SRS Capstone Group 2

Alex Buckley, Jeremy Gutierrez, Cole Manchester, Aty Sokoy, and Yonatan Yemiru

Central Washington University

January 2026

Document Information

Document Owner	Rental Project
Issue Date	January 9, 2026
Last Saved Date	—
File Name	Software Requirements Specification

Document History

Version	Issue Date	Changes
1.0	Jan 9, 2026	SRS created
1.1	Jan 9, 2026	Introduction (Purpose, scope)
1.2	Jan 15, 2026	All actors and features identified
1.3	Jan 21, 2026	Revised scope and implemented use cases
1.4	Jan 23, 2026	First Complete Document

Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to provide a description of the features, behavior, and constraints of the HOSE Rental (Housing for Students in Ellensburg) Application project. This document serves as the blueprint for the HOSE (Housing for Students in Ellensburg) Rental team and the client, ensuring a shared understanding of the system's design, APIs, functional requirements, and non-functional expectations. It acts as a single reliable reference for all system requirements, and any changes to the application's scope or features must be reflected and updated within this document.

1.2 Scope of the Project

Table 1: Features In Scope

In Scope	
Feature	**Description**
Interactive Map	Interactive map of Ellensburg showing all properties.
Walking Distance	Showing walking distance calculation from user-selected locations to each property or from the CWU campus.
Public Transit Information	Public transit information relevant to each property, such as routes, bus stops, and travel time.
Searchable Database	Containing available rental options in the Ellensburg area.
Property Details	Lease agreement summary Monthly rent Number of rooms Pet policy
Comparison Tool	Allowing users to compare properties by their prices, number of rooms, distance, etc.
AI Assistant (LLM-Based)	Summarizing user reviews Answering user queries Provide rental history

Table 2: Not In Scope

Not in Scope	
Feature	**Description**
Payment Processing	The application and backend services must remain free.
Real-time Rental Availability Update	The application will not provide real-time availability updates from landlords or property management systems.

Landlord/Property Management Dashboard	Unique views or management tabs for the Ellensburg properties.
User/Admin Accounts	Because HOSE is not storing emails or passwords, there will be no need for account creation.

1.3 Glossary

Table 3: Glossary of Terms

Term	Definition
API	Set of protocols that allow different software systems to communicate and interact with each other.
Test	Ensures the functionality of an API or service, verifying that it behaves as expected.
Prototype	An early functional model of a product or feature, used to visualize and test concepts before full development.
Payload	Data being sent or received by the API, often containing information relevant to a request or response.

1.4 References

Table 4: References

Name	Source (Link)	Description
Google Maps	Google Maps	We use it as a reference for UI design, interactive maps, and navigation features.
LLM with Geospace	https://doi.org/10.3390/ijgi13100348	Refer to how an AI assistant can be integrated and how it has already been applied in geospatial systems.

2 Use Case Diagram

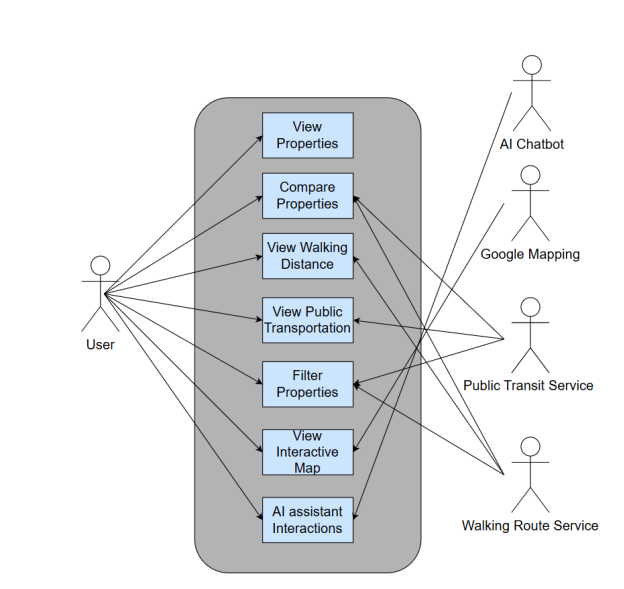

Figure 1: HOSE Use Case Diagram

3 Use Case Descriptions

3.1 Use case 1 - View Properties

3.1.1 Description

Use Case ID: ViewProperties-01

Priority: High

Source: Functional Requirements

Primary Actor: User

Secondary Actors: N/A

Description: Allows users to view properties within the Ellensburg area. Each property includes data and filter options.

Preconditions: User can access the HOSE landing page.

Trigger: User is viewing the interactive map.

Main Flow:

- User is on the HOSE landing page.
- User clicks the interactive map feature.
- The system displays current known properties available.

Alternative Flows:

- User selects a specific property to view detailed apartment or home information.
- User exits without viewing any property details.

Postconditions: Properties are displayed to the user.

Conclusion: Users can view properties and contextual information about those properties within Ellensburg.

Business Rules: The system must respond within 5 seconds.

Implementation Constraints: Property data shall be stored using SQLite. HOSE accesses the database for each requested location to display relevant property data.

Assumptions: All location data is up to date for the year 2026.

3.2 Use case 2 - Compare Properties

3.2.1 Description

Use Case ID: CompareProperties-01

Priority: High

Source: Functional Requirements

Primary Actor: User

Secondary Actors: AI Chatbot

Description: Allows users to compare property details within the Ellensburg area. Selected properties are displayed side-by-side with highlighted data points for comparison.

Preconditions: User can access the HOSE landing page and has one property already selected.

Trigger: User selects a new property location to compare with the currently selected property.

Main Flow:

- User selects a new property location on the map.
- The system displays data for both the originally selected property and the newly selected property.
- The system highlights the data points being compared on the map.

Alternative Flows:

- User selects a property to view its data; the system displays the selected property information.
- User selects a second property to compare; the system displays the second property data adjacent to the original property.
- User selects a third property; the system deselects the original property and compares the second and third selected properties.

Postconditions: A new comparison property is selected and displayed.

Conclusion: Users can compare details between properties located in Ellensburg.

Business Rules: All location data must be loaded within 5 seconds.

Implementation Constraints: The system allows comparison of a maximum of two properties at a time.

Assumptions: Users will compare different locations and are willing to wait for data to load.

3.3 Use case 3 - View Walking Distance

3.3.1 Description

Use Case ID: ViewDistance-01

Priority: High

Source: Functional Requirements

Primary Actor: User

Secondary Actors: Walking Service

Description: Allows users to view properties within the Ellensburg area. Each property includes data, filter options, and walking distance information to CWU and nearby public transport access locations.

Preconditions: Users can access the Rental Assistant Tool landing page.

Trigger: User selects an individual property page to view detailed property info.

Main Flow:

- User selects a property to view detailed info.
- The system queries the database for the selected property's data.
- The system displays walking distance and time from the property to CWU and nearby public transport access locations.

Alternative Flows:

- **Property data missing or corrupted:**
  - User selects property to view detailed info.
  - System queries database for selected property data.
  - System does not find valid property data.

- System displays "Property not found" error.

- **Walking data missing from property:**
  - User selects property to view detailed info.
  - System queries database for selected property data.
  - System finds property data with walking data missing.
  - System displays property without walking data.
- **Site admin uploads new property successfully:**
  - Site admin uploads new property.
  - System validates property data.
  - System queries Walking Service for walking data to CWU and bus stop locations.
  - Walking Service responds with walking distance and time.
  - System saves walking data with property.
  - System informs site admin of successful upload.
- **Walking service fails during property upload:**
  - Site admin uploads new property.
  - System validates property data.
  - System attempts to query Walking Service a set number of times.
  - Walking Service fails to respond successfully.
  - System saves property with walking data missing.
  - System informs site admin of upload walking data failure and property success.
  - System runs scheduled CRON jobs to query Walking Service for properties with walking data missing.

Postconditions: Property data has been retrieved and displayed (or error shown). No property data is modified.

Conclusion: User has viewed walking time and distance from the selected property to CWU and nearby bus stops.

Business Rules: Only properties within Ellensburg, applicable to CWU students, will be displayed. Walking distance is calculated using pedestrian-friendly routes only.

Implementation Constraints: Walking distance and time must be retrieved from the Walking Service API. Property data must be retrieved from the system's property database.

Assumptions: Users have a stable internet connection and a device suitable for running client software. The Walking Service will eventually work if an error is encountered.

3.4 Use case 4 - View Public Transportation

3.4.1 Description

Use Case ID: ViewTransportation-01

Priority: High

Source: Functional Requirements

Primary Actor: User

Secondary Actors: Public Transit Service

Description: Allows users to view public transit options in the nearby area for each property location.

Preconditions: Users can access the HOSE landing page.

Trigger: User initiates viewing public transportation data.

Main Flow:

- User clicks on public transportation data.
- User selects two destinations to travel between.
- HOSE performs a call to Google Maps Transit API with travelMode=Transit.
- System receives the transit schedule and map display.
- System displays the map with the transit path.

Alternative Flows:

- **User edits travel preference:**
  - User edits travel preference to walking instead of transit.
  - HOSE automatically updates the map display and travel path with the appropriate method of travel.

Postconditions: The user has an understanding of how far they are from transit, home, school, and other locations.

Conclusion: The user has transportation data to help them choose the right apartment.

Business Rules: Users have reliable transportation to multiple destinations.

Implementation Constraints: Users will see the closest transit center to the selected apartment first.

Assumptions: Users will search for one transit center at a time per apartment.

3.5 Use case 5 - Filter Properties

3.5.1 Description

Use Case ID: ViewProperties1

Priority: High

Source: Functional Requirements

Primary Actor: User

Secondary Actors: N/A

Description: Allows users to view and filter properties within the Ellensburg area. Each property includes data and multiple filter options.

Preconditions: User can access the Rental Assistant Tool landing page.

Trigger: User selects at least one filter option on the filter modal while using the interactive map.

Main Flow:

- User navigates to the interactive map.
- System retrieves viewable properties from the database.

- System displays properties on the interactive map.
- User selects one or more fields to filter from the filter modal.
- System filters properties by all selected fields.
- System displays only filtered properties on the interactive map.

Alternative Flows:

- **User resets filter options:**
  - User is viewing the map with one or more filters applied.
  - User selects "remove all filters" from the filter modal.
  - System displays all properties within the current viewport.
- **User updates or modifies filters:**
  - User removes, adjusts values, or adds new filters in the filter modal.
  - System filters all properties by the user-selected filters.
  - System displays the updated collection of filtered properties.
- **No properties match all filter requirements:**
  - User selects filters from the filter modal.
  - System filters properties by all selected fields.
  - No properties pass all filter criteria.
  - System displays the interactive map with no properties.
  - System informs the user that no properties match the selected filters.

Postconditions: Property data has been retrieved and displayed on the interactive map. Filtered-out properties will not be displayed. No property data is modified.

Conclusion: User sees all properties passing filter criteria within the viewport on the interactive map. If no property meets the criteria, the user is informed.

Business Rules: Only properties within Ellensburg, applicable to CWU students, will be displayed. Filterable fields include, but are not limited to, minimum rent price, maximum rent price, lease type, pets allowed, walking distance to CWU, and walking distance to the nearest bus stop.

Implementation Constraints: Property data must be retrieved from the system's property database. Properties must be displayed at the correct location on the interactive map.

Assumptions: Users have a stable internet connection and a device suitable for running client software.

3.6 Use case 6 - View Interactive Map

3.6.1 Description

Use Case ID: ViewProperties1

Priority: High

Source: Functional Requirements

Primary Actor: User

Secondary Actors: Mapping Service

Description: Allows users to view properties and other relevant locations on an interactive map within the Ellensburg area. Each property includes data and filter options, and map locations such as bus stops are overlaid dynamically.

Preconditions: User can access the Rental Assistant Tool landing page.

Trigger: User navigates to the interactive map page.

Main Flow:

- User navigates to the interactive map page.
- System queries the database for locations, including properties and bus stops.
- System requests map area from the Map Service.
- Map Service responds with map data.
- System displays received map data.
- System overlays locations from the database on the map.
- User pans or scrolls the interactive map.
- System hides and displays locations as they come in and out of the viewport.
- System requests additional map data from Map Service if viewport bounds exceed the loaded area.
- Map Service responds with additional map data.
- System displays new map data and overlays locations from the database.
- User clicks a property on the interactive map.
- System queries the property database for detailed property data.
- System displays detailed data of the selected property.

Alternative Flows:

- **User pans to an area with no properties:**
  - User pans outside of Ellensburg.
  - System queries the property database for locations.
  - System displays no locations, as none exist within the viewport.
- **Map Service fails to respond with valid data:**
  - User loads or pans the map.
  - System queries the database for locations and requests map data from the Map Service.
  - Map Service fails to respond after a set number of retries.
  - System informs the user that the map failed to load.
  - System displays properties as a list, still allowing access to individual property detail pages.
  - System intermittently retries requests to the Map Service.

Postconditions: User has sufficient data to compare multiple properties on the interactive map.

Conclusion: Users can explore and select properties from the interactive map to view detailed information.

Business Rules: Only properties within Ellensburg, applicable to CWU students, will be displayed. Bus stops within Ellensburg will also be displayed.

Implementation Constraints: Location data must be received from the system's database. Map data must be received from the Map Service. Locations must be displayed at the correct coordinates.

Assumptions: Users have a stable internet connection and a device suitable for running client software. The Map Service will eventually respond if an error is encountered.

3.7 Use case 7 - AI Assistant Interactions

3.7.1 Description

4 Functional Requirements

Table 5: Functional Requirements

Identifier	Description
FR-ViewProperties-01	The system should display list/grid available Ellensburg rental properties.
FR-ViewProperties-02	The system should display properties address for each listing.
FR-ViewProperties-03	The system should display monthly rent for each property.
FR-ViewProperties-04	The system should display properties room type (1 Bedroom).
FR-ViewProperties-05	The system should display each property's pet policy.
FR-ViewProperties-06	The system should display each property's lease type.
FR-ViewProperties-07	The system should display each property manager's numbers.
FR-ViewProperties-08	The system should display each property's distance from CWU campus.
FR-ViewProperties-09	The system should display property detail photo on detail card.
FR-ViewProperties-10	The system should display "No provided" for missing data field.
FR-Viewproperties-11	The system should support the application responsive layout.
FR-Viewproperties-12	The system should handle errors and inform the user
FR-Viewproperties-13	The system should fetch all available properties from the API.
FR-CompareProperties-01	The system should allow users to add properties to a comparison list.
FR-CompareProperties-02	The system should allow users to remove properties from comparison list.
FR-CompareProperties-03	The system should show "Compare now" action button when properties are selected.
FR-CompareProperties-04	The system should display a compare badge showing the number of selected properties.
FR-CompareProperties-05	The system should display a side-by-side comparison of the properties.
FR-ViewWalkingDistance-01	The system should use CWU campus as default origin for distance calculation.
FR-ViewWalkingDistance-02	The system should display estimated waking distance (in miles) for each property.
FR-ViewWalkingDistance-03	The system should display estimated waking time (in minutes) for each property.
FR-ViewWalkingDistance-04	The system should handle walking distance error and inform the user
FR-ViewWalkingDistance-05	The system should fetch walking-distance data from the API or routing service.
FR-ViewTransporation-01	Thes system should list the nearest public transit stop to each property and show the distance.
FR-ViewTransporation-02	The system should show all available bus route numbers connecting the property to CWU campus.
FR-ViewTransporation-03	The system should display estimates of public transit travel time to CWU campus.

Identifier	Description
FR-ViewTransporation-04	System should notify the user when the transit data is unavailable.
FR-ViewTransporation-05	The system should fetch transit information from the API or transit data source.
FR-ViewTransporation-06	The system should handle transitrelated errors and inform the user.
FR-FilterProperties-01	The system should provide filters for price range.
FR-FilterProperties-02	The system should provide filters for bedroom numbers.
FR-FilterProperties-03	The system should provide filters for pet policy.
FR-FilterProperties-04	The system should provide filters for property type.
FR-FilterProperties-05	The system should provide filters for lease term.
FR-FilterProperties-06	The system should update the result list immediately upon filter change.
FR-FilterProperties-07	The system should update the map pins immediately upon filter change.
FR-FilterProperties-08	The system should provide “Clear filter” option.
FR-FilterProperties-09	The system should provide individual filter reset control.
FR-FilterProperties-10	The system should handle filter-related errors and inform the user.
FR-InteractiveMap-01	The system should fetch map-related property data from the API.
FR-InteractiveMap-02	The system should display an interactive map of Ellensburg
FR-InteractiveMap-03	The system should allow map zooming.
FR-InteractiveMap-04	The system should display property pins on the map.
FR-InteractiveMap-05	The system should handle map-related errors and inform the user.
FR-AIAssistance-01	The system should allow users to type natural language queries.
FR-AIAssistance-02	The system should translate natural language queries into filter or action.
FR-AIAssistance-03	The system should support property comparison through chat.
FR-AIAssistance-04	The system should answer questions regarding the properties.
FR-AIAssistance-05	The system should provide non-legal education guidance with link to source.
FR-AIAssistance-06	The system should use current filters and selected properties to personalize responses.
FR-AIAssistance-07	The system should avoid giving legal or financial advice.
FR-AIAssistance-08	The system should include disclaimers for AI-generated content.
FR-AIAssistance-09	The system should avoid generating sensitive or personal information.
FR-AIAssistance-10	The system should fetch AI-related property or context data from the API.
FR-AIAssistance-11	The system should handle AI-related errors and inform the user.

5 Non-Functional Requirements

Table 6: Non-Functional Requirements

Identifier	Description
NFR-ViewProperties-01	The property list should be loaded within 2 seconds under normal network conditions.
NFR-ViewProperties-02	Property photos and details should be rendered without noticeable layout shifts.
NFR-ViewProperties-03	The system should display property detail apply/schedule tour link
NFR-ViewProperties-04	The system should handle up to 5000 property records without performance degradation.
NFR-ViewProperties-05	The system should display error messages for property loading within 1 second or detection.
NFR-ViewProperties-06	The system should display review.
NFR-ViewProperties-06	The system should allow users to save/ unsaved properties.
NFR-CompareProperties-01	The system should visually highlight differing values between properties.
NFR-CompareProperties-02	The system should allow users to export the comparison list as pdf.
NFR-CompareProperties-03	The system should allow users to save a named comparison set.
NFR-CompareProperties-04	The comparison table should display within 1 second after selecting “Compare now.”
NFR-ViewDistance-01	The system should allow users to change origin by typing the address.
NFR-ViewDistance-02	The system should allow users to change origin by pinned map point.
NFR-ViewDistance-03	The system should have “view route” button.
NFR-ViewTransporation-01	The system should display Walking distance to the nearest transit stop.
NFR-ViewTransporation-02	The system should display transfer information.
NFR-ViewTransporation-03	The system should predict real-time live delay.
NFR-ViewTransporation-04	The system should provide transit lines and stop in map.
NFR-FilterProperties-01	The system should be provided “sort by”.
NFR-FilterProperties-02	Filter operations should not require page reloads and must update smoothly.
NFR-FilterProperties-03	If filters fail to apply, the fallback list should display within 1 second.
NFR-InteractiveMap-01	The map must load and become interactive within 1 seconds.
NFR-InteractiveMap-02	Map panning and zooming must maintain at least 30 FPS.
NFR-InteractiveMap-03	The system shall not support streetlevel satellite imagery.
NFR-InteractiveMap-04	The system shall not support streetlevel 3D map view.
NFR-InteractiveMap-05	If map data fails to load, a fallback “Unable to load map” message must appear within 1 second.
NFR-AIAssistance-01	The system AI responses should generate within 2 seconds for typical queries

6 API Interfaces

6.1 Map and Distance API Interface

Description:

Provides geographic data, walking distances, and estimated travel times between rental properties and key locations such as Central Washington University and local hospitals.

Actors Involved:

- HOSE Rental Application
- External Map Service API

Inputs:

- OriginLocation – String – Address or latitude/longitude of the starting point
- DestinationLocation – String – Address or latitude/longitude of the destination
- TravelMode – String – Mode of travel (walking, transit)

Outputs:

- Distance – Float – Distance between locations in miles or kilometers
- EstimatedTime – Integer – Estimated travel time in minutes
- RouteData – JSON Object – Encoded route information for map rendering

Error Conditions:

- Invalid address input
- API request limit exceeded
- Network failure

6.2 AI Assistant API Interface

Description:

Enables natural-language interaction with users by processing rental-related queries and generating informational responses based on application data.

Actors Involved:

- HOSE Rental Application
- External LLM Service

Inputs:

- UserQuery – String – Natural language question submitted by the user
- ContextData – JSON Object – Relevant rental data retrieved from the database

Outputs:

- AIResponse – String – Generated natural language response

Error Conditions:

- Invalid or empty user query
- LLM service unavailable
- Response timeout

6.3 Property Database API Interface

Description:

Retrieves and manages rental property information stored in the system database.

Actors Involved:

- HOSE Rental Application
- Property Database

Inputs:

- SearchCriteria – JSON Object – Filter parameters such as price range and number of rooms
- PropertyID – Integer – Unique identifier for a rental property

Outputs:

- PropertyList – JSON Array – List of properties matching search criteria
- PropertyDetails – JSON Object – Detailed information for a specific property

Error Conditions:

- Database connection failure
- Invalid search parameters

6.4 Interface Data

Table 7: Data Items Sent to the API

Datatype and Size / Field	**Comments**
Places API	
Query	String (255 char) - Search itemed
Location	LatLng (Float, Float) - City to zone for
radius	String - Search radius
type	String - Place type
Key	String - Google Maps API Key
Geocoding API	
address	String - Full address
region	String (2 char) - Country code
Key	String
Routes & Transit API	
Origin.latLng	Object (2 floats) - Start point
Destination.latLng	Object (2 floats) - End point
travelMode	Enum - Set to transit for route calculations and public transit availability.
departureTime	ISO-8601 DateTime - Default to current time
routingPreference	Enum - Alter course based off preferences

Datatype and Size / Field	Comments
key	String - Google Maps API Key for transit

Table 8: Data Items Returned by the API

Datatype and Size / Field	Comments
Places API	
Place_id	String - Unique ID
Name	String - Property name
Formatted_address	String - Address in readable form
Geometry.location.lat	Float - Latitude
Geometry.location.lng	Float - Longitude
types	Array[String] - Place categories
rating	Float (0-¿5) - Rating for property, unused for now
User_ratings_total	integer - Ratings for property unused for now
Geocoding API	
Formatted_address	String
address	
Place_id	String - Google's place reference
Address_components	Array - City, state, postal code
Routes & Transit API	
Routes[].duration	String - Total travel time
Routes[].distanceMeters	Integer - Distance (meters)
Legs[].steps[].travelMode	Enum - Walk vs Transit
TransitDetails.line.vehicle.type	Enum - Bus, Subway, or Train
TransitDetails.line.name	String - Transit line name
DepartureStop.name	String - Ending Stop
ArrivalStop.name	String - Starting Stop
DepartureTime	DateTime - Scheduled transit time
Polyline.encodedPolyline	String - Path for map display

7 LLM Model

7.1 AI Model Selection and Deployment

Model Selection:

The HOSE Rental application will utilize an open-source Large Language Model (LLM) from the LLaMA (Large Language Model Meta AI) family to power the AI assistant functionality.

Rationale:

The LLaMA model was selected due to its open-source availability, zero per-request cost, and suitability for lightweight natural-language tasks such as answering user questions, summarizing rental information, and explaining property attributes. This approach allows the system to operate without reliance on third-party paid APIs and ensures full control over model deployment and data usage.

Deployment Environment:

The LLaMA model will be hosted locally on project-owned computing resources. The model will run as a backend service accessible by the HOSE Rental application through a local API interface.

Capabilities:

- Interpret natural-language user queries related to rental housing.
- Generate informational responses based on data supplied by the application.
- Summarize user reviews and rental property details.
- Assist users in comparing rental options using descriptive language.

Limitations:

- The AI assistant will not provide legal, financial, or contractual advice.
- The model will not independently calculate distances or access real-time external data.
- All factual information used by the model must be supplied by the application database or supporting APIs.

Data Privacy and Control:

Because the LLaMA model is hosted locally, no user queries or system data are transmitted to external AI service providers. This ensures improved privacy, reduced latency, and compliance with academic project constraints.

Future Scalability:

The system architecture allows for future replacement or augmentation of the LLaMA model with alternative LLMs or cloud-based services if scalability or performance requirements change.

8 Non-Functional Requirements

8.1 Hardware Requirements

Local LLM Hosting:

The HOSE Rental application shall support execution of a locally hosted LLaMA-based Large Language Model (LLM) on project-owned computing hardware without reliance on external cloud resources.

Minimum Hardware Requirements:

- CPU: Quad-core 64-bit processor or equivalent
- System Memory (RAM): Minimum 16 GB

- Storage: Minimum 20 GB of available disk space for model files and application data
- Operating System: Windows, macOS, or Linux with support for local backend services

Recommended Hardware Requirements:

- CPU: Multi-core processor (8 cores or higher recommended)
- System Memory (RAM): 32 GB or more
- GPU (Optional): Dedicated GPU with a minimum of 8 GB VRAM to improve response latency

Justification:

The selected LLaMA model variant is intended for lightweight natural-language processing tasks, such as informational question answering and summarization. These tasks do not require large-scale computational resources or model training. The listed hardware requirements are achievable using standard consumer-grade or academic lab computers and are sufficient to support acceptable response times for a limited number of concurrent users.

Scalability Considerations:

The system is designed for small-scale academic usage. Increased user demand may require additional hardware resources or migration to a cloud-based LLM service in future iterations.