CLI Task Manager

This project is a command-line task management application built with Python. It allows users to manage tasks directly from the terminal. You can add, list, update, mark as done or undone, delete, clear completed, and purge all tasks using simple commands. The app uses SQLite for persistent storage and is organized for maintainability and extensibility.

This project uses **Poetry** for dependency management and packaging, making it easy to install, update, and manage Python packages. **Pytest** is integrated for running unit tests to ensure code reliability and correctness.

Project Structure

```
cli-taskmanager/
├── pyproject.toml
├── README.md
├── src/
│   └── cli_taskmanager/
│       ├── app.py              # Main entry point for the CLI
application
│       ├── cli/
│       │   └── commands.py       # CLI command definitions and argument
parsing
│       ├── controller/
│       │   └── task_service.py   # Business logic for managing tasks
│       ├── model/
│       │   └── task.py           # Task data model
│       ├── storage/
│       │   ├── schema.sql        # SQL schema for the SQLite database
│       │   └── sqlite_storage.py # Database interaction and persistence
│       └── __init__.py           # Package initialization
│       └── tests/
│           ├── test_cli.py           # Tests for CLI commands
│           ├── test_storage.py       # Tests for storage layer
│           ├── test_task_service.py  # Tests for business logic
│           └── __init__.py           # Test package initialization
```

Main Components

- **app.py**: Launches the CLI and connects commands to the controller.
- **cli/commands.py**: Defines available CLI commands (add, list, update, delete) and handles user input.
- **controller/task_service.py**: Implements the core logic for task operations, such as creating, updating, and deleting tasks.
- **model/task.py**: Defines the Task class and its attributes.
- **storage/sqlite_storage.py**: Handles all interactions with the SQLite database, including CRUD operations.
- **storage/schema.sql**: Contains the SQL schema for initializing the database.

- **tests/**: Contains unit tests for CLI, storage, and service layers to ensure reliability.

How to Use

1. **Install dependencies**:

   ```
   poetry install
   ```

2. **Run the application**:

   ```
   poetry run python src/cli_taskmanager/app.py
   ```

3. **Available commands**:

   - `add`: Add a new task
   - `done`: Mark a task as done
   - `undone`: Mark a task as not done
   - `del`: Delete a task
   - `clear`: Clear all completed tasks
   - `ls`: List tasks (with status filter)
   - `upd`: Update a task's description
   - `purge`: Delete all tasks

Extending the App

- Add new commands in `cli/commands.py`.
- Update business logic in `controller/task_service.py`.
- Modify the data model in `model/task.py`.
- Change storage backend in `storage/sqlite_storage.py`.

License

MIT