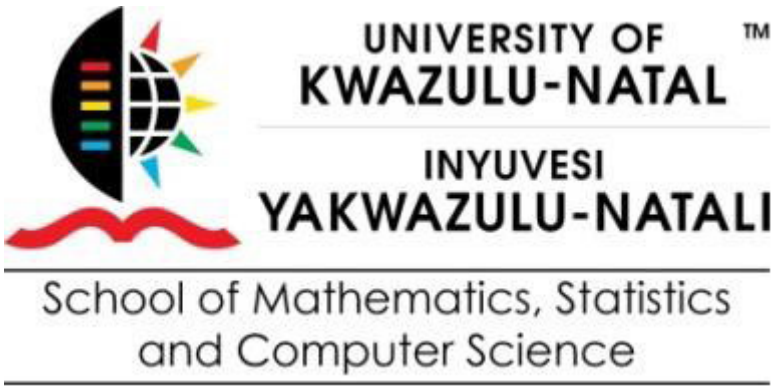

COMP301 Group Assignment Description

Vibe-coded Tool-Enhanced Agent-Based Assistant

Sem 2 2025

Introduction

In the fast-paced world of research, coding, and personal development, individuals often encounter challenges in finding relevant information, generating insights, and solving complex problems efficiently.

In the era of AI-driven development, traditional coding paradigms are being augmented by **vibe coding**—a workflow where developers describe systems in natural language while large language models (LLMs) generate functional code through iterative experimentation. To explore this paradigm shift, your group will develop a **Vibe-coded Tool-Enhanced Agent-Based Assistant** that leverages function calls, multi-agent hubs (e.g., AutoGen, LangChain), and tool augmentation (e.g., retrieval, code execution).

This project requires you to *embrace entropy*: measure the trade-offs between human prompting effort, debugging complexity, and architectural robustness when using AI to bootstrap AI systems.

Your agent is to provide a combination of one or more of the following roles:

- Web-Supplementation
- Data Science
- Private/Research Advisor

Project Goals

The primary goal of the Assistant System is to create an intelligent generative AI application that assists users with their work and/or personal development in some significant way. The system will leverage GPT technology - inference endpoints, local model training/pretrained models, fine-tuning, prompt engineering, and agent-based architectrures, or other LLM design patterns, to provide personalised and contextually relevant assistance somewhere across the spectrum ranging from information gathering and data analysis to sensitive and personal topics.

Create an intelligent assistant that:

- Integrates **agent-based architectures** (multi-agent collaboration, tool hubs) with **function-calling capabilities** (e.g., OpenAI tools, HuggingFace Tool Transformers).
- Is developed primarily through **vibe coding**: Your role shifts from manual coding to guiding LLMs via natural language prompts, feedback, and iterative refinement.
- Measures the **entropy cost** of AI bootstrapping: Quantify human effort (prompts vs. debugging time) and system efficiency (agentic architectures vs. monolithic LLMs).

Functionality

Choosing Your Topic

Your assignment is to choose a topic (domain) that is of interest to your group within the parameters defined by the three roles - the topic must imply at least *one of the roles*.

Then, choose appropriate datasets/datasources of content (if embedding/fineturing), and develop a set of GPT prompts that comprise the LLM aspect of the application.

Develop an agent-based architecture/design with which to embed the LLM into your application/API [1].

The baseline functionality defnitions (roles) are:

Web-Supplementation

The Web-Supplementation module will cater to the GPT, providing it with relevant information and answers to user queries by leveraging web search capabilities. The system will offer contextual insights and summaries of web search results to streamline the information gathering process and untimately keep users up-to-date with the latest information on their topic of interest.

Data Science

The Data Science module will cater to a GPT intended for data scientists and analysts, capable of generating python code to execute in a Jupyter sandbox environment for interactive data analysis, visualisations, explanations, and opther analysis suggestions based on data-related queries. The module will assist with data exploration to enhance the data science workflow.

[1] if API, you need demonstrations

Private/Research Advisor

The Private/Research Advisor module will cater to individuals, providing personalised advice and guidance on sensitive topics such as mental health, finance, and career development, or on their field of research with regard to their own work. The module will utilise local GPT models with embeddings and possibly fine-tuning to ensure user expected input/output consistency. The module will offer empathetic and non-judgmental responses to user queries, helping them make informed decisions and navigate complex life situations or research work.

Required Tool Enhancements (Choose Two+)

- **Multi-Agent Hubs**: Orchestrate agents (e.g., "planner" + "coder" + "validator") using frameworks like AutoGen.
- **Function Calling**: Integrate tools for web search, code execution, or API interactions via LLM function calls.
- **Vibe-Coding Scaffolds**: Implement tools that generate code from natural language specs (e.g., `@tool` decorators for prompt-to-function conversion).

User Interface and Customisation

The application will have an intuitive and user-friendly interface that allows users to seamlessly blend the required mix of Assistant functionalities. Users can customise the assistant's behavior and preferences within their selected domain, making it a personalised tool for their individual needs.

Local Hosting and Data Privacy

To ensure data privacy and security, the system will be hosted locally, ensuring that sensitive research data and coding projects remain within the user's environment. If you choose to use public APIs as part of, or instead of (e.g. OpenAI, Vertex, or MS Azure AI), please consult your lecturer before you commit to the topic.

It is recommended that you make yourself familiar with HuggingFace ASAP - before making hosting/API decisions.

Design Thinking Approach

The development process will adopt a Design Thinking approach, where user empathy will guide the design of an efficient and user-centric assistant system from beginning to the end. You are required to make use of existing design patterns. Your lecturer will be your client. For the empathy phase you will have to consult him so that he explains the business problem for you to scope and define a clear design challenge.

Collaborative Group Project

The project will be undertaken by your group of 3-5 students, with each member contributing their expertise to various aspects of the development process. The team will collaborate to ensure a seamless integration of the one or more roles your agent will be capable of.

You will be evaluated both individually and as a team on each Design Thinking process for your contribution. There is absolutely no limit to the tools you can use, as long as they will help you come up with an innovative solution. Preferably, make use of open source software for the obvious reason that we do not have a budget.

Your project evaluation includes but is not limited to bi-weekly progress reports by each team to the lecturer, final solution, technical documentation of the final system, final presentation of the project (zoom or pre-recorded screencast with zoom interview at the time). You will have to think out of the box and show some creativity to come up with an innovative solution that caters for future functionality (extensibility). There will be bonus points for going an extra mile (e.g. include multiple datasets/resources, social network integration etc. Or advanced prompt engineering feats.)

Phases and Timeline

Phase 1: Vibe-Coding Setup (2 Weeks)

- Choose a core role and tools; design agent interaction patterns.
- Choose a topic (domain) that aligns with one or more of the three roles (Web-Supplementation, Data Science, Private/Research Advisor)
- Select and prepare appropriate datasets/datasources of content for embedding/finetuning
- Establish entropy-tracking metrics (e.g., prompt/error logs).

Phase 2: Tool-Enhanced Agent Development (4 Weeks)

- Develop a set of prompts that comprise the LLM aspect of the application
- Design and implement an agent-based architecture to embed the LLM into the application/API
- Integrate the LLM
- Integrate tools/function calls (e.g., using LangChain, LlamaIndex, or custom hubs).
- Use vibe coding: Describe components in natural language, then refine LLM outputs.

Phase 3: Entropy Analysis & UI (3 Weeks)

- Build a minimal UI to demonstrate tool/agent interactions (e.g., Gradio, Streamlit).
- Analyze entropy data: Compare human effort across development stages.
- Design and implement an intuitive and user-friendly interface that allows users to seamlessly blend the required mix of Assistant functionalities
- Develop customization options for users to tailor the assistant's behavior and preferences within their selected domain
- Document entropy metrics at each milestone.

Phase 4: Optimization & Reflection (3 Weeks)

- Optimize prompts using advanced techniques (e.g., self-critique, chain-of-tooling).

- Refine and fine-tune the LLM using advanced prompt engineering techniques
- Test and evaluate the system's performance, ensuring it meets the project goals and user expectations.
- Submit a *Vibe-Coding Manifesto*: Lessons on AI bootstrapping entropy.

Conclusion

The Tool-Enhanced Assistant is an ambitious project that aims to revolutionise how researchers and developers access relevant information, generate code snippets, and make sensitive decisions based on private information. By offering some or all of the functionalities defined and leveraging GPT technology, local model training, fine-tuning, and prompt engineering, the project ensures a comprehensive and personalised experience for users in their respective domains.

Through this project you will gain valuable hands-on experience in model training/pretrained mod- els, fine-tuning, prompt engineering, and designing user-centric and AI-enhanced applications, preparing you for roles in the rapidly evolving fields of research and software development.

This project explores the frontier of **vibe coding** and **tool-enhanced agents**, challenging you to quantify the "entropy cost" of AI-driven development. By building systems where LLMs bootstrap tooling and multi-agent architectures, you'll gain insights into the future of software engineering—where developers orchestrate, rather than author, code.