User Guide

Hospital Management System

TABLE OF CONTENTS

User Guide: Hospital Management System

This user guide explains how to use the Hospital Management System, covering **patient**, **doctor**, **service**, **appointment**, and **billing** modules. It is intended for hospital staff, administrators, and users who need to manage hospital operations efficiently.

1. GETTING STARTED

1. **Database Setup**:
   Ensure your MySQL database is running and all required tables (patients, doctors, services, appointments, billing, billed_services, temp_service_usage) are created.
2. **Configuration**:
   Edit db_config.py with your database credentials and ensure it provides a get_connection() function.
3. **Modules**:
   The system is organized into modules:
   - patient.py: Manage patients
   - doctor.py: Manage doctors
   - service.py: Manage services and usage
   - appointment.py: Manage appointments
   - billing.py: Manage bills and invoices

2. PATIENT MANAGEMENT

Register a New Patient

- Use the Patient module to add a new patient with details like name, age, gender, address, phone, and email.
- Example:

```
from patient import Patient

p = Patient(patient_id="P001", name="John Doe", age=30, gender="Male", address="123 Main St", phone="9876543210", email="john@example.com")

p.add()
```

Update or Delete Patient

- Update patient details using the update() method.
- Remove a patient using delete(patient_id) (ensure no active appointments or bills).

View/Search/Export Patients

- List of all patients: Patient.view()
- Search by name: Patient.search_by_name("John")
- Export to CSV: Patient.export_patients_to_csv("patients.csv")

3. Doctor Management

Register a New Doctor

- Add a doctor with name, specialization, contact, department.
- Example:

```
from doctor import Doctor

d = Doctor(doctor_id="D001", name="Dr. Smith", specialization="Cardiology", phone="9876543210", email="smith@hospital.com", department="Cardiology")

d.add()
```

Update/Delete/View/Search Doctors

- Update: d.update()
- Delete: Doctor.delete("D001")
- View all: Doctor.view()
- Search by specialization: Doctor.search_by_specialization("Cardiology")
- Export to CSV: Doctor.export_doctors_to_csv("doctors.csv")

4. SERVICE MANAGEMENT

Add/Update/Delete Services

- Add a service (e.g., MRI, Blood Test) with cost.
- Example:

```
from service import Service
s = Service(service_id="S001", service_name="MRI Scan", cost=2500.0)
s.add()
```

- Update: s.update()
- Delete: Service.delete("S001")
- View all: Service.view()
- Search by name: Service.search_by_name("MRI")
- Export to CSV: Service.export_services_to_csv("services.csv")

Record Service Usage

- When a patient uses a service, record it:

```
from service import ServiceUsageDB
ServiceUsageDB.add_service_for_patient(patient_id="P001",service_id="S002",
service_name="Blood Test", cost=500.0)
```

5. APPOINTMENT MANAGEMENT

Schedule an Appointment

- Generate a new appointment ID:

```
from appointment import Appointment, auto_appt_id
appt_id = auto_appt_id()
appt = Appointment(appt_id, patient_id="P001", doctor_id="D001", date="2025-05-22", diagnosis="Checkup", consulting_charge=500.0)
appt.add()
```

Update/Cancel/View Appointments

- Update: Change details and call appt.update()
- Cancel: Appointment.delete("A001")
- View all: Appointment.view()
- Filter by date: Appointment.filter_appointments()
- Calculate days between appointments: Appointment.days_between_appointments("P001")
- Export to CSV: Appointment.export_appointment_summary_to_csv("appointments.csv")

6. BILLING & INVOICING

Generate a Bill

- Generate bill ID:

```
from billing import Bill, auto_bill_id
bill_id = auto_bill_id()
bill = Bill(bill_id, patient_id="P001", billing_date="2025-05-22")
bill.add()
```

- The system will fetch all unbilled services for the patient, calculate the total, and store the bill.

Update/Delete/View Bills

- Update: Change details and call bill.update()
- Delete: Bill.delete("B001")
- View all: Bill.view()

Generate Invoice

- After billing, print a detailed invoice:

```
bill.generate_invoice()
```

7. REPORTING & EXPORT

- Export patient, doctor, service, appointment, or billing data to CSV for backup, reporting, or analytics.
- Use the respective module's export function.

8. BEST PRACTICES

- Always validate input data (IDs, dates, charges).
- Use auto-generated IDs for appointments and bills.
- Regularly export data for backup.
- Ensure all dependencies (db_config.py, MySQL server, required tables) are in place.

9. TROUBLESHOOTING

- **Database Connection Errors:**
  Check your db_config.py and MySQL server status.
- **Validation Errors:**
  Ensure all fields (especially IDs, dates, charges) are correctly formatted.
- **Record Not Found:**
  Confirm the existence of referenced patients, doctors, or services before performing operations.
- **Duplicate ID Errors:**
  Always use the provided auto ID functions.

10.WORKFLOW SUMMARY

Patient Journey

1. Registration → 2. Appointment Scheduling → 3. Service Usage → 4. Billing & Invoice

Doctor Workflow

1. Registration → 2. Schedule Management → 3. Appointment Assignment →

   4. Consultation → 5. Billing Integration

Service Workflow

1. Add/Update Service → 2. Record Usage → 3. Billing → 4. Reporting

Appointment Workflow

1. Schedule → 2. Update/Cancel → 3. View/Filter → 4. Export

Billing Workflow

1. Generate Bill → 2. Record Services → 3. Generate Invoice → 4. Export