Mobile Web Development (MAPD-722)

CENTENNIAL COLLEGE

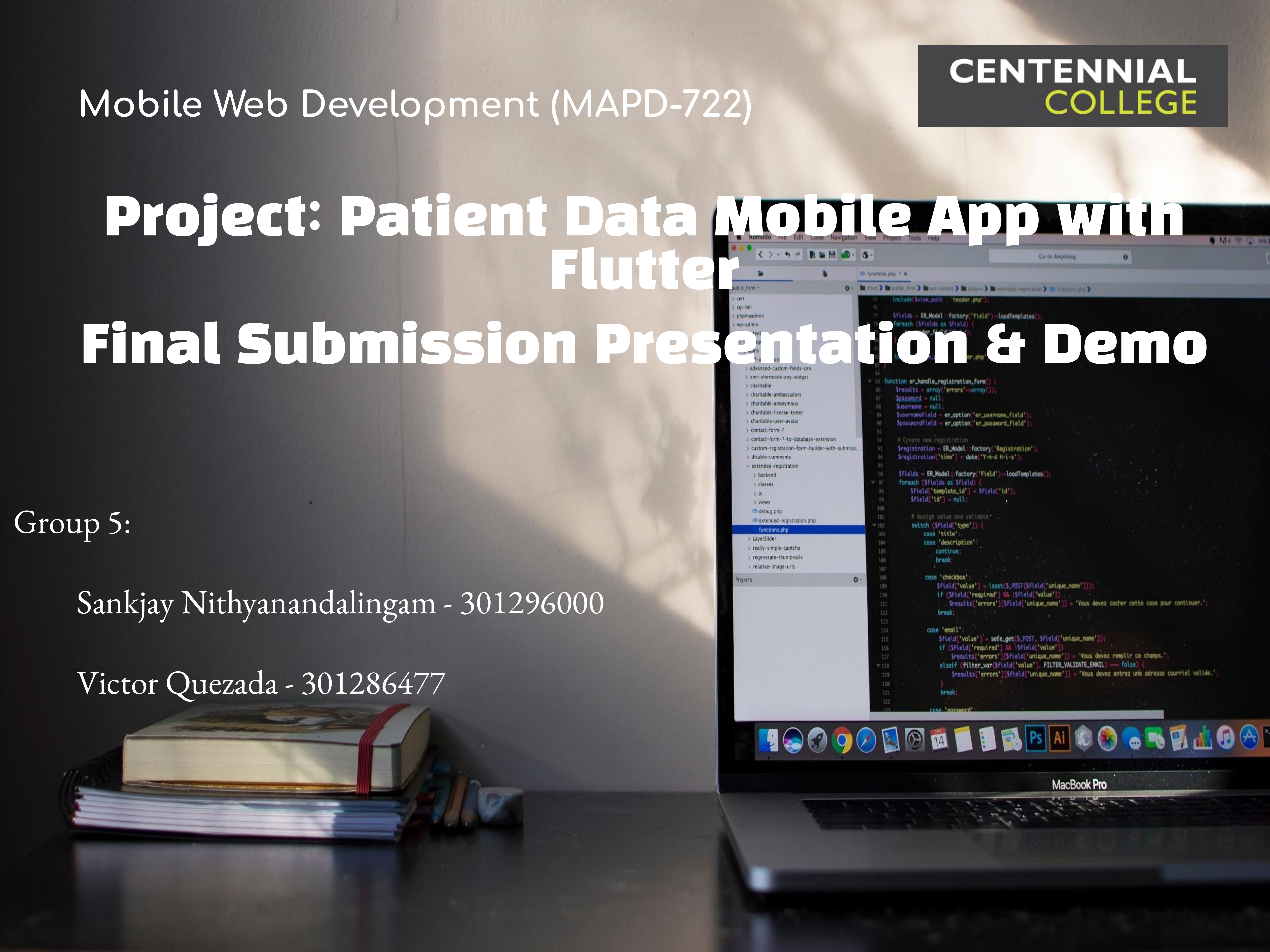
Project: Patient Data Mobile App with Flutter

Final Submission Presentation & Demo

Group 5:

Sankjay Nithyanandalingam - 301296000

Victor Quezada - 301286477



In this Presentation

- Project Introduction and Description
- Requirements
- Use Case Diagram
- Solution
- List of Functionalities
- Technical Stack
- Work Breakdown
- Challenges

Project Introduction and Description

- The project to be undertaken is to design and implement a Patient Clinical Data Management Application which will help the Health Care Providers to manage patient data.

- The product needs to be a Mobile Client built using Flutter.

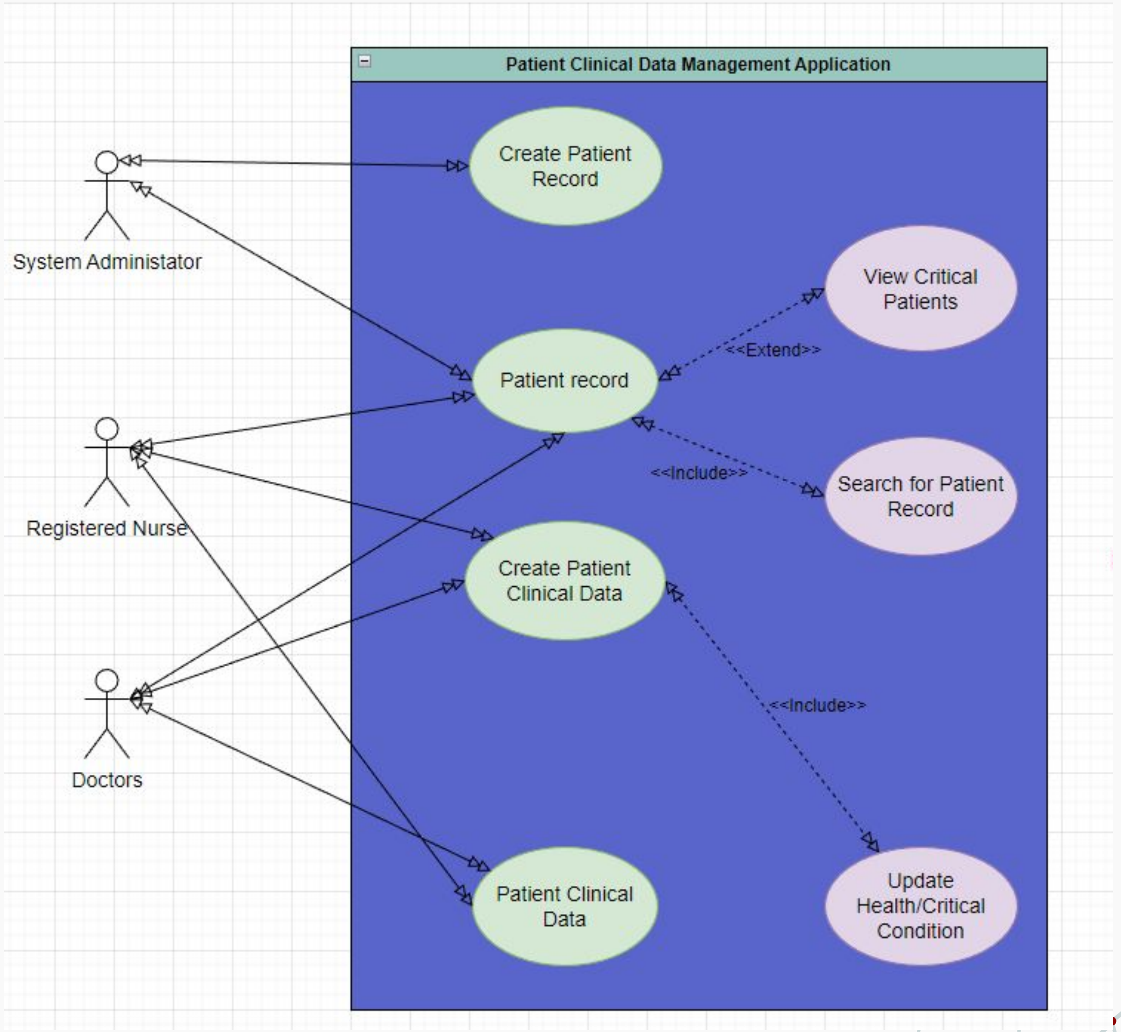
- Basically it is a Mobile App that will support Health Care Providers to manage their patients.

Requirements

Requirements have been provided to satisfy the relevant business needs. Therefore the requirements that need to be satisfied by the application are,

- The system will provide features to add and view patient details to track and monitor them.
- Users should be able to view a detailed list of an individual patient's clinical data.
- Users should be able to find critically ill patients using the system.
- Users should be able to add clinical data to patients' records and flag them if they are in critical condition.
- The data in the patients clinical file should include (Not limited to) the following,
  - Date/Time
  - Medical Data and its Measurement Reading/Value
  - The following readings should be added to the patients' clinical file (But not limited to)
    - Blood Pressure - X/Y mmHg
    - Respiratory Rate - X/min
    - Blood Oxygen Level - (X%)
    - Heartbeat Rate (X/min)

Use Case Model



Solution

- To implement the client we will be using Flutter and Node JS for API.

- Therefore the user interface elements can be individually designed and implemented while the backend services to Add and Fetch data have already been implemented with Rest APIs.

- The main advantage of using APIs for the backend service is that both the Frontend and the Backend of the application are allowed to evolve independently.

- To integrate Rest APIs, we will be using State Management Package
  - GetX - Used to fetch Patient lists, Critical Patients list, and Test history list

- Development will follow “Separation of concerns” as much as possible.

Functionalities

Feature	Data Input/Output
User Login and User Registration	• **Inputs:-** User name, Password for Login. User Details (Name, email, Un, Pw) for Registration • **Actions :-** Submit button
Add a New Patient	• **Inputs:-** Patient Name, Patient DOB, Address, Contact Number, E-Mail, Emergency Contact Person, Emergency Contact Number, Department, Assigned Doctor • **Actions :-** Submit Button
List All Patients	• **Outputs:-** List of all patients, Results of search input • **Actions:-** View Patient Records Button
Search Patient in Patient List and Critical Patient List	• **Inputs:-** Patient First Name • **Outputs:-** List matching patients in view
View Patient	• **Outputs:-** Lists all the patient’s personal details • **Actions:-** View Patient Clinical Data Button
Add Patient Clinical Record	• **Inputs:-** Date/Time, Blood Pressure - X/Y mmHg, Respiratory Rate - X/min, Blood Oxygen Level - (X%), Heartbeat Rate (X/min), Nurse Name, Auto Calculate Critical Status of Patient • **Actions:-** Submit Button
View Patient Clinical Record	• **Outputs:-** Lists all the Clinical data of the patient, Displays the critical condition of the patient
List All Critical Patients	• **Outputs:-** List of all critically ill patients, Results of search input • **Actions:-** View Patient Records Button
Search in Tests	• **Inputs:-** Date • **Outputs:-** List matching tests in view

Technical Stack

- **Flutter**
  - http
  - GetX
- **Node JS**
  - Express
  - Nodemon
- **MongoDB - Cloud**
  - Atlas

Work Breakdown

Name	Student ID	Contribution Modules
Sankjay Nithyanandalingam	301296000	1. View Designs - List Patients, Patient Details, Critical Patient List, Add Patient, Add Test, List Tests, View Test 2. API Integration - List all patients, List One Patient, Add Patient, Add test, List all tests, List one test, Update Critical Status 3. Critical Status Calculation Logic 4. Unit Tests 5. Presentation 6. Documentation and Diagrams
Victor Quezada	301286477	1. Login view design and API integration 2. Register view design and API integration 3. Presentation

Challenges

- **Learning and applying for coding**
  - New concepts were learnt to implement the project. The team had to assert long hours to apply the learnt modules for this project.
- **Splitting the work**
  - With both team members having no experience in the Development field until the start of the course, The team member who had more exposure to IT relevant experience was to take the lead and work on the most modules.
- **Integrating the API**
  - Merging to different technologies required a lot of research and time.
- **Coordinating with team**
  - Work hours were different, therefore it was a challenge to communicate.
- **Unit tests**
  - Writing the unit test scripts at the end were troublesome, since there were many errors and fixing them was time consuming.

Demo