University Distrital Francisco Jose de Caldas

Systems Engineering, Faculty of Engineering

PinBerry: A Lightweight and Personalized System for Photo Navigation

Juan Sebastian Herrera Rodriguez

Isabela Chica Becerra

Supervisor: CARLOS ANDRÉS SIERRA

July 11, 2025
Bogota, Colombia

Declaration

I, Isabela Chica and Juan Sebastian Herrera, of the Department of Computer Engineering, Universidad Francisco Jose de Caldas, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

Isabela Chica
Juan Sebastian Herrera
July 11, 2025

Abstract

PinBerry is a minimalist web application inspired by Pinterest, designed to provide users with a streamlined platform for personal image management. The primary objective of the system is to deliver an intuitive and user-centered interface where individuals can register, authenticate, and perform essential actions such as uploading, viewing, deleting, and downloading images. In contrast to conventional social platforms—which often overload users with complex features and algorithmic distractions—PinBerry emphasizes clarity and simplicity, eliminating non-essential components like folders and social feeds to reduce cognitive load and improve usability.

The development process was firmly grounded in object-oriented programming principles, with a strong focus on encapsulation, modularity, and separation of concerns. Core system components, including classes such as User, Photo, Gallery, AuthenticationService, and StorageService, were designed with clear responsibilities, promoting cohesion and reusability. Throughout the project, thoughtful design decisions were made to avoid overengineering. Polymorphism was implemented only in contexts where flexibility and extension were justified, while inheritance and abstraction were applied conservatively to maintain a clean and sustainable architecture.

Additionally, performance and scalability considerations informed the inclusion of structures such as HashMap for efficient user-data mapping and session handling. Although still under development, PinBerry currently demonstrates a robust architectural foundation that balances functionality, maintainability, and user experience. The project not only serves as a practical image management solution but also as a case study in applying modern software engineering principles to real-world applications.

Keywords: Photo management system, Object-oriented programming, Minimalist user interface, Web application development, User experience design.

Contents

List of Tables **v**

1 Introduction **1**
1.1 Background . . . 1
1.2 Problem Statement . . . 1
1.3 Aims and Objectives . . . 2
1.4 Solution Approach . . . 2
1.5 Summary of Contributions and Achievements . . . 3
1.6 Organization of the Report . . . 3

2 Literature Review **4**
2.1 Literature Review: Overcomplexity in Visual Platforms and the Rise of Minimalist Needs . . . 4
2.2 Background: Rethinking Simplicity in User Interface Design . . . 5
2.3 Summary and Conceptual Synthesis . . . 5

3 Scope **7**

4 Assumptions **9**
4.1 Operational Environment Assumptions . . . 9
4.2 Data and Input Assumptions . . . 9
4.3 Design and Interaction Assumptions . . . 10
4.4 Methodological Assumptions . . . 10
4.5 Conclusion . . . 10

5 Limitations **11**
5.1 Technical Limitations . . . 11
5.2 Methodological Limitations . . . 11
5.3 Contextual and Resource Constraints . . . 12
5.4 Conclusion . . . 12

6 Methodology **13**
6.1 Development Methodology . . . 13
6.2 User Stories and acceptance criteria . . . 13
6.3 Implementation Strategy . . . 16
6.4 System Architecture and Design . . . 16
6.5 Ethical Considerations . . . 17
6.6 Tools and Technologies . . . 18
6.7 Use of Inheritance and Polymorphism . . . 18
6.8 Class Relationships and Interaction . . . 18

6.9 Summary 19

7 Results **20**
7.1 Overview of Results 20
7.2 Implemented Functionalities 20
7.3 Structural and Design Outcomes 21
7.4 Example Table: Supported Media Types 21
7.5 User Interface Results 22
7.6 Summary 22

8 Discussion and Analysis **23**
8.1 Evaluation of Results 23
8.2 Significance of the Findings 24
8.3 Limitations 24
8.4 Summary 25

9 Conclusions and Future Work **26**
9.1 Conclusions 26
9.2 Future Work 27

10 Reflection **28**

References **30**

List of Tables

User Story – View Photos 14
User Story – Upload Photos 14
User Story – Delete Photos 14
User Story – Save Photos 15
User Story – Create Folders 15
User Story – Delete Folders 15
User Story – Add Photos to Folders 16
6.1 Main Classes and Their Responsibilities 17

7.1 Supported Media Types and Corresponding Classes 22

List of Abbreviations

OOP	Object Oriented Programing
UI	User Interface
UML	Unified Modeling Language
CRC	Class-Responsibility-Collaborator (cards)
SOLID	Single responsibility, Open/Closed, Liskov substitution, Interface segregation, and Dependency inversion

Chapter 1

Introduction

1.1 Background

PinBerry is a minimalist web application inspired by Pinterest, developed with the goal of offering a lightweight, user-centered solution for image management. Unlike mainstream social platforms that prioritize engagement and feature expansion, PinBerry focuses exclusively on essential photo-related tasks: registering an account, logging in, uploading, viewing, deleting, and downloading images. The application is designed to offer a distraction-free experience that supports personal photo organization without social interaction or algorithmic recommendation systems.

The project was conceived and developed using core object-oriented programming (OOP) principles, with an emphasis on encapsulation, modularity, and class responsibility. Throughout the development process, deliberate design decisions were made to limit complexity and improve maintainability. Non-essential features such as folder hierarchies were removed to streamline the interface and reduce cognitive load. The system architecture includes well-defined classes such as `User`, `Photo`, `Gallery`, `AuthenticationService`, and `StorageService`, each responsible for a distinct set of operations.

The architecture has evolved through iterative improvements, informed by feedback and conceptual clarification. For instance, class diagrams were refined to replace vague relationships like "extends" with more precise verbs like "implements." Additionally, the sequence diagrams were revised after a deeper understanding of the request-response model, and a justified integration of `HashMap` structures was included to optimize user-session and photo associations. These changes reflect a commitment to clarity and robustness in system design. Although implementation is ongoing, the current structure establishes a strong foundation for building a fully functional and sustainable application.

1.2 Problem Statement

Contemporary image-sharing platforms offer a wide range of functionalities, but this breadth often comes at the expense of simplicity. Applications such as Instagram, Pinterest, and Tumblr introduce advanced features, algorithmic feeds, and social engagement mechanisms that can overwhelm users who are simply looking for a space to store and revisit personal memories. The increasing complexity of these platforms creates barriers for users who seek a lightweight alternative that emphasizes core functionality over social interaction.

This project addresses the lack of such alternatives by aiming to develop a simple and focused web application. The challenge lies in designing a tool that enables users to manage personal images efficiently—covering key actions like viewing, saving, and deleting—without the need for social media infrastructure or excessive interface elements. The solution must also reflect sound software engineering practices, particularly through the disciplined application of object-oriented design principles to ensure code clarity, flexibility, and maintainability.

1.3 Aims and Objectives

Aim: To design and implement a user-friendly, object-oriented web application that enables users to manage their personal images through a focused set of features: uploading, viewing, deleting, and downloading photos.

Objectives:

- To define a clean object-oriented architecture where each class has a single responsibility and clear interaction boundaries.
- To build a responsive and intuitive user interface adaptable to both mobile and desktop environments.
- To develop core functionalities such as user account creation, login/logout, photo management, and secure storage.
- To revise and refine UML diagrams—including class, activity, and sequence diagrams—based on deeper understanding of request/response mechanisms and component roles.
- To integrate data structures like `HashMap` to manage session and photo references efficiently and securely.
- To replace excessive bullet-point documentation with concise explanatory paragraphs, improving clarity and narrative cohesion.
- To maintain a clean and extensible codebase, ensuring future features can be added without compromising the architecture.
- To exclude social media-like components (e.g., feeds, likes, comments) in order to promote a healthy, pressure-free environment focused on personal content.

1.4 Solution Approach

The proposed solution is grounded in a layered and object-oriented software architecture, with well-defined separation between the domain model, service layer, and user interface. Each component of the system is built around a central class that encapsulates specific behaviors: the `User` class handles registration and authentication, `Photo` stores and manages image data, `Gallery` controls image display logic, while `AuthenticationService` and `StorageService` handle secure access and persistent storage.

Design tools such as CRC cards, UML class diagrams, and sequence diagrams were employed to model the application and ensure alignment with object-oriented best practices. Improvements from Workshop 4 included refining terminology (e.g., replacing “extend” with “implement”), updating the sequence diagrams to more accurately reflect the flow of control in request/response interactions, and incorporating data structures such as `HashMap` to improve efficiency in mapping user sessions to their data.

The development methodology follows an iterative and incremental approach, starting with the core system functionalities. Testing and evaluation are conducted continuously, ensuring each new feature adheres to the established design standards and maintains architectural integrity.

1.5 Summary of Contributions and Achievements

To date, the PinBerry project has accomplished several key milestones in terms of design, documentation, and architectural planning. These include:

- A well-structured, class-based software architecture grounded in object-oriented principles.
- The definition of core components (e.g., `User`, `Photo`, `Gallery`) with clear roles and responsibilities.
- The creation of detailed CRC cards and UML diagrams to guide development and ensure design traceability.
- A refined user interface mockup designed to accommodate both desktop and mobile usage scenarios.
- A narrative-driven documentation format, replacing long bullet lists with clear paragraphs to enhance readability.
- Conceptual clarification and integration of performance-related data structures such as `HashMap`.
- A strong alignment between software theory (e.g., SOLID principles) and practical implementation strategies.

Although the system is not yet fully implemented, these contributions serve as the foundation for a stable, scalable, and user-friendly application. The design choices made thus far reflect a balance between simplicity, usability, and maintainability.

1.6 Organization of the Report

This report is organized into six chapters. Chapter 1 introduces the project background, problem statement, aims and objectives, the proposed solution, and a summary of contributions. Chapter 2 outlines the functional and non-functional requirements, as well as detailed user stories. Chapter 3 presents the system design, including CRC cards, UML class diagrams, activity diagrams, and sequence diagrams. Chapter 4 describes the implementation of the application's core components, detailing how object-oriented design decisions are translated into actual code. Chapter 5 discusses the theoretical principles behind the design, including a justification for key software engineering choices. Finally, Chapter 6 concludes the report with a summary of the work done and suggestions for future improvements and extensions.

Chapter 2

Literature Review

2.1 Literature Review: Overcomplexity in Visual Platforms and the Rise of Minimalist Needs

Research on image-centric social platforms has consistently demonstrated their capacity to engage users, yet also exposed persistent usability challenges. Instagram's shift to infinite-scroll interfaces and algorithmically curated feeds has amplified user engagement but at the cost of cognitive well-being and interface navigability (**?**). Usability studies of Instagram's mobile design have shown that its dense feature set—stories, reels, shopping tags—can overwhelm non-technical users, decreasing perceived learnability and usability (**? ?**). Similarly, Tumblr's unstructured content architecture and lack of built-in tools for visual organization push users to adopt informal workarounds, which reduce efficiency and scalability in managing visual media (**?**). Furthermore, studies examining the propagation of unmoderated content (e.g., thinspiration imagery) on Tumblr highlight the mental health risks associated with platforms that prioritize virality over personal control (**?**).

Pinterest, while initially perceived as a platform for visual inspiration and idea curation, has also faced criticism for its increasing interface complexity. Heuristic evaluations report that features such as smart recommendations and layered boards introduce significant cognitive overhead, particularly for users interested in straightforward image storage (**?**). User experience studies combining surveys and performance metrics indicate that Pinterest's perceived complexity reduces satisfaction among non-professional users who prefer simplicity over discovery-driven interaction (**?**). Health-communication research further supports these findings, showing that complex visual layouts can impair information retrieval even in platforms with positive emotional framing (**?**).

Moreover, broader critiques of social platforms such as Twitter and Spotify have identified patterns of interface manipulation—so-called "dark patterns"—which subtly guide user behavior through visual prioritization, notifications, and embedded engagement loops (**?**). These manipulations can result in decreased user autonomy and familiarity, contributing to mental fatigue and dissatisfaction.

Collectively, these findings expose a clear gap in the ecosystem: while modern platforms optimize for engagement, they often sacrifice simplicity, intentionality, and user control. In response, PinBerry is positioned as a minimalist, user-centered alternative that reclaims basic interaction—uploading, viewing, deleting, and downloading photos—within an uncluttered interface devoid of algorithmic pressure.

2.2 Background: Rethinking Simplicity in User Interface Design

The growing complexity of digital platforms underscores the need to revisit foundational theories of interface design. At the core of this reconsideration lies the concept of *cognitive load*, defined as the mental effort required to perform a task. Studies have shown that excessive interface elements—notifications, icons, nested menus—increase extraneous cognitive load and hinder users from accomplishing basic goals efficiently (**?**). In the context of photo-based platforms, this means that users may expend more effort navigating features than engaging meaningfully with their content.

Perceptual Load Theory, as proposed by Lavie, further refines this idea by distinguishing between tasks with high and low perceptual complexity (**?**). High-load tasks deplete attentional resources, limiting users' ability to process irrelevant stimuli. Conversely, low-load interfaces allow distractions to seep in, highlighting the need to manage complexity thoughtfully—not simply reduce it. In response, a well-calibrated interface should strike a balance: offering just enough structure to guide users, but not so much as to overwhelm them.

Emotional design also plays a pivotal role in shaping user experience. According to Donald Norman, systems should elicit positive emotional responses at three levels: visceral (appearance), behavioral (usability), and reflective (personal meaning) (**?**). This triadic framework suggests that effective tools are not only functionally usable, but also emotionally rewarding—particularly relevant in applications dealing with personal memories such as photographs.

The Media Naturalness Theory complements this view by asserting that interfaces should emulate the cues and rhythms of face-to-face communication to reduce user cognitive strain (**?**). Although applied more often to communication platforms, this theory supports the idea that digital systems should behave in familiar, predictable ways.

PinBerry adopts these principles in both its system and user interface design. Through clean layouts, restrained visual elements, and functionally constrained interactions, the platform minimizes cognitive and emotional friction. As part of Workshop 4, design choices were further refined: bullet-heavy documentation was replaced by narrative summaries to promote clarity; folder-based image organization was removed; and key components such as sequence diagrams were restructured after deeper understanding of request-response models. The deliberate integration of structures like `HashMap` reflects a focus on efficiency and scalability while maintaining architectural simplicity.

2.3 Summary and Conceptual Synthesis

The academic literature on social image-sharing platforms reveals a fundamental contradiction: systems designed to engage users often do so by sacrificing simplicity, transparency, and mental well-being. Instagram, Tumblr, and Pinterest—though powerful and feature-rich—are frequently cited for their usability limitations, especially by users seeking basic organizational tools. These findings align with the design philosophy behind PinBerry: to reintroduce focus, calm, and clarity into photo management applications.

Grounded in established theories such as Cognitive Load Theory, Perceptual Load Theory, Affective Design, and Media Naturalness Theory, PinBerry prioritizes user needs over platform metrics. It discards social networking features to reduce performance anxiety, embraces minimalist interaction patterns, and favors intuitive navigation. Technical improvements implemented during Workshop 4—such as the restructured sequence diagrams and modular service architecture—reflect this same commitment to balance clarity and functionality.

Ultimately, PinBerry is not only a technical implementation, but a conceptual counterpoint to the increasingly complex design paradigms of mainstream platforms. It addresses a critical gap in digital experiences: the need for a peaceful, reliable, and simple tool for personal image interaction. In doing so, it offers a vision of software that is both user-centered and technically disciplined.

Chapter 3

Scope

The scope of this project is defined by the intentional delimitation of both its functional objectives and technical boundaries, with the aim of producing a focused, maintainable, and ethically grounded image management application. Rather than pursuing a feature-rich or socially networked platform, this study concentrates on core functionalities that enable personal control over digital media in a minimalistic environment.

This project includes the design, implementation, and evaluation of a desktop-based system that allows users to upload, view, delete, and organize multimedia files—specifically photos, videos, and GIFs—through a streamlined graphical user interface. The system is designed using object-oriented programming principles in Java, emphasizing modularity, extensibility, and adherence to software design best practices, such as the SOLID principles.

The boundaries of this research were set to ensure that the project remained technically feasible, methodologically coherent, and aligned with its conceptual vision. The following components were explicitly included within the project's scope:

- **System Architecture:** Design and implementation of a modular class-based architecture, featuring key components such as `Usuario`, `Foto`, `Gif`, `Video`, `Folder`, and `App`, all organized under object-oriented principles.
- **Interface Development:** Creation of a desktop-based GUI using Java Swing, with interface layouts informed by mockups developed in Canva, emphasizing simplicity, cognitive clarity, and visual accessibility.
- **Core Functionalities:** Development of essential user actions, including multimedia upload, local storage management, folder association, deletion, and download.
- **Polymorphic Media Handling:** Introduction of a `Multimedia` interface to abstract and generalize operations across different media types (images, GIFs, videos), enabling flexible extension of the media domain.
- **Persistence Mechanisms:** Implementation of file-based storage using text files to maintain user data and folder structures across sessions.
- **Documentation and Modeling:** Use of UML class diagrams, CRC cards, and sequence diagrams to document system logic and ensure traceability from design to implementation.

However, to maintain a focused and achievable development scope within time and resource constraints, several features and dimensions were explicitly excluded from this version of the system. These exclusions do not indicate a lack of importance, but rather reflect deliberate design decisions to prioritize simplicity and proof-of-concept clarity. The following components are considered outside the scope of this project:

- **User Authentication System:** The current implementation assumes a static user identity and does not include multi-user support, credential management, or session control.
- **Database Integration:** Data persistence is handled through plain text files, and no relational or non-relational database systems (e.g., SQLite, PostgreSQL) have been implemented.
- **Advanced Media Rendering:** While media files are uploaded and stored correctly, playback or preview functionalities for GIFs and videos have not been included.
- **Mobile or Web Deployment:** The application is designed solely for desktop environments and does not support mobile platforms or web-based interaction.
- **Cloud Storage or Synchronization:** All operations are performed locally; there is no integration with cloud APIs or online backup solutions.
- **Usability Testing and Analytics:** No empirical user testing has been conducted to evaluate usability, and no telemetry or behavioral data is collected.

In summary, the scope of the PinBerry project is carefully bounded to emphasize simplicity, ethical user experience, and sound software architecture. The system serves as a proof-of-concept for a minimalist media platform, with a clearly defined path for future expansion. By deliberately constraining complexity, the project achieves depth in design and clarity in purpose, laying a strong foundation for subsequent iterations and improvements.

Chapter 4

Assumptions

The development of the PinBerry application was guided by a series of methodological and technical assumptions that served to constrain the design space, simplify implementation, and focus on the project's core objectives. These assumptions were necessary to ensure feasibility within the available timeframe, tools, and scope of the research. By clearly stating them, this section provides transparency about the system's operational context and highlights the boundaries within which the findings and conclusions remain valid.

4.1 Operational Environment Assumptions

- **Single-user environment:** It was assumed that the application would be used by a single user operating on a personal device. As such, multi-user support, account creation, authentication mechanisms, and session management were deliberately excluded from the initial implementation.
- **Local execution and storage:** The system was designed under the assumption that it would run locally, without dependence on network connectivity. All media files are expected to reside on the user's device, and persistence is handled through local text files, not remote databases or cloud infrastructure.
- **Sufficient computational resources:** It was assumed that the user's system possesses adequate memory, storage, and processing capabilities to support the application's functionality, including media loading and graphical interface rendering, without significant performance degradation.

4.2 Data and Input Assumptions

- **Valid user input:** The application assumes that users will provide valid input, particularly in terms of supported file types and naming conventions. While some input validation is implemented, comprehensive input sanitization and exception handling for edge cases were not prioritized during development.
- **File-type inference from extensions:** Media classification is based solely on file extensions (e.g., `.jpg`, `.gif`, `.mp4`), and it is assumed that these accurately reflect the file's format. No deep inspection or content-based validation is performed to confirm file integrity or type consistency.

- **Stable directory structure:** The application assumes that the required directories and text files either already exist or can be created and accessed reliably during runtime. It further assumes that these files will remain uncorrupted and free from concurrent modifications by external processes.

4.3 Design and Interaction Assumptions

- **Independent and non-collaborative use:** It is assumed that the user interacts with the application in isolation. Collaborative features such as content sharing, commenting, or concurrent sessions were not considered within the scope of this study.
- **Desktop usage model:** The system is designed for desktop environments, with user interaction expected through traditional input devices (mouse and keyboard). No assumptions were made about touch interfaces, mobile responsiveness, or accessibility support.
- **No adversarial behavior:** The user is presumed to act in good faith, without attempting to exploit, disrupt, or reverse-engineer the application. Consequently, no security mechanisms were implemented to protect against malicious use or intentional misuse.

4.4 Methodological Assumptions

- **Focus on functionality over optimization:** It was assumed that demonstrating correct functionality was a higher priority than achieving maximum performance or scalability. Therefore, performance testing and efficiency tuning were not part of the development cycle.
- **Conceptual validation over empirical evaluation:** Due to time and scope constraints, the evaluation of the system was primarily conceptual and based on internal testing. It is assumed that the implemented features adequately represent the design goals, even though no formal usability testing or external validation was conducted.

4.5 Conclusion

These assumptions formed the foundation upon which the project's technical and methodological choices were made. While they facilitated the realization of a functional prototype within the defined limits of the research, they also introduce constraints that may impact the generalizability and extensibility of the results. Future iterations of the system should revisit these assumptions—particularly those regarding user management, input validation, and system scalability—to enable broader deployment and more robust application behavior.

Chapter 5

Limitations

As with any software engineering project conducted under academic constraints, the development of PinBerry presents a number of limitations that must be acknowledged to ensure a fair and accurate interpretation of its results and implications. These limitations are categorized into technical, methodological, and contextual dimensions, each of which impacted the scope, functionality, and generalizability of the system. Recognizing these constraints is essential to framing the project's contributions realistically and to guiding future improvements.

5.1 Technical Limitations

One of the primary technical limitations lies in the system's user model. The current implementation supports only a single, static user, without authentication, identity management, or session persistence. This restricts its deployment to controlled or individual environments and excludes multi-user scenarios, which are common in modern media applications.

Data persistence was achieved using plain text files, which, while suitable for prototyping, impose considerable restrictions on scalability, data integrity, and concurrent access. The lack of structured query capabilities or indexing mechanisms further limits performance in larger datasets, making the system inefficient for high-volume media collections.

Additionally, while polymorphism was correctly implemented to handle multiple media types—photos, GIFs, and videos—the application does not currently support media playback or previews. The system is capable of storing and organizing dynamic content, but it does not provide an interface to render or interact with it beyond metadata access. This reduces the practical utility of handling non-static media files.

From a user interface perspective, the application relies on Java Swing, which—although sufficient for basic desktop interaction—offers limited responsiveness, lacks modern styling flexibility, and does not support adaptive design principles. The interface is not mobile-responsive, nor does it incorporate accessibility features, thereby limiting the diversity of its user base.

5.2 Methodological Limitations

No formal usability testing was conducted with real users. While the interface design was guided by cognitive and affective design theories, the absence of empirical evaluation leaves unanswered questions about the actual user experience. Similarly, no automated testing frameworks or performance benchmarks were implemented to measure stability, efficiency, or error tolerance under varied conditions.

Input validation and error handling are basic, and the application's robustness against invalid operations (e.g., uploading corrupt files, invalid paths, or unsupported formats) was not systematically tested. This creates uncertainty regarding the system's behavior in edge cases or under misuse.

Furthermore, the lack of integration with versioned deployment environments, automated build pipelines, or software instrumentation tools reduces the ability to observe runtime behavior or trace faults effectively in a production context.

5.3 Contextual and Resource Constraints

Given the academic nature of the project, development was limited by time and resources. The system was implemented and tested by a small team of two, which necessarily restricted the complexity and scale of features that could be reasonably developed and integrated. Decisions regarding simplification—such as omitting cloud storage, real-time collaboration, or metadata editing—were made to ensure that a stable and functional prototype could be delivered within the established deadlines.

Additionally, the project was carried out without access to domain experts, professional UX designers, or a representative user group. As a result, the application's design reflects theoretical considerations and developer assumptions rather than validated user feedback.

5.4 Conclusion

These limitations do not detract from the value of the work, but they do delimit the scope of its contributions. PinBerry fulfills its intended role as a minimalist photo management tool grounded in object-oriented principles and ethical design, but its generalization to broader use cases requires additional development. Future work should address these technical and methodological gaps through empirical validation, architectural refinement, and enhanced feature sets to support broader deployment and long-term sustainability.

Chapter 6

Methodology

6.1 Development Methodology

PinBerry was developed using an iterative and incremental methodology, inspired by agile principles. The goal was to modularize the system into core functional components, allowing for continuous design, implementation, and evaluation cycles. This approach facilitated ongoing adjustments based on both functional needs and non-functional concerns such as performance, maintainability, and scalability.

Object-Oriented Programming (OOP) principles played a central role to promote clarity, flexibility, and code reuse. The system was structured with multiple classes, each assigned a specific responsibility aligned with the Single Responsibility Principle (SRP). Early design favored composition over inheritance to maintain simplicity and avoid unnecessary complexity, ensuring that the architecture remained clear and manageable as the system evolved.

Furthermore, the design adhered to other SOLID principles to ensure a robust, scalable, and maintainable architecture:

- **Open/Closed Principle (OCP):** The system was designed to be open for extension but closed for modification, primarily through the implementation of interfaces such as Multimedia. This allowed adding new media types without altering existing code.
- **Liskov Substitution Principle (LSP):** Media classes like Photo, Video, and Gif implement a common interface, allowing them to be treated interchangeably, enhancing modularity and interoperability.
- **Interface Segregation Principle (ISP):** Interfaces were kept focused and minimal, ensuring classes depended only on the methods relevant to their responsibilities.
- **Dependency Inversion Principle (DIP):** High-level modules depended on abstractions rather than concrete classes, fostering loose coupling and facilitating testing and future scalability.

This strategic application of SOLID principles ensured that PinBerry's architecture remained flexible, maintainable, and aligned with best practices in object-oriented design.

6.2 User Stories and acceptance criteria

To guide the development of PinBerry, we created a set of user stories. Each story outlines a specific user need, along with acceptance criteria and task estimates to support prioritization and implementation planning.

View Photos

User Story – View Photos

As a User,
I want to view the available images on the platform,
So that I can explore interesting visual content.

Acceptance Criteria:

- Given that I am on the homepage
- When I navigate through the gallery
- Then I should be able to see all the available images.

Priority: Medium **Estimate:** 16–20 hours

Upload Photos

User Story – Upload Photos

As a User,
I want to upload images to the platform,
So that I can share my memories and experiences.

Acceptance Criteria:

- Given that I am on the upload page
- When I select an image and confirm the upload
- Then the image should be stored on the platform.

Priority: High **Estimate:** 20–24 hours

Delete Photos

User Story – Delete Photos

As a User,
I want to delete images that I have uploaded,
So that I can remove content I no longer need.

Acceptance Criteria:

- Given that I am on my gallery page
- When I select an image and press the delete button
- Then the image should be permanently removed from the app.

Priority: Medium **Estimate:** 16–20 hours

Save Photos

User Story – Save Photos

As a User,
I want to save images by downloading them,
So that I can have the picture on my device.

Acceptance Criteria:

- Given that I am viewing an image
- When I choose to download
- Then the image should be saved on my device.

Priority: Medium **Estimate:** 16–20 hours

Create Folders

User Story – Create Folders

As a User,
I want to create personalized folders (MiniFolders),
So that I can organize my uploaded photos efficiently.

Acceptance Criteria:

- Given that I am logged into the application
- When I enter a valid folder name and confirm
- Then the system should create a MiniFolder associated with my account

Priority: High **Estimate:** 12–16 hours

Delete Folders

User Story – Delete Folders

As a User,
I want to delete custom folders that I no longer need,
So that I can keep my photo organization clean and relevant.

Acceptance Criteria:

- Given that I am viewing my list of MiniFolders
- When I select a folder and confirm deletion
- Then the folder and its contents should be permanently removed, if it belongs to me

Priority: Medium **Estimate:** 12–16 hours

Add Photos to Folders

User Story – Add Photos to Folders

As a User,
I want to upload photos into specific folders,
So that I can organize my visual content according to themes or categories.

Acceptance Criteria:

- Given that I am uploading a photo
- When I select a folder (UserFolder, MiniFolder, etc.)
- Then the photo should be stored in the selected folder
- And it should also appear in the GeneralFolder

Priority: High **Estimate:** 14–18 hours

6.3 Implementation Strategy

The implementation was divided into the following phases:

1. Class model design
   The responsibilities, attributes and methods of each class were defined, applying the principles of encapsulation and separation of responsibilities.
2. Interface prototyping
   Mockups were created for the mobile versions using Canva, focusing on a clear, visually pleasing experience, free of distracting elements.
3. Coding of essential functionalities The following functions were prioritized for implementation:
   - User registration and login.
   - Image upload.
   - Gallery viewing.
   - Deleting and downloading images.

6.4 System Architecture and Design

The system was structured around a set of well-defined classes that reflect core responsibilities within the PinBerry application:

- **Multimedia (interface):** Defines common behaviors for all media types, such as retrieving name, route, size, and creation date.
- **Photo, Gif, and Video:** Concrete classes that implement the `Multimedia` interface, each representing a specific type of content.

- **Folder:** Manages collections of multimedia objects. Three types of folders are supported: general folders (shared content), user folders (personal uploads), and mini folders (custom user collections).
- **User:** Handles user-specific actions such as uploading, downloading, and organizing media within their own folders.
- **App:** Orchestrates the overall application flow, manages the GUI (built with Java Swing), and connects user actions with backend logic.

The design was modeled using object-oriented tools such as UML class diagrams, CRC cards, and sequence diagrams. These resources ensured traceability between the user stories, class responsibilities, and system interactions, promoting clarity and maintainability throughout the development process.

Table 6.1: Main Classes and Their Responsibilities

Class	**Responsibility**	**Key Methods**	**Collaborators**
`App`	Controls the main flow of the application and manages interactions between users, folders, and media. Handles GUI via Java Swing.	`uploadMultimedia()`, `deleteMultimedia()`, `downloadMultimedia()`, `createUserFolder`, `createMiniFolder()`, `createGeneralFolder()`	`User`, `Folder`, `Multimedia`
`User`	Represents the user and provides functionalities to manage media content and authentication.	`addMultimedia()`, `eliminateMultimedia()`, `saveMultimedia()`, `login()`, `registrate()`	`App`, `Folder`, `Multimedia`
`Folder`	Organizes media content into structured categories: general, user, and mini folders.	`generalFolder()`, `miniFolder()`, `getMultimediaDevice()`, `getMultimediaGeneral()`	`User`, `App`, `Multimedia`
`Multimedia` (interface)	Defines common media operations shared by all content types.	`getRoute()`, `getName()`, `getSize()`, `getDate()`	`Photo`, `Gif`, `Video`, `Folder`
`Photo`, `Gif`, `Video`	Represent specific types of multimedia content, each implementing the `Multimedia` interface.	`getRoute()`, `getName()`, `getSize()`, `getDate()`	`Multimedia`, `Folder`

6.5 Ethical Considerations

This project does not collect real personal data nor does it involve human users in its development phase. However, the importance of:

- User privacy and mental health: Any functionality related to social interaction or public exposure of images, such as comments, “likes” or followers, is avoided to reduce digital social pressure.
- Integrity of the development: All the code and design of the system is original, without plagiarism, and respecting the principles of academic and professional ethics.

- Product impact: The application seeks to offer a simpler and healthier alternative to current visual social networks, minimizing distractions and risks associated with self-image on social platforms.

6.6 Tools and Technologies

The main tools used in the development of the project are:

- Programming language: Java
- Modeling: Canva, Lucidchart
- Version control: Git and GitHub
- Documentation: Overleaf (LaTeX)

6.7 Use of Inheritance and Polymorphism

During the design of the PinBerry application, the use of inheritance and polymorphism was initially approached with caution in order to maintain structural simplicity and avoid premature abstraction. However, as the system evolved, these object-oriented features were implemented selectively to support flexibility and reusability.

A key application of inheritance and polymorphism was the introduction of the `Multimedia` interface, which defines a set of common behaviors shared by different types of media content. The classes `Photo`, `Gif`, and `Video` all implement this interface, allowing them to be treated polymorphically within the application. This design enables consistent access to metadata such as name, route, size, and creation date, regardless of the specific media type.

By using interface-based inheritance, the system adheres to the principles of modularity and low coupling. This implementation of polymorphism simplifies future extension of the system; new media types can be added with minimal changes to existing code. Furthermore, this design aligns with the Open–Closed and Liskov Substitution principles from SOLID, ensuring that the system remains both scalable and maintainable.

The use of inheritance and polymorphism in this controlled and purposeful manner enhanced the clarity and extensibility of the system without introducing unnecessary complexity.

6.8 Class Relationships and Interaction

The relationships between the core classes in PinBerry follow object-oriented design associations such as association, aggregation, and composition, depending on the degree of dependency and ownership involved.

- **Association:** The `App` class is associated with both `User` and `Folder`, coordinating interactions between them. This is a bidirectional but loose relationship, as `App` connects components without owning their internal state.
- **Aggregation:** The `Folder` class aggregates a collection of `Multimedia` objects, including `Photo`, `Gif`, and `Video`. While folders group and organize these elements, the media objects themselves can exist independently from any specific folder.

- **Composition:** The `User` class has a composition relationship with its personal folders (`UserFolder` and `MiniFolder`), as these are created and removed in conjunction with the user. Additionally, when a user uploads media, the associated operations are tightly bound to that user's lifecycle.

This classification of relationships allows the system to maintain a modular and maintainable structure, where each component has clearly defined responsibilities and interactions.

6.9 Summary

This chapter presented the methodology followed during the development of the PinBerry application. An iterative approach was adopted, focusing on the implementation of essential image-related features while applying object-oriented programming principles such as encapsulation and class responsibility. The system design was structured around five main classes, each with clearly defined roles to support user interaction and photo management. The design decisions were documented through diagrams and CRC cards, and ethical considerations were taken into account to promote a healthier user experience by avoiding social network features.

Chapter 7

Results

7.1 Overview of Results

The implementation of PinBerry has resulted in the development of a functional prototype that demonstrates the feasibility of a lightweight, user-oriented photo management system built using object-oriented design principles. The platform fulfills its primary objectives by enabling users to upload, view, organize, and download multimedia content through an interface that avoids cognitive overload and social pressure.

The system successfully separates concerns across multiple architectural layers, and models responsibilities through clearly defined classes such as `Usuario`, `Foto`, `Gif`, `Video`, `Folder`, and `App`. Furthermore, the design leverages polymorphism via the `Multimedia` interface, allowing uniform handling of different media types while ensuring the extensibility of the system for future enhancements.

The main contributions of this phase include:

- The complete implementation of a Swing-based GUI that adheres to usability and simplicity standards.
- A domain model grounded in SOLID and object-oriented principles, structured for long-term maintainability.
- Support for multiple file types and structured user-folder relationships.
- Integration of persistent data storage through text files that simulate a basic file system.
- A system architecture that maintains a clear separation between user interaction, domain logic, and data persistence.

These results confirm that it is possible to design a functional image management system that resists the complexity and psychological pressures of current social media platforms, while still offering essential features to meet user needs.

7.2 Implemented Functionalities

The following core functionalities were developed and tested in the current version of PinBerry:

1. **User Account Simulation:** A static user model (“User”) is used to simulate session-based behavior. All multimedia uploads and folder creations are attributed to this default user, enabling focused testing of features without login infrastructure at this stage.

2. **Media Upload and Representation:** Users can upload files in different formats (`.jpg`, `.gif`, `.mp4`, etc.). Upon upload, each file is instantiated as an object of the respective class (`Foto`, `Gif`, or `Video`) implementing the `Multimedia` interface. Each object stores metadata such as file path, name, type, and timestamp.

3. **Gallery Navigation and Infinite Scroll:** The application displays uploaded media in a scrollable interface simulating an infinite gallery. The GUI dynamically loads content as the user interacts with the system, enhancing usability and performance without overwhelming memory.

4. **Folder Creation and Association:** Users can create folders and associate existing media items from the general gallery to personalized folders. The media items are not moved but referenced, preserving data integrity and allowing reuse.

5. **Deletion and Download:** Users can delete uploaded content or download it locally. The deletion feature supports logical deletion by marking the media as inactive while maintaining its entry in storage.

6. **Persistent Storage:** All user interactions that involve uploading, deleting, or organizing media are persistently stored in plain text files (`GeneralFolder.txt`, `User.txt`, etc.), which are read and written by the system during execution.

This implementation validates the architectural choices made during the design phase and demonstrates the applicability of the system to address the needs identified in the problem statement.

7.3 Structural and Design Outcomes

The structural integrity of the system was validated through the use of CRC cards, UML class diagrams, and sequence diagrams. These design tools served not only to plan the system architecture but also to validate that each class and method fulfilled its responsibility without redundancy or ambiguity.

The inclusion of the `Multimedia` interface significantly improved design flexibility. By allowing media types to be treated generically, the system adheres to the Open/Closed Principle, making it easier to introduce new media types in the future without modifying existing logic. This polymorphic treatment is central to the maintainability and extensibility of the platform.

Furthermore, the introduction of a `HashMap` data structure for mapping users to folders or media items improved runtime access efficiency. This approach ensures rapid lookup and consistent behavior, especially as the number of media objects increases.

7.4 Example Table: Supported Media Types

Table 7.1 summarizes the types of multimedia content currently supported by PinBerry and the corresponding implementation classes responsible for their processing.

Table 7.1: Supported Media Types and Corresponding Classes

Media Type	**File Extensions**	**Handled by Class**
Photo	.jpg, .jpeg, .png	`Foto`
GIF	.gif	`Gif`
Video	.mp4, .avi	`Video`

7.5 User Interface Results

A mobile-first interface was designed using Canva mockups and implemented using Java Swing. The interface avoids unnecessary graphical elements, reducing perceptual load and increasing usability. Figure 7.1 shows a conceptual prototype of the interface.

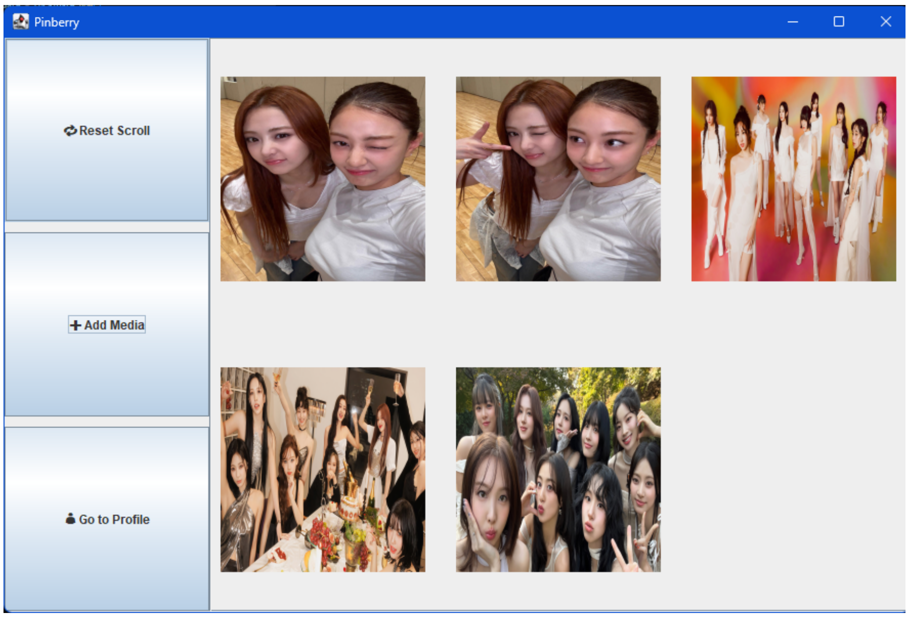

Figure 7.1: interface prototype emphasizing simplicity and clarity

The desktop implementation includes intuitive controls for uploading, viewing, deleting, and managing folders. Dynamic layout components allow content to be refreshed in real-time as the user interacts with the system.

7.6 Summary

The results of the PinBerry implementation confirm that a minimal yet well-structured system can meet the core needs of image-based content management without the complexity typical of social platforms. The use of object-oriented design, modular class responsibilities, and polymorphic interfaces produced a scalable, flexible codebase that meets the identified user requirements.

The platform serves as both a practical application and a demonstration of applying software engineering principles such as SOLID, interface-based design, and layered architecture. The current system offers a robust foundation for future expansions such as multi-user support, cloud-based persistence, search/filter capabilities, and support for additional media formats.

Chapter 8

Discussion and Analysis

This chapter presents a comprehensive interpretation of the results obtained during the development and evaluation of PinBerry. Beyond validating the functional correctness of the system, this section reflects on the architectural choices, analyzes the significance of the design in the context of existing literature, and outlines limitations that emerged during implementation. The chapter aims to position PinBerry not only as a functioning prototype but as a conceptual contribution to the broader discourse on minimalist digital tools for media management.

8.1 Evaluation of Results

PinBerry was conceptualized in response to the increasing complexity and cognitive load imposed by traditional social platforms such as Instagram, Pinterest, and Tumblr. These platforms, while powerful in fostering engagement, often overload users with a mix of social, commercial, and algorithmically curated content, which can interfere with the primary act of storing and revisiting personal memories (**? ?**).

In contrast, PinBerry was deliberately designed to remain lightweight, functionally focused, and ethically aware. The system delivers a carefully selected subset of features—uploading, deleting, downloading, organizing, and viewing images—within a clean interface free from social comparison mechanisms. The use of Java Swing provided an appropriate balance between visual structure and low system resource usage, contributing to the application's lightweight nature.

Architecturally, the results support the decision to use object-oriented design with strict adherence to the SOLID principles. The introduction of the `Multimedia` interface enabled polymorphic handling of different media types, improving the system's flexibility and maintainability. Each core class—`User`, `Foto`, `Gif`, `Video`, `Folder`, and `App`—was constructed with a single responsibility in mind, reducing code complexity and enhancing testability. These outcomes reflect the theoretical benefits of applying modularity, encapsulation, and low coupling in early-stage system development.

Importantly, the project's visual layer was validated not only through implementation but also through UI mockups created in Canva. These design exercises informed layout decisions in the Java Swing interface, helping to ensure that the final product adhered to principles of visual clarity, low perceptual load, and affective design (**?**).

8.2 Significance of the Findings

The findings from PinBerry's implementation are significant for several reasons, both from a technical and human-centered design perspective.

First, the successful demonstration of a functional photo management system without social networking elements illustrates a growing demand for software that values simplicity and autonomy over engagement metrics. In a landscape where most digital platforms are engineered to maximize time-on-platform, PinBerry challenges this norm by emphasizing user control, cognitive well-being, and data transparency. This aligns with recent literature that criticizes overuse of algorithmic content delivery and infinite scrolling patterns for their adverse psychological effects (**? ?**).

Second, from a systems engineering standpoint, the success of the project reinforces the value of architectural discipline in early development stages. By adhering to core object-oriented design patterns and using CRC cards, UML diagrams, and sequence diagrams, the design process remained traceable and adaptable. The resulting architecture is one that could feasibly support additional features (such as metadata search, cloud storage, or image filtering) without refactoring the core logic.

Third, the system supports the concept of a healthier relationship between users and their visual data. The absence of followers, likes, or notifications eliminates behavioral nudges that encourage superficial interaction. Instead, PinBerry promotes reflective engagement with personal content—an approach consistent with affective design theory, which emphasizes emotional satisfaction and long-term positive association with tools (**? ?**).

These findings suggest that PinBerry is not only a viable technical solution but also a conceptual alternative that contributes to the discussion on ethical interface design.

8.3 Limitations

Despite its contributions, the current version of PinBerry exhibits several limitations that must be acknowledged. These limitations span across architectural, functional, and experiential dimensions:

Architectural Constraints

The project currently supports only a single static user, without a registration or login system. This limits its applicability in multi-user scenarios and prevents personalization or access control. Moreover, the use of plain text files for data storage—while useful for prototyping—lacks the robustness, scalability, and security of a database system. The structure is vulnerable to data corruption, lacks indexing, and is inefficient for larger datasets.

Functional Limitations

Although the system supports multiple media types through polymorphism, it does not provide playback or preview capabilities for videos or GIFs. These formats are treated structurally, but not interactively. Furthermore, folder management remains basic: users cannot rename folders, add descriptions, or perform bulk actions. Additionally, there is no search or filtering mechanism to navigate large collections of media, which may impair usability in real-world scenarios.

UI and UX Limitations

The GUI, implemented with Java Swing, is functional but limited in aesthetic flexibility and responsiveness. Animations, responsive resizing, and modern visual effects are absent. Mobile interaction was considered in mockups but not implemented; therefore, the application remains constrained to desktop environments.

Process and Testing Limitations

Due to the scope of the project, formal unit testing and usability testing were not conducted systematically. This absence means that interface behavior under edge cases (e.g., uploading corrupt files or managing large media collections) is not fully validated.

Scalability and Maintainability

While the application's architecture is modular, certain aspects—such as file I/O and GUI logic—remain tightly coupled within the controller class. This may limit the ability to adapt the interface or data layer independently in future iterations. Abstracting these concerns into separate service classes or using a model–view–controller (MVC) framework could improve scalability.

8.4 Summary

This chapter has provided a detailed interpretation of the implementation results of PinBerry, situating its functional outcomes within broader discussions on usability, ethical design, and system architecture. The findings confirm the feasibility of creating a minimalist, user-centered image management platform that contrasts with the complexity of conventional visual social media.

Technically, the project demonstrates the effectiveness of applying object-oriented and SOLID principles to build a clean and maintainable codebase. From a design ethics perspective, the system shows how removing social comparison features can lead to a calmer and more intentional user experience.

However, the limitations—such as single-user functionality, lack of playback, and the use of text-based persistence—highlight areas for improvement and offer a foundation for future research and development. With proper refactoring, expanded features, and user testing, PinBerry could evolve into a robust, cross-platform application that embodies both technical excellence and psychological mindfulness in software design.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

The PinBerry project was initiated with the aim of designing and implementing a lightweight, user-centered photo management system that avoids the complexities and social pressures associated with mainstream visual platforms. Grounded in object-oriented programming principles and developed through an iterative software engineering process, PinBerry seeks to offer a focused tool for uploading, organizing, and viewing personal media without the cognitive and emotional burden typical of algorithm-driven social environments.

The initial problem identified was the lack of simple, distraction-free tools for media organization in a digital landscape dominated by socially saturated applications. Through the analysis of platforms such as Instagram, Pinterest, and Tumblr, it became evident that many users face a trade-off between engagement and usability. PinBerry responds to this gap by providing core functionalities—uploading, deleting, downloading, and organizing images—within an architecture that emphasizes usability, maintainability, and ethical design.

The main objectives of the project were successfully fulfilled. A clear and modular class architecture was implemented using Java, with polymorphic support for multiple media types through a shared `Multimedia` interface. The interface design, influenced by affective and perceptual load theories, prioritized visual clarity and cognitive simplicity. A file-based persistence model allowed for session continuity, and the graphical user interface provided smooth interaction through Java Swing.

From a methodological standpoint, the use of UML diagrams, CRC cards, and sequence modeling ensured the traceability of design decisions. The application was also shaped by ethical principles: all social features commonly associated with self-image pressure and engagement addiction were omitted intentionally. This marks one of the project's most significant conceptual contributions.

While the system remains a prototype, it effectively demonstrates how principled design choices, combined with targeted functionality, can lead to a usable and thoughtful software solution. The results validate that technical minimalism—when correctly implemented—can support both software quality and positive user experience. This aligns with current conversations in HCI and digital ethics, where there is increasing concern over how interface design influences mental health, user autonomy, and digital well-being.

9.2 Future Work

While the core functionalities of PinBerry were successfully implemented, several areas for future development and improvement have been identified, both at the technical level and in terms of user experience.

First, the current application supports only a single-user mode with fixed identity assumptions. Future iterations should incorporate a fully functional multi-user system, including user registration, authentication, and secure session management. This enhancement would enable personalization and enable the deployment of the platform in multi-user environments.

Second, the storage system—currently based on text files—should be replaced with a scalable database solution. Adopting relational or document-oriented databases would allow for faster queries, stronger data integrity, and better scalability. This transition is critical for handling large volumes of user data and multimedia content.

Third, although videos and GIFs are handled structurally via class polymorphism, playback functionality and media preview capabilities are currently absent. Integrating media renderers would enrich the interaction model, especially for users organizing dynamic content. Similarly, folder operations could be expanded to include editing, filtering, and tagging, improving content discoverability.

Additionally, the graphical user interface, while functional, could benefit from modernization. Migrating from Java Swing to a more modern UI framework—such as JavaFX for desktop or Flutter/React Native for cross-platform deployment—would improve responsiveness and visual appeal. Given that mobile-first design principles guided the early prototyping phase, developing a dedicated mobile version remains a logical extension.

On the architectural side, further modularization can be introduced by decoupling the GUI, logic, and data layers more explicitly. Applying an MVC (Model-View-Controller) or MVVM (Model-View-ViewModel) pattern would enhance testability, maintainability, and facilitate future scaling.

Finally, future research could explore the integration of basic computer vision tools to automatically tag or cluster photos, further assisting users in organizing their content without manual sorting. Additionally, usability studies with real users would provide empirical validation of the system's strengths and uncover additional improvement opportunities.

In conclusion, PinBerry has laid a solid foundation for a sustainable, ethical, and minimalistic image management tool. The path forward includes expanding technical capabilities, enhancing interface usability, and refining the system's alignment with human-centered design principles. These enhancements would help PinBerry transition from a conceptual prototype into a deployable, real-world solution.

Chapter 10

Reflection

The development of PinBerry was not only a technical challenge but also a significant collaborative learning experience. As a team of two, we had to coordinate our ideas, balance responsibilities, and adapt our decision-making processes in order to bring a shared vision to life. The experience taught us not only about software engineering practices but also about the interpersonal and strategic aspects of working effectively as a development team.

From the outset, we aimed to go beyond simply building a functioning application. We wanted to create something that responded to real user needs—particularly the need for a simple, distraction-free tool to manage personal images. This goal shaped both our design philosophy and our technical implementation strategy. We applied object-oriented programming principles rigorously, emphasizing class responsibility, modularity, and the use of interface-based polymorphism. Throughout this process, we each developed stronger design thinking skills and a deeper appreciation for the role of abstraction and encapsulation in maintainable systems.

Our decision to avoid common social features such as comments, likes, and followers stemmed from research into the cognitive and emotional strain these features can impose on users. This research phase not only enriched our understanding of current design ethics but also pushed us to think more critically about the purpose of our system. Instead of designing for engagement, we focused on clarity, user autonomy, and ease of use—values we now see as fundamental to responsible software development.

Working as a pair also taught us the importance of communication, version control, and collaborative planning. We used Git and GitHub to manage changes and resolve conflicts, which improved our technical fluency and gave us practical experience with tools widely used in the software industry. Writing in LaTeXas co-authors forced us to articulate our ideas clearly and merge different writing styles into a coherent academic report.

Of course, we faced several challenges. Some limitations, such as the absence of a multi-user system and the lack of media previews, stemmed from time constraints. Others arose from our early design choices. In hindsight, we could have adopted a more decoupled architecture—perhaps using an MVC pattern—to reduce dependencies between the interface and logic layers. We also realized too late the importance of early feedback from potential users, which could have guided more effective interface design decisions.

Despite these challenges, we are proud of what we have achieved. PinBerry represents not just a functioning prototype, but a thoughtful approach to digital simplicity and mental wellness. The process taught us to navigate both the technical and ethical dimensions of system development. We leave this project with a stronger command of Java, UML modeling, Git, LaTeX, and GUI design—but more importantly, with a better understanding of how to plan, execute, and reflect critically on a collaborative project.

This experience has strengthened our teamwork, enhanced our problem-solving capabilities, and deepened our understanding of what it means to build software that is both useful and responsible. These lessons will serve as a foundation for our future academic and professional endeavors.

References

[1] E. K. Taylor, *Using the Evolution of the Instagram User Interface as a Case Study to Explore How Users Experience Platform Changes*, University of Virginia, 2020. [Online]. Available: https://libraetd.lib.virginia.edu/public_view/02870x510

[2] S. M. Safitri, R. Firdaus, and M. N. Dhiya, "Analysis on User Interfaces Readability: A Case Study of Instagram," *International Journal of Engineering Research and Technology*, vol. 9, no. 7, pp. 690–694, 2020. [Online]. Available: https://www.researchgate.net/publication/343511368

[3] S. Aggarwal, "Usability Evaluation of Instagram: A Comprehensive Analysis," *International Journal of Computer Applications*, vol. 183, no. 47, pp. 8–13, 2021. [Online]. Available: https://www.researchgate.net/publication/376798583

[4] D. A. Shamma et al., "Self-Presentation Effects on Content Propagation in Tumblr," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020. [Online]. Available: https://dl.acm.org/doi/fullHtml/10.1145/3394231.3397893

[5] K. Chancellor, S. Lin, and M. De Choudhury, "A content analysis of thinspiration images and text posts on Tumblr," *International Journal of Human-Computer Studies*, vol. 108, pp. 29–40, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1740144517302462

[6] V. Coronado, "Pinterest Usability: Too Much and too Little!," *Medium*, 2020. [Online]. Available: https://medium.com/@vc2497/ux-1-pinterest-usability-too-much-and-too-little-81417855119d

[7] T. Bakker and S. de Vreese, "Pinterest: A Unicorn Among Social Media? An Investigation of the Platform's Quality and Specifications," *ResearchGate*, 2020. [Online]. Available: https://www.researchgate.net/publication/343392042

[8] S. R. Smith and J. R. Caruso, "Public Health Implications of Image-Based Social Media," *The Permanente Journal*, vol. 23, 2019. [Online]. Available: https://www.thepermanentejournal.org/doi/10.7812/TPP/18.307

[9] C. Tolentino, "Why Twitter's New Interface Makes Us Mad," *The New Yorker*, Jul. 2020. [Online]. Available: https://www.newyorker.com/culture/infinite-scroll/how-social-media-redesigns-manipulate-us

[10] J. Sweller, "Cognitive Load Theory," The Alien Design, 2025. [Online]. Available: https://www.thealien.design/insights/cognitive-load-in-ux-design?

[11] N. Lavie, "Perceptual Load Theory," Wikipedia, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Perceptual_Load_Theory

[12] Contributors to Wikimedia projects "Affective design", Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Affective_design?

[13] N. Kock, "Media Naturalness Theory: Human Evolution and Behaviour Towards Electronic Communication Technologies," Oxford Academic, 2011. [Online]. Available: https://academic.oup.com/book/8882/chapter/155142638