# Task Manager Documentation

**Project Overview**

**Task Manager** is a Flask application designed for managing tasks with user authentication. Users can register, log in, and perform CRUD operations on their tasks.

**Installation**

1. **Clone the Repository:**
   * Clone the project repository from GitHub.
   * Navigate into the project directory.
2. **Set Up Virtual Environment:**
   * Create a virtual environment.
   * Activate the virtual environment.
3. **Install Dependencies:**
   * Install the required Python packages using a requirements file.
4. **Initialize Database:**
   * Run database migrations to set up the initial database schema.
5. **Run the Application:**
   * Start the Flask development server.

**Configuration**

* **SECRET\_KEY**: Used for securing sessions and cookies.
* **SQLALCHEMY\_DATABASE\_URI**: Specifies the database to use (e.g., SQLite).

**Usage**

* **Home Page** (/): View and manage tasks.
* **Register** (/register): Create a new user account.
* **Login** (/login): Log in to your account.
* **Logout** (/logout): Log out of your account.
* **Add Task** (/add): Add a new task.
* **Edit Task** (/edit/<task\_id>): Edit an existing task.
* **Delete Task** (/delete/<task\_id>): Delete a task.
* **About Page** (/about): View project information.

**Models**

**User**

* **ID**: Unique identifier for the user.
* **Username**: Unique username for login.
* **Password**: Securely stored password.
* **Tasks**: Relationship with tasks assigned to the user.

**Task**

* **ID**: Unique identifier for the task.
* **Content**: Main text or title of the task.
* **Description**: Detailed information about the task.
* **Priority**: Task priority level.
* **Due Date**: Deadline for the task.
* **Timestamp**: Time when the task was created.
* **User ID**: Reference to the user who created the task.

**Extensions**

* **Flask-SQLAlchemy**: Manages database interactions using SQLAlchemy.
* **Flask-Login**: Handles user authentication and session management.
* **Flask-Migrate**: Manages database schema changes and migrations.

**Testing**

* **Run the Application**: Ensure the application is running in development mode.
* **Navigate and Test**: Check the functionality of different pages and features.

For automated testing, use frameworks like pytest or unittest.

**Contributing**

1. **Fork the Repository**: Create a copy of the project repository.
2. **Create a Branch**: Make a new branch for your changes.
3. **Make Changes**: Implement your improvements or features.
4. **Commit Changes**: Save your changes with a descriptive commit message.
5. **Push Changes**: Push your branch to the remote repository.
6. **Create a Pull Request**: Submit a pull request for review and merging.