CHEM274B Final Project - Delivery II

Software Engineering Reflection
Name: Priscilla Vaskez
Group: Group 6
Team Members: Trinity Ho, Dongwan Kim, Priscilla Vaskez

1. **Description of My Role in the Project**

   My contribution to this project both covered design and core functionality across. I implemented the transfer method in Level 1, this handled money movement between accounts and checked for errors like self-transfers and insufficient funds. I used methods that were taught in this course to implement the level 1 function. In level 4, I implemented merge_accounts, a more advanced feature that allowed account consolidation while keeping the history and supporting future balance queries through alias resolution. In addition I created the UML diagram for all four levels on this project. I created the UML by using Lucid Charts. I implemented all 4 levels to the UML in order to help clarify our system's structure and relationship across levels.
   I also contributed to reviewing the code and README files to ensure that sufficient comments were added and all was documented. I also contributed to the Git repository ensuring things were merged and if conflict were created they were resolved.

2. **Contribution to Successful Project Completion**

   I contributed to feature implementation and team collaboration. This includes:
   - Developing key methods: transfer(Level 1) and merge_accounts (Level 4).
   - Designing the UML diagram that documents all classes and relationships across project levels.
   - Contributed by final reviews of code and README file.
   - Contributed to testing the code to make sure it functions properly.
   - Participating in meeting and maintaining good team communication around every step of the project.

3. **Challenges and How They were Solved**

One challenge was handling the complexity of merged accounts in Level 4. It was a little difficult for me to piece together the current account identities, balance histories, and how to support consistent behavior before and after merges. I reviewed examples, walked through edge cases manually, and built in logic for alias resolution and timestamp-based filtering. Another challenge we encountered was debugging across independently developed features. Since team members worked on separate functions in parallel some integration issues only became clear only when we began testing the code. For example Level 3's cashback logic

needed to be processed before Level 4 balance queries to return the correct results. We resolve this by carefully tracing method dependencies, discussing expected behaviors as a team, and adjusting the call order of the functions. I also encountered a Git merge conflict due to uploading a branch on an outdated main, this taught me to always remember while collaborating to double check before I push changes and update my local repository first.

4. Algorithmic and Performance Analysis of Each Method

- **transfer**
  Time: O(1) — This function does constant-time checks and dictionary updates.
- **merge_accounts**
  Time: O($m_1 + m_2$ + log n) — This function does merges histories and payment data; sorts combined record.
- **_record_balance**
  Time: O(1) — This function is simply appended to a list of (timestamp, balance) pairs.
- **get_balance**
  Time: O(log n) — This function uses binary search for efficient retrieval of balance at time_at.
- **_binary_search_record**
  Time: O(log n) — This function finds the latest applicable balance efficiently.

5. What Could Have Been Done Differently

Software Project Management

We could have improved project workflow by establishing a consistent branching strategy and integration schedule from the start. Some features were built in isolation and needed adjustments during merging. More regular syncs or early automated test runs would have helped catch this earlier.
We also could have written some of the docstrings and comments progressively instead of waiting until the final pass. That may have saved time and improved readability during development.

Final Product Improvements

If time allowed, we might improve the following:

- Refactor long methods like merge_accounts into smaller helper functions.
- Replace None returns in failure cases with custom error types for better traceability.