Cloud Shop Assignment Report

R12945037 卓均而

Executed Environment

The application was developed and executed in the following environment:

- **Operating System**: Linux, macOS, or Windows (cross-platform)
- **Python Version**: Python 3.x (recommended version: 3.7 or higher)
- **SQLite**: The database is built-in with Python and does not require external installations.
- **Libraries**:
  - `sqlite3`: Used for database interactions (built into Python).
  - `argparse`: Used for command-line argument parsing.
  - `re`: For regular expressions to parse the command inputs.

Package & Library

The following packages and libraries were used in the project:

1. **sqlite3**: A lightweight database library that is part of the Python standard library.
2. **argparse**: A standard Python library for parsing command-line arguments, used in this project to handle file input for batch processing of commands.
3. **re**: The regular expression library used to validate and extract data from commands (e.g., quoted strings in `CREATE_LISTING`).

No external dependencies were needed, making the application easy to deploy and run in any Python environment.

Codebase Description (Architecture)

This codebase follows a **modular architecture** based on the **Onion Architecture**. The application is split into three main layers:

1. **CLI Layer (User Interface)**:
   - The `CLIHandler` class in the `main.py` file is responsible for interacting with the user through the command line. It processes user inputs, verifies commands, and routes them to the appropriate service functions.
   - The command parsing in `CLIHandler` uses regular expressions to capture data from commands and then calls the service layer to perform the necessary actions.
2. **Service Layer**:

- The `MarketplaceService` class is the core business logic layer. It decouples the `CLIHandler` from the repository. The service layer processes the requests (e.g., registering a user, creating a listing) and returns appropriate responses.
- The service layer also validates and organizes data before passing it to the repository layer.

3. **Repository Layer (Data Access)**:
   - The `MarketplaceRepository` class interacts directly with the SQLite database, performing CRUD operations (Create, Read, Update, Delete) on the users and listings tables.
   - This layer is responsible for abstracting all database-related logic, such as checking if a user exists, adding new listings, deleting listings, and fetching data from the database.
   - The repository layer ensures **data integrity** and **separation of concerns** between the business logic and data storage.

Database Design

The application uses a simple relational database (`SQLite`), with two tables:

1. **Users Table**:
   - Stores registered users with a unique, case-insensitive username.
   - Primary Key: `username`.
2. **Listings Table**:
   - Stores marketplace listings created by users.
   - Primary Key: `id` (automatically incremented).
   - Foreign Key: `username` (references the `users` table).
   - Other columns include: `title`, `description`, `price`, `category`, and `created_at`.

Onion Architecture Overview

The **Onion Architecture** is applied to ensure that:

- **Business logic** (service layer) is independent of infrastructure (repository and CLI layers).
- The **core logic** (service) is not tightly coupled with external systems (like the database or user interface).