Secure Programming Project

Paavo Kemppainen

Topics

- Project Goals
- Used Security Technologies
- Testing the Security Features
- Demo: Login
- Demo: Profile Update
- Extra Thoughts

Project Goals

- Create a Node.js + React Single Page Application.
  - React client makes HTTPS requests to the Node.js API which uses PostgresSQL database to serve data.
- Implement Security Features on the application.
- Test implemented Security Features.

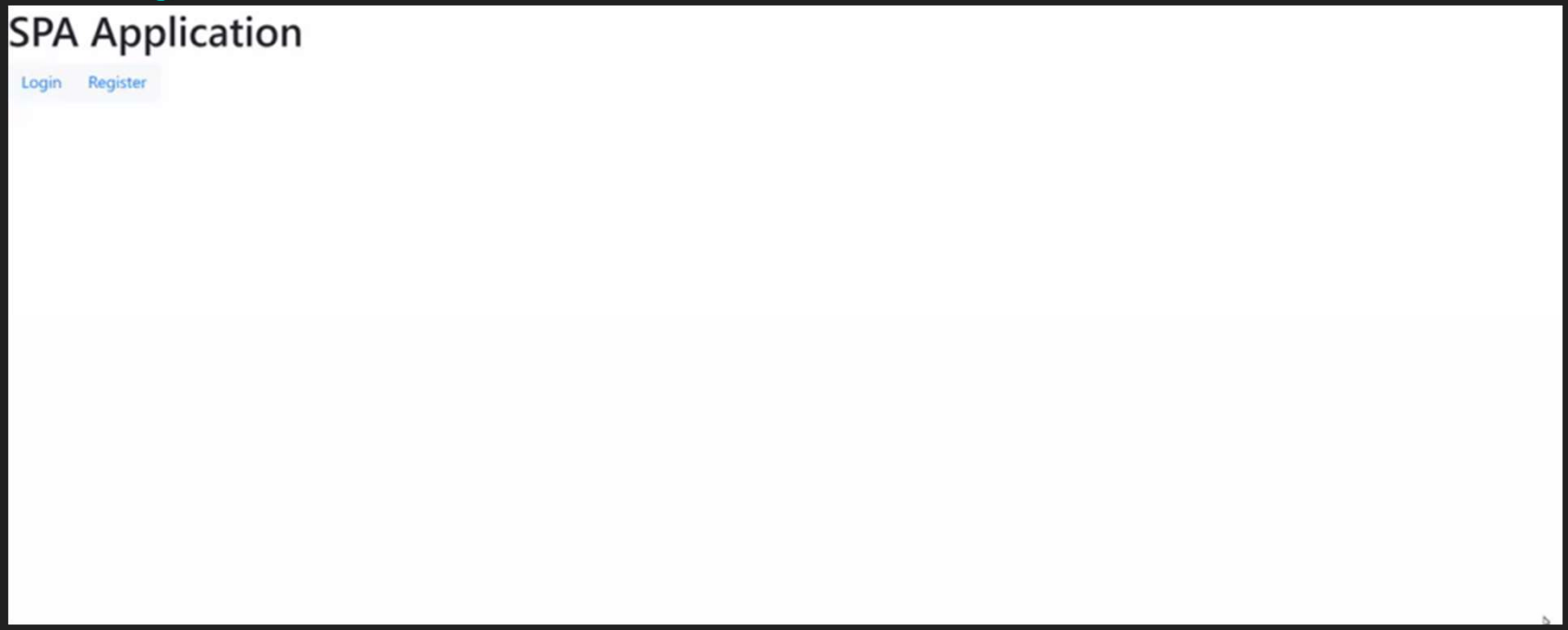
Used Security Technologies

- Passport: Used for authentication and log in session management.
- HTTPS: Used for encrypted communication between the React client and the Node.js API.
- Parametrized SQL Queries: Prevents SQL injections in to the database. (Done using packages ‘pg’ and ‘pg-format’)
- Bcrypt: used to hash and salt the passwords before they are saved in to the database. Also used when comparing login password to saved passwords.
- Static Code analyser: Used to find any obvious code errors.
- Jest+supertest: Used for unit testing the security features.

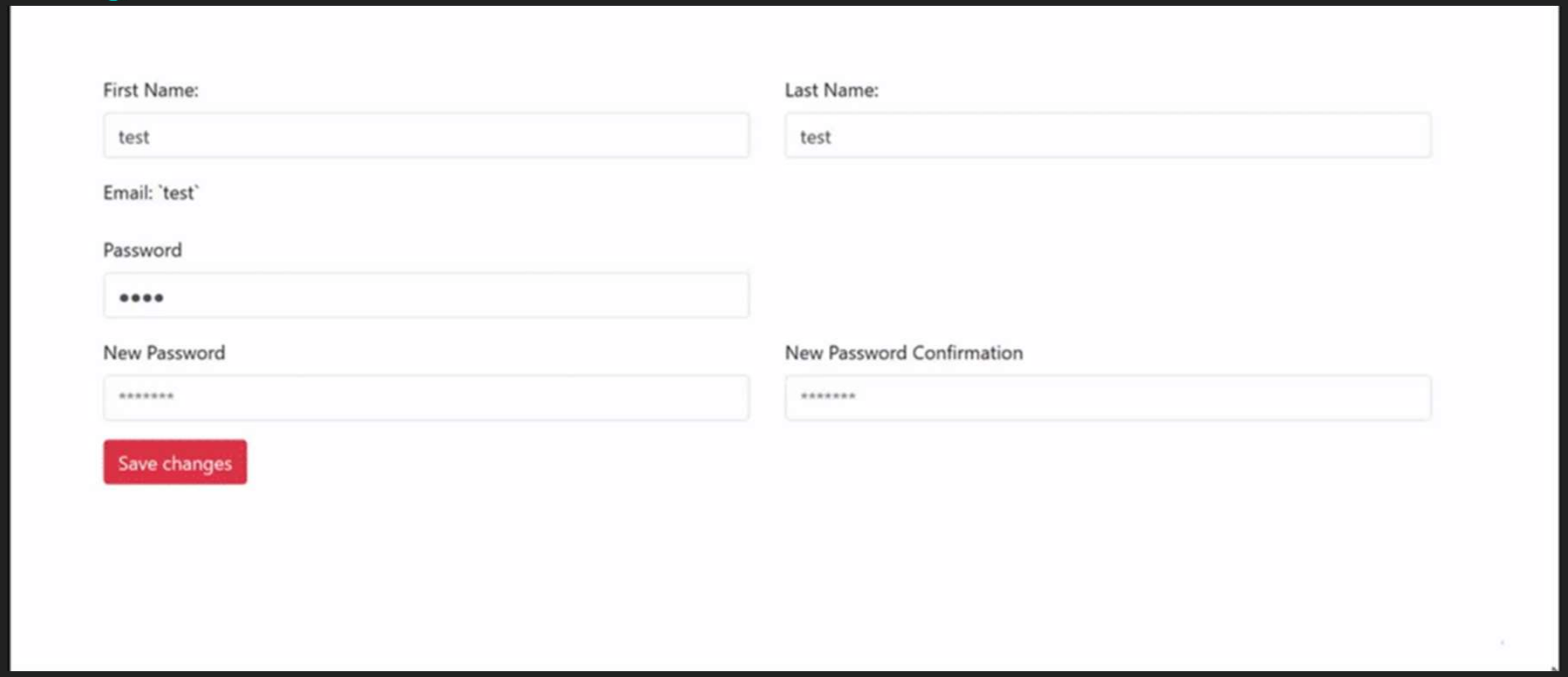
Testing the Security Features

- Backend Features: Jest unit test framework was used for internal testing of the database functions.
  - Results: SQL injection is not possible.
- Backend API: Jest+Supertest were used to test the HTTPS requests to the API and their security.
  - Result: Only authorised users have access to their profile. Only registered users can login.
- Library/package dependency vulnerabilities were tested with Grunt and ‘npm audit’
  - Most of the library dependency vulnerabilities came from the testing tools themselves.

Demo: Login



Demo: Profile Update



Some Extra Thoughts

- A lot of time was spent on learning new technologies and methods.
- Many tools cost money (especially in testing), so finding the right tools took some time.
- Compatibilities between testing tools and Node.js packages gave some trouble. (Automated session management testing has some configuration problems.)