PART 1: 10 Different Ideas for LLM Powered Experience

1. U.S. History Tutor LLM
    a. High school level students learning U.S. history
    b. Students are having issues studying (LLM is like a tutor)
    c. Can dynamically help students based on current knowledge, direct students in the right direction without a teacher present.
2. LLM to help learn Spanish
    a. People who want to learn a new language (Spanish)
    b. Can be supplemental to tools like Duolingo , Babbel, etc
    c. Can work with the apps to fine tune and correct the user based off mistakes in their comprehension
3. Ochem compound tool
    a. Help college students taking organic chemistry learn the structures of compounds
    b. Organic chemistry is a hard class so it can help them study
    c. Adapts to the student's current knowledge, 24/7 access, etc
4. Library Database LLM
    a. Users of any library system
    b. Trying to find books with specific talking points
    c. Can find books by context, not just based on title, author, genre, and other surface level descriptions.
5. Computer Part LLM
    a. People purchasing or curious about computer parts
    b. Can dynamically explain or compare parts
    c. Can take the user's budget into account, help decide which parts are most important, which parts are compatible and which aren't, and can compare overall benchmarks, essentially a human guide but available at any time.
6. Course Curriculum LLM
    a. Students enrolled in at a specific college
    b. 24/7 Counselor
    c. Can help students decide which courses to take based on interests and credit requirements. Should know all the courses available at that school, when the class is available, how it fits into a student's schedule, possibly how it can lead to a minor in that field, etc
7. Product "Sale" LLM
    a. People interested in buying an item
    b. Can search many websites and stores for sales

    c. Ability to search for the cheapest price of a certain item, or if the product has been cheaper in the past and advise you to wait for a better sale.
8. SAT tutor
    a. High school students studying for the SAT
    b.
    c. Similar to the other tutor ideas above, it will be more accessible for people who can’t afford/access other SAT tutoring services,
9. Video Game Guide LLM
    a. People struggling to beat a game
    b. Helps with people who can't find a good guide or where they are in the game
    c. Can either describe the problem or/and supplement with images of the area in the game to help the LLM determine the problem and help solve it.
10. Housing LLM
    a. People trying to buy a house or rent
    b. Makes process of finding an area more simple
    c. Can help user find the area they want to live and competitive prices for that area, based on the important factors they want.

PART 2: Prioritize one idea - Course Helper LLM

We wanted to choose the Course Scheduling related topic. This is because it is a common pain point for students. It can be tough to figure out what courses students should be taking in what order, which prerequisites they need, and which semesters certain courses are available. An LLM is good here because it will be available 24/7. It can instantly provide answers about requirements or schedule conflicts based on the student's major, year, and plans for the future.

For the scope, we'd most likely focus on just the Computer Science & Innovation program at Champlain rather than attempting all the majors. It's the most familiar to us since we are CSI students and keeps it manageable while still showing that it can be extended. This makes it more reasonable to check graduation course requirements for one category of student.

The primary risk that could come up is the LLM recommending courses that don't exist or being incorrect. This especially can be a problem with any user that attempts to break the program. We can reduce this risk by providing the exact course catalog for our prompt and telling the LLM to not make up course information. Of course, that's not foolproof either so it's possible this tool we create is supplemental to a human advisor.

We chose this tool over our other ideas because we felt that this was an issue for students here at Champlain. You need a lot of meetings with your advisors over the four years as a student here to double check courses, minors, concentrations, when courses run, etc. We also have some of the resources necessary to accomplish this, such as a partial course catalog.

PART 3: The Core LLM Experience

Short Explanation:
Our tool is a conversational AI tool that acts as a 24/7 academic counselor for Champlain College CSI and CNCS students. Students can ask questions about the two programs, prerequisites, requirements, etc and the system can give personalized responses.

The system prompt defines how the LLM will behave. It should act as a counselor that's friendly and student centered with a pedagogical approach. The prompt will stay constant no matter what program or school the tool is set for.

The subject prompt defines what the LLM knows. We had Claude generate a course catalog for the CS and CNCS programs with course names, credit counts, prerequisites, etc. To use this tool for a different school or major, this single file could just be swapped out for a different one.

When a user uses our app, they see a welcome screen with four suggested common questions. They can also just type in their own questions. The message gets sent to Claude via the Anthropic API along with the prompt layers and the entire conversation history so the LLM can remember stuff.

Future improvements that could be made to the app may include integration of more courses for different majors and possibly tailoring the UI to look more Champlain-coded. Another improvement would be a live database with Champlain course data as opposed to the current static course data in subjectPrompt.js.

Who did what:
Part 1 was done together in class
Part 2 was Written up by William and then checked/edited by Thomas
Prototype/Part3 was "vibe-coded" through Claude and we looked over it afterwards to double check things

PART 4: Define Learning Representations

Tokens
Tokens are the fundamental unit of data in AI. They are words, parts of words, or characters that LLMs use to process and generate language. For our project, our tokens would be things like the course attributes. Things like the course code (CSI-240), the number of credits, which semester it might be in, its prerequisites, program requirements, etc.

Words
Words are tokens that are assembled together into something with more meaning. In our system, words would be the entire course itself. An example would be CSI-281: Data Structures & Algorithms, Fall & Spring, prerequisite CSI-240, required for CSI program, etc. It will be a single entity that the system can reason as a whole.

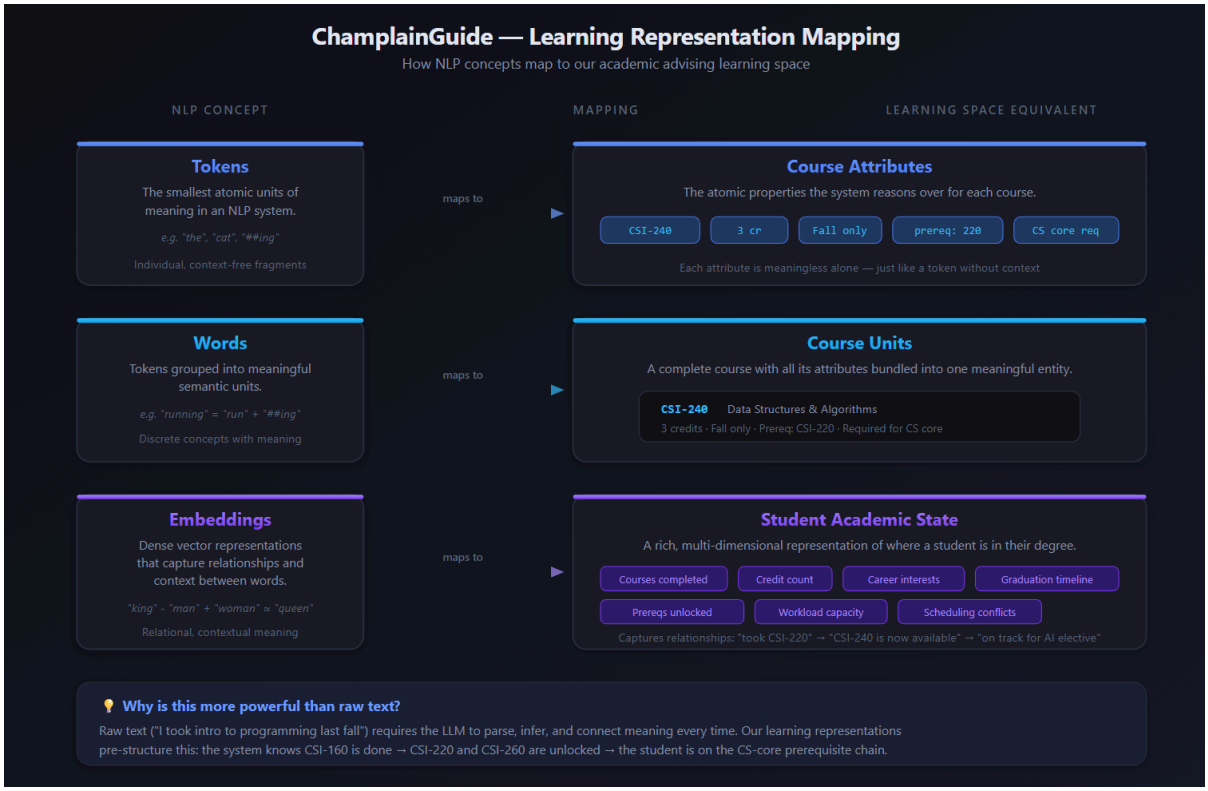
Embeddings
Embeddings are vectors that encode relationships between words. They help capture the relationship between words like “king” and “queen” and “man” and “woman”. Our tool’s embeddings will be the student’s current academic state. It would be a multi-dimensional representation of where the student is in their program. This includes things like how many credits they have, which courses they’ve already taken, their interests, graduation timeline, etc. For example, knowing that a student has taken CSI-240 means they’ll be able to take CSI-281 the next semester and that would open up a whole ton of other courses.

What units of learning this system reasons over
Starting with individual courses, our system can reason over their metadata (prerequisites, credits, learning outcomes, etc). Then it can also connect those individual courses to understand course sequences like CSI-140 -> CSI-240 -> CSI-281. Then at the program level it can reason over program requirements like “complete 6 CSI-300+ level courses” treating them as containers that only specific courses can satisfy.

How these units are inferred from student interaction
Our program can build its understanding of the student’s current academic state from their conversations. So when a student says that they’re in their third semester (sophomore fall) taking CSI-281, they can assume that they’ve already taken CSI-140 and CSI-240 as a freshman. Then they can also approximate credit count (30 per year) and which prerequisites can/will be unlocked for the future.

Why this representation is better than raw text
Building off the answer above, the system will continue refining the student’s state as the student continues chatting. If the student tells the model, “I’m really interested in AI and also want to

graduate a semester early", the dimensions might change, with a vector maybe pointing towards AI-related electives/courses. This is what makes the system better than raw text. A course catalog can only tell you the prerequisite for a course. But the system can tell a student an optimal path to get there based on current interests, classes taken, and timeline.



PART 5: Using a Frontier Model to Level Up

How could a frontier model help?
A frontier model can add many capabilities and features to the program we have, including a deeper level understanding of what misconceptions a student may have. It's easy if a student knows where they are lacking in terms of information, and they can directly ask the LLM, but it's harder for the model to recognize where a student may be making a mistake without it being clearly pointed out. They are also much more capable at creating visuals, which may be helpful for certain students who learn better with a graphical representation.

The model will also be able to fully take in all of the information the student has been giving out across many different messages. This can matter if the conversation goes on for a long time, or if a student mentions information briefly and tries to refer back to it later. Weaker models will usually struggle at doing this, and will focus primarily on analyzing what the text means over the situation as a whole.

Where inference, memory, or representation quality matters
One of the most important reasons for inferences would be deciding which semester works best for each class, as this can be highly subjective. This can especially be difficult with electives, as they aren't always built into course structure that require you to take them in order. If a student as a computer science major wants to take a business class, they may ask the LLM when they should take it, and it will need to infer based on the information it has been given which time(s) would work best for that student.

Memory matters for keeping the conversation going over a longer period of time, and allows the model to make callbacks to earlier things stated by the user. This can be important when talking about the amount of credits the student has, classes they have taken, and topics they are and aren't interested in. It can also help with catching inconsistencies, such as if a student thought they took a course or had certain credits which actually wasn't the case. It's important when working in the role of a student advisor to be super observant of the student's schedule as complications could lead to them having to take extra classes.

Representation quality matters most when it comes to understanding the relationship between students' interests and what the classes teach. Not every class that dives into a topic will have that topic in the name, so it's important for the AI to map the relationships between words properly. Representation quality also improves the quality of the embedding, which is arguably the most important part of this model as that's where the foundation of all the courses comes from, which is the main thing that differentiates it from other models.

What new learning values are there?

With the new features added, the model can go from mostly just spewing information that it has digested to making more complex thoughts like an actual advisor. It should be able to make more accurate assumptions of the difficulty and complexity of certain classes, and steer students towards what their goals are. It should also allow the LLM to be able to guide the student more efficiently, with guiding questions that don't force the user to start and steer the conversation if they don’t choose to. The model should also be better at gauging student emotions, in case there are feelings of annoyance with common problems that come up with registration.

What data is used for the embedding?

The primary data for the embedding will be the actual courses and their content taken from the official school databases. This includes the course prerequisites, descriptions, and days/times that the class is being taught. This information will help the model determine what fits the student most, especially if the student gives vague information when asking what classes they should take. The model will also be able to take into account what a student expects from a class or major and compare it with what is stated by course descriptions. This would also help the student for choosing minors, and concentrations, based on their majors and interests.

PART 6: Reflection and Tradeoffs

Challenges
Immediately one of the biggest problems and challenges with making your own LLM and more specifically using an API Key is the token cost. Especially working with a frontier model, the costs of each call will continue to grow, including with conversations getting longer and longer. Thankfully as this is a smaller project and not a product ready to ship, too many users overloading the model shouldn't be a problem that needs to be faced. Latency is also a concern if the conversation drags on for too long, which may add a few seconds onto the thinking time for the model.

Another problem also present in the previous model is having the course content be updated in real time, as courses could change at any point. In a perfect implementation the correct data would be scraped in certain time intervals to ensure that the information given out is correct and up to date. Hallucinations are also always a risk, even with specific prompting that tells the LLM to only say information based on what it knows, it may end up saying something made up anyways. Especially when the model may need to make tougher decisions closer to an actual advisor, it may get confused between what it thinks it knows and reality. It has been held as a very high priority to do as much as possible to keep this model from hallucinating.

Trust is a hard concept to balance with AI, on one side we want the tool to be usable and trustable and fulfill the same role as an advisor. But we also know that it will never quite be the same as a real advisor, and it is suggested that a student's academic advisor views and confirms plans before they are put into place. It is best when the users have a good understanding of the challenges of AI, and aren't too over dependent on it, while also seeing the values it can give and gaining from said values.

This leads to another concern also present in other AI features, that being of over scaffolding. It is a real possibility that if this kind of tool was proficient enough, it could lead to students ignoring this part of their education. Things like planning your schedule around the time you have and navigating classes based on interests could become skills people don't have to or don't want to learn. Ideally users will take the information given and it will be explained in a way that doesn't leave them with nothing to do, they should still need to apply the knowledge that the model gives them.

Accomplishment(s)
It felt good making something that is closely applicable to being at Champlain College, and it definitely feels like in the future it may become standard for some sort of AI Tool that is linked

to a college’s backend. So it was interesting to sort of take a sneak peek at what that could look like.

Insight(s)

The areas where AI tended to hallucinate were more mundane things the AI was saying, often when it was more critical information it would more easily recognize when it didn't know something and point it out. It seems that sometimes it doesn't know that it doesn't know something, and then proceeds to make it up.

Next Step(s)

As mentioned prior, an important step that could be implemented later is the real time scraping of courses. This would ensure that the model is up to date as much as possible, and also includes all of the available courses. Something more complex that could be added conceptually would be integration with Champlain Self Service, allowing the model to more easily see students' prerequisites, current and past classes, and credits, before talking to them. This would involve a full overhaul of most features and isn't very realistic as a small project.