
**The Maharaja Sayajirao University**

**Faculty Of Technology & Engineering**

Computer Science and Engineering

# DMS PROJECT REPORT

***Hospital management System - Care Foundation***

## Submitted By:

*Class: SS-BE-II-CSE*



| SR | PRN | NAME |
|---|---|---|
| **1** | 8022053249 | DAVID BERNARDO FRANCISCO |
| **2** | 8022057524 | MISHRA NILESH SANJAYKUMAR |
| **3** | 8023057802 | TINANI BHUVAN JITENDRA |



## For Acadamic year

## 2023-2024

# Abstraction

Care Foundation runs a bunch of healthcare centers, each looked after by a boss called the Medical Director. When people visit their regular doctors, those doctors might suggest they go to one of these centers for treatment, either staying overnight or just visiting during the day. Sometimes, people need help taking care of themselves at home, so they can stay at these centers instead. Everything that happens during a patient's time at the center, like appointments, treatments, and leaving, is carefully recorded.

When someone needs to stay at the center, they go through a process where the staff assesses how much help they need. Then, they're put on a list based on how much help they need, and when a spot opens up, they get in. Some of the costs might be paid by the government. When someone's ready to leave, the staff might help set them up with more help if they need it, like moving to a nursing home.

The people who work at the centers, like doctors, nurses, and office staff, all have their jobs carefully planned out to make sure everything runs smoothly. By keeping detailed records, the centers can make sure everyone gets the same good care and figure out how to make things even better.

Residential care at Care Foundation includes a formal admissions procedure, which frequently consists of facility visits and evaluations to assess the level of need. Applicants are assigned priority codes based on their care needs, which are used to manage waiting lists for beds that become available. Financial evaluations determine funding origins, with certain expenses being taken care of by municipal entities. Discharge plans may require working with outside organizations, especially when patients move to nursing homes. Staffing involves a combination of employees on salary and workers paid by the hour, carefully assigned according to patient requirements and operational needs to guarantee the best service provision.

Effective operations depend on strong staffing plans and careful record-keeping procedures. Care Foundation has a varied staff that includes doctors, nurses, administrative staff, and support personnel, all playing a role in the efficient operation of care centers. The organization's focus on thorough patient records supports consistency in treatment, allowing healthcare providers to monitor patient improvement from admission to discharge through various appointments. This method guarantees that patients get uniform, top-notch treatment, and allows Care Foundation to improve how resources are allocated and how operations are efficient.

## LIST OF TABLE

## SHORT DESCRIPTION

1. Treatment and Care Centres:

- Each care centre has a Medical Director overseeing operations.
- Patients are referred by their general practitioner and treated as in-patients or out-patients.
- Hospitals provide in-home care for patients unable to care for themselves.
- Treatment plans may include drug regimens, testing sessions, and therapy.
- There are waiting lists for appointments and beds, which need to be managed.

2. Residential Treatment:

- Long-term care beds have a priority system and waiting lists.
- Charges may be covered by local government depending on financial situation.
- The source of payment is verified before admitting a patient as a resident.

3. Discharge Plans:

- External organizations may need to be notified upon patient discharge.
- Some patients may become residents instead of returning home.

4. Staffing:

- Staff includes medical professionals, nurses, administrative staff, porters, and cleaners.
- Different holiday and pension rights for salaried and hourly employees.
- Most employees work shifts, with specific requirements for nursing and ancillary staff.
- The new system aims to simplify roster creation.

5. Medical Records and Fees:

- Patient information, treatment plans, and prescription history must be maintained.
- The patient's journey from admission to discharge should be tracked comprehensively.

# Assumptions

1. Patient - Residential Room (One-to-One): Each patient may be assigned to one residential room, and each residential room is assigned to one patient at a time.

2. Patient - Care Center (Many-to-Many): Patients receive care at care centers, and each care center serves multiple patients. Conversely, each patient may receive care from multiple care centers over time.

3. Treatment - Course of Drug Detail (One-to-Many): A treatment may consist of multiple courses of drugs, but each course of drug detail belongs to only one treatment.

4. Patient - Appointment (One-to-Many): Patients may have multiple appointments for treatments, but each appointment is associated with only one patient.

5. Patient - Admit Detail (One-to-Many): Patients may have multiple admission details (e.g., multiple admissions to the hospital), but each admission detail corresponds to one patient.

6. Doctor & Nurse - Shift (Many-to-Many): Each Doctor & Nurse member may work multiple shifts, and each shift may be worked by multiple Doctor & Nurse members.

7. Ward - Bed (One-to-Many): Each ward may contain multiple beds, but each bed belongs to only one ward.

8. Care Center - Medical Director (One-to-One): Each care center has exactly one medical director, and each medical director is responsible for only one care center.

9. Doctor - Patient (Many-to-Many): Doctors may treat multiple patients, and each patient may be treated by multiple doctors.

10. Nurse - Patient (Many-to-Many): Nurses may care for multiple patients, and each patient may be cared for by multiple nurses.

11. Patient - Outside Agency (Many-to-Many): Patients may interact with multiple outside agencies, and each outside agency may be involved with multiple patients.

12. Patient - Waiting List (One-to-Many): Each patient may be on multiple waiting lists, but each waiting list contains only one patient.

# Functions & Procedures

**Function check_todays_appointment:**

- Takes a patient name as input.
- Retrieves the patient's number based on the provided name.
- Searches for appointments scheduled for the current day for the identified patient.
- Returns appointment information if found, or appropriate messages if no appointments are found or if an error occurs.

**Function generate_patient_id:**

- Generates a unique patient ID based on the current date and existing patient records.
- It constructs the ID by concatenating the current date (in MMDDYYYY format) with a sequence number.
- Handles exceptions if any errors occur during the process.

**Function get_next_contact_id:**

- Retrieves the next available contact ID from the contact_detail table.
- If the table is empty, it starts with ID 1.
- Handles exceptions if any errors occur during the process.

**Function get_next_patient_dependent_id:**

- Retrieves the next available patient dependent ID from the patient_dependent table.
- If the table is empty, it starts with ID 1.
- Handles exceptions if any errors occur during the process.

**Procedure insert_patient_details:**

- Inserts patient details, contact details, and dependent details into their respective tables.
- Utilizes the previously defined functions to generate IDs.
- Commits the transaction if successful, or rolls back and displays an error message if an exception occurs.

**Procedure display_patient_details:**

- Displays comprehensive patient details including patient information, contact details, and dependent details.
- Retrieves the details by joining multiple tables (patient_main, contact_detail, and patient_dependent) based on the provided patient name.
- Utilizes exception handling to manage cases where no data is found or if an error occurs during the process.
- Overall, this package body provides functionality for managing patient appointments, generating patient IDs, retrieving contact IDs, managing patient-dependent IDs, inserting patient details, and displaying comprehensive patient information.

# Packages & Specifications

**Package Specification**:

This is a package body in PL/SQL named hospital_management, which likely forms part of a hospital management system.

Function check_todays_appointment:

Takes a patient name as input.

Retrieves the patient's number based on the provided name.

Searches for appointments scheduled for the current day for the identified patient.

Returns appointment information if found, or appropriate messages if no appointments are found or if an error occurs.

Function generate_patient_id:

Generates a unique patient ID based on the current date and existing patient records.

It constructs the ID by concatenating the current date (in MMDDYYYY format) with a sequence number.

Handles exceptions if any errors occur during the process.

Function get_next_contact_id:

Retrieves the next available contact ID from the contact_detail table.

If the table is empty, it starts with ID 1.

Handles exceptions if any errors occur during the process.

Function get_next_patient_dependent_id:

Retrieves the next available patient dependent ID from the patient_dependent table.

If the table is empty, it starts with ID 1.

Handles exceptions if any errors occur during the process.

Procedure insert_patient_details:

Inserts patient details, contact details, and dependent details into their respective tables.

Utilizes the previously defined functions to generate IDs.

Commits the transaction if successful, or rolls back and displays an error message if an exception occurs.

Procedure display_patient_details:

Displays comprehensive patient details including patient information, contact details, and dependent details.

Retrieves the details by joining multiple tables (patient_main, contact_detail, and patient_dependent) based on the provided patient name.

Utilizes exception handling to manage cases where no data is found or if an error occurs during the process.

Overall, this package body provides functionality for managing patient appointments, generating patient IDs, retrieving contact IDs, managing patient-dependent IDs, inserting patient details, and displaying comprehensive patient information.

Code:

```
CREATE OR REPLACE PACKAGE BODY hospital_management AS

     function check_todays_appointment(p_patient_name IN VARCHAR2)
    RETURN VARCHAR2
IS
    v_patient_no INTEGER;
    v_appointment_info VARCHAR2(255);
BEGIN
    -- Get the patient's number based on the given name
    SELECT patient_no INTO v_patient_no
    FROM patient_main
    WHERE first_name || ' ' || middle_name || ' ' || last_name = p_patient_name;

    -- Check for appointments scheduled for today
    SELECT appointment_no || ', ' || degree_of_care || ', ' ||
TO_CHAR(appointment_date, 'DD-MON-YYYY') || ', ' || TO_CHAR(appointment_time,
'HH24:MI:SS')
    INTO v_appointment_info
    FROM appointment
    WHERE fk1_outpatient_id = v_patient_no
    AND TRUNC(appointment_date) = TRUNC(SYSDATE);

    RETURN v_appointment_info;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'No appointments found for today.';
    WHEN OTHERS THEN
        RETURN 'Error occurred while checking appointments.';
END check_todays_appointment;


    FUNCTION generate_patient_id
    RETURN NUMBER
IS
    v_date_part VARCHAR2(8);
    v_seq_number NUMBER;
    v_patient_id NUMBER;
BEGIN
    -- Get the current date in the format MMDDYYYY
    v_date_part := TO_CHAR(SYSDATE, 'MMDDYYYY');

    -- Get the count of existing patient records for the current date
```

```
    SELECT COUNT(*) INTO v_seq_number
    FROM patient_main
    WHERE TO_CHAR(date_of_birth, 'MMDDYYYY') = v_date_part;

    -- Increment the count by 1 to get the next patient ID
    v_seq_number := v_seq_number + 1;

    -- Concatenate date part and sequence number to form the patient ID
    v_patient_id := TO_NUMBER(v_date_part || LPAD(v_seq_number, 1, '0'));

    RETURN v_patient_id;
END generate_patient_id;


    FUNCTION get_next_contact_id
    RETURN NUMBER
IS
    v_last_id NUMBER;
    v_next_id NUMBER;
BEGIN
    -- Get the last ID from the contact_detail table
    SELECT MAX(contact_id) INTO v_last_id FROM contact_detail;

    -- Increment by 1 to get the next ID
    v_next_id := v_last_id + 1;

    RETURN v_next_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        -- If the table is empty, start with ID 1
        RETURN 1;
    WHEN OTHERS THEN
        RETURN -1; -- Return -1 for any errors
END get_next_contact_id;


    FUNCTION get_next_patient_dependent_id RETURN NUMBER IS
        v_dependent_id NUMBER;
    BEGIN
        -- Your implementation for get_next_patient_dependent_id function goes
here
        RETURN v_dependent_id;
    END get_next_patient_dependent_id;

    PROCEDURE display_patient_details(
    p_patient_name IN VARCHAR2
)
IS
    v_patient_id INTEGER;
BEGIN
    -- Get the patient's ID based on the given name
    SELECT patient_no INTO v_patient_id
    FROM patient_main
    WHERE first_name || ' ' || middle_name || ' ' || last_name = p_patient_name;

    -- Display patient details using inner join
    DBMS_OUTPUT.PUT_LINE('Patient Details:');
    FOR patient_rec IN (
        SELECT pm.patient_no, pm.first_name, pm.middle_name, pm.last_name,
pm.date_of_birth, pm.gender, pm.blood_group,
```

```
                pm.source_payment, cd.contact_id, cd.telephone, cd.address_line1,
cd.address_line2,
                pd.patient_dependent_id, pd.first_name AS dependent_first_name,
pd.middle_name AS dependent_middle_name,
                pd.last_name AS dependent_last_name, pd.address_line1 AS
dependent_address_line1,
                pd.address_line2 AS dependent_address_line2, pd.area_name AS
dependent_area_name, pd.telephone AS dependent_telephone
        FROM patient_main pm
        INNER JOIN contact_detail cd ON pm.patient_no = cd.fk1_patient_no
        INNER JOIN patient_dependent pd ON pm.patient_no = pd.fk1_patient_no
        WHERE pm.patient_no = v_patient_id
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Patient ID: ' || patient_rec.patient_no);
        DBMS_OUTPUT.PUT_LINE('First Name: ' || patient_rec.first_name);
        DBMS_OUTPUT.PUT_LINE('Middle Name: ' || patient_rec.middle_name);
        DBMS_OUTPUT.PUT_LINE('Last Name: ' || patient_rec.last_name);
        DBMS_OUTPUT.PUT_LINE('Date of Birth: ' ||
TO_CHAR(patient_rec.date_of_birth, 'DD-MON-YYYY'));
        DBMS_OUTPUT.PUT_LINE('Gender: ' || patient_rec.gender);
        DBMS_OUTPUT.PUT_LINE('Blood Group: ' || patient_rec.blood_group);
        DBMS_OUTPUT.PUT_LINE('Source of Payment: ' ||
patient_rec.source_payment);
        DBMS_OUTPUT.PUT_LINE('Contact ID: ' || patient_rec.contact_id);
        DBMS_OUTPUT.PUT_LINE('Telephone: ' || patient_rec.telephone);
        DBMS_OUTPUT.PUT_LINE('Address Line 1: ' || patient_rec.address_line1);
        DBMS_OUTPUT.PUT_LINE('Address Line 2: ' || patient_rec.address_line2);
        DBMS_OUTPUT.PUT_LINE('Dependent ID: ' ||
patient_rec.patient_dependent_id);
        DBMS_OUTPUT.PUT_LINE('Dependent First Name: ' ||
patient_rec.dependent_first_name);
        DBMS_OUTPUT.PUT_LINE('Dependent Middle Name: ' ||
patient_rec.dependent_middle_name);
        DBMS_OUTPUT.PUT_LINE('Dependent Last Name: ' ||
patient_rec.dependent_last_name);
        DBMS_OUTPUT.PUT_LINE('Dependent Address Line 1: ' ||
patient_rec.dependent_address_line1);
        DBMS_OUTPUT.PUT_LINE('Dependent Address Line 2: ' ||
patient_rec.dependent_address_line2);
        DBMS_OUTPUT.PUT_LINE('Dependent Telephone: ' ||
patient_rec.dependent_telephone);
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Patient details not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred while displaying patient details.');
END display_patient_details;

END hospital_management;
```

# Triggers on our System

## trg_ancillary_working_hour

Trigger Logic:

For each row being inserted or updated in the ancillary table, the trigger checks if the value of the ancillary_working_hour column in the new row is less than 4 or greater than 8.

If the working hours fall outside the range of 4 to 8, the trigger raises an application error and a corresponding message indicating that working hours for ancillary must be between 4 and 8.

Code:

```
CREATE OR REPLACE TRIGGER trg_ancillary_working_hour
BEFORE INSERT OR UPDATE ON ancillary
FOR EACH ROW
BEGIN
    IF :NEW.ancillary_working_hour < 4 OR :NEW.ancillary_working_hour > 8 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Working hours for ancillary must be
between 4 and 8.');
    END IF;
END;
```

## trg_cleaner_working_hour

Trigger Logic:

For each row being inserted or updated in the cleaner table, the trigger checks if the value of the cleaner_working_hour column in the new row is less than 4 or greater than 8.

If the working hours fall outside the range of 4 to 8, the trigger raises an application error and a corresponding message indicating that working hours for cleaner must be between 4 and 8.

Code:

```
CREATE OR REPLACE TRIGGER trg_cleaner_working_hour
BEFORE INSERT OR UPDATE ON cleaner
FOR EACH ROW
BEGIN
    IF :NEW.cleaner_working_hour < 4 OR :NEW.cleaner_working_hour > 8 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Working hours for cleaner must be
between 4 and 8.');
    END IF;
END;
```

**trg_porter_working_hour**

Trigger Logic:

For each row being inserted or updated in the porter table, the trigger checks if the value of the porter_working_hour column in the new row is less than 4 or greater than 8.

If the working hours fall outside the range of 4 to 8, the trigger raises an application error and a corresponding message indicating that working hours for a porter must be between 4 and 8.

Code:

```
CREATE OR REPLACE TRIGGER trg_porter_working_hour
BEFORE INSERT OR UPDATE ON porter
FOR EACH ROW
BEGIN
    IF :NEW.porter_working_hour < 4 OR :NEW.porter_working_hour > 8 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Working hours for porter must be between
4 and 8.');
    END I
```

**prevent_duplicate_nurse_assignment**

Trigger Logic:

For each row being inserted or updated in the assign_nurse table, the trigger checks if there's already an assignment for the nurse (fk1_nurse_id) on the same date (assign_date).

It performs a query to count the number of rows in the assign_nurse table where the nurse ID and assignment date match the new values being inserted or updated.

If the count (v_count) is greater than 0, it means there's already an assignment for the nurse on the same date, and it raises an application error.

Code:

```
CREATE OR REPLACE TRIGGER prevent_duplicate_nurse_assignment
BEFORE INSERT OR UPDATE ON assign_nurse
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM assign_nurse
    WHERE fk1_nurse_id = :NEW.fk1_nurse_id
    AND assign_date = :NEW.assign_date;

    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20013, 'The same nurse cannot be assigned to
more than one ward at the same time');
    END IF;
END;
/
```

# Conclusion

Ultimately, Care Foundation runs a system of care facilities overseen by Medical Directors who have the independence to run their centers. The company offers a wide range of healthcare services, such as inpatient and outpatient treatment, therapy sessions, and residential care, for those who require assistance. Care Foundation places a high priority on patient welfare, conducting detailed admission processes, overseeing residential care waiting lists, and working with outside organizations for discharge preparation.

The organization's dedication to efficient operations also includes staffing arrangements, where a varied team of healthcare professionals and support staff are strategically assigned to fulfill patient requirements. By maintaining detailed records, the Care Foundation ensures that care is continuous and allows healthcare providers to monitor patient improvement from the time of admission until discharge. This method improves both patient results and resource distribution while also increasing operational effectiveness in the organization. In general, Care Foundation's commitment to providing top-notch healthcare services reflects its goal of offering high-quality care to all patients.

.