UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Universidad Distrital Francisco José de Caldas

Facultad de Ingenieria

Habitra: A Habit Tracker Focused on Fun by Making it a Video Game

Fabián Cabrera-20242020020

Juan Avila-20242020030

Proffesor: Carlos Sierra Vargas

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Students in *Systems engineering*

July 10, 2025

Declaration

We, Fabián Cabrera and Juan Avila, of the Facultad de Ingenieria, Universidad Distrital Francisco José de Caldas, confirm that this is our 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. We understand that failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalized accordingly.

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

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

Fabian Cabrera , Juan Avila
July 10, 2025

Abstract

Habit tracking is commonly used for sustaining life-improving routines. However, it often becomes monotonous. Habitra introduces a gamified habit-tracking solution that rewards users with an experience-based system inspired by role-playing games. This report presents the design philosophy, development process, and expected outcomes of the application.

Keywords: CRC Cards,UML Diagrams, Mockups, Object-Oriented Design, Gamified Interface,

Contents

List of Figures

List of Abbreviations

SMPCS	School of Mathematical, Physical and Computational Sciences
OOP	Object-oriented programming
UML	Unified modeling language
CRC	class-responsibility-collaborators
GUI	Grafic User Interface

Chapter 1

Introduction

1.1 Background

What is habitra?: Habitra is a mobile app designed to improve users' lifestyles by helping them track and complete habits and tasks. It rewards users using a gamified system that includes experience points and levels, making habit formation more engaging.It was created thinking on the problem of the bored process of acquiring a habit

It was created thinking on the problem of the bored process of acquiring a habit The scope of the Habitra app encompasses the development of a mobile habit-tracking tool that integrates gamification to encourage users to build and maintain positive habits while discouraging negative ones. The application allows users to add and manage habits, tasks, and bad habits, offering a visual interface inspired by role-playing game.The initial version focuses on individual habit management on a single device, without incorporating social features or cross-platform synchronization, ensuring a streamlined and focused user experience.

1.2 Problem statement

Exist many individuals face in consistently maintaining positive habits and eliminating negative ones. Traditional habit trackers often become repetitive and lack motivation, leading users to abandon them over time. This leads to the intention and action, where users struggle to sustain behavior change despite knowing its importance for personal development and productivity.

1.3 Aims and objectives

Habitra aims to make an interactive Habit Tracker that keep the habits in check , rewards completing said habits and help the user to fulfill tasks without losing any motivation.

Thanks to interactive tools like the use of a deadline , a difficulty , a streak, etc. in the app reaches to keeping the attention of the users making them improve on their ability to acquire a habit and be able to be constant.

1.4 Solution approach

Habitra's approach centers on transforming the often monotonous task of habit tracking into a dynamic and engaging experience. By incorporating gamification principles and user-friendly design, the app enhances user motivation and fosters consistency in habit formation.

1.4.1 Engaging User Interface

The application features a visually appealing interface with vibrant colors—primarily purple—that evoke a sense of creativity and energy. This design deviates from traditional, rigid layouts and provides a more enjoyable and less formal user experience.

1.4.2 Gamified Mechanics

Habitra introduces game-inspired features that are uncommon in conventional productivity apps. These elements encourage users to interact regularly and build lasting behavioral change.

Leveling System

Users gain experience points and level up by completing tasks and habits. The amount of experience gained depends on the difficulty and importance of the task, which encourages users to take on more meaningful challenges.

Reward System

The application provides tangible in-app rewards for habit completion. These rewards serve as motivational incentives and reinforce positive behavior, helping users maintain their momentum over time.

1.5 Summary of Contributions and Achievements

The development of Habitra has resulted in a functional and well-structured application that effectively demonstrates the integration of gamification with habit tracking. Key contributions and achievements include:

- The complete implementation of the core application logic using object-oriented programming in Java, applying principles such as encapsulation, inheritance, and polymorphism.
- A modular and maintainable architecture guided by SOLID principles, ensuring code scalability and clarity.
- The design and development of an intuitive and visually appealing graphical user interface based on previously created mockups.
- The creation and use of CRC cards and UML diagrams (class and sequence) to guide the system design, illustrating class responsibilities and interactions.
- A reward system inspired by role-playing games, including experience points, levels, and performance feedback to reinforce habit formation.

- Integration of user stories into the development process, ensuring that the final product aligns with the intended goals and provides a meaningful user experience.

Overall, the project demonstrates a successful application of both software engineering principles and behavioral psychology, resulting in a robust foundation for future expansion and enhancement.

1.6 Organization of the report

This report is structured into six chapters, each contributing to a comprehensive understanding of the development and functionality of the Habitra application:

1. **Literature Review**: Presents a survey of related works, applications, and theoretical foundations that influenced the design of Habitra.
2. **Methodology**: Describes the development process, including design choices, use of object-oriented principles, and tools such as CRC cards and UML diagrams.
3. **Results**: Showcases the implementation progress, including core features, the user interface, and code functionality.
4. **Discussion**: Analyzes the strengths, limitations, and potential improvements of the current version of the application.
5. **Conclusion**: Summarizes the key achievements, insights gained, and the effectiveness of the solution.
6. **Reflection**: Offers personal insights and lessons learned during the development process.

Chapter 2

Literature Review

2.1 Theoretical Foundations in Habit Formation

The field of behavior modification has been extensively studied across disciplines such as psychology, personal productivity, and software development. One of the most influential models is the **habit loop**, which comprises three components: a cue, a routine, and a reward. This model is foundational in modern habit-tracking applications, as it enables the creation of systems that repeatedly reinforce desired behaviors through feedback loops.

James Clear, in his book *Atomic Habits* Clear (2018), argues that forming good habits depends less on ambitious goals and more on small, consistent daily actions. He emphasizes that habit formation is closely linked to identity and that immediate, tangible rewards are crucial in reinforcing behavior. This principle has directly influenced the design of gamified tools such as Habitica Habitica (2024) and also the development of Habitra, which applies a reward logic based on experience points and levels to reinforce positive habits.

Recent studies have also explored the psychological effects of dependency in digital habit-tracking tools. For instance, Renfree et al. highlight that the dependency relationship formed between users and apps can play a crucial role in sustained engagement and habit maintenance Renfree and Cox (2016).

2.2 Relevance to the Development of Habitra

The reviewed literature provides both theoretical and empirical support for the design of Habitra. Concepts from *Atomic Habits* underscore the significance of incremental behavior reinforced by immediate rewards—concepts that are at the heart of Habitra's reward system.

Similarly, studies on gamification highlight its effectiveness in improving user engagement and motivation through systems that offer feedback, progression and a sense of accomplishment. These ideas validate Habitra's use of experience points, levels and tracking user progress as mechanisms to reinforce habits. By basing the core mechanics of the application on proven behavioral theories and successful implementations such as Habitica and insights from dependency theory Renfree and Cox (2016), the review ensures that Habitra is not only technically sound, but also psychologically effective in addressing the core problem: maintaining the user's long-term motivation for habit formation.

2.3 Critical Review of Sources

The literature reveals that consistent habit formation is closely linked to psychological reinforcement mechanisms such as rewards, identity-based behaviors and repetition. One of the most influential sources, James Clear's *Atomic Habits*, emphasizes that small, incremental changes—when combined with immediate positive reinforcement—lead to long-term behavioral transformation. This idea supports the use of gamification in habit-tracking systems, where rewards such as points or levels can sustain motivation. Additionally, the study by Renfree et al. Renfree and Cox (2016) highlights the importance of app-user dependency in sustaining long-term behavioral change, suggesting that emotional attachment and perceived support may further influence adherence.

2.4 Summary

The review of relevant literature demonstrates a strong connection between habit formation, behavioral psychology, and gamification techniques. Key theories highlight the importance of small, consistent actions reinforced by immediate rewards—principles that form the core of the Habitra application. Studies and examples like *Atomic Habits* Clear (2018), Habitica Habitica (2024), and Renfree et al. Renfree and Cox (2016) confirm that reward-based and dependency-aware systems can significantly improve user motivation and adherence.

Chapter 3

Methodology

3.1 Analysis Using CRC Cards and UML Diagrams

The development of Habitra followed a structured object-oriented analysis and design methodology aimed at building a scalable, maintainable, and modular system. The initial step in this process involved the use of CRC (Class-Responsibility-Collaborator) cards, which allowed the team to conceptualize the application's components in a simplified and intuitive format. Each CRC card defined a class, its main responsibilities (i.e., its core functions), and the classes it collaborates with. For example, the `Habit` class is responsible for managing habit data and interacts with the `User` class for tracking purposes and with the `RewardSystem` class to allocate experience points.

Once the foundational responsibilities and relationships between classes were outlined via CRC cards, the architecture was further refined using UML (Unified Modeling Language) diagrams. A class diagram was developed to represent the static structure of the system, capturing the relationships among major classes such as `User`, `Habit`, `Task`, `BadHabit`, and `RewardSystem`. This diagram made explicit the inheritance, associations, attributes, and methods relevant to each class.

Additionally, a sequence diagram was constructed to model dynamic behavior—specifically, how objects interact over time during key operations, such as task completion. This clarified the communication pathways between the graphical interface and the backend, ensuring consistency in data flow and behavior.

The combined use of CRC cards and UML diagrams ensured alignment between conceptual design and implementation. This methodology facilitated team collaboration, clarified system structure early in development, and ultimately contributed to a robust and coherent application architecture.

3.1.1 CRC Cards

CRC (Class–Responsibility–Collaborator) cards are a lightweight object-oriented design tool used during the early stages of software development. Each card represents a single class and outlines its primary responsibilities and the collaborators needed to fulfill them. This approach supports modular design by emphasizing separation of concerns and promoting class cohesion. In Habitra, CRC cards were used to define essential classes such as `User`, `Habit`, `Task`, and `RewardSystem`. These helped in establishing the expected behavior and interactions between components before transitioning to formal diagrams.

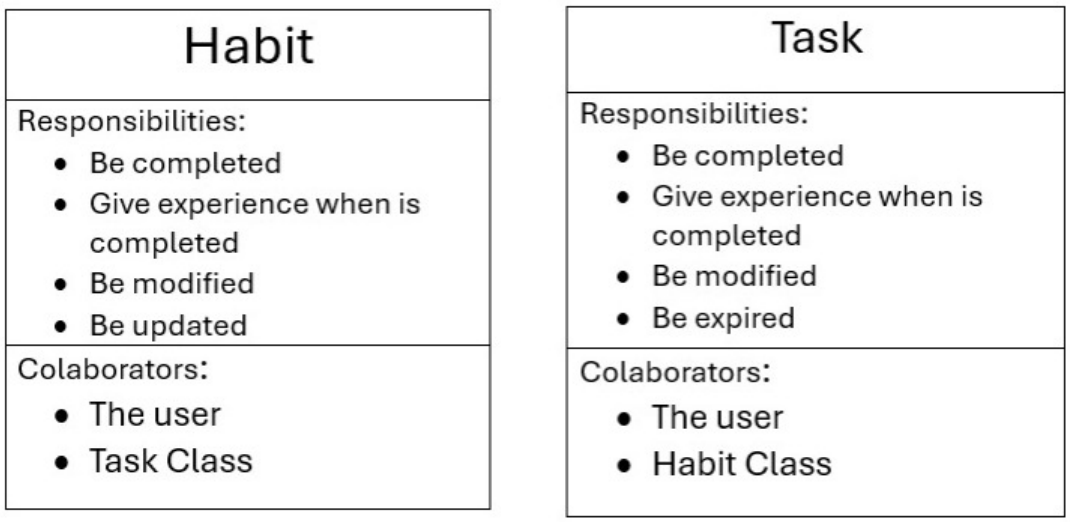


Figure 3.1: Task and Habit CRC cards

3.1.2 UML Diagrams

UML diagrams serve as visual representations of both the static and dynamic aspects of a software system. Structural diagrams, such as class diagrams, illustrate the relationships and hierarchy between classes. Behavioral diagrams, such as sequence diagrams, depict the flow of interactions among objects over time. In the context of Habitra, UML diagrams provided a blueprint of the system's architecture and interactions. These diagrams aided in visualizing key behaviors like experience calculation and task tracking, ensuring alignment between the conceptual model and eventual implementation.

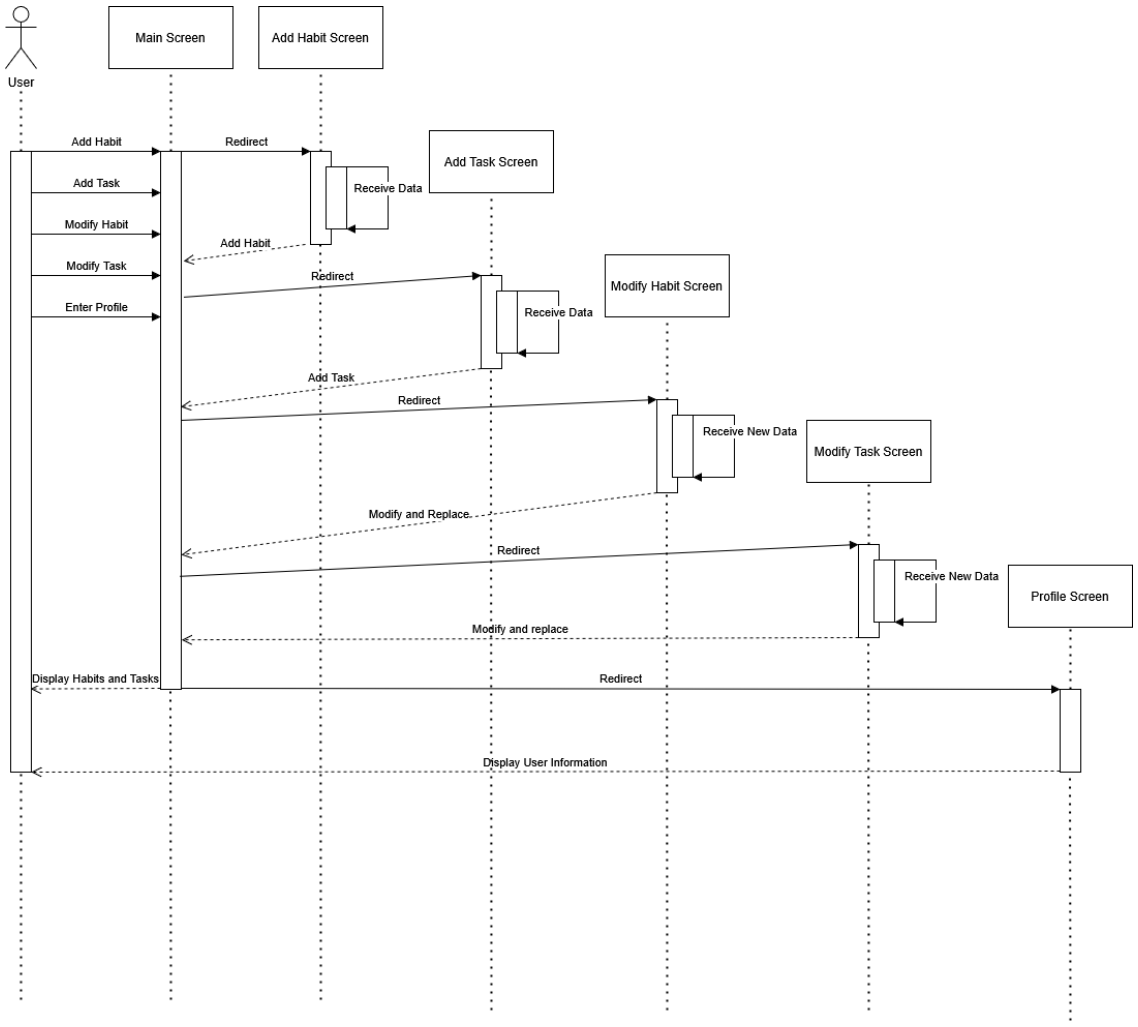


Figure 3.2: Sequence Diagram

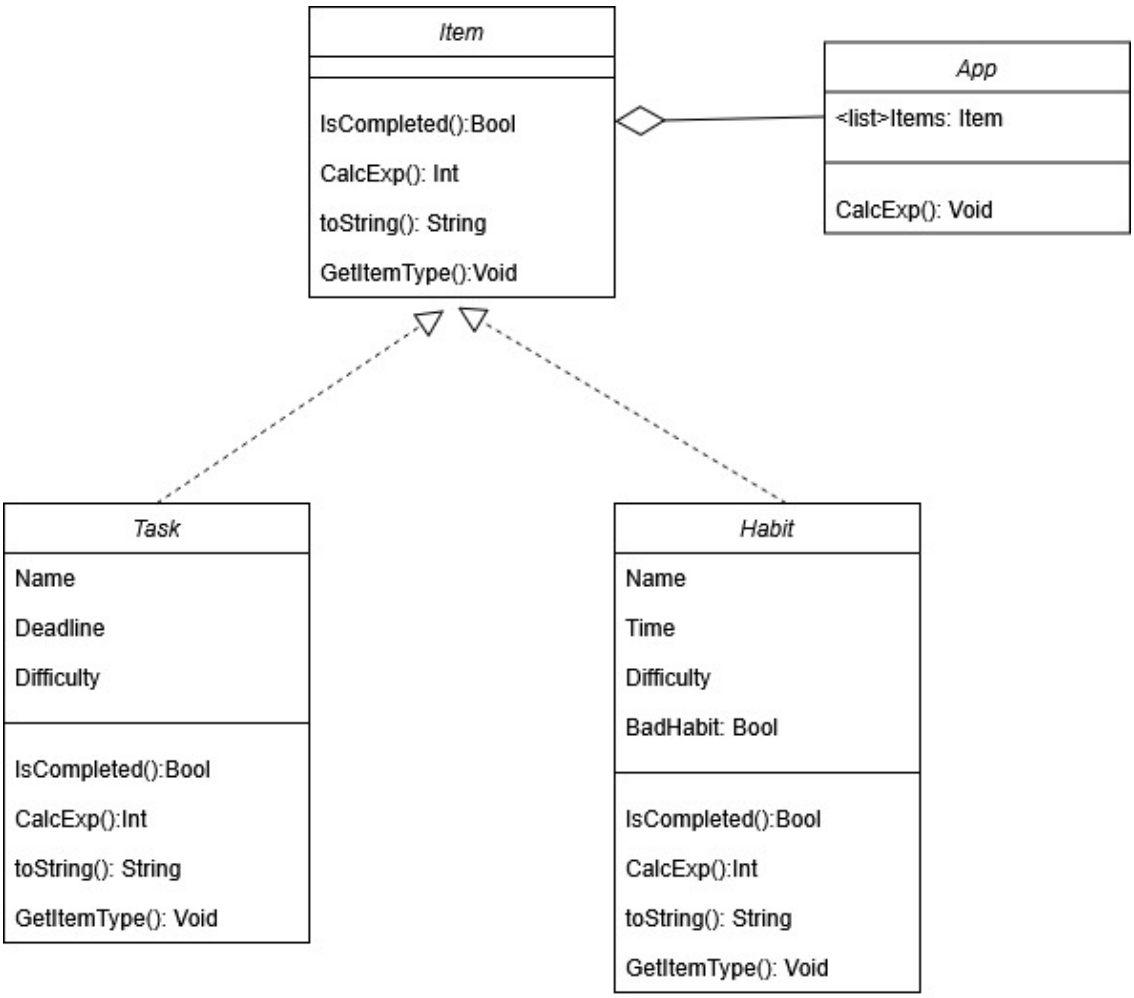


Figure 3.3: Class Diagram

3.1.3 Mockups

Mockups were developed to visualize the application's interface and guide user-centered design. They served as visual prototypes representing how the different components interact from a user perspective. These mockups supported the logical structure of the backend by linking interface features with functionality, helping bridge the gap between visual design and code implementation.

Figure 3.4: Mockup of Main Interface

Figure 3.5: Mockup add new habit

3.1.4 Application of Object-Oriented Principles and SOLID Design

The development of Habitra was guided by the core principles of object-oriented programming (OOP), including encapsulation, inheritance, abstraction, and polymorphism. These principles were fundamental in ensuring that the codebase remained modular, reusable, and maintainable.

Encapsulation Encapsulation will be achieved by keeping user data private: Username and password will be kept private. To allow access to the app, a set method for both username and password will check if they are correct. If both are correct, the user will have access to the application.

Inheritance was utilized to promote code reuse and logical class hierarchy. For example, common attributes and behaviors between `Habit`, `Task`, and `BadHabit` were abstracted into a shared superclass, and making use of interfaces reducing redundancy and facilitating polymorphism.

Polymorphism Polymorphism will be mainly achieved by overriding, since child classes and classes that implements from an interface will implement some methods and attributes differently, or have no need to use them at all. Such as buttons, that, despite having the same mother class, work differently, since not all of them receive data or redirect to other screens.

Overcharging is also present, with methods that have the same name, but different parameters.

Abstraction was enforced through the use of abstract classes and interfaces, defining general behaviors while leaving specific details to be implemented by subclasses. This design approach supported flexibility and scalability.

Moreover, the system adhered to the **SOLID** principles, a set of five design guidelines aimed at improving software architecture:

- **Single Responsibility Principle (SRP)**: Single responsibility is achieved when each class and object have a unique role and responsibility that doesn't overlap with one another. In Habitras case, each class is designed in such a way that it complies with this principle, by having multiple classes that assure the flow of information in the application by each processing the information they are concerned about, without displaying any information and vice versa.

- **Open/Closed Principle (OCP)**:Open/Closed tells us that "classes should be open to extension, but closed to modification",what this means is that we should be able to add more functionalities without modifying the class itself. Habitra does this by having multiple interfaces that can be easily accessed by other classes to check instances of said interfaces, while also allowing us to add new methods and behaviors without modifying any of the already existing methods in any other class.

- **Liskov Substitution Principle (LSP)**: Liskov Substitution is, in simple terms, the ability of a child class to replace its mother or base class. In Habitras case, this principle does not apply, since all the classes have a very concrete and strict hierarchy along with their responsibilities.

- **Interface Segregation Principle (ISP)**: Interface Segregation says that no class should be forced to use methods or interfaces it doesn't need. Habitra complies with this principle by having classes that implement interfaces in a clear and practical manner, using universal and abstract methods that apply to every class that implements said interface.

- **Dependency Inversion Principle (DIP)**: Dependency Inversion states that no class should connect to low level modules or concretions, but abstractions instead, this means that classes should depend on interfaces and other high-level classes. Habitra achieves this by connecting its classes via interfaces and not directly with each other.

The integration of these OOP and SOLID principles ensured that the Habitra codebase remained clean, robust, and adaptable to future enhancements.

3.2 Implementation

During development, CRC cards, UML diagrams, mockups and all the OOP principles were key in guiding both the structure and logic of Habitra. The CRC analysis provided a conceptual foundation by identifying core components and their responsibilities. UML diagrams, including class and sequence diagrams, refined this foundation by visually detailing relationships and processes—such as how rewards are assigned after completing a habit.

Simultaneously, the mockups helped in designing an intuitive and gamified interface. These visual guides ensured consistency between the back-end logic and user interface, inspired by role-playing game elements like experience levels and task rewards.

The combined use of these tools enabled a smooth transition from design to implementation, ensuring that the user experience and system logic aligned effectively.

3.3 Summary

The development methodology for Habitra integrated object-oriented design principles with visual planning tools to support both backend structure and user interface design. CRC cards facilitated the definition of modular class responsibilities. UML diagrams provided a visual overview of system behavior and architecture. Mockups ensured a user-friendly and engaging interface. Together, these elements formed a cohesive approach that enhanced clarity, collaboration, and implementation efficiency.

Chapter 4

Results

4.1 Implementation Results

The initial design phase of the application focused on conceptualizing the idea, identifying key components, and analyzing how each part could be modeled using Java classes and objects. This included the creation of class diagrams and CRC cards to represent the logical structure of the application, as well as mockups of the user interface.

The main functionalities implemented so far include:

- A dynamic main menu that displays all habits, tasks, and bad habits.
- The ability to check off completed items and receive rewards or penalties based on completion status.

These features were developed by applying object-oriented programming principles such as abstraction, inheritance, and polymorphism. The app architecture was also guided by SOLID principles, resulting in a modular and maintainable structure. As a result, the application behaves according to the predefined user stories and fulfills the core requirements laid out in the design phase.

4.2 Goals

The main goal of the project is to create an interactive and engaging user interface that aligns with the previously designed mockups and functions correctly across all features.

We aim to:

- Fully implement object-oriented programming concepts such as encapsulation, inheritance, abstraction, and polymorphism.
- Apply each of the SOLID principles where appropriate to enhance maintainability, scalability, and readability.
- Finalize and improve any incomplete features, particularly within the graphical user interface, to ensure a polished and user-friendly experience.

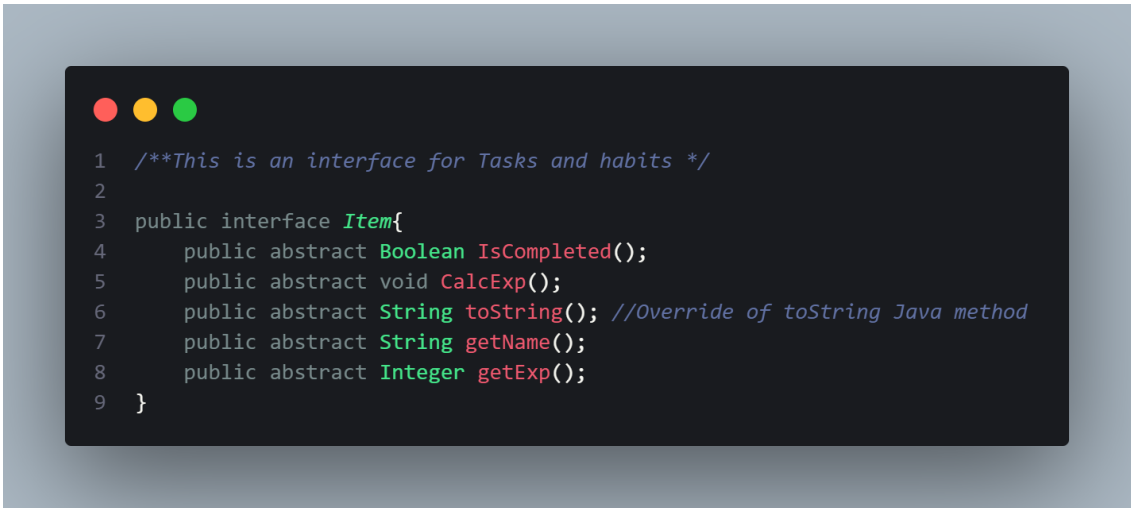
4.3 Visual Progress

Figures 4.1 through 4.7 show the current state of the interface and the structure of the application.

```java
/**This is a class for Habits */
public class Habit implements Item{
    private String habName;
    private int time;
    private int exp;
    private int diff;
    private Boolean status;
    private Boolean isBad;

    /**Constructor for habits
     * @param name: The name of the habit
     * @param time: Periodicy of the habit
     * @param difficulty: The difficulty of the habit
     * @param status: If the habit is completed or not
     * @param bad: Wheter the habit is flagged as bad or not
     */
    public Habit(String name, int time, int difficulty, boolean status, Boolean bad){
        this.habName= name;
        this.time= time;
        this.diff= difficulty;
        this.status= status;
        this.isBad= bad;
    }
```

Figure 4.1: class habit that implements from Item



```java
/**This is an interface for Tasks and habits */

public interface Item{
    public abstract Boolean IsCompleted();
    public abstract void CalcExp();
    public abstract String toString(); //Override of toString Java method
    public abstract String getName();
    public abstract Integer getExp();
}
```

Figure 4.2: Item interface

```
loadHabitsFromFile();
        loadBadHabitsFromFile();
        loadTasksFromFile();

        startFrame();
        startButtons();

        //Set main panel
        panel.setLayout(null);
        panel.add(addH);
        panel.add(addT);
        panel.add(addB);
        panel.add(chngH);
        showHabitsInPanel();
        panel.setBackground(Color.decode("#472287"));

    }
    /**Redirects to the corresponding panel */
    private static void event(int id){
        panel.setVisible(false);
        newPanel = new JPanel();
        newPanel.setBounds(0, 0, 1920, 1080);
        switch (id) {
            case 1:
                newPanel.setBackground(Color.decode("#911bcc"));
                AddHabit();
                break;
            case 2:
                newPanel.setBackground(Color.decode("#8c1176"));
                AddTask();
                break;
            case 3:
                newPanel.setBackground(Color.decode("#2c118c"));
                AddBadHabit();
                break;
            case 4:
                newPanel.setBackground(Color.decode("#9b59b6"));
                ChangeHabit();
                break;
            default:
                newPanel.setBackground(Color.RED);
                break;
        }
```

Figure 4.3: GUI Code

Figure 4.4: Mockup of the main screen

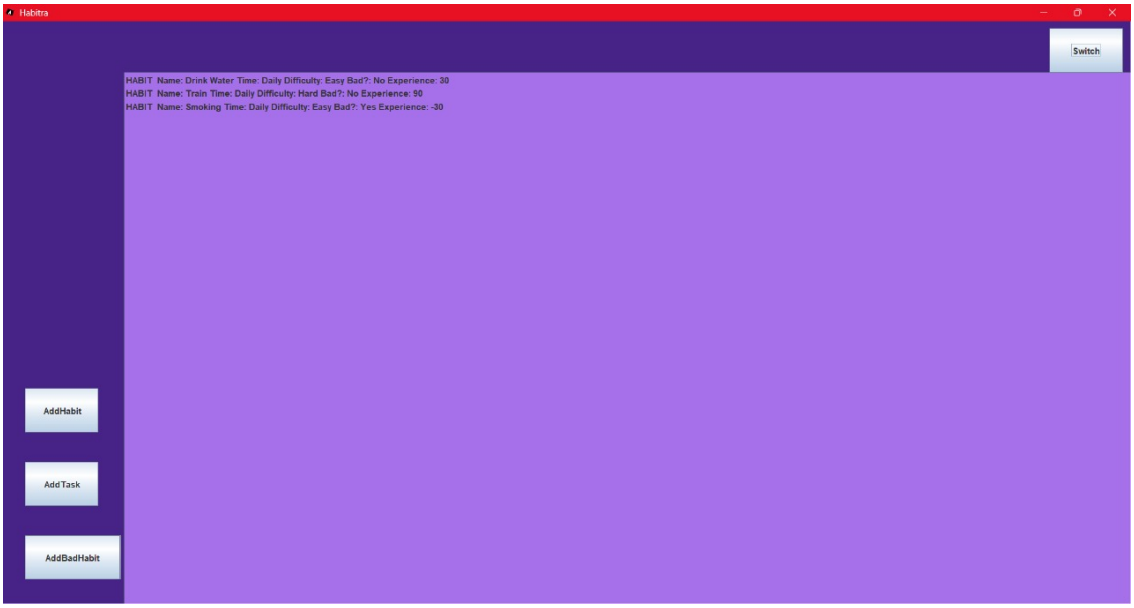


Figure 4.5: Main Screen Done

Figure 4.6: Mockup of the add habit screen

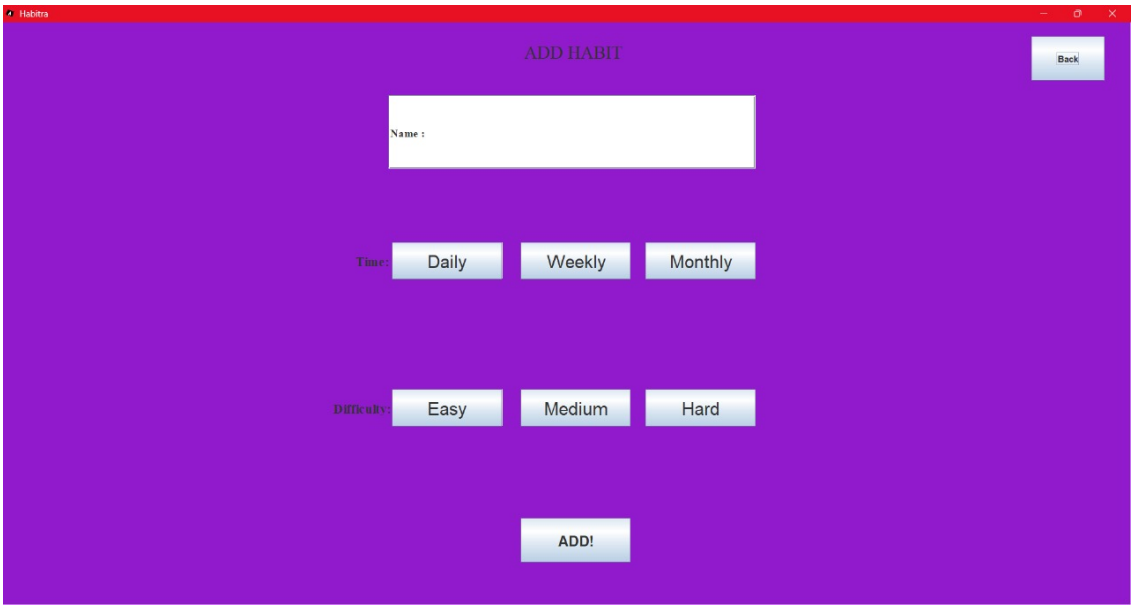


Figure 4.7: Add Habit screen done

4.4 Summary

The development of the Habitra application has been successfully completed, fulfilling the core objectives outlined in the design phase. The integration of object-oriented programming principles with a gamified user experience has resulted in a fully functional habit-tracking tool.

The implementation closely followed the initial mockups and design models, including CRC cards, UML diagrams, and interface sketches. All major functionalities—such as habit and task tracking, reward mechanisms, profile management, and graphical interface—have

been completed and are working as intended. The final product reflects a coherent structure that is both modular and extensible, setting a strong foundation for future improvements or additional features.

Chapter 5

Discussion and Analysis

This chapter discusses the outcomes of the development and implementation of the Habitra application. It evaluates how well the system meets its intended goals and reflects on the effectiveness of its design and functionality. It also addresses the significance of the results, the limitations encountered, and opportunities for future enhancement.

5.1 Evaluation of Implementation

The results demonstrate that Habitra successfully integrates gamification mechanics within a habit-tracking environment. The core features—such as habit/task tracking, user progression through levels, and the application of rewards and penalties—are fully operational. The use of object-oriented principles ensured a modular structure, allowing the application logic and interface to be developed independently and then integrated smoothly.

User stories defined in the early stages were fully covered, and the mockups were faithfully translated into the final interface. The application structure, following SOLID principles, made it easier to implement changes, test components individually, and maintain code clarity.

5.2 Significance of the Findings

The completed application illustrates how gamification can enhance user engagement in personal productivity tools. The integration of role-playing game elements (such as leveling and rewards) makes the process of building habits more interactive and motivating. From an academic and development standpoint, Habitra serves as a successful case study of applying behavioral theory—particularly from sources like *Atomic Habits*—into functional software.

Furthermore, the project demonstrates the importance of using structured design tools such as CRC cards and UML diagrams in guiding the development of a scalable and maintainable system. It shows that a clear theoretical and design base improves not only the code structure but also the user experience.

5.3 Limitations and Areas for Improvement

While the project was successfully completed, several limitations remain:

- **Single-user system:** The current implementation does not support multi-user profiles or cloud synchronization, limiting usability across multiple devices.

- **Poor data persistence:** The application stores data locally and temporarily trough a txt archive ; future versions could benefit from file-based or database-backed persistence.
- **Limited customization:** Users cannot yet customize aspects such as habit categories, color schemes, or difficulty labels beyond predefined values.
- **Minimal analytics or feedback:** The system does not currently provide long-term tracking or visual statistics that could help users analyze their progress.

These limitations present opportunities for future development to further increase functionality and improve user satisfaction.

5.4 Summary

This chapter provided an analysis of the completed application and its effectiveness in achieving the intended objectives. The findings show that Habitra delivers a functional and engaging tool for habit tracking, supported by solid software engineering practices. Despite a few limitations, the project stands as a strong base for future growth, both in terms of technical improvement and user-centered features.

Chapter 6

Conclusions

6.1 Conclusions

At this stage of development, the Habitra application successfully demonstrates how gamification principles can be effectively applied to habit tracking to enhance user motivation and engagement. By incorporating a reward system inspired by role-playing games, the application encourages users to build and maintain positive habits while discouraging negative ones.

The use of object-oriented programming (OOP) concepts—such as inheritance, encapsulation, and polymorphism—combined with design techniques like CRC cards and UML diagrams, enabled a modular and scalable architecture. Furthermore, the implementation of user interface mockups resulted in an intuitive and visually appealing experience aligned with the initial design goals.

The system currently allows users to create and manage habits, tasks, and bad habits, while tracking their progress through levels and rewards. Grounded in behavioral science theory, particularly concepts from *Atomic Habits*, the application bridges theoretical foundations with practical functionality to address the challenges of habit formation. Overall, Habitra establishes a solid framework that can be expanded with future features and refinements.

6.2 Future Work

The core functionality of the system is already operational, with the main modules implemented in Java, and an integrated graphical user interface based on the designed mockups. However, there are several areas identified for future enhancement:

- **GUI Refinement:** Further polish the user interface to improve usability, responsiveness, and visual consistency across screens.
- **Feature Expansion:** Add new interactive elements such as habit streaks, deadline alerts, and additional reward types to enrich the user experience.
- **Persistence Improvements:** Enhance data storage and retrieval mechanisms, potentially using file serialization or lightweight databases to support persistent user data across sessions.
- **Scalability:** Prepare the codebase for future extensions like multi-user support, cloud synchronization, and analytics.
- **Testing and Optimization:** Conduct systematic testing to identify and fix bugs, measure performance, and optimize logic and UI behavior.

These future improvements aim to transition Habitra from a functional prototype into a polished, fully-featured application capable of long-term usage and continued development.

Chapter 7

Reflection

Developing Habitra has been a comprehensive and enriching experience that combined technical design, user-centered thinking and behavioral science. By using CRC cards, We gained a clear understanding of how to decompose a system into modular and collaborative components, which significantly improved code organization and scalability. The implementation of UML diagrams helped visualize the architecture and interactions of the system, allowing for better planning and communication within the team. Designing mockups forced me to think from the user's perspective, ensuring that the interface was not only functional, but also attractive and intuitive. Basing the application on behavioral theories, particularly those presented in Atomic Habits, taught me the value of integrating psychological principles into software to effectively influence user behavior. Overall, this project deepened my knowledge of software engineering, design thinking, and research application, and reinforced the importance of creating technically sound, user-centered systems.

References

Clear, J. (2018), *Atomic Habits: An Easy and Proven Way to Build Good Habits and Break Bad Ones*, Avery.

Habitica (2024), 'Habitica: Gamify your life', https://habitica.com. Available at: https://habitica.com.

Renfree, I., H. D. M. P. S. K. and Cox, A. (2016), Don't kick the habit: The role of dependency in habit formation apps, *in* 'Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems', ACM, San Jose, CA, USA, pp. 3222–3233.