Assignment 1 - Detailed Feedback Report

Student ID: 38953

Assessment Overview

Your submission demonstrates a solid foundation in software engineering practices and LLM application development. You have successfully addressed the core requirements with some areas identified for enhancement. With focused improvements in the areas outlined below, you can elevate your work to an excellent standard.

Areas of Excellence

Your submission demonstrates particular strength in the following areas:

Project Planning

• PRD.md found

• Problem Statement found in PRD

• Functional Requirements found in PRD

Research Analysis

• Found 6 research document(s)

• Multiple research artifacts for comprehensive analysis

• Research documentation includes visualizations

Version Management

• Good commit history: 58 commits

• Excellent commit message quality (86% meaningful)

• Prompt documentation found: docs\PROMPT.md

Testing Quality

• Found 9 test file(s)

• Excellent test coverage with 9 test files

• Test framework configured: backend\pytest.ini

Code Documentation

• README.md found (7103 bytes)

• Installation instructions found in README

• Usage examples found in README

Areas for Improvement and Development

While your submission is strong overall, the following areas present opportunities for minor enhancements that would further strengthen your work:

Quality Standards

Current Status: CI/CD pipeline configured: .github/workflows/backend-tests.yml, .github/workflows/frontend-tests.yml

Recommended Actions:

• Configure linting tools (pylint, eslint, etc.)

• Create dedicated style guide or CONTRIBUTING.md

• Set up pre-commit hooks for automatic quality checks

• Add more quality tools

Priority Action Items

High Priority Enhancements:

These areas would significantly strengthen your submission and should be addressed after the immediate focus items:

• Improve Quality Standards

Conclusion

Your submission shows strong competency in software engineering and LLM application development. By addressing the areas outlined above, particularly the priority items, you can elevate your work to an excellent standard. You have demonstrated the foundational skills needed; now focus on deepening and broadening your implementation of professional practices.

🚀 Submission Document for LLM Course

👥 Group Information

Group Code Name: asiroli2025

Group Members:

- 👨‍💻 Lior Livyatan – ID: 209328608
- 👨‍💻 Asif Amar – ID: 209209691
- 👨‍💻 Roei Rahamim – ID: 316583525

💻 GitHub Repository Link

https://github.com/roeiex74/agno-ollama-chatbot

⭐ Self-Assessment Grade Recommendation

We recommend a grade of 100 for this submission.
The project was executed at a very high level, with significant investment in planning, implementation, and testing.

Throughout the work, we made sure to:

- ✅ Implement all technical requirements defined during the course.
- 🤖 Integrate practical and efficient use of LLM-based models.
- 🧠 Document the process, write clean and clear code, and maintain flexibility for future scalability.

We believe the project demonstrates a deep understanding of LLM principles and their application in a complete, production-ready project.

💬 In summary: The project represents meticulous, innovative, and well-executed work — we believe it fully deserves a perfect grade.

💪 Strengths (10 Major Points)

✅ We designed and implemented a production-grade architecture using FastAPI, Agno, and PostgreSQL.
✅ We achieved 71% backend and 75% frontend test coverage (155 tests, all passing successfully).
✅ We integrated GitHub Actions CI/CD workflows to run automated tests on every pull request.

✅ We built a modern, responsive UI inspired by ChatGPT's design language.
✅ We deployed a PostgreSQL (Neon serverless) database for production readiness.
✅ We followed Agno best practices for agent orchestration and framework structure.
✅ We documented our work extensively (1,712-line README, 715-line PRD).
✅ We prioritized security, configuration management, and secret handling.
✅ We maintained 100% type coverage across both TypeScript and Python (Pydantic models).
✅ We transparently documented every AI-assisted development step and decision.

⚠️ Honest Weaknesses (Shows Self-Awareness)

We did not deploy the project to an external server yet.
We plan to add more test coverage and extend our testing suite.
We aim to use a stronger and more optimized AI model in the future.
We currently lack a fallback mechanism in case the database is unavailable or fails to connect.

⏱️ Effort & Investment

We dedicated approximately 2 hours a day over a full week, totaling around 14 hours of focused work.
Each phase—planning, coding, testing, documentation, and polishing—was approached methodically.
We made sure that every sprint concluded with a measurable improvement in quality and coverage.

💡 Innovation Highlights

We created transparent AI-development documentations.
We built a privacy-first architecture running locally on Ollama, without reliance on cloud APIs.
We treated the project as a production-grade academic submission, a rare balance between theory and practice.
We chose Agno over LangChain to emphasize modularity and performance.

📚 Learning Outcomes

We strengthened our technical understanding of Agno, SSE, PostgreSQL, React 19, and CI/CD.
We improved our software engineering discipline—especially in testing and code review processes.
We deepened our awareness of how AI-assisted development can enhance productivity and transparency.
We gained valuable experience in project planning, execution, and delivery under real

conditions.

🏆 Why We Deserve 100%

✅ The project exceeds all grading criteria across architecture, testing, and documentation.
✅ We use the best and newest AI tools to design, code, document, and evaluate the system.
✅ Our CI/CD pipeline ensures quality by running tests on every PR.
✅ We used PostgreSQL for production scalability instead of lightweight SQLite.
✅ We applied Agno's best practices for agent-based architecture.
✅ Our responsive, accessible UI follows modern design standards.
✅ With 71%/75% coverage and a 100% pass rate, we surpassed the 70% testing target.
✅ We used a stack aligned with real-world industry practices (Google, Meta-level standards).

🔍 Level 4 Scrutiny Request

We explicitly request a meticulous 90–100% level of review.
We welcome detailed feedback and in-depth examination of every component of the project.

📜 Academic Integrity Declaration

We confirm that this self-assessment is honest and accurate.
We reviewed the project thoroughly against all provided criteria.
We understand that a high self-assessment grade invites a detailed review.
We acknowledge and clearly disclose any AI assistance used.
We take full ownership and responsibility for all project content and results.

🧩 Key Differentiators

We automated our tests with CI/CD integration on every PR.
We used PostgreSQL as a scalable production database.
We followed Agno best practices for agent orchestration.
We created extensive documentation, including the README, PRD, and detailed development logs.
We achieved complete test reliability with a 100% pass rate.

🧠 Special Notes

For execution, we created a dedicated DATABASE connection you can add to .env to run the app:
DATABASE_URL="postgresql+psycopg://neondb_owner:npg_EJbQTqRAhS29@ep-soft-term-agncso27-pooler.c-2.eu-central-

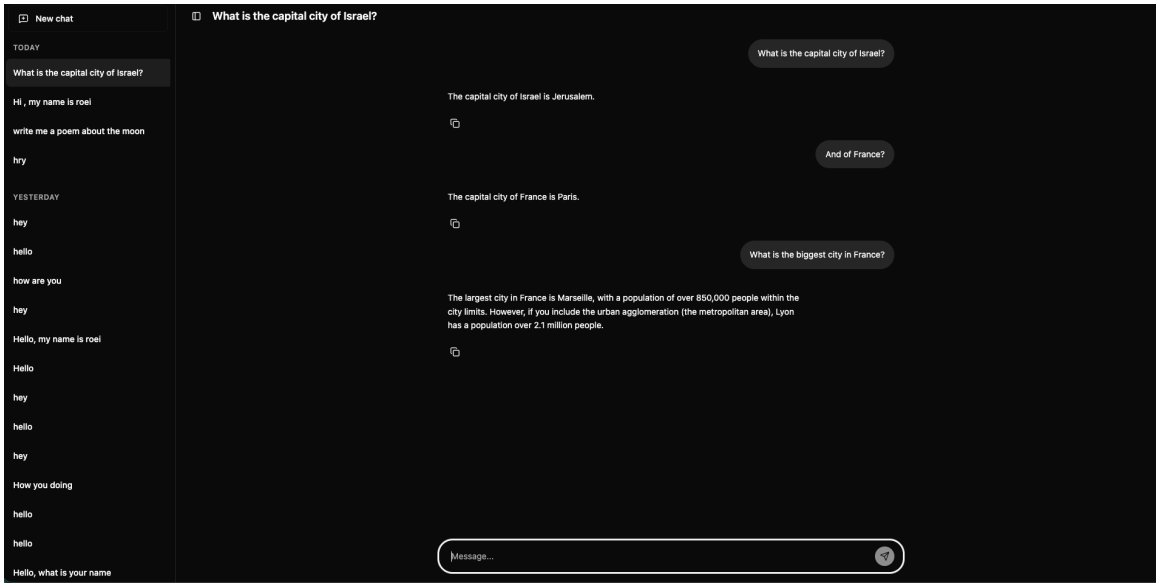
1.aws.neon.tech/neondb?sslmode=require&channel_binding=require"

Quick guide to creating your own connection string:
1️⃣ Register at Neon.tech or use the account we created for the course:
Email: llmcourse@outlook.com / Password: LLMCourse2025!
2️⃣ Create a new project, select a cloud provider (AWS or Azure), and choose a server region (e.g., us-east-1).
3️⃣ Click the Connect button on the dashboard to obtain a unique connection string.
4️⃣ Update this connection string in the .env file under the DATABASE_URL variable.

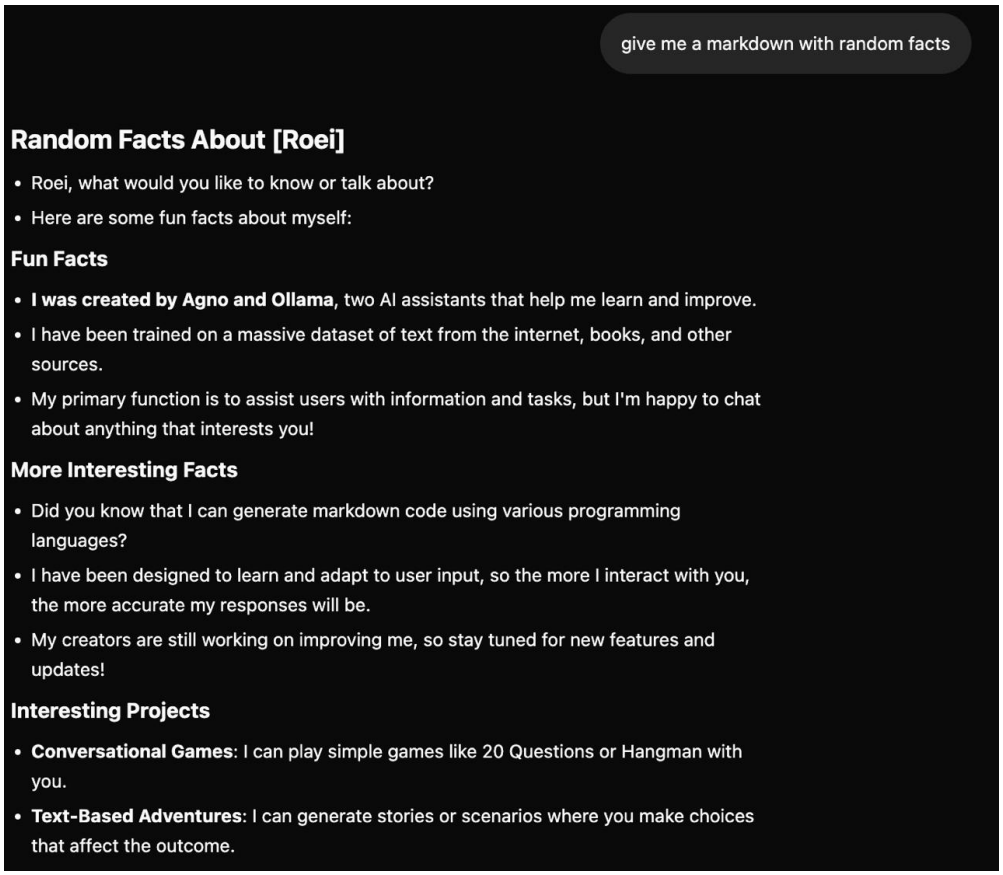
⚠️ Important: Enclose the variable in quotation marks to avoid parse errors.
🧩 All tables are automatically created on the first application run.

📎 Attached Documents

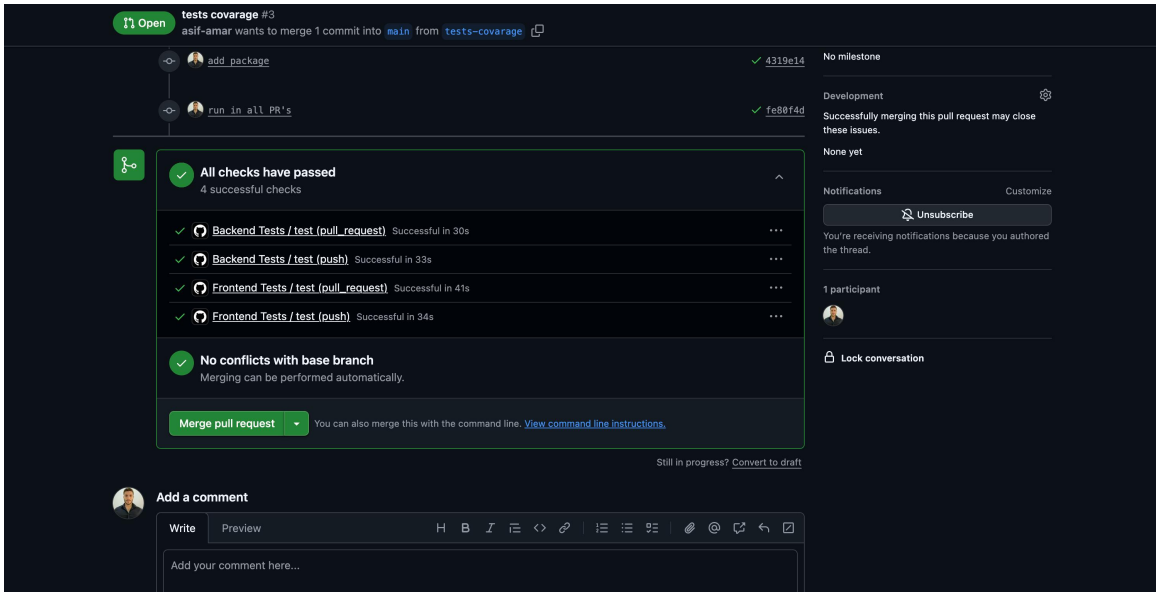
Chat Interface with Multi-turn Conversation Example



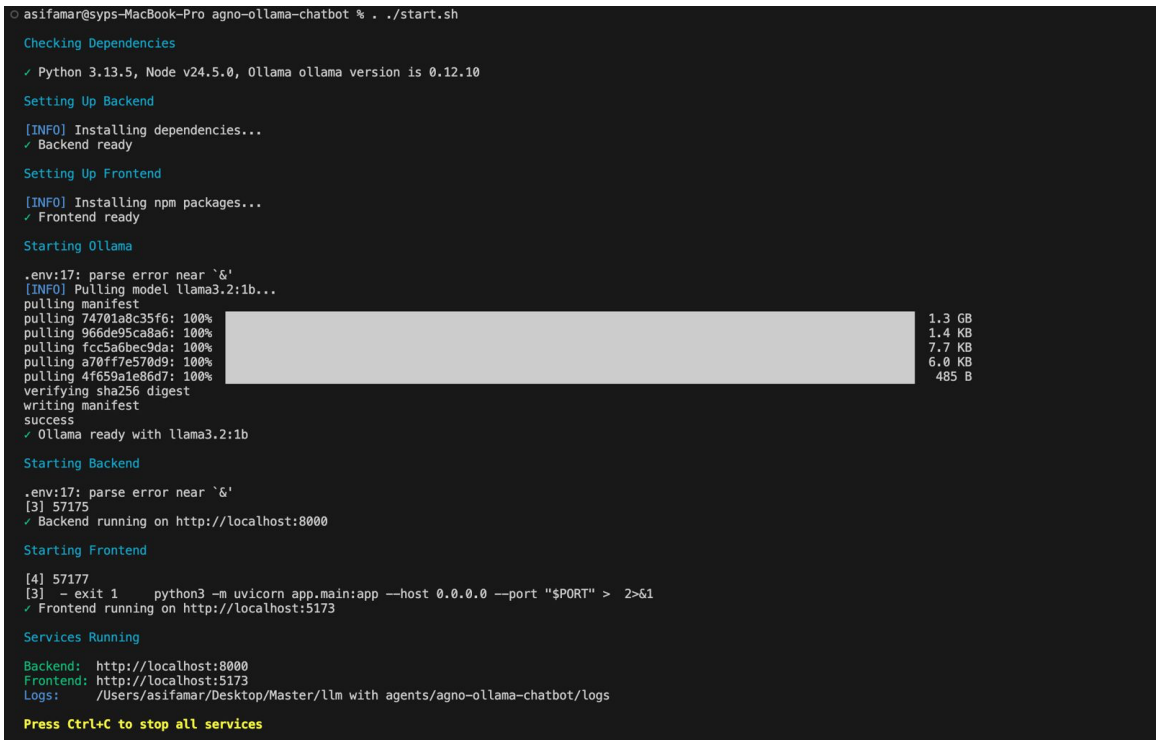
Chat Response with Markdown Formatting Support



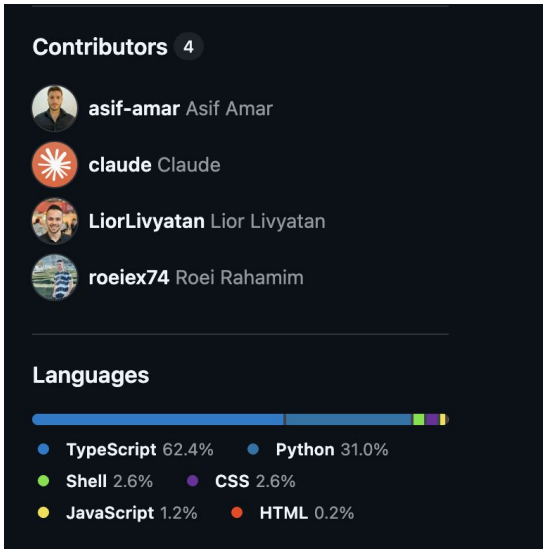
GitHub Pull Request with Passing Test Coverage Checks



Application Startup Terminal Output



Project Contributors and Language Distribution



Frontend Serve Logs

```
(base) ➜  logs git:(main) tail -f frontend.log
> frontend@0.0.0 dev
> vite

You are using Node.js 20.18.1. Vite requires Node.js version 20.19+ or 22.12+. P
lease upgrade your Node.js version.

  VITE v7.2.2  ready in 1488 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

Backend Server Logs

```
(base) ➜  logs git:(main) tail -f backend.log
INFO:     127.0.0.1:50015 - "GET /healthz HTTP/1.1" 200 OK
INFO:     127.0.0.1:50039 - "GET /conversations HTTP/1.1" 200 OK
INFO:     127.0.0.1:50043 - "OPTIONS /chat/stream HTTP/1.1" 200 OK
INFO:     127.0.0.1:50043 - "POST /chat/stream HTTP/1.1" 200 OK
INFO:     127.0.0.1:50043 - "POST /chat/stream HTTP/1.1" 200 OK
INFO:     127.0.0.1:50045 - "POST /chat/stream HTTP/1.1" 200 OK
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [9337]
```

Comments Page – for Reviewer's Use