# Developer Project Report Outline

# 1. Requirements Analysis

### What is the aim and scope of the project?

Clearly describe the problem your project addresses and define its overall scope. Indicate whether it is a console-based or interface-based application and state any limitations or assumptions.

---

### To-Do List

List the core features and tasks to be implemented throughout the project, along with their status:

- Task 1
- Task 2
- Task 3

### Who are the ideal users of the system?

Identify the main types of users who will interact with the system. Define their roles, permissions, and expectations from the software.

### User Scenarios in an Ideal System

Describe how each user type is expected to interact with the system. These scenarios should reflect real-life use cases in which the user performs specific tasks using your software.

**Example:** In a **shopping application**, there are two user roles: *User* and *Admin*.

- Users can manage their shopping cart: they can add, remove, and update items.
- Admins can manage product categories by adding or removing them.
- The application is a **console-based system**, not a graphical user interface in this project scope.

- Once the user logs in, a role-based menu is displayed depending on their user ID.
- Users select operations from the menu via numeric input.
- Notifications and error messages are displayed in the console output.

### Functional Requirements and Program Behavior (I/O)

Based on the defined scope, explain how the program will operate:

- What kind of **input** will it take from the user?
- What will be the **output**, and how will it be presented?
- List **functional requirements** such as authentication, data processing, storage, or reporting.

**Example** (continued):

- **Inputs**: user ID, product ID, quantity
- **Outputs**: confirmation messages, error messages, updated cart view
- **Functional requirements include:**
    - User authentication
    - Role-based menu display
    - Item management (CRUD operations)
    - Console notifications

# 2. Software Design

## 2.1. Data Structures

Describe the main data structures used in your project. For each structure, explain **why it was chosen** and **how it supports your requirements**.

**Example:**

- An `ArrayList` is used to store the items in the shopping cart. It allows dynamic resizing and easy access by index.

- A `HashMap<String, User>` is used to store user accounts. It allows fast lookup based on usernames.

**Questions to guide you:**

- What structures did you use (List, Map, Queue, Stack, Set, Tree, etc.)?
- Why did you choose each one?
- Which requirement does it address?

## 2.2. Object-Oriented Design

Explain the **classes and objects** you designed for your system. Define their responsibilities, relationships, and interactions.

**Example classes:**

- `User`: stores username, password, and role
- `Cart`: manages cart items for a user
- `Product`: holds product details (name, price, ID)
- `StoreManager`: handles inventory and user operations

You should also include your **Class Diagram (UML)** to visually present your design.

**Questions to answer:**

- What are the core classes in your system?
- What are their roles and interactions?
- How do they reflect the requirements?

## 2.3. Algorithms and Flow

Present **flowcharts** or **pseudo-code** for your key processes or algorithms. (Focus on major flows )

**Example: Adding a Product to Cart**

[User selects product]
↓
[System checks product availability]
↓
[If available → add to cart]
↓
[Update cart and show confirmation message]

# 3. Implementation (Coding Phase)

## Development Process

- In which order did you build the features?

  **Example:**

  1. Built the user login system
  2. Implemented product catalog
  3. Developed cart functionality

## Challenges Faced

- Which parts were hard to implement?

  **Example:** Handling concurrent access to cart

## Support and Resources

- From whom or where did you get help, how?

  **Example:** Stack Overflow, ChatGPT, Team Leader

# 4. Error Handling & Testing

## Test Cases and Outputs

Describe the test cases you wrote and their results. Include both **successful** and **unsuccessful** cases if possible.

**Example Test Case Table:**

| Test Case Description | Input | Expected Output | Actual Output | Status |
|---|---|---|---|---|
| Add valid product to cart | productID: 101 | “Product added to cart” | Same | Passed |
| Add out-of-stock product | productID: 404 | “Out of stock” | Same | Passed |
| Empty username on login | username: “”, pwd: x | “Username required” | Same | Passed |

### Bugs Encountered and Solutions

List the errors or bugs you faced during development, along with how you resolved them.

**Example:**

**Bug:** Product list did not update after deletion
**Cause:** List view not refreshed
**Solution:** Added `refreshView()` call after deletion

### Support and Resources for Bug Fixes

Mention any external help or collaboration used during bug fixing.

**Examples of sources:**

- Stack Overflow, GPT, Team Leader

# 5. Version Control

### Version Control Tool & Strategy

What tool did you use for version control and how did you apply it? What is your Github account/ cloud folder link for your versions repo?

### How the Project Was Split and Managed

Explain how the project was divided and progressed over time.

Include:

- Feature modules and branches (e.g., login, cart, checkout)

**Example Commit Practices:**

- Commit messages like:
  ```
  feat: add cart logic
  fix: handle empty input on login
  ```
- Daily commits
- Weekly integration via `merge` to main branch

# 6. Discussion / Process Evaluation

## 6.1. Requirements Analysis and Design Process

### Support from the Team Leader

- How did the team leader contribute to this phase?
- On a scale of 1 to 5, how would you rate their support?

### Leadership and Feedback

- Was the team leader responsive and willing to provide feedback during this phase?
- Do you think they acted responsibly in managing and guiding the team?
- What did you learn from the team leader?
- In what ways did you improve yourself during this phase?
- Would you like to provide any **constructive feedback** about your team leader's role?

### Personal Observations & Reflections

- What were your weaknesses at the beginning of this phase? *(Benchmark: Start vs End)*
- What skills or practices did you gain or improve during this process?

## 6.2. Implementation / Coding Phase

**Support from the Team Leader**

- How did the team leader help during development?
- On a scale of 1 to 5, how would you rate their technical or managerial input?

**Leadership and Feedback**

- Was your team leader open to feedback and discussion?
- Did they show responsibility in organizing and directing the implementation?
- What did you learn from their approach to problem-solving or structuring code?
- In what ways did you grow technically or professionally during this phase?
- What feedback would you offer to your team leader based on this experience?

**Personal Observations & Reflections**

- What challenges did you face early on? *(Benchmark: Start vs End)*
- What did you gain throughout the implementation?
- Which practices (e.g., clean coding, debugging, version control) improved the most?

## 6.3. Error Handling & Version Control Process

**Support from the Team Leader**

- How did your team leader support testing and version control?
- Rate their involvement on a scale from 1 to 5.

**Leadership and Feedback**

- Were they consistent in giving feedback on errors or Git usage?
- Did they help resolve merge conflicts or guide testing practices?
- What techniques did you learn from their approach?

- How did you personally grow in areas like debugging or collaborative coding?
- Provide any constructive feedback for their involvement in this phase.

**Personal Observations & Reflections**

- What were your limitations at the beginning? *(Benchmark: Start vs End)*
- What did you learn or improve specifically in debugging and version control?
- Which skills are now more developed thanks to this process?

---

## Attachment

Please, attach weekly meeting notes and other related docs.