Day 6:

Python Systems Engineering

🏗️ "From Raw Data to Structured Systems — OOP, Data Structures, and File Mastery"

📋 Daily Agenda (150 min active)

- Create Phase (45 min): Build a Student Grade System.
- Fix Phase (90 min): Work on tickets #701–#705 simulating real bugs/issues.
- Verification (15 min): Client proof + GitHub commit.

🛠️ CREATE PHASE (45 min)

Task:
Build a simple Student Grade System with these features:

- ☐ A Student class (name, id, grades).
- ☐ A method to calculate GPA.
- ☐ Store multiple students in a list/dict.
- ☐ Save/load data to grades.txt.

✅ Deliverable after create: *a working but messy v1*.

👷👷 FIX PHASE (Tickets)

🎟️Ticket #701 — GPA Calculation Broken
Client Note:

"GPA results are wrong. 4.3 scale not supported."

```
class student:
 def __init__(self,name,id,grades):
  self.name = name
  self.id = id
  self.grades = grades
 def GPA(self):
  return sum(self.grades)/len(self.grades)
```

Focus:
- Class naming → CamelCase.
- Method naming → snake_case.
- Handle GPA calculation with scale.
- PEP 8 For Classes (**FYI**)

🎟️Ticket #702 — Missing File Crash
Client Note:

"System crashes if grades.txt is missing."

```
f = open("grades.txt","r")
data = f.read()
print(data)
```

Focus:
- Use try/except.
- Check file existence with os.path.exists().
- Use with open(...).

🎟️Ticket #703 — System Fails with 1000+ Students
Client Note:

"Performance issues — lists not scaling."

```
students = [("Ali",101),("Sara",102),("Omar",103)]
print(students[1000])
```

Focus:
- Replace list with dict or defaultdict.
- Access by key (ID).
- Clean data structures.

🎟️Ticket #704 — No Audit Trail
Client Note:

"No logs — we can't see when grades are saved/loaded."

```
with open("grades.txt","w") as f:
 f.write("Ali, A")
```

Focus:
- Add logging module.
- Log read/write operations.

🎟️Ticket #705 — Code is Unreadable
Client Note:

"Intern code violates standards. Impossible to maintain."

```
class stu: pass
def gpA(x): return sum(x)/len(x)
```

Focus:
- Refactor to clean OOP with PEP 8.
- Add docstrings.
- Separate code into functions/classes.

📲 CLIENT VERIFICATION (15 min)

1. Send text: *"Deploying fixes for Tickets #701–#705 per PEP 8 — live in 15 min."*
2. Run acceptance check:
   - Add 3 students, calculate GPA, save to file, log operations.
3. Save proof (screenshot of file + logs).
4. Commit to GitHub:
   - v1-naive branch (original).
   - v2-hardened branch (fixed).

✅ END-OF-DAY DELIVERABLES

- ☐ Fixed code for Tickets #701–#705.
- ☐ Client texts saved.
- ☐ grades.txt file with sample data.
- ☐ GitHub repo with two branches (v1-naive, v2-hardened).
- ☐ One-line business impact note per fix.
- ☐ Post On **Linkedin**
