S/W Development Completion Report

Project Name	Photo Studio Management System		
Author	**Team**	**Approver**	Anna Kyselova
Team	Cebanu Dan (@Nik0gda) Nikita Springis (@nikitaspringis21) Manana Bebia (@Mabebi29) Anna Sukhanishvili (@annasulkh)		

Contents

▪ Revision History

Version	Date	Revised contents	Author	Approver
1.0	23.09.2025	Release 1	Team	Anna Kyselova
1.1	26.10.2025	Release 2	Team	Anna Kyselova
1.2	24.11.2025	Release 3	Team	Anna Kyselova
1.3	13.12.2025	Release 4	Team	Anna Kyselova

▪ Terms and Abbreviation

DIP	Dependency Inversion Principle
RAII	Resource Acquisition Is Initialization
UML	Unified Modeling Language
SRP	Single Responsibility Principle
OCP	Open/Closed Principle

1 Project Result

1.1 Development Period and Responsibilities

Table 1-1. Development Term and Responsibilities

Development Period	September 2025 - December 2025	
Development Type	New Development Project	
Responsibility	**Position**	**Name**
Project Lead	Development PL	Nikita Springis (@nikitaspringis21)
S/W Development	S/W Development PL	Cebanu Dan (@Nik0gda)
Implementation & Tests	Developer / Tester	Manana Bebia (@Mabebi29)
Documentation & QA	Documentation / QA	Anna Sukhanishvili (@annasulkh)

1.2 Project Achievements

Contribution Summary:

- Nikita Springis:
  - overall project coordination, integration testing
- Dan Cebanu:
  - lead architecture and implementation of persistence integration,
  - code reviews.
- Manana Bebia:
  - implemented domain entities, managers, and consumables features.
- Anna Sukhanishvili:
  - DLD and project documentation,
  - release notes,
  - verification of validation rules and exception handling.

Table 1-2. Project Achievements

Item	Achievements
Release 1	Created the initial SRS; defined domain model; implemented basic order lifecycle and simple demo flows (Order creation, item addition, basic output)
Release 2	Introduced layered architecture and abstractions (IDisplay), implemented ConsoleDisplay, updated UML class diagrams and refactored code toward SOLID design (clear separation of presentation, business logic and domain)
Release 3	Added robust validation rules and a centralized exception policy (PhotoStudioException and derived types); enforced constructor and manager pre- and post-

	conditions; improved error reporting and behavior for invalid inputs
Release 4	Implemented persistent storage

1.3 Development Effect and Utilization of Results

Development effect

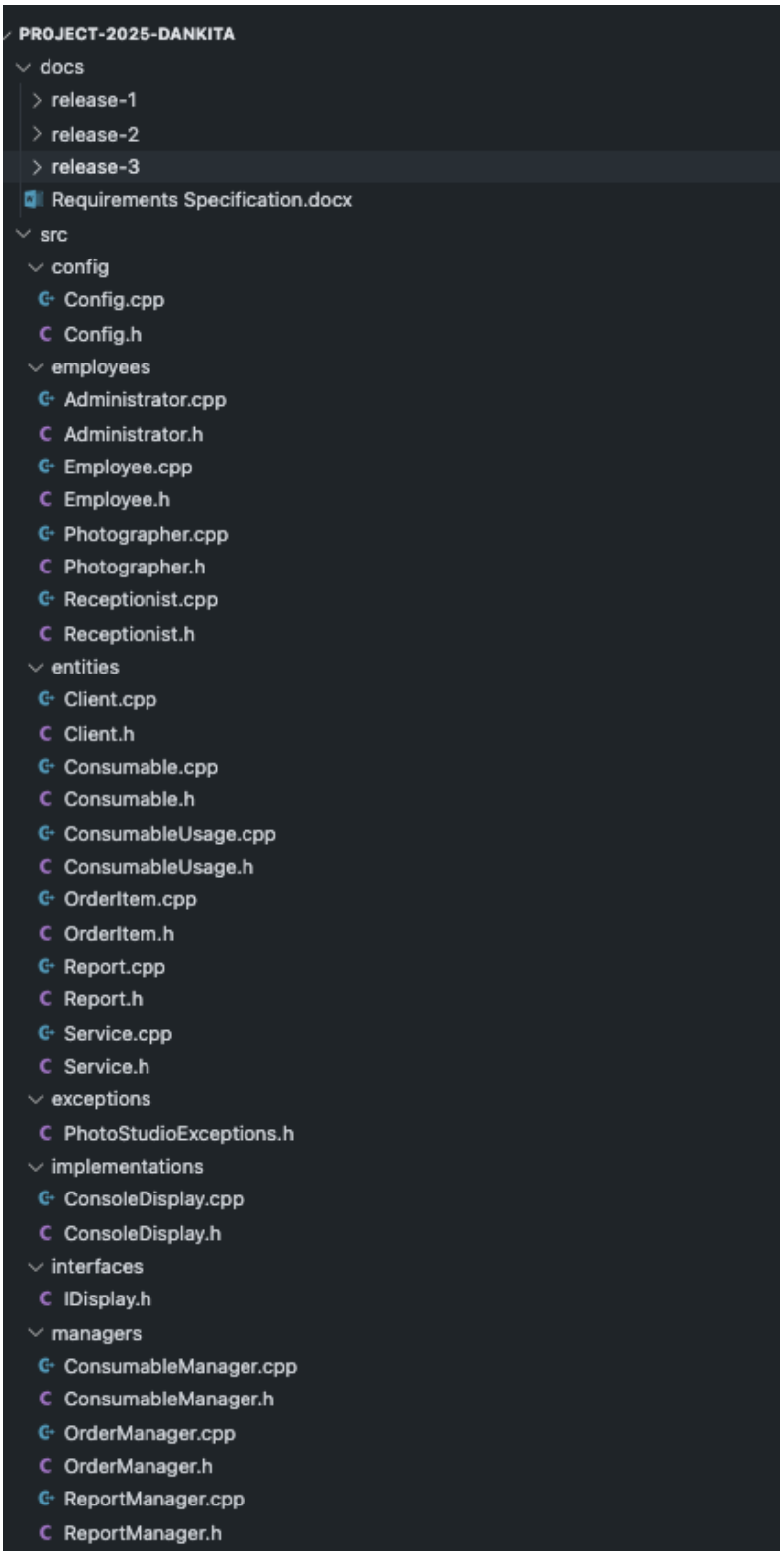
- Final architecture follows a layered design (Presentation, Business Logic, Domain, Infrastructure). Release 4 extends the architecture with a persistence layer that cleanly separates file I/O (FileManager) and in-memory record management (OrderRepository) from domain logic (Order, OrderManager).
- The system demonstrates robust validation and safe reconstruction of runtime objects from persisted data.

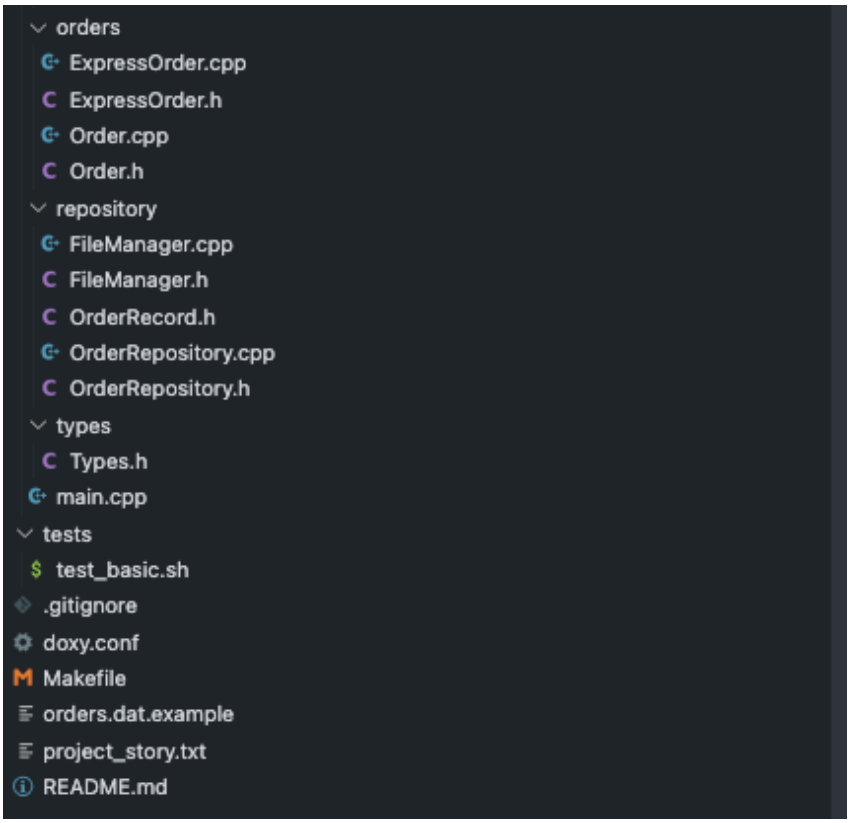
Utilization of results

- The project is suitable as an educational demonstration of:
  - layered architecture and SOLID principles in C++,
  - implementing a simple repository pattern with file persistence,
  - handling data validation and exception-driven error reporting.
- Reuse: the repository and file manager components can be adapted to other small-scale projects that require simple file-backed persistence (orders, logs, small inventories).

2 Development History

2.1 Development S/W Structure





- The size of source code is couple thousands of lines of code suitable for small codebase for educational purposes.

2.2 Development Deliverables

Documentation:

- SRS
- DLD
- UML diagram for class structure, state machine, sequence diagrams

Source code:

- All C++ sources under /src

Persistence sample:

- orders.dat

2.3 Development Environment

Table 2-1. Hardware Environment

Type	Operating System	Major Use	Remarks
Development workstations	Windows, macOS	Development, build and test	Primary dev environment for the team

Table 2-2. Software Environment

Type	Model & Spec.	Major Use	Remarks
Compiler	g++ / clang++ (C++11/14/17 compatible)	Build and compile C++ sources	Project uses a Makefile at repo root to build
Build tool	Make (Makefile)	Build orchestration	Simple Makefile based build
IDE / Editor	Cursor, VS Code	Code editing and debugging	
Shell / Terminal	zsh	Running build and scripts	User environment is zsh on macOS
Version control	git	Source control and history	