

**Software Engineering and Testing. BSC Year 2, 2024/2025**

**(Assignment 3 - 20%)**

**Assessment 3: Design and Draft Implementation**

**Submitted by:**

**Mark Hopkins – B00160560**

**Submission date**

**24/03/2025**

**Declaration**

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ordinary Degree in Computing in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated.

Author: \_\_\_\_\_\_\_**Mark Hopkins**\_\_\_\_\_\_\_ Dated: \_**23/03/2025**\_

Author: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_

**Table of Contents**

# Title: Country GP

# Project Definitions

The purpose of this document is the creation of a general practitioner's (GP) office that makes managing patients, scheduling appointments, and monitoring medications simple and user friendly. The application was created with PHP and is hosted on a XAMPP localhost server. phpMyAdmin is used to hold and pull from the databases used. The project's primary objective is to provide an easy-to-use system that makes it simple for office workers to handle patient data, appointments, and medications. In one location, staff members can view medication orders and access patient records. Real-time appointment tracking, medication management, and a structured database that connects all required data are some of the system's primary features. The six tables in the database, account, appointment, , faq, employee, medications, and patient, are connected to one another to provide seamless communication. For instance, the medication table keeps track of each patient's prescriptions, while the account table holds the login details to each account. This system is intended to increase the efficiency of routine operations at a general practitioner's office by offering a simple instrument for handling administrative and medical procedures.

# Document Revision

Rev. 1.0 date – [Draft Proposal.docx](https://tudublin-my.sharepoint.com/:w:/r/personal/b00163440_mytudublin_ie/Documents/Country%20GP/Lab%201/Draft%20Proposal.docx?d=w1ffe698326214b12b0d050966b7b204f&csf=1&web=1&e=f5WUGJ)

# Methodology

System models – UML:

UML is a way to visualize, design, and document the structure and behaviour of a system using diagrams, helping members of the project communicate system designs and plan development. In this project, two UML diagrams are used: a Use Case Diagram, which illustrates how users interact with the system and achieve their goals, and a Class Diagram, which shows the system's structure, including classes, attributes, methods, and relationships. These diagrams help in system communication, documentation, and design to ensure clear understanding and reduce development errors.

Use of, and necessity of OOAD:

OOAD is used extensively in the code to structure the system. The system is organized and managed using classes like Appointment, Patient, Account, Employee, and Medications. Each class is in charge of particular duties and aids in problem isolation, which facilitates system management.

Purpose of using classes / What is a class diagram?:

Each class in the system encapsulates both data and behaviour, like the Appointment class that manages appointment details and methods to handle them, while classes such as Patient and Account help organize related functionality, with reusable methods like addPatient() used across multiple pages; a class diagram would visualize these relationships, showing how the Appointment class links to both Patient and Employee, and how the Patient class connects to the Account class to represent the relationship between patients and user accounts.

Static Versus Dynamic Case Diagrams?:  
Static Case Diagrams show the system's structure, focusing on how classes like Patient, Appointment, and Employee are organized and how their data is structured. For example, it would show how the Appointment class connects to Patient and Employee, or how Patient links to Account.

Dynamic Case Diagrams illustrate how the system reacts over time, like when a user creates an appointment. It shows how the form data is sent to the server, processed by methods like addAppointment(), and stored in the database for future use. These diagrams help visualize data flow and system behaviour.

What is an ERD?:  
An Entity-Relationship Diagram is used to visualize the database structure and show how entities like Patient, Employee, Appointment, and Account are related. For example, the Appointment entity connects to both Patient and Employee, as each appointment involves both. A Patient can have multiple appointments, and an Employee can handle several. The Patient entity may also link to the Account entity, where each patient has a user account.

The ERD helps in designing a clear database structure and understanding the relationships between entities.

Purpose of using classes?:

Grouping related data and functionality into classes, such as the Patient class managing patient data and the Appointment class handling scheduling, helps organize the system, while encapsulating both data and methods within each class ensures a clean structure, and promotes code reusability by allowing methods like addPatient() to be used across multiple parts of the system.

Volatile versus Persistent storage – Object Instances / Database?:

Volatile storage refers to temporary data stored in memory, like when objects such as Patient or Appointment are created during user interaction, but are lost if the page refreshes or the server restarts. Persistent storage, on the other hand, stores important data in a database (e.g., MySQL), ensuring that details like patient records and appointments are saved even after the page reloads or the server restarts. This difference ensures that data is kept safe and accessible across sessions.

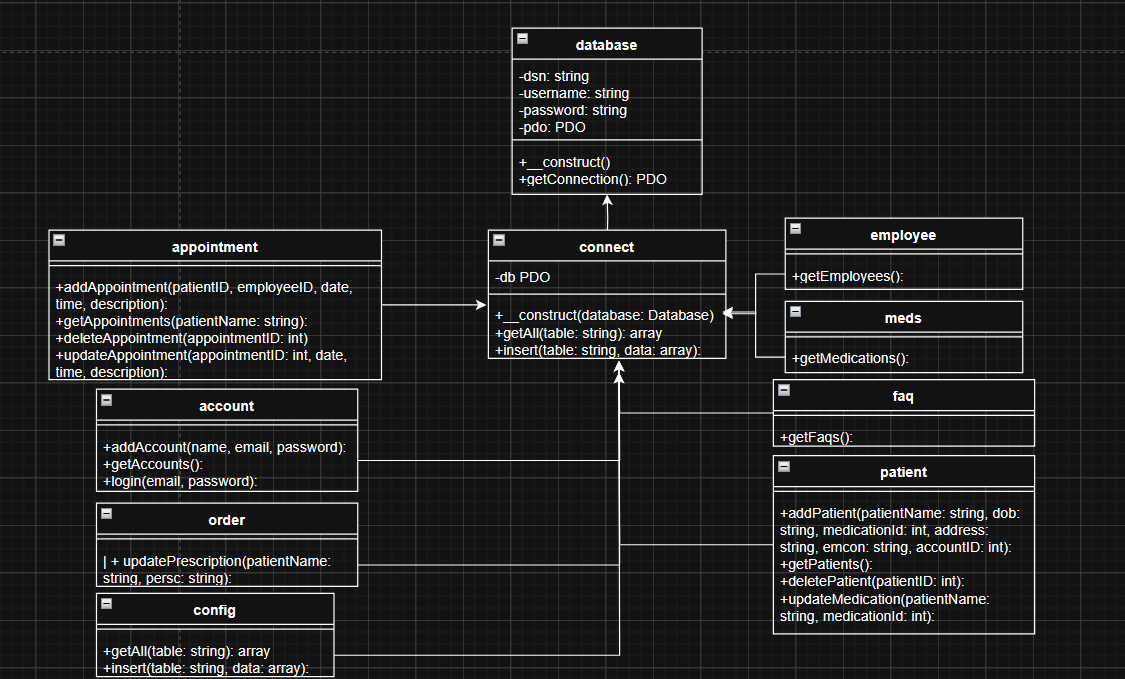
User Interface template chosen and how it can aid in executing the functional specification of the project.:  
The user interface for the project, built with a HTML template styled with CSS, includes data entry forms for managing patient records and appointments, input validation for accuracy, and consistent navigation elements like headers and footers to ensure a user-friendly and functional experience.

1. **Requirements**

4.1 Use Cases - [useCaseDiagram.docx](https://tudublin-my.sharepoint.com/:w:/r/personal/b00163440_mytudublin_ie/Documents/Country%20GP/Lab%203/useCaseDiagram.docx?d=w8572b08c4a2845abbae94f684a2d9a05&csf=1&web=1&e=0c9WDq)

4.2 Use Case Specifications - [Use Case Spec.docx](https://tudublin-my.sharepoint.com/:w:/r/personal/b00163440_mytudublin_ie/Documents/Country%20GP/Lab%204/Use%20Case%20Spec.docx?d=w9121893da6d74fe49ac9b9a00a5c8e48&csf=1&web=1&e=ychltL)

1. **Case Diagrams**

**Class Diagram** – 

Our class diagram shows how the database system is organized, including tables for managing appointments, accounts, orders, configurations, employees, medications, FAQs, and patients. Each table has methods for adding, updating, and retrieving data, all connected through a central database class. The classes define important attributes like connection details and methods for handling data, highlighting how different parts of the system interact with each other.

**Entity Relationship Diagram** – A computer screen shot of a computer flowchart

AI-generated content may be incorrect.

The Entity Relationship Diagram was designed to clearly show the relationships between different entities in the database. The Account entity includes fields like account ID, name, email, and password to handle user login and account management, it holds a 1-to-1 relationship with the patient table. The Patient entity, identified by a unique patient ID, stores basic details like the patient’s name, date of birth, and contact information. It also connects to both the Account and Medication entities, linking the patient’s account and their prescribed medications.

The Appointment entity is used to connect patients with healthcare employees, tracking important information like the appointment date, status, and which employees are involved. This ensures that patient visits are properly recorded. The Medications entity lists various medications with descriptions, helping to manage prescriptions for patients.

Finally, the Employee entity stores important data about healthcare staff, including their job title, name, age, hire date, and educational background. This helps keep track of employee details and their roles in the system. These design decisions were made to ensure that the database is well-organized and can effectively support the healthcare system’s needs, from managing patient care and appointments to handling medications and employee information.

# Conclusions

In conclusion, the design and early development of the Country GP project has made good progress toward meeting the goals set out in the initial proposal. We’ve established a solid structure using UML diagrams, which help visualize the system design and how users will interact with it. By applying Object Oriented Programming, we’ve improved the organization of the code and made the system easier to manage. The ERD has also been useful in showing how the different parts of the system are connected, making the database structure clearer.

Some of the initial ideas and assumptions from the draft proposal have been updated as we’ve worked through the design process. For example, we’ve added better data validation to the user interface to improve its reliability. The database schema has also been updated to include additional fields and relationships, ensuring we can properly manage both patient and employee data, as originally planned.

Looking ahead, we’ll need to keep refining the system as development continues. More testing will also be required to make sure everything works as expected before the project is complete.