Day-1

DAY 1 (MON DEC 16): GET IT WORKING

BEFORE HACKATHON STARTS (Do this Sunday night!)

Setup Checklist:

```
# 1. Create project folder
mkdir MicroscoPilot
cd MicroscoPilot

# 2. Create Python virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# 3. Install basics
pip install jupyter numpy matplotlib anthropic pillow
```

Get Anthropic API Key:

1. Go to: https://console.anthropic.com/
2. Sign up (you get $5 free credits!)
3. Create API key
4. Save it in `.env` file:

```
ANTHROPIC_API_KEY=your_key_here
```

Clone DTMicroscope:

```
git clone https://github.com/pycroscopy/DTMicroscope.git
cd DTMicroscope
pip install -e .
cd ..
```

Test it works:

```
cd DTMicroscope/notebooks
jupyter notebook
```

```
# Open and run: 01-AFM-Demo.ipynb
# If you see a microscope image, you're good!
```

HOUR 1-3 (9:00 AM - 12:00 PM): OPENING & FOUNDATION

9:00-10:30 AM: Watch Opening Ceremony

- Take notes on challenges they mention
- Screenshot any specific tasks
- Note judging criteria
- Join Slack immediately

10:30 AM - 12:00 PM: Build Project Skeleton

PROMPT #1 TO CLAUDE/CURSOR:

```
I'm building MicroscoPilot for the Microscopy Hackathon. I'm a beginner in
ML.

PROJECT: Autonomous microscopy agent using Claude Vision API + DTMicroscope
simulator

REQUIREMENTS:
- Use DTMicroscope's AFM digital twin (already installed)
- Agent makes decisions using Claude Sonnet 4 vision API
- Simple autonomous exploration: move around, capture images, find
interesting features
- Track discoveries and show visualizations

CREATE:
Project structure:
/MicroscoPilot
  /src
    __init__.py
    agent.py          # Main agent logic
    vision.py         # Claude vision API calls
    microscope.py     # DTMicroscope wrapper
    memory.py         # Store what agent has seen
    visualizer.py     # Display results
  /tests
```

```
    test_basic.py
  /outputs             # Save results here
  /data                # Sample images
  main.py              # Run the agent
  requirements.txt
  README.md
  .env.example
  .gitignore

Use SIMPLE Python (I'm learning!):
- Clear variable names
- Lots of comments explaining what each part does
- Error handling with helpful messages
- Logging to see what's happening

Start with main.py that imports everything and has a basic run loop.
```

WHAT YOU'LL GET: Full project structure. Test that everything imports correctly.

HOUR 4-6 (12:00 PM - 2:00 PM): VISION MODULE

PROMPT #2:

```
Build src/vision.py - the module that uses Claude Vision API to analyze
microscope images.

REQUIREMENTS:

1. Function: analyze_image(image_path) -> dict
   - Loads image from file
   - Converts to base64 (required for API)
   - Sends to Claude Sonnet 4 with specialized prompt
   - Returns structured analysis

2. Prompt engineering (CRITICAL - this is where intelligence comes from):
   Tell Claude it's analyzing Atomic Force Microscopy (AFM) images and
should:
   - Identify features: grain boundaries, defects, particles, rough regions,
smooth regions
   - Rate image quality (0-10)
   - Assess interestingness (is there something worth investigating?)
   - Suggest next action (zoom in, move left/right/up/down, adjust settings)
   - Provide confidence score
   - Explain reasoning

3. Response parsing:
   - Extract Claude's response into Python dict
   - Handle errors gracefully (API might be slow or fail)
   - Log the analysis to a file

4. Include retry logic (API calls can fail):
   - Try up to 3 times
   - Wait 2 seconds between retries
   - Give helpful error message if all fail

TEACHING NOTES:
- Explain what base64 encoding is (turning image into text)
- Show example of the API call format
- Add comments explaining each step
- Include a test function at the bottom that analyzes a sample image

IMPORTANT: Make it work with FREE API tier (don't spam requests)
```

WHAT YOU'LL GET: Working vision module. Test with:

```
python src/vision.py --test data/sample_afm_image.png
```

HOUR 7-8 (2:00 PM - 4:00 PM): MICROSCOPE CONTROLLER

PROMPT #3:

```
Build src/microscope.py - wrapper around DTMicroscope for easy control.

REQUIREMENTS:

1. Class: MicroscopeController
   - Initialize DTMicroscope AFM simulator
   - Provide simple methods like:
     * move_to(x, y) - move stage to coordinates
     * capture_image() - get current image
     * get_current_position() - where are we?
     * zoom_in() / zoom_out() - adjust magnification
     * get_state() - full microscope status

2. Safety checks:
   - Don't allow moving outside valid boundaries (0 to sample_size)
   - Validate all inputs
   - Return clear error messages

3. State tracking:
   - Remember where we've been (position history)
   - Track number of images captured
   - Store settings used for each image

4. Simulation mode (for testing without DTMicroscope):
   - If DTMicroscope not available, create fake microscope
   - Generate random but realistic images
   - Still track positions correctly

TEACHING NOTES:
- Explain what a "wrapper class" is (simpler interface to complex code)
- Show how to check if a library is installed
- Document the coordinate system (x,y axes)
- Add diagram in comments showing how positions map to sample

MAKE IT BEGINNER-FRIENDLY:
- Use descriptive method names
- Include docstrings with examples
- Add assertions to catch mistakes
- Print warnings for weird inputs
```

WHAT YOU'LL GET: Microscope control module. Test with:

```
python src/microscope.py --demo
```

HOUR 9-10 (4:00 PM - 6:00 PM): BASIC AGENT BRAIN

PROMPT #4:

Build src/agent.py - the autonomous decision-making agent.

This is THE CORE of your project. The agent needs to:
1. Explore the sample intelligently
2. Recognize interesting features
3. Learn from what it sees
4. Make decisions about where to go next

REQUIREMENTS:

Class: AutonomousAgent

__init__(microscope, vision_analyzer):
  - Takes microscope controller and vision module
  - Initialize empty memory
  - Set starting strategy (e.g., random exploration)

explore(num_steps=100):
  - Main loop that runs autonomously
  - For each step:
    1. Capture current image
    2. Analyze with vision AI
    3. Decide next action based on analysis
    4. Execute action
    5. Store results in memory
    6. Print progress update

decide_next_action(vision_analysis, current_state):
  - Decision logic:
    * If interesting (score > 7): zoom in and capture more
    * If seen before: skip and move elsewhere
    * If quality is poor: adjust settings
    * If nothing interesting for N steps: jump to random new area
    * If near boundary: turn around
  - Return action: {'type': 'move', 'x': 10, 'y': 20}

TEACHING MOMENT - Explain the agent loop:
"The agent is just a while loop! It keeps running until you tell it to stop. Each loop: sense the world (vision) → think (decide_next_action) → act (move microscope)"

KEEP IT SIMPLE FOR NOW:
- No fancy ML yet, just if/else logic
- Random exploration is fine
- Focus on getting the loop working
- We'll add intelligence tomorrow

```
LOGGING:
- Print emoji-based updates so demos look cool:
  "🔍 Analyzing position (45, 120)..."
  "✨ Interesting feature detected! Moving closer..."
  "📍 Exploring new region..."
  "😴 Nothing interesting here, moving on..."
```

WHAT YOU'LL GET: Basic working agent! Test with:

```
python main.py --steps 20
```

HOUR 11 (6:00 PM - 7:00 PM): INTEGRATION TEST

Your Tasks (No Claude needed):

1. **Create main.py:**

```python
# This is where everything comes together
from src.microscope import MicroscopeController
from src.vision import VisionAnalyzer
from src.agent import AutonomousAgent

# Initialize components
microscope = MicroscopeController()
vision = VisionAnalyzer(api_key="your_key")
agent = AutonomousAgent(microscope, vision)

# Run exploration
print("🚀 Starting autonomous exploration...")
results = agent.explore(num_steps=50)
print(f"✅ Complete! Found {len(results['discoveries'])} interesting features")
```

2. **Test everything:**

```
python main.py
```

EXPECTED: Agent moves around, captures images, makes decisions. You see progress logs. No crashes!

3. **Debug if needed** (save Claude messages for actual bugs):
    - Read error messages carefully
    - Check that all imports work
    - Verify API key is loaded
    - Make sure DTMicroscope is accessible

HOUR 12 (7:00 PM - 8:00 PM): DINNER + TEAM CHECK-IN

Your Tasks:

- EAT REAL FOOD (seriously, brain fuel!)
- Review what works and what doesn't
- Push code to GitHub
- Post in Slack: "Built autonomous microscopy agent. Looking for teammates interested in [ML/microscopy/visualization]. Have working prototype!"
- Plan tomorrow's features

GitHub Setup:

```
git init
git add .
git commit -m "Day 1: Basic agent working"
git remote add origin https://github.com/yourusername/MicroscoPilot.git
git push -u origin main
```

HOUR 13-14 (8:00 PM - 10:00 PM): BASIC VISUALIZATION

PROMPT #5:

Build src/visualizer.py - real-time visualization of agent exploration.

REQUIREMENTS:

Create a dashboard using matplotlib that shows:

1. Main window with 4 panels:

   TOP LEFT: Current microscope view
   - Show the latest image agent captured
   - Mark current position

   TOP RIGHT: Exploration map
   - 2D heatmap of where agent has been
   - Color: blue = never visited, yellow = visited once, red = visited many times
   - Show current position as red dot
   - Mark discovered features with stars ⭐

   BOTTOM LEFT: Discoveries timeline
   - Line graph: time vs. number of features found
   - Show when interesting things were discovered

   BOTTOM RIGHT: Statistics
   - Text display showing:
     * Total steps taken
     * Areas explored
     * Features found
     * Current strategy
     * Time elapsed

2. Real-time updates:
   - Update every time agent takes a step
   - Don't freeze (use matplotlib.animation or just plt.pause())
   - Save final state as PNG

3. Export capability:
   - Save exploration video as MP4 (optional, might need ffmpeg)
   - Save heatmap as high-res image
   - Export data as JSON

TEACHING NOTE:
"Visualization is CRITICAL for demos! Judges need to SEE your agent working.
This is the difference between 'cool code' and 'OH WOW' reaction."

KEEP IT FUNCTIONAL, NOT FANCY (for now):

```
- Basic matplotlib is fine
- Don't worry about making it beautiful yet
- Focus on showing the agent's behavior clearly
- Tomorrow we'll make it pretty
```

WHAT YOU'LL GET: Live visualization. Test with:

```
python src/visualizer.py --demo  # Shows fake data
python main.py --visualize      # Shows real agent
```

END OF DAY 1 CHECKLIST

☑ ~~Project structure created~~
☑ ~~Vision module working (can analyze images with Claude)~~
☑ ~~Microscope controller working (can move and capture)~~
☑ ~~Basic agent working (explores autonomously)~~
☑ ~~Visualization shows agent behavior~~
☑ ~~Code on GitHub~~
☑ ~~You understand what each part does~~
☑ ~~You have a demo you can show~~

EXPECTED STATE: You have a working autonomous agent that:

- Moves around the microscope sample
- Captures images
- Uses Claude Vision to analyze them
- Makes simple decisions
- Shows its exploration visually

It's basic, but it WORKS! That's more than many teams will have.