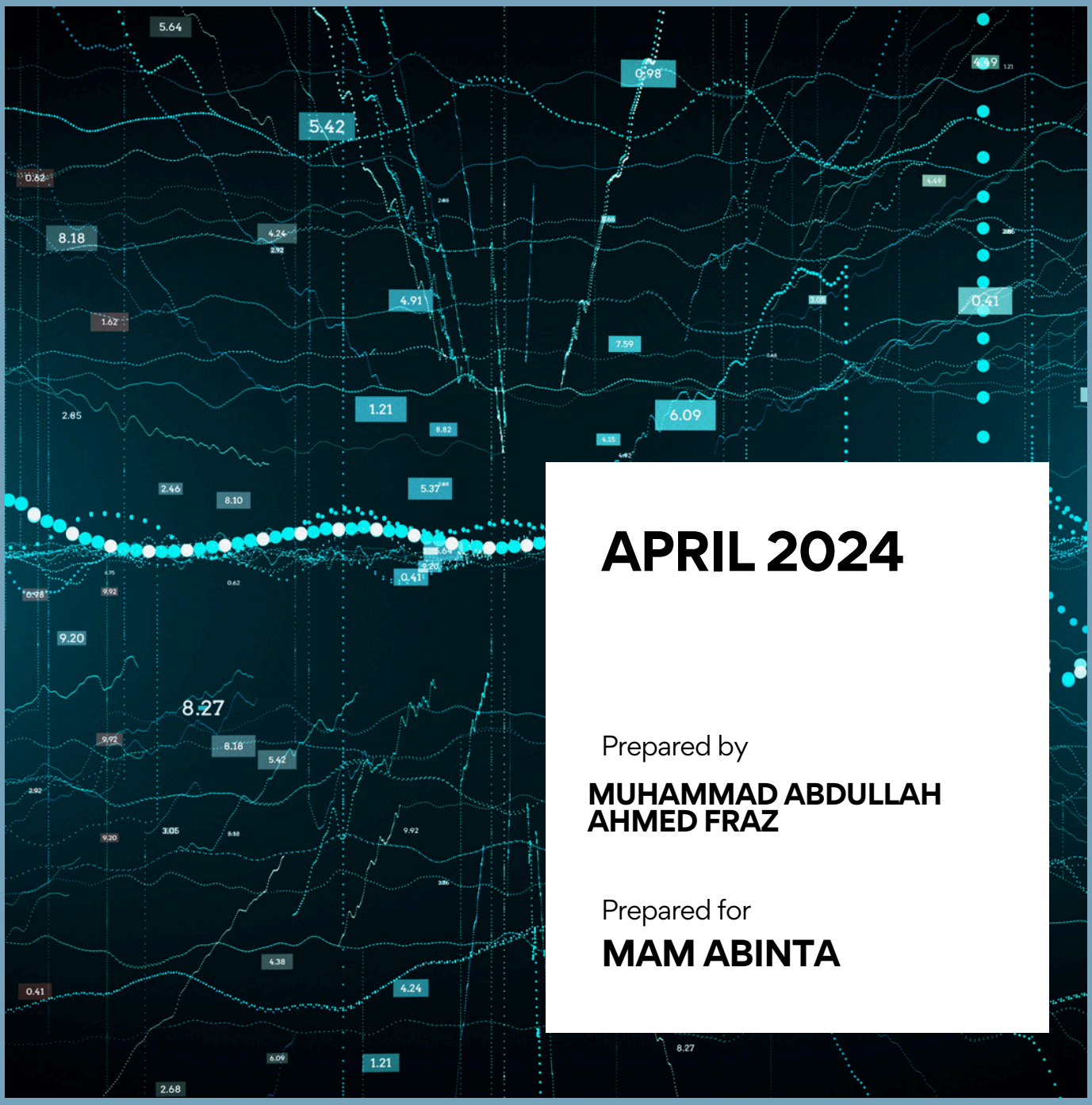
HOTEL MANAGEMENT SYSTEM REPORT

DESIGNING AND IMPLEMENTING A DATABASE MANAGEMENT SYSTEM



APRIL 2024

Prepared by

MUHAMMAD ABDULLAH AHMED FRAZ

Prepared for

MAM ABINTA

TABLE OF CONTENTS

BUSINESS SCENARIO ANALYSIS

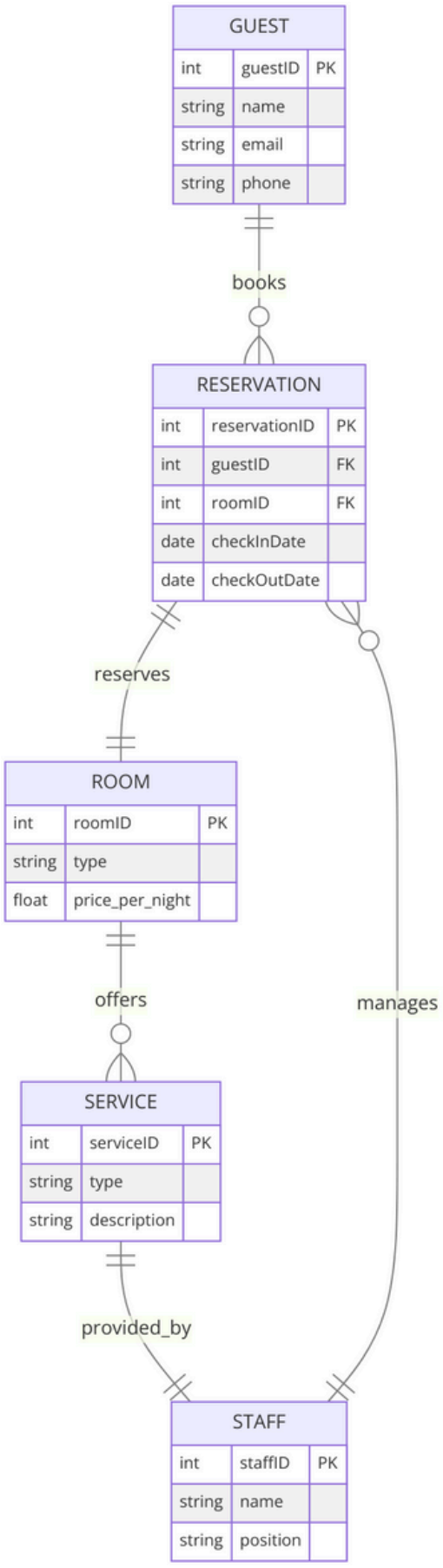
WHY WE CHOOSED?

A Hotel Booking System is a practical and rich scenario to explore for your database management project. Let's start by defining the main entities and their relationships for your Entity Relationship Diagram (ERD).

ENTITIES AND ATTRIBUTES

- Guests
- GuestID (Primary Key)
- FirstName
- LastName
- Email
- Phone
- Rooms
- RoomID (Primary Key)
- RoomNumber
- RoomType (e.g., Single, Double, Suite)
- Rate
- Status (e.g., Occupied, Available, Cleaning)
- Bookings
- BookingID (Primary Key)
- BookingID (Primary Key)
- GuestID (Foreign Key)
- RoomID (Foreign Key)
- CheckInDate
- CheckOutDate
- NumberOfGuests
- SpecialRequests
- Staff
- StaffID (Primary Key)
- FirstName
- LastName
- Position
- ContactNumber
- Services (optional, if you want to track services like Spa, Dining, Room Service)

ENTITY RELATIONSHIP DIAGRAM (ERD) DESIGN:



DRAWING THE ERD:

- Start by placing your entities (Guests, Rooms, Bookings, Staff, Services if included) as boxes and label them with the entity names.
- Add attributes under each entity, marking the primary keys. For foreign keys, ensure they are noted (usually with an FK label).
- Connect entities with relationships. Use lines to connect entities and add relationship labels (like "books", "assigned to", etc.). Also, depict the type of relationship (one-to-many, many-to-many) using crow's feet notation.

RELATIONSHIPS:

- Guests to Bookings: One-to-Many (One guest can have multiple bookings, each booking is made by one guest)
- Rooms to Bookings: One-to-Many (One room can be booked multiple times over different dates, each booking is for one room)
- Staff to Rooms: Many-to-Many (optional, for scenarios where multiple staff members are responsible for multiple rooms, like cleaning or maintenance; this would require a junction table if implemented)

STEPS FOR NORMALIZATION:

✳ 1ST NORMAL FORM (1NF)

- Ensure all tables' columns have atomic values.
- Ensure each column in a table is unique.

✳ 2ND NORMAL FORM (2NF)

- Meet all requirements of 1NF.
- Remove partial dependencies, i.e., no column should be dependent on only a part of the primary key in a composite primary key scenario.

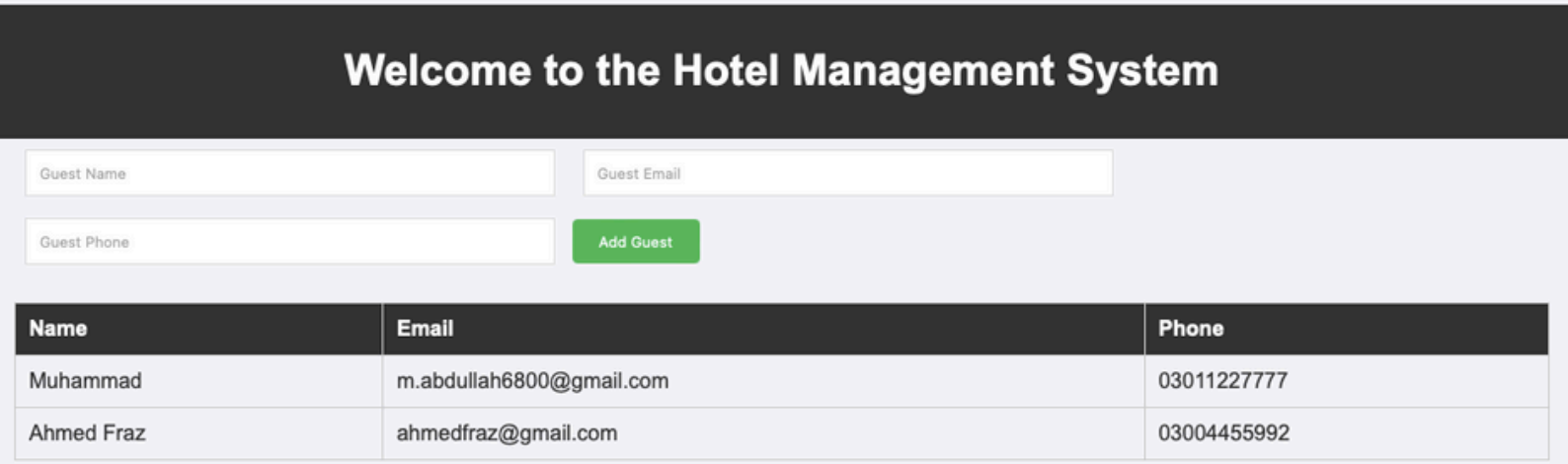
✳ 3RD NORMAL FORM (3NF)

- Meet all requirements of 2NF.
- Remove transitive dependencies, i.e., no non-primary key column should depend on another non-primary key column.

✳ OUR TABLE ALREADY IN 3NF

- Each table has a primary key.
- Attributes in each table depend only on the primary key (2NF and 3NF compliant).

USER INTERFACE



- *To create a user interface using Flask and PostgreSQL*
- *Set up your environment by installing Python, Flask, PostgreSQL, and psycopg2-binary.*
- *Initialize a Flask application in app.py, configure it to connect to PostgreSQL, and define SQLAlchemy models to represent your database structure.*
- *Use HTML forms for user input and Jinja2 templates for dynamic content rendering.*
- *Style the interface with CSS for better user experience. Run the Flask server locally.*

THANK YOU