Project Title:
AI Learning Companion – CodeMentor

Solo Project by:
Rayyaan Haamid

Course:
ITAI 2376 – Deep Learning and AI Agent Development

Submission Date:
April 7, 2025

The problem statement:

Most students struggle to acquire programming skills, particularly by themselves. Conventional online courses and tutorials tend to offer one-size-fits-all content, which fails to cater to unique learning needs. CodeMentor seeks to bridge this void by creating an Interactive Learning Companion—an adaptive AI agent that understands every student's background, adjusts content in real-time, and offers tailored guidance along with constructive feedback. With its tailored exercises, adaptive hints, and performance-based learning tracks, CodeMentor will help students learn to program more effectively and with more confidence.

Project option and rationale:

Selected Option:
Interactive Learning Companion

Rationale:

- There is increasingly high demand for adaptive computer tutors that can tailor education to the learner's learning style.
- By creating an interactive programming tutor, the project meets both the needs for individualized education and the technology requirements of agent-based learning systems.

- With Azure AI Studio, pre-trained language models (such as GPT-3.5-Turbo), and automation toolkits, the project is within easy grasp of a single developer on a proof-of-concept.
- The interactive model can be extended to support incremental improvements—from initially providing rule-based suggestions and exercise generation to later adding reinforcement learning for better adaptation based on user performance.

Agent design:

Architecture Overview:
CodeMentor will consist of four integrated modules:

Input Processing:

- Student inputs—questions, code submissions, or exercise requests—will be accepted through a minimal text-based interface (or a web application front-end). Natural language processing (NLP) will parse queries and ascertain learners' current requirements.

Memory System:

- A modular memory system will store learner profiles, past interactions, and performance data. This can be achieved with Azure Cosmos DB or even local file storage for a prototype. The system will maintain a record of exercise performance, mistakes, and trend in progress.

Reasoning component:

- This module employs chain-of-thought reasoning in conjunction with light-footed reinforcement learning (RL) iterations. The agent will initially develop exercises and advice on the basis of rules and pre-trained models (e.g., GPT-3.5-Turbo via Azure) and then fine-tune accordingly based on simulated or user-fed performance. The RL module will adjust difficulty and content style based on feedback metrics.

Output Generation:

- CodeMentor produces personalized exercises, advice, and step-by-step feedback. Interactive reports with error breakdown, suggestions for improvement, and adjustments in the learning path will be generated by the system. Visualization tools such as Power BI or Plotly can be used to show progress dashboards.

Tool selection:

The agent will leverage external tools of different kinds to offer its service:

Azure AI Studio & Machine Learning Services:

- Function: Use pre-trained language models (e.g., GPT-3.5-Turbo) for program exercise generation, code submission testing, and hint generation.
- Integration: Azure's managed services provide scalable model deployment. Budget reminders will be implemented to monitor resource usage, and cache mechanisms will prevent superfluous API calls.

Financial/Code Evaluation APIs & Document Processing Purpose:

- While previously used for other functions, APIs employed to verify code validity or process textual input (e.g., CodeRunner for running sandboxed code) can now be utilized to test student submissions.
- Integration: APIs of this sort will be coupled with sufficient error handling and contingency plans (e.g., local unit tests) to confirm code functionality before delivering feedback.

Other tools could involve n8n for automating workflow processes (e.g., triggering assessment procedures on new user inputs) and a vector database (through Azure Cosmos DB) for data storage and retrieval of learner profiles and past records.

Development Plan

Planning Phase (March 24 – April 7):

- Define the core requirements for the interactive tutor.
- Research Azure AI Studio documentation and relevant programming tutorial datasets.

- Create initial system architecture diagrams detailing module interactions and data flow.
- Finalize the project proposal and development timeline.

Implementation Phase (April 8 – April 28):

- Set up the development environment in Azure AI Studio and establish a GitHub repository.
- Implement core modules: input parser, memory module, and basic exercise generation with GPT-3.5-Turbo.
- Integrate code evaluation functionality and set up error handling for external API calls.
- Develop a feedback mechanism using simulated learner performance metrics.
- Conduct iterative tests, refining the adaptation of content based on performance feedback.

Finalization Phase (April 29 – May 5):

- Complete end-to-end integration ensuring input processing, reasoning, and output generation work together.
- Enhance safety measures and document fallback procedures.
- Create comprehensive documentation (README, usage instructions, and code comments).
- Record a demonstration video highlighting real examples, key features, and system robustness.
- Prepare deliverables for final submission.

Evaluation Strategy:

CodeMentor will be evaluated based on:

- Technical Implementation: Functionality of input processing, memory storage, correctness of reasoning, and clarity of output. Unit tests and integration tests will be conducted.
- User Feedback & Reinforcement Loop: Feedback will come from simulated user performance data and ratings from experts, with success measured in terms of gains in exercise difficulty adaptation and learner progress.

- Performance & Usability: Response time, quality of content generated, and feedback clarity are a few of the measures that will be tracked. Benchmarking against common educational metrics will be included in the test plan.
- Safety and Transparency: Agent input validation, error handling, and clear communication of its limitations will be exhaustively tested.

Resource Requirements and Risk Assessment:

Resources Required:

- Compute Platform: Azure AI Studio (with $100 student credits), to provide access to strong AI models and compute resources.
- APIs and External Tools: Financial/code evaluation APIs (e.g., CodeRunner), GPT-3.5-Turbo through Azure, and a vector database for student profiles. (This will be worked out along the way)
- Development Tools: Python, n8n for workflow automation, data visualization libraries, and GitHub for version control.

Risks & Mitigation:

- API Limitations/Cost: The countermeasures for the same are cache API responses, minimal viable parameter usage, and strict budget alarms on Azure.
- Model Adaptation Issues: Begin with rule-based and subsequently use reinforcement learning in pieces. Test extensively with simulated users.
- Scope Management: Keep features lean to focus on one-on-one tutoring for Python. Subsequent releases can include additional features (such as multi-language support or other subjects).
- Data Privacy & Security: Implement strict input validation and error handling. Inform users of the system's limitations and data use policy in a clear manner.

Final Remarks:

CodeMentor wants to be an innovative, interactive learning companion that learns from each user and gives constructive, clear feedback. By leveraging the rich capabilities of Azure AI Studio, integrating tried-and-tested APIs, and utilizing an adaptive reinforcement learning system, this project has the promise to significantly influence personalized learning. Through a focused intensity that allows for incremental optimization,

CodeMentor is not just a feasible solo venture but also a feasible product with significant market potential for transforming mundane learning experiences.