# FastAPI and Pydantic Exercise: Building a Secure Todo List API

In this exercise, you will build a Todo List API using FastAPI and Pydantic. The API will include authentication and authorization functionalities to ensure secure access to resources. You will progressively enhance the functionality and security of the API as you complete the exercise.

## Part 1: Pydantic Basics

1. Build a Pydantic model called `User` with the following fields:
    - `username` (string): A username with a minimum length of 3 characters.
    - `email` (string): A valid email address.
    - `password` (string): A password with a minimum length of 8 characters.
    - `disabled` (boolean): Indicates whether the user is disabled or not.
2. Implement basic validation for the fields of the `User` model. Ensure that the username meets the minimum length requirement, the email is a valid email address, and the password meets the minimum length requirement.

## Part 2: Basic Authentication

1. Create a fake database (`fakeDB`) containing some existing users as a dictionary. Each user should have a unique username and a hashed password.
2. Based on the JWT code example provided, create an endpoint `/login` that allows users to authenticate and obtain an access token. If any more functions or capabilities need to be taken from the JWT code example, use them.

## Part 3: Basic Authorization

1. Create a Pydantic model called `Todo` representing a todo item. It should have the following fields:
    - `id` (integer): Unique identifier for each todo item.
    - `title` (string): Title of the todo item.
    - `description` Optional(string): Description of the todo item.
    - `owner` (string): Username of the user who owns the todo item.
    - `completed` (boolean): Indicates whether the todo item is completed or not.
2. Implement the following endpoints for authenticated users:

- `/todos` (POST): Allows users to create a new todo item. The user should provide the `title` and `description` in the request body. The endpoint should associate the todo item with the authenticated user.
- `/todos` (GET): Retrieves a list of all todo items owned by the authenticated user.
- `/todos/{todo_id}` (GET): Retrieves a specific todo owned by the authenticated user. Validate only the owner of the todo can access a todo via the endpoint.

## Part 4: Deeper Pydantic + Authentication

1. Expand the `User` model with additional fields that allow users to register:
   - `confirm_password`(String) : a string that must match the password field.
   - `created_time`(datetime) : defaults to now - read about Field(default_factory)
   - `birthday`(str->datetime) :should be provided in one of the following formats ("19/07/1983" or "19-07-1983"). validate it fits the structure and turn it into a datetime object. i.e. - the input is a string but the field is a datetime.
2. Add more validation rules to the newly added fields in the `User` model. Validate on user creation that the 2 provided passwords match, and that the birthday was given in the correct formats.
3. Create an endpoint `/register` that will recieve the relevant information and create a new user in the `fakeDB`.
4. Do we always need the `confirm_password` field or is it only relevant for the `/register` endpoint? Try to think of a way to make sure users saved in `fakeDB` only have their necessary data.
5. add an `age` computed property based on the user's `birthday`.

## Part 5: Deeper Authorization

1. Add the following endpoints for authenticated users:
   - `/todos/{todo_id}` (PUT): Allows users to update their own todo items by providing the updated `title`, `description`, and `completed` status in the request body.
   - `/todos/{todo_id}` (DELETE): Allows users to delete their own todo items.
2. Add an `is_admin` field to the `User` model. Users with `is_admin` set to `True` should have additional privileges.

3. Implement the following endpoints for administrators:
    - `/todos/all` (GET): Retrieves a list of all todo items owned by all users.
    - Also, admins can perform the regular endpoints (CRUD) for all users.