Разработать систему аутентификации и управления аккаунтом пользователя с использованием Express.js, MongoDB через Mongoose, Passport.js для аутентификации и Joi для валидации данных.

Цели задания:

- Разработать REST API для аутентификации и управления аккаунтом пользователя.
- Использовать MongoDB и Mongoose для работы с базой данных.
- Применить Passport.js для реализации аутентификации.
- Использовать Joi для валидации входящих данных на сервере.

Задачи:

1. Настройка проекта:

- Инициализируйте новый проект с использованием NPM.
- Установите необходимые зависимости: express, mongoose, passport, passport-local, jsonwebtoken, bcryptjs, joi, dotenv, express-session.
- Настройте подключение к MongoDB с использованием Mongoose.
- Настройте переменные окружения для безопасного хранения конфигурационной информации (например, строки подключения к базе данных, секреты для токенов).

2. Модели данных:

- Создайте схему пользователя с полями: имя пользователя, электронная почта, хешированный пароль.

3. Валидация данных:

- Используйте Joi для создания схем валидации для регистрации и входа пользователей.

4. Аутентификация:

- Реализуйте маршруты для регистрации (`/register`) и входа (`/login`) пользователей с использованием Passport.js.
- Используйте bcrypt для хеширования паролей перед их сохранением в базу данных.
- При успешной аутентификации возвращайте JWT пользователю.

5. Маршруты для управления аккаунтом:

- `/logout` — выход пользователя из системы.
- `/profile` — получение информации о профиле пользователя. Доступен только для аутентифицированных пользователей.
- `/updatePassword` — обновление пароля пользователя. Доступен только для аутентифицированных пользователей.
- `/updateInfo` — обновление информации пользователя (например, имени пользователя и электронной почты). Доступен только для аутентифицированных пользователей.

6. Авторизация:

- Добавьте middleware для проверки JWT в запросах к защищённым маршрутам (`/profile`, `/updatePassword`, `/updateInfo`, `/logout`).

Требования к выполнению:

- Все маршруты должны корректно обрабатывать успешные и ошибочные сценарии, возвращая соответствующие HTTP статусы и сообщения об ошибках.
- Пароли пользователей должны храниться в базе данных в зашифрованном виде.

- JWT должны использоваться для аутентификации на защищённых маршрутах.
- Входные данные должны проходить валидацию с использованием Joi перед обработкой на сервере.

Дополнительное задание:

- Реализуйте функционал для восстановления забытого пароля, отправив пользователю ссылку на электронную почту для сброса пароля.