Yew Brown
33880799
BSc Creative Computing
Dynamic Web Applications
11.12.25

Lab 10: 10_health_33880799 Documentation

Outline

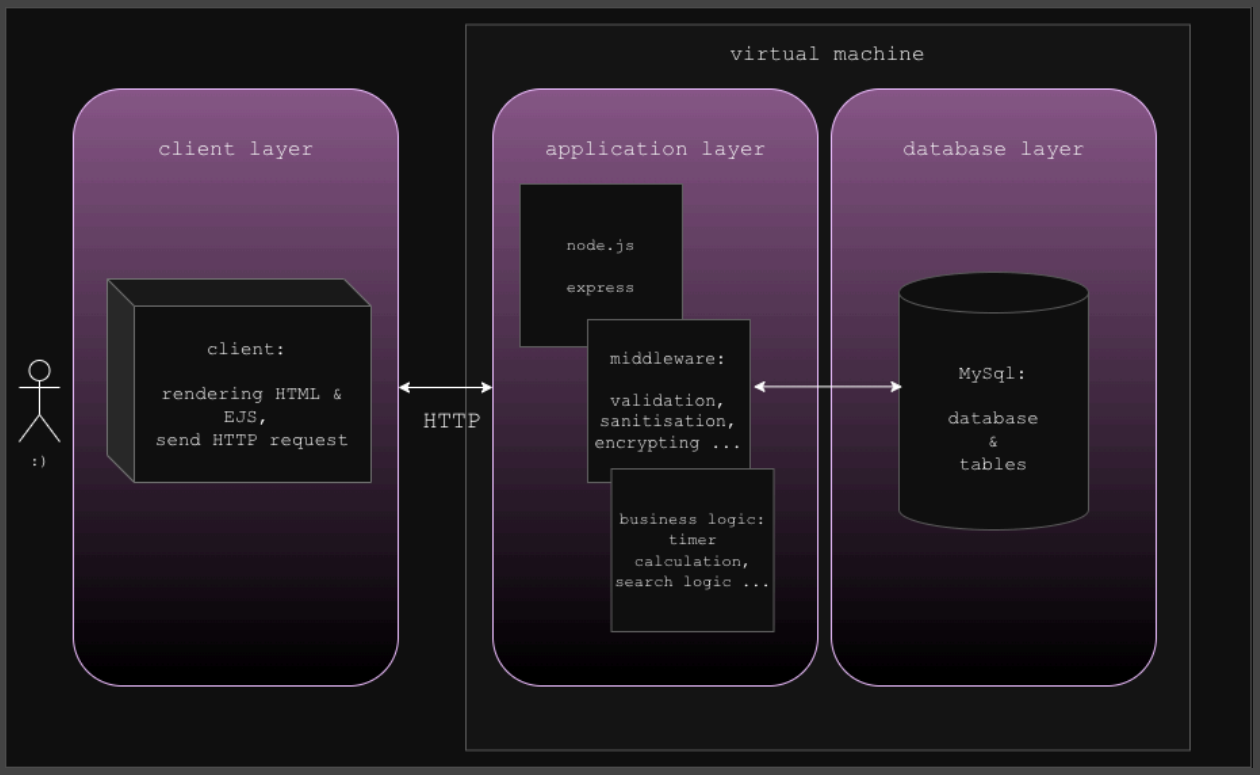
The name of the application is *mustard seed*.
mustard seed is a dynamic web application built using a core technology stack of Javascript, Node.js, Express, EJS and MySql. *mustard seed* handles the input of user data to personalise its functionality.

Building on the theme of Health and Fitness, *mustard seed* is an application that aids its users in their meditation practice.
Its features, such as the timer and calendar, allow users to schedule and track their meditation sessions. The resources section of the website also provide users with reading material about meditation and spiritual practice.
It utilises user data securely stored in its database to customise individual user experiences, received as preferences and information directly input by its users.

mustard seed allows users to create user accounts. These accounts are used to access its full features, acting as a means of authorisation and validation to do so through user sessions.
User accounts also act as the means to store user data in the database securely against the account, enabling the application to provide its personalised services.

Architecture

The web architecture of the application involves three layers: a Client Layer, Server Layer and Database Layer.



The Client Layer is where the application runs in the user's browser. Interactions between it and the Server Layer via HTTP are stateless and do not retain data.
It typically deals with processes such as rendering HTML and EJS files.

The Server Layer is where the application logic and processing is largely executed.
This is achieved by communicating with both the Client Layer and Database Layer and executing middleware functions. Including processes such as handling GET and POST requests, sanitisation, validation and interacting with the database via SQL queries.

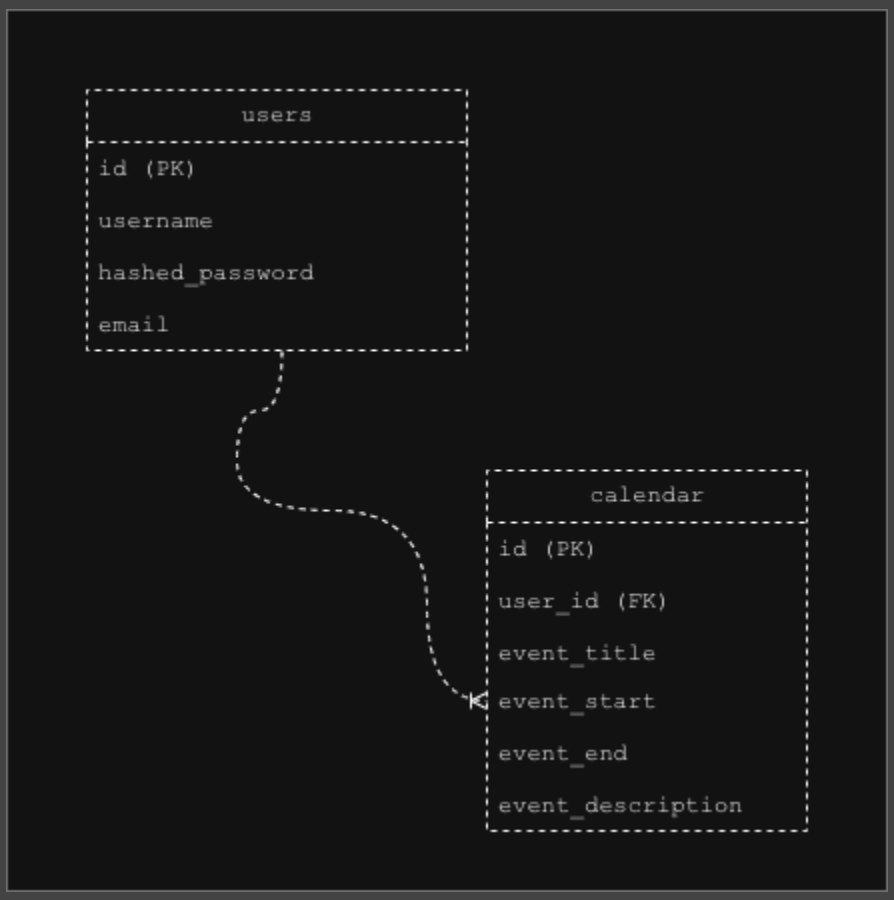
The Database Layer is where necessary user and application data can be stored longterm. Its contents can be interacted with by the Server Layer of the application or manually via command line and software.

It houses the essential data for the application’s dynamic functionality.

Data Model

This application uses a relational database model. It contains two tables - *users* and *calendar* - both identify their records using primary keys in the form of an ID, and are linked by the use of the *users* primary keys as foreign keys to connect calendar records to the specific users that created them.

Having this relational modal enables lots of one to one or one to many relationships between users and calendar records. The data stored can then easily be served back to the users as their own stored data and preferences.



User Functionality

The user functionality of *mustardseed* with dynamic backend features includes user accounts, search functionality and an interactive and

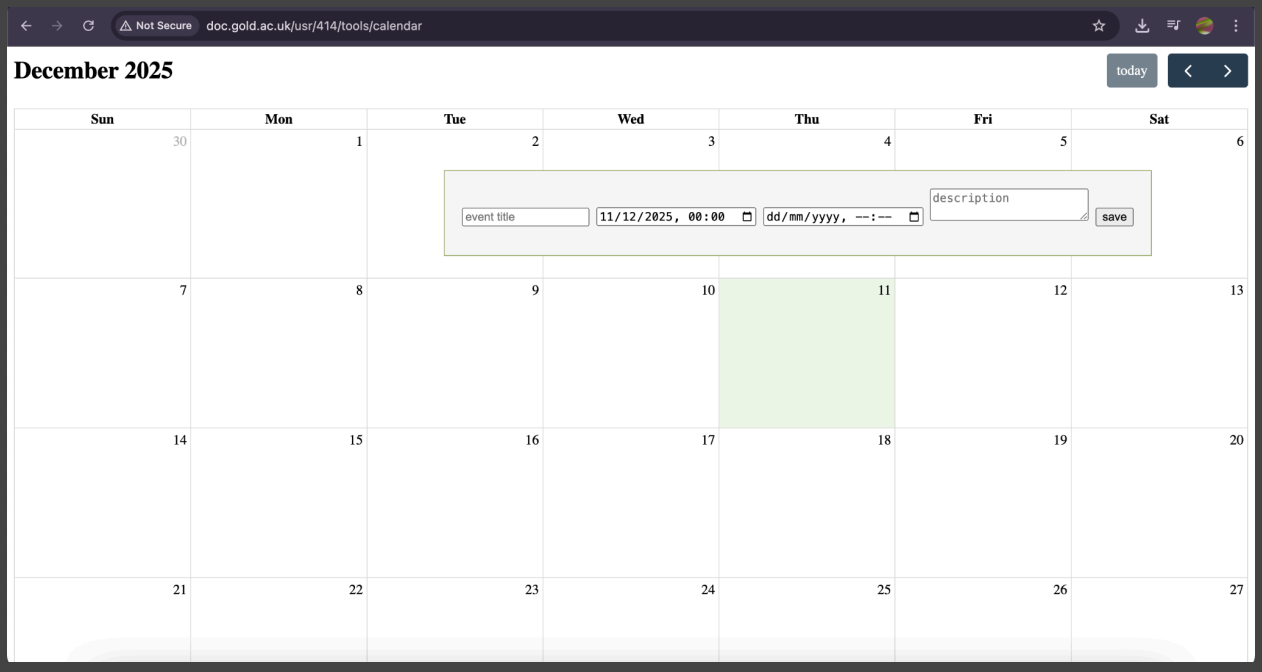
customisable calendar. Some other features that are entirely frontend and/or middleware that could be extended for more dynamic, data-driven user interaction are the timer and reading sections of the site.
User accounts are interacted with via a sign up page to create them, a login page, log out page, search features and the calendar feature.

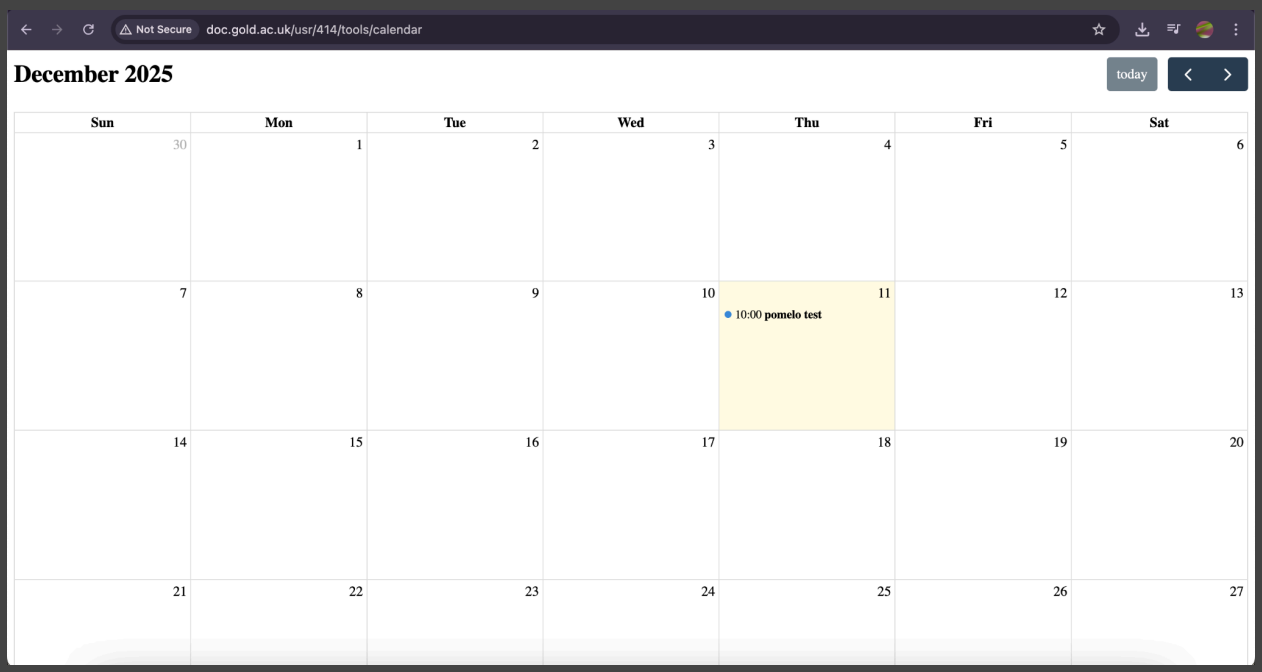
Upon creating an account with sign up, users' usernames, hashed passwords and emails are stored in the database under the table *user*. The stored user data is then used to create user sessions, validate access to application features, for example - the calendar and logout features, and store calendar data against their user record through the relational data model.

The search features allow both registered and unregistered users to look up registered users in the database, as a primitive community feature. The search only returns users' usernames, so does not include sensitive or identifying information. An initial search bar is found on the home page, which leads the user onto a dedicated search and results page after searching.

The calendar feature utilises user input via API to communicate between the front and backends of the application. User data is stored and retrieved from the database by user interaction with the calendar - users are able to input events (a title, times and

description) into the calendar. Upon revisiting the calendar their stored data will be displayed back to them.
The calendar is restricted to registered users only. It will also only show users their own data and only allow them to input data to their own calendar. Both access and user-calendar relationship are validated using user sessions, created at login.





mustardseed also features functional frontend timer and reading corner features, although they are not dynamic and do not use userdata currently. Allowing user data to shape the user interactions with both would be the next steps in development.

Advanced Techniques

Advanced techniques used in development of *mustardseed* are validation, sanitisation, encryption, sessions and API.

Validation, sessions and sanitisation were both used to improve the security of the application.
Validation in the form of user sessions created using users' IDs from the database, allow me to restrict user access to certain features for security and functionality purposes.
Santisation prevents attacks such as cross site scripting (XSS) and SQL injection by scrubbing user inputs such usernames, passwords and emails of excess characters without breaking them and affecting their functionality.
Encryption is used to disguise user passwords using hashing and salt to protect against brute force attacks. Only encrypted passwords are stored in the database, so no one but the original user can know what they are. Logging in then requires decryption by the middleware functions.
Sessions go hand in hand with these two to verify users. Beyond security they are also used to maintain users' interactions with their own personalised features across stateless HTTP interactions - as in the case of the calendar.

```
router.post('/signedup', [
    body('username') // santise username to avoid XSS
      .trim()
      .escape()
      .isLength({ min: 4, max: 20 })
      .withMessage('username must be 4-20 characters long'),
    body('password') // secure password check
      .isStrongPassword({
        minLength: 4,
        minLowercase: 1,
        minUppercase: 1,
        minNumbers: 1,
        minSymbols: 1,
      })
      .withMessage(
        'password must be at least 4 characters long, including one each of uppercase, lowercase, number, and symbol'
      ),
    body('email') // ensure email is valid then sanitise
      .trim()
      .isEmail()
      .normalizeEmail(),
  ],
function (req, res, next) {
    // saving data in database
    const saltRounds = 10;
    const plainPassword = req.body.password;
    const errors = validationResult(req);

    if(!errors.isEmpty()) {
        res.render('./signup');
    } else {
        // encrypt password
        bcrypt.hash(plainPassword, saltRounds, function(err, hashedPassword) {
            // store hashed password in your database.
            if(err) {
                next(err);
            }
```

API was made use of in *mustardseed* as a way of communicating between the frontend, client side of the application and the server side. It passes user input data and data retrieved from the database between the client and server layers using json. Displaying users' calendar entries to them from the database and passing new calendar entries to the middleware function on the server to be inserted into the database.

```
// add event to calendar
router.post('/calendar/events', redirectLogin, function(req, res, next){
    const userId = req.session.userId;
    const title = req.body.title;
    const start =  req.body.start;
    const end = req.body.end;
    const description = req.body.description;

    const sqlQuery = 'INSERT INTO calendar (user_id, event_title, event_start, event_end, event_description) VALUES (?,?,?,

    db.query(sqlQuery, [userId, title, start, end, description], (err, result) => {
        if(err) {
            next(err);
            return;
        };

        res.json({ // send data as json to FullCalendar front end
            eventId: result.insertId,
            title,
            start,
            end,
            description
        });
    });
});
```

AI Declaration

My direct use of AI throughout the development of this project has been limited to only ChatGPT.
I have used it to assist me much like I would a tutor, by asking it for help and examples when I am stuck - as a learning resource.

I have not prompted it to generate significant sections of code or any of the documentation(README.md, links.txt or this document). With the slight exception of leaning on it more for the frontend, in order to speed up the process, as that is not what we are being assessed on.

I am personally against the use of AI to bypass meaningful, actual learning and work. I have every intention of learning the contents of my course and modules fully through applying my own knowledge and skills to my assessment work.