Digital ATM

Contact Info

Nermin Dedovic	ndedovic@depaul.edu
Justin MacIsaac	jmacisaa@depaul.edu
Jessica Solis	jsolis4@depaul.edu

Project Specifications

A. Context

Developing a new software project from scratch can be a daunting task that involves many hours of work, several different contributors, complex design choices, and state of the art hardware. In order to distribute a successful product, the project must be well designed.

The main determination in a well designed product is the quality of the source code. “Writing clean code requires the disciplined use of a myriad little techniques applied through a painstakingly acquired sense of “cleanliness.”[1]. Ensuring that the project has good design allows for internal team understanding, future additional features, transfer of responsibility, thorough testing, and most importantly, a successful product. Generally, a team can follow good practices and take advantage of useful tools to check the quality of their code throughout the design process. These processes include refactoring, code smells, and determining quality metrics such as CK-Metrics.

In this project we will be developing a new project from scratch and incrementally checking the quality of our code with tools such as Understand and Sonar to guarantee we are developing a well designed product. As the quality is measured, the code will be refactored and improved to deliver a polished final version. The final version will be tested intensely to ensure functionality, efficiency, and robustness.

B. Motivations/Problem Statement

Relative to the requirements gathering or implementation phases of the software development cycle, testing is critical and challenging yet a time-consuming and expensive phase. However, lack of well-designed testing scenarios can lead to significant failures if the defects that testing is meant to identify are not found. Testing is not debugging or quality assurance, and according to the IEEE, “software quality can be defined as the capability of a software product to satisfy stated and implied needs under specified conditions…Quality refers to the degree to which software products meet their stated requirement.”[2]. While testing cannot capture every weakness in a software product, testing does ensure that the software is as reliable, maintainable, efficient, and easy to use as it could be.

In order to conduct software testing, metrics such as number of children, coupling between classes, and comment percentage are used to analyze from which one can draw conclusions. We will be using JUnit to conduct both unit and integration testing. Doing this will allow us to detect anti-patterns/code smells to refactor and improve the software product.

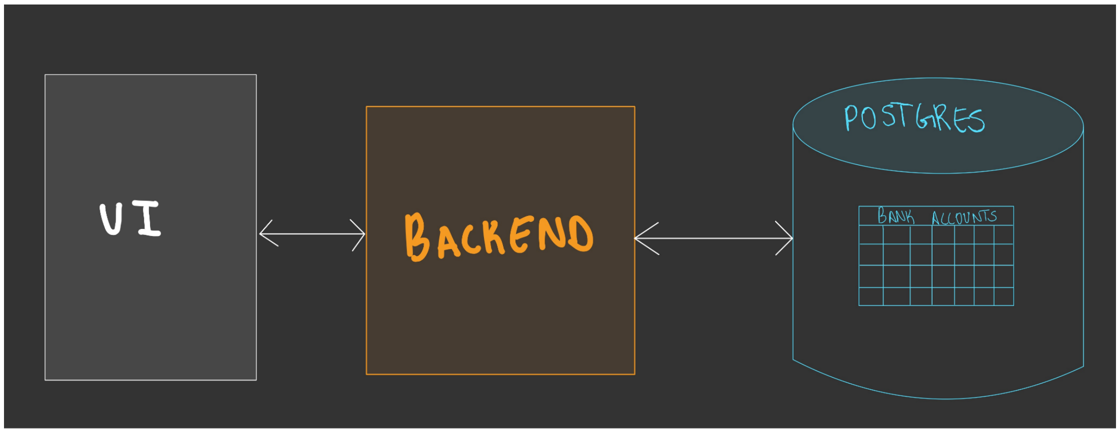
C. Expected contributions(Solution) and Objectives

Expected contributions (Solution)

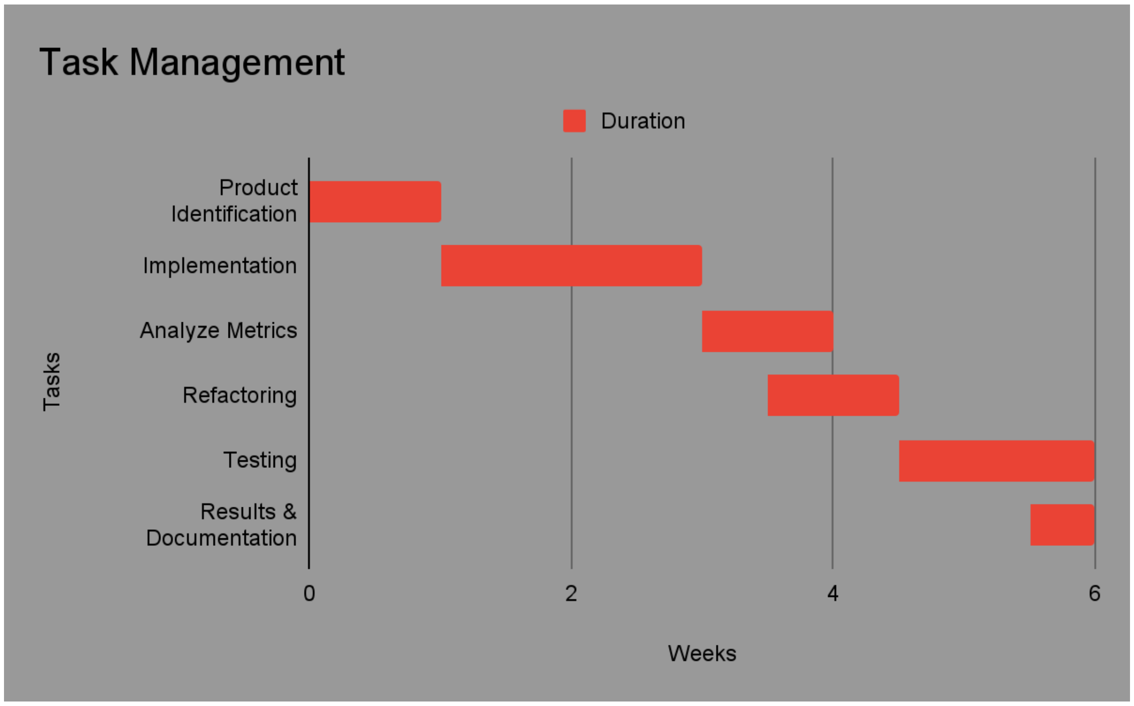
- Java/Spring application with Maven as our package manager
- UI representing a digital ATM. Users with an account will use the UI to make deposits and withdrawals from their account
- Basic login authentication
- Backend implementation of the digital ATM will ensure user input is valid before making any potential changes to an account
- Postgres database with account and balance information. Backend application will leverage the database for persisting/retrieving user data
- Docker-compose file with prepopulated Postgres database for ease of use / demo
- Unit testing project components with JUnit5 and Mockito
- Integration testing: database communication with backend logic. We will leverage an embedded H2 database to imitate Postgres functionality
- Leverage Understand and Sonar to analyze quality software metrics and detect error-prone code

Objectives:

- Clean and well-documented code
- Minimal code smells
- Application should be maintainable and have minimal complexity.
- No unnecessary coupling
- Application should be easy to change and add new features to (pragmatic)
- Application should be fully tested and have quality code coverage through unit/integration tests
- Configuration for the database should be pluggable / easy to change
- Application should be well-designed, easy to use, and efficiently replicate the functionality of an ATM



D. Task management



E. References

[1] Robert C. Martin. (2008). Clean Code, A Handbook of Agile Software Craftsmanship. Pearson
[2] “Software Quality Course: IEEE Computer Society.” *IEEE Advancing Technology for Humanity*, https://www.computer.org/product/education/software-quality-course.