Euan Mendoza Year 12 Major Project Logbook

# 15/02/2019

* Tested different web frameworks for the potential use
* Server Side frameworks tested:
  + <https://rocket.rs/>
  + <http://flask.pocoo.org/>
  + <https://www.djangoproject.com/>
  + <https://ktor.io/>
  + <https://spring.io/projects/spring-boot>
  + <https://expressjs.com/>
  + <https://golang.org/>
* Client Side libraries tested
  + <https://angular.io/> + <https://material.angular.io/>
  + <https://reactjs.org/> + <https://material-ui.com/>
  + <https://vuejs.org/> + <https://vuematerial.io/>
  + <https://elm-lang.org/> + <https://github.com/debois/elm-mdl>
* Settled on rocket rust and react + material-ui with web pack and yarn

# 22/02/2019

* Created a material ui test project, with react and typescript
* Had problems learning how to use the typing system with material-ui and react, not well documented
* Decided to use Javascript instead of typescript

# 01/03/2019

* Created a simple rocket rest API for a test project
* Used react redux https://redux.js.org/basics/usage-with-react for state management, e.g. If I wanted to share information between react components

# 08/03/2019

* Decided on using react-router as a router instead of having a single page application or multiple links https://reacttraining.com/react-router/
* Moved from a rest api to a graphql api, https://blog.pusher.com/rest-versus-graphql/
* The primary reason for choosing graphql over rest is to get rid of redux in favour of graphql-apollo <https://blog.apollographql.com/the-future-of-state-management-dd410864cae2>
* Remade server in go for a couple of reasons
  + Go has its own web server built in, doesn’t require a library
  + Go uses github repos for dependency management, larger selection of libraries
  + Go is an officially supported graphql ready language https://graphql.org/code/

# 25/03/2019

* Restructured repository in order to conform to the golang project structure guidelines <https://github.com/golang-standards/project-layout>

# 01/04/2019

* Created separate repository for developing the javascript frontend for the application
* Started developing web server
* Setup git submodules to link the two repositories

# 05/04/2019

* Refactored pseudo code in order to better suit the requirements
* Planned CRUD features for database, based on certain features of the Django Framework
  + The admin site <https://docs.djangoproject.com/en/2.2/ref/contrib/admin/>
  + Database Models <https://docs.djangoproject.com/en/2.2/topics/db/models/>

# 09/04/2019

* Started development of GraphQL API
* Tried to implement Go GraphQL library <https://github.com/graphql-go/graphql>
* Had problems splitting the code up into smaller sub-functions, and sub-modules as examples mostly contain a single file and represent small scale applications
* Considered switching to a different language GraphQL implementation such as Java due to it being familiar and object oriented
* Decided to continue using Go, using a python Django like architecture, with a models package to hold type definitions
* Type definitions are used in the schema package which contains resolvers which serialize data from the database into json for the GraphQL api
* The graphql request and response is handled by the graphql library in the handler from <https://github.com/graphql-go/handler>

# 10/04/2019

* Had problems with the implementation for authentication and authorization in GraphQL
* The implementation from graphql-go handler did not work with authorization
* The web token that is passed in the header did not get passed into the context of the graphql application
* I decided to make my own graphql handler that implements the features

# 11/04/2019

* Refactored code
* Added intrinsic documentation

# 12/04/2019

* Created a go package called env, which is used to store local data such as the secret key used to sign a web token
* The constant variables are stored in a json file called config.json which is serialised into struct using native go libraries

# 14/04/2019

* Http handler implemented did not work
* Cloned <https://github.com/graphql-go/handler> and added a small section which passes the token in the context of the graphql query parameters

# 15/04/2019

* I created project for the frontend utilising react router and react to create prototype login screen using the material ui library.
* Switched from rolling my own setup and project configuration of the react project to a preconfigured option with create react app.
* https://facebook.github.io/create-react-app/docs/getting-started

# 16/04/2019

* Refactored server functions
  + Instead of recreating the same function three different times per user type, I switched merged into a single function, which uses multiway selection to differentiate between users.
  + Implemented a new switch in my login screen to login as different user types.
  + Switched the form submission from the traditional html submit method into a react state management solution.

# 17/04/2019

* Instead of using a html switch component to handle multiple user logins, I use the url name and pass it to my graphql component

# 19/04/2019

* Completed the login screen, but contains routing and graphql errors
* The program returns a buffer overflow with to many calls to state
* The bug is ambiguous so I can’t tell what the source of the error is.
* Refactored backend functions while I think about a possible solution to the error.

# 24/04/2019

* Authorization and authentication functions in the backend API is finished, tested with graphql playground.
* Recreated the entire project functionality in typescript for compile time typechecking.
* Remade the assignment in a functional programming style using react hooks for state management.
* Loops are replaced by iterators that use callback functions inorder to modify data
* Removed all complicated classes for react components because they were confusing with too many properties and only one render method.
* Removed usertype state variable because it never changed on a form and state variables are usually for dynamic data on screens.
* I refactored my server side code to contain some object oriented features such as to share data between packages, I use a getItem function encapsulating the code.

# 25/04/2019

* Refactored frontend
* Added type definitions for various components passing all parameters with a corresponding type definition.
* Used the component props and interface utility.
* Switched from Apollo to Apollo-hooks because it decreased the size of the react components.

# 26/04/2019

* Tested authentication frontend with the backend.
* Finished authentication.

# 27/04/2019

* Fixing backend bugs, logic errors where post requests for getting a list of classes would return incorrect values.
* Removed the env.json file.
* Hard coded certain components into the program.
* Created a function that creates a single instance of a given constant and can share a pointer of it with various functions within the server.
* Made a token verification function in the graphql api which verifies a users token.
* Made the secret key a function that updates every time the server restarts, making the program less convenient but more secure.

# 30/04/2019

* Made all userType variables into the same usertype lowercase variant.

# 02/05/2019

* Started working on a timetable function that handles terms, public holidays and events.
* The timetable has a term type that defines what term it is right now.

# 04/05/2019

* Remade year configuration as graphql input types in order to make the timetable crud

# 08/05/2019

* Removed usertype variable and replaced with role
* Merged the admin, teacher, and student table into one table with an extra column of role
* Had a weird problem where the version of the project existing on my computer was not updating even though I had rebuilt the project.

# 09/05/2019

* Looking for a way to fix bug, rebuilt the project multiple times
* Modified the code, still not displaying any changes on web browser
* Restarted computer clearing cache

# 10/05/2019

* The project works, the entire website is built in javascript so browsers store the javascript in cache in order to avoid having to rebuild the website every time the page is refreshed, the cache held an old version of my program.
* Setting up the year configuration for my program, to hold terms, public holidays and such things.

# 11/05/2019

* Create Year configuration does not work, input type not supported error

# 12/05/2019

* Scrapping year configuration idea, was not working
* Year config will have to be created in the database natively
* Started working on CRUD for managing users on the server

# 14/05/2019

* Finshed CRUD user management on the server

# 20/05/2019

* Authorization and authentication bug fixes on the client side
* Removed the admin dashboard, and reroute to a single dashboard component which is the teachers dashboard

# 22/05/2019

* Implemented an app bar for the teachers dashboard, added a login and logout button for the teachers dashboard.
* The login and logout button did not work.

# 24/05/2019

* Implemented a toml configuration file to hold server configurations

# 26/05/2019

* Made the toml configuration a yaml configuration and had it hold a secret key, and a server configuration