**Documentation for the Code**

**Overview**

This program facilitates a basic hospital management system where employees and patients can interact with the system to access or manage data. It includes modules for doctors, pharmacies, vitals, insurance, and patient information access.

The system includes three primary functionalities:

1. **Employee Portal (E):** Allows employees to log in and manage their respective department data.
2. **Patient Portal (P):** Enables patients to access their medical information securely through security questions or OTPs.
3. **New User Creation:** Permits the creation of new employee accounts under various roles (Doctor, Pharmacy, Insurance, etc.).

The program interacts with several files like employee.json, insurance\_payment\_report.json, eye\_patient\_info.csv, and health\_data.json for data storage and retrieval.

**Code Breakdown**

1. **Global Data:**
   * reg: A dictionary containing appointment and metadata information.
   * dic: Patient IDs as keys with placeholder values.
   * sq: Security questions for validating patient identity.
2. **Utility Functions:**
   * **grant\_access(x)**: Displays patient details, diagnosis, prescriptions, payment, and rejection information from insurance\_payment\_report.json.
   * **diagnostics(id)**: Placeholder for patient diagnostic data.
   * **insurance(id)**: Reads patient insurance information from eye\_patient\_info.csv.
   * **doctor(id)**: Retrieves health data for a patient from health\_data.json.
   * **pharmacy(id)**: Reads diagnostic results for a patient from diagnosis\_results.csv.
   * **send\_email(mail, num)**: Sends an OTP to a patient's email address for identity verification.
3. **Main Workflow:**
   * **Option E:** Employee login to access department-specific functions based on ID and password validation.
   * **Option P:** Patient portal requiring authentication via security questions or OTP.
   * **Option new:** Creation of new users for different departments after ensuring the uniqueness of ID.

**File Descriptions**

1. **employee.json:**
   * Stores employee IDs and passwords.
   * Example format:

[

{"ID": "D123", "Password": "pass123"},

{"ID": "P456", "Password": "pass456"}

]

1. **insurance\_payment\_report.json:**
   * Contains detailed insurance and diagnosis data for patients.
   * Example format:

{

"Patient\_ID": "1",

"Patient\_Name": "Chinyemba Kalenga",

"Doctor\_ID": "D123",

"Doctor\_Name": "Dr. Smith",

"Diagnosis": [

{

"CPT\_Code": "12345",

"CPT\_Description": "Routine Checkup",

"Condition\_Name": "Hypertension",

"ICD10\_Code": "I10",

"ICD10\_Description": "Primary hypertension",

"Price": 150.00

}

],

"Prescribed\_Drugs": ["Amlodipine", "Losartan"],

"Payment\_Details": {"Insurance Covered": 100.00, "Out-of-pocket": 50.00},

"Rejections": ["Insurance claim rejected for lab tests."]

}

1. **eye\_patient\_info.csv:**
   * Stores patient insurance information.
   * Example format:

csv

Copy code

ID,Name,Age,Gender,Insurance

1,Chinyemba Kalenga,29,F,Yes

2,Ali,34,M,No

1. **health\_data.json:**
   * Contains health data per patient.
   * Example format:

json

Copy code

[

{"health\_data": {"Blood Pressure": ["120/80"], "Allergies": ["Penicillin"]}},

{"health\_data": {"Blood Pressure": ["140/90"], "Allergies": ["None"]}}

]

**Team Member Contributions**

1. **Dingani Freddie Kandiwo: Backend Developer**
   * **Contributions:**
     + Implemented the main functionality for the employee and patient portals.
     + Developed functions for data retrieval (grant\_access, doctor, insurance, pharmacy).
     + Worked on OTP generation and email integration in send\_email.
2. **Josep Boateng: Data Specialist**
   * **Contributions:**
     + Created and formatted JSON and CSV files for storage.
     + Ensured compatibility of file formats with the parsing logic.
     + Debugged issues related to file handling and data validation.
3. **Khloe Edwards: Security and Testing**
   * **Contributions:**
     + Designed and implemented the patient authentication system using OTP and security questions.
     + Validated login processes for employees and patients.
     + Performed rigorous testing to ensure system robustness and handled error cases (e.g., invalid inputs).

**Future Enhancements**

* Add a database backend for scalable and robust data storage.
* Implement encryption for sensitive data like passwords.
* Develop a GUI or web-based front-end for better usability.
* Integrate advanced security measures like two-factor authentication.

This modular structure and division of responsibilities ensured the project was completed efficiently while maintaining code quality.