1. What are you trying to do?

A) I aim to design and implement a hardware-accelerated OCR system optimized for handwritten text recognition, focusing on segmentation and preprocessing acceleration. The goal is to outperform traditional software-only OCR systems (e.g., Tesseract) in latency, power efficiency, and modularity, especially for embedded or edge AI applications.

2. How is it done today, and what are the limitations of current practice?

A) Current OCR systems (e.g., Tesseract, PaddleOCR, EasyOCR) run entirely on general-purpose CPUs/GPUs and are optimized for printed text. Limitations:

- High latency due to sequential image processing and CNN inference.
- Not specialized for handwritten input, leading to lower accuracy.
- No hardware offloading, resulting in inefficiency for embedded/low-power use cases.
- No pipelining between preprocessing, segmentation, and classification stages.

3. What is new in your approach, and why do you think it will be successful?

- Hardware acceleration of bottlenecks (thresholding, contour extraction, segmentation) using Verilog modules.
- Focused on handwriting, not general OCR.
- Combines HW/SW co-design: lightweight CNN in SW, heavy pre-processing in HW.
- Enables faster real-time OCR on low-cost hardware (FPGAs, SoCs).
- Modular pipeline allows selective offloading to improve speed and reduce CPU load.

4. Who cares?

- Developers targeting embedded OCR devices (e.g., smart pens, mobile OCR scanners).
- Assistive tech (e.g., reading aids for visually impaired) requiring fast local OCR.
- Industrial applications like postal address scanning, document digitization, or form reading in low-resource environments.

- AI engineers interested in low-power AI edge solutions.

5. What are the risks and the payoffs?

Risks:

- HW implementation may not provide significant improvement without careful profiling.
- Accuracy might drop if segmentation isn't tuned well.
- Integration with existing OCR frameworks may require bridging SW/HW gaps.

Payoffs:

- 2–5× latency improvement for segmentation/preprocessing.
- Lower energy consumption, enabling portable use cases.
- Demonstrates real hardware/software co-design benefits.

6. How much will it cost?

- Zero hardware cost for simulation and synthesis using tools like Icarus Verilog, Verilator, or OpenLane.
- Optional FPGA cost (~$50–100) for physical implementation (e.g., Nexys A7).
- No licensing cost — all based on open-source tools and models.

7. How long will it take?

- Week 1–2: Research OCR pipelines, pick datasets (e.g., EMNIST, IAM).
- Week 3–4: Software baseline using Python + OpenCV + CNN.
- Week 5–6: Identify bottlenecks (thresholding, contour extraction).
- Week 7–8: Implement and simulate hardware blocks (Verilog).
- Week 9–10: Integrate HW/SW and benchmark results.

8. What are the midterm and final "exams" to check for success?

Midterm Checkpoints:

- Completed software pipeline with accurate text extraction.
- Identified HW bottleneck and module candidate (e.g., thresholding unit).
- Simulated Verilog block showing correct output.

Final Exams:

- Compare CPU-only vs HW-accelerated performance (latency, power).
- Show modular block diagram with working HW/SW co-simulation.
- Show recognition of real handwritten input with better speed or lower power.