一、 系統功能介紹

本系統旨在幫助醫院簡單有效地管理患者、醫生和其它重要資訊。

系統主要功能:

- 患者功能

  註冊帳號:患者可以在系統註冊,並獲得 Patient ID。

  更新資料:患者可修改自己的聯絡資訊,但無法更改治療相關記錄。

- 醫生功能

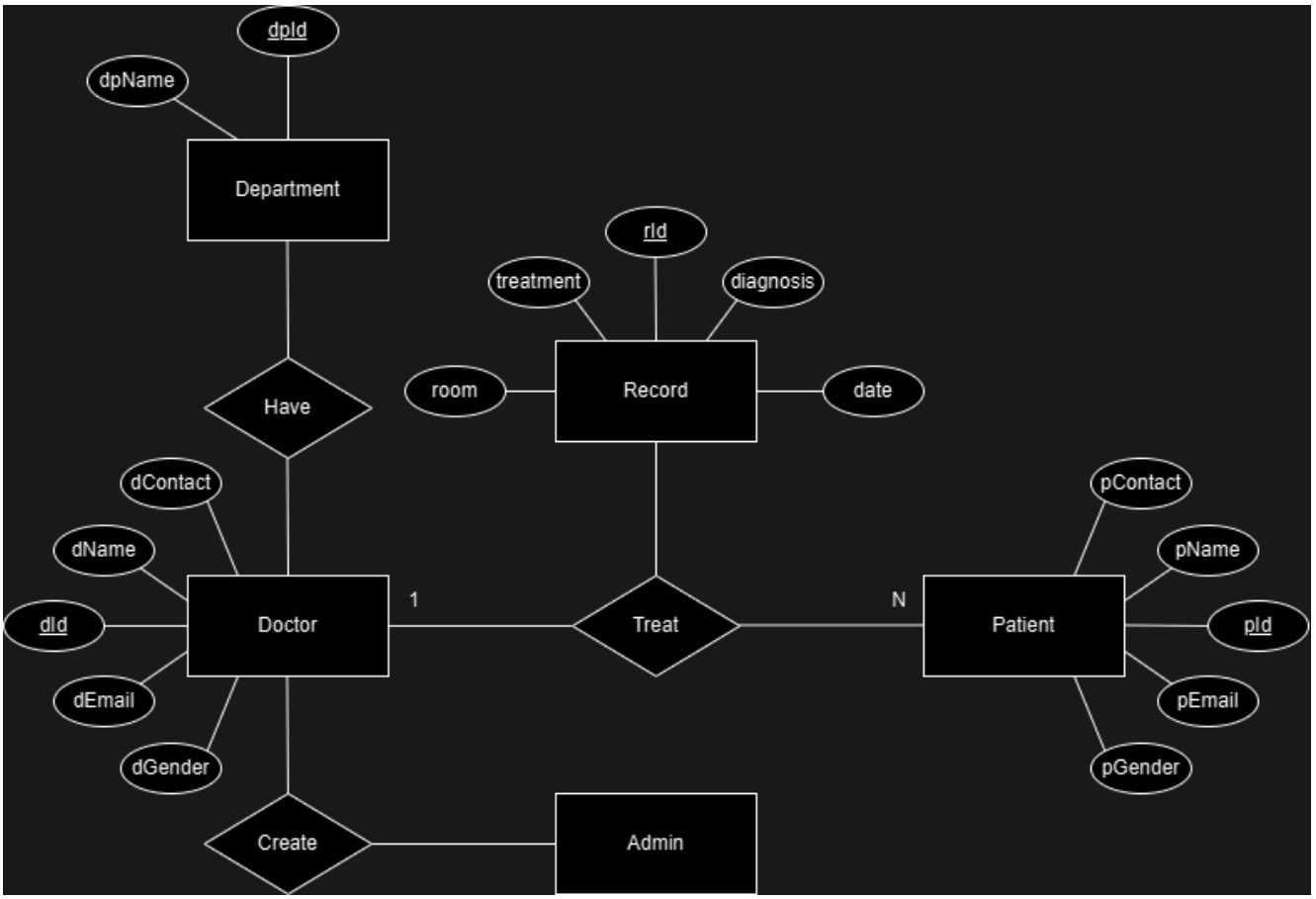
  登入系統:醫生使用 Doctor ID 登入,查詢或更新其負責的患者資料。

  資料管理:新增、修改或查看患者的治療相關記錄

- 管理員功能

  資料管理:新增、修改或查看醫生的相關資訊

二、 對應系統功能的ER-diagram



注意:

Department 實體中的 dpName 是部門名稱, 而 Doctor 實體中的 dName 是醫生名稱。另外, Record 實體中的 treatment 是治療描述, 而 Patient 實體中的 pName 是病人名稱。

三、 表格以關SQL DDL定義

1. Departments 表格

```
CREATE TABLE `departments` (
    `dpt_ID` int(11) NOT NULL,
    `dpt_name` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
ALTER TABLE `departments`
    ADD PRIMARY KEY (`dpt_ID`);
ALTER TABLE `departments`
    MODIFY `dpt_ID` int(11) NOT NULL AUTO_INCREMENT,
    AUTO_INCREMENT=7;
```

dpt_ID	dpt_name
1	Dermatology
2	Cardiology
3	Neurology
4	Pediatrics
5	Psychiatry
6	Emergency

- [**dpt_ID**] 記錄各部門的編號，為 **Primary Key**。
- **[dpt_name]** 記錄各部門的名稱。

因為 **dpt_ID** 代表一個特定的部門，可決定唯一的 **dpt_name**，又 **dpt_ID** 是一個 **Candidate Key**，所以此表格符合 **3NF** 和 **BCNF**。

2. Doctors 表格

```
CREATE TABLE `doctors` (
        `doc` int(11) NOT NULL,
        `d_ID` text NOT NULL,
        `d_name` text NOT NULL,
        `d_pic` blob NOT NULL,
        `d_department` text NOT NULL,
        `d_num` text NOT NULL,
        `d_email` text NOT NULL,
        `d_pass` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
ALTER TABLE `doctors`
        ADD PRIMARY KEY (`doc`),
        ADD UNIQUE KEY `d_num` (`d_num`,`d_email`) USING HASH;
ALTER TABLE `doctors`
    MODIFY `doc` int(11) NOT NULL AUTO_INCREMENT,
        AUTO_INCREMENT=32;
```

doc	d_ID	d_name	d_pic	dpt_name	d_num	d_email	d_pass
1	d20	William	“profile picture”	Dermatology	0910010101	william123@gmail.com	doctor1
2	d22	Jacob	“profile picture”	Pediatrics	0998765432	jacob123@gmail.com	doctor3
3	d24	Nicolas	“profile picture”	Psychiatry	0921123789	nicolas123@gmail.com	doctor5

- **[d_ID]** 記錄各醫生的編號，為 **Primary Key**。
- **[d_name]** 記錄各醫生的名字。
- **[d_pic]** 記錄各醫生的照片。
- **[dpt_name]** 記錄各醫生所屬的部門名稱。
- **[d_num]** 記錄各醫生的聯絡號碼。
- **[d_email]** 記錄各醫生的電子郵件地址。

- **[d_pass]** 記錄各醫生的密碼。

因為 **doc** 代表一個特定的醫生，可決定唯一的 **d_ID**、**d_name**、**d_pic**、**d_department**、**d_num**、**d_email** 和 **d_pass**，又 **doc** 是一個 **Candidate Key**，所以此表格符合 **3NF** 和 **BCNF**。此外，**d_num** 和 **d_email** 具有唯一限制，確保沒有多餘的依賴關係。

3. Patients 表格

```
CREATE TABLE `patients` (
        `pat` int(11) NOT NULL,
        `p_ID` text NOT NULL,
        `p_name` text NOT NULL,
        `p_gender` text NOT NULL,
        `p_num` text NOT NULL,
        `p_email` text NOT NULL,
        `p_pass` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `patients`
        ADD PRIMARY KEY (`pat`),
        ADD UNIQUE KEY `contact number` (`p_num`) USING HASH,
        ADD UNIQUE KEY `email` (`p_email`) USING HASH,
        ADD UNIQUE KEY `id` (`p_ID`) USING HASH;
ALTER TABLE `patients`
        MODIFY `pat` int(11) NOT NULL AUTO_INCREMENT,
       AUTO_INCREMENT=9;
```

pat	p_ID	p_name	p_gender	p_num	p_email	p_pass
1	p1	Bitty	M	0852	bitty@gmail.com	kukubet
2	p3	Kuru	F	0878	kuruturu@gmail.com	kurinsideme
3	p7	Kevin	M	2605	kevinw@gmail.com	wijaya

- **[p_ID]** 記錄各病人的編號，為 **Primary Key**。
- **[p_name]** 記錄各病人的名字。
- **[p_gender]** 記錄各病人的性別。
- **[p_num]** 記錄各病人的聯絡號碼。
- **[p_email]** 記錄各病人的電子郵件地址。
- **[p_pass]** 記錄各病人的密碼。

因為 **pat** 代表一個特定的病人，可決定唯一的 **p_ID**、**p_name**、**p_gender**、**p_num**、**p_email** 和 **p_pass**，又 **pat** 是一個 **Candidate Key**，所以此表格符合 **3NF** 和 **BCNF**。此外，**p_ID**、**p_num** 和 **p_email** 具有唯一限制，確保資料唯一性且無部分依賴。

4. Record 表格

```
CREATE TABLE `record` (
        `case_ID` int(11) NOT NULL,
        `d_ID` text NOT NULL,
        `p_ID` text NOT NULL,
        `date` date NOT NULL,
        `room` int(11) NOT NULL,
        `diagnosis` text NOT NULL,
        `treatment` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
ALTER TABLE `record`
        ADD PRIMARY KEY (`case_ID`);
ALTER TABLE `record`
        MODIFY `case_ID` int(11) NOT NULL AUTO_INCREMENT,
       AUTO_INCREMENT=27;
```

case_ID	d_ID	p_ID	date	room	diagnosis	treatment
18	d25	p3	2024-12-01	303	anxiety disorder	medication and psychotherapy
20	d27	p2	2024-12-01	124	hypertension	medication and lifestyle changes
23	d30	p4	2024-12-10	205	leg injury	cast on leg

- **[case_ID]** 記錄每個病歷的編號，為 **Primary Key**。
- **[d_ID]** 記錄負責醫生的編號。
- **[p_ID]** 記錄病人的編號。
- **[date]** 記錄就診的日期。
- **[room]** 記錄病人接受治療的房間號碼。
- **[diagnosis]** 記錄病人的診斷結果。
- **[treatment]** 記錄病人所接受的治療方法。

因為 **case_ID** 代表一個特定的病歷紀錄，可決定唯一的 **d_ID**、**p_ID**、**date**、**room**、**diagnosis** 和 **treatment**，又 **case_ID** 是一個 **Candidate Key**，所以此表格符合 **3NF** 和 **BCNF**。

四、 將系統功能以SQL DML定義

A. Add Function

1. Add Doctor

```
const insertDoctorQuery = 'INSERT INTO doctors (d_name, d_department, d_num,
    d_email, d_pass) VALUES (?, ?, ?, ?, ?)';
Server.query(insertDoctorQuery, [d_name, d_department, d_num, d_email, d_pass],
    (err, result)
```

2. Add Patients

```
const registerPatientQuery = 'INSERT INTO patients (p_name, p_gender, p_num,
    p_email, p_pass) VALUES (?, ?, ?, ?, ?)';
Server.query(registerPatientQuery, [p_name, p_gender, p_num, p_email, p_pass], (err,
    result)
```

3. Add Record

```
const insertRecordQuery = 'INSERT INTO record (d_ID, p_ID, date, room, diagnosis,
    treatment) VALUES (?, ?, ?, ?, ?, ?)';
Server.query(insertRecordQuery, [d_ID, p_ID, date, room, diagnosis, treatment], (err,
    result)
```

B. Delete Function

1. **Delete doctor**

```
const deleteDoctorQuery = "DELETE FROM doctors WHERE d_ID = ?";
  Server.query(deleteDoctorQuery, [doctorId], (err, result)
```

C. Update Function

1. **Update doctor**

```
const updateDoctorQuery = `UPDATE doctors SET d_name = ?, d_pic = ?,
  d_department = ?, d_num = ?, d_email = ? WHERE d_ID = ?`;
Server.query(updateDoctorQuery, [d_name, pictureBuffer, d_department, d_num,
  d_email, d_ID], (err, result)
```

2. U

D. Join and Aggregation Function

1. **To fetch doctors for a specified department (join)**

```
const fetchDoctorsQuery = `
  SELECT d_name, d_pic
  FROM doctors
  JOIN departments ON doctors.d_department = departments.dpt_name
  WHERE departments.dpt_name = ?`;
 Server.query(fetchDoctorsQuery, [department], (err, result)
```

2. **To fetch record by patient ID (join)**

```
const patientId = req.query.patientId;
 const fetchPatientQuery = `
  SELECT
   record.case_ID, doctors.d_name, record.date, record.room, record.diagnosis,
record.treatment
  FROM
   record
  JOIN
   doctors
  ON
```

```
    record.d_ID = doctors.d_ID
  WHERE
    record.p_ID = ?`;
 Server.query(fetchPatientQuery, [patientId], (err, result)
```

3. To fetch record by doctor ID (join)

```
const fetchDoctorQuery = `
  SELECT
    record.case_ID, record.p_ID, patients.p_name, record.date, record.room,
record.diagnosis, record.treatment
  FROM
    record
  JOIN
    patients
  ON
    record.p_ID = patients.p_ID
  WHERE
    record.d_ID = ?`;
 Server.query(fetchDoctorQuery, [doctorId], (err, result)
```

4. To search records (join)

```
const searchRecordsQuery = `
  SELECT record.case_ID, record.p_ID, patients.p_name, record.room,
record.diagnosis, record.treatment, record.date
  FROM record
  LEFT JOIN patients ON record.p_ID = patients.p_ID
  WHERE record.d_ID = ? AND (
    record.case_ID LIKE ?
    OR record.p_id LIKE ?
    OR patients.p_name LIKE ?
    OR record.room LIKE ?
    OR record.diagnosis LIKE ?
    OR record.treatment LIKE ?
```

```
    OR record.date LIKE ?) `;
  const searchTerm = `%${keyword}%`;
  Server.query(searchRecordsQuery, [doctorId, searchTerm, searchTerm,
searchTerm, searchTerm, searchTerm, searchTerm, searchTerm], (err, results)
```

5. Doctor add record of patient (aggregation)

```
  const checkPatientQuery = 'SELECT COUNT(*) AS count FROM patients
WHERE p_ID = ?';
  Server.query(checkPatientQuery, [p_ID], (err, result)
```