



# System Development Project

Done by Group A6 2024/25

Amar Ali , Sultan Ali, Berke Sinar, Adeeb Imam, Rehan Sharif

# Aims And Objectives

## ➤ **Aim:**

To design, develop, and implement a cross-platform Feeding Dashboard for NHS Critical Care Units (CCU) that helps prioritize patient referrals to dietitians based on physiological data.

## ➤ **Objectives:**

- Real-time monitoring of CCU patients, with urgent referrals clearly flagged
- Support CSV upload and filtering functionality
- Enable detailed individual patient views and health metrics
- Provide report generation in PDF and Excel formats
- Ensure system compatibility across Windows, Mac, and Linux
- Adhere to security and performance standards required in healthcare environments

# Literature Review



Reviewed existing healthcare dashboards (e.g. NHS systems) to identify common features like data filtering, alerts, and real-time visualizations.



Investigated the role of **machine learning** in predicting patient health outcomes using physiological data.



Studied CSV integration methods and the importance of data validation in clinical software.



Explored tools like **Electron** for cross-platform app development.



Emphasized healthcare security standards such as **GDPR** and **HIPAA** for data protection.



# Project Planning And Team Roles

## Development Phases:

- Requirements Analysis
- System Design
- Implementation
- Testing
- Deployment

## Agile Development:

- Iterative sprints
- Frequent feedback
- Flexibility for changes



## Development Phases:

- Requirements Analysis
- System Design
- Implementation
- Testing
- Deployment

## Agile Development:

- Iterative sprints
- Frequent feedback
- Flexibility for changes

## Team Roles:

- **Implementation:** Adeeb
- **Coding:** Sultan/Amar/Rehan
- **Testing:** Sultan/ Adeeb
- **Report:** Adeeb/ Rehan/ Berke
- **Database:** Rehan/ Berke
- **Machine Learning:** Sultan/Amar
- **Logical Design:** Adeeb



## Requirnments

### **Functional Requirements:**

- Upload and process CSV patient data
- Display all and filtered lists of patients
- Visualize individual patient health data
- Flag patients needing dietitian intervention
- Generate reports in tables and graphs

### **Non-Functional Requirements:**

- Security
- UI responsiveness
- performance
- cross-platform support
- Scalability and maintainability for future enhancements



# Design And Test Plan

## System Design:

- **Use Case Diagram:** Shows all user interactions (e.g., login, upload, filter, generate report)
- **Class Diagram:** Represents entities like Patient, Report, User
- **Sequence Diagrams:** Show flow for login, CSV upload, filtering, and report generation
- **DFD (Level 1):** Maps data flow from CSV upload to report generation

## Test Plan Includes:

- **Unit Testing:** For backend functions
- **Integration Testing:** Between UI, backend, and ML model
- **System Testing:** Simulating end-to-end use
- **Performance Testing:** Stress tested with 5000+ records
- **Security Testing:** Checked access control and data protection
- **User Acceptance Testing (UAT):** To be conducted by tutors/mentors as proxy for NHS staff

# Implementation And Testing

## Tech Stack Overview:

- **Frontend:** HTML, CSS, JavaScript, Flask (Jinja2 templates)
- **Backend:** Flask (Python), Scikit-learn, Pandas
- **Database:** MySQL with SQLAlchemy ORM
- **Machine Learning:** Logistic regression model for flagging referrals
- **Visualization:** Chart.js for graphs
- **Packaging:** **Electron** used to wrap web app into installable desktop app

## Testing Summary:

- CSV upload and processing
- Data accuracy and filtering
- Report generation with graphs
- Large dataset handling
- Security/authentication testing underway
- Electron tested across OS platforms







# Evaluation

## **Achievements:**

- Fully functional dashboard meeting functional goals
- Flagging via ML logic is accurate and efficient
- User-friendly UI tested with positive feedback
- Electron deployment successful across Windows/macOS
- Reports generated dynamically from real data

## **Challenges:**

- CSV format inconsistencies required custom validators
- Implementing encryption and session control was complex
- Time constraints affected full user acceptance testing