Sample of How a Fresher Should Explain Project Documentation

Example: Pay Later Application

---

1. Project Overview

(Keep this short and confident)

What to Explain

- Project name
- Purpose of the application
- Problem it solves

Example:

"The Pay Later Application allows users to purchase products on credit and repay the amount later through EMIs. It helps customers with flexible payments and merchants with better sales."

2. Scope of the Application

Explain boundaries clearly.

Example:

- Handles user registration, credit approval, EMI generation
- Focuses on core Pay Later functionality
- Advanced integrations are planned later

3. User Roles

Explain responsibilities, not screens.

Example:

- User: registration, purchase, EMI repayment
- Admin: approval, monitoring, credit control

4. Application Flow / Working Conditions

(THIS IS WHERE EVALUATORS FOCUS)

How to Explain:

“The application works in a sequential and role-based manner.”

Flow Explanation:

1. User registers and logs in
2. User applies for Pay Later credit
3. Admin reviews and approves credit limit
4. User makes purchases using approved limit
5. System generates EMI schedule
6. User makes repayments
7. Admin tracks repayment status

☑ Explain logic
❌ Don’t explain UI or code lines

5. Tools & Technologies Used

Explain **what + why**, not just names.

Example:

- Frontend: React – for dynamic and reusable UI
- Backend: Spring Boot – to handle business logic securely
- Database: MySQL – structured data storage
- Version Control: GitHub – for tracking code changes
- API Testing: Postman – to validate backend APIs
- IDE: IntelliJ / VS Code

6. How the Application Is Built (Build Flow)

Explain **how things are connected**.

Example Explanation:

“The frontend sends requests to the backend through REST APIs. The backend validates inputs, applies business logic like credit checks and EMI calculations, and stores the data in the database. Responses are then sent back to the frontend.”

☑ Use words like:
request → validation → logic → response

7. Sections Included in the Documentation

Explain structure, not content detail.

Typical Sections:

- Project overview
- Application flow
- User roles
- Tools & technologies
- Functional behavior
- Future enhancements

Example Line:

“The document focuses on how the application works rather than technical code-level details.”

8. Timely Code Updates as per Requirements

Shows professionalism.

How to Explain:

“Whenever requirements were updated, the code was modified accordingly, tested, and pushed to GitHub with proper version control.”

Evaluator likes:

- Incremental updates
- Requirement alignment

9. Pre-Check Before Demo (VERY IMPORTANT)

Explain readiness.

What to Say:

“Before the demo, I ensured the application runs without errors, the latest code is updated, and all core flows are working as per requirements.”

Checklist You Can Mention:

- Application running
- APIs responding
- Data present
- Login credentials ready

10. Clear Understanding & Explanation of Topics

End strong.

Example Closing Line:

“I ensured I clearly understand the application flow, tools used, and business logic so I can explain the project confidently without relying on notes.”

What NOT to Say in Documentation Discussion

- “I just followed instructions”
- “My teammate did this part”
- “I copied the format”
- Over-technical jargon

What Trainers / Evaluators Expect

- Ownership
- Logical explanation
- Requirement awareness
- Professional preparation

Sample Documentation Template

(For Full-Stack Fresher Projects)

1. Project Title

- Name of the application

2. Project Overview

- Purpose of the application
- Business problem it solves

3. Scope of the Application

- What is included
- What is excluded (future scope)

4. User Roles

- User responsibilities
- Admin responsibilities

5. Application Flow / Working Conditions

- Step-by-step flow of how the system works
- Role-based sequence

6. Tools & Technologies Used

- Frontend
- Backend
- Database
- Version control
- Testing tools
- IDE

7. Build & Coding Flow

- Request → Validation → Business Logic → Response
- Frontend–Backend–Database interaction

8. Requirement Handling & Code Updates

- How requirement changes were handled
- Code versioning approach

9. Pre-Demo Readiness

- Application health check
- Latest code status
- Data readiness

Pre-Demo Checklist for Students

(Must Do Before Any Demo or Mock Interview)

☑ Technical Readiness

- Application runs without errors
- Latest code is pushed to GitHub
- APIs tested and working
- Database has sample data

☑ Documentation Readiness

- Requirements document updated
- Application flow clearly understood
- Tools & technologies explanation ready

☑ Explanation Readiness

- Can explain flow without notes
- Clear on business logic
- Ready to answer “why” questions

☑ Professional Readiness

- Login credentials ready
- Browser tabs organized
- Calm and confident mindset