Key Components:** SQL migrations define the relational database schema. Repositories provide typed CRUD (Create, Read, Update, Delete) operations. Object storage (S3/Azure compatible) is used for document files and recordings. The Attendee Layer utilizes the Django ORM for its data models.
- **Interactions:** The Query Processing and Attendee Layers call typed repositories/ORMs to execute data operations. Binary files and large documents are written to object storage, with references (IDs/URLs) stored in the primary database.

3.2.7 ML Layer (Machine Learning Runtime)

- **Main Responsibilities:** Executes machine learning inference for core system functions: claim detection and claim verification/fact-checking.
- **Key Components:** The ML Detect service exposes an HTTP endpoint to classify transcript segments as claims. The ML Verify component contains the RAG

implementation to score evidence and classify a claim's stance (Supported, Refuted, or Not Enough Information).
- **Interactions:** The Backend Worker calls these services over HTTP to receive classification and verification results, which are then used to update the persistent data stores.

3.2.8 Infrastructure / Deployment Layer

- **Main Responsibilities:** Provides the foundational network, containerization, and development tooling for the entire system, ensuring all services can run reliably.
- **Key Components:** Includes SQL scripts for database schema management, configuration files for tunneling, Docker files for containerizing services (ML and STT), and environment/queue configuration glue within the applications.
- **Interactions:** This layer is the underlying environment on which all other layers are deployed. It ensures the DB, Redis, ML services, and application backends are correctly configured and exposed to each other and to external services like Zoom.

3.3 Persistent Data Management

VeriFact requires persistent data management in order to store organizational documents, document embeddings, meeting metadata, and verification results in a reliable and scalable manner. Since the system continuously processes documents and verification outputs across multiple meetings and organizations, the architecture includes a structured storage model that separates different types of data according to their lifecycle and purpose.

The system manages two primary categories of documents: **permanent documents** and **temporary meeting documents**. Permanent documents represent long term organizational knowledge such as policies, reports, technical documentation, or financial records. These documents remain stored in the system across multiple meetings and serve as the main knowledge base for claim verification. Temporary documents, on the other hand, are uploaded specifically for a single meeting session. These may include draft materials, presentations, or working documents relevant only to that meeting. Temporary documents are automatically removed after the meeting ends to ensure that sensitive or context specific information does not remain accessible in future sessions.

When a document is uploaded to the system, it undergoes a preprocessing pipeline before being stored. The file is first parsed to extract textual content. The extracted text is then divided into smaller chunks to improve retrieval accuracy during the verification stage. Each chunk is converted into a vector embedding using a semantic embedding model. These embeddings allow the system to perform semantic similarity searches when retrieving evidence related to detected claims.

The system stores the **original document files and associated metadata** in an object storage layer that follows S3-compatible storage conventions. This storage layer provides scalable and durable storage for large document collections while supporting lifecycle policies for document retention and deletion. In addition to the file storage layer, the system maintains a **vector database** that stores the embeddings generated from document text. The vector database enables efficient semantic search over document content and allows the verification pipeline to retrieve relevant passages within milliseconds.

To support multi-organization usage, the storage architecture follows a **tenant isolation model**. Each organization has its own logical storage namespace, ensuring that documents and verification data belonging to one organization cannot be accessed by another. Organization identifiers are attached to document metadata, embeddings, and verification records so that retrieval queries are always restricted to the correct organizational context.

The system also maintains additional persistent data related to system operation. This includes user accounts, organization membership information, meeting sessions, document metadata, and verification results. Verification outputs such as claim classifications, confidence scores, and evidence references are stored so that they can be reviewed after the meeting. These records support post-meeting analysis, transcript review, and export features.

To ensure data integrity and maintainability, document operations such as upload, deletion, or modification are recorded in audit logs. These logs provide traceability for document lifecycle events and help maintain compliance with security and organizational policies.

Overall, the persistent data management design enables VeriFact to efficiently handle large document collections, support semantic retrieval for claim verification, maintain secure organization-level data separation, and preserve verification results for later analysis while automatically removing temporary meeting data when it is no longer required.

3.4 Access Control and Security

Access control and security are fundamental components of the VeriFact system because the platform processes potentially sensitive organizational documents and meeting transcripts. The system implements a role-based access control (RBAC) model to ensure that only authorized users can access specific documents, evidence sources, and system functions. Users are assigned roles within an organization, such as Owner, Admin, or Member, and each role determines the level of permissions available during meetings and document management operations. This design ensures that sensitive information is only accessible to users with the appropriate authorization level.

The permission model governs several actions within the system, including uploading documents, deleting documents, viewing verification evidence, and accessing confidential files. For example, document upload and management operations are restricted to users with elevated privileges, such as Owners or Admins, while Members may only view information that has been explicitly shared with them. During meetings, the system dynamically enforces these permissions when displaying verification results and evidence snippets. If a claim is verified using a document that requires higher access privileges, users without sufficient permissions will see the verification status but will not be able to open the underlying source document. This prevents unauthorized access while still maintaining transparency about the verification outcome. The hierarchical distribution of these permissions and the specific operational limits of each role are detailed in the following table:

Table 1: RBAC Matrix and Operational Permissions

Action	**Owner**	**Admin**	**Member**	**Participant**
1- Create organization	Allowed	Not allowed	Not allowed	Not allowed
2-Create invite link to organizations	Allowed	Allowed	Allowed	Not allowed
3- Init application and invite BOT to meeting	Allowed	Allowed	Not allowed	Not allowed
4- Set which permanent docs will be active in meeting	Allowed	Allowed	Not allowed	Not allowed
5- Change user permissions	Allowed	Allowed	Not allowed	Not allowed
6- Upload permanent documents	Allowed	Allowed	Not allowed	Not allowed
7- Delete permanent documents	Allowed	Allowed	Not allowed	Not allowed
8- Upload temporary documents	Allowed	Allowed	Not allowed	Not allowed
9- Delete temporary docs	Allowed	Allowed	Not allowed	Not allowed
10- Ask question / Q&A mode	Allowed	Allowed	Allowed	Not allowed
11- Inspect evidence sources (That is, it displays a small proof description indicating refuted or supported, along with the evidence source document name.)	Allowed	Allowed	if document permission states **‘member**’. If permission is admin: member sees only the final verdict, not source and proof details.	Not allowed

12- Open proof documents	Allowed	Allowed	if document permission states **‘member**’. If permission is admin: member may not open the doc	Not allowed
13- View verdicts(support/refutes)	Allowed	Allowed	Allowed	Not allowed
14- See transcription	Allowed	Allowed	Allowed	Allowed
15- Post meeting view (with full details, proof documents and such) This is the same logic with in meeting view since every permission level is embedded inside docs, same permission checks will be applied here and the post meeting summary will be displayed according to those document levels.	Allowed	Allowed	if document permission states **‘member**’. If permission is admin: member sees only the final verdict, not source and proof details. if document permission states **‘member**’. If permission is admin: member may not open the doc	Not allowed

4. Subsystem Services

This section defines the specific functional services provided by each subsystem. These services represent the operational interfaces through which components interact to process meetings and verify claims.

4.1 Live Meeting Services

The Attendee and Presentation layers collaborate to provide real-time interaction within the Zoom environment.

- **Stream Capture Service:** Operated by the Attendee Layer, this service joins Zoom meetings as a digital participant to capture multi speaker audio and convert it into a normalized text transcript.
- **UI Synchronization Service:** Provided by the Presentation Layer, this service maintains real-time state between backend processing and the user’s Zoom side panel, ensuring claims appear as they are detected.

- **Session Handshake Service:** Validates the Zoom App context to establish a secure, authenticated bridge between the Zoom client and the VeriFact infrastructure.

4.2 Knowledge and Document Services

The Data Storage and Query Processing layers manage the lifecycle and retrieval of organizational data.

- **Ingestion and Chunking Service:** A background process that parses uploaded documents, strips formatting, and breaks text into semantically meaningful segments for the vector database.
- **Semantic Indexing Service:** Generates high dimensional vector embeddings for document chunks, allowing the system to perform conceptual searches rather than simple keyword matching.
- **Lifecycle Management Service:** Automatically handles the removal of temporary meeting documents, ensuring meeting specific data is purged once the session is marked as closed.

4.3 Verification and Inference Services

The ML Layer provides the core intelligence required to analyze meeting content.

- **Claim Identification Service:** An NLP service that monitors incoming transcript buffers to identify checkable statements/sentences containing factual assertions rather than opinions or greetings.
- **Contextual Retrieval Service:** When a claim is identified, this service queries the vector database to retrieve the most relevant snippets of evidence from the organization's knowledge base.
- **Stance Classification Service:** Analyzes the relationship between a detected claim and retrieved evidence to output a final status of Supported, Refuted, or Not Enough Information.

4.4 Administrative and Security Services

The Authentication and Infrastructure layers ensure the system remains secure and performant.

- **Token Exchange Service:** Converts short lived Zoom OAuth tokens into internal JWTs, mapping Zoom user IDs to internal organization roles.
- **Tenant Isolation Service:** A security service that injects organization identifiers into every database query and storage request to prevent cross tenant data leaks.
- **Job Orchestration Service:** Utilizes BullMQ to manage the priority and retries of machine learning tasks, ensuring high latency verification does not block the real time transcript flow.

5. Test Cases

5.1 Functional Test Cases

5.1.1 Test Case Scenario 1 (Verification - Evidence Supports Claim)

Table 2: Test Case Scenario

Test ID	TC-01	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case is to verify that the system correctly detects a factual claim, retrieves supporting evidence, and marks the claim as "Supported" with a visible citation.				
Steps	1. Ensure a Permanent Document exists that confirms the statement "The Q1 marketing budget is $100,000" and has a high confidence score. 2. Start a meeting with VeriFact enabled. 3. A Participant makes the claim: "The Q1 marketing budget is $100,000." 4. Observe the transcript for real-time verification status. 5. Verify that a citation link is generated and that the Member can click it to view the source.				
Expected	The system detects the claim, retrieves the document section, classifies the verdict as "Supported," and displays a green "Supported" badge next to the transcript line, along with a citation link.				
Date-Result	5 March 2026 - PASS. The claim was correctly supported and cited.				

5.1.2 Test Case Scenario 2 (Verification - Evidence Refutes Claim)

Table 3: Test Case Scenario

Test ID	TC-02	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case is to verify that the system correctly detects a factual claim, retrieves contradictory evidence, and marks the claim as "Refuted."				
Steps	1. Ensure a Permanent Document (e.g., "Project Charter") states the project deadline is "November 30th." 2. Start a meeting with VeriFact enabled. 3. A Participant makes the claim: "The project deadline is set for December 30th."				

	4. Observe the transcript for real-time verification status. 5. Verify that the conflicting evidence snippet is accessible via the citation.
Expected	The system detects the claim, identifies a logical conflict with stored data, classifies the verdict as "Refuted," and displays a red "Refuted" warning/badge, logging the correction and providing a citation link.
Date-Result	5 March 2026 - PASS. The claim was correctly refuted and conflicting evidence was provided.

5.1.3 Test Case Scenario 3 (Verification - Evidence Not Found / NEI)

Table 4: Test Case Scenario

Test ID	TC-03	**Category**	**Functional**	**Severity**	**Medium**
Objective	This test case is to verify that the system correctly handles factual claims when no relevant or sufficiently confident evidence can be retrieved (NEI).				
Steps	1. Start a meeting with VeriFact enabled. 2. A Participant makes a niche claim about an internal detail not contained in any uploaded documents (e.g., "The latest server configuration is running Linux kernel version 6.5"). 3. Allow the system to search the authorized document embeddings. 4. Observe the transcript for the verification outcome.				
Expected	The search returns low relevance scores (below the confidence threshold). The system classifies the verdict as "Not Enough Information" (NEI) and displays a neutral/gray "Unverified" or "NEI" indicator without generating a citation link.				
Date-Result	5 March 2026 - PASS. The system returned NEI when no evidence was found.				

5.1.4 Test Case Scenario 4 (Permission-Based Evidence Inspection)

Table 5: Test Case Scenario

Test ID	TC-04	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case verifies that a standard Member is correctly denied access to view the source document if the supporting evidence is restricted to Admin-Only permissions.				
Steps	1. A claim is verified (Supported or Refuted) using a document with "Admin-Only" permissions.				

	2. A standard Member (not Admin/Owner) clicks the citation link. 3. The system checks the user's role against the document's permission level. 4. Observe the system's response.
Expected	The system identifies the user as lacking permission, denies the request to open the full document text, and displays a "Restricted Access" toast notification. Security of the underlying document is maintained.
Date-Result	11 March 2026 - PENDING.

5.1.5 Test Case Scenario 5 (Organization Onboarding)

Table 6: Test Case Scenario

Test ID	TC-05	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case verifies the successful creation, provisioning, and isolation of a new Organization environment.				
Steps	1. The Owner logs into the web portal for the first time. 2. The Owner initiates "Create Organization". 3. The Owner uploads the first set of "Permanent Documents" and selects the default accessibility level. 4. Verify that a secure, isolated tenant is provisioned for this Org.				
Expected	The Organization is active. The System provisions a secure, isolated tenant (database shard) and is ready to accept meeting connections from this Org's hosts.				
Date-Result	11 March 2026 - PENDING.				

5.1.6 Test Case Scenario 6 (Initializing Meeting with Organization Context)

Table 7: Test Case Scenario

Test ID	TC-06	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case verifies the secure initialization of a meeting, ensuring the bot loads only the inviting organization's context.				
Steps	1. A Host belonging to a valid Organization invites the VeriFact Bot into a Zoom meeting.				

	2. The System identifies the Host's Organization ID. 3. The System loads *only* the specific "Organization Context" (Permanent and relevant Temporary Documents). 4. Observe the Bot's "Ready" message.
Expected	The Bot joins and posts a "Ready" message: "VeriFact is active for [Organization Name]." Data strictness is enforced, preventing access to documents from any other Organization.
Date-Result	11 March 2026 - PENDING.

5.1.7 Test Case Scenario 7 (Updating the Permanent Knowledge Base)

Table 8: Test Case Scenario

Test ID	TC-07	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case is to verify the successful update of an outdated permanent document, ensuring that the old document's vector embeddings are deleted and replaced with new ones.				
Steps	1. Ensure a document named “HR Policy 2024” currently exists in the Permanent Document set. 2. Admin navigates to the Permanent Documents management console. 3. Admin selects “HR Policy 2024” and chooses “Update/Replace”. 4. Admin uploads the new “HR Policy 2025” file. 5. Verify the system deletes old vector embeddings and generates/stores new embeddings for “HR Policy 2025”.				
Expected	The system deletes the vector embeddings associated with the old 2024 version. The system generates and stores new embeddings for the 2025 version. Future claims/queries will be verified against the 2025 policy only.				
Date-Result	11 March 2026 - PENDING.				

5.1.8 Test Case Scenario 8 (Uploading and Using Temporary Meeting Documents)

Table 9: Test Case Scenario

Test ID	TC-08	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case is to verify that Admin/Owner can upload temporary, meeting-specific documents and that these documents are immediately				

	indexed and used for real-time claim verification alongside permanent documents.
Steps	1. Admin or Owner initializes a meeting context. 2. Admin uploads temporary documents intended for the current meeting and sets permissions. 3. The system parses and indexes the documents into the knowledge base. 4. During the meeting, a Participant makes a claim verifiable only by the newly uploaded temporary document. 5. Verify the claim is successfully verified using evidence from the temporary document, and a citation is provided.
Expected	Temporary documents are successfully indexed and made available. The claim is verified against both temporary and permanent knowledge bases, and a green “Supported” badge with a citation to the temporary file is displayed.
Date-Result	11 March 2026 - PENDING.

5.1.9 Test Case Scenario 9 (Post-Meeting Cleanup of Temporary Documents)

Table 10: Test Case Scenario

Test ID	TC-09	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case is to verify that temporary documents and their embeddings are automatically deleted after the meeting ends.				
Steps	6. Upload one or more documents as **Temporary Documents** during a meeting. 7. Start a meeting with VeriFact enabled and ensure the temporary documents are available for verification. 8. Generate a claim that uses the temporary document as evidence. 9. End the meeting so that Zoom sends the **Meeting Ended** signal to the system. 10. Check the document storage and vector database to verify whether the temporary document still exists.				
Expected	The system identifies documents tagged as temporary for the meeting, deletes the document files and their vector embeddings, logs the deletion event for				

	auditing, and retains only the transcript verification logs without linking to the deleted documents.
Date-Result	2 March 2026 - PASS. Temporary documents were successfully deleted after meeting end signal.

5.1.10 Test Case Scenario 10 (Deleting Permanent Documents)

Table 11: Test Case Scenario 10

Test ID	TC-10	**Category**	**Functional**	**Severity**	**Medium**
Objective	This test case is to verify that an Admin can permanently delete an obsolete document from the organization's knowledge base.				
Steps	1. Log into the VeriFact system as an **Admin** user. 2. Navigate to the **Permanent Documents dashboard**. 3. Select an existing document (e.g., *Project X_2020.pdf*). 4. Click the **Delete** option and confirm the deletion request. 5. Check the document repository and vector database to verify that the document has been removed.				
Expected	The system permanently deletes the selected document and its associated vector embeddings from the knowledge base so that it cannot be retrieved or used in future claim verification.				
Date-Result	2 March 2026 - PASS. The selected permanent document and embeddings were removed successfully.				

5.1.11 Test Case Scenario 11 (Filtering Non-Factual Statements)

Table 12: Test Case Scenario 11

Test ID	TC-11	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case is to verify that the system correctly identifies subjective statements and prevents unnecessary verification processing.				
Steps	1. Start a Zoom meeting with the VeriFact bot enabled. 2. A participant says a subjective sentence such as "I feel like our team synergy is improving drastically." 3. Allow the system to transcribe the spoken sentence.				

	4. Observe how the claim detection module processes the sentence. 5. Check whether the system triggers claim verification or skips the process.
Expected	The system classifies the sentence as a subjective or non-factual statement, skips the evidence retrieval and verification steps, and displays the sentence in the transcript without any verification badge.
Date-Result	20 February 2026 - PASS. The claim detection system worked as expected, filtering out the subjective statements.

5.1.12 Test Case Scenario 12 (Correction of Temporary Meeting Materials)

Table 13: Test Case Scenario 12

Test ID	TC-12	**Category**	**Functional**	**Severity**	**Low**
Objective	This test case is to verify that an Admin can delete or replace an incorrect temporary document before the meeting ends.				
Steps	1. Upload a temporary document (e.g., *Budget_v1.pdf*) before the meeting starts. 2. Start the meeting with VeriFact enabled. 3. Navigate to the document management panel. 4. Select the incorrect temporary document and choose **Delete** or **Replace** with a corrected file. 5. Verify that the incorrect file is removed and the replacement file is indexed if provided.				
Expected	The incorrect temporary document and its vector embeddings are deleted immediately. If a replacement document is uploaded, the system indexes the new document and makes it available for verification during the meeting.				
Date-Result	2 March 2026 - PASS. Temporary documents were successfully deleted and replaced.				

5.1.13 Test Case Scenario 13 (Agent Q&A - Answer Successfully Retrieved)

Table 14: Test Case Scenario 13

Test ID	TC-13	**Category**	**Functional**	**Severity**	**Medium**

Objective	This test case is to verify that the Q&A Agent can successfully retrieve an answer from authorized documents and display it with correct citations.
Steps	1. Ensure the VeriFact Bot is active in the meeting. 2. Ensure documents answering the question exist and the user has permission to view them. 3. A Member asks a specific question (e.g., "What is the budget cap for Q4 marketing?"). 4. The system interprets the intent and searches the vector database. 5. The system finds a matching section in the relevant document with high relevance.
Expected	The Agent generates a natural language answer (e.g., "The Q4 marketing budget is capped at $50,000."). The Agent appends a citation to the source document. The answer and citation are displayed in the chat interface
Date-Result	25 Feb 2026 - PASS. The Agent successfully retrieved the answer and provided the correct citation.

5.1.14 Test Case Scenario 14 (Agent Q&A - Conflicting Information Found)

Table 15: Test Case Scenario 14

Test ID	TC-14	**Category**	**Functional**	**Severity**	**Low**
Objective	This test case is to verify that the Q&A Agent correctly handles conflicting information across multiple documents and reports the discrepancy to the user.				
Steps	1. Ensure the VeriFact Bot is active. 2. Ensure the Knowledge Base contains outdated or contradictory documents (e.g., "Draft v1" vs "Final v2") that the user has access to. 3. A Member asks: "What is the project deadline?". 4. The system retrieves two high-ranking snippets with conflicting dates. 5. The Agent detects the inconsistency between the retrieved contexts.				
Expected	The Agent generates a synthesized answer highlighting the conflict (e.g., "I found conflicting information. The Project Charter states Dec 1st, but a recent email mentions Nov 15th."). The Agent cites both sources so the user can investigate.				

Date-Result	3 March 2026 - PASS. The Agent accurately reported the conflicting evidence and cited both sources for manual verification.

5.1.15 Test Case Scenario 15 (Agent Q&A - Answer Not Found / NEI)

Table 16: Test Case Scenario 15

Test ID	TC-15	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case is to verify that the Q&A Agent properly handles queries where the answer does not exist in the uploaded documents or the user lacks permission to view it.				
Steps	1. Ensure the VeriFact Bot is active. 2. A Member asks a question about an external topic (e.g., "What is the competitor's stock price today?"). 3. The system searches the authorized document embeddings. 4. The search returns low relevance scores (below the confidence threshold).				
Expected	The Agent generates a fallback response: "I cannot find information regarding that topic in the current document set.". No citations are displayed.				
Date-Result	3 March 2026 - PASS. The Agent provided the appropriate fallback response and informed the user that the Knowledge Base was insufficient for the query.				

5.1.16 Test Case Scenario 16 (Granting Permissions During a Meeting)

Table 17: Test Case Scenario 16

Test ID	TC-16	**Category**	**Functional**	**Severity**	**Critical**
Objective	This test case verifies that an Admin can update a document's accessibility level mid-meeting, granting immediate viewing access to a standard Member who was previously denied.				
Steps	1. A standard Member is denied access when clicking a citation link to an Admin-Only document (as per TC-04). 2. An Admin updates the document permission from "Only Owner" to "All Members" via the dashboard. 3. The Member clicks the citation again. 4. Verify that the document successfully opens for the Member.				

Expected	The system immediately updates the Member's session permissions. The Member can successfully open and view the previously restricted document. The access log records the permission change.
Date-Result	11 March 2026 - PENDING.

5.1.17 Test Case Scenario 17 (Document Upload - Unsupported File Format)

Table 18: Test Case Scenario 17

Test ID	TC-17	**Category**	**Functional**	**Severity**	**Medium**
Objective	This test case is to verify that the system correctly validates and rejects document uploads of unsupported file formats (e.g., EXE, ZIP, non-text files), maintaining system integrity.				
Steps	1. Admin navigates to the Permanent Documents management console. 2. Admin attempts to upload an unsupported file format (e.g., virus.exe) using the "Upload/Replace" function. 3. Observe the system's response to the file validation check.				
Expected	The system rejects the file at the format validation stage and displays an "Unsupported File Format" error message, preventing processing and storage of the file.				
Date-Result	11 March 2026 - PENDING.				

5.1.18 Test Case Scenario 18 (Post-Meeting Review - Export)

Table 19: Test Case Scenario 18

Test ID	TC-18	**Category**	**Functional**	**Severity**	**Medium**
Objective	This test case verifies the successful export of the complete post-meeting review, including the transcript, claims, and verdicts, in a user-selected format.				
Steps	1. A meeting is concluded with multiple detected claims and verification verdicts (Supported, Refuted, NEI). 2. User navigates to the Post-Meeting Review dashboard. 3. User selects "Export Summary" and chooses the PDF format.				

	4. Verify that the generated PDF file contains the full transcript, speaker identities, detected claims, their final classifications, and associated evidence excerpts.
Expected	A complete, correctly formatted PDF document is generated and downloaded, containing all verification and transcription data.
Date-Result	11 March 2026 - PENDING.

5.1.19 Test Case Scenario 19 (Reliability - Claim Buffer Test)

Table 20: Test Case Scenario 19

Test ID	TC-19	**Category**	**Functional**	**Severity**	**Medium**
Objective	This test case verifies the system's ability to handle temporary failure of the Claim Classification Service by correctly buffering and resuming processing claims.				
Steps	1. Initiate a failure state on the Claim Classification Service. 2. A Participant makes 10 factual statements that should be detected as claims. 3. The system detects the claims but is unable to classify them. 4. Restore the Claim Classification Service. 5. Observe the system's queue.				
Expected	The 10 unprocessed claims are buffered. Upon service restoration, all buffered claims are processed sequentially and their verdicts are displayed retroactively in the transcript.				
Date-Result	11 March 2026 - PENDING.				

5.1.20 Test Case Scenario 20 (Access Control - Denied Upload)

Table 21: Test Case Scenario 20

Test ID	TC-20	**Category**	**Functional**	**Severity**	**Medium**
Objective	This test case verifies that a standard Member user is correctly prevented from uploading permanent or temporary documents, enforcing the permission model.				

Steps	1. A standard Member (Role: Member) logs into the system. 2. The Member attempts to upload a document to the Permanent Document set. 3. The system checks the user's role against the required upload permission. 4. Observe the system's response.
Expected	The system denies the upload request and displays an "Insufficient Permissions" error or disables the upload button in the UI.
Date-Result	11 March 2026 - PENDING.

5.2 Non-Functional Test Cases

5.2.1 Non-Functional Test Case NTC-01

Table 22: Test Case Scenario 21

Test ID	NTC-01	**Category**	**Non-Functional**	**Severity**	**Critical**
Objective	This test case verifies that the system meets the strict latency requirements for real-time feedback during live meeting				
Steps	1. Start a meeting with the VeriFact bot active. 2. A participant makes a factual claim that requires document retrieval. 3. Measure the time from the end of the spoken sentence to the appearance of the verification badge in the UI. 4. Repeat across 10 different claims to calculate the average response time.				
Expected	The verification result, including the verdict and citation, must be displayed within 5 seconds to ensure the meeting flow is not disrupted.				
Date-Result	11 March 2026 - PENDING.				

5.2.2 Non-Functional Test Case NTC-02

Table 23: Test Case Scenario 22

Test ID	NTC-02	**Category**	**Non-Functional**	**Severity**	**Critical**
Objective	This test case verifies that the system can support multiple simultaneous meetings and organizations without performance degradation.				
Steps	1. Simulate 20 concurrent meeting sessions across 5 different organizations. 2. Trigger simultaneous claims across at least 10 of these meetings. 3. Monitor the response time for evidence retrieval and the CPU/Memory usage of the backend services.				
Expected	The system maintains stable performance, and the average verification latency does not exceed the threshold despite the high load.				
Date-Result	11 March 2026 - PENDING.				

5.2.3 Non-Functional Test Case NTC-03

Table 24: Test Case Scenario 23

Test ID	NTC-03	**Category**	**Non-Functional**	**Severity**	**Critical**
Objective	This test case verifies the system's ability to maintain stable verification performance and data integrity during long-duration meetings				
Steps	1. Initialize a meeting session and keep the VeriFact bot active for 120 minutes. 2. Periodically perform claim verifications throughout the duration. 3. Verify that the memory usage of the bot and the transcription pipeline remains stable (no leaks).				
Expected	The system maintains stable performance, and the average verification latency does not exceed the threshold despite the high load.				
Date-Result	11 March 2026 - PASS.				

5.2.4 Non-Functional Test Case NTC-04

Table 25: Test Case Scenario 24

Test ID	NTC-04	Category	Non-Functional	Severity	Critical
Objective	This test case verifies the system's ability to maintain stable performance and data integrity during long-duration meetings				
Steps	1. Initialize a meeting session and keep the VeriFact bot active for 120 minutes. 2. Periodically upload temporary documents and perform claim verifications throughout the duration. 3. Verify that the memory usage of the bot and the transcription pipeline remains stable (no leaks).				
Expected	The system maintains stable performance, and the average verification latency does not exceed the threshold despite the high load.				
Date-Result	11 March 2026 - PASS.				

5.2.5 Non-Functional Test Case NTC-05

Table 26: Test Case Scenario 25

Test ID	NTC-05	Category	Non-Functional	Severity	Critical
Objective	This test case verifies that document and verification data are strictly isolated between different organizations.				
Steps	1. Create two separate organizations, Org A and Org B, with distinct sets of permanent documents. 2. Start a meeting for Org B. 3. Attempt to query or retrieve evidence that exists only in Org A's knowledge base.				
Expected	The system correctly restricts retrieval to Org B's context only, returning no results or evidence from Org A's repository.				
Date-Result	11 March 2026 - PASS.				

5.2.6 Non-Functional Test Case NTC-06

Table 27: Test Case Scenario 26

Test ID	NTC-06	Category	Non-Functional	Severity	Low
Objective	This test case verifies that the UI follows usability standards to ensure that verification alerts support rather than disrupt meeting interactions.				
Steps	1. Conduct a mock meeting with five participants. 2. Trigger at least three "Refuted" or "Supported" alerts within a 2-minute window. 3. Use a post-test survey to measure if participants found the visual indicators (green/red badges) distracting from the verbal discussion.				
Expected	Participants should report that the indicators provided high-value information without requiring them to stop speaking or lose track of the conversation flow.				
Date-Result	11 March 2026 - Usability (Cognitive Load) was found optimal. PASS.				

5.2.7 Non-Functional Test Case NTC-07

Table 28: Test Case Scenario 27

Test ID	NTC-07	Category	Non-Functional	Severity	Medium
Objective	This test case verifies that the system correctly buffers unprocessed claims when the claim classification service is unavailable and processes them upon service restoration.				
Steps	1. Start a meeting with the VeriFact bot active and the claim classification service running. 2. Simulate a failure in the claim classification service (e.g., shut down the service or block its endpoint). 3. Have participants make factual claims continuously until at least 10 claims are queued. 4. Attempt to submit a 11st claim and observe how the system handles the overflow				

	5. Verify that all 11 buffered claims are processed correctly and their verification results appear in the UI.
Expected	While the classification service is down, claims must be buffered up to a maximum of 10. The 11st claim should be handled gracefully (e.g., user notification or silent drop with logging). Upon service restoration, all 10 buffered claims must be processed in order without data loss, and verification badges must appear in the UI.
Date-Result	11 March 2026 - PENDING

5.2.8 Non-Functional Test Case NTC-8

Table 29: Test Case Scenario 28

Test ID	NTC-08	**Category**	**Non-Functional**	**Severity**	**Critical**
Objective	This test case verifies that the Q&A bot acknowledges user queries within 2 seconds of activation phrase detection during a live meeting.				
Steps	1. Start a meeting with the VeriFact bot and Q&A bot active. 2. Speak the activation phrase followed by a query. 3. Measure the time from activation phrase detection to the first acknowledgment appearing in the UI. 4. Repeat across 10 different queries to calculate the average acknowledgment time.				
Expected	The Q&A bot must display an acknowledgment in the UI within 1 second of activation phrase detection for each query. The average acknowledgment time across 10 queries must not exceed 2 seconds.				
Date-Result	11 March 2026 - PENDING				

5.2.9 Non-Functional Test Case NTC-09

Table 30: Test Case Scenario 29

Test ID	NTC-09	Category	Non-Functional	Severity	Low
Objective	This test case verifies that the system properly handles Q&A bot timeout scenarios by notifying the user and logging the event when a response is not delivered within 10 seconds.				
Steps	1. Start a meeting with the VeriFact bot and Q&A bot active. 2. Simulate a delayed or unresponsive Q&A bot backend (e.g., inject artificial latency exceeding 10 seconds). 3. Speak the activation phrase and submit a query and observe whether the system displays the "Agent is temporarily unavailable" message after 10 seconds. 4. Check the diagnostic logs to confirm the timeout event was recorded with a timestamp, query content, and failure reason. 5. Repeat with 3 different queries to ensure consistent behavior.				
Expected	When the Q&A bot fails to respond within 10 seconds, the system must display "Agent is temporarily unavailable" to the user and log the timeout event.				
Date-Result	11 March 2026 - PENDING				

5.2.10 Non-Functional Test Case NTC-10

Table 31: Test Case Scenario 30

Test ID	NTC-10	Category	Non-Functional	Severity	Medium
Objective	This test case verifies that the document upload and vectorization pipeline supports parallel processing and meets the required completion times across small, medium, and large document sizes.				
Steps	1. Prepare a set of test documents: 3 small (≤10 pages), 3 medium (11–50 pages), and 3 large (51–100 pages).				

	2. Upload a small document and measure the time from upload initiation to vectorization completion. Repeat for all 3 small documents and calculate the average. 3. Upload a medium document and measure the time from upload initiation to vectorization completion. Repeat for all 3 medium documents and calculate the average. 4. Upload a large document and measure the time from upload initiation to vectorization completion. Repeat for all 3 large documents and calculate the average. 5. For each document, calculate the processing throughput by dividing the page count by the total processing time in seconds.
Expected	Small documents (≤10 pages) must complete upload and vectorization within 8 seconds. Medium documents (11–50 pages) must complete within 15 seconds. Large documents (51–100 pages) must complete within 30 seconds. Parallel uploads must not cause any individual document to exceed its respective time threshold.
Date-Result	11 March 2026 - **PASS**. Small docs averaged **2.24**s (≤8s), medium docs averaged **5.83**s (≤15s), large docs averaged **14.21**s (≤30s). Overall average throughput was **5.38** pages/second (≥3 pp/s). All thresholds met across 9 test documents.

5.2.11 Non-Functional Test Case NTC-11

Table 32: Test Case Scenario 31

Test ID	NTC-11	**Category**	**Non-Functional**	**Severity**	**Medium**
Objective	This test case verifies that the speaker identification subsystem correctly attributes speech to the correct participant and that the RAG retrieval subsystem returns relevant evidence for detected claims.				
Steps	1. Start a meeting with 5 participants. 2. Have each participant make at least 5 factual claims (25+ total claims). 3. For each claim, record which participant ID the system attributed it to and compare against ground truth. 4. Calculate speaker identification accuracy as (correct attributions / total claims) × 100. 5. For each detected claim, check if the RAG subsystem returned at least one evidence snippet with cosine similarity > 0.6.				

	6. Calculate RAG hit rate as (claims with valid evidence / total detected claims) × 100.
Expected	Speaker identification accuracy must be at least 95% across all participants. The RAG retrieval subsystem must return at least one evidence snippet with cosine similarity > 0.6 for at least 90% of detected claims.
Date-Result	11 March 2026 - Pending

5.2.12 Non-Functional Test Case NTC-12

Table 33: Test Case Scenario 32

Test ID	NTC-12	**Category**	**Non-Functional**	**Severity**	**Critical**
Objective	This test case verifies that the vector database maintains acceptable query performance at scale and that the claim classification service meets throughput requirements under load.				
Steps	1. Populate the vector database with 1 million embedded document chunks across multiple organizations. 2. Execute 100 similarity search queries and measure the response time for each. 3. Calculate the average and 95th percentile query response times. 4. Simultaneously, send a burst of 100 claims to the claim classification service over a 5-second window. 5. Measure the classification throughput (claims processed per second) and verify no claims are dropped.				
Expected	Vector database query response time must remain below 150 ms at the 95th percentile with 1 million chunks stored. The claim classification service must process at least 20 claims per second under load without dropping any claims.				
Date-Result	11 March 2026 - **Failed** Performance Metrics Results: **Vector Query Latency:** The system achieved a **P95 response time of 420.19 ms**, significantly exceeding the maximum allowable threshold of **150 ms** even at a 1% load scale. (10k chunks) **(Failed)**				

	Classification Throughput: The claim classification service reached a peak throughput of **19.8 claims/sec**, narrowly missing the minimum requirement of **20 claims/sec**.**(Can be considered as Passed)**

5.2.13 Non-Functional Test Case NTC-13

Table 34: Test Case Scenario 33

Test ID	NTC-13	**Category**	**Non-Functional**	**Severity**	**Low**
Objective	This test case verifies that the system enforces document retention policies by archiving permanent documents older than 365 days to cold storage and deleting temporary documents after the meeting ends.				
Steps	1. Upload a permanent document and a temporary document to an organization's repository. 2. Start and end a meeting that references the temporary document. 3. Verify that the temporary document is deleted from the active storage after the meeting concludes. 4. Simulate the passage of 365 days for the permanent document (e.g., modify the document's upload timestamp in the database). 5. Trigger the archival job and verify the document is moved to cold storage. 6. Request retrieval of the archived document and measure the time until it becomes accessible.				
Expected	Temporary documents must be deleted from active storage after the meeting ends. Permanent documents older than 365 days must be archived to cold storage. Retrieval of archived documents must complete within 24 hours.				
Date-Result	11 March 2026 - PENDING				

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

The development of VeriFact is constrained by several technical factors related to real-time processing requirements. Since the system performs speech transcription, claim detection, evidence retrieval, and verification during live meetings, strict latency requirements must be satisfied. Each stage of the pipeline must operate efficiently so that verification results can be produced within a few seconds, ensuring that the feedback remains useful during ongoing discussions. These constraints influence architectural decisions such as modular pipeline design, lightweight machine learning models, and efficient vector retrieval mechanisms.

Privacy and regulatory considerations also impose significant constraints on the system design. Because VeriFact processes meeting transcripts and potentially sensitive organizational documents, it must comply with data protection regulations such as GDPR. To address these constraints, the system adopts a data minimization approach, stores only necessary information, and ensures secure communication and storage through encryption mechanisms. Additionally, temporary meeting data must be automatically deleted after sessions to prevent unintended retention of sensitive information.

Economic and practical constraints further shape the implementation choices of the project. Limited computational resources and project budgets restrict the use of expensive large-scale models or extensive cloud infrastructure. As a result, the system prioritizes efficient algorithms and scalable components that can operate reliably without requiring excessive GPU resources. These constraints encourage the use of optimized retrieval methods, modular architecture, and lightweight machine learning models that balance performance, cost, and scalability.

Another important constraint arises from the integration with the Zoom platform. Since VeriFact operates as a native Zoom application, its functionality depends on the capabilities and limitations of the Zoom SDK and APIs. Changes in Zoom's platform policies, API endpoints, or supported features may affect the system's ability to capture audio streams, manage meeting events, or display the in-meeting interface. Therefore, the system architecture must remain modular and adaptable so that potential SDK updates or deprecations can be handled without requiring a complete redesign of the system.

Social and organizational constraints also influence the design of the system. Because VeriFact automatically analyzes statements during meetings, it may affect the dynamics of discussions between participants. To avoid creating a surveillance-like atmosphere or disrupting natural conversation flow, the system presents verification outcomes carefully using evidence excerpts and confidence indicators rather than authoritative judgments. This design approach helps maintain trust among participants while still providing useful verification assistance during discussions.

Scalability constraints must also be considered when designing the system architecture. VeriFact is expected to support multiple organizations, meetings, and users simultaneously while maintaining reliable performance. This requires careful design of storage systems, vector databases, and retrieval pipelines so that they can handle increasing amounts of documents and claims without significant degradation in response time. Efficient indexing and scalable cloud-based storage solutions are therefore necessary to ensure that the system can grow with organizational usage.

Environmental and infrastructure considerations provide another constraint in system design. Large-scale machine learning systems can consume substantial computational resources and energy, especially when deployed in cloud environments. To minimize unnecessary resource consumption, VeriFact aims to use models that are sufficiently powerful for the task while avoiding excessive computational overhead. This approach supports more sustainable system operation and allows the platform to scale without disproportionately increasing infrastructure costs or energy usage.

6.2 Standards

The VeriFact project follows established software engineering, security, AI, and usability standards to ensure correctness, reliability, and compliance throughout design and implementation.

- IEEE 830 / IEEE 29148 (Requirements Engineering): Used to structure functional and non-functional requirements so that they are complete, testable, and traceable across the system lifecycle [5].
- IEEE 1016 (Software Design Descriptions): Applied to document system architecture, modules, interfaces, and data flows in a consistent and industry-recognized format [6, 7].
- UML 2.5.1 (Unified Modeling Language): Used for use case, component, and sequence diagrams to ensure standardized system modeling [8].
- IEEE 829 / IEEE 29119 (Software Test Documentation): Referenced to structure test plans and reports and link requirements to verifiable test cases [9].
- GDPR (General Data Protection Regulation): Guides user consent handling, data retention policies, and deletion of temporary meeting data.
- ISO/IEC 27001 (Information Security Management): Used to manage risks related to document storage, user data, and backend access control [10].
- ISO/IEC 27018 (Protection of PII in Clouds): Applied to ensure safe handling of personally identifiable information in cloud environments [11].

- OWASP ASVS (Application Security Verification Standard): Referenced for securing APIs through authentication, input validation, and protection against common web vulnerabilities [12].
- ISO/IEC 23894 (AI Risk Management): Used to identify and mitigate AI-related risks such as bias and misclassification [13].
- NIST AI Risk Management Framework: Guides transparency, reliability, and human oversight practices in AI-driven verification workflows [14].
- ML Reproducibility Best Practices: Ensures reproducible model training through versioned datasets, pipelines, and model checkpoints [15].
- TLS 1.2+ (Encrypted Communication): Ensures secure communication between the Zoom bot, backend services, and storage layers.
- AES-256 (Encryption at Rest): Used to encrypt stored documents, embeddings, and metadata [4].
- S3-Compatible Object Storage Conventions: Used to enforce lifecycle rules for document expiration and retention.
- ISO 9241-210 (Human-Centered Design): Followed to reduce cognitive load and ensure fact-checking alerts support, rather than disrupt, meeting interactions [16, 17].
- Zoom App Framework UI Consistency Standards: Ensures the interface integrates seamlessly with native Zoom controls and user expectations.

7. Teamwork Details

7.1 Contributing and Functioning Effectively on the Team

Each team member contributed actively to the development of the VeriFact system by taking responsibility for specific work packages while coordinating closely with others during integration phases. The project was structured around clearly defined modules such as Zoom integration, claim detection and verification, document management, and frontend visualization. This modular structure enabled parallel development while ensuring that components could later be integrated into a single real-time verification pipeline.

Team members regularly communicated implementation progress and technical challenges through meetings and shared repositories. Code reviews and design discussions were used to ensure that architectural decisions were consistent across modules. When integration issues arose between components such as the transcription pipeline, RAG retrieval system, and frontend interface, the team worked collectively to debug and resolve them.

In addition to their individual responsibilities, team members supported one another during the implementation and testing phases. For example, members working on backend infrastructure coordinated with those implementing machine learning components to ensure compatible data formats, APIs, and latency requirements. This collaborative workflow allowed the team to maintain steady development progress and meet project milestones.

7.2 Helping Create a Collaborative and Inclusive Environment

The team prioritized maintaining an open, respectful environment in which all members could contribute ideas and feedback. Regular meetings were held to discuss design decisions, evaluate alternative solutions, and review progress. During these discussions, every team member was encouraged to voice concerns, suggest improvements, and participate in decision-making.

Task allocation was handled transparently based on members' technical strengths and interests, while still allowing opportunities for learning new technologies. When difficulties occurred, such as debugging integration issues or optimizing performance in the verification pipeline, team members supported each other by sharing knowledge and troubleshooting together.

The team also maintained shared documentation and version control practices so that all members had access to project resources and updates. This ensured transparency in development progress and allowed every member to stay informed about changes in the system architecture and implementation.

7.3 Taking a Lead Role and Sharing Leadership on the Team

Leadership responsibilities were distributed across different work packages in the project plan. Each major component of the system had a designated leader responsible for coordinating development tasks, ensuring progress, and facilitating communication among members involved in that module. For example, leadership roles were assigned for areas such as claim detection and verification integration, document management infrastructure, and frontend interface development.

This distributed leadership structure allowed the team to manage a complex system efficiently while ensuring accountability for each subsystem. Leaders organized tasks, monitored technical progress, and coordinated integration with other components. At the same time, leadership was flexible and shared across the team when solving cross-module problems, especially during system integration and testing phases.

By combining individual ownership of modules with collaborative decision-making, the team maintained both clear responsibility and collective accountability for the final system.

7.4 Project Plan

Table 35: Factors that can affect analysis and design.

	Effect level	Effect
Public health	High	Misleading verification outputs may influence clinical discussions or policy discussions. This requires a cautious and detailed view of verification results, for it to be verifiable and reproducible.
Public safety	High	In legal and regulatory contexts, incorrect verification outputs could lead to unlawful decisions. To mitigate this, the system ensures to provide evidence excerpts and use RAG to minimize hallucinations.
Public welfare	Medium	Organizational trust and meeting outcomes may be affected. False positives could disrupt collaboration or reduce confidence in using the software.
Global factors	Medium	Regulations such as GDPR, regarding data processing and retention, are constraints that influence the architecture choices related to encryption and data minimization (store what you need style approach).
Cultural factors	Low	Cultural factors are not too considerable, especially since VeriFact is only planned to support English.
Social factors	Medium	The presence of VeriFact, being an automatic verification system, may alter the interaction between participants and presenters. Therefore, VeriFact’s goal is to

		avoid surveillance and help provide more transparent information.
Environmental factors	Low	Since we plan to host our infrastructure on the cloud at a later stage, the environmental and energy consumption can be considered. We will ensure our models are not too overpowered for our use cases, as well as having our system scale independently to not waste any resources.
Economic factors	Medium	Project budget limitations restrict use of expensive GPU compute and large-scale experimentation.

Table 36: List of work packages

WP#	Work package title	Leader	Members involved
WP1	Requirements Finalization & System Integration Design	Alhassan Raad Jassim Al-badri	Orhun Ege Çelik, Egehan Yıldız
WP2	Zoom Integration & Audio Ingestion Pipeline	Orhun Ege Çelik	Alhassan Raad Jassim Al-badri, Eray İşçi
WP3	Claim Detection & Verification Pipeline Integration	Egehan Yıldız	İrem Damla Karagöz
WP4	Document Management & Organization Infrastructure	Eray İşçi	Orhun Ege Çelik
WP5	Frontend UI & Real-Time Visualization	İrem Damla Karagöz	Alhassan Raad Jassim Al-badri

WP6	Testing, Evaluation & Finalization	Egehan Yıldız	All Members

Table 37: Complete details of work packages

WP 1: *Requirements Finalization & System Integration Design*			
Start date: Week 1 **End date:** Week 3			
Leader:	Alhassan Raad Jassim Al-badri	**Members involved:**	Orhun Ege Çelik, Egehan Yıldız
Objectives: The objective of this work package is to finalize system requirements and align existing components into a coherent end-to-end architecture. Since several core features are already implemented, this package focuses on refining interfaces, defining pipeline boundaries, and resolving integration assumptions. It also ensures consistency between functional requirements, ethical constraints, and system behavior.			
Tasks: **Task 1.1: Requirements Refinement** – Review and finalize functional and non-functional requirements based on the current implementation status. **Task 1.2: Pipeline Definition** – Define the final data flow between Zoom integration, STT, claim detection, verification, and UI layers. **Task 1.3: Integration Planning** – Identify missing links between existing components and define integration milestones.			
Deliverables **D1.1:** Finalized Requirements Specification **D1.2:** Updated System Architecture Diagram			
WP 2: *Zoom Integration & Audio Ingestion Pipeline*			
Start date: *Week 3* **End date:** *Week 6*			
Leader:	Orhun Ege Çelik	**Members involved:**	Eray İşçi
Objectives: This work package focuses on stabilizing and extending the existing Zoom integration. The goal is to ensure reliable meeting joining, audio capture, and event			

handling under real-time conditions. Emphasis is placed on robustness, speaker identification, and clean audio streaming to downstream components.			
Tasks: **Task 2.1: Zoom Bot Stabilization** – Improve reliability of bot join/leave behavior and meeting lifecycle handling. **Task 2.2: Audio Stream Normalization** – Ensure captured audio is consistently formatted and streamed to the STT module. **Task 2.3: Event Handling** – Capture participant and meeting events for synchronization with the pipeline.			
Deliverables **D2.1:** Stable Zoom Bot Integration **D2.2:** Audio Ingestion Interface			
WP 3: *Claim Detection & Verification Pipeline Integration*			
Start date: *Week 6* **End date:** *Week 8*			
Leader:	Egehan Yıldız	**Members involved:**	İrem Damla Karagöz
Objectives: This work package integrates the already available claim detection and claim verification components into a single real-time pipeline. While claim verification exists as an MVP, the focus here is on connecting it to live transcripts, evidence retrieval, and result streaming. Performance and latency constraints are central concerns.			
Tasks: **Task 3.1: Claim Detection Integration** – Connect the claim detection model to live STT output. **Task 3.2: Verification Pipeline Wiring** – Integrate the existing claim verification MVP into the main pipeline. **Task 3.3: Latency Optimization** – Optimize pipeline execution to meet real-time constraints.			
Deliverables ***D3.1:*** Integrated Claim Detection Module			

D3.2: End-to-End Claim Verification Pipeline			
WP 4: *Document Management & Organization Infrastructure*			
Start date: *Week 8* **End date:** *Week 12*			
Leader:	Eray İşçi	**Members involved:**	Orhun Ege Çelik
Objectives: This work package extends the partially implemented organization and document management flow. The goal is to transition from local storage to a structured, secure storage model and enable document usage within the verification pipeline. Access control and organization-level separation are key priorities.			
Tasks: **Task 4.1: Organization Flow Completion** – Finalize organization creation, invitations, and role handling. **Task 4.2: Document Storage Integration** – Replace local storage with a scalable object storage solution (e.g., S3-compatible). **Task 4.3: Document Indexing** – Enable document embedding and indexing for retrieval during verification.			
Deliverables **D4.1:** Organization & Permission Management Module **D4.2:** Secure Document Storage & Indexing System			
WP 5: *Frontend UI & Real-Time Visualization*			
Start date: *Week 12* **End date:** *Week 14*			
Leader:	İrem Damla Karagöz	**Members involved:**	Alhassan Raad Jassim Al-badri
Objectives: This work package focuses on presenting system outputs to users in a clear and non-intrusive manner. The UI will display live transcripts, detected claims, verification results, and evidence while respecting permission constraints. The design prioritizes usability within the Zoom side panel.			
Tasks: **Task 5.1: Transcript & Claim Visualization** – Render live transcripts and detected claims in real time.			

Task 5.2: Verification Result Display – Show verification status, confidence, and evidence snippets. **Task 5.3: Q&A Interface Integration** – Integrate the in-meeting Q&A interaction flow.			
Deliverables ***D5.1:*** Zoom-Embedded Frontend Interface ***D5.2:*** Real-Time Visualization Components			
WP 6: *Testing, Evaluation, Iteration & Finalization*			
Start date: *Week 14* **End date:** *Week 18*			
Leader:	Egehan Yıldız	**Members involved:**	All Members
Objectives: The objective of this work package is to validate the system against functional, performance, and ethical requirements. It includes testing under realistic meeting scenarios, measuring latency and accuracy, and preparing the final documentation and demonstration.			
Tasks: **Task 6.1: Functional & Integration Testing** – Verify correctness of end-to-end workflows. **Task 6.2: Performance Evaluation** – Measure latency, throughput, and reliability. **Task 6.3: Documentation & Presentation** – Prepare final reports and project presentation.			
Deliverables **D6.1:** Test & Evaluation Report **D6.2:** Final Project Report and Demo			

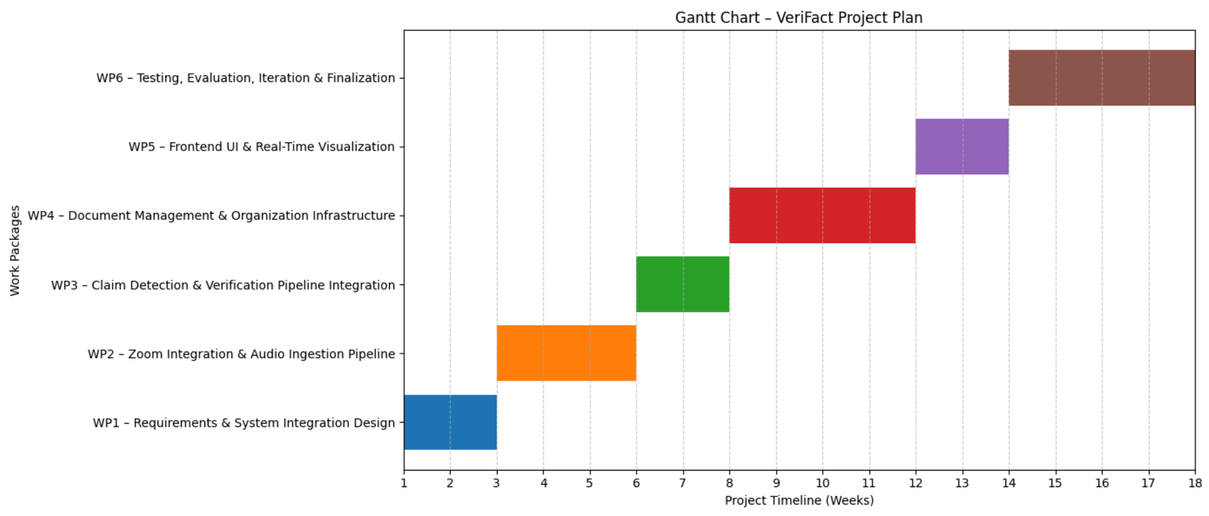

Figure 2: Gantt Chart of VeriFact.

8. Glossary

Claim Detection Model: A machine learning model that determines whether a sentence should be considered a claim. Our system uses a lightweight local model to perform this analysis in real time.

Confidence Score: A probability value that reflects the model's certainty in its prediction. This score is used to filter out low-confidence outputs.

Data Retention Policy: Rules that specify how long audio, transcripts, and embeddings remain available in memory.

Embedding: A numerical vector that represents the meaning of text. These vectors help the system perform semantic search during evidence retrieval.

Encryption (AES-256): An encryption algorithm used to encrypt stored documents, embeddings, and metadata at rest, where stored (e.g., in S3-compatible object storage).

Evidence Retrieval Module: The part of the system that searches a curated dataset to locate information relevant to a detected claim. It relies on embeddings and similarity search.

Pipeline: The full sequence of operations that data passes through. This includes audio streaming, transcription, segmentation, claim detection, retrieval, scoring, and presentation in the user interface.

RAG (Retrieval-Augmented Generation): A method that enriches model outputs with information retrieved from a knowledge source. In this project, retrieval is used to find relevant evidence for each detected claim.

Zoom Bot: An automated participant in a Zoom meeting that receives audio, processes it, and sends results to the user interface.

Zoom SDK (Software Development Kit): The toolkit that enables integration with Zoom and access to audio streams and meeting events [3].

9. References

[1]“Class ZoomSdk,” ZoomSdk | @zoom/appssdk - v0.16.36, https://appssdk.zoom.us/classes/ZoomSdk.ZoomSdk.html (accessed Nov. 27, 2025).

[2] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, “Fever dataset,” Fact Extraction and VERification, https://fever.ai/dataset/fever.html (accessed Nov. 27, 2025).

[3] Granola AI, *Granola*, San Francisco, CA, USA: Granola AI, 2026. [Online]. Available: https://granola.ai (accessed Mar. 11, 2026).

[4] National Institute of Standards and Technology (NIST), “Advanced Encryption Standard (AES),” FIPS PUB 197, Nov. 2001.

[5] IEEE Standard 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE, 1998.

[6] ISO/IEC/IEEE 29148:2018, *Systems and Software Engineering — Life Cycle Processes — Requirements Engineering*, 2018.

[7] IEEE Standard 1016-2009, *IEEE Standard for Information Technology — Systems Design — Software Design Descriptions*, IEEE, 2009.

8] Object Management Group (OMG), *Unified Modeling Language (UML) Specification, Version 2.5.1*, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/

[9] IEEE Standard 829-2008, *IEEE Standard for Software and System Test Documentation*, IEEE, 2008.

[10] ISO/IEC/IEEE 29119-1:2013, *Software and Systems Engineering — Software Testing — Part 1: Concepts and Definitions*, 2013.

[11] ISO/IEC 27018:2019, *Information Technology — Security Techniques — Code of Practice for Protection of Personally Identifiable Information (PII) in Public Clouds Acting as PII Processors*, 2019.

[12] OWASP Foundation, *OWASP Application Security Verification Standard (ASVS) Version 4.0.3*, OWASP, 2021. [Online]. Available: https://owasp.org/ASVS/

[13] ISO/IEC 23894:2023, *Information Technology — Artificial Intelligence — Guidance on Risk Management*, 2023.

[14] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST, Jan. 2023.

[15] P. Pineau *et al.*, “Improving reproducibility in machine learning research,” *arXiv preprint arXiv:2003.12206*, 2020.

[16] T. Gebru *et al.*, “Datasheets for datasets,” *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, 2021.

[17] ISO, *ISO 9241-210:2019 — Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*. International Organization for Standardization, Geneva, Switzerland, 2019.