Explanation

By default, relationships in JPA like `@OneToMany`, `@ManyToOne`, etc., are **lazy-loaded** unless you specify `fetch = FetchType.EAGER`. Lazy loading means that related entities are not loaded from the database **until you access them**.

Problem Without `@Transactional`:

Without an active transaction:

- When you fetch an entity (e.g., `Customer`) with lazy-loaded associations (e.g., `accounts`), and then the persistence context is closed (which happens at the end of the method call if there is no transaction),
- Any attempt to access a method outside that method results in a `LazyInitializationException` because the entity manager is already closed and can no longer fetch the associated data.

With `@Transactional`:

- When the `AccountService` class is annotated with `@Transactional`, Spring ensures that for every method:
    - A **transaction is started**.
    - A **Hibernate session (persistence context)** is opened and remains alive for the **entire method execution**.
- As long as you are inside that method, **lazy-loaded properties can be accessed**, because the session is still active.
- This makes it safe to use lazy-loading without forcing eager-loading.