Interim Report – Final Project (Module B8IT131)
Higher Diploma in Science in Computing (Software Development)
Software Development
Cian Walker
Student no: 10021419

Word Count: 1870

Acknowledgements

I would like to acknowledge my three technical mentors where I work at IBM:

-Jon Harry for giving me my foundational knowledge in Identity and Access technology.

-Robert Graham for giving me direction and steering in terms of what skills I should master (node.JS!) to flourish in my profession as a consumer-IAM engineer.

-Gonzalo Lorca for showing me how to chain systems together and always being the encyclopaedia I need

In terms of DBS staff, I'd like to acknowledge Claire Caulfield and Jennie Byrne, my first programming lecturers in C# and SQL DB respectively. They explained to a beginner so well, and gave such excellent practice exercises early on, that the programming bug took root in me right at the outset of the course, these two figures arguably more than any others contributed to the trajectory I went on.

Finally, Dr. Obinna Ozima has so far been a marvellous, positive, and encouraging project supervisor.

Table of Contents

Cian Walker

Introduction & Background

Project Aims

This project had two primary and aims. I wished to build a simple yet effective solution to a practical business or societal problem while simultaneously developing skills that would be of use in my career as an identity and access management engineer or architect.

First, the business problem and my proposed solution. The Republic of Ireland is yet to implement an ePrescribing software across is pharmacy and general practitioner ecosystem. In practice, this means that patients can either carry a hardcopy prescription to <u>any given pharmacy</u>, or a physician's surgery can email prescriptions to <u>one given pharmacy.</u> This is especially cumbersome for recurring prescriptions or for people who business travel regularly. The technology required should not be especially arduous to build a system that allows authenticated physicians to (to use HTTP parlance) '*post'* and '*put'* prescriptions; and for authenticated pharmacists to '*get'* prescriptions. It calls for, essentially, a series of CRUD (create, read, update, delete) APIs in addition to a registration and authentication flow to '*post', 'put', 'get', and 'delete'* identity records for authenticated physicians and pharmacists.

Regarding the second objective related to skill-building in an area relevant to my career path, it has been recommended to me by several of my mentors at IBM that I build some muscle memory and familiarise myself with node.js. This is in large part because I have a particular interest in consumer-facing identity and access management (CIAM) and because the node.js environment is becoming extremely popular with developers. It's been indicated to me that growing skills, particularly around OAuth/Open ID Connect flows for authentication and authorisation will make me an indispensable resource in the marketplace. For reasons I can expound on further below, it so happens that the node.js runtime environment is extremely suitable to this kind of web application. I have become somewhat besotted with JavaScript as a programming language. It's versatility and agility is extremely powerful, particularly now that it is increasingly being run from the server and not just on browsers as in the node.js runtime environment. It is also the primary language of identity & access management. JavaScript is a language I simply must obtain a grasp of. I would like to use this opportunity to also familiarise myself with non-relational database work since all of my database work to date has been with SQL-formatted structures. Finally, my area of particular interest career-direction-wise is in application integration, thus API development is <u>the</u> crucial skillset for me to work on.

Project Scope

The following functionalities are in scope as part of this project:

- A respectable user interface up to modern expected standards.
- A user authentication flow for two kinds of users: Physicians and Pharmacists.
- A JavaScript API layer, run from the server-side using Express, as per the Node.JS runtime paradigm.
- A data structure (Obinna; I'm thinking non-relational JSONs living on a MongoDB – we can chat through on our call).

Project Approach

The solution will be a web application back-ended by a server-side layer of JavaScript REST APIs and a (Obinna, most likely) non-relational database of JSONs residing on either npm Registry or MongoDB.

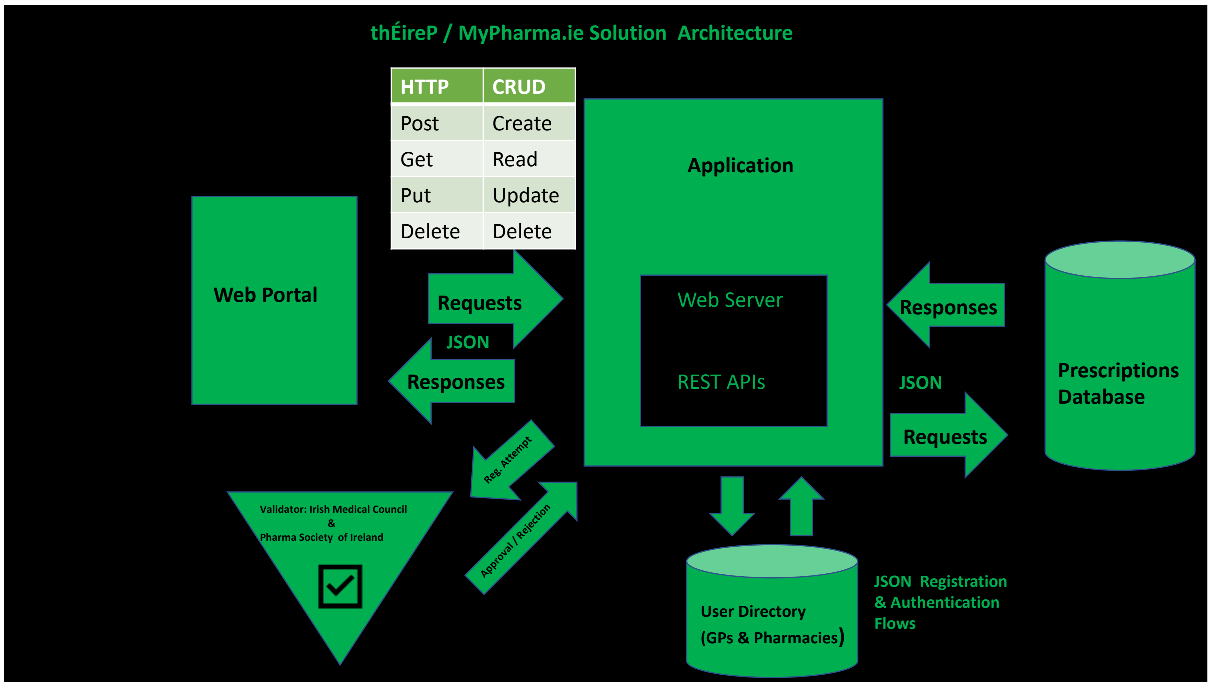
Specification & Design

This solution requires the ability to handle many, many HTTP requests to 'POST', 'GET', 'PUT', and 'DELETE' relatively simple medical prescription records in and out of a database.

It does not call for heavy-duty data processing, machine learning, algorithms, or complex calculations on large sets. (In a production solution, firms may want to gain more granular insights or monetise their data, but for the purposes of this project, the core functionalities are the focus).

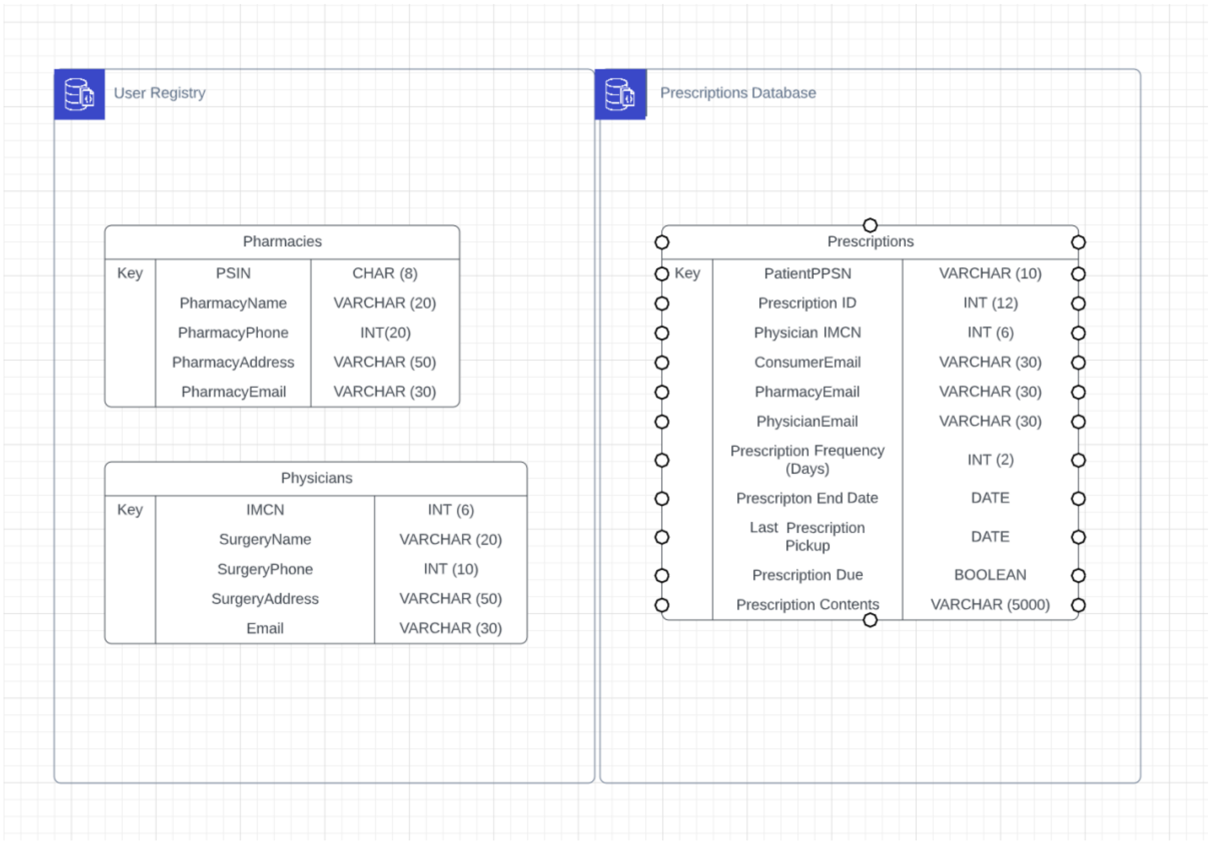
These are the considerations that have informed my decision to implement the solution in Node.JS as its asynchronous, non-blocking architecture is optimised for requirements that involve handling high volumes of input/output, call for high-scalability, and only require single-threaded processes.

Other relevant advantages of non-relational databases is their flexibility and low cost of revision in the early stages of a product's lifecycle. This allows easy iteration and refinement to the data structure as use cases dictate.


thÉireP / MyPharma.ie Solution Architecture
Application
Web Portal
Requests
JSON
Responses
Web Server
REST APIs
Responses
JSON
Requests
Prescriptions Database
Reg. Attempt
Approval / Rejection
Validator: Irish Medical Council & Pharma Society of Ireland
User Directory (GPs & Pharmacies)
JSON Registration & Authentication Flows

HTTP	CRUD
Post	Create
Get	Read
Put	Update
Delete	Delete

Data Structures



Pharmacies

Key		
Key	PSIN	CHAR (8)
	PharmacyName	VARCHAR (20)
	PharmacyPhone	INT(20)
	PharmacyAddress	VARCHAR (50)
	PharmacyEmail	VARCHAR (30)

Physicians

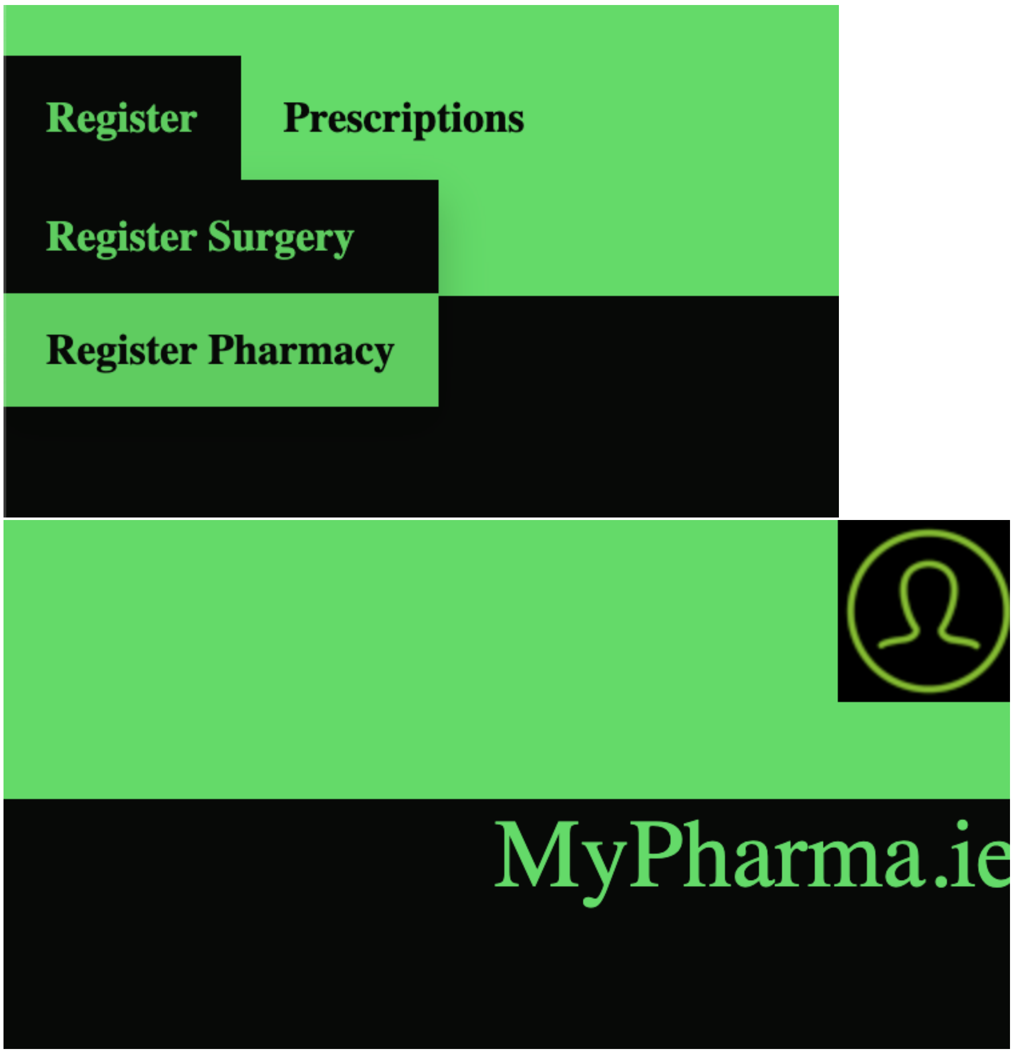
Key	IMCN	INT (6)
	SurgeryName	VARCHAR (20)
	SurgeryPhone	INT (10)
	SurgeryAddress	VARCHAR (50)
	Email	VARCHAR (30)

Prescriptions

Key	PatientPPSN	VARCHAR (10)
	Prescription ID	INT (12)
	Physician IMCN	INT (6)
	ConsumerEmail	VARCHAR (30)
	PharmacyEmail	VARCHAR (30)
	PhysicianEmail	VARCHAR (30)
	Prescription Frequency (Days)	INT (2)
	Prescripton End Date	DATE
	Last Prescription Pickup	DATE
	Prescription Due	BOOLEAN
	Prescription Contents	VARCHAR (5000)

Project Testing, Evaluation, and Demonstration of Progress

To date, no weekends have been sacrificed to the project, thus only user interface work has been carried out. Testing is carried out continuously as the UI is coded, rendered, and refined on a loop. Much improved dropdowns and hover colour reversals have been executed successfully, however. The user icon/dropdown login credentials form is the UI element I'm currently trying to implement.



Luckily, I have eight days of paid study leave in hand with four booked in this month in addition to the next two weekends being fully dedicated to the project. While the project is somewhat behind against schedule, I am confident that the sprint over the coming weeks should see the user interface all but complete by the end of November, with only refinements remaining of this layer. This will leave the entire month of December for API development. I do not foresee the development of the data layer being time intensive.

Future Work & Dependencies

I have divided this section into application layers plus an extra section for the user authentication flow.

User Interface: The next element to be developed is the dropdown login credentials form to render when the user icon is clicked. Next, I will develop a form page prototype template that renders the input data for the user to review before confirming/submitting. When this is developed satisfactorily, I can version it out thrice for the two kinds of user registration, prescription submission, and prescription amendment use cases. Compared to my last project, I need to add significantly more validation and testing of use cases where forms are filled incorrectly etc. I have realised after some recent mentoring that these cases are still use cases in themselves.

User Authentication: This is both the part of the workflow that most intimidates me but also I think to be of most value to me from a learning standpoint. OpenIDConnect is a universal standard for user authentication consisting of an extended OAuth 2.0 flow. In future, the application will identity federation via this mechanism if the authentication flow is built to this standard. This will be especially valuable for social sign-on if the solution is extended to patient/consumer users but even before then, will be of value for federation with pharmacist or GP enterprise apps. Identity Federation removes a lot of friction from the user registration process and so will drive adoption.

CRUD APIs Layer: On a previous build of this application, the CRUD use cases consisted of:

- Physician Create Prescription
- Physician Update Prescription
- Pharmacist Read Prescription
- Physician User Registration
- Pharmacy User Registration

One additional CRUD operation I have identified as a crucial addition is 'Pharmacy Update Last Pickup Date'. This is because I wish to retain a patient's PPS Number as the primary key of a prescription due to its ease of remembrance to a consumer. The initial drawback of this approach was that, in cases of multiple prescriptions per patient, the risk was run that a patient may pick up additional iterations of a prescription before it was due. With the addition of the 'Last Pickup Date' attribute to the prescriptions data structure, a function can be written that will return a Boolean as to whether the prescription is due again at any given time or not. This means that a patient can simply walk into any pharmacy, quote their PPS number and a pharmacist can retrieve any and all prescriptions against that PPS number. A decision remains to be made on whether to display all prescriptions including those which are not due or whether to only display prescriptions due. Perhaps displaying just a heading on undue prescriptions without the contents or frequency will work best since it allows the pharmacist to inform the patient what the prescription is without seeing details aside from when it is next due. Either way, this will require a pharmacist to either always write a value into the last pickup date attribute or, better yet, to click a button that will trigger a timestamp function to populate this field.

Data Layer: Given that the solution is in an early stage of development, the flexibility a non-relational database confers is of value. The coming fortnight will see me familiarise myself with npm registry and MongoDB as data structures for storing JSON documents in for users and prescriptions. From my initial research, the HTTP / CRUD operations seem slightly less arduous to implement in JSON/MongoDB than they are in SQL.

Appendices

Functionality Development Hierarchy

Here is a list of functionalities to be developed in order of precedence, with functionalities further down to be developed 'if time allows':

Layer	**Functionality**
User Interface	Dropdown Login Credentials Form
User Interface	Prototype Form with Confirm Button JavaScript
User Interface	Versioning of forms for user registrations, prescription submission and prescription amendments.
User Authentication	Registration and Validation (against Irish Medical Council & Pharmaceutical Society of Ireland) Flows
User Authentication	Ongoing Authentication to Open ID Connect Standards
CRUD APIs	Physician Create, Update, Delete Subscriptions

CRUD APIs	Pharmacist Read Prescription, Update Last Pickup Attribute with timestamp.
Database	Implement User Registry with separate tables for Pharmacists and Physicians
Database	Implement Prescriptions Database
Database	Format User Registry to LDAP Standards?

Bibliography

How to design schema for NoSQL Data models (no date) *MongoDB*. Available at: https://www.mongodb.com/nosql-explained/data-modeling (Accessed: November 8, 2022).

Kakar, S. (2021) *Building a restful crud API with node JS, Express, and mongodb, DEV Community 👩‍💻👨‍💻*. DEV Community 👩‍💻👨‍💻. Available at: https://dev.to/suhailkakar/building-a-restful-crud-api-with-node-js-express-and-mongodb-1541 (Accessed: November 8, 2022).

MongoDB CRUD Operations (no date) *MongoDB CRUD Operations - MongoDB Manual*. Available at: https://www.mongodb.com/docs/manual/crud/ (Accessed: November 8, 2022).

Relational vs. non-relational databases (no date) *MongoDB*. Available at: https://www.mongodb.com/compare/relational-vs-non-relational-databases#:~:text=Relational%20databases%20are%20also%20the,database%20is%20still%20the%20answer. (Accessed: November 8, 2022).

Salapu, R. (2022) *CRUD operations using JSON data in SAPUI5, SAP Blogs*. SAP Community Blogs. Available at: https://blogs.sap.com/2022/03/10/curd-operations-using-json-data-in-sapui5/ (Accessed: November 8, 2022).

Start using your openid (2012) *OpenID*. Available at: https://openid.net/start-using-your-openid/ (Accessed: November 8, 2022).

Why and when to use node.js in 2021 [Complete guide] (2021) *Relevant Software*. Available at: https://relevant.software/blog/why-and-when-to-use-node-js/#Are_there_any_disadvantages_of_Nodejs (Accessed: November 8, 2022).