# Tutorbot Backend/API Project Proposal

## Group Members:

Colin Cassell (2 credits)

- Skills: Java, Python, MongoDB
- Tasks: Create an API to interface between TutorJS and the Chatbot. The API should be able to convey messages from TutorJS into the Chatbot as questions, and then it should be able to return the Chatbot's answer to the question back to TutorJS.

Jin Park (2 credits)

- Skills: JS, Java, C, React, node, MongoDB, SQL, Express
- Tasks: Develop api for Chatbot to integrate with TutorJS. If time exists afterwards, work on adding more functionality to the frontend (instructor interface, graphing utility).

Krishan Patel (2 credits)

- Skill Sets: Strongest in Java, HTML, & CSS. Some experience with SQL, MongoDB, JavaScript, and Python
- Overarching goal this semester: Instead of having the IDE output generic error messages with hardcoded Stack Overflow link, have the IDE output the actual error message and then the student can use Chatbot to ask for a more personalized 'human readable error message' which can give more helpful Stack Overflow links.

## Project Goals

On the backend side of TutorJS, we need to create the API for the Chatbot to allow it to communicate with the React app. We will do this by first finding an appropriate cloud host for the chatbot neural network. This both aims to alleviate some of the computing load on our local machines as well as possibly ease the process of creating an api interface if the hosting service has that functionality baked in. Currently we have our chatbot in Google Colab, but this may or may not change. Our minimum viable product will be to demonstrate information going back and forth between the chatbot and the react app through the api we will develop. One example of this would be a student asking the chatbot a conceptual question like, "What is a first-difference filter?", and being able to see the chatbot's response in the TutorJS system. This product could also be in the form of error messages being interpreted by the chatbot and it recommending Stack Overflow links to help students resolve bugs in their program or definitions/clarifications on vocab that pull information from the textbook chatbot model. Here is a rough timeline of the project goals and what time frame we would like to implement them by:

## Project Timeline

- Week 1-3: Project planning, team building, proposal drafting
- Week 4-7: Find an appropriate cloud hosting service. Plan exactly what functionalities the API will need, and research how best to implement it. Plan any data model / database schema changes that are needed to integrate the Chatbot. Implement human-readable error messages into TutorJS. Add all routes, models, controllers into the backend API.
- Week 7-10: Develop the axios methods that will allow us to create an API that takes a question for the Chatbot, queries the Chatbot, and returns the Chatbot's answer back to TutorJS. Need to figure out how to format the input data into json format to better communicate with the React app. If there is time, look into Chatbot error message functionality with stack overflow data.
- Week 10-15: Hook up new UI to new API changes and test. Validate changes made to Chatbot are working. Iterate from here to ensure everything works as expected. Prepare final presentation.

## Project Description

The backend side of TutorJS, at least in terms of the student view, is working thanks to our efforts from last semester. Hence, this semester, we plan to shift our focus on the integration of the chatbot that the other ITS teams were working on. We believe that by integrating these separate aspects of the ITS vip project, we can deliver a more intuitive and smart method of teaching students both concepts in ECE and programming. This will involve planning and developing a separate api to interface with the chatbot code and query into and extract responses that can hopefully assist students in their learning endeavors. There are a lot of interesting directions we can take this aspect of the project. Given that the chatbot will be trained to be an expert in the subject matter at hand, we can incorporate essentially a virtual professor into the TutorJS web app, a step that will bring the entire service closer to the "Intelligent Tutoring System" we aim to build.

As far as challenges go, the most significant obstacle we will face will be trying to understand the functionality and codebase behind the chatbot itself. Although this may not be strictly necessary, a deeper comprehension of the system that drives the neural network might prove to be a useful tool in planning and developing what sort of functionality we want to integrate into TutorJS itself. This will involve a lot of communication between the chatbot team and us, each trying to learn more about the program we are separately making. Another challenge is our unfamiliarity with interfacing with python scripts using javascript, especially when the code is running off a cloud platform like Google Colab. We will need to spend some of our time learning the tools and libraries that come with working with Google Colab.

Although we are still not sure exactly what tools we will use to build our api, the team is familiar with node.js and express as a framework to run node servers, as well as axios to interface with the React app using http methods, so if there is a way to run the chatbot in this environment, that will be extremely helpful in moving the project along. Finally, we will be using microsoft teams as our main communication platform and working off github to collaborate on our code. We hope that our work this semester will produce a truly intelligent system to teach our students the fundamentals of both digital signal processing and coding.