National University – Manila
College of Computing and Information Technologies

Artificial Ledger Technology Bank System Software Development Life Cycle

Submitted to:
Professor Jay A. Abaleta

Submitted by:
Maquilan, Joshua
Navarro, Kristine Vine
Pangilinan, Vince Erol
Talosig, Jay Arre
Tiquio, Anilov
Viñas, Gabriel Angelo

COM23P1
Intermediate Programming (CCPRGG2L)

PHASE 1: Planning and Requirement Analysis

Project Objectives and Goals

- To construct a JAVA OOP program using *Java* Swing *GUI environment* for Personal New Bank Account that the new user will register the following information: *username, password, first name, middle name, last name, birthdate, gender, address, father's name, mother's name, contact number, pin code, initial deposit of 500 pesos.*
- The program will store all the information in a *txt file* for the storage of the user information's bank account.
- The program should have features of *Deposit*, *Withdraw*, and *Balance Inquiry*. The initial deposit is the only constant value in the user's registration part which is 500.
- The account number of the user will start at a default value of 2024100000.

Resource Planning

- Kanban Board in GitHub (used for documenting and assigning tasks for each member)
- Kanban Board GitHub Link: https://github.com/users/flexycode/projects/3/views/1
- Tasks and assigned members:

  Assigned to Jay Arre Talosig:
  - Create Login Form with Video Background
  - Setup the task for each member

  Assigned to Jay Arre Talosig and Angelo Vinas:
  - Fixing Object Class
  - Login Form Mockup Design
  - Setup GitHub Project for Java

  Assigned to Joshua Maquilan, Vince Pangilinan, Angelo Vinas:
  - Develop AccountManagement class
  - Develop BankAccount class

  Assigned to Maquilan, Navarro, Pangilinan, Tiquio, Vinas:
  - Testing the codes

  Assigned to Maquilan, Pangilinan, Talosig, Tiquio:
  - Create Software Life Development Cycle (SDLC)

  Assigned to Kristine Navarro and Anilov Tiquio:
  - Selection for background images
  - Create Registration Form

  Assigned to Joshua Maquilan:
  - Sound Effects for all GUI Form

  Assigned to ALL (Maquilan, Navarro, Pangilinan, Talosig, Tiquio, Vinas)
  - Forms for Bank Features
  - Enforce Conventional Commits when contributing to the repository
  - Error Handling
  - Identify the Java Project Requirements

PHASE 2: Defining Requirements

Program Requirements

- Secure the Username and Password.
- Set an exception handling to avoid errors.

PHASE 3: Designing Architecture

Code Project Structure using Maven

- Artificial Ledger Bank diagram tree used "package-by-features" or "package-by-components" pattern. This pattern is commonly used in Object-Oriented Programming (OOP) and is particularly prevalent in frameworks and libraries that promote a modular and component-based approach to application development.

- Project Tree Structure

```
ArtificialLedger.java
│
├── src
│   └── main
│       ├── java
│       │   └── your.package.structure (ArtificialLedger)
│       │       ├── components
│       │       │   ├── EventHomeOverlay ( 💡Interfaces)
│       │       │   ├── HeaderButton (components for Home)
│       │       │   ├── BankAccount ( integration for Account, AccountManager)
│       │       │
│       │       ├── forms
│       │       │   ├── Account (□ GUI )
│       │       │   ├──
│       │       │   ├── Home ( Homepage )
│       │       │   ├── HomeOverlay (Home: )
│       │       │   ├──
│       │       │   ├── Login (📓 Side Panel in Home )
│       │       │   ├── ModelLocation ( Extra: Background Video)
│       │       │   ├── RegistrationForm ( □ GUI )
│       │       │
│       │       └── main
│       │       │   ├── Main ( 🚀 Application )
│       │       │
│       │       ├── utils
│       │           ├── RippleEffect (Extra:)
│       │           ├── AccountManager (integration for Account, BankAccount,)
│       │
│       └── resources
│           └── account-details
│           │   └── username-details.txt
│           │   └── username-details.txt
│           │
│           └── images
│           │   └── your-image-file.png
│           │
│           └── voice-effect
│               └── voice-effect-file.wav
│
├── pom.xml (or build.gradle, etc.)
│
└── video
    ├── video 1.mp4
    ├── video 2.mp4
    └── video 3.mp4
```

Code Structure Explanation:

The code structure provided follows the "package-by-components" approach, where the code is organized based on the different components or features of the application. Here's a breakdown of the structure:

1. ArtificialLedger.java: The main class of the application.
2. src/main/java/your.package.structure: The root package for the Java source code.
   a. components: package for different components or interfaces used in the application.
   b. forms: package for different GUI forms or panels.
   c. main: package for the main application logic.
   d. utils: package for utility classes or helper methods.

- EventHomeOverlay: Interface for handling events in the home overlay.
- HeaderButton: Component for the header buttons in the home page.
- BankAccount: Integration for the account, account manager, and account details.
- Account: GUI form for the account page.
- Home: Homepage form.
- HomeOverlay: Overlay form for the home page.
- Login: Side panel form in the home page.
- ModelLocation: Extra component for background video.
- RegistrationForm: GUI form for the registration page.
- Main: The main application class.
- RippleEffect: Extra utility for ripple effect.
- AccountManager: Integration for the account, bank account, and account details.

3. src/main/resources: The resources directory for the application.
   a. account-details: Directory for account details text files.
   b. images: Directory for image files used in the application.
   c. voice-effect: Directory for voice effect files.
4. pom.xml: The Maven project configuration file.
5. video: Directory for video files.

The structure follows a modular approach, where each component or feature of the application is organized into separate packages. This makes it easier to manage and maintain the codebase, as well as promote code reusability and separation of concerns. Using Maven allows for easy project management and dependency resolution, making it convenient to build and manage the project. Overall, the structure provides a clear organization of the different components and resources of the application, making it easier to navigate and understand the codebase.

Features of the Program

- User Registration: Allows users to create an account and securely register their personal information.
- Account Management: Enables users to manage their bank accounts, including creating new accounts, viewing balances, and making transactions.
- Transaction History: Provides a detailed transaction history for each user, allowing them to track their financial activities.
- Security Measures: Implements robust security measures, such as encryption and authentication, to ensure the safety of user data.
- Create new bank accounts with unique account numbers:
- Deposit money into existing accounts:
- Withdraw money from existing accounts and with checks for sufficient balance:
- Check the current balance:
- Display the detailed information of an account:

- Registration Form’s fantastic background (video or animated style):
- Graphical User Interface for Login, Registration, Bank Account and more:
- Sound Effects for certain features:

PHASE 4: Developing Product

Development

1. Set up the development environment: Ensure that all developers have the necessary tools and software installed, including Java Development Kit (JDK), Integrated Development Environment (IDE) such as IntelliJ or Eclipse and Maven.
2. Create a project structure: Establish a directory structure that follows best practices for organizing Java projects. This typically includes separate directories for source code (src/main/java), resources (src/main/resources), and tests (src/test/java).
3. Define project dependencies: Use Maven pom.xml file to specify the projects dependencies. This includes any external libraries or frameworks that your project relies on. Maven will automatically download and manage these dependencies.
4. Start writing the actual Java code for this project. Follow the design specifications outlined in the Design Document Specification (DDS).

Installation

1. To run the the Artificial Ledger Bank locally, you will need the following:

    - Java 11 or higher, I prefer Java 22 LTS (latest version)
    - IDE: IntelliJ Idea Community Edition or Ultimate
    - VLC Media Player
    - Locate the designated important file such as video, resources file like wav, png, jpg in appropriate directory file in your idea folder located in your system drive: C:\Users\Anilov\IdeaProjects\ArtificialLedger\

Once you have the required tools installed, follow these steps to install the Artificial Ledger Bank System:

1. Clone this repository:
 Git clone: https://github.com/flexycode/CCPRGG2L_INTERMEDIATE_FINAL_EXAM.git
2. You can also download the file by Download ZIP or Open with GitHub Desktop.
3. Once the file downloaded, locate the directory file in your IdeaProjects folder located in your system drive: C:\Users\Anilov\IdeaProjects\ArtificialLedger\.
4. Open your IntelliJ IDE and click Plugins. Check the box if the Maven is Installed in the Build Tools. Once finish open the Project file located on your IdeaProject folder.
5. After you open your file, click the Main Menu or press Alt + \ then select Project Structure or Ctrl + Alt + Shift + S.
6. In the Project Settings > Project: Make sure to select the best possible SDK for this project. Openjdk-22 Oracle OpenJDK 22.0.1 (currently the latest version)
In Language Level: you can leave it as default.
7. Configure the Dependencies in the Modules section.

Coding Standard

1. Implement functionality: Begin implementing the desired functionality of the ALT Bank System. This involves writing the necessary algorithms, logic, and business rules using Java programming language. Follow coding best practices, such as writing clean and modular code, using appropriate naming conventions and adding comments for clarity.
2. Test your code: Write unit tests to verify the correctness of your code. Use testing frameworks like JUnit to create test cases that cover various scenarios and edge cases. Run these tests regularly to catch and fix any bugs or issues early in the

development process. Understand more on this in Phase 5: Product Testing and Integration for more actual code testing using IntelliJ IDE.

Scalable Code

1. Design for scalability: Start by designing the ALT Bank application with scalability in mind. Identify potential areas that may need to handle increased workload or data volumes in the future. Consider factors such as performance optimization, modularity and distributed architecture.
2. Use appropriate data structures and algorithms: Choose data structures and algorithms that are efficient and scalable. Optimize your code for time and space complexity, ensuring that it can handle larger datasets without significant performance degradation.
3. Modularize your codebase: Break down your code into smaller, manageable modules. Use Maven's project structure to organize your code into separate modules or components. This allows for independent development, testing and deployment of different parts of Artificial Ledger Bank application and making it easier to scale and maintain.
4. Continuously optimize and refactor: Regularly review and optimize your codebase to improve performance and scalability. Refactor code to eliminate bottlenecks, reduce dependencies, and improve modularity. Use profiling tools to identify performance hotspots and optimize critical sections of your code.

Version Control

1. Git: Use a version control system like Git to manage your codebase. Create branches for different features or bug fixes and merge them back to the main branch (e.g., master) once they are tested and ready.
2. GitHub: web-based platform that provides version control functionality. Best option for collaboration, code reviews and seamless integration of changes.

Code Review

1. Documentation: Document your code, including class and method descriptions, API references and any important design decisions.
2. GitHub Issues & Project Planning: Create issues, break them into task, track some progress, add custom fields, and have team conversations. Visualize Artificial Ledger Bank system projects as tables, boards or roadmaps and automate everything with code. Tackle complex issues with task lists and track their status with new progress indicators. Convert tasks into their own issues and navigate your work hierarchy.

PHASE 5: Product Testing and Integration

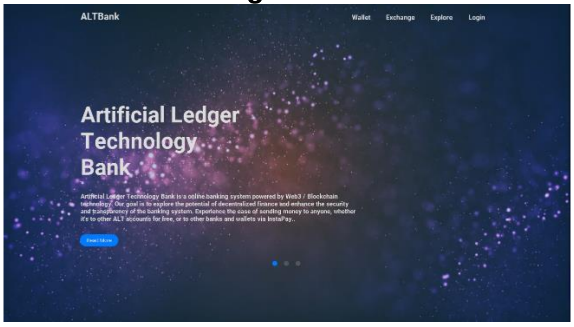
Program Requirements

1. Secure the Username and Password
2. Set an Exception Handling for Errors

Project Testing Steps

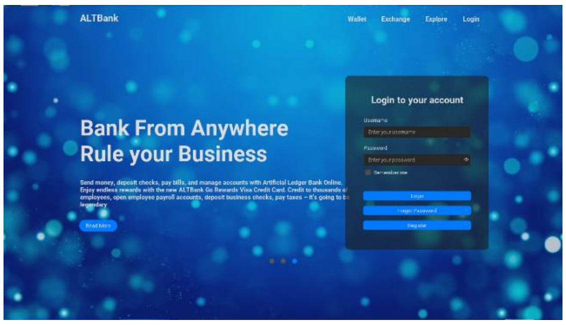
1. Tests will be done with Intellij and will run the program with "Run with Coverage".
2. Program must meet all program requirements.
3. Run the program and check for bugs.
4. Tester will run three different accounts for fool-proofing.
5. Program must hit the 83% benchmark to confirm the program is bug-free and ready for deployment.

Functional Testing



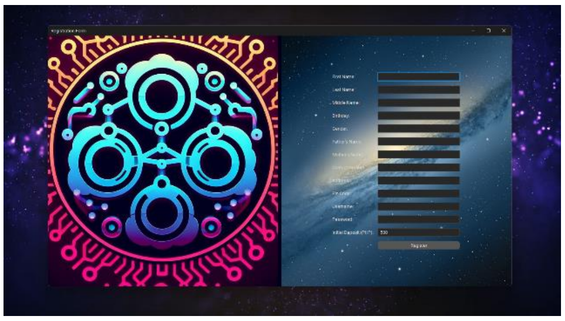
Artificial Ledger Technology Bank is a online banking system powered by Web3 / Blockchain technology. Our goal is to explore the potential of decentralized finance and enhance the security and transparency of the banking system. Experience the ease of sending money to anyone, whether it's to other ALT accounts for free, or to other banks and wallets via InstaPay..

Main Window

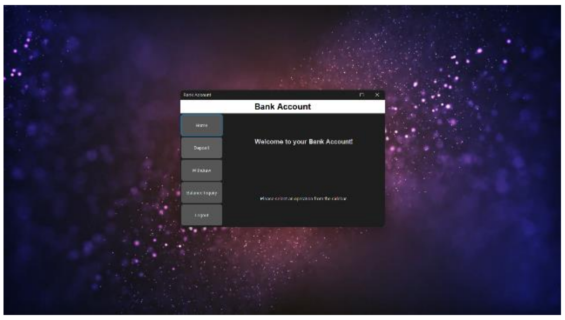


Send money, deposit checks, pay bills, and manage accounts with Artificial Ledger Bank Online. Enjoy endless rewards with the new ALTBank Go Rewards Visa Credit Card. Credit to thousands of employees, open employee payroll accounts, deposit business checks, pay taxes – It's going to be legendary

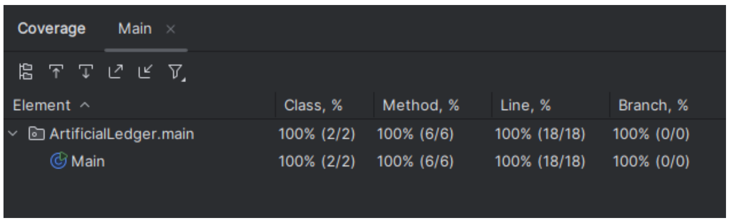
Login Window



Registration Form

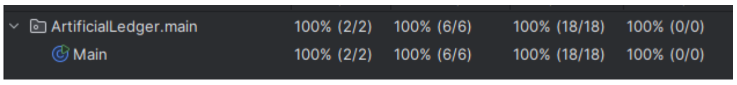


Transaction Window



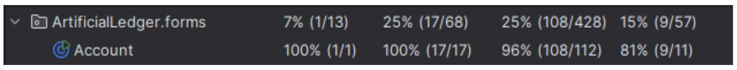
Element	Class, %	Method, %	Line, %	Branch, %
ArtificialLedger.main	100% (2/2)	100% (6/6)	100% (18/18)	100% (0/0)
Main	100% (2/2)	100% (6/6)	100% (18/18)	100% (0/0)

Main Test 1



ArtificialLedger.main	100% (2/2)	100% (6/6)	100% (18/18)	100% (0/0)
Main	100% (2/2)	100% (6/6)	100% (18/18)	100% (0/0)

Main Test 2



ArtificialLedger.forms	7% (1/13)	25% (17/68)	25% (108/428)	15% (9/57)
Account	100% (1/1)	100% (17/17)	96% (108/112)	81% (9/11)

Account Test

1. The program passed Functional Testing. All Functions and windows are working as intended.
2. Based on the Test Results, the program exceeds the 83% benchmark. The program is free of bugs and meets all program requirements.

PHASE 6: Deployment and Maintenance of Product

Development and Maintenance

1. Continuously develop and maintain your Artificial Ledger Bank application based on user requirements and feedback.
2. We use agile development methodologies, such as Teams for meeting and Kanban for managing and prioritizing development tasks.
3. We use GitHub Features such as Kanban board to collaborate with the development team to implement new features, fix bugs, and improve the overall functionality of the application.

Release Planning

1. Plan and schedule releases of your Artificial Ledger Bank application.
2. Define the scope of each release, including the features and enhancements to be included.
3. Prioritize and sequence the release plan based on project deadline priorities and user needs.
4. Consider factors such as risk assessment, resource availability, and dependencies.

Deployment Automation

1. Implement deployment automation to streamline the process of deploying your Artificial Ledger Bank application.
2. Use tools like Jenkins, Ansible, or Docker to automate the build, testing, and deployment of your application.
3. Create deployment scripts or configuration files that can be easily executed to deploy the application to different environments, such as development, staging, and production.

Maintenance

1. Establish a maintenance process to address any issues or bugs that arise after the deployment of your Artificial Ledger Bank application.
2. Monitor the application's performance, availability, and security to proactively identify and resolve any issues.
3. Regularly apply updates, patches, and security fixes to keep the application up-to-date and secure.
4. Document and track maintenance activities, including bug fixes, enhancements, and optimizations.

Feedback

In this section, this would be our project presentation for Professor Jay Abaleta but in real world project scenario you need to:

1. Gather feedback from users, stakeholders, and support teams regarding the deployed Artificial Ledger Bank application.
2. Set up channels for users to report issues, provide suggestions, and share their experience with the application.
3. Analyze and prioritize feedback to identify areas for improvement and plan future updates or releases.
4. Use the feedback to drive continuous improvement and refine the application based on user needs and expectations.

By following these steps, you can effectively manage the Deployment and Maintenance phase of our Artificial Ledger Bank project. This ensures a smooth deployment process, ongoing maintenance, and continuous improvement of the application based on user feedback and evolving requirements.