Fall 2025-2026

CS491 - Senior Design Project

Project Specification Report

T2526

Ahmed Hatem Haikal - 22001482

Amirhossein Ahani - 22101535

İrfan Hakan Karakoç - 22003421

Mehmet Hakan Yavuz - 22002119

Türker Köken - 22102331

Supervisor: Anıl Koyuncu

Innovation Expert: Haluk Altunel

Contents

1. Introduction **5**
1.1. Description 5
1.2. High Level System Architecture & Components of Proposed Solution 5
1.2.1. Frontend Layer 5
Consensus Displayer 5
Persona Customizer 6
Task Customizer 6
1.2.2. Backend Layer 6
Consensus Controller 6
Persona Creator 6
Task Creator 6
1.2.3. API Layer 7
LLM API 7
Data Scraping API 7
1.2.4. Storage Layer 7
Prompt Created Personas 7
Scraped Data Created Personas 7
Scraped Data 7
1.3. Constraints 7
1.3.1. Implementation Constraints 7
Dependency on External LLM Providers 8
Asynchronous Multi-Agent Execution Complexity 8
Frontend–Backend Contract 8
Limited Local Storage and Persona Management 9
Scraping & Data Processing Constraints 9
Local Hardware & Development Environment 9
Security & Privacy Safeguards 10
Limited Team Size and Development Timeline 10
1.3.2. Economic Constraints 10
Limited Budget 10
Hardware Overhead 10
Cost of API Integrations 11
Licensing 11
1.3.3. Ethical Constraints 11
1.4. Professional and Ethical Issues 11
1.4.1. Data privacy and Consent 11
1.4.2. Bias Possibility and Fair Representation 12
1.4.3. Misinterpretation of Consensus 12
1.4.4. Accountability and Transparency 12
1.4.5. Ethical Persona Simulation 12
1.5. Standards 13
2. Design Requirements **13**

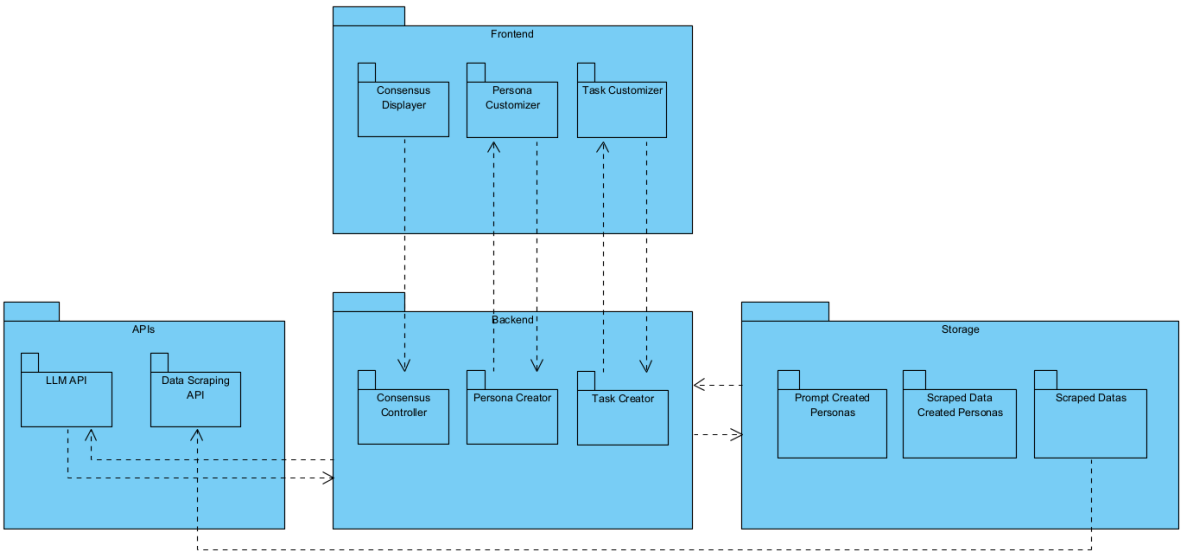
2.1. Functional Requirements 13
2.2. Non-Functional Requirements 14
2.2.1. Usability 14
2.2.2. Reliability 14
2.2.3. Performance 14
2.2.4. Supportability 15
2.2.5. Scalability 15
3. Feasibility Discussions **16**
3.1. Market & Competitive Analysis 16
3.1.1. Existing Tools and Rival Companies: 16
3.1.2. Competitive Gap: 17
3.2. Academic Analysis 18
4. Glossary **19**
5. References **20**

1. Introduction

1.1. Description

This project aims to create customizable large language model personas [1], [2] that can be used at completing any sort of task regarding Software Engineering. Our application will provide an environment for users to create their own tasks, allowing them to test the reliability of the application by comparing the results to a real dataset.

1.2. High Level System Architecture & Components of Proposed Solution



1.2.1. Frontend Layer

Consensus Displayer

Displays the consensus created by the cooperation of backend design and LLM API.

Persona Customizer

Allows users to create personas using different methods. Users can create personas just using prompt, they can choose to upload CV or they can use data scraping method to create more detailed persona. Users can modify the existing personas' names. Users can also remove personas which will trigger the backend to delete personas from the database.

Task Customizer

Allows users to upload files and define the task, which then will be asked to personas to complete.

1.2.2. Backend Layer

Consensus Controller

Creates the correct environment and details for what judge LLM should care about when deciding the consensus of the discussion.

Persona Creator

Connects personas from the frontend to the database when changing, modifying or removing personas.

Task Creator

Connects tasks from the frontend to personas for them to perform the task.

1.2.3. API Layer

LLM API

Allows receiving and sending of data between LLM and backend [1], [2].

Data Scraping API

Allows scraping data from the internet to have a database for persona creation [3].

1.2.4. Storage Layer

Prompt Created Personas

Stores all prompt based created personas.

Scraped Data Created Personas

Stores all scraped data based created personas.

Scraped Data

Stores all scraped data from the internet, then to pass it to create personas perhaps multiple times.

1.3. Constraints

1.3.1. Implementation Constraints

The implementation of Consensia is influenced by several technical limitations stemming from architectural decisions, technology stack, and available computational resources. These constraints define what is realistically achievable in the development timeline and influence both system performance and design choices.

Dependency on External LLM Providers

The system relies heavily on external APIs such as Google Gemini [2], OpenAI [1], and potentially other third-party AI services.

Because these models run fully on cloud endpoints rather than locally:

- Stable internet connectivity is required for all debate and persona generation processes.

- Backend processing speed is limited by API response times, rate limits, and quota.

- Any change in API availability, pricing, model versions, or request/response formats may require code refactoring.

Asynchronous Multi-Agent Execution Complexity

The debate pipeline requires:

- Multiple persona prompts,
- Sequential message passing,
- Judge evaluation,
- Asynchronous LLM calls.

similar to multi-agent LLM workflows studied in recent research [3], [4].

This introduces constraints such as:

- Increased complexity in request orchestration,
- Requirement for async-safe code on backend (FastAPI + asyncio),
- Risk of race conditions or partial failures if one persona fails to respond.

Frontend–Backend Contract

The system requires strong consistency between:

- Persona schemas,
- Task schemas,
- Consensus request formats.

Because the backend validates all input using Python Pydantic schemas [6] :

- Any minor schema modification requires synchronized updates on the frontend.
- Mismatches between versions can cause request rejections or failed debates.

Limited Local Storage and Persona Management

Persona creation from:

- Prompts,
- CVs,
- Scraped data

requires storage and indexing.

Constraints:

- Storing raw documents (CVs, scrape data) increases disk requirements.
- Persona regeneration from large scraped datasets may be slow.
- Database must maintain consistency between personas, tasks, and debate logs.

Scraping & Data Processing Constraints

The scraping API introduces [3]:

- Restrictions from website anti-bot measures,
- Rate limitations and CAPTCHAs,
- Ethical rules preventing scraping of restricted/private content,
- Inconsistent formatting in external sources requiring preprocessing.

This limits the speed and reliability of data-driven persona creation.

Local Hardware & Development Environment

Since multi-agent interactions can require multiple LLM calls per debate, development environments face limitations such as:

- Increased latency when running many personas,
- Long-running async tasks during testing,
- High costs and time delays when debugging LLM behavior.

Developers often need:

- Stable network,

- Sufficient CPU/RAM for running backend + frontend + development tools,
- Keep-alive servers to prevent timeouts during multi-round debates.

Security & Privacy Safeguards

Handling user-uploaded CVs and documents requires:

- Sanitizing all inputs,
- Preventing raw file execution,
- Ensuring no harmful content is passed between personas and judge.

These constraints limit:

- Full freedom of persona creation,
- File formats supported (e.g., PDF, DOCX but not executable files),
- Ability to store long-term raw personal data.

Limited Team Size and Development Timeline

As a student project with a fixed timeline:

- Implementation must favor modular, simple, maintainable components,
- Certain advanced features (distributed workers, cloud autoscaling, heavy scraping) must be simplified,
- Optimization is limited to what is feasible within academic deadlines.

1.3.2. Economic Constraints

Limited Budget

The project is conducted by a team of college students, limiting the budget of the project. The budget allocation for the project will be carefully planned, tracking and minimizing costs as necessary. We will also seek external funding from sources like ‘Microsoft for Startups’ [7].

Hardware Overhead

The application has a computationally intense process when running multiple LLMs which are all roleplaying and generating refined responses. As such, the user may be required to have strong hardware to run it. This process will be optimized to reduce computational overhead, to have a program suited to run in less advanced environments.

Cost of API Integrations

The application regularly executes LLM API calls such as the Gemini API [2], and may use other APIs like SerpAPI (Google Search API), significantly increasing operating cost of the project [3].

Licensing

The project will utilize various already existing technical tools like labeled datasets, which can often come with commercial licenses. We will carefully consider the licensing costs and choose which services are worth investing in, while utilizing open-source and academically licensed works as much as possible.

1.3.3. Ethical Constraints

The project will operate within ethical boundaries, and comply with legal restrictions. The system will only use data that is voluntarily provided and avoid sensitive and personal information. These constraints are further discussed in the subsections under “Professional and Ethical Issues” section.

1.4. Professional and Ethical Issues

1.4.1. Data privacy and Consent

Sources such as CV's, research papers, or scraped contents to build a persona might violate privacy expectations unless the data is publicly available or reached by consent of a real person. Even publicly available data (LinkedIn, GitHub,

academic websites) requires ethical considerations if the individuals did not explicitly consent to being modeled by AI. The system must ensure GDPR-style principles such as purpose limitations and storage security [8].

1.4.2. Bias Possibility and Fair Representation

Created personas might misrepresent the intended individual unless the description process is thoroughly shaped, as persona-based behavior can vary significantly between models [9]. Seniority-based stereotypes might create a biased environment (overestimating a professor's rigidity or underestimating a junior CS student's knowledge). Since the created personas debate and come with a consensus, biased personas might negatively affect the trust-worthy result.

1.4.3. Misinterpretation of Consensus

Users must be informed that; reached consensus is not an authoritative technical truth, disclaiming:

- Consensus is not guaranteed to be correct.
- Personas are simulated creations, not real experts in the field.
- The product should not be taken as the only authority for critical software decisions in sensitive fields.

1.4.4. Accountability and Transparency

The system must ensure accountability and transparency to users by providing what data is used to create a persona, how personas debate and the insights of created personas initial thoughts on a given question, and the consensus reasoning.

1.4.5. Ethical Persona Simulation

During persona creation process possible harmful or offensive inputs that result with violative outputs must be avoided. For simulating real individuals, avoiding impersonation must be ensured.

1.5. Standards

The project must comply with ethical and technical standards that ensure safe handling of personal data, transparent model behavior, and responsible use of real individuals' information when generating personas. All CV-based personas must be created with explicit consent, and any sensitive data must be minimized, anonymized, or excluded where possible [8]. The system should avoid reinforcing biases, ensure fairness across personas, and prevent harmful or misleading outputs during multi-agent debates. It should also maintain clear boundaries between real individuals and their simulated personas, ensuring that generated behaviors are probabilistic approximations rather than definitive representations. Finally, the platform must follow general software engineering best practices such as reliability, auditability, data security, and version-controlled experimentation to support trustworthy consensus generation.

2. Design Requirements

2.1. Functional Requirements

- The user can register to the system and sign in.
- The user can create custom personas by giving desired prompts and/or give relevant documents such as CVs, academic documents, LinkedIn/Github profiles etc. and customize the avatar of each created persona.
- The user can deactivate any persona, and edit or permanently delete the custom personas they created.
- The system wıll provide default personas as presets for the user to add to their active pool of personas.
- The user can enter their prompt into a text area to start the debate process, creating the environment with the active personas in the pool.
- The user can run the debate by clicking a button.
- The system returns the individual responses of each persona and the final consensus of the judge LLM with reasoning.

- The backend shall process requests from the frontend and return appropriate responses.
- The backend validates all incoming data for type and format correctness.
- The backend connects to LLM APIs (Gemini etc.) to process user prompts [1], [2].
- The system logs all requests and responses for debugging and audit purposes.
- The frontend displays data dynamically fetched from the backend.

2.2. Non-Functional Requirements

2.2.1. Usability

- The system interface must be intuitive and accessible so all users from all age groups and technical expertise can use it
- Persona creation, task definition, and debate execution must require minimal steps and should be clearly guided by UI elements.
- The system must provide a clear and separate visual representation between the Persona's responses, judge reasoning and final consensus.
- Error messages must be descriptive and thorough, it should suggest corrective actions and guidance (e.g., invalid file format, missing fields).
- The platform must support responsive design for all different platforms and resolutions like desktop, mobile, etc.

2.2.2. Reliability

- The system should produce consistent and reproducible outputs when the same personas and tasks are provided, it ensures that the backend has a deterministic logic and is not random every time.
- Backend services must handle any unexpected failures without any problems, with fallback mechanisms for API timeouts, LLM unavailability, or malformed user inputs.
- All Persona data, logs, and debate histories must be persistently stored and remained intact across system restart in our database.
- System reliability must ensure that the debate cycle (persona responses → judge → consensus) completes without any interruption under typical usage loads and sessions.

2.2.3. Performance

- Persona creation, task parsing, and system response times should remain within acceptable latency, ideally returning results in maximum a few seconds depending on the LLM API response times.
- The system must handle multiple personas and multi-round debates without any significant degradation and delays in responsiveness.
- The Frontend interactions, including loading personas and rendering consensus and results, must remain responsive under usage conditions
- To minimize unnecessary LLM API calls we must use efficient caching or throttling mechanisms to reduce overall response time and increase performance.

2.2.4. Supportability

- The system should be maintainable with modular architecture, enabling any updates or changes to individual components (e.g., LLM API handler, persona generator).
- All components must be well documented, including backend APIs, persona creation pipeline, and database schema.
- The system should allow developers to integrate new LLM providers or update model versions with minimal architectural changes [1], [2].
- Logs must be structured and accessible for debugging system audits, and experiment reproducibility.
- Clear separation between development, testing, production environments must be maintained.

2.2.5. Scalability

- The platform must support scaling to multiple personas, larger debate sizes, and more complex tasks without requiring major redesigns and architectural changes.
- Backend architecture must support horizontal scaling (e.g., multiple worker instances handling requests concurrently) when deployed in cloud environments, this is where we increase the power of our system (using servers maybe) to have better performances
- Storage design must accommodate increasing volumes of data like Persona, debate logs, and user-uploaded materials.

- The system should be capable of handling more computationally intensive scenarios (e.g., multiple LLM calls per persona, long-form reasoning) using batch processing or asynchronous request handlers.
- The architecture must enable the developers to be able to integrate distributed computing or external task queues in the future (e.g., Celery, Cloud Tasks) if required for high-load scenarios and intensive work.

3. Feasibility Discussions

3.1. Market & Competitive Analysis

This application aims to assess the reasoning, role-playing, and interaction skills of large language models (LLMs) [1], [2], primarily targeted at researchers and developers looking to benchmark LLM performance in simulated multi-agent interactions, a concept explored in recent studies [4], [5]. Current options mainly concentrate on single-agent evaluations, conversational AI systems, or versatile chatbots, creating a void for organized, multi-agent role-playing and reasoning tests. The project creates a controlled setting for various LLMs to engage with specified personas, enabling users to assess reasoning, adaptability, and emerging behaviors, fulfilling the demand for more advanced evaluation instruments. Its significance is in facilitating empirical research and comparative evaluations, which can guide AI advancement, optimization strategies, and model assessment. This establishes the project as a targeted instrument in the expanding AI research infrastructure sector, providing important insights distinctive to this platform

3.1.1. Existing Tools and Rival Companies:

While many tools are available in AI-assisted software engineering, the majority emphasize single-agent interactions, code generation, or code review instead of multi-agent reasoning. Several significant instances consist of:

- GitHub Copilot / Copilot Environment

A robust individual coding assistant that produces code and clarifies modifications [10].

Nonetheless, it lacks support for various personas, role-oriented discussions, or consensus processes.

- CodeAgent (Research Initiative)

A multi-agent framework centered on automated code evaluations utilizing dedicated LLM agents.Its range is restricted to assessing code changes and does not encompass business personas, students, architects, or reasoning across roles.

- CodePori (Research Initiative)

A multi-agent system that creates complete software projects based on requirements.

It emphasizes code generation rather than discussions influenced by personal views or assessments of reasoning.

- ChatGPT Group / Claude Materials

General-purpose conversational platforms can mimic roles through prompts, yet they only enable single-agent reasoning and miss structured debate, judgment modules, or ongoing persona modeling

3.1.2. Competitive Gap:

Among these tools, the absent elements are:

- Precise persona development according to career stage

(student → junior → senior → architect).

- Organized multi-agent discussion with defined roles.
- Judge LLM assessing reasoning excellence and generating a conclusive agreement.
- Benchmarking and replicability for research and empirical investigations.
- Assistance for non-technical positions (PM, Finance, Operations), providing a wider perspective.

This gap establishes Consensia as a distinctive and specialized platform in the developing field of AI assessment and multi-agent reasoning for software engineering.

3.2. Academic Analysis

This research adds to academic knowledge by examining how organized interactions between large language model personas can enhance reasoning effectiveness in software engineering activities, building upon recent multi-agent LLM frameworks [4], [5]. Although current LLM studies mainly assess outputs from a single agent, Consensia explores if debate, disagreement, and consensus techniques produce more trustworthy and understandable outcomes.

A fundamental scholarly contribution is the structured approach for creating authentic personas through prompt engineering and role modeling, aligning with findings on persona-driven LLM behavior [9]. This enables the assessment of whether personas reflecting varying seniority levels (e.g., 3rd-year computer science students versus 10-year engineers) generate unique reasoning patterns consistent with human expertise distributions.

The initiative also incorporates a judge LLM tasked with evaluating persona results, spotting discrepancies, and forming a consensus. Examining this judicial procedure offers understanding into LLM interpretability, bias transmission, and multi-agent dependability — subjects currently examined in ongoing LLM investigations [5].

Moreover, the system facilitates reproducible benchmarking by saving each debate as a session. This facilitates regulated testing on:

- the impact of personas on the quality of reasoning,
- the impact of structured debates on accuracy,
- how consensus measures against ground-truth datasets,
- and if disagreements among agents forecast model uncertainty.

Ultimately, the platform permits ablation studies, enabling researchers to deactivate specific persona traits, prompting styles (e.g., CoT vs. non-CoT), or LLM setups to analyze their effects on reasoning. This provides a consistent and

scalable structure for examining role-oriented reasoning, consensus creation, and multi-agent LLM actions — a developing and largely overlooked area in AI investigation.

4. Glossary

API: Application Programming Interface
AI: Artificial Intelligence
LLM: Large Language Model
CV: Curriculum Vitae
GDPR: General Data Protection Regulations

5. References

[1] OpenAI, “GPT-4 Technical Report,” 2024. [Online]. Available: https://openai.com

[2] Google DeepMind, “Gemini: A Family of Highly Capable Multimodal Models,” 2024.

[3] SerpAPI, “Google Search API Documentation,” 2024. [Online]. Available: https://serpapi.com

[4] A. Yang, Y. Bai, et al., “Large Language Model Agents,” arXiv preprint arXiv:2401.11016, 2024.

[5] C. Chen, S. Zhao, et al., “Multi-Agent Debate Improves Reasoning in LLMs,” arXiv preprint arXiv:2305.14387, 2023.

[6] Pydantic Developers, “Pydantic v2 Documentation,” 2024. [Online]. Available: https://docs.pydantic.dev

[7] Microsoft, “Microsoft for Startups Founders Hub,” 2024. [Online]. Available: https://www.microsoft.com/startups

[8] European Union, “General Data Protection Regulation (GDPR),” Official Journal of the European Union, 2016.

[9] R. Xu, X. Li, et al., “Persona-Based Evaluation of LLM Behavior,” arXiv preprint arXiv:2403.01245, 2024.

[10] GitHub, “GitHub Copilot Technical Documentation,” 2024. [Online]. Available: https://docs.github.com/en/copilot