Разработать простое консольное приложение для регистрации и авторизации пользователей, используя паттерн Model-View-Controller

1. Структура проекта

```
auth_system/
├── models/          # Модели данных
│   ├── __init__.py
│   └── user.py      # Модель пользователя
├── controllers/     # Контроллеры
│   ├── __init__.py
│   └── auth_controller.py # Контроллер авторизации
├── views/           # Представления
│   ├── __init__.py
│   └── auth_view.py # Консольный интерфейс
├── storage/         # Хранение данных
│   ├── __init__.py
│   └── user_storage.py # Работа с данными пользователей
└── app.py           # Главный файл приложения
```

2. Функционал

Модель (User):

- Хранит информацию о пользователе: id, username, email, password (хэшированный)
- Методы для проверки пароля, сериализации/десериализации

Хранилище (UserStorage):

- Сохраняет/загружает пользователей в файл (JSON)
- Методы: add_user, get_user_by_username, get_all_users

Контроллер (AuthController):

- Обрабатывает логику регистрации и авторизации
- Методы: register, login, logout, is_authenticated

Представление (AuthView):

- Консольный интерфейс с меню:
    1. Регистрация
    2. Авторизация
    3. Выход
- Отображает сообщения об ошибках/успехе

3. Дополнительные требования

- Использовать хэширование паролей (библиотека hashlib)
- Валидация данных (длина username/email, сложность пароля)
- Обработка ошибок (пользователь уже существует, неверные данные и т.д.)
- При регистрации создавать пользователя с уникальным ID
-

Пример работы приложения

Добро пожаловать в систему авторизации!

1. Регистрация
2. Авторизация
3. Выход
> 1

Введите username: testuser
Введите email: test@example.com
Введите password: ********

Регистрация прошла успешно!

1. Регистрация
2. Авторизация
3. Выход
> 2

Введите username: testuser
Введите password: ********

Добро пожаловать, testuser!