

A detailed explanation of all project files

* **Main project files**

1. main.py (startup point)

**Function:** The main file that runs the application.

**Importance:** Connects the user interface (CLI) and core functionality (authentication + tasks).

**Content:**

ToDoApp class: Manages the application lifecycle.

|  |  |
| --- | --- |
| run() | Manages the main menu. |
| show\_unauth\_menu() | Displays the login/login menu. |
| show\_auth\_menu() | Displays the menu after login. |
| task\_management() | Manages the task list. |

2. auth.py (User Management)

**Function:** Handles login, registration, and account updates.

**Importance:**

* Secures access to the application through identity verification.
* Encrypts passwords before saving them (bcrypt).

**Content:**

AuthManager class:

|  |  |
| --- | --- |
| register(): | Registers a new user after data verification. |
| login(): | Logs the user in after matching the password. |
| update\_profile() (incomplete): | Modifies user data. |



3. manager.py (Task Management)

**Function:** Creates, displays, and edits tasks.

**Importance:** The functional core of the application (managing daily tasks).

**Content**: TaskManager class:

|  |  |
| --- | --- |
| create\_task(): | Creates a new task with date validation. |
| view\_tasks(): | Displays the user's tasks. |
| update\_profile() (incomplete): | Modifies user data. |

Other functions (update\_task, delete\_task) are incomplete.

* **Data Files and Models**

4. models.py (Data Models)

**Function:** Contains definitions for user and task classes.

**Importance:**

* Handles reading/writing data to/from files.
* Separates logic (AuthManager/TaskManager) from storage.

**Content**

User Class:

Stores user data (name, email, encrypted password, etc.)

|  |  |
| --- | --- |
| to\_dict(): | Converts data to JSON for saving. |

UserRepository Class:

|  |  |
| --- | --- |
| save\_users(): | Saves users in users.json. |
| find\_user\_by\_email(): | Searches for a user by email. |

Task and TaskRepository Class:

Same functionality but for tasks (stored in tasks.json).



5. validation.py (Data Validation)

**Function:** Validates inputs (names, email addresses, passwords, etc.).

**Importance:** Prevents invalid data from being entered into the system.

**Content:** Validator class:

|  |  |
| --- | --- |
| validate\_name(): | Verifies that the name is written in English only. |
| validate\_email(): | Verifies the email format. |
| validate\_phone(): | Verifies a valid Egyptian phone number. |
| validate\_password(): | Verifies the strength of the password. |
| hash\_password(): | Encrypts the password before saving it. |

* **Storage Files (JSON)**

6. users. json and tasks.json

**Function:** Storing user and task data.

**Importance:** A simple alternative to a database (used for small applications).

**Content:** Arrays [] are filled with JSON objects when registering or creating tasks.

Example:

[

{

"first\_name": "Mohamed",

"email": "mohamed@example.com",

"password": "$2b$12$..."

}

]



**Release Features:**

* Professional project structure with functionality separated into different packages
* Comprehensive error handling for every part of the application
* Full functionality covering all project requirements
* Easy-to-use interface with clear messages
* Task reminders for near and late dates
* Security enhancements such as:
* Password encryption
* Validation of all inputs
* Easily expandable to add new features