**1)Static Testing**

Static testing is a type of software testing technique that **examines the code, documentation, and design without executing the program**. It helps identify defects in the early stages of development, reducing costs and effort in later phases.

**Types of static testing**: Review-based testing, static analysis

**Tools used for static analysis:** SonarQube, EsLint,PyLint, PMD, FindBugs.

**2) Dynamic Testing**

Dynamic testing is a software testing technique that involves executing the code to validate the functionality, performance, and behavior of the application. It helps detect runtime issues such as memory leaks, incorrect output, and security vulnerabilities.

**Types of Dynamic Testing**

1. **White Box Testing**
   * Tests internal logic, code structure, and data flow.
2. **Black Box Testing**

* Tests software functionality without looking at the internal code.

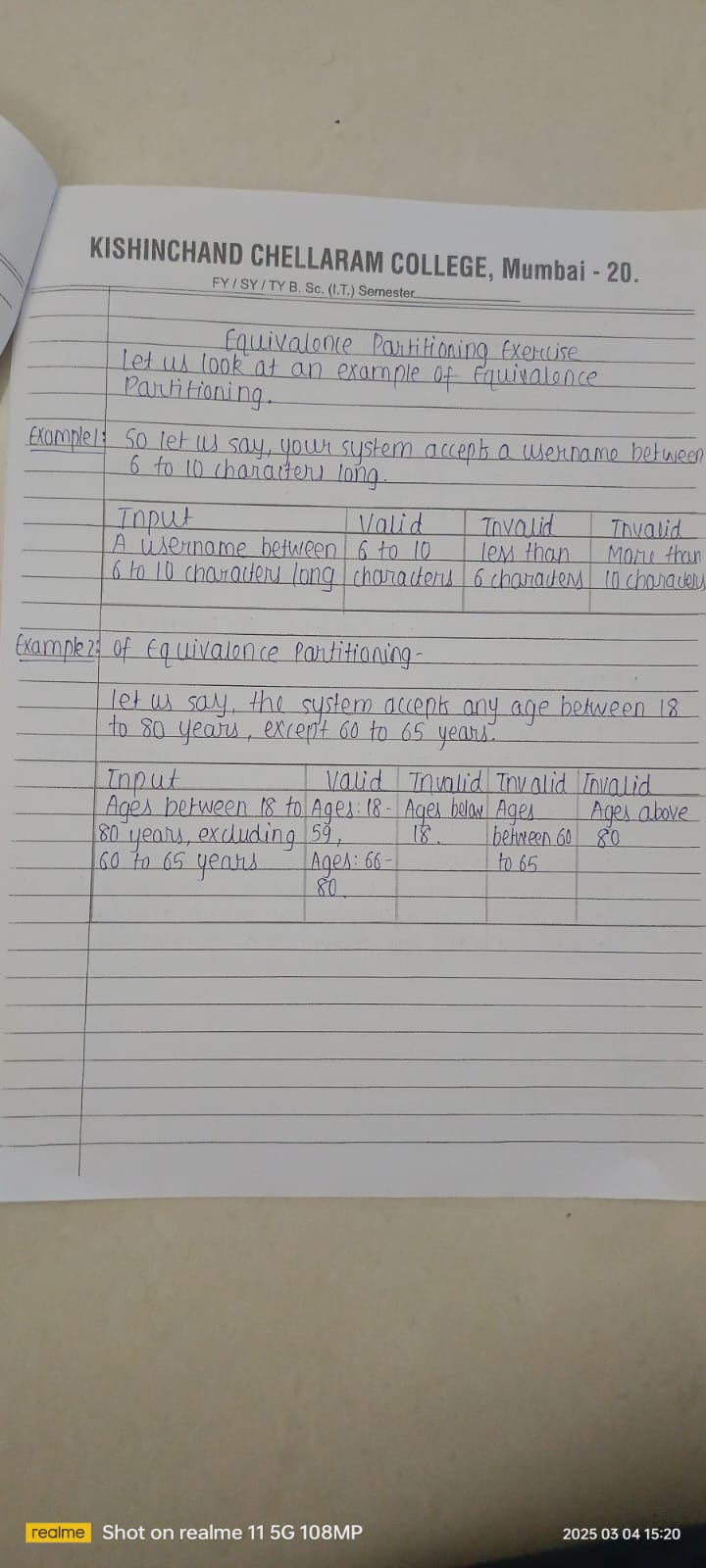
1. Experience based
2. Defect based
3. Dynamic analysis

**Tools used for dynamic analysis:** IProfiler, Valgrid, VisualVM, WhiteShark

**Equivalence Partitioning**

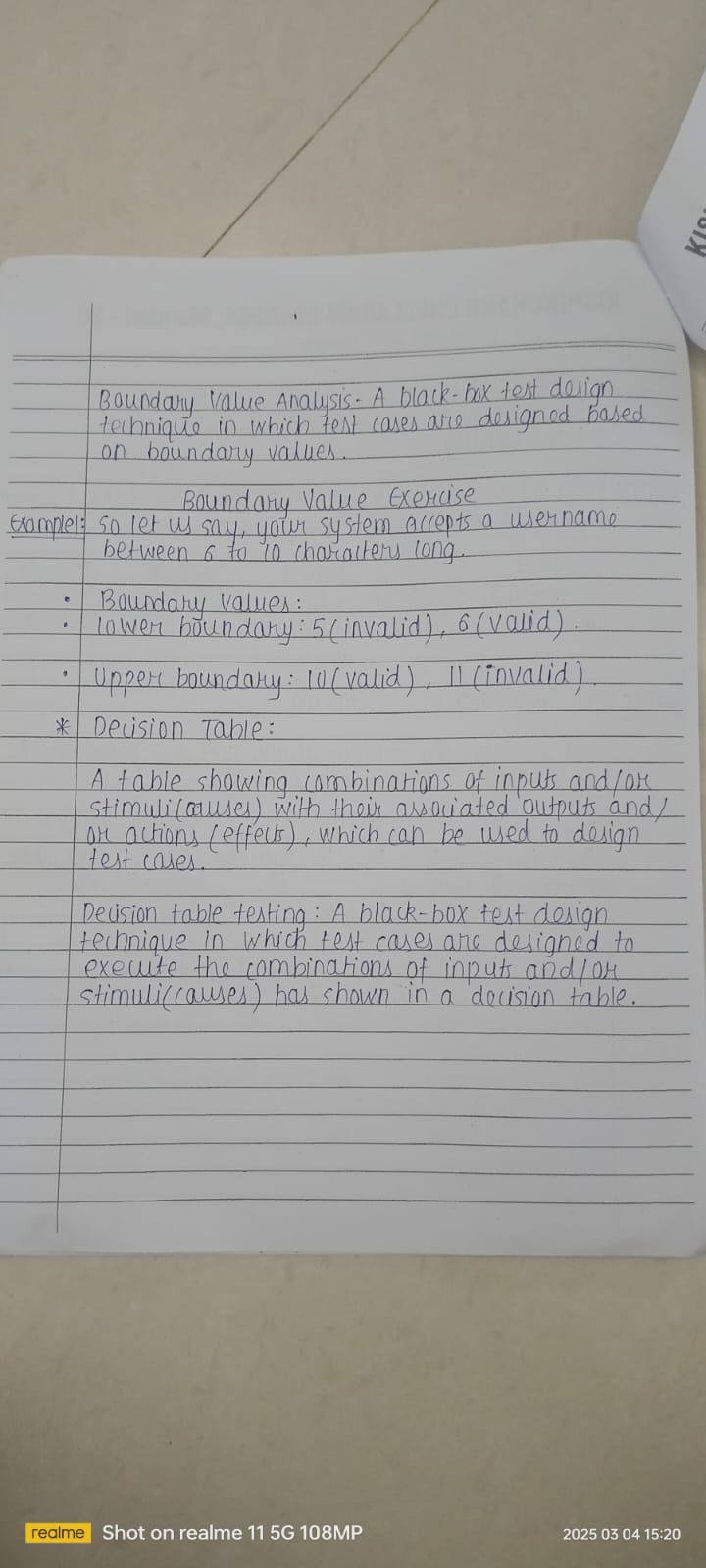
**Equivalence Partitioning (EP)** is a **black-box testing technique** used to divide input data into different classes (partitions) to reduce the number of test cases while maintaining good test coverage.

Instead of testing every possible input, **only one value from each partition is tested**, assuming all values in the partition will behave similarly.



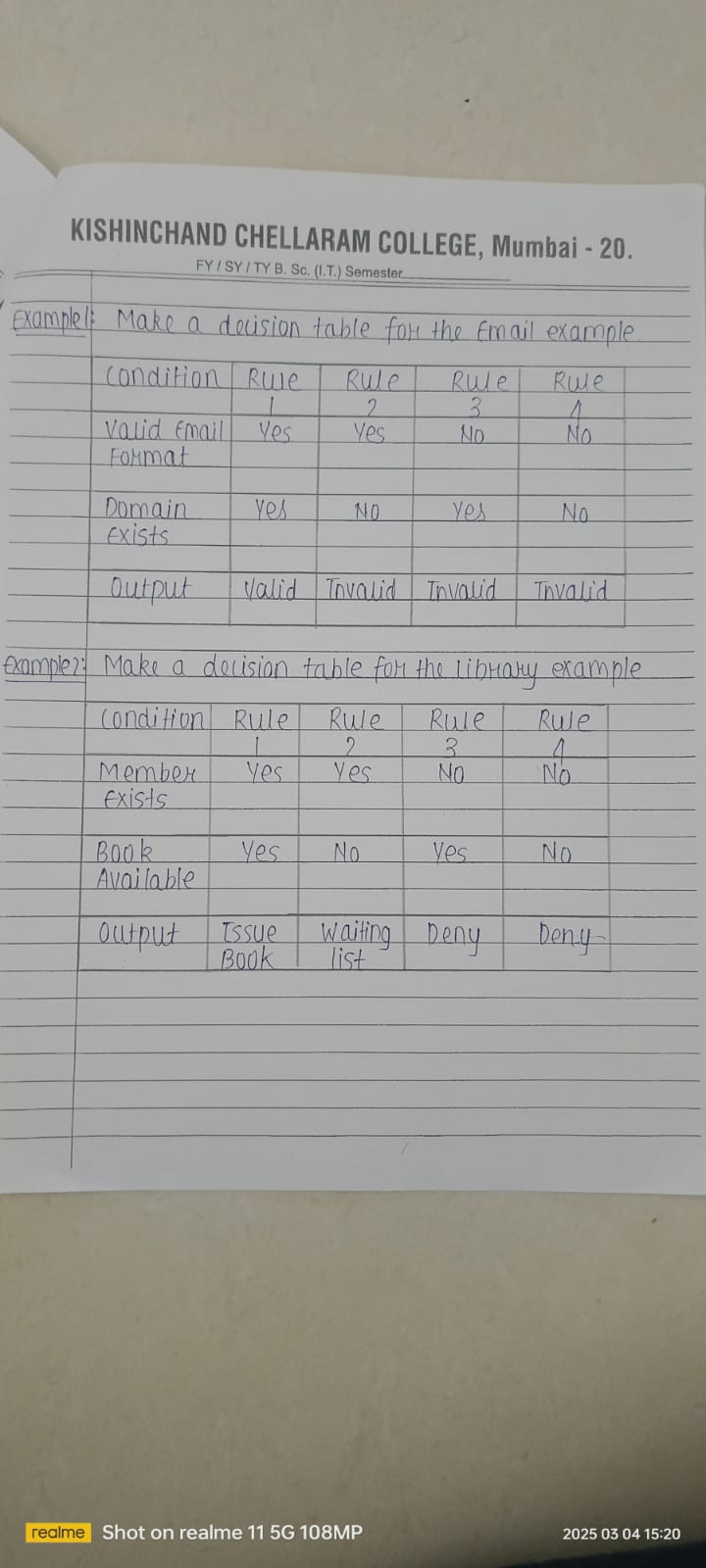
**Boundary Value Analysis (BVA)**

**Boundary Value Analysis (BVA)** is a **black-box testing technique** that focuses on testing the **edges (boundaries) of input ranges** rather than typical values. Since defects often occur at boundaries, this technique helps detect errors efficiently.



**Decision Table in Software Testing**

A **Decision Table** is a **black-box testing technique** used to model complex business logic by representing different conditions and corresponding actions in a tabular format. It helps testers ensure all possible input combinations are considered.



**Defect (or bug or fault or problem):** A flaw in a component or system that cancause the component or system to fail to perform its required function, e.g., anincorrect statement or data definition. A defect, if encountered during execution,may cause a failure of the component or system.

**Incident (or deviation):** Any event occurring that requires investigation.

**Incident Management Process:**

1. Incident Identification: An issue or disruption to service is detected.

2. Incident Logging: Record the incident in an incident management

system.

3. Incident Categorization: Categorize the incident by type (e.g., hardware

failure, network outage).

4. Incident Prioritization: Prioritize based on severity (e.g., P1, P2, P3,

P4).

5. Investigation and Diagnosis: IT team investigates the issue.

6. Incident Resolution: A solution is found and implemented.

7. Incident Closure: The incident is closed and documented.

8. Post-Incident Review (PIR): After the incident is resolved, a post-

incident review is conducted to identify lessons learned.

**Cyclomatic Complexity**

**Cyclomatic Complexity (CC)** is a software metric that measures the complexity of a program by counting the number of independent paths through the code. It helps in identifying areas that may be difficult to test and maintain.

**Formula for Cyclomatic Complexity**

CC=E−N+2P

Where:

* **E** = Number of **edges** (connections between nodes) in the control flow graph
* **N** = Number of **nodes** (decision points, statements) in the control flow graph
* **P** = Number of **connected components** (usually 1 for a single program)