Name: Faizan Akbar.

Reg no: L1F23BSCS0277

Section: E-14

Assignment # 1

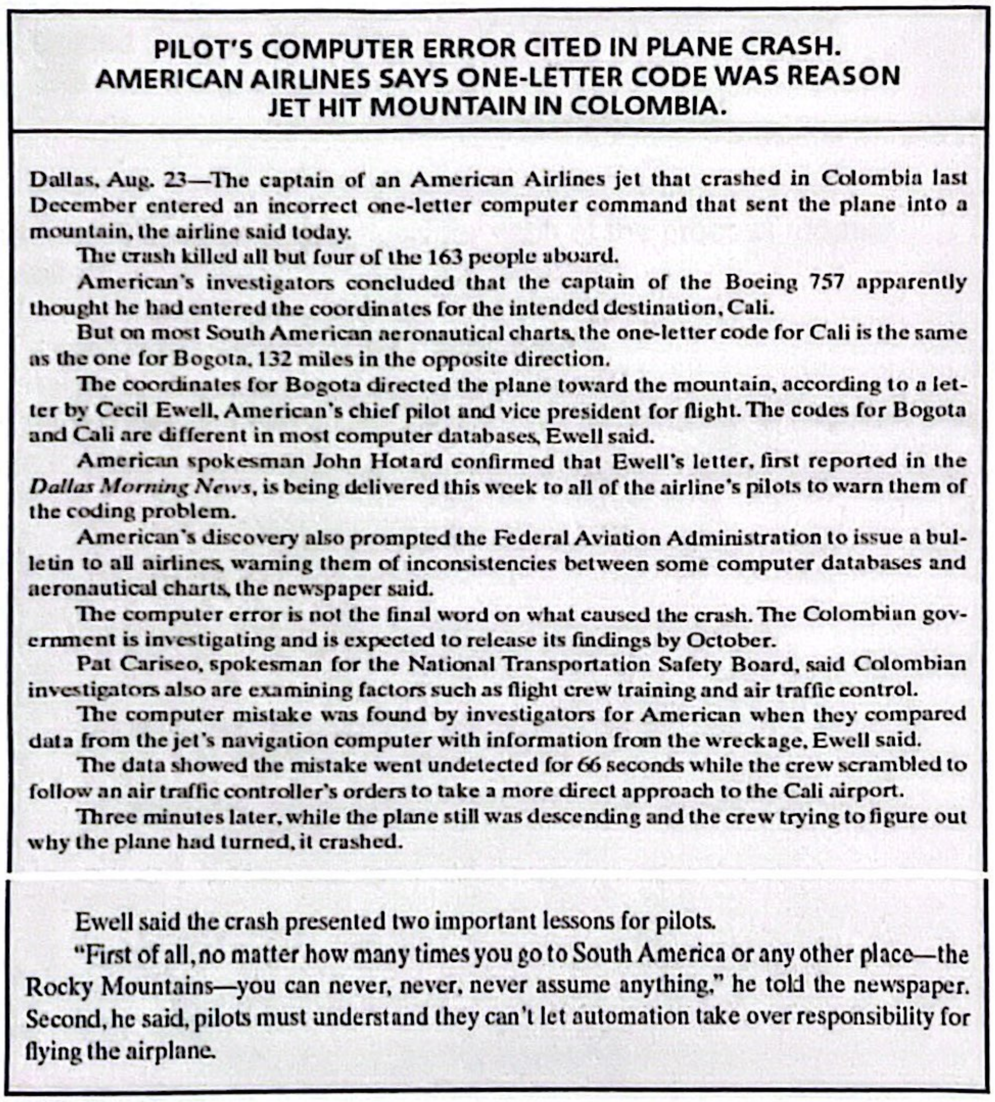
1 (Deadline 28/October/2025 – 11:50pm)

Note:

1. Assignment should be hand written and converted to pdf or word before submitting it on portals.
2. Typed assignment will get -2.5 marks as penalty.
3. Assignment submitted after deadline but within 24 hours will get -2.5 marks as penalty.
4. Assignment submitted after 24 hours but before 48 hours will get -5 marks as penalty.
5. Assignment submitted after 48 hours will get 00 marks.
6. Plagiarized assignments will get 00 marks.

Part 1

1. The following article appeared in the *Washington Post* (Associated Press 1996):



PILOT'S COMPUTER ERROR CITED IN PLANE CRASH. AMERICAN AIRLINES SAYS ONE-LETTER CODE WAS REASON JET HIT MOUNTAIN IN COLOMBIA.

Dallas, Aug. 23—The captain of an American Airlines jet that crashed in Colombia last December entered an incorrect one-letter computer command that sent the plane into a mountain, the airline said today.

The crash killed all but four of the 163 people aboard.

American's investigators concluded that the captain of the Boeing 757 apparently thought he had entered the coordinates for the intended destination, Cali.

But on most South American aeronautical charts, the one-letter code for Cali is the same as the one for Bogota, 132 miles in the opposite direction.

The coordinates for Bogota directed the plane toward the mountain, according to a letter by Cecil Ewell, American's chief pilot and vice president for flight. The codes for Bogota and Cali are different in most computer databases, Ewell said.

American spokesman John Hotard confirmed that Ewell's letter, first reported in the *Dallas Morning News*, is being delivered this week to all of the airline's pilots to warn them of the coding problem.

American's discovery also prompted the Federal Aviation Administration to issue a bulletin to all airlines, warning them of inconsistencies between some computer databases and aeronautical charts, the newspaper said.

The computer error is not the final word on what caused the crash. The Colombian government is investigating and is expected to release its findings by October.

Pat Cariseo, spokesman for the National Transportation Safety Board, said Colombian investigators also are examining factors such as flight crew training and air traffic control.

The computer mistake was found by investigators for American when they compared data from the jet's navigation computer with information from the wreckage, Ewell said.

The data showed the mistake went undetected for 66 seconds while the crew scrambled to follow an air traffic controller's orders to take a more direct approach to the Cali airport.

Three minutes later, while the plane still was descending and the crew trying to figure out why the plane had turned, it crashed.

Ewell said the crash presented two important lessons for pilots.

"First of all, no matter how many times you go to South America or any other place—the Rocky Mountains—you can never, never, never assume anything," he told the newspaper. Second, he said, pilots must understand they can't let automation take over responsibility for flying the airplane.

Question 1) Is this article evidence that we have a software crisis?

Question 2) How is aviation better off because of software engineering?

Question 3) What issues should be addressed during software development so that problems like this will be prevented in the future?

Part 2

- Prototyping model
- V model
- Phased Development
- Spiral Model
- Unified Process Modeling
- Extreme Programming
- Scrum

Question: Answer all 3 questions for each of the process models mentioned.

1. Give 02 benefits of using the model?
2. Give 02 drawbacks of using the model?
3. How does the model handle a significant change in requirements, late in the development?

~~~~~~~~~~~BEST OF LUCK~~~~~~~~~~~
~~~~~~~~~~~

Part#1

Question #1:

Yes, it is evidence of a software crisis in critical system design.

- Data Inconsistency: The use of the same single letter code for two different cites. (Cali and Bogotá) across charts and computer systems shows a fundamenta flaw in data standardization and requiremts.
- Lack of safety: The navigation system allowed the incorrect error handling and system reliability.
- Poor Interface: The confusion and the Pilot's inability to effectively manage the automated system higlight a failure in Human-Computer Interaction(HCI

Question #2:

Software engineering makes aviation safer and more efficient.

- Precision Navigation. Software enables Flight Management System(FMS) that guide planes with precision, optimizing routes and saving fuel.
- Safety Nets: Complex software runs life-saving systems like Enhanced Ground Proximity warning systems (EGPWS), preventin collisions with terrian

• Perdictive Maintaince: Software analyzes data to predict equipment failures, increasing operation reliability and safety.

Question 3:

Future development must address these issue

• Standardize Data: Ensure all navigational codes are unique and consistent across all charts & computer databases to eliminate ambiguity.

• Implement Real-time Validation: The software must have instant, robust checks to immediately reject or warn against illogical or dangerous inputs (like entering code that takes the plane toward a mountain.

• Improve Pilot Feedback: Design the interface to provide clear, umambigaous feedback on the system's status so the pilots always understand what the automation is doing.

-: Part 2:-

Process Model:

- **Waterfall Model:**

  Benefits:

  •) Clear, Sequential Phases simplify Project management.

  •) Detailed documentation supports maintainance.

  Dranback:

  •) Inflexible to changes after phase compdetion.

  •) Late testing delays defect discovery

Handling Late Changes:

•) struggles with late changes; requires revisting earlier Phases, causing delays and cost increases.

- **Waterfall with Prototyping:**

  Benefits:

  - Prototyping clarify requirements via early feedback.
  - Combines waterfall's structure with same flexibility.

Drawback:

- Prototyping adds time to the process
- Risk of scope creep from excessive Changes.

Handling LATE CHANGES:

Better than watefall; Prototypes reduce major changes, but late changes still require rework delaying progress.

- **Prototyping Model:**

BENEFITS:

- Early use feedback ensure allignment with needs.
- Refine unclear requirement iteractively.

DRAWBACK:

- Prototypes may set unrealistic expectations
- Iterative process can extend timelines.

Handling LATE Changes:

Flexible; incorporates changes in new Prototypes, but extensive changes may prolong development.

- **V MODEL:**

Benefits:

•) Pairs each Phase with testing, ensuring validation.

•) Clear milestones enhance project tracking.

DrawBack:

•) Rigid, limiting changes post-Phase.

•) High upfront planning increase costs.

Handling Late Changes:

Poorly handles late changes; requires reworking developement & testing Phase, causing delays.

- **Phased Development:**

Benefits:

•) Deliver partial functionality early for user use.

•) Early phases identifying risks, reducing later issues.

Draw back:

•) Integration of Phases can cause compatibility issues.

•) Requires ongoing management across phases

Handling Late Changes:

Moderately flexible; changes can be added to future phases, but rework of completed phases may delay progress.

• **Spiral Model:**

Benefits:

•) Iterative risk analysis mitigates potentia issues.

•) Supports evolving requirements via prototyping

Drawbacks:

•) Complex, requiring expertise in risk managemen

•) Multiple iterations increase costs.

Handling Late Changes.

Highly adaptable ; incorporates changes in new spirals with risk assessment, through costs may rise.

• **UNiFied Process Modeling:**

Benefits:

•) Iterative proses ensure incremental Progress.

•) use- case-driven design alligns with use needs.

Drawbacks:

•) Complex, needing skilled teams and resources.

•) Multiple Phases can extend timelines.

Handling late Changes:

Handles changes well; new requirements fil into later iteration, but significant changes may delay completion.

- **Extreme Programming:**

Benefits:

•) Frequent realease enable rapid user feedbac

•) Close customer colloboration ensures relevant outcomes.

Drawback:

•) Relies on highly skilled, collaborative teams.

•) Minimal documentation hinder- maintaince.

Handling Late Changes:

Excels at late changes; iterative cycles and continous integration allow quick incorporation with minimal disruption.

- **Scrum:**

Benefits:

•) spirts allow flexible priority adjusment

•) Self-organizing teams boost Productivit

Drawbacks:

•) Risk of scope creep from frequent Changes.

•) Requires strict adherence to Scrum Practices.

Handling LAte Changes:

Highly adoptable; new requirements are added to the backlog and priortized in future sprints, ensuring flexibility.