University of Essex

School of Computer Science and Electrical Engineering

CAPSTONE PROJECT DISSERTATION

GoLearn — Learning Resource Management System

Bradley Beasley

Supervisor: **Dr Alexandoros Voudoris**

April 17, 2024
Colchester

Abstract

There are plenty of services and programs out there for allowing Teachers to upload resources and for their learners to access them. Most of these are not purpose designed for places of learning, such as school, colleges and universities. Those that exist for this specific use-case tend to be over-engineered and can be overly complicated (Moodle is a good example of this). The purpose of GoLearn is to provide a simple LRM experience for teachers and students to interact with the same resources without a large amount of overhead in setting up the system: it should work out of the box.
This report contains a look into methods of implementation for web applications in general, plus the choices in technology and design that were made for GoLearn.

Contents

CHAPTER 1

List of Symbols

API Application Programming Interface

CRUD Create Read Update Delete

DB Database

JS JavaScript

LAMP Linux, Apache, MySQL, PHP

LEMP Linux, (E)Nginx, MySQL, PHP

LRM Learning Resource Management

MEAN Mongo, Express, Angular, Node

MEVN Mongo, Express, Vue, Node

MERN Mongo, Express, React, Node

NPM Node Package Manager

ORM Object Relational Mapper

OS Operating System

PHP Hypertext Preprocessor

PWA Progressive Web Application

QA Quality Assurance

SEO Search Engine Optimisation

SPA Single Page Application

SQL Structured Query Language

SSR Server Side Rendering

TS TypeScript

UAT User Acceptance Testing

WP WordPress

XHR XMLHttpRequest

CHAPTER 2

Introduction

CHAPTER 3

Background

3.1 Similar Services

There are various LRMS's available in the market, with some of the more popular one being:

Udemy is a free to access learning site that puts a paywall over all of the courses. Creators can sign up to the site and create a course very quickly. Learners can sign up to courses (paying whatever charge is applied) and have unlimited access to them. They can access these from their own 'Courses' dashboard.

Skillshare is a similar service to Udemy, but is subscription based rather than pay per course. This means that learners can access all of the courses on the site for a monthly fee. Creators can create courses and upload them to the site, with the same access restrictions as Udemy.

LearnDash[1] is a WordPress plugin that allows you to create and sell courses on your WordPress site. It is a very popular plugin, with over 50,000 active installs and a 4.5-star rating on the WordPress plugin repository. It is a very powerful plugin, with a lot of features, but it is also very complex and can be difficult to use.

Being a plugin for WP, it is very easy to customise how the plugin looks by modifying the theme on the site to best suit the business / individual's needs. The plugin allows teachers to create courses and students to access them, with a lot of customisation options for the courses.

Moodle[2] is the closest comparison that is currently available to GoLearn. It is a PHP based LMS that is very powerful and feature-rich. It allows learning institutions to design their own courses/modules and add specific 'student' users to them. This is a very similar model to what GoLearn is aiming to achieve. The main difference is that Moodle is very complex and can be difficult to use. It is also very resource-intensive, requiring a lot of server resources to run effectively.

3.2 Tech Stacks

A 'Tech Stack' is the term used to describe the combination of programming languages, frameworks, libraries, and tools that are used to build a software application. The choice of a tech stack is crucial

as it can affect the performance, scalability, and maintainability of the application. The following are some of the popular tech stacks used in web development:

LAMP/LEMP Stacks This stack consists of 4 major components: Linux, Apache/Nginx, MySQL and PHP. Linux is the OS that is used in the stack, with Apache or (E)nginx as the web server, MySQL as the database and PHP as the server-side language. This stack is very popular and is used by many web developers. It is used by various Frameworks (such as Laravel), as well as in vanilla PHP applications (no framework used).

PHP is the programming language of choice for this kind of stack for websites and applications that need to render the pages dynamically, but don't need a lot of reactivity to user input on the page. For for example, a blog or a news site would be a good use case for this stack, as they largely need to retrieve data from a db when a page is loaded, and can use caching strategies easily to only make a DB request for cached pages after a certain period after the cached content was stored.

On the other hand, applications that have a lot of user interaction to perform CRUD operations would be harder to write within this stack, as JS would have to be loaded and ran to submit the changed data. Other stacks (as mentioned below) instead rely less on page loads to perform actions, and instead are designed with responsive and reactive implementations in mind.

MEAN/MERN/MEVN Stacks The 'M' refers to the database implementation used (such as mySql, MongoDB)..., the 'E' refers to ExpressJS, which is a NodeJS framework used for all forms of JS applications, and is the layer between the templating framework and NodeJS to serve an application. The 'V', 'A' and 'R' refer to the templating framework used to create the layouts used, and reference 'VueJS', 'AngularJS' and 'ReactJS' respectively. The 'N' refers to 'NodeJS', which is a server-side JS runtime environment that is used to run JS code on the server.

This stack is very popular for creating SPAs and PWAs. These are applications that are designed to be very reactive to user input, and can update the page without needing to reload the page. This is done by using JS to update the page, rather than relying on the server to render the page and send it to the client. This is achieved by requesting information from the server about what to load and how to load it, and then using JS to render the page based on the information received. This is a very powerful stack, but can be difficult to learn and use effectively.

Full Stack Frameworks A full stack framework is a framework that is designed to be used for both the front-end and back-end of an application. This means that the framework can be used to create the server-side code, as well as the client-side code. This is very useful as it means that development teams can rely on a single codebase for the entire application, rather than having to use multiple codebases. This can make development faster and easier. As well as this, it allows type safety and code completion to be used across the entire application easier than in separate front- and back-end applications.

Some examples of full stack frameworks are NextJS and NuxtJS. These frameworks are designed to be used for both the front-end and back-end of an application. They are very powerful and can be used to create very complex applications. They each use a different templating framework (NextJS uses ReactJS, and NuxtJS uses VueJS). The benefit of these frameworks is that they allow

for server-side rendering as well as client-side rendering, which can be very useful for SEO and performance reasons.

One drawback of these frameworks is that there can be a steep learning curve to get started with them. As well as this, the codebase can get very complex and the boundaries between client and server code can get blurred, which can make it difficult to debug and maintain the codebase if it is not well organised.

3.3 Databases and ORMs

A database is a collection of data that is stored in some structured format. Databases are used to store data that can be accessed and manipulated by software applications. There are many different types of databases, each with its own strengths and weaknesses, but all fall into one of two categories: relational or non-relational.

Relational Databases[3] store data in tables, with each table containing rows and columns. The columns represent the attributes of the data, while the rows represent individual records. Relational databases use SQL to query and manipulate the data. Some popular relational databases include MySQL, PostgreSQL and SQLite.

This category of database is called relational because it allows for relationships to be defined between the columns in different tables. For example, a database for a school might have a 'students' table and a 'courses' table. The 'students' table might have a column that references the 'courses' table, which would allow for a relationship to be defined between the two tables. This is called a foreign key, and can be used to enforce that a student can only be enrolled in a course that exists in the 'courses' table.

This is a suitable database design for many kinds of applications, such as those that require complex relationships between the data, or that require consistency on the data. A drawback that this can lead to is that the data can be inflexible, as the schema has to be defined before the data can be stored, which can make it problematic to change the data model once the data has been stored.

Non-Relational Databases[4] store data in a more flexible format, such as key-value pairs, documents, or graphs. Non-relational databases are often used when the data is unstructured or when the data model is likely to change frequently. Some popular non-relational databases include MongoDB, Cassandra, and Redis. This kind of database is often referred to as a NoSQL database.

This category of database stores data in an unstructured format, which allows for it to be more flexible and horizontally scalable. This is useful, as it means that the data can be stored in such a way that the load can be distributed across multiple servers, which can help to improve performance and scalability. A drawback that this can lead to is that the data can be inconsistent, as there may be duplicated data across multiple servers, which can lead to issues with data integrity.

This is a suitable database design for many kinds of applications, such as those that require high performance and scalability, or that require the ability to store large amounts of data. As well

as this, applications that require a lot of flexibility in the data model can benefit from using a NoSQL database, as it allows for the change in the data model over time, without having to make adjustments to the existing data.

An ORM is a layer between the application and the database that allows the application to interact with the database using an object-oriented interface. These are used to simplify the process of interacting with the database, and to make it easier to work with the data in the database. They supply an API for developers to be able to perform CRUD operations on the database without having to construct many (if any) queries or statements to do so.

Some examples of ORMs include Prisma, TypeORM, and Sequelize. These ORMs are designed to be used with different databases, and have different features and capabilities. For example, Prisma is designed to be used with PostgreSQL, MySQL, and SQLite, and has a strong focus on type safety and code generation. TypeORM is designed to be used with TypeScript, and has a strong focus on type safety and code generation. Sequelize is designed to be used with MySQL, PostgreSQL, SQLite, and MSSQL, and has a strong focus on performance and scalability.[5]

3.4 Testing methods

Unit testing[6] is a type of testing that is used to test individual units or components of an application. It is used to ensure that the individual units of an application are working as expected, and that the various functions behave as expected in isolation from each other. Unit testing is a very important part of the development process, as it can help to catch bugs and issues early in the development process, before they become more difficult and expensive to fix. Unit testing can also help to ensure that the application is working as expected, and that there are no bugs or issues that could cause the application to fail.

User acceptance testing[7] is a type of testing that is used to ensure that the application is working as expected from the user's perspective. It is used to ensure that the application is easy to use, and that the user interface is intuitive and user-friendly. UAT is a very important part of the development process, as it can help to ensure that the application is meeting the needs of the users, and that the users are able to use the application effectively. UAT can also help to identify any issues or bugs that may have been missed during the development process, and can help to ensure that the application is working as expected.

3.5 Authentication

Authentication can be defined as: 'Verifying the identity of a user, process or device, often as a prerequisite to allowing access to resources in an information system.'[8]. Simply put, it refers to any processes used to ensure that only those with the right to access a resource or perform functions can do so. This is a very important part of any application, especially those that can be accessed from a remote location (such as via the internet). This is because protecting the data that is managed by the application is a legal requirement (see Chapter 7).

There are several methods of authentication that can be implemented. A few are listed and described below:

Basic Authentication[9] This is the simplest form of authentication, and is often used in conjunction with a username and password. The user is prompted to enter their username and password, which is then sent to the server. The server then checks the username and password against a list of valid credentials, and if they match, the user is granted access to the resource. This is a very simple form of authentication, but it is not very secure, as the username and password are sent in plain text over the network. This means that anyone who is able to intercept the request can read the username and password and use them to access the resource. For this reason, it is not recommended to use basic authentication for anything other than testing purposes.

Token-Based Authentication[10] This is a more secure form of authentication that is often used in conjunction with a username and password. The user is prompted to enter their username and password, which is then sent to the server. The server then checks the username and password against a list of valid credentials, and if they match, the server generates a token and sends it back to the client. The client then includes the token in all subsequent requests to the server, and the server checks the token against a list of valid tokens to ensure that the user is authenticated. This is a more secure form of authentication, as the token is not sent in plain text over the network. This means that even if someone is able to intercept the request, they will not be able to read the token and use it to access the resource.

OAuth[11] is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords. This is done by authorizing the application to access the data on the user's behalf, and then the application is given a token that can be used to access the data. This is a very secure form of authentication, as the user's password is never sent over the network. This means that even if someone is able to intercept the request, they will not be able to read the password and use it to access the resource.

CHAPTER 4

Technical Specification

4.1 Framework

Of the stacks mentioned in Section 3.2, the NextJS framework was chosen for the GoLearn project. This was chosen for the following reasons:

ReactJS[12] ReactJS is a very powerful and popular front-end framework that is used to create SPAs and PWAs. It is very flexible and can be used to create very complex applications. It is also very easy to learn and use, which makes it a good choice for the GoLearn project.

As well as this, it was a framework that the author had used before, meaning that he could get started with the project quickly and easily and wouldn't have to spend a lot of time learning to use a new framework or architect his own solution from scratch. This proved to save a lot of time in the development process, as the author was able to get started on the project and make progress quickly.

Server Actions[13] As of NextJS 14, the framework included a feature called 'Server Actions'. This allowed functions to be shared between server- and client-side code. Whenever a server action was called, a request is sent to the server as a Fetch/XHR request from the client, the server would handle the request and return the result to the client. This would prove to be a very useful feature, as it allows for CRUD operations to be ran.

4.2 Database and ORM

The ORM chosen for the GoLearn project was Prisma. This was chosen for the following reasons:

Type Safety[14] Prisma is designed to be used with TypeScript, and has a strong focus on type safety and code generation. This means that the author could be confident that the code he was writing was correct and that the database schema was being enforced correctly. This was very important for the GoLearn project, as it was a project that was being developed by a single developer, and there was no QA team to review the code.

Code Generation[15] Prisma uses code generation to generate the database schema and the client-side code that is used to interact with the database. This means that the author could be confident

that the code he was writing was correct and that the database schema was being enforced correctly. This was very important for the GoLearn project, as it was a project that was being developed by a single developer, and there was no QA team to review the code.

Prisma can be used with many different databases, including PostgreSQL, MySQL, and SQLite. One of the factors that was considered was the ease of creating and maintaining the database. Originally the author had considered using a service called 'Supabase', which is a service that provides a PostgreSQL database, and is a service that Prisma has a lot of support for.

Prisma is used by defining a schema in a file called 'schema.prisma', which defines the data models that are used in the application, as well as the relationships between the models. Prisma then uses this schema to generate the database schema and the client-side code that is used to interact with the database. Whenever changes are made to the schema, Prisma can be used to update the database schema and the client-side code, which makes it very easy to maintain the database and the codebase.

Later in the project when the application hosting was being considered, it was decided that the author would the application on Vercel, which is a service that provides hosting for NextJS applications. Vercel has a feature called 'Vercel for Databases', which allows you to create a PostgreSQL database that is hosted on Vercel. It was a relatively simple process to migrate the database from Supabase to Vercel, and the author was able to do this without too much difficulty. The Vercel database is also a PostgreSQL database, which meant that the author could continue to use Prisma with the database without any issues.

CHAPTER 5

Implementation

5.1 Data model

There was a lot of thought put into the data model for the GoLearn project. The data model was designed to be as simple as possible, while still being able to store all of the information that was needed for the application to function correctly. The following models were implemented, with various relations between them:

User The User model was used to store data about a User, such as username, password, role, etc. The role was used to determine what permissions the user had, such as being able to edit content or not.

Student The Student model was used to store data about Students, such as their home and term addresses, enrolled course and modules. These fields were used to determine which modules and course pages they could access. If a student was not enrolled in a course, they would not be able to access the course page, the same being for the modules.

Teacher The Teacher model was used to store data about Teachers, such as their address and the modules they were teaching. These fields were used to determine which modules pages they could access. If a teacher was not teaching a module, then they can not add content to those modules, nor make edits to them.

Course A course has a name, description and a list of modules that are part of the course. The course is the top-level resource in the GoLearn project, with all other resources being children of the course, or children of children, etc. The students that are enrolled in the course are also associated to a course.

Module A module has a name, description, assignments, units, students and teachers. This is an important model, as it acts as a container for students and teachers to interact with the content. Teachers can create and modify the content in a module and it's children, and students can view the content if they are enrolled in the module.

Assignment Assignments are a type of resource that can be created by teachers within a module, which students can then create submissions for. Assignments have a name, description, due

date, and a list of submissions that have been made for the assignment.

Unit A unit is a resource that teachers can create within a module. This is a method for organising the content within a module into smaller, more manageable chunks. Units have a name, description, and a list of resources that are part of the unit. There can also be sections within a unit, which are used to further organise the content within the unit.

Section A section is a resource that acts as a container for Resources within a Unit. Sections have a name, description, and a list of resources that are part of the section. The reason for this inclusion into the model was to allow teachers to better organise content within a unit such that they can group together resources that are related to each other without having to use naming convention to organise it for them.

Resource A resource is a leaf resource that can be created by teachers within a section and/or. This is the lowest level of resource that can be created, and is used to store the content that is displayed to the students. Resources have a name, description, and a content field that is used to store the content that is displayed to the students. As well as this, they can have any number of files attached to them, which can be used to provide additional resources to the students that is more complex than can be expressed by markdown.

5.2 Authentication

The method of authentication that was chosen for the GoLearn project was token-based authentication. This was chosen as it is a relatively simple form of authentication that is secure and easy to implement. Within the database, a 'UserSession' table was created to store the tokens that are generated for each user. When a user logs in, a token is generated and stored in the 'UserSession' table, and the token is sent back to the client. Whenever a user makes a request from the server, it checks if a token has been included in the request, and checks that token against the list of valid tokens in the 'UserSession' table. If the token exists and is not too old to be used, the user's request is granted, and the user is able to access the resource.

This was effective, but has a few drawbacks. The main drawback is that the token is stored in the database, which means that the database has to be queried every time a user makes a request. This can be slow and can cause performance issues if the database is not optimised. A better implementation of this might have been to use a library that was written to interact on the server fully and store the token in memory, rather than in the database. This would have been faster as querying memory is significantly faster than querying a remote database, as well as allowing the database to not get cluttered with old tokens that have either expired or are no longer in use.

5.3 CRUD Operations

All of the CRUD operations that had to be performed on the data would usually be in response to some event on the front-end of the application on some user interaction. To perform these, it was decided to use Server Actions. These are functions that are defined and designed to be ran on the server, but can be called from anywhere within Next application. The benefit to this is that it meant tat

making direct requests to an endpoint on the server via AJAX or the JS Fetch API is required, and the return types of the functions is known at compile time.

For each of the data models (Course, Module, etc) has their own file in the '/actions' directory, which contains the functions that are used to perform the CRUD operations. The reason for this codebase design was to keep the server actions bundled together, and have each method in a file related to the data that it was retrieving. For the Read operations, the functions could accept a list of strings that could be used to fetch relationship data from the database. This was to allow for the functions to be more diverse.

Each method can also accept a list of the roles that are able to access the data. This was a security feature that was included to ensure that only the correct kind of user can access the data that is being requested. This inclusion ended up not being used however, as role checks we performed on page load itself, so only users that could access a page would trigger the request for the data that would be required to render the page.

5.4 Data Loading

Each of the pages requires data to be retrieved from the database. The method that was used in this project was loading the data on the client side. When a page is mounted to the client (i.e. when the page is loaded), and once the user has been authenticated, the client would make a request to the server to get the data that is required to render the page. This meant that there would be some disconnect between the browser receiving the web page and the page fully rendering. During development, this was not an issue and was overlooked. In the latter stages of the project, it was realised that this was a problem and that the page should be rendered on the server and sent to the client, rather than the other way round.

This is a problem that has not been fixed yet, but is something that would be addressed if there was more time to work on the project. Attempts were made, but many of the implementation methods used for User Authentication and Server Actions were not compatible with the methods used to perform the server-side rendering. This would require a large amount of refactoring to fix, and would be a large task to undertake.

5.5 Known Issues

As mentioned before in section 5.4, there is an issue with the way that the pages are rendered. This is related to the the way that the user authentication and data retrieval is currently implemented. At the time of writing this report, the method of loading data onto a page is performed on the client once the page has been sent. When some pages are being loaded, the code to load the data (a 'useEffect' hook) is not ran, which is causing the data to not be loaded, and for the 'loading' state to be stuck on 'true'. To fix this, the pages themselves would need to make use of SSR to validate the auth token and load the data on the server before the page is sent. Then, the data that various components require to run can be passed to specific components as props which can then hold onto that data as state that can then manage the values and make calls back to the server to perform required CRUD operations.

CHAPTER 6

Testing

Unit testing and UAT were both intended to be used for this project. The reason for this was to ensure that the CRUD methods were returning the expected values under different conditions, and that the application was working as expected from a user's perspective.

6.1 Unit Testing

To implement unit testing, the Jest testing framework was used. Jest is a very powerful testing framework that is designed to be used with JS/TS applications. The documentation was referenced to learn how to write tests[16], as well as a number of tutorials that were found online[17][18]. The tests were written to test the Server Actions. The reason that these alone were tested was because they were the only functions that were written that could be tested. The other code that was written was either React components, which have a myriad of problems associated with Unit Testing.

As these server actions were functional wrappers for Prisma queries, the tests needed to also mock out the responses expected from Prisma. This was done by using the 'jest.mock' function to mock out the Prisma functions that were being used in the server actions. This proved difficult to do, largely because there were no video tutorials for doing this, and the documentation was not very clear on how to do it. This page in the documentation was used to implement the mocking of the Prisma functions: `https://www.prisma.io/docs/orm/prisma-client/testing/unit-testing`. This involved setting up a singleton mocking object that could be used to return values that were expected from the Prisma functions.

6.2 UAT

User Acceptance Testing was intended to be performed by the author, as well as by a number of friends and family members. The author would be responsible for testing the application in a number of different scenarios, such as logging in as a student and a teacher, creating and editing content, and enrolling in courses and modules. The friends and family members would also make the same checks, and be watching out for anything that seemed out of place, as well as providing feedback on the design and usability of the application.

This was not performed, as the application was not in a state where it could be tested by anyone

other than the author. The application was deployed to a Vercel server, but the application itself was not working as intended (as discussed in section 5.5). This meant that the application was not readily testable for end users. Despite this, during the development of the application, the author was able to get some face-to-face feedback from friends and family members, which was used to make changes to the design and usability of the application.

CHAPTER 7

Legal and Ethical

CHAPTER 8

Project Planning

CHAPTER 9

Conclusions

Bibliography

[1] J. Ferrimanm, "Learndash 3.0." https://www.learndash.com/best-wordpress-lms-plugin/, 5 2019.

[2] Moodle, "Moodle." https://moodle.org/, 1 2024.

[3] Wikipedia and contributors, "Relational database." https://en.wikipedia.org/wiki/Relational_database, 4 2024.

[4] Wikipedia and contributors, "Nosql." https://en.wikipedia.org/wiki/NoSQL, 3 2024.

[5] M. Wanyoike, "9 best javascript and typescript orms for 2024." https://www.sitepoint.com/javascript-typescript-orms/, 3 2023.

[6] Wikipedia and contributors, "Unit testing." https://en.wikipedia.org/wiki/Unit_testing, 4 2024.

[7] Wikipedia and contributors, "User acceptance testing." https://en.wikipedia.org/wiki/Usability_testing, 4 2024.

[8] M. Niels, K. Dempsey, and V. Yan Pilliteri, "An introduction to information security." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-12r1.pdf, 4 2024. Section 9.1.4.

[9] Wikipedia and contributors, "Basic access authentication." https://en.wikipedia.org/wiki/Basic_access_authentication, 4 2024.

[10] Okta, "What is token-based authentication?." https://www.okta.com/uk/identity-101/what-is-token-based-authentication/, 2 2023.

[11] Okta, "What is token-based authentication?." https://www.okta.com/uk/identity-101/what-is-token-based-authentication/, 2 2023. Section 4: JSON Web Token (JWT): A Special Form of Auth Token.

[12] D. Abramov and R. Nabors, "Introducing react.dev." https://react.dev/blog/2023/03/16/introducing-react-dev, 4 2024.

[13] Next.js, "Server actions and mutations." https://nextjs.org/docs/app/building-your-application/data-fetching/server-actions-and-mutations, 4 2024.

[14] Prisma, "Type safety." https://www.prisma.io/docs/orm/prisma-client/type-safety, 4 2024.

[15] Prisma, "Generating prisma client." https://www.prisma.io/docs/orm/prisma-client/setup-and-configuration/generating-prisma-client, 4 2024.

[16] yepitschunked and Jest, “Getting started.” https://jestjs.io/docs/getting-started, 1 2024.

[17] D. Grey, “Next.js with react testing library, jest, typescript.” https://www.youtube.com/watch?v=AS79oJ3Fcf0, 8 2023.

[18] M. Maksi, “Practical beginner guide to testing - next.js | react | jest | react testing library (rtl).” https://www.youtube.com/watch?v=pnLC-9waA44, 4 2024.