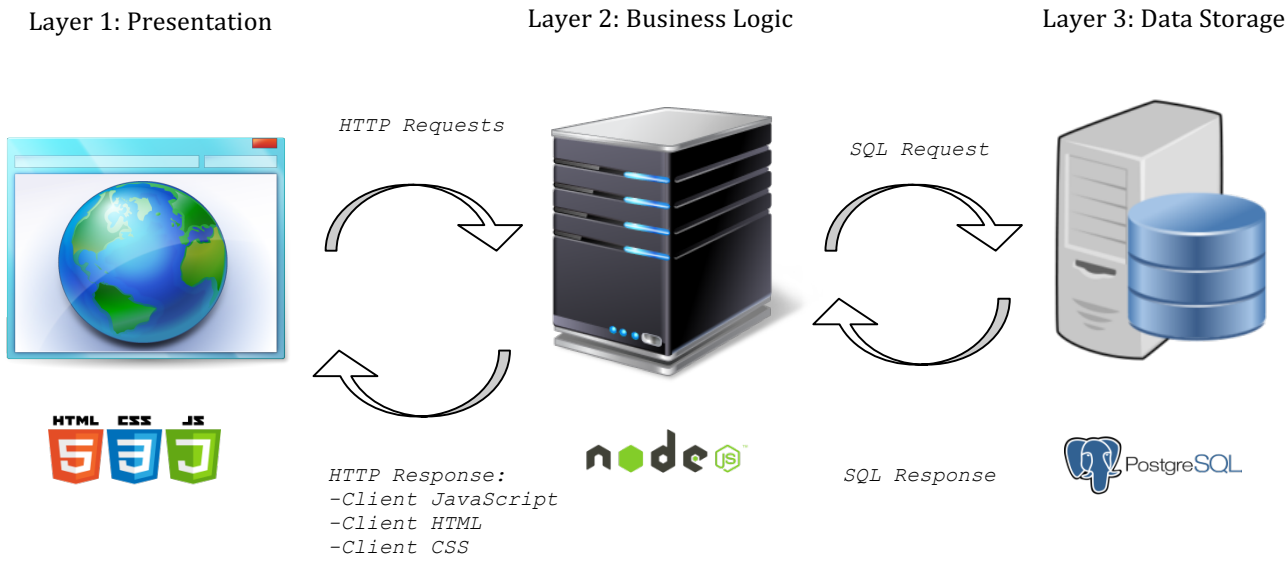
BookMi – Project Report

url: http://bookmi.herokuapp.com

Overall Design

The BookMi web application follows the *3-tier architecture* as the user interface, business logic and data storage are all developed and maintained as three independent layers.



There are two fundamental interactions throughout this web application: those between layers 1 & 2, and those between layers 2&3.

Layer one issues HTTP requests from the layer 2 for the necessary HTML, CSS and JavaScript files to be rendered by the client browser – this is what the client sees and uses to interact with the web application.

Layer two issues data requests in the form of SQL queries from layer three and processes the response accordingly based on the previous interaction above.

Security Vulnerabilities

In the BookMi application, we looked at the following security vulnerabilities:

1. **SQL Injection**

   An SQL injection is an attack where a malicious attacker tries to tamper with a database by inputting SQL commands directly into input fields for execution.

   Bookmi mitigates the risk by using parameterized queries for all our database queries. This is a node-postgres method of submitting a query that ensures no SQL injections can be made.

   Instead of:

   ```
   client.query('SELECT * FROM Users WHERE id=' + user_input,....
   ```

   We use:

   ```
   client.query('SELECT * FROM Users WHERE id=$1', [user_input],....
   ```

   Attempting to put the following string into the sign in page will return an error as the site checks for valid usernames first (< 20 chars) and also runs the query parameterized and will return "Username or password is not correct."

   ```
   admin' UNION select id, username, ...hash..., user_type FROM users
   WHERE user_type = "admin" LIMIT 1 --
   ```

   The parameterized queries are parsed, analyzed, rewritten and planned before execution, providing safety against SQL injection attacks.

2. **Cross-Site Request Forgery (CSRF)**

A CRSF attack happens when a malicious agent causes a user's own browser to make unwanted requests to a website where the user is currently authenticated. The attack is particularly harmful when the user is an administrator on the site as they can make requests that impact many other users.

BookMi mitigates this risk by using the csurf (https://github.com/expressjs/csurf) protection middleware. Csurf generates a token that is stored server side and also returned to the user with each page. The token is then returned when the form is submitted and csurf verifies that the token matches before allowing the request. The token is not visible to other browsers or malicious software.

We implement the token by embedding it into the form

```
<input type="hidden" name="_csrf" value="{{ csrfToken }}">
```

and also the meta tags for dynamic AJAX POST/DELETE requests

```
<meta content="{{ csrfToken }}" name="csrf-token" />
```

Because malicious applications do not have access to the csrf token, trying to submit any form requiring such a token will return "403 Forbidden". CSRF protection has been added to all methods requiring a POST/DELETE.

For example: `POST /profile?id=39` will return 403 Forbidden if a CSRF token is not included.

3. Other General Security Concerns

- Passwords: We try to address password security by requiring users to use at least an eight-digit password. In addition, passwords are hashed and stored using bcrypt so that in case the passwords database is compromised, user security is not.

- Input validation: In general, Bookmi does sanity checks on all user input. Checks are provided on the front-end using HTML5 validation. The inputs are also validated on the server side to prevent users from modifying the HTML and JS to access the server side maliciously.

For example:

- Uploaded images are verified by reading the magic numbers and not just reading the extension to prevent malicious uploads. Files are renamed when stored.
- Messages to private users and reviews cannot be longer than a certain number of characters.
- Usernames cannot be longer than 20 chars.
- Valid emails, ages, gender must be provided and verified on the server side.

Performance Improvements

Using the *3-tier architecture* for the website application, provided a valuable framework for identifying areas of operation that could be further optimized.

Layer 1: Presentation

- Initially, the entirety of the presentation was meant to be through the use of "hides" and "shows" in Javascript to prevent the number of requests to the api. However, this resulted in blank screens while the information was loading, without the end-user being aware of what has happening. Thus the entire presentation was adapted to refreshes, making the end-user aware of the loading process and hopefully mitigating frustrations and confusion during waiting times between requests.

Layer 2: Business Logic

- Efforts were made to use asynchronous execution of code (via the "async" library for NodeJs) to prevent I/O blocking and the use of excessive logic in order to guarantee the execution of queries and GET requests in a particular order.
- The number of CSS style rules were shortened due to their impact on overall performance speed.
- JavaScript files were minified in release-mode to minimize the file size being sent over by the NodeJS server.
- There was a notable lack of efficiency in our recommendations and feed page as they were being treated as separate requests then redirected to one another.

In order to improve speed, we combined the code in order to minimize requests and prevent redirects that take up unnecessary time.

- Initially, there was a significantly greater number of GET requests to the Google Books API that were rather unnecessary. In order to improve performance and capacity, we decreased the number of requests made by using the Async module.

Layer 3: Data Storage

- In the relational model for the application database, schemas were refactored to eliminate any update, or insertion anomalies, which provided addition support for insuring data integrity and minimal record storage (which in turn leads to faster searches within the system).

App Enhancements

- ***News Feed***: Users for the BookMi web application can see an activity feed for other users they are following which enhances the overall user experience. It allows users to keep up to date with the users they follow so that the following functionality has a clear and userful purpose, and is implemented on a users home page, reached upon signing in.

- ***Responsive Design***: The web application implements a responsive design so that users may have an equally enjoyable experience using the web application, whether on a desktop computer, tablet or a Smartphone device.

- ***Aesthetic UI***: Having an aesthetic user interface allows the user to have both an intuitive and enjoyable experience while using this web application.

- ***Book Reviews***: Users are allowed to read and write book reviews which add to the user experience as they have another way to participate within the BookMi social community. Furthermore, it prevents the user from having to go outside the app in order to find inspiration of

which books to read as they can rely on the users they follow for their reviews on their favourite or least-favourite books.

- ***Nunjucks***: The usage of the Nunjucks templating engine for JavaScript enforces backend modularity which in turn provides additional support for the overall application development process.

- ***Image Upload***: Users are able to upload their own profile pictures for the BookMi website application. This enhances the user experience as they are given another way in which they can perform some meaningful personal customization to their profiles and share it within their community.

- ***Information Hover:*** When displaying book images, when hovering over the book, its title and author information are presented in a floating text box. The end-user no longer needs to click the book in order to read the most basic of information about it. And when no image is provided for a book, the user still has the ability of determining if they are interested in the book.

Things We Could Change If Given More Time

1. Due to the initial restrictive nature of Heroku, there was a fixed limit on how much data a hosted database could offer (1,000 rows for free hosting and additional costs for unlimited data storage). This means that the overall evaluation of scalability for this web app could not be discretely tested. However, due to the *3-tier architecture* design and the elements outlined within the "Performances" section of this report, there is evidence to support the claim that this web application is highly scalable – a desired feature to have for modern day usage.

2. The BookMi web application issues a number of GET requests to Google RESTful API to retrieve book information. While this may be fine as far as the course project may be concerned, a further extension for this web app would be to store book information on

our own database, thus limiting the number of necessary GET requests mentioned above to retrieve book information (authors, title ...etc). This would allow less reliance on external APIs and ensure further scalability, as currently there is a maximum number of GET requests that can be made to the Google Books API per day.