Software Requirements Specification

for

OmniView

Version 2.0

Prepared by Divyansh Yadav, Akhil Dhyani, Harshil Agrawal, Harshit Dhangar

Software Engineering Course

Date: 04/02/2026

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Revised permission handling to support on-demand user control and align with REQ-2	28/01/26	To ensure the system respects user-controlled permission enablement and revocation.	2
Justified screenshot capture interval and clarified storage management approach	28/01/26	Earlier no justification was given for the time interval chosen and storage management for screenshot capture.	2
Sensitive application handling to support user-defined blacklist	28/01/26	Clarified that the user can add or remove applications from the blacklist.	2
Updated Component, Component diagram	28/01/26	Connection between storage and search modules established, Search module added and justified intelligence module being part of the AI-core.	2
Updated Timing diagram	04/02/26	Added work-based diagrams.	2
Added Traceability matrix	04/02/26	Added traceability matrix to record updates..	2
Assigned Alpha Managers	04/02/26	Divided the roles of alpha managers between the team members.	2
New requirement documented: Timeline Visualisation	04/02/26	Added more features on the UI side.	2
Updated existing functional requirements	04/02/26	Updated REQ-3, 4, 5, 10, 14 to better match the product.	2
Added Development Plan and Sprint Breakdown	04/02/26	Added a new section to document a core plan to start coding.	2
Updated Document Conventions	04/02/26	Added naming conventions, commenting style, file structure.	2

Alpha Managers

Name	Alpha Managers
Akhil Dhyani	Requirements Software System
Divyansh Yadav	Work Way of Working
Harshil Agrawal	Stakeholders Opportunity
Harshit Dhangar	Team

Requirements Traceability Matrix

Attribute	Requirements	Component	Class	Functions	Start Date of Code	End Date of Code	Team Member(s) Designing Code	Test / Integration Status Details	
								Team Member(s) Testing Code	Testing Status
Description	It traces the requirements of the OmniView system.	OmniView System	Multiple	Multiple			Akhil Divyansh Harshil Harshit	Akhil Divyansh Harshil Harshit	
Examples	Screenshot Capture	Ingestion Module	SnapshotManager	captureScreen()					
				calculatePHash()					
				manageDuplicate()					
			BlacklistManager	isBlacklisted()					
				addApp()					
				removeApp()					
			PermissionPolicyManager	isCaptureAllowed()					
				onPermissionChanged()					
				onUserPause()					
	Context Extraction	Intelligence Module	VectorEngine	encodeText()					
				computeSimilarity()					
			WorkManagerScheduler	scheduleNightCrawler()					
			RAGOrchestrator	generateResponse(query, context)					
	Local Storage	Storage Module	IngestionRepository	saveToWAL()					
				processPendingData()					
			HistoryManager	deleteByTimeRange()					
				deleteAll()					
	User Queries	Search Module	SearchRepository	searchKeyword()					
				searchSemantic()					
				getTimeline()					
	User Experience	UI Module	SearchViewModel	search(query)					
				getTimeline()					

1. Introduction

1.1 Purpose

This document specifies the software requirements for OmniView, an Android-based on-device screen understanding system. This SRS covers the complete OmniView system and corresponds to the initial prototype developed as part of a Software Engineering course.

1.2 Document Conventions

Requirements are uniquely identified using the format REQ-x. Each toggle indicates priority using High, Medium, or Low classification.

To ensure maintainability, readability, and uniformity across modules, OmniView follows a consistent design and coding template.

1.2.1 Naming Conventions

- **Classes:** PascalCase
  - Example: SnapshotManager, VectorEngine
- **Functions:** camelCase with action-oriented names
  - Example: captureScreen(), encodeText()
- **Variables:** camelCase
  - Example: isRecording, blacklistedApps
- **Constants:** UPPER_SNAKE_CASE
  - Example: CAPTURE_INTERVAL_MS

1.2.2 File and Module Structure

Each module resides in a dedicated package

- ingestion/
- storage/
- intelligence/
- search/
- ui/

1.2.3 Commenting and Documentation style

Function-level comments describe intent, inputs, and outputs. Complex logic blocks include inline explanations. Public interfaces include brief docstring style comments.

Example:

```
/**
 * Captures the current screen if permissions are granted
 * and the active application is not blacklisted.
 */
fun captureScreen(): Bitmap
```

1.3 Intended Audience and Reading Suggestions

This document is intended for faculty members and academic evaluators. Readers are encouraged to begin with Sections 1 and 2 for an overview and then proceed to Sections 4 and 5 for detailed functional and nonfunctional requirements. Section 6 outlines the overall development plan.

1.4 Product Scope

OmniView is a standalone Android application that continuously captures screen content at fixed intervals, extracts information, stores contextual vector embeddings locally, and allows users to query historical screen activity. The system aims to provide screen intelligence functionality on mobile devices while respecting mobile constraints.

1.5 References

Screenpipe, a similar product for desktop – https://screenpi.pe/

2. Overall Description

2.1 Product Perspective

OmniView is a new, self-contained product designed as a mobile alternative to desktop-based screen understanding systems such as Screenpipe. The system follows a modular architecture where the Storage Module exposes controlled interfaces for data ingestion, retrieval, and batch processing.

2.2 Product Functions

- Periodic screenshot capture
- Context extraction using Android's Accessibility API, Google's ML Kit or other local first on-device models
- Vector embedding based context storage
- Query answering over stored context
- Visualise timelines
- Permission policy management
- History deletion
- Battery state monitoring
- Optional RAG-based response generation for complex user queries

2.3 User Classes and Characteristics

OmniView supports a single user class: the device owner. The intended user is non-technical and interacts with the system passively.

2.4 Operating Environment

The system operates on Android mobile devices. The application is developed using Kotlin.

2.5 Design and Implementation Constraints

The system is constrained by strict battery usage limits, storage limitations, and Android OS permission restrictions related to screen capture and background execution.

2.6 User Documentation

A user manual will be delivered along with the OmniView software. The user manual will provide instructions for installation, permission configuration, and basic usage of the system. The documentation will be provided in a standard digital format.

2.7 Assumptions and Dependencies

The system assumes that required screen capture and accessibility permissions can be granted by the user on-demand during system operation. Users may enable or revoke these permissions at any time, and the system is designed to dynamically adapt by pausing all capture and context extraction activities when permissions are revoked, in accordance with REQ-2. While core functionality operates on-device when permissions are active, network connectivity is assumed for processing complex queries via external large language model APIs. The system relies on Android Accessibility APIs and Google ML Kit; therefore, changes to these services, permission models, or device resource availability may affect system behaviour. If the user explicitly consents, the system may transmit retrieved contextual summaries (not raw screenshots or full stored data) to external large language model APIs for RAG-based response generation. This external interaction is optional and does not affect core system functionality.

3. External Interface Requirements

3.1 User Interfaces

The application operates primarily in the background with minimal user interaction. Always-on capture mode is supported. A chat-like interface is needed for query answering and timeline visualisation.

3.2 Hardware Interfaces

The system interfaces with the device screen buffer and media output only. No microphone or sensor access is required.

3.3 Software Interfaces

The system interfaces with Android OS services, SQLite for local storage, and lightweight NLP libraries for question answering.

3.4 Communications Interfaces

The system does not require network communication for core functionality. Optional external communication may occur only if the user explicitly opts in for external LLM-based query augmentation (REQ-10). External communication is limited to optional RAG-based query

augmentation and occurs only after explicit user consent. All core capture, extraction, storage, and search operations function entirely on-device.

4. System Features

This section organises the functional requirements of OmniView by system features, each representing a major service provided by the system. Requirements are specified to clearly describe system behaviour and support effective design and evaluation.

4.1 Screenshot Capture

4.1.1 Description and Priority : Periodically captures device screenshots on visual content change. Priority: High

4.1.2 Stimulus/Response Sequences : System timer triggers screenshot capture and stores image locally.

4.1.3 Functional Requirements

REQ-1: The system shall capture screenshots every 10 seconds. The 10 second interval is decided to maintain contextual accuracy with resource efficiency. Shorter intervals would increase processing and memory overhead without proportional gains in context, while longer intervals risk missing intermediate screen changes. To ensure sufficient storage availability, screenshots are retained only for the duration required for on-device processing and feature extraction. Once the relevant contextual information is embedded and stored in the vector database, the raw screenshots are immediately discarded, ensuring that local storage usage remains minimal and bounded. Hash-based change detection is used to discard redundant frames. The capture interval defines the upper bound, while hash-based change detection ensures screenshots are processed only when meaningful visual changes occur.

REQ-2: The system shall pause capture when permissions are revoked.

4.2 Context Extraction and Processing

4.2.1 Description and Priority:

This feature is responsible for extracting textual and contextual information from captured screen content using Android's Accessibility API, Google ML Kit, other local libraries/models and storing the processed context as a vector embedding for later retrieval. Priority: High

4.2.2 Stimulus/Response Sequences

When a new screen capture occurs, the system invokes the Accessibility API and Google ML Kit to obtain screen content. The extracted information is processed and converted into contextual representations, which are then stored in the database.

4.2.3 Functional Requirements

REQ-3: The system shall extract on-device text from Accessibility APIs.

REQ-4: The system shall extract context from the screenshots using on device libraries/models or Google ML kit. If heavy processing is required, then it is to be done only on device charging.

REQ-5: The system shall store extracted context locally.

4.3 Historical Search and Retrieval

4.3.1 Description and Priority :

This feature enables users to retrieve previously stored contextual data through historical search mechanisms. The system supports two search modes: keyword-based search for direct text matching and natural language–based search for semantic understanding of user queries. The feature operates entirely on locally stored data and leverages vector similarity matching for NLP-based queries. In addition to fully local keyword-based and semantic search, OmniView optionally supports Retrieval Augmented Generation (RAG) using external large language models. This functionality is

strictly opt-in and is invoked only after local context retrieval has been completed.
Priority: High

4.3.2 Stimulus/Response Sequences :

When a user initiates a search request, the system determines the query type (keyword-based or natural language)

- For keyword-based searches, the system scans stored textual metadata for exact or partial matches.
- For NLP-based searches, the system converts the query into a vector embedding and performs similarity matching against stored context embeddings in the vector database. Optionally, using external LLMs for Retrieval Augmented Generation or specialised models can be considered.

4.3.3 Functional Requirements:

- REQ-6: The system shall allow users to search historical data using keyword-based queries.
- REQ-7: The system shall allow users to search historical data using natural language queries.
- REQ-8: The system shall retrieve and rank relevant historical context based on query relevance.
- REQ-9: The system shall perform all search operations locally without transmitting data externally
- REQ-10: If the user has consented to external usage, the system shall generate a RAG based response using external LLMs.

4.4 Sensitive Application Blacklisting

4.4.1 Description and Priority:

This feature prevents screen capture and context processing for sensitive applications such as banking and payment apps to

ensure user privacy. The system enforces a blacklist to block all recording-related operations for these applications. The user will be allowed to add or remove applications to this blacklist at his will. Priority: High

4.4.2 Stimulus/Response Sequences

When a screen capture is triggered, the system checks the active application. If the application is blacklisted, the capture and processing are blocked; otherwise, normal processing continues.

Functional Requirements7

- REQ-11: The system shall maintain a blacklist of sensitive applications.
- REQ-12: The system shall block screen capture and context extraction for blacklisted applications.
- REQ-13: The system shall allow users to manage the application blacklist.
- REQ-14: The system shall also allow on-demand stoppage of screen capture in non-blacklisted applications.

4.5 Timeline Visualisation and Statistics

4.5.1 Description and Priority :

This feature provides users with a chronological overview of their device activity and analytical insights into their app usage patterns. By leveraging stored contextual metadata, the system generates a visual timeline and statistical reports to help users understand their digital habits.
Priority: Low

4.5.2 Stimulus/Response Sequences :

- When the user navigates to the Timeline section, the system queries the local database for historical

entries and renders them as a chronological sequence of events.

- When the user requests usage statistics, the system aggregates data based on app category, time spent, or frequency of use and displays it through charts or summary cards.

4.5.3 Functional Requirements:

- REQ-15: The system shall display a chronological timeline of captured user activity.
- REQ-16: The system shall provide a graphical summary (e.g., charts or graphs) of app usage and frequent topics based on extracted context.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Screenshot capture on visual content change (difference > 10%) with adaptive throttling to minimise battery impact.

5.2 Safety Requirements

The system shall never capture a screen without explicit user permission.

5.3 Security Requirements

Data is stored locally on the device and access is restricted by the operating system's permission and application sandboxing mechanisms.

5.4 Software Quality Attributes

Primary quality attributes are battery efficiency (less than 3-5% battery per hour), storage efficiency (less than 1 GB per week), latency (less than 200 ms per query) and usability.

5.5 Business Rules

Only the device owner may access stored data.

5.6 Privacy Rules

Completely on-device. Optional external LLM calls/external services if opted in.

6. Development Plan and Sprint Breakdown

6.1 Development Methodology

The development of OmniView is based on an Agile development sprint process. This is because of the exploratory nature of on-device intelligence, performance tuning with mobile constraints, and design decisions regarding privacy and energy efficiency. The development process is divided into short time-bound sprints, which result in a testable system increment.

The duration of each sprint is about 1-2 weeks, which enables rapid feedback and traceability to functional requirements (REQ-x).

6.2 Sprint-wise Breakdown

6.2.1 Sprint 1: Core Ingestion and Permission Control

Functionalities Covered:

- Screenshot Capture Service
- Permission Handling
- Sensitive Application Blacklisting

Rationale:

These modules are fundamental to system operation and user privacy. Implementing them first ensures early validation of Android OS constraints, permission revocation handling (REQ-2), and privacy enforcement (REQ-11 to REQ-14).

Mapped Requirements: REQ-1, REQ-2, REQ-11–REQ-14

6.2.2 Sprint 2: Context Extraction and Storage

Functionalities Covered:

- Accessibility API integration
- On-device text extraction
- Local database and storage scheduler

Rationale:

Once ingestion is stable, contextual data extraction and persistence can be safely implemented. This sprint validates assumptions related to storage limits and processing frequency.

Mapped Requirements: REQ-3, REQ-4, REQ-5

6.2.3 Sprint 3: Intelligence and Embedding Engine

Functionalities Covered:

- Vector embedding generation
- Similarity search engine
- Charging-aware batch processing

Rationale:

Embedding and similarity computation are resource-intensive and are isolated to allow focused optimisation and benchmarking under real mobile conditions.

Mapped Requirements: REQ-7, REQ-8, REQ-9

6.2.4 Sprint 4: Search, Retrieval, and Visualisation

Functionalities Covered:

- Keyword-based search
- Natural language search
- Timeline visualisation

Rationale:

This sprint integrates backend intelligence with user-facing components, ensuring that the interface reflects actual system capabilities and performance.

Mapped Requirements: REQ-6, REQ-15, REQ-16

6.2.5 Sprint 5: Optimisation, Testing, and Extensions

Functionalities Covered:

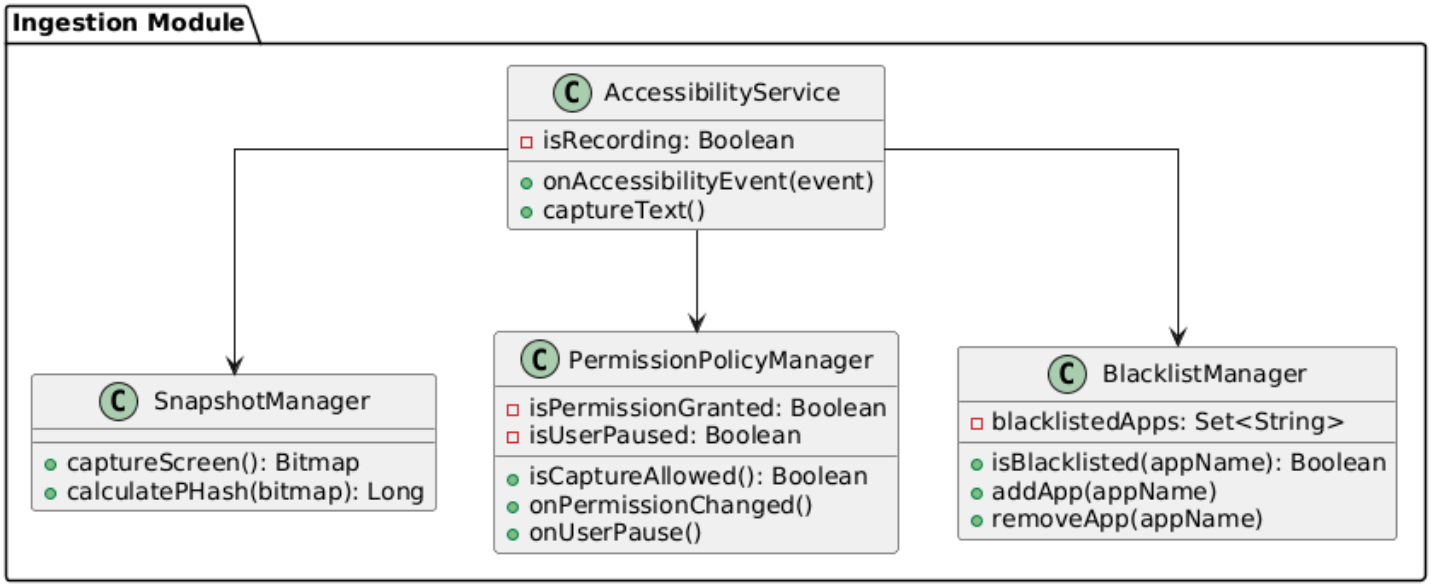
- Battery and storage optimisation
- Stress testing and validation
- Optional external LLM-based RAG support

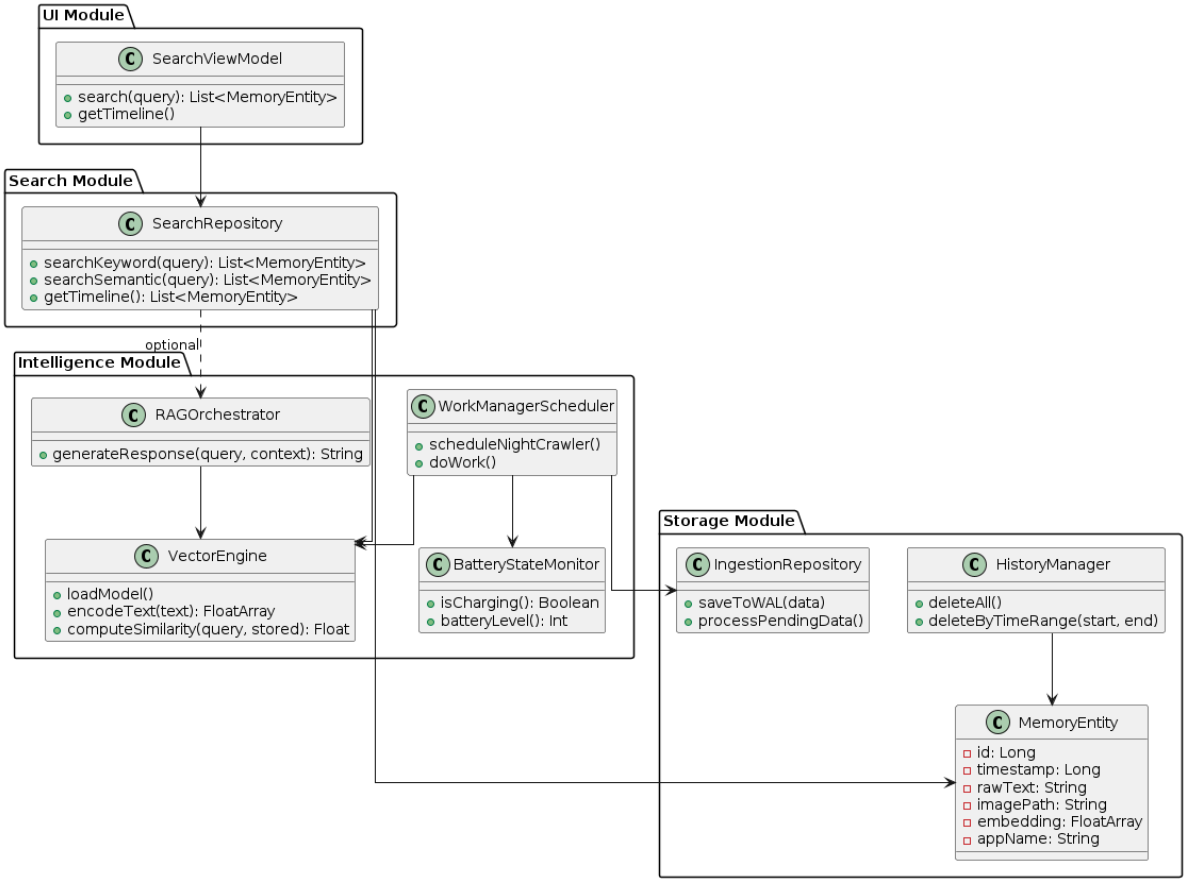
Rationale:

The final sprint focuses on fulfilling non-functional requirements such as performance, latency, and robustness, ensuring the system meets defined quality attributes.

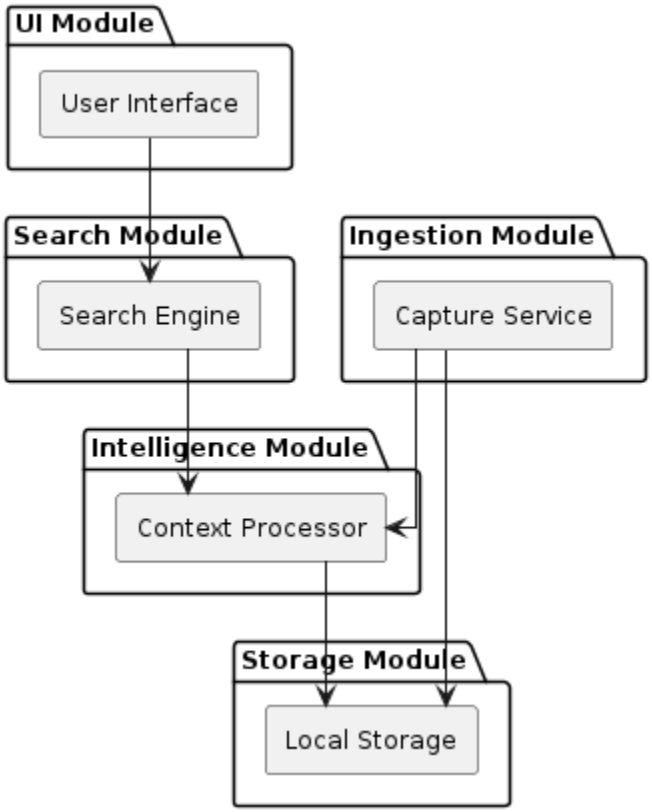
Mapped Requirements: REQ-10, Section 5

Appendix : Analysis Models

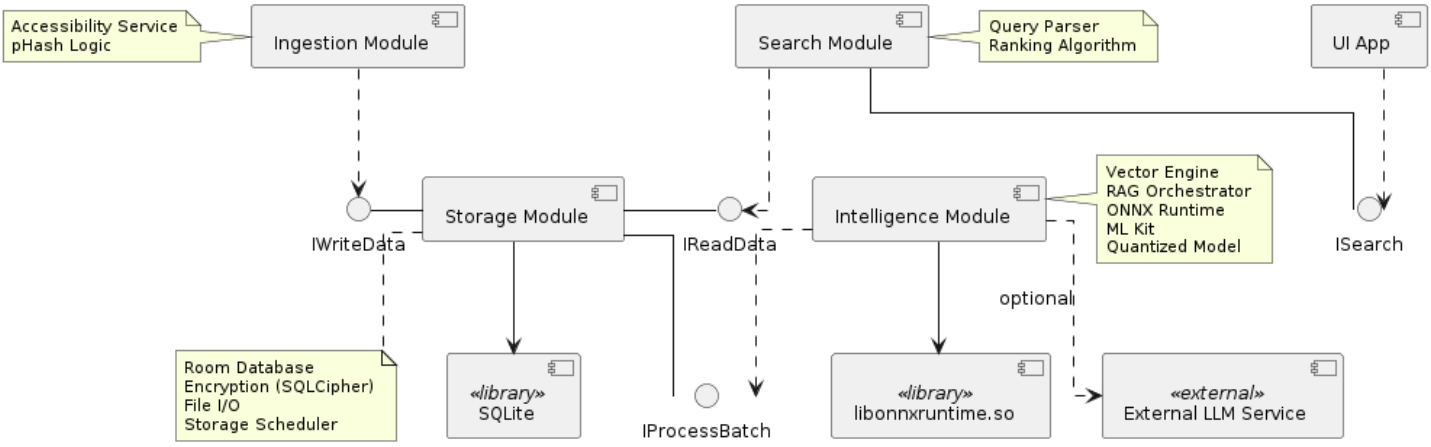


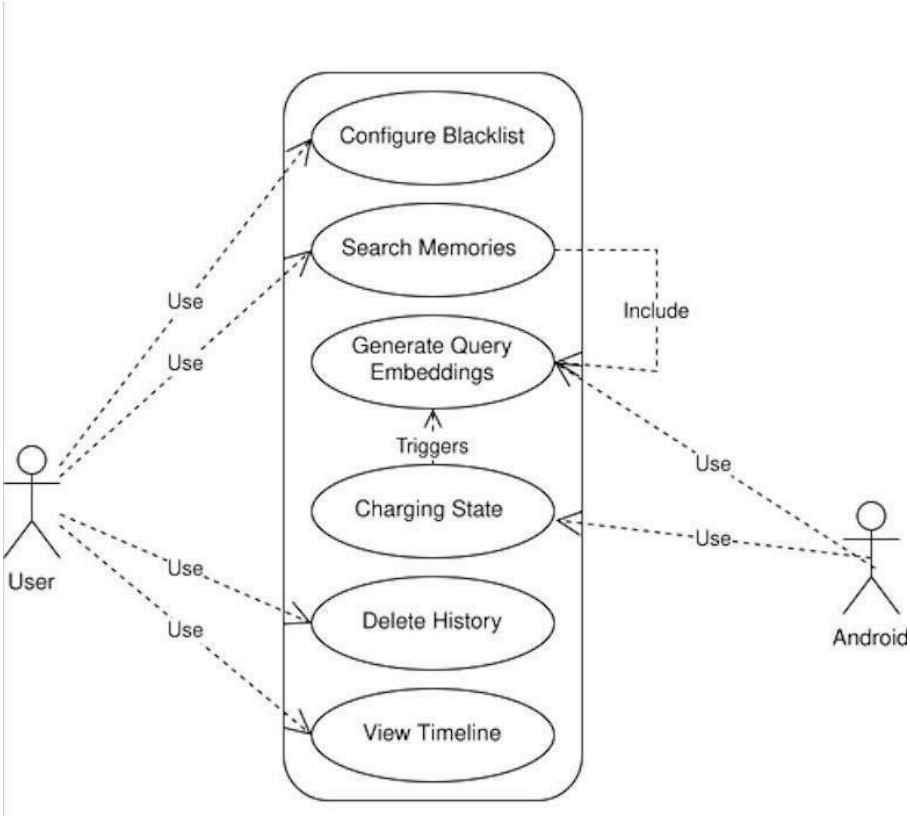


COMPOSITE UML DIAGRAM



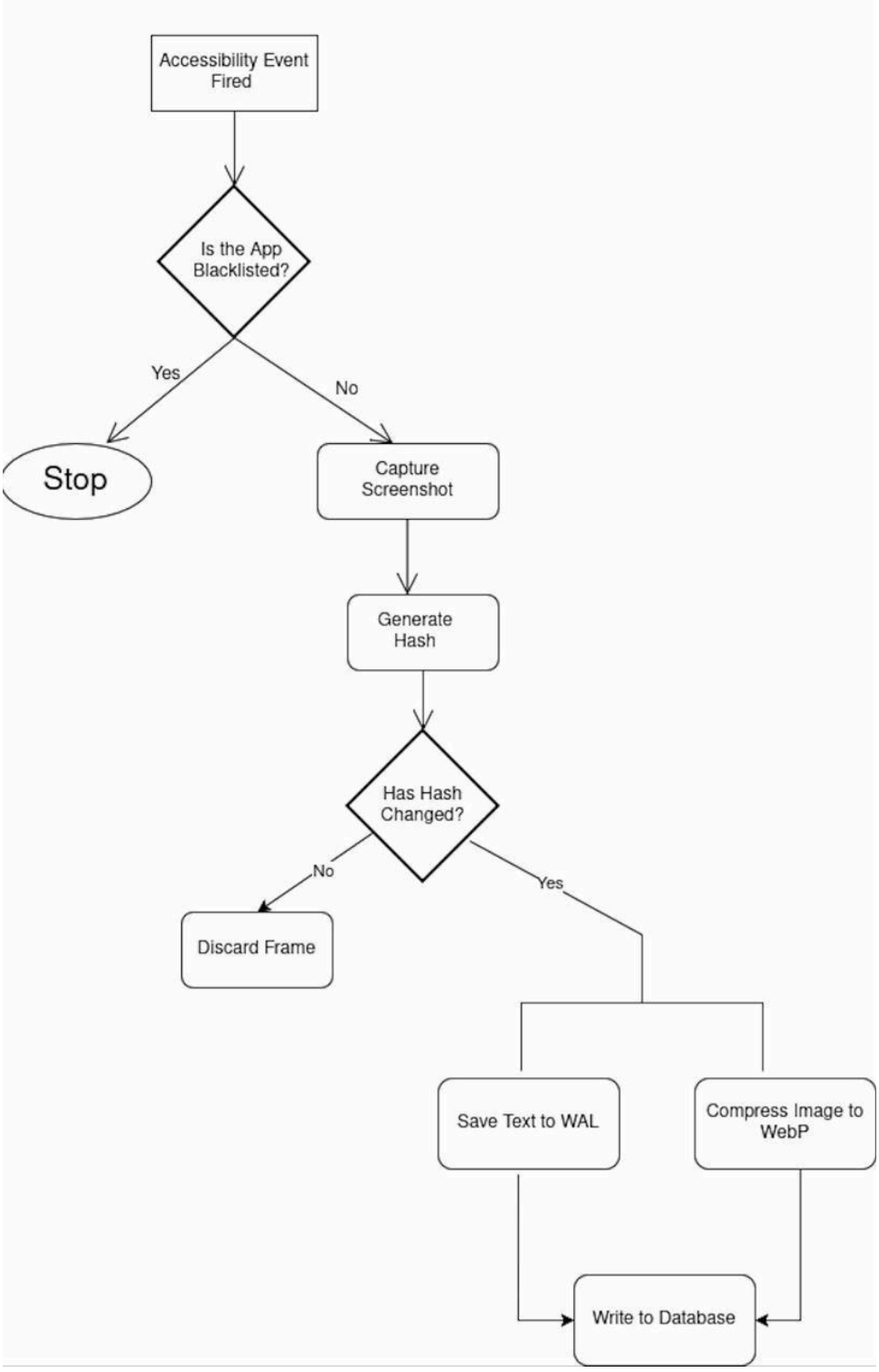
COMPONENT UML DIAGRAM



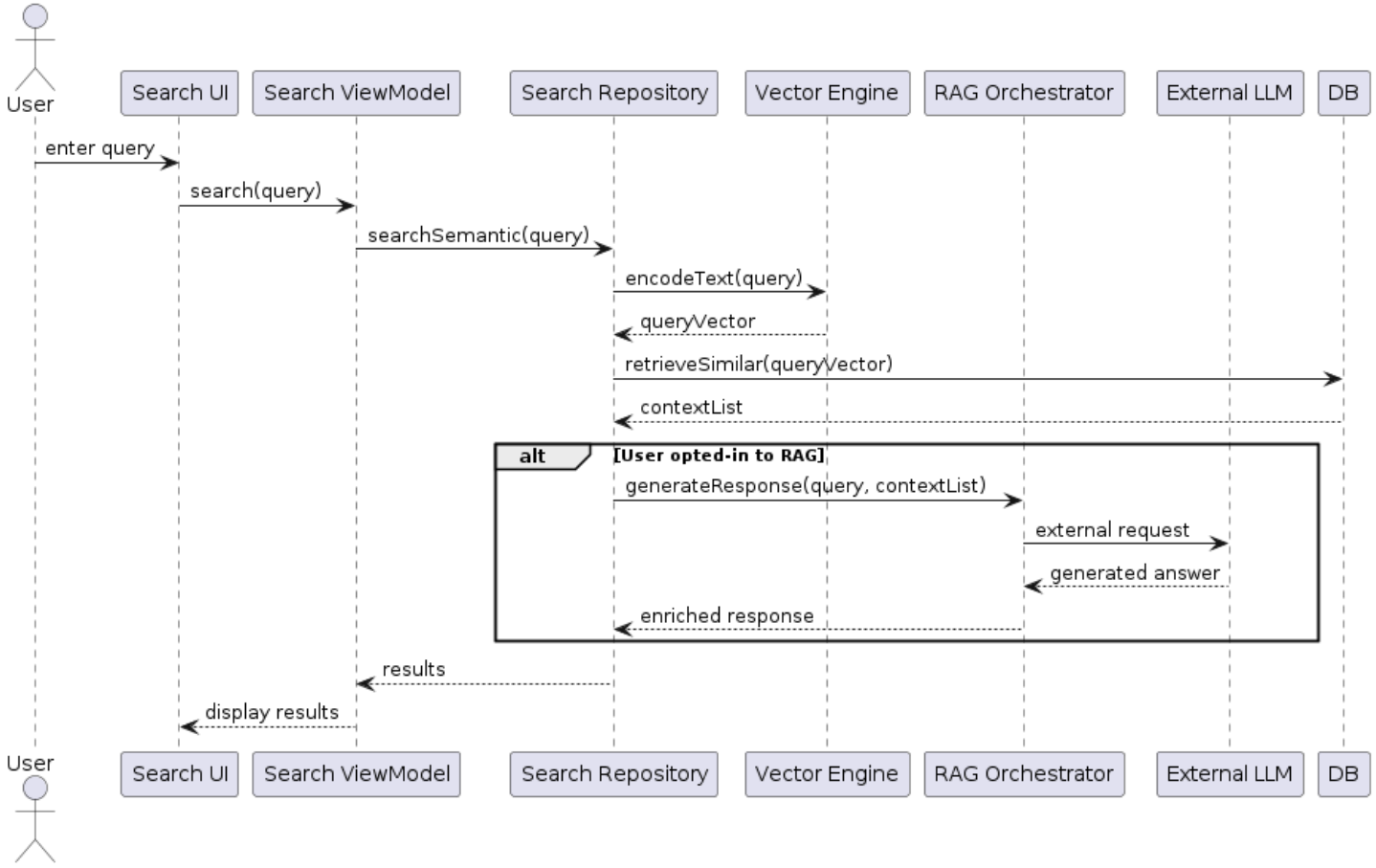


USECASE UML DIAGRAM

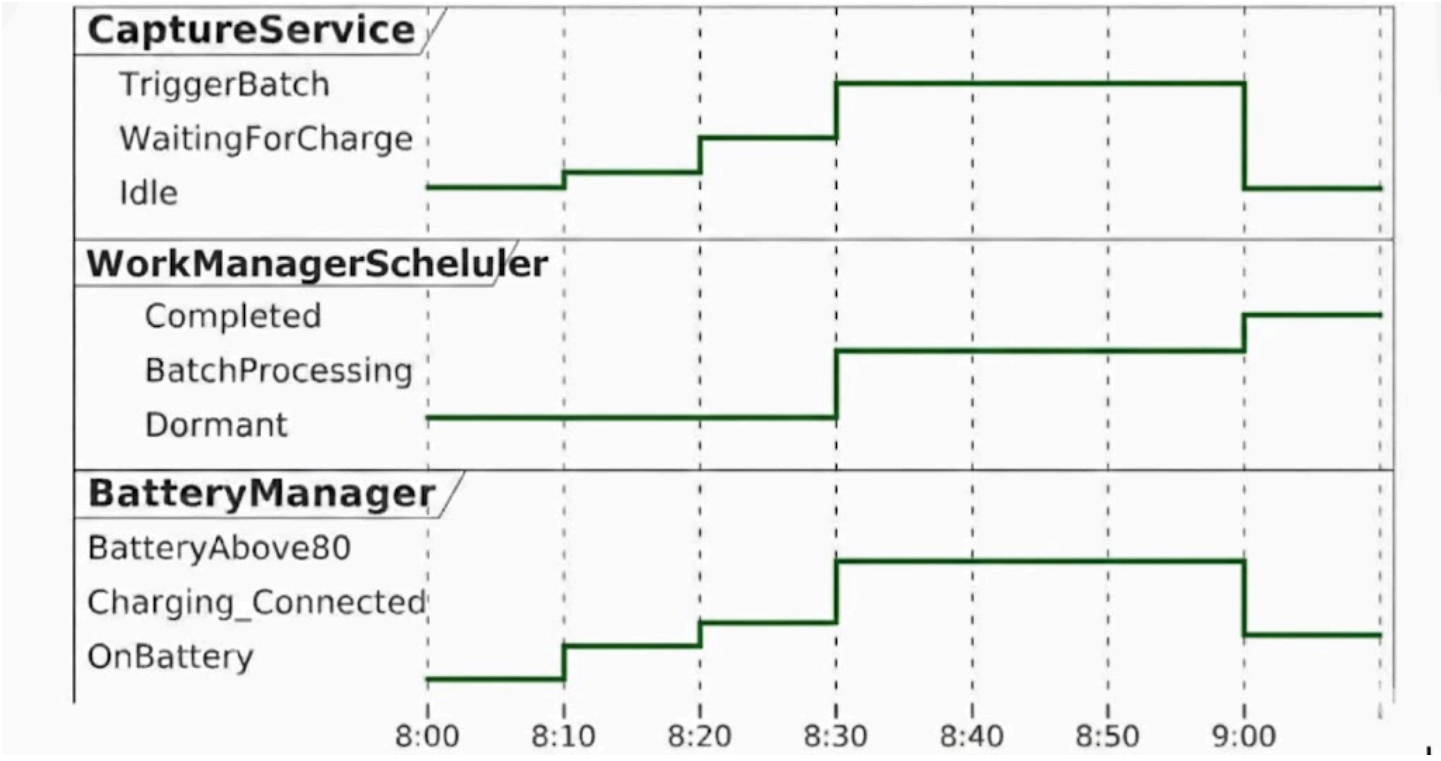
ACTIVITY UML DIAGRAM



SEQUENCE UML DIAGRAM



TIMING DIAGRAM



DEPLOYMENT UML DIAGRAM