**Doctor Appointment System**

Software Engineering: Catherine Woods

Submitted By: David O’ Connor (TL\_KCOMP\_GY6)

Submitted on:

Table of Contents

[**1.** **Introduction** 4](#_Toc406751898)

[**2.** **Functional Components** 5](#_Toc406751899)

[**3.** **User Requirements** 6](#_Toc406751900)

[**3.1.** **The system performs patient administration** 6](#_Toc406751901)

[**3.2.** **The system performs Doctor Administration** 6](#_Toc406751902)

[**3.3.** **The system performs appointment management** 6](#_Toc406751903)

[**3.4.** **The System performs Finance management** 6](#_Toc406751904)

[**4.** **System Requirements** 7](#_Toc406751905)

[**4.1.** **System Level Use Case Diagram** 7](#_Toc406751906)

[**4.2.** **Patient Administration** 8](#_Toc406751907)

[**4.2.1.** **Register Patient** 8](#_Toc406751908)

[**4.2.2.** **De-Reg Patient** 10](#_Toc406751909)

[**4.2.3.** **Amend Patient** 12](#_Toc406751910)

[**4.2.4.** **List Patients** 14](#_Toc406751911)

[**4.3.** **Doctor Administration** 15](#_Toc406751912)

[**4.3.1.** **Register Doctor** 15](#_Toc406751913)

[**4.3.2.** **De-Reg Doctor** 17](#_Toc406751914)

[**4.4.** **Appointment Management** 19](#_Toc406751915)

[**4.4.1.** **Create Appointment** 19](#_Toc406751916)

[**4.4.2.** **Cancel Appointment** 21](#_Toc406751917)

[**4.4.3.** **Show Doctor’s Schedule** 23](#_Toc406751918)

[**4.4.4.** **Check-in Patient** 25](#_Toc406751919)

[**4.5.** **Finance Management** 26](#_Toc406751920)

[**4.5.1.** **Record Payment** 26](#_Toc406751921)

[**4.5.2.** **Issue Invoice** 28](#_Toc406751922)

[**5.** **System Model** 30](#_Toc406751923)

[**5.1.** **Data Flow Diagram Elements** 30](#_Toc406751924)

[5.1.1. External entities 30](#_Toc406751925)

[5.1.2. Data stores 30](#_Toc406751926)

[5.1.3. Processes 30](#_Toc406751927)

[**5.2.** **Level-0 DFD** 31](#_Toc406751928)

[**5.3.** **Level\_1 DFD** 32](#_Toc406751929)

[**5.4.** **Level\_2 DFD** 33](#_Toc406751930)

[5.4.1. Process P1 33](#_Toc406751931)

[5.4.2. Process P2 34](#_Toc406751932)

[5.4.3. Process P3 35](#_Toc406751933)

[5.4.4. Process P4 36](#_Toc406751934)

[**6.** **Data Model** 37](#_Toc406751935)

[Doctor Appointment System Data Model: 37](#_Toc406751936)

[**7.** **Database Schema** 38](#_Toc406751937)

[**7.1** **Schema: Doctor Appointment System** 38](#_Toc406751938)

[**8.** **Conclusion** 40](#_Toc406751939)

# **Introduction**

The system we will be evaluating and designing will be for a Doctor’s practice. The system will allow for the basic functions of registering new patients and doctors, creating appointments, and handling payments for appointments. The system will be able to carry out a few more functions which may be needed in a Doctor Appointment system.

Accompanying this paper is a prototype of the system for programmers and developers to reference. Created in Microsoft Visual Studio using C#, this prototype gives an example of a working system.

# **Functional Components**

# **User Requirements**

The following user requirements have been identified for the system (Doctor Appointment Sys):

## **The system performs patient administration**

* + 1. Doctor Appointment Sys will allow the registration of a new patient to be recorded
    2. Doctor Appointment Sys will allow the de-registration of existing patient from records
    3. Doctor Appointment Sys will allow for amendment of patient information
    4. Doctor Appointment Sys will allow access and display of patient list

## **The system performs Doctor Administration**

* + 1. Doctor Appointment Sys will allow the registration of a new Doctor to be recorded
    2. Doctor Appointment Sys will allow the de-registration of an existing Doctor from record

## **The system performs appointment management**

* + 1. *Doctor Appointment Sys will record new appointment*
    2. *Doctor Appointment Sys will amend appointment if canceled*
    3. *Doctor Appointment Sys will generate a listing of doctor’s’ schedule*
    4. *Doctor Appointment Sys will record the check-in of a patient*

## **The System performs Finance management**

* + 1. *Doctor Appointment Sys will record any required payment from Patient*
    2. *Doctor Appointment Sys will generate an invoice for unpaid appointments*

# **System Requirements**

The system requirements relating to the user requirements identified in section 3 are defined by:

## **System Level Use Case Diagram**

**Doctor Appointment System**

Administrator

Patient Admin

Patient

Finance

Appointments

Doctor Admin

Doctor

## **Patient Administration**

This module of the Doctor Appointment System deals with handling patient administration (Register, de-register, amend info, and list patients).

### **Register Patient**

Administrator

Patient

<<Includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Register Patient** | |
| **Use Case Id** | 01 | |
| **Priority** | 04 | |
| **Source** | Administrator | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** |  | |
| **Description** | Every patient must be registered on the system. This function creates a new entity in the Patient File with required patient information. | |
| **Preconditions** | Patients under the age of 18 must be accompanied adult/guardian. | |
| **Trigger** | The creation or deletion of a new entity in the database. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Registering a patient** | **Step 1:** New patient requests to register with the practice.  **Step 2:** Admin requests to register a patient to the system.  **Step 4:** Admin fills out the required information.   * Forename * Surname * Address * Date of birth * Allergies | **Step 3:** System displays the register patient UI.  **Step 5:** System creates a unique Patient Id for the new patient.  **Step 6:** System validates data entry.   * All fields are required * DoB must not be in future   **Step 7:** Registration Date is assigned System Date.  **Step 8:** Stores patient information in the Patient File.  **Step 9:** Patient Status is assigned a default value ‘Registered’.  **Step 9:** Confirmation message displayed. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entry** | **Step 9:** Admin re-enters the invalid data entry | **Step 7:** System displays appropriate error message.  **Step 8:** System opens the register patient UI with the invalid field highlighted. |
| **Conclusions** | The creation of unique patient Id and update of the Patient File. | |
| **Post conditions** | Only the patient may provide the information for their record. | |
| **Business Rules** | Identity verification is required for registration. | |
| **Implementation Constraints** |  | |

### **De-Reg Patient**

Administrator

Patient

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **De-register Patient** | |
| **Use Case Id** | 02 | |
| **Priority** | 06 | |
| **Source** | Administrator | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Patient | |
| **Description** | A patient may wish to de-register themselves from the practice. | |
| **Preconditions** | Patient must be registered on the system before de-registration may occur. | |
| **Trigger** | The removal of patient information stored on the database. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **De-register patient using PatientID** | **Step 1:** Patient requests to de-register from the practice  **Step 2:** Admin selects de-register a patient.  **Step 4:** Admin chooses the option to de-register a patient using patient surname or patient Id.  **Step 7:** The user confirms that the patient is to be de-registered | **Step 3:** System displays de-register patient UI.  **Step 5:** System validates the input data:   * Either Patient ID or Surname must be entered * Both PatientID and Surname may not be entered   **Step 6:** If PatientID entered, the system retrieves the patient details from the Patient File using Patient ID and displays on the UI  **Step 8:** Patient status is updated in the Patient File and set to ‘De-registered’.    **Step 7:** Confirmation message displayed. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **De-register patient using Patient Surname** | **Step 7:** The user selects the required patient from the list displayed.  **Step 9:** The user confirms that the patient is to be de-registered | **Step 6:** If Surname is entered, the system retrieves summary of all patient details from the Patient File with the Patient Surname entered and displays on the UI  **Step 8:** The system retrieves the details for the selected patient and displays on the UI  **Step 10:**  Patient status is updated in the Patient File and set to ‘De-registered’.    **Step 11:** Confirmation message displayed. |
| **Patient is not found in Patient File** | **Step 7:** Admin re-enters patient name or Id | **Step 6:** System displays appropriate error message |
| **Conclusions** | Patient information is updated in the Patient File. | |
| **Post conditions** | Only the patient may request to be de-registered from the practice. | |
| **Business Rules** | Patient identity verification is required. | |
| **Implementation Constraints** |  | |

### **Amend Patient**

Administrator

Patient

<<Includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Amend Patient** | |
| **Use Case Id** | 03 | |
| **Priority** | 05 | |
| **Source** | Administrator | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Patient | |
| **Description** | this function amends a patients details | |
| **Preconditions** | Patient must be registered on the system before information can be amended. | |
| **Trigger** | patient requests data to be changed | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Amend patient information** | **Step 1:** Patient requests update of information.  **Step 2:** Admin requests to amend a patient’s information on the Patient File.  **Step 4:** Admin chooses fields to amend.   * Forename * Surname * Address * Date of birth * Allergies | **Step 3:** System displays patient information UI.  **Step 5:** System validates the new information provided.  **Step 6:** Patient File is updated with new information.  **Step 7:** Confirmation message displayed. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid data entered** | **Step 9:** Admin re-enters the invalid data entry | **Step 6:** System displays appropriate error message.  **Step 8:** System opens the register patient UI with the invalid field highlighted. |
| **Conclusions** | Patient information is amended on the Patient File. | |
| **Post conditions** | Only the patient may provide new information for their record. | |
| **Business Rules** | Patient cannot be amended if de-registered | |
| **Implementation Constraints** |  | |

### **List Patients**

Administrator

**Example Listing**

**PatID** **Name** **Street Date of Birth**

1501 Adams, Betty The Square 15/03/1976

1024 Jones, John 23 High Street 12/01/1954

1178 Smith, Mary Laharn 24/08/1988

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List Patients** | |
| **Use Case Id** | 04 | |
| **Priority** | 06 | |
| **Source** | Administrator | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** |  | |
| **Description** | Administrator may wish to print/display a list of current patients registered at the practice. | |
| **Preconditions** |  | |
| **Trigger** | The display and/or print of a patient list from the Patient File. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Display Patient List** | **Step 1:** Admin requests patient list UI. | **Step 2:** System opens patient list UI.  **Step 3:** System displays a list of all registered patients and information from Patient file.   * Forename * Surname * Address * Date of birth * Allergies |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Print patient list** | **Step 4:** Admin requests a print of patient list. | **Step 5:** System sends list to default printer. |
| **Conclusions** | Patient list is displayed showing patient information. This list will be displayed on the monitor and has an option to be printed out. | |
| **Post conditions** | No editing of information allowed in this module of the system. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## **Doctor Administration**

The system allows for registering a new doctor who is moving to work in the practice. Also the ability to de-register a doctor who is leaving the practice. The doctor records must be amended is each case.

### **Register Doctor**

Administrator

Doctor

<<extends>>

<<Includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Register Doctor** | |
| **Use Case Id** | 05 | |
| **Priority** | 01 | |
| **Source** | Administrator | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** |  | |
| **Description** | Every Doctor at the practice must be registered on the system. | |
| **Preconditions** |  | |
| **Trigger** | The creation of a new entity in the Doctor File. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Registering a Doctor** | **Step 1:** New Doctor requests to register with the practice.  **Step 2:** Admin requests to register a Doctor to the system.  **Step 4:** Admin fills out the required information.   * Forename * Surname * Address * Date of birth * Salary * Work Hours | **Step 3:** System displays the register Doctor UI.  **Step 5:** System creates a unique Doctor Id for the new Doctor.  **Step 6:** System validates data entry.   * All fields are required * DoB must not be in future   **Step 7:** Registration Date is assigned System Date.  **Step 8:** Stores Doctor information in the Doctor File.  **Step 9:** Doctor Status is assigned a default value ‘Registered’.  **Step 9:** Confirmation message displayed. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entry** | **Step 9:** Admin re-enters the invalid data entry | **Step 7:** System displays appropriate error message.  **Step 8:** System opens the register Doctor UI with the invalid field highlighted. |
| **Conclusions** | The creation of unique Doctor Id and update of the Doctor File. | |
| **Post conditions** | Only the said Doctor may provide the information for their record. | |
| **Business Rules** | Identity verification is required for registration. | |
| **Implementation Constraints** |  | |

### **De-Reg Doctor**

Administrator

Doctor

<<extends>>

<<Includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **De-Reg Doctor** | |
| **Use Case Id** | 06 | |
| **Priority** | 02 | |
| **Source** | Administrator | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Doctor | |
| **Description** | A Doctor may wish to de-register themselves from the practice. | |
| **Preconditions** | Doctor must be registered on the system before de-registration may occur. | |
| **Trigger** | The removal of Doctor information stored on the database. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **De-register Doctor** | **Step 1:** Doctor requests to de-register from the practice  **Step 2:** Admin requests to amend a Doctor’s information on the Doctor File.  **Step 4:** Admin chooses the option to de-register a Doctor using Doctor name and Doctor Id. | **Step 3:** System displays Doctor information UI.  **Step 5:** System validates that Doctor name and Doctor Id is in Doctor File.  **Step 6:** Doctor information is removed from the Doctor File.    **Step 7:** Confirmation message displayed. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Doctor is not found in Doctor File** | **Step 7:** Admin re-enters Doctor name or Id | **Step 6:** System displays appropriate error message |
| **Conclusions** | Doctor information is removed from the Doctor File. | |
| **Post conditions** | Only the Doctor may request to be de-registered from the practice. | |
| **Business Rules** | Doctor identity verification is required. | |
| **Implementation Constraints** |  | |

## **Appointment Management**

The system will allow for the creation of appointments. Each appointment will have an option for the cancellation of appointments if required. The appointments must refer to the doctor’s schedule to avoid conflicts. If an appointment time must change, system will allow for the cancelation of the appointment and a new one made through the create appointment module. The appointment module must be updated when a patient has checked in.

### **Create Appointment**

Administrator

Patient

<<extends>>

<<Includes>>

* Name
* Patient ID
* Date
* Time
* Doctor
* Fee

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Create Appointment** | |
| **Use Case Id** | 07 | |
| **Priority** | 01 | |
| **Source** |  | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Patient | |
| **Description** | This module creates an appointment for the patient to see the doctor. This shall include Patient name, ID, Date, Time, which doctor, and the fee for the appointment. | |
| **Preconditions** | Appointments cannot overlap. Patient must agree upon time and date before appointment can be recorded. | |
| **Trigger** | Creates an appointment in the Appointment File. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Create Appointment** | **Step 1:** Admin requests to create an appointment.  **Step 4:** Admin enters the required details:   * Patient ID * Doctor * Date/time   **Step 6:** User selects the desired appointment time  **Step 7:** Admin confirms that appointment is to be made | **Step 2:** System Retrieves details of all doctors and displays on UI in order of name  **Step 3:** System retrieves summary details of all patients from the Patient File and displays on UI in order of name  **Step 5:** System retrieves available appointment times for date and doctor specified and displays on UI  **Step 8:** Set appointment status ‘Pending’ as default  **Step 9:** Fee is set to zero  **Step 10:** System saves appointment details in Appointments file  **Step 11:** System displays confirmation message  **Step 12:** System resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data entry** | **Step 6:** Admin re-enters required appointment data. | **Step 5:** System displays appropriate error message |
| **Conclusions** | Appointment is recorded in the appointments file | |
| **Post conditions** |  | |
| **Business Rules** | Hours available for appointments must fall within doctors’ work hours. | |
| **Implementation Constraints** |  | |

### **Cancel Appointment**

Administrator

Patient

<<extends>>

<<Includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Cancel Appointment** | |
| **Use Case Id** | 08 | |
| **Priority** | 02 | |
| **Source** |  | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Patient | |
| **Description** | Allows for the cancellation of an appointment. This will remove the entry in the appointments file for the requested cancellation. | |
| **Preconditions** | Appointment must already be recorded. | |
| **Trigger** | Removal of appointment from Appointments File | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Cancel Appointment** | **Step 1:** Admin requests to cancel appointment.  **Step 3:** Admin selects, using patient name and ID, an appointment to cancel. | **Step 2:** System displays Appointment UI.  **Step 4:** System validates data   * Valid Patient name and ID   **Step 5:** System displays a confirmation message, (are you sure?)  **Step 6:** System removes the appointment entry from the Appointment file. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data entered** | **Step 6:** Admin re-enters Patient name and patient ID. | **Step 5:** System displays appropriate error message. |
| **Conclusions** | Appointment record is removed from Appointment file | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Show Doctor’s Schedule**

Administrator

Doctor

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Show Doctor’s Schedule** | |
| **Use Case Id** | 09 | |
| **Priority** | 07 | |
| **Source** |  | |
| **Primary Business Actor** | Admin | |
| **Other Participating Actors** | Doctors | |
| **Description** | This module displays a list of appointments a doctor has on his schedule. The list will show appointment date and time, patient name and ID, for the requested doctor. | |
| **Preconditions** |  | |
| **Trigger** | Display/print list of a doctor’s appointments within a given time frame. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Show doctor schedule** | **Step 1:** Actor requests doctor schedule UI.  **Step 3:** Actor selects registered doctor name and doctor ID. | **Step 2:** System displays Doctor schedule UI.  **Step 4:** System retrieves from appointment file, the appointments set to the requested Doctor ID.  **Step 5:** System displays, in list form, appointments scheduled for the Doctor in the requested time frame. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Print Schedule** | **Step 6:** Actor chooses to print out list. | **Step 7:** System sends the list to the default local printer. |
| **Conclusions** | A list of appointments registered to a Doctor is displayed and/or printed. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Check-in Patient**

Administrator

Patient

<<extends>>

<<Includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Check-in Patient** | |
| **Use Case Id** | 10 | |
| **Priority** | 04 | |
| **Source** |  | |
| **Primary Business Actor** | Admin | |
| **Other Participating Actors** | Patient | |
| **Description** | When a patient arrives for scheduled appointment, the appointment status will be updated in the appointments file. | |
| **Preconditions** | Patient must be present at practice. | |
| **Trigger** | Updates appointments file to set status to complete. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Check-in patient** | **Step 1:** Patient arrives at practice.  **Step 2:** Admin requests appointment UI.  **Step 4:** Admin selects to check in the patient using their Patient name and ID. | **Step 3:** System displays appointment UI.  **Step 5:** System validates data   * Valid patient entity   **Step 6:** System amends Appointment file and sets status to ‘Complete’  **Step 7:** Confirmation message displayed. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid data entered** | **Step 7:** Admin re-enters patient ID | **Step 6:** System displays appropriate error message. |
| **Conclusions** | Appointment file is updated with status set to ‘Complete’ | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## **Finance Management**

The system will have a module to handle finance management. This module will have the function to record a payment and issue an invoice. This module will have a Record file for database to monitor due payments and to set payment status from due to paid when done so.

### **Record Payment**

Administrator

Patient

<<extends>>

<<Includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Record Payment** | |
| **Use Case Id** | 11 | |
| **Priority** | 03 | |
| **Source** |  | |
| **Primary Business Actor** | Admin | |
| **Other Participating Actors** | Patient | |
| **Description** | This will accommodate the record payment function. From the appointments file, a fee will be input from the admin when an appointment is created. This will then be forwarded to the payment file when the patient has been checked-in | |
| **Preconditions** | Patient must be checked in. Fee must be in Euros. | |
| **Trigger** | A payment file will be updated with fees for the patient who has been checked-in | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Record Payment** | **Step 1:** Admin checks in patient.  **Step 4:** Admin receives payment from patient after meeting doctor. Admin updates payment file with amount paid. | **Step 2:** System receives data from the Fee column in the appointments file and updates the payment file for the patient.  **Step 3:** System sets payment status to default, ‘pending’.  **Step 5:** System validates amount paid equals amount due.  **Step 6:** Confirmation message displays amount paid and amount due to admin.  **Step 7:** System updates payment status to ‘paid’ |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Not paid in full** |  | **Step 6:** System records outstanding payment.  **Step 7:** Updates payment file with new fee due. |
| **Conclusions** | A record of payment is created. This record will indicate the status of payment whether it was paid or still outstanding. | |
| **Post conditions** | If still outstanding, data gets sent to the Issue invoice module | |
| **Business Rules** | Fees must be collected, preferably at time of patient check-out | |
| **Implementation Constraints** | Patient may not have all or any of the money at time of check out. An invoice shall be sent out in this case. | |

### **Issue Invoice**

Administrator

Patient

<<extends>>

<<Includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Issue Invoice** | |
| **Use Case Id** | 12 | |
| **Priority** | 07 | |
| **Source** | Record Payment | |
| **Primary Business Actor** | Admin | |
| **Other Participating Actors** |  | |
| **Description** | When a patient has not paid any or full amount of fees due, an invoice shall then be sent out to the patient’s home address indicating amount due, date due, and various methods of payment. This will be carried out at the end of every working week. | |
| **Preconditions** | Patient must have outstanding fees. | |
| **Trigger** | Create a new entry into the Invoice File which holds the information needed to post out due fees to patients. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| **Issue Invoice** | **Step 1:** Admin chooses Invoice UI.  **Step 3:** Admin chooses to print out the invoice for patients with outstanding fees for the week.  **Step 5:** Admin posts out invoices | **Step 2:**  System displays Invoice UI. This displays list of patients with outstanding fees. It shall include:   * Patient name * Patient Address * Amount due * Date due * Different payment methods   **Step 4:** System sends invoices to default printer.  **Step 6:** System updates invoice file with status ‘sent’ for sent invoices. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | A paper invoice is sent out to patients with outstanding fees. System updates the invoice file with status ‘sent’ | |
| **Post conditions** | Patients with overdue payments must also be informed of missed payment with threat of additional charges on their fee. This is found by patients with status pending after their payment due date. | |
| **Business Rules** | Business must inform of preferred payment methods. | |
| **Implementation Constraints** | Only done on Friday morning. This allows for the week to be catalogued effectively and batch process the invoices. | |

# **System Model**

## **Data Flow Diagram Elements**

### External entities

* Patients
* Doctors

### Data stores

* D1: Patient File
* D2: Doctor File
* D3: Appointment File
* D4: Payment File

### Processes

#### P1: Process Patients

P1.1: Register Patient

P1.2: Amend Patient

P1.3: De-Reg Patient

P1.4: List Patients

#### P2: Process Doctors

P2.1: Register Doctor

P2.2: De-Reg Doctor

#### P3: Process Appointments

P3.1: Create Appointment

P3.2: Cancel Appointment

P3.3: Show Doctor’s Schedule

P3.4: Check-in Patients

#### P4: Process Finance

P4.1: Record Payment

P4.2: Issue Invoice

## **Level-0 DFD**

Patient

Doctor Appointment System

## **Level\_1 DFD**

Doctor

Patient

P1

Process Patients

Patient Details

Patient

Request Appointment

Assign Appointment

P3

Process

Appointments

P2

Process Doctors

Doctor Details

Appointment Details

Doctor Details

Payment Details

D3

Appointments File

D2

Doctor File

Patient Details

D4

Payment File

Patient Details

Payment Details

Payment Details

D1

Patient File

P4

Process

Finance

Payment Details

Payment

Issue Invoice

Patient

## **Level\_2 DFD**

### Process P1

Request De-registration

Patient

De-register Patient

Patient

P1.3

De-Reg Patient

Patient Details

D1

Patient File

Patient Details

P1.1

Register Patient

Patient Details

Patient Details

Amend Details

P1.2

Amend Patient

P1.4

List Patients

Patient

### Process P2

Doctor

Doctor

Request De-registration

Doctor Details

D2

Doctor File

De-register Doctor

P2.1

De-Reg Doctor

P2.1

Register Doctor

Doctor Details

### Process P3

D4

Payment File

Doctor

Doctor Details

Payment Details

D2

Doctor File

Request Schedule

Patient

P3.3

Show Doctor’s Schedule

D3

Appointment File

Request Appointment

Appointment Details

Appointment Details

Appointment Details

P3.2

Cancel Appointment

P3.1

Create Appointment

Patient

Request Cancellation

Patient Details

P3.4

Check-in Patient

D1

Patient File

Patient Details

### Process P4

Invoice

Patient

Payment

P4.2

Issue Invoice

P4.1

Record Payment

Patient Details

Invoice Status

Payment Details

D1

Patient File

Invoice Details

D4

Payment File

# **Data Model**

## Doctor Appointment System Data Model:

Patient

PatID {PK}

Forename

Surname

DoB

Street

Town

County

Allergies

Phone

Doctor

DocID {PK}

Forename

Surname

Street

Town

County

DoB

Salary

Phone

Payment

PayID {PK}

AppID

PayDate

Amount

0…\*

1

Is assigned to

🡪

1

1…\*

Is for

🡪

0…\*

1

🡪

Is assigned to

Appointment

AppID {PK}

PatID

DocID

Time

Date

# **Database Schema**

## **Schema: Doctor Appointment System**

**Relation: Patients**

Attributes: PatID numeric(8)

Forename varchar(20) NOT NULL

Surname varchar(20) NOT NULL

DoB Date NOT NULL

Street varchar(25) NOT NULL

Town varchar(20) NOT NULL

County varchar(20) NOT NULL

Alergies varchar(50) NOT NULL

Phone numeric(10) NOT NULL

Primary Key: PatID

**Relation: Doctors**

Attributes: DocID numeric(8)

Surname varchar(20) NOT NULL

Forename varchar(20) NOT NULL

DoB Date NOT NULL

Street varchar(25) NOT NULL

Town varchar(20) NOT NULL

County varchar(20) NOT NULL

Salary Decimal(5, 2) NOT NULL

Phone numeric(10) NOT NULL

Primary Key: DocID

**Relation: Appointments**

Attributes: AppID numeric(10)

AppDate Date NOT NULL

AppTime Timestamp NOT NULL

PatID numeric(8) NOT NULL

DocID numeric(8) NOT NULL

Primary Key: AppID

Foreign Key: PatID REFERENCES Patients

Foreign Key: DocID REFERENCES Doctors

**Relation: Payments**

Attributes: PayID numeric(10)

PayDate Date NOT NULL

Amount Decimal(5, 2) NOT NULL

AppID numeric(10) NOT NULL

Primary Key: PayID

Foreign Key: AppID REFERENCES Appointments

# **Conclusion**

In this paper I analyzed the requirements needed in a doctor appointment system. Through these requirements I then started to design how the system will incorporate the necessary database. Through the design of the prototype I found a few design flaws in my original ideas such as only needing to search a patient’s surname instead of their Patient ID and surname. Flaws like this were not obvious to me at first but through running the prototype, it became clear as to what was needed in the various areas of the system.