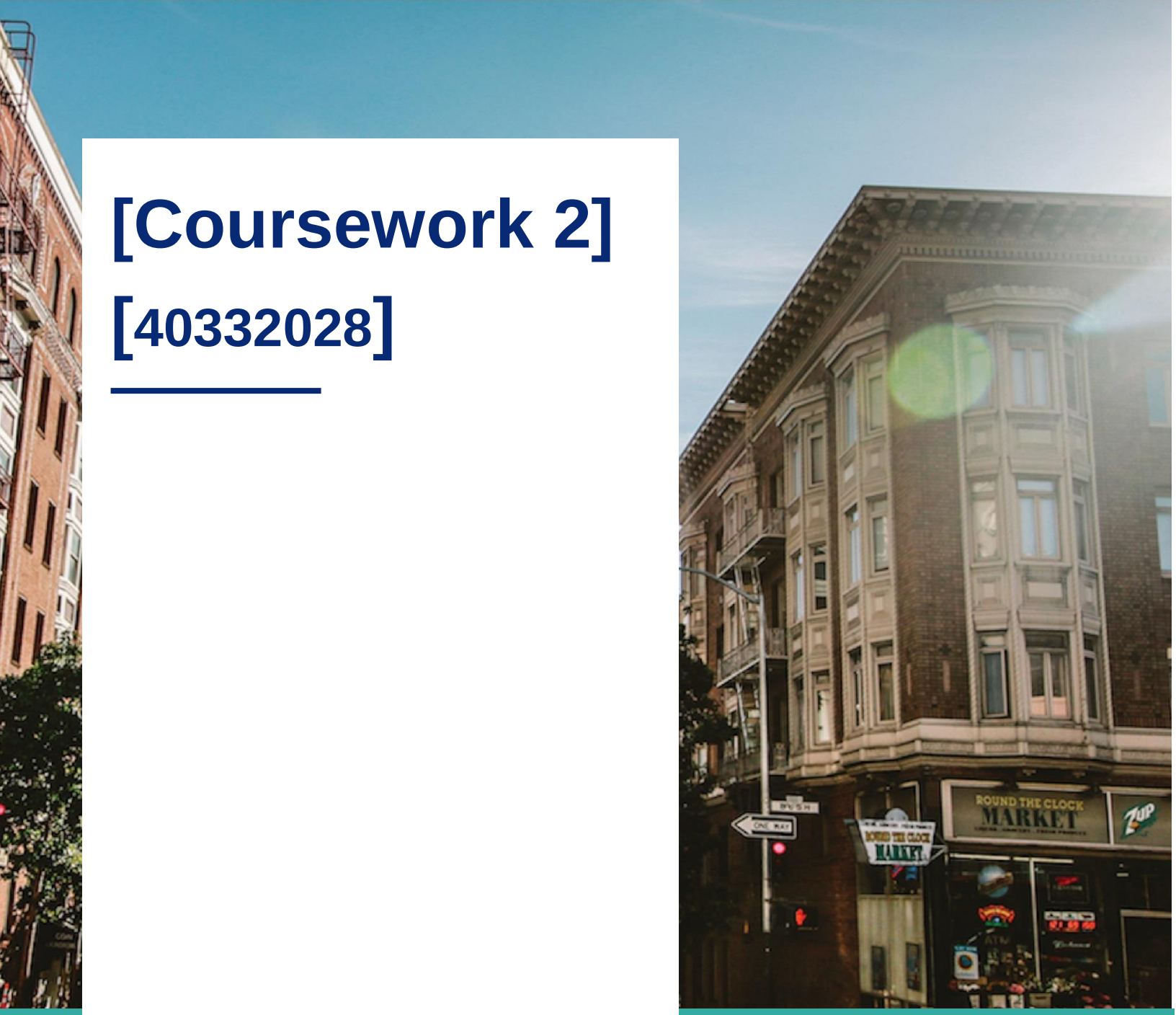

[Coursework 2]

[40332028]

[18/04/2019]

Authored by: Amund Søyland - 40332028

Introduction.

In this project the JavaScript code for encoding and decoding from CW1 has been adapted and further developed into a simple client-server web-application for sending, receiving, encoding and decoding of messages. The client-server implementation was achieved by using Node.js along with the express framework.

The web application enables a user to create an account, login, send messages to all other users and the ability to read said messages. Although the implementation is rudimentary and the visual design, while not abhorrently off-putting will probably not be considered to abide by best standards and practices in interactive web design.

Software design.

The application is built on an autogenerated express framework, this is done to avoid having to do the groundwork every time, some additional modules have been installed to enable additional functionality; these modules are Bcrypt and Sqlite3.

Bcrypt was added to enable easy hashing functionality for comparing and storing hashed versions of passwords, and while it adds security is not strictly speaking a vital component for the function of the application.

Sqlite3 is the database module in use, and unlike Bcrypt is wholly essential for the dynamic functions of this web application, all user data sent to the server (except for cookies) gets evaluated and stored in the database served by sqlite3.

Most of the development has been in the routing and rendering of the .pug files associated with express, the original html site was broken up and repurposed for pug layouts.

Critical evaluation

One thing that is quite apparent when looking back on the development of this app is that my knowledge of node and express were very limited when I started writing this app, therefore it suffers from less than optimal design, and vestiges of things repurposed for other things. One of my biggest oversights was probably the lack of more general functions built for the app, as such the routing functions have become very messy.

Specifications

The scope of this app was always to fulfill the most basic requirements of the coursework, with plans to add some extra features that would be easily implemented like password encryption.

Improvements

This app has much room for improvement. The visual design is rather spartan, and the routing still bears the name of the default generated routes in express. The way messages are displayed could also really do with some nicer visual formatting.
An attempt was also made at keeping the views structured in an inside and outside folder, but this fell apart as small changes were made during development.

Personal evaluations

When I started developing this app my knowledge of node was very limited, but after doing some of the labs at home I gained the confidence to get started on the coursework. The pug templates we're not immediately intuitive, but after playing around with them it suddenly clicked, and after that it was a lot easier to get a wholistic view of what was needed.

I also had good use of my previous scripting experience when dealing with certain challenges pertaining to JavaScript. I also got to apply my knowledge of SQL to a more limited degree.

The list of "new" things I learned isn't particularly long, but it did help me put much of my previous knowledge into practice and put it into the perspective of web applications.

References

pugjs - Pug language reference

Expressjs - Express framework reference

MDN JS - Mozilla dev JS reference

Bcrypt - Hashing module

Sqlite3 - doc - Database module and documentation

pughtml - HTML to PUG converter