CS353 Project

Hospital Appointment Management System - MediSync

Final Report

Group 3: Veribaz

Asya Ünal - 22204017

Barış Yaycı - 22202981

Aybars Buğra Aksoy - 22202680

Eren Berk Eraslan - 22201772

Bora Yetkin - 22203661

Instructor: Özgür Ulusoy

TA: Hasan Alp Caferoğlu

Brief Description of the Application System

- Overview of the application, its purpose, and main functionalities.

Contribution of Each Group Member to the Term Project in Detail

Asya Ünal:
Aybars Buğra Aksoy:
Barış Yaycı:
Bora Yetkin:
Eren Berk Eraslan:

Final E/R (Entity-Relationship) Diagram

- The final version of the E/R diagram, including any modifications made after the design report.

Final List of Tables

1. Report(reportID, created_by, time_stamp**)**

Domains:

- reportID: Integer
- created_by: String
- time_stamp: Timestamp

Candidate Keys:

- reportID

Primary Key:

- reportID

Foreign Keys:

- created_by FK to Admin(userID)

2. PatientStatistics(reportID,statID, patientID, totalAppointments, totalProcesses, totalPaid, lastVisit, reportDate**)**

Domains:

- statID: Integer
- reportID: Integer
- patientID: Integer
- totalAppointments: Integer
- totalProcesses: Integer
- totalPaid: Numeric
- lastVisit: Date
- reportDate: Date

Candidate Keys:

- **(reportID,statID)**

Primary Key:

- **(reportID,statID)**

Foreign Keys:

- reportID FK to Report(reportID)
- patientID FK to Patients(patientID)

3. DoctorStatistics(reportID,statID, doctorID, prescriptionCount, appointmentCount, totalRevenue, reportDate, ratings**)**

Domains:

- statID: Integer
- reportID: Integer
- doctorID: Integer
- prescriptionCount: Integer
- appointmentCount: Integer
- totalRevenue: Numeric
- reportDate: Date
- ratings: Float

Candidate Keys:

- (reportID,statID)

Primary Key:

- (reportID,statID)

Foreign Keys:

- reportID FK to Report(reportID)
- doctorID FK to Doctors(doctorID)

4. EquipmentStatistics(statID, reportID, resourceID, usageCount, lastUsedDate, totalRequests**)**

Domains:

- statID: Integer
- reportID: Integer
- resourceID: Integer
- usageCount: Integer
- lastUsedDate: Date
- totalRequests: Integer

Candidate Keys:

- (statID, reportID)

Primary Key:

- (statID, reportID)

Foreign Keys:

- reportID → Report(reportID)
- resourceID → MedicalResources(resourceID)

MedicalResources(**resourceID**, name, availability)

Domains:

- resourceID: Integer (unique identifier)
- name: String (equipment name)
- availability: Boolean/String (e.g., "Available", "In Use", "Maintenance")

Candidate Keys:

- resourceID
- name - assuming equipment names are unique in the system

Primary Key: **resourceID** (as indicated in bold in the schema)
Foreign Keys: None

Dept(**deptName**, deptLocation)

Domains:

- deptName: String (department name)
- deptLocation: String (physical location)

Candidate Keys:

- deptName
- deptLocation - assuming each location houses only one department

Primary Key: **deptName** (as indicated in bold in the schema)
Foreign Keys: None

Doctors(**employeeID**, specialization, doctorLocation, *deptName*)

Domains:

- employeeID: Integer (unique identifier)
- specialization: String (medical specialty)
- doctorLocation: String (office/clinic location)
- deptName: String (department name)

Candidate Keys:

- employeeID

Primary Key: **employeeID** (as indicated in bold in the schema)

Foreign Keys:

- deptName references Dept(deptName)
- employeeID references Employee(employeeID)

Staff(**employeeID**, role)

Domains:

- employeeID: Integer (unique identifier)
- role: String (job title/role)

Candidate Keys:

- employeeID

Primary Key: **employeeID** (as indicated in bold in the schema)
Foreign Keys: employeeID references Employee(employeeID)

DoctorPatient(**doctorID,patientID**)

Domains:

- doctorID: Integer (doctor identifier)
- patientID: Integer (patient identifier)

Candidate Keys:

- (doctorID, patientID)

Primary Key: **(doctorID, patientID)** (as indicated in bold in the schema)
Foreign Keys:

- doctorID references Doctors(employeeID)
- patientID references Patient(patientID)

HospitalAdministirators(**employeeID**, role)

Domains:

- employeeID: Integer (unique identifier)
- role: String (administrative role)

Candidate Keys:

- employeeID

Primary Key: **employeeID** (as indicated in bold in the schema)

Foreign Keys: employeeID refer

ences Employee(employeeID)

Patients(**patientID**, name, DOB, email, phoneNumber, Balance)

Domains:

- patientID: Integer (unique identifier)
- name: String (patient name)
- DOB: Date (date of birth)
- email: String (valid email format)
- phoneNumber: String (valid phone number format)
- Balance: Decimal/Money (account balance)

Candidate Keys:

- patientID
- email - assuming email addresses are unique
- phoneNumber - assuming phone numbers are unique
- (name, DOB) - assuming name and birth date combinations are unique

Primary Key: **patientID** (as indicated in bold in the schema)
Foreign Keys: patientID references User(userID)

Slots(**doctorID**, **startTime, endTime**, availability)

Domains:

- doctorID: Integer (doctor identifier)
- startTime: DateTime (appointment start time)
- endTime: DateTime (appointment end time)
- availability: Boolean/String (e.g., "Available", "Booked")

Candidate Keys:

- (doctorID, startTime, endTime) - assuming no overlapping slots for same doctor

Primary Key: **(doctorID, slotID, startTime, endTime)** (as indicated in bold in the schema)
Foreign Keys: doctorID references Doctors(employeeID)

Appointment(**appointmentID**, status, rating, review, *patientID*, *doctorID,startTime,endTime*)

Domains:

- appointmentID: Integer (unique identifier)
- status: String (e.g., "Scheduled", "Completed", "Cancelled")
- rating: Integer/Float (typically 1-5 scale)
- review: Text (patient feedback)
- patientID: Integer (patient identifier)
- doctorID: Integer (doctor identifier)
- startTime: DateTime (appointment start time)
- endTime: DateTime (appointment end time)

Candidate Keys:

- appointmentID

Primary Key: **appointmentID** (as indicated in bold in the schema)
Foreign Keys:

- patientID references Patients(patientID)
- doctorID,startTime,endTime references **Slots**(employeeID)

Process(**processID**, processName, processDescription, status, *appointmentID*)

Domains:

- processID: Integer (unique identifier)
- processName: String (name of medical procedure/service)
- processDescription: Text (detailed description)
- status: String (e.g., "Scheduled", "In Progress", "Completed")
- appointmentID: Integer (appointment identifier)

Candidate Keys:

- processID
- (appointmentID, processName) - assuming each procedure type is performed only once per appointment

Primary Key: **processID** (as indicated in bold in the schema)
Foreign Keys: appointmentID references Appointment(appointmentID)

Billing(**billingID**, billingDate, amount, paymentStatus, *processID*)

Domains:

- billingID: Integer (unique identifier)
- billingDate: Date (invoice date)
- amount: Decimal/Money (charged amount)
- paymentStatus: String (e.g., "Paid", "Pending", "Overdue")
- processID: Integer (process identifier)

Candidate Keys:

- billingID
- (processID, billingDate) - assuming one billing record per process per date

Primary Key: **billingID** (as indicated in bold in the schema)
Foreign Keys: processID references Process(processID)

Medications(**medicationName**, description, information)

Domains:

- medicationName: String (name of medication)
- description: Text (usage instructions)
- information: Text (additional information)
- doctorID: Integer (doctor identifier)
- appointmentID: Integer (appointment identifier)

Candidate Keys:

- medicationName

Primary Key: **medicationName** (as indicated in bold in the schema)

Prescribes(**medicationName**,**appointmentID**)

Domains:

- medicationName: String (name of medication)
- appointmentID: Integer (appointment identifier)

Candidate Keys:

- **(medicationName,appointmentID)**

Primary Key: **(medicationName,appointmentID)** (as indicated in bold in the schema)

Foreign Keys: MedicationName references Medicatios

Employee(**employeeID**, salary)

Domains:

- employeeID: Integer (unique identifier)
- salary: Decimal/Money (employee salary)

Candidate Keys:

- employeeID

Primary Key: **employeeID** (as indicated in bold in the schema)
Foreign Keys: employeeID references User(userID)

Admin(**AdminID**)

Domains:

- AdminID: Integer (unique identifier)

Candidate Keys:

- AdminID

Foreign Keys: AdminID references User(userID)

User(**userID**, name, email, identityNumber, password)

Domains:

- userID: Integer (unique identifier)
- name: String (user name)
- email: String (valid email format)
- identityNumber: String (government-issued ID)
- password: String (hashed password)

Candidate Keys:

- userID
- email - assuming email addresses are unique

- identityNumber - assuming government IDs are unique

Primary Key: **userID** (as indicated in bold in the schema)
Foreign Keys: None

Request(***doctorID*****,** ***resourceID***, status)

Domains:

- doctorID: Integer (doctor identifier)
- resourceID: Integer (resource identifier)
- status: String (e.g., "Pending", "Approved", "Denied")

Candidate Keys:

- (doctorID, resourceID)

Primary Key: **(doctorID, resourceID)** (composite key as indicated)
Foreign Keys:

- doctorID references Doctors(employeeID)
- resourceID references MedicalResources(resourceID)

Implementation Details

- Description of how key concepts were implemented in the programming environment.
- How you connect and access the database (e.g., SQL queries, connection methods).
- Details of the GUI (if applicable), constraints enforcement, and other significant implementation details.

Advanced Database Components

- Discussion on the use of advanced database features like views, triggers, and constraints.

- How these components were utilized to handle user operations and enhance the system.

User's Manual

- Detailed instructions for using and maintaining the system.
- Information for all possible user groups (e.g., administrators, end-users).
- GUI elements and how users interact with the system.