# Magic 8-Ball App Exploring Generative AI Tools

Group 5: Vivin Vinil, Andrew Onozuka, Alexander Tahan, Yuantian Zhou, Abhimanyu Srivastava, Merrick Qiu, Joseph Del Val, Julia Poon, Jacob Felts, Leo Lee

## AI tools Explored

- Chat Gpt 3.5
- Github Copilot
- Bing search/chat

## Benefits

## Benefits - Probability and “Certainty”

- These LLMs are fundamentally probabilistic— using them effectively calls for high-certainty situations
  - Highly specific questions work better
  - Performs best with syntax or documentation-related questions
    - What function does X?
    - What does X function do?
  - Questions with an answer easily discoverable on the internet are also dealt with effectively
- The higher a situation’s certainty, the more likely it is that the model will perform well.

## Benefits - Code Generation

- *Predictive* models work best for *predictable* code
  - Generates boilerplate code without issue
  - Can reliably create “skeleton code” or “starter code”
- Small-scale “units” with clearly-defined functionalities are easily generated
  - Can be arbitrarily specific— define the exact functionality
  - Comments are also included
- In saving time otherwise spent writing this type of code, more attention can be given to design and structure

## Benefits - Saving Time

- Time saved can lead to substantial benefits
  - More time can be devoted to low-certainty elements of software
    - Design, architecture, all things with room for interpretation
  - Lets you refocus attention to the more challenging aspects of software development
- Less of a builder, more of an architect?
- Can also save time spent on research
  - Bing chat recommends website links
  - Includes information on interactions between languages
- Limited— but present— debugging and explanation capabilities, for sufficiently common code/errors

## Negatives/Issues/Bugs

## Negatives/Issues/Bugs - Memory Issues

- Unable to respond appropriately to wide range of topics
  - Sometimes gives an answer not relevant to context.
- Will forget what context of the conversation is, must be consistently reminded about what you are doing.
- When asked a complex question, It will often ignore aspects important parts of the question.
- Poor for longer prompts
  - Example: when asking for code generation
  - Works best when asking for short code snippets; asking for code generation from longer prompts causes it perform badly

## Negatives/Issues/Bugs - Potential Bias in Data

- Responses are usually refactorization of responses found on internet.
- Not an encyclopedia— data is very human
  - The sources for its predictions can come from very subjective sources, including reddit.
- No sense of correct/incorrect — only likely/unlikely given data.
- If something does not exist in the data, it will not be predicted.

## Negatives/Issues/Bugs - Probability

- Probability makes it inherently inconsistent— the same prompt can and will give different results
- Unknown inputs lead to unknown behavior
    - It’s predictive— if you give it something it’s unlikely to have seen before, its behavior will be the result of probabilistic “guesses”
- Performs poorly with very complex prompts
    - Unknown how it “weighs” each word/phrase in your prompt
    - Will occasionally “ignore” things you’ve said
- Isn’t really “reasoning”-- it predicts the most likely output for this input

## Negatives/Issues/Bugs - Limitless Compliance

- If you express disagreement, ChatGPT will automatically comply and agree it was wrong.
  - Relic of prediction over reasoning— if you said it was wrong then it *must* have been
  - Oftentimes the explanation as to why it is wrong is wrong.
- Does not actually understand what is being explained
  - It’s a model, not a brain

## Takeaways

- Remember that these models are predictors, not thinkers
    - Correctness is not guaranteed— remain skeptical and verify that the outputs are correct
    - They’re useful and time-saving for predictable boilerplate code, unreliable for more complex or uncertain prompts
- They’re not a substitute for knowledge of the tools you’re using
    - You still need to know how to incorporate their outputs into your code.
    - For complicated errors, you still need to know how to fix them