NAVIGATE AI

High School Curriculum

Grades 9-12 | Semester 1

16-Week AI Foundations & Application Development Program

90-Minute Weekly Sessions | Detailed Lesson Plans

Navigate AI LLC
San Antonio, Texas
2026

Program Overview

Navigate AI's High School Semester 1 curriculum is a 16-week program designed to take students from zero coding experience to building real AI-powered applications. Students' progress through four units that build on each other, beginning with AI foundations and data literacy, advancing through Python programming, and culminating in two capstone projects: a PDF Summarization app and a Document Q&A RAG Chatbot.

By the end of this semester, every student will have built two working AI applications in Python, including a chatbot that uses Retrieval-Augmented Generation (RAG) to answer questions grounded in real documents. These are tangible portfolio pieces that demonstrate real technical skill.

Course Description

AI Foundations & Application Development

Learn AI fundamentals and Python, then build two real applications: a PDF Summarizer and a RAG Chatbot powered by real documents. No experience needed.

Students go from zero coding experience to building real AI-powered applications in one semester. Starting with AI fundamentals and Python programming in Google Colab, students' progress into working with industry tools like Hugging Face, Gradio, and large language models. The course is anchored by two capstone projects: a PDF Summarization and Q&A app using pre-trained AI models, and a Document Q&A RAG Chatbot that answers questions grounded in real documents. Students leave with deployable projects, practical Python skills, and a foundation in the tools and techniques used by professional AI engineers.

Class Format

Each weekly session is 90 minutes, structured to maximize engagement and hands-on learning time:

- Opening Hook & Review (~10 min) - Quick recap, warm-up question, or "what did you try since last week?"
- Core Lesson (~25 min) - New concept introduction, kept tight and interactive
- Hands-On Activity (~40 min) - The main build/explore/create block where students apply what they learned
- Wrap-Up & Reflection (~15 min) - Share work, discuss surprises, preview next week

Semester at a Glance

Unit	Title	Weeks	Duration
Unit 1	AI Foundations & the Builder Mindset	Weeks 1-2	2 weeks
Unit 2	Python for AI	Weeks 3-8	6 weeks
Unit 3	PDF Summarization App (Capstone 1)	Weeks 9-11	3 weeks
Unit 4	Chatbot + RAG (Capstone 2)	Weeks 12-16	5 weeks

Package Bundles

Package	What's Included	Duration
AI Kickstart	Unit 1 only	2 weeks
AI Explorer	Units 1-2 (Foundations + Python)	8 weeks
Full Semester	All 4 units with both capstone projects	16 weeks
Python for AI Add-On	Unit 2 standalone for students with prior AI exposure	6 weeks

Prerequisites and Requirements

Student Prerequisites

- No prior coding experience required
- A laptop with internet access and a Google account (for Google Colab)
- Ability to type and navigate a web browser

Student Hardware Requirements

- Laptop (Windows, Mac, or Linux) with internet access. Chromebooks work for all lessons except the Week 2 Ollama demo, however, laptops are recommended.
- Google account for Google Colab access (free)
- Week 2 only: Ollama installation requires approximately 200 MB for the application plus approximately 2.3 GB for the required model (phi3:mini). Students should install Ollama and pull the model the night before class.

Software and Tools (All Free)

- Google Colab (browser-based Python environment, free with Google account)
- Ollama (Week 2 only, free download from ollama.com)
- Gemini API (free tier, no credit card required. Rate limits vary by model and change periodically; see Week 14 Teacher Key for current limits and API key management strategies)
- Hugging Face (free model hub and inference API)
- ChromaDB (free, open source vector store, runs in Colab)
- Gradio (free, open source UI library for Python)

Cost to Students

The entire curriculum can be completed at zero cost. All tools, APIs, models, and environments used in this course have free tiers that are sufficient for classroom use. No paid subscriptions or API credits are required.

Technical Stack Overview

This section summarizes the tools and models used across the semester for instructor reference.

Units 1-2: Foundations and Python

- Google Colab (Jupyter notebooks in the browser)
- Python standard libraries: csv, os, random
- Third-party libraries: pandas, matplotlib
- Hugging Face transformers library and pipeline() function
- Gradio for building web interfaces
- Ollama for running models locally (Week 2 demo only)

Unit 3: PDF Summarization App (Capstone 1)

- PyPDF2 for PDF text extraction
- facebook/bart-large-cnn model for summarization (approx. 1.6 GB, downloads to Colab runtime)
- deepset/roberta-base-squad2 model for Q&A (approx. 500 MB, downloads to Colab runtime)
- Gradio for the web interface
- All models run on Google's Colab servers. No local storage impact on student devices.

Unit 4: Chatbot + RAG (Capstone 2)

- Gemini 2.5 Flash API for chat generation (free tier, no credit card required, pure API call, zero download. Daily request limits vary by model and region)
- ChromaDB for vector storage (free, runs in Colab, in-memory)
- ChromaDB's built-in embeddings model: all-MiniLM-L6-v2 (approx. 80 MB, downloads to Colab runtime)
- All processing runs on Google's Colab servers. No local storage impact on student devices.

Key Differentiators from Middle School Curriculum

This high school curriculum is designed for students in grades 9-12 and differs from the Navigate AI Middle School curriculum in several important ways:

- Aggressive conceptual pacing. Unit 1 covers in 2 weeks what the middle school version spreads across 9 weeks.
- Two capstone projects instead of one. The PDF Summarizer builds foundational skills that feed directly into the RAG Chatbot.
- Deeper technical depth. Students work with RAG pipelines, embeddings, vector stores, and chunking strategies.
- More independence. Less scaffolding, more self-directed building.
- Career connections. Explicit ties to industry roles (ML Engineer, Data Scientist, AI Researcher) and college pathways.
- Portfolio focus. Students leave with two deployable projects and documented work.
- Prompt engineering relocated. Moved from standalone intro to Week 13 where students write system prompts for their chatbots.

Detailed Lesson Plans

The following section provides detailed 90-minute lesson plans for each of the 16 weeks in the semester. Each lesson plan includes learning objectives, materials needed, a minute-by-minute class flow, and instructor notes.

Unit 1: AI Foundations & the Builder Mindset

Weeks 1-2 | The gateway to understanding artificial intelligence

Week 1: What Is AI + Data Literacy

Learning Objectives

- Students can define artificial intelligence and distinguish it from traditional software
- Students can explain the difference between AI, machine learning, and deep learning at a high level
- Students can identify at least five examples of AI in their daily lives

SAMPLE DETAILED LESSON PLAN

Week 7: AI/ML Concepts in Code

Week 7: AI/ML Concepts in Code

Learning Objectives

- Students can explain what a pre-trained model is and why it matters (training vs. using)
- Students can define tokens, context windows, and model parameters in plain language
- Students can install and import the Hugging Face transformers library in Google Colab
- Students can use the pipeline() function to load and run AI models for different tasks
- Students can run AI models for summarization, text generation, question answering, and sentiment analysis

Materials Needed

- Student laptops with Google accounts and Google Colab access
- Projector/screen for live coding demos
- Internet access (required for downloading models from Hugging Face)
- Note: Models download to the Colab runtime, NOT to student laptops. The largest model used today (facebook/bart-large-cnn) is approximately 1.6 GB and downloads in 2-3 minutes on a typical school connection.
- **Teacher-Prepared Resources (included with curriculum):**
- 📎 Week 7 PowerPoint - **NavigateAI_HS_Week07_AI_Models.pptx**
- 📎 Week 7 Google Colab Notebook - **NavigateAI_HS_Week07_AI_Models.ipynb** (upload to Google Colab: File > Upload Notebook)
- 📎 Week 7 Teacher Answer Key - **NavigateAI_HS_Week07_Teacher_Key.ipynb** (instructor only: complete solutions and expected outputs)
- 📎 Week 7 Opening Hook Code - **NavigateAI_HS_Week07_Hook_Code.docx** (copy-paste cheat sheet for live demo)

Instructor Preparation: Understanding the Hugging Face Pipeline

This lesson uses the Hugging Face transformers library, which gives students access to thousands of pre-trained AI models through a simple Python interface. The key function is pipeline(), which loads an AI model and makes it ready to use in one line of code.

Before class, run the Week 7 Colab notebook yourself. The first time you run each pipeline, it downloads the model (this takes 1-3 minutes depending on model size and internet speed). After the first download, the model is cached in the runtime and loads instantly. If you want to save class time, you can pre-run the notebook in your Colab account and share it with students, but students should also experience the download process since it teaches them that models are real files with real sizes.

The models run entirely in the Colab cloud runtime. They do NOT download to student laptops. When the Colab session ends, the models are deleted. This is important to communicate to students so they understand that their personal devices are not affected.

Class Flow - 90 Minutes

0:00 - 0:10 - Opening Hook: Watch AI Summarize a Wikipedia Article

This hook gives students an instant 'wow' moment by showing a real AI model working in just 3 lines of code.

Setup:

- Open a new Google Colab notebook on the projector. Have the code ready to paste (or type it live if you are comfortable).
- Say: 'For the last 6 weeks, you have been learning Python: variables, functions, loops, files, libraries. All of that was building toward today. Today, you use Python to run actual AI models. Not just talk about AI. Run it.'

Demo:

- Run the install cell first (this takes about 30 seconds):

```
%%capture

!pip install transformers==4.44.0
```

- Then run the summarization demo:

```
from transformers import pipeline

summarizer = pipeline('summarization')
```

- This line downloads the default summarization model (about 1.6 GB). It will take 2-3 minutes. While it downloads, explain: 'Right now, Python is downloading an AI model that was trained on millions of documents. It learned to read text and compress it into shorter summaries. The model is about 1.6 gigabytes, which is the weight of everything it learned during training. It is downloading to Google's cloud computer, not to your laptop.'
- Once downloaded, run the summary:

```
text = '''Artificial intelligence has transformed many industries over the past decade. From healthcare to finance, transportation to entertainment, AI systems are being deployed to automate tasks, analyze data, and make predictions. Machine learning, a subset of AI, allows computers to learn from data without being explicitly programmed. Deep learning, which uses neural networks with many layers, has been particularly successful in tasks like image recognition, natural language processing, and game playing. Major tech companies like Google, Meta, and OpenAI have invested billions of dollars in AI research and development.'''


result = summarizer(text, max_length=50, min_length=20)

print(result[0]['summary_text'])
```

- The AI produces a summary in about 2-3 seconds. Read it to the class. Say: 'Three lines of code. That is all it took to run an AI model that can read and summarize text. This is the same technology behind AI features in Google, ChatGPT, and dozens of other tools. And you just ran it yourself.'
- 'Today you will run four different AI models: one that summarizes text, one that generates text, one that answers questions, and one that detects sentiment. By the end of class, you will have used real AI in Python.'

0:10 - 0:30 - Core Lesson: How AI Models Work

Open the Week 7 PowerPoint presentation.

What Is a Pre-Trained Model? (5 min)

What Students Are Learning: Pre-trained models are the foundation of modern AI applications. Instead of training a model from scratch (which requires millions of data points, weeks of computing time, and expensive GPUs), developers use models that someone else already trained. This is like using a library: someone else did the hard work, and you benefit from it.

- Analogy: 'Think of a pre-trained model like hiring an expert. If you need legal advice, you do not go to law school for 7 years. You hire a lawyer who already did that. A pre-trained AI model already learned from millions of examples. You just give it a task.'
- Training vs. Using: Training is like going to school (expensive, slow, requires massive data). Using is like hiring the graduate (fast, cheap, just give them work). In this course, we USE pre-trained models. We do not train them.
- Where do models come from? Hugging Face (huggingface.co) is the largest open-source hub for AI models. It has over 500,000 models for every task you can think of. Students explored Hugging Face model cards back in Week 2.

Tokens, Context Windows, and Parameters (5 min)

What Students Are Learning: These three concepts help students understand what AI models can and cannot do, and why they sometimes fail. They will encounter these terms constantly in AI discussions and documentation.

- Tokens: AI models do not read words. They read tokens. A token is roughly 3/4 of a word. The word 'hamburger' might be split into 'ham', 'bur', 'ger' (3 tokens). The sentence 'I love Python' is about 4 tokens. This matters because models have a maximum number of tokens they can process at once.
- Context window: The maximum number of tokens a model can read at one time. If a model has a context window of 1,024 tokens, it can process about 750 words. If your text is longer, the model cannot read all of it at once. This is why the summarizer has max_length and min_length parameters, to control the output size.
- Parameters: The numbers inside the model that were learned during training. More parameters generally means a smarter model, but also bigger and slower. The summarization model (BART) has about 400 million parameters. GPT-4 has over 1 trillion. More parameters = more knowledge, but also more memory and processing time.
- Keep this section brief and conceptual. Students do not need to understand the math. They need to understand: tokens are how models see text, context windows limit how much text they can process, and parameters are what makes models smart.

The pipeline() Function (5 min)

What Students Are Learning: The pipeline() function is Hugging Face's simplest interface. It is one line of code that downloads a model, sets it up, and makes it ready to use. This is the function students will use in the capstone projects.

- The pattern is always the same: pipe = pipeline('task_name'). The task name tells Hugging Face what kind of model to load. The most common tasks are: 'summarization', 'text-generation', 'question-answering', and 'sentiment-analysis'.
- You can also specify a model: pipe = pipeline('summarization', model='facebook/bart-large-cnn'). If you do not specify a model, Hugging Face picks a default.
- Calling the model: result = pipe(your_input). The input format depends on the task. For summarization, you pass a string of text. For question-answering, you pass a dictionary with 'question' and 'context' keys.
- Live code the pipeline pattern step by step. Show that it is just: install, import, create pipeline, call it.

Four AI Tasks Demo (5 min)

- Briefly preview the four tasks students will try in the hands-on section. Show one line of code for each to demonstrate how simple the interface is:
- Summarization: summarizer('long text here') - Compress text into a shorter version.
- Text generation: generator('Once upon a time') - Continue writing from a prompt.
- Question answering: qa(question='Who invented Python?', context='Python was created by Guido van Rossum...') - Answer questions about a given passage.
- Sentiment analysis: sentiment('I love this class!') - Detect whether text is positive or negative.
- Say: 'Same pattern every time. Create a pipeline, pass in text, get results. Four different AI superpowers, all with the same interface.'

0:30 - 1:15 - Hands-On: Run Your First AI Models

What Students Are Learning and Why It Matters

This is the most transformative lesson in the course. Students go from writing basic Python programs to running actual AI models. The pipeline() function they learn today is the exact same function they will use in both capstone projects. In Capstone 1, they will use pipeline('summarization') and pipeline('question-answering') to build a PDF summarizer. In Capstone 2, they will use AI models to power a RAG chatbot. Everything they do today directly prepares them for those projects.

Task 1: Summarization (10 min)

- Students install transformers and run the summarizer on a paragraph of text (provided in the notebook).
- The first model download takes 2-3 minutes. Warn students: 'The first time you run a pipeline, it downloads the model. This is normal. After the first download, it loads instantly.'
- Students experiment with max_length and min_length parameters. What happens with max_length=30? max_length=100?
- Challenge: Students summarize a paragraph of their choice (paste from Wikipedia, a news article, or their own writing).

Task 2: Text Generation (10 min)

- Students create a text generation pipeline and give it a prompt to continue.

- Note: Text generation uses a different (smaller) model. It downloads quickly.
- Students experiment with prompts: 'The future of AI is', 'In a world where robots', 'Python is the best programming language because'.
- Discussion: 'Is the generated text factually accurate? Can you trust it? Why or why not?' This connects to AI bias and hallucination, topics from Week 1.

Task 3: Question Answering (10 min)

- Students create a question-answering pipeline and give it a context passage plus a question.
- The model reads the context and extracts the answer. It does NOT make up information; it only answers based on the context you provide.
- Students write their own context paragraphs and questions. Challenge: Can you stump the model?
- This task directly prepares students for Capstone 1 (PDF summarizer with Q&A) and Capstone 2 (RAG chatbot that answers questions from documents).

Task 4: Sentiment Analysis (10 min)

- Students create a sentiment analysis pipeline and test it with different sentences.
- The model returns POSITIVE or NEGATIVE with a confidence score.
- Students test edge cases: sarcasm, mixed feelings, neutral statements. 'The movie was not bad.' Is that positive or negative? The model might struggle with sarcasm.
- Discussion: 'Where would sentiment analysis be useful in the real world?' (Product reviews, social media monitoring, customer support, stock market predictions based on news sentiment.)

1:15 - 1:30 - Wrap-Up and Reflection

- Have 2-3 students share their most interesting result. Did the AI make any mistakes? What surprised them?
- Connect to capstones: 'The summarizer you used today? That is Capstone 1. You will build a full application that reads a PDF and summarizes it using this exact pipeline. The question-answering model? That is part of both capstone projects. You already know how to use the core technology. The capstones are about building a complete application around it.'
- 'Think about what you accomplished today. Six weeks ago, you wrote print("Hello World"). Today, you ran an AI model that can read, understand, and summarize text. That is the power of building knowledge step by step.'
- Exit ticket: "Name two of the four AI tasks you ran today and explain in one sentence what each one does."
- Preview Week 8: "Next week we learn Gradio, which lets you build a real user interface for your AI models. Instead of running code in a notebook, you will have buttons, text boxes, and a shareable link that anyone can use."

Homework

AI Exploration

SAMPLE WEEKLY POWERPOINT

Week 12: Introduction to RAG + Gemini API

Week 12

Introduction to RAG + Gemini API

Navigate AI | High School Semester 1 | Capstone 2 Begins

The Librarian Analogy

Bad Librarian

- Guesses based on training data
- Might get it wrong
- Makes things up
- Says "I don't know"

Good Librarian (RAG)

- Finds the right book
- Opens the relevant section
- Reads the actual text
- Answers from the document

RAG makes AI work like the good librarian.

Retrieval-Augmented Generation

Capstone 1 vs. Capstone 2

Both use documents — but in very different ways

Capstone 1: PDF Summarizer + Q&A

1 Upload ONE PDF

▼

2 Click Summarize:

▼

a. Extracts text (PyPDF2)

b. Truncates to ~750 words

c. BART writes a summary

3 Now Ask a Question:

▼

RoBERTa finds a phrase word-for-word

Step 3 only works after Step 2 because text extraction is inside the Summarize button. RoBERTa uses the extracted text, not the summary itself.

Limitations:

- Only 1 document at a time
- Truncates long docs (loses content)
- Q&A copies exact words (can't explain)
- No follow-ups, no memory, no personality

Capstone 2: RAG Chatbot

1 Load MANY docs into ChromaDB

▼

2 Retriever FINDS relevant chunks

▼

3 Only relevant text sent to Gemini

▼

4 Gemini WRITES an answer

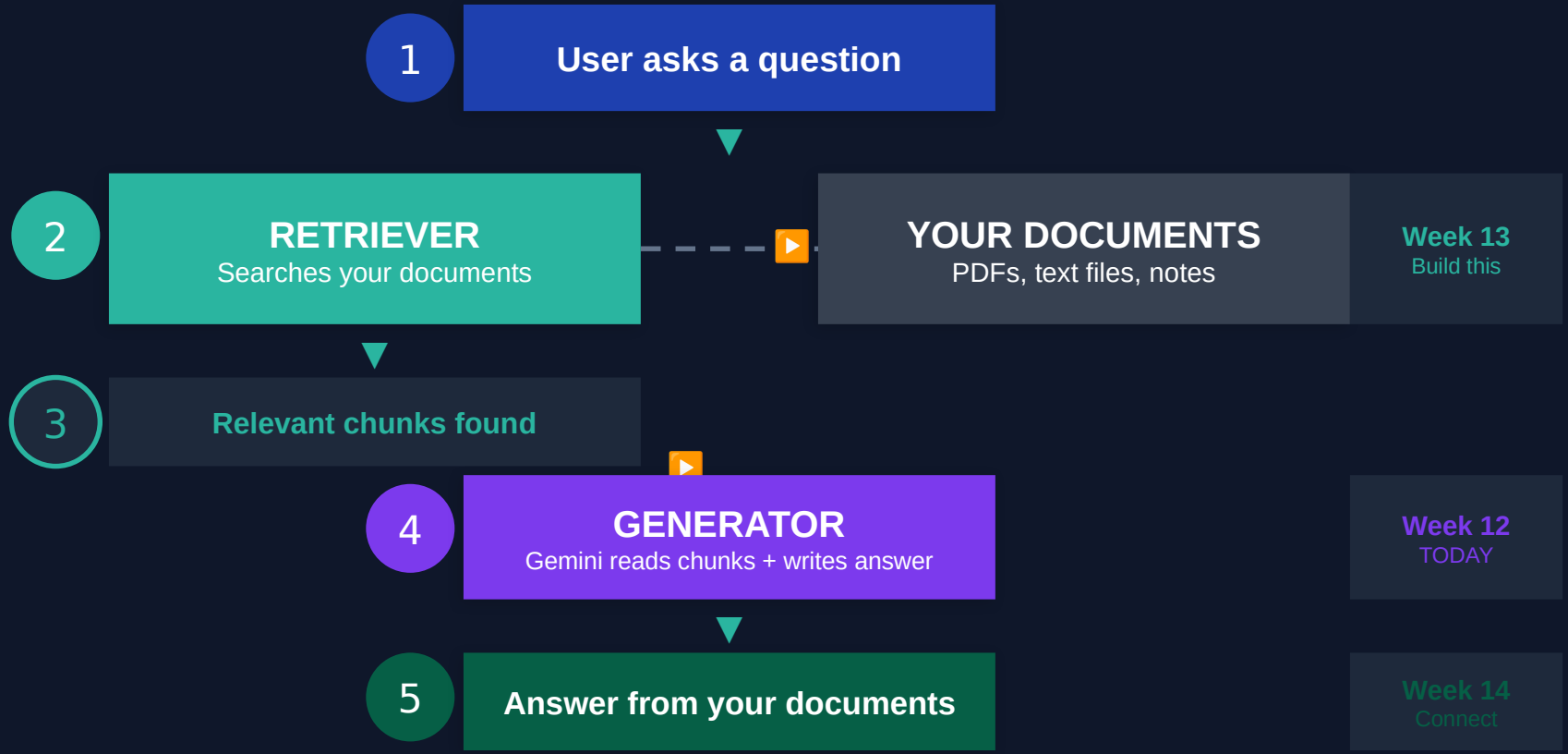
Gemini is generative: reads chunks, writes a new answer in its own words, explains, and handles follow-up questions

Advantages:

- Many documents at once
- Finds the right sections automatically
- Explains and rephrases (not just copy/paste)
- Full conversation with memory + system prompt

RAG Architecture

How data flows through the system



SAMPLE CODE NOTEBOOK

Week 12: Introduction to RAG + Gemini API

Navigate AI - Week 12: Introduction to RAG + Gemini API

Capstone 2 Begins!

Welcome to Capstone 2! Over the next 5 weeks, you will build a **RAG Chatbot** — a chatbot that answers questions by looking up information in real documents.

Today's goals:

1. **Understand RAG** — what it is and why it matters
2. **Set up the Gemini API** — connect to Google's AI model
3. **Make your first API call** — send a message and get a response
4. **Add conversation memory** — build a chatbot that remembers what you said
5. **See the problem RAG solves** — understand why a chatbot needs documents

---

Part 1: What is RAG?

The Librarian Analogy

Imagine you walk into a library and ask the librarian: "*What are the school's rules about cell phones?*"

A **bad librarian** would try to answer off the top of their head. They might get it wrong, make something up, or say "I don't know."

A **good librarian** would:

1. Go find the student handbook
2. Open it to the relevant section
3. Read the actual policy
4. Give you the answer based on what the document says

RAG (Retrieval-Augmented Generation) makes AI work like the good librarian.

How is this different from Capstone 1?

In Capstone 1, your Q&A app *did* use the document — but in a limited way. You extracted ALL the text, truncated it to 750 words, and the model searched through that one block.

RAG is smarter:

	Capstone 1 (Q&A)	Capstone 2 (RAG)
Documents	ONE PDF at a time	MANY documents at once
How it finds answers	Searches ALL the text (truncated to 750 words)	Retriever finds only the RELEVANT chunks
What it returns	Copies a phrase from the text	Writes a new answer in its own words
Follow-up questions?	No — each question is independent	Yes — full conversation with memory
Long documents?	Truncates — loses most of the content	Handles any length — searches smartly

Both use documents. But Capstone 1 is a **highlighter** (finds and copies a phrase). Capstone 2 is a **knowledgeable assistant** (reads your documents and has a conversation about them).

What is a system prompt?

A **system prompt** is a set of instructions that tells the AI how to behave. It's like giving an actor their character description before the show starts.

Without a system prompt: the AI is generic. With a system prompt: the AI has a specific personality, tone, and set of rules.

"But I never gave Claude or ChatGPT a system prompt..."

Actually, you did — you just didn't write it. When you use Claude, ChatGPT, or the Gemini chatbot website, the company (Anthropic, OpenAI, Google) includes a system prompt behind the scenes before your first message. It tells the AI how to behave, what safety guidelines to follow, and how to format responses. You never see it, but it's there in every conversation.

When you use the Gemini **API** from code (like we're doing today), there is no built-in system prompt. The API gives you the raw model. If you don't provide one, Gemini will still respond — but it won't have any specific personality or rules. **You** write the system prompt. That's what makes the API powerful for building custom apps — you control the behavior.

This is the same skill you will use in Week 15 when you design your RAG chatbot's personality.

In [ ]:
```
# ============================================================
# STEP 3: System prompts
# ============================================================

system_prompt = """You are a helpful cybersecurity tutor for high school students.
Keep explanations simple and use analogies.
If a student asks something unrelated to cybersecurity or technology,
politely redirect them back to the topic."""

user_question = 'What is a firewall?'

response = client.models.generate_content(
    model=MODEL,
    contents=user_question,
    config=types.GenerateContentConfig(
        system_instruction=system_prompt
    )
)
print(response.text)
```

In [ ]:
```
# Now try asking something off-topic. Does the system prompt work?

off_topic = 'What is the best pizza topping?'

response = client.models.generate_content(
    model=MODEL,
    contents=off_topic,
    config=types.GenerateContentConfig(
        system_instruction=system_prompt
    )
)
print(response.text)

# Document what happened:
# Did the AI answer the pizza question?
# Did it redirect to cybersecurity?
# How does this compare to BART (which had no system prompt ability)?
```

Part 6: Conversation Memory

Right now, each API call is independent. The AI doesn't remember what you said 10 seconds ago.

SAMPLE CODE NOTEBOOK

Week 9: PDF Extraction + Summarization Pipeline

```
# Show the results
print(f"Total characters extracted: {len(pdf_text)}")
print(f"Total words: {len(pdf_text.split())}")
print(f"\n--- First 500 characters ---")
print(pdf_text[:500])
```

Did it work? You should see the text from the PDF printed above.

Note: The extraction is not always perfect. PDFs sometimes have weird spacing or missing characters. That is normal. The text is good enough for summarization.

Try it yourself: Page-by-page extraction

What if you only want certain pages?

In [ ]:
```
# Extract text page by page
reader = PyPDF2.PdfReader('NavigateAI_HS_Week09_Sample.pdf')

print(f"This PDF has {len(reader.pages)} pages\n")

for i, page in enumerate(reader.pages):
    page_text = page.extract_text()
    word_count = len(page_text.split()) if page_text else 0
    print(f"Page {i+1}: {word_count} words")
    print(f"  Preview: {page_text[:100]}...\n")
```

Step 3: Summarize the Extracted Text

Now we use the BART summarization model from Hugging Face.

First run downloads the model (~1.6 GB). This takes 2-3 minutes. After that, it loads instantly.

In [ ]:
```
# Load the summarization model
# Using facebook/bart-large-cnn for high-quality summaries
summarizer = pipeline('summarization', model='facebook/bart-large-cnn')

model_name = summarizer.model.config._name_or_path
print(f"Summarizer loaded: {model_name}")
```

In [ ]:
```
# The summarization function
def summarize_text(text, max_length=150, min_length=50):
    """Summarize text using the BART model"""
    # BART can handle about 1,024 tokens (~750 words)
    # If the text is too long, we truncate it
    words = text.split()
    if len(words) > 750:
        text = ' '.join(words[:750])
        print(f"Note: Text truncated from {len(words)} words to 750 words (model limit)")

    result = summarizer(text, max_length=max_length, min_length=min_length, do_sample=False)
    return result[0]['summary_text']

# Test it!
summary = summarize_text(pdf_text)
print("SUMMARY:")
print(summary)
```

Experiment with summary length

Try different `max_length` values. How does the summary change?

In [ ]:
```
# Short summary
short = summarize_text(pdf_text, max_length=50, min_length=20)
print("SHORT SUMMARY:")
```

```
print(short)
print(f"({len(short.split())} words)\n")

# Medium summary
medium = summarize_text(pdf_text, max_length=150, min_length=50)
print("MEDIUM SUMMARY:")
print(medium)
print(f"({len(medium.split())} words)\n")

# Long summary
long_sum = summarize_text(pdf_text, max_length=300, min_length=100)
print("LONG SUMMARY:")
print(long_sum)
print(f"({len(long_sum.split())} words)")
```

Step 4: The Complete Pipeline

Now we connect extraction and summarization into one flow. This is the **capstone pattern**: Read Data > Process Data > Return Results.

In [ ]:
```
# THE COMPLETE PDF SUMMARIZATION PIPELINE

def summarize_pdf(pdf_path, max_length=150, min_length=50):
    """Complete pipeline: PDF file in, summary out"""
    # Step 1: Extract text
    print("Extracting text from PDF...")
    text = extract_text_from_pdf(pdf_path)
    word_count = len(text.split())
    print(f"Extracted {word_count} words\n")

    # Step 2: Summarize
    print("Generating summary...")
    summary = summarize_text(text, max_length=max_length, min_length=min_length)
    summary_words = len(summary.split())
    print(f"Summary: {summary_words} words\n")

    # Step 3: Return results
    return {
        'original_words': word_count,
        'summary_words': summary_words,
        'summary': summary
    }

# Run the complete pipeline!
result = summarize_pdf('NavigateAI_HS_Week09_Sample.pdf')

print("=" * 50)
print("PDF SUMMARY REPORT")
print("=" * 50)
print(f"Original document: {result['original_words']} words")
print(f"Summary: {result['summary_words']} words")
print(f"Compression: {result['summary_words']/result['original_words']:.0%} of original\n")
print(result['summary'])
```

You just built the backend of a real AI application!

What you accomplished today:

- Extracted text from a PDF using PyPDF2
- Summarized text using the BART AI model
- Connected them into a complete pipeline
- Controlled the output with max_length and min_length

Next week: We add question-answering (ask questions about the PDF) and build the Gradio web interface so anyone can use your app.

```
================================================================================
  NAVIGATE AI â€" HIGH SCHOOL SEMESTER 1
  AI Foundations & Application Development
  README
================================================================================

  Navigate AI LLC | navigateai.io | San Antonio, Texas
  2026

================================================================================
  PROGRAM OVERVIEW
================================================================================

This is a 16-week AI curriculum for high school students (grades 9-12) with
zero prior coding experience. Students learn Python, AI fundamentals, and
build two working AI applications:

  Capstone 1 (Weeks 9-11): PDF Summarizer & Q/A Application
    Built with PyPDF2, Hugging Face transformers, and Gradio

  Capstone 2 (Weeks 12-16): RAG Chatbot
    Built with Gemini API, ChromaDB, and Gradio

Each week includes a 90-minute lesson plan with PowerPoint slides, a Google
Colab notebook for hands-on activities, a teacher answer key, and supporting
documents.

================================================================================
  WHAT'S INCLUDED
================================================================================

  Curriculum_Guide.docx        Main 16-week curriculum with detailed lesson plans
  Teacher_Resources/           PowerPoints, teacher keys, and companion guides
  Colab_Notebooks/             Google Colab notebooks for student activities
  Certificate_of_Completion.docx   Printable certificate (landscape, fill-in)
  README.txt                   This file

================================================================================
  SEMESTER STRUCTURE
================================================================================

  UNIT 1: AI Foundations & the Builder Mindset (Weeks 1-2)
    Week 1   What Is AI + Data Literacy
    Week 2   AI Tools & the Builder Mindset

  UNIT 2: Python for AI (Weeks 3-8)
    Week 3   Google Colab + Python Fundamentals
    Week 4   Control Flow: Conditionals & Loops
    Week 5   Functions & Working with Files
    Week 6   Libraries, pip & Data Manipulation
    Week 7   AI/ML Concepts in Code
    Week 8   Building User Interfaces with Gradio

  UNIT 3: PDF Summarization App â€" Capstone 1 (Weeks 9-11)
    Week 9   PDF Extraction + Summarization Pipeline
    Week 10  Question-Answering + Gradio App
    Week 11  Polish, Present & Reflect

  UNIT 4: RAG Chatbot â€" Capstone 2 (Weeks 12-16)
    Week 12  Introduction to RAG + Gemini API
    Week 13  Document Loading, Chunking & Vector Stores
    Week 14  The Full RAG Pipeline
    Week 15  Customize & Build Your RAG Chatbot
    Week 16  Demo Day â€" Present Your RAG Chatbot

================================================================================
  FILES BY WEEK
================================================================================

  WEEK 1: What Is AI + Data Literacy
    NavigateAI_HS_Week01_What_Is_AI.pptx              PowerPoint
    NavigateAI_HS_Week01_AI_Cards.docx                "AI or Not AI?" Sorting Cards
    NavigateAI_HS_Week01_Sample_Dataset.csv           Sample dataset for Google Sheets

  WEEK 2: AI Tools & the Builder Mindset
    NavigateAI_HS_Week02_AI_Tools.pptx                PowerPoint

  WEEK 3: Google Colab + Python Fundamentals
    NavigateAI_HS_Week03_Python_Fundamentals.pptx     PowerPoint
    NavigateAI_HS_Week03_Python_Fundamentals.ipynb    Student Colab notebook
```

```
  NavigateAI_HS_Week03_CodeSnippets.docx            Code snippets reference

WEEK 4: Control Flow: Conditionals & Loops
  NavigateAI_HS_Week04_Conditionals_Loops.pptx      PowerPoint
  NavigateAI_HS_Week04_Conditionals_Loops.ipynb     Student Colab notebook

WEEK 5: Functions & Working with Files
  NavigateAI_HS_Week05_Functions_Files.pptx          PowerPoint
  NavigateAI_HS_Week05_Functions_Files.ipynb         Student Colab notebook
  NavigateAI_HS_Week05_Teacher_Key.ipynb             Teacher answer key

WEEK 6: Libraries, pip & Data Manipulation
  NavigateAI_HS_Week06_Libraries_Data.pptx           PowerPoint
  NavigateAI_HS_Week06_Libraries_Data.ipynb          Student Colab notebook
  NavigateAI_HS_Week06_Movies.csv                    Movie dataset
  NavigateAI_HS_Week06_Hook_Code.docx                Opening hook code (copy-paste)
  NavigateAI_HS_Week06_Teacher_Key.ipynb             Teacher answer key
  NavigateAI_HS_How_To_Explore_Libraries.docx        Student take-home reference

WEEK 7: AI/ML Concepts in Code
  NavigateAI_HS_Week07_AI_Models.pptx               PowerPoint
  NavigateAI_HS_Week07_AI_Models.ipynb              Student Colab notebook
  NavigateAI_HS_Week07_Teacher_Key.ipynb             Teacher answer key
  NavigateAI_HS_Week07_Hook_Code.docx                Opening hook code (copy-paste)

WEEK 8: Building User Interfaces with Gradio
  NavigateAI_HS_Week08_Gradio.pptx                  PowerPoint
  NavigateAI_HS_Week08_Gradio.ipynb                 Student Colab notebook
  NavigateAI_HS_Week08_Hook_Code.docx                Opening hook code (copy-paste)
  NavigateAI_HS_Week08_Teacher_Key.ipynb             Teacher answer key

WEEK 9: PDF Extraction + Summarization Pipeline
  NavigateAI_HS_Week09_PDF_Summarizer.pptx          PowerPoint
  NavigateAI_HS_Week09_PDF_Summarizer.ipynb          Student Colab notebook
  NavigateAI_HS_Week09_Sample.pdf                    Sample PDF for testing
  NavigateAI_HS_Week09_Teacher_Key.ipynb             Teacher answer key
  NavigateAI_HS_Capstone1_Architecture.docx          Capstone 1 architecture guide

WEEK 10: Question-Answering + Gradio App
  NavigateAI_HS_Week10_QA_Gradio.pptx               PowerPoint
  NavigateAI_HS_Week10_QA_Gradio.ipynb              Student Colab notebook
  NavigateAI_HS_Week10_Teacher_Key.ipynb             Teacher answer key

WEEK 11: Polish, Present & Reflect
  NavigateAI_HS_Week11_Polish_Present.ipynb          Student Colab notebook
  NavigateAI_HS_Week11_Teacher_Key.ipynb             Teacher answer key
  NavigateAI_HS_Week11_Reflection.docx               Capstone 1 reflection worksheet (printable)

WEEK 12: Introduction to RAG + Gemini API
  NavigateAI_HS_Week12_RAG_Intro.pptx               PowerPoint
  NavigateAI_HS_Week12_RAG_Intro.ipynb              Student Colab notebook
  NavigateAI_HS_Week12_Teacher_Key.ipynb             Teacher answer key
  NavigateAI_HS_Week12_Companion_Guide.docx          Companion guide

WEEK 13: Document Loading, Chunking & Vector Stores
  NavigateAI_HS_Week13_Doc_Chunking_ChromaDB.pptx    PowerPoint (with speaker notes)
  NavigateAI_HS_Week13_Doc_Chunking_ChromaDB.ipynb   Student Colab notebook
  NavigateAI_HS_Week13_Teacher_Key.ipynb             Teacher answer key
  NavigateAI_HS_Week13_Companion_Guide.docx          Companion guide

WEEK 14: The Full RAG Pipeline
  NavigateAI_HS_Week14_RAG_Pipeline.pptx             PowerPoint (with speaker notes)
  NavigateAI_HS_Week14_RAG_Pipeline.ipynb            Student Colab notebook
  NavigateAI_HS_Week14_Teacher_Key.ipynb             Teacher answer key (includes API key management guide)
  NavigateAI_HS_Week14_Companion_Guide.docx          Companion guide

WEEK 15: Customize & Build Your RAG Chatbot
  NavigateAI_HS_Week15_Customize_Build.pptx          PowerPoint (with speaker notes)
  NavigateAI_HS_Week15_Customize_Build.ipynb         Student Colab notebook
  NavigateAI_HS_Week15_Teacher_Key.ipynb             Teacher answer key (includes presentation rubric)
  NavigateAI_HS_Week15_Companion_Guide.docx          Companion guide

WEEK 16: Demo Day â€" Present Your RAG Chatbot
  NavigateAI_HS_Week16_Demo_Day.pptx                 PowerPoint (with speaker notes)
  NavigateAI_HS_Week16_Teacher_Key.ipynb             Teacher answer key (rubric, grading guide, feedback form)
  NavigateAI_HS_Week16_Companion_Guide.docx          Companion guide

CERTIFICATE
  NavigateAI_HS_Certificate_of_Completion.docx       Printable certificate (landscape)

================================================================================
```

SOFTWARE AND TOOLS (ALL FREE)

Google Colab	Browser-based Python environment (free with Google account)
Ollama	Local AI model runner (Week 2 only, free from ollama.com)
Gemini API	Google AI API (free tier, no credit card required)
Hugging Face	Free model hub and inference API
ChromaDB	Free, open source vector store (runs in Colab)
Gradio	Free, open source UI library for Python

No software purchases required. All tools run in the browser or install free.
Students need: a laptop with internet access and a Google account.

TECHNICAL NOTES

Hugging Face transformers: Pin to version 4.44.0 in all install cells.
Newer versions removed the summarization and question-answering pipeline task aliases. All notebooks use: !pip install transformers==4.44.0

Gemini API keys: Free tier rate limits vary by model and change periodically. See the Week 14 Teacher Key for current limits and the API key management guide (creating multiple projects for key rotation during testing and presentations).

Model downloads: AI models download to the Colab runtime, not to student laptops. The largest model (facebook/bart-large-cnn) is approximately 1.6 GB. Model files are lost when the Colab session disconnects and must be re-downloaded on the next session.

ChromaDB: Runs in-memory in Colab. All stored data is lost when the session disconnects. Students re-run the pipeline each session. Persistent storage is not needed for this curriculum.

PyCryptodome: Required for reading AES-encrypted PDFs (including the Catan rulebook used as the demo PDF). All install cells in Weeks 13-15 include PyCryptodome.

GETTING STARTED

1. Read the Curriculum Guide (NavigateAI_HS_Semester1_Curriculum.docx)
   Start with the Program Overview, Course Description, and Class Format sections to understand the semester structure.

2. Prepare for Week 1
   Download the Week 1 PowerPoint and AI Cards document.
   Set up the Google Sheets sample dataset (instructions in the curriculum).

3. For each subsequent week
   Open the week's lesson plan in the Curriculum Guide.
   Download the PowerPoint, Colab notebook, and any supporting files.
   Review the Teacher Answer Key Colab notebook before class.
   Read the Companion Guide (Weeks 12-16) for deeper concept explanations.

4. Colab notebooks
   Upload .ipynb files to Google Colab: File > Upload Notebook.
   Students should each make their own copy: File > Save a copy in Drive.

CONTACT

Navigate AI LLC
San Antonio, Texas
navigateai.io
navigate.ai.llc@gmail.com