Algorithm

*Name: Kien Luu*

**Classes**

1. **User Class**:
   * \_\_init\_\_(name, email, password, role, uid): Initializes a user with basic attributes.
   * check\_password(input\_password): Verifies if the entered password matches the stored one.
   * to\_dict(): Converts the user's data into a dictionary format for CSV storage.
   * from\_dict(data): Creates a User object from a dictionary (used for loading data).
2. **Patient Class** (inherits from User):
   * \_\_init\_\_(name, email, password, uid, conditions, prescriptions): Initializes patient-specific attributes.
   * update\_condition(condition): Adds a new medical condition to the patient's record.
   * update\_prescription(prescription): Adds a new prescription to the patient's record.
   * to\_dict(): Converts the patient's data (including conditions and prescriptions) into a dictionary for CSV storage.
3. **Admin Class** (inherits from User):
   * \_\_init\_\_(name, email, password, uid): Initializes admin-specific attributes.
   * reset\_password(user, new\_password): Resets a patient's password to a new value.
   * update\_condition(patient, condition): Allows admin to add or edit a patient's condition.
   * update\_prescription(patient, prescription): Allows admin to add or edit a patient's prescription.
4. **HealthCareApp Class**:
   * \_\_init\_\_(self): Initializes the app, loads user data, and shows the login screen.
   * load\_users(): Loads all users from a CSV file.
   * save\_users(): Saves all users to a CSV file after changes.
   * find\_user(email): Finds a user by their email address.
   * show\_login\_screen(): Displays the login screen for user authentication.
   * show\_admin\_screen(): Displays the admin dashboard where they can manage patients.
   * show\_patient\_screen(patient): Displays the patient dashboard where they can view and update their information.
   * handle\_register(): Handles the registration process of a new patient.
   * handle\_login(): Handles the login authentication for patients and admins.
   * show\_register\_screen(): Shows the registration screen for new patients.

**Chart:**

| **Input** | **Process** | **Output** |
| --- | --- | --- |
| User input (email, password) | Check user credentials against the database. | Display login success or failure. |
| New patient data (name, email, password, etc.) | Register new patient by saving their information to CSV. | Show success message or error message. |
| Admin actions (update conditions, reset password) | Admin modifies patient data (e.g., conditions, prescriptions). | Display updated information. |
| Patient input (new condition, prescription) | Patient updates their medical records (conditions, prescriptions). | Show success message or error message. |
| Admin selects patient (email) | Admin manages the selected patient's information (edit, delete). | Update patient data and show feedback. |

**Variable Chart**

| **Variable Name** | **Description** |
| --- | --- |
| self.users | List that stores all user objects (patients and admins). |
| self.current\_user | Holds the currently logged-in user (either patient or admin). |
| USER\_FILE | Path to the CSV file where user data is stored. |
| self.users\_data | Dictionary containing user data to be read or written to/from the CSV file. |
| self.email\_entry | Tkinter entry widget for email input. |
| self.password\_entry | Tkinter entry widget for password input. |
| self.patient\_conditions | List that stores medical conditions for a patient. |
| self.patient\_prescriptions | List that stores prescriptions for a patient. |