Using the Template

This document provides detailed guidance on using the CM2040 Database Networks and the Web mid-term coursework template.

Quick Start

Follow the instructions in the `README.md` file to set up your computer and install and run the template.

Contents of the Template

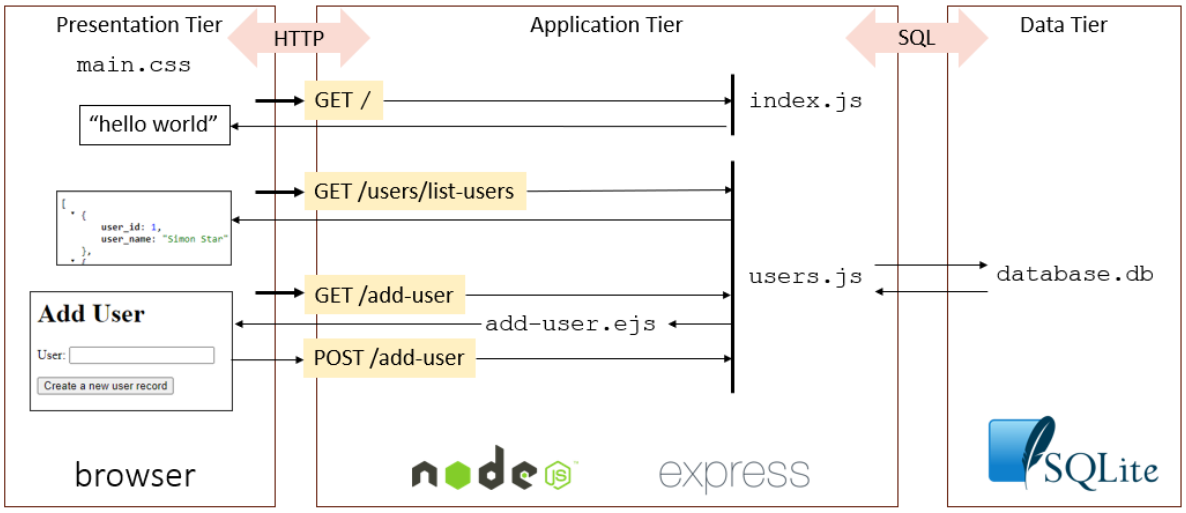
The following table describes the contents of the template:

Component	Description
`public/` `main.css`	The `public` folder contains static files, such as css files and client-side javascript, that you want your web server to serve.
`routes/` `users.js`	The `routes` folder contains your middleware route handlers. Add files here to implement specific middleware functionality that you want. In the template, a sample handler for user functionality is provided. You may wish to develop that further if your application requires user functionality, or remove it if it doesn't.
`views/` `add-user.ejs`	The `views` folder contains your EJS templates, which are used to render your data-driven web pages. Add additional EJS files here and develop or delete the add-user.ejs file as required.
`index.js`	This is your main application start point. It sets up Express connects to your database, loads all your route handlers and starts your web application listening for HTTP requests. Add calls to your route handers here.
`package.json` `package-lock.json`	These tell node.js what packages and versions of packages you are using as well as how to run your project. You shouldn't need to edit these.
`db_schema.sql`	This defines your database structure. When you run `npm run build-db` or `npm run build-db-win` a new database is created (in a file called `database.db`) and the contents of this file are exectuted against the database. Build your data model here. Changes made here won't be applied until you exectute them against the sqlite database.
`.gitignore`	If you use git, this file will exclude files you don't want to commit to git.
`README.md`	Contains instructions for setting up your computer and running the sample code. You should update this document so that it provides suitable instructions for the application that you develop.

The Running Application

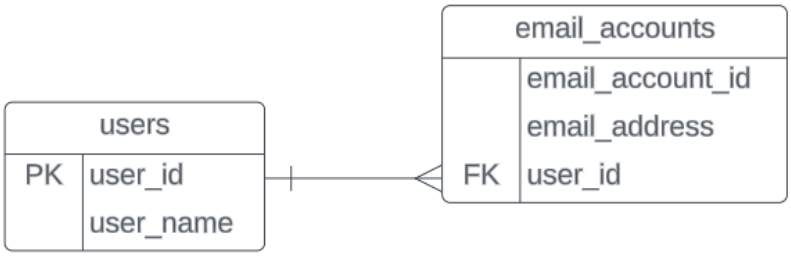
The template as it stands partially implements a dynamic web application that manages users and their email addresses.

The following diagram shows how the interaction between the layers of the 3-tier architecture are implemented in the template, together with the application endpoints:



The Data Model

The following is the underlying data model, implemented in the file `db_schema.sql` :



The following data is inserted into the two tables within the same `db_schema.sql` file :

users:

user_id	user_name
1	Simon Star
2	Dianne Dean
3	Harry Hilbery

email_accounts:

email_address	user_id
simon@gmail.com	1
simon@hotmail.com	1
dianne@yahoo.co.uk	2

Working with SQLite

SQLite works a little differently to MySQL but all of the key concepts are the same.

You can find the documentation for the sqlite3 node module here:

https://github.com/TryGhost/node-sqlite3/wiki/API

You can find tutorials on using the sqlite3 node module here:

https://www.sqlitetutorial.net/sqlite-nodejs/

For a general tutorial on using SQLite start here:

https://www.sqlitetutorial.net/

The SQLite Command Line

When you install SQLite, you will have a command line tool installed. You can use this to run SQL commands on your database so you can check the contents of tables, check the structure of the database and practice your SQL queries before embedding them in your node.js code.

Start the command line tool by typing `sqlite3 database.db` from your command line or terminal, ensuring you are in the project directory containing your database:

```
c:\>sqlite3 database.db
SQLite version 3.38.5 2022-05-06 15:25:27
Enter ".help" for usage hints.
```

Ensure you turn on the option to enforce foreign key constraints:

```
sqlite> PRAGMA foreign_keys=ON;
```

You can see a list of tables in the database by using the `.tables` command:

```
sqlite> .tables
email_accounts  users
```

You can see a table definition by usin the `.schema` command:

```
sqlite> .schema users
CREATE TABLE users (
    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_name TEXT NOT NULL
);
```

You can also type in SQL statements, such as a SELECT command:

```
sqlite> SELECT * FROM users;
1|Simon Star
2|Dianne Dean
3|Harry Hilbert
```

If you want to dump our the entire database schema, including the INSERT commands used to insert data, you can use `the .dump` command:

```
sqlite> .dump
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE users (
    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_name TEXT NOT NULL
);
INSERT INTO users VALUES(1,'Simon Star');
INSERT INTO users VALUES(2,'Dianne Dean');
INSERT INTO users VALUES(3,'Harry Hilbert');
CREATE TABLE email_accounts (
    email_account_id INTEGER PRIMARY KEY AUTOINCREMENT,
    email_address TEXT NOT NULL,
    user_id  INT, --the user that the email account belongs to
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
INSERT INTO email_accounts VALUES(1,'simon@gmail.com',1);
INSERT INTO email_accounts VALUES(2,'simon@hotmail.com',1);
INSERT INTO email_accounts VALUES(3,'dianne@yahoo.co.uk',2);
COMMIT;
```

To exit the SQLite command line, use the `.exit` command:

```
sqlite> .exit
```

Modifying the Schema

You will need to modify the database schema to implement your application. You **must** do this by modifying `db_schema.sql`. This allows us to review and recreate your database simply by running npm run build-db. Do NOT create or alter database tables through other means.

Suggested Next Steps

Spend some time working with the template:

- Explore the file structure and code
- Read all the comments
- Try accessing each of the routes via the browser - make sure you understand what they do
- Try adding a new route to display all the email accounts for a particular user
- Try adding a new form to allow new email accounts to be added for a user

Once you are comfortable with the template you can start to plan your implementation of the mid-term project. You might start working on wireframes for your web pages and designing the data model before trying to write the node.js code.