Password hashing

Welcome to this lab activity

In this lab activity, you will modify your register form to capture a username and password when a user registers on your web application. You will then store the user's registration details in the database, applying hashing to encrypt the password to secure it.

Task 1: Make a copy of your previous lab

You will start with your previous lab code.

1. Make a copy of the folder `18_database_search` from your previous lab and call it `24_password_hashing`.
2. Open the new folder in Visual Studio Code.
3. Run the code and confirm that it works as expected.
4. In particular, run the `/register` route to remind yourself about what it does.

Task 2: Modify the register form

5. Modify the register form in `register.ejs` so that it additionally captures a username and password. Your new `register.ejs` should look like this:

```
<!doctype html>
<html>
  <head>
    <title>This is the register page title!</title>
  </head>
  <body>
    <h1> This is the register page </h1>
    <form method="POST" action="/registered">
        <p>First Name: <input id="first" type="text" name="first"/></p>
        <p>Last Name: <input id="last" type="text" name="last" /></p>
        <p>Email: <input id="email" type="text" name="email"/></p>
        <p>Username: <input id="username" type="text"
name="username"/></p>
        <p>Password: <input id="password" type="text"
name="password"/></p>
        <input type="submit" value="OK" />
    </form>
  </body>
```

```
</html>
```

Task 3: Create a database table to hold the new user registration details

6. Create a new table in your myBookshop database that matches the fields in the register form:

```
CREATE TABLE users (
  id              INT AUTO_INCREMENT,
  firstname       VARCHAR(50),
  lastname        VARCHAR(50),
  email           VARCHAR(50),
  username        VARCHAR(50),
  hashed_password VARCHAR(255),
  PRIMARY KEY(id)
);
```

Note that you will store the hashed password, rather than the original unencrypted password, so we have called the field `hashed_password`.

Task 4: Modify the registered route handler

Now you will modify the `/registered` route handler. It will need to get the data from the form and store it in the new users table that you created. However, before you store the password, you need to encrypt it. You will use the `bcrypt` module to do this.

7. Change your /registered route handler so it contains the following code:

```
router.post("/registered", (req, res, next) => {
    const saltRounds = 10
    const plainPassword = req.body.password

    // Hash the password
    bcrypt.hash(plainPassword, saltRounds, function (err, hashedPassword) {
        // Store hashed password in your database.
        let sqlquery = "INSERT INTO users (username, firstname, lastname,
                       hashed_password) VALUES (?,?,?,?)"

        // Execute sql query
        let newrecord = [req.body.username, req.body.first, req.body.last,
                         hashedPassword]
        db.query(sqlquery, newrecord, (err, result) => {
```

```
            if (err) {
                next(err)
            } else {
                result = 'Hello ' + req.body.first + ' ' + req.body.last
                    + ' you are now registered!  We will send an email to you at '
                    + req.body.email
                res.send(result)
            }
        })
    })
});
```

8. You will also need to require `bcypt`, so add the following at the top of `main.js`, after the `require("express");` line.

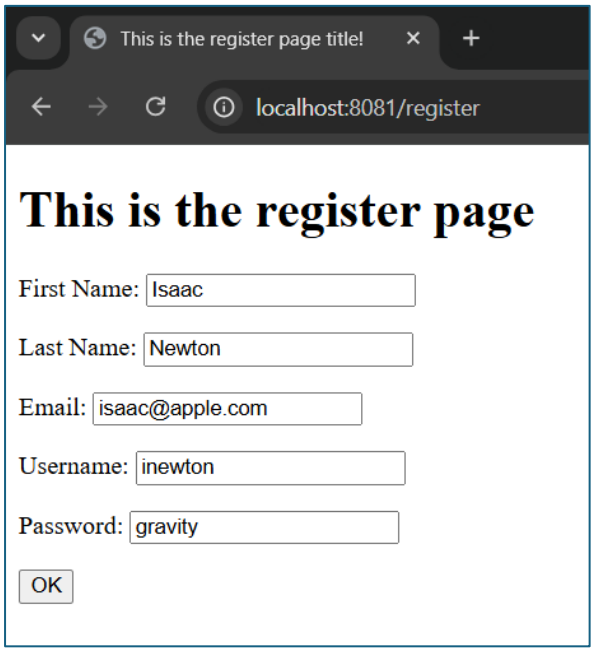
```
// Set up bcrypt
const bcrypt = require('bcrypt')
```

9. You will also need to install the bcrpyt module in your application. Run the following from a terminal session:

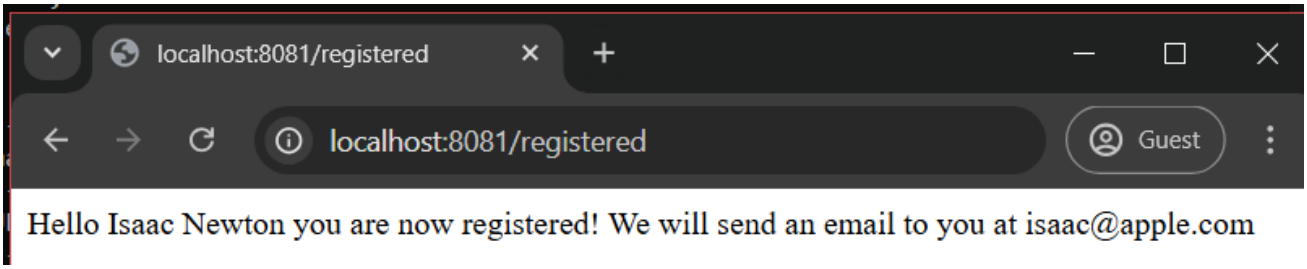
```
npm install bcrypt
```

Task 5: Test the app

10. Run the application and browse to the /register route. Fill in the registration form.



You should see a confirmation that the registration was successful:



11. Check that the user has been added to the database and that the password stored there is securely encrypted. Run the following command in MySQL:

```
SELECT * FROM users;
```

You should see something like this:

```
mysql> SELECT * FROM users;
+----+-----------+----------+-------+----------+--------------------------------------------------------------+
| id | firstname | lastname | email | username | hashed_password                                              |
+----+-----------+----------+-------+----------+--------------------------------------------------------------+
|  1 | Isaac     | Newton   | NULL  | inewton  | $2b$10$frX2A0FU0YwzsMW4ycCgWOrpoMoJMTYjv56aPWMQym/c39ICELAx6 |
+----+-----------+----------+-------+----------+--------------------------------------------------------------+
1 row in set (0.00 sec)
```

Task 6: Explore further

When tackling these lab activities, it's always good to stretch yourself by doing some research and attempting some changes on your own.

Hiding the password

When typing in the password on the register page, the password is visible. Most forms will show "*" for each character entered instead of the actual character typed. Change the type of the password field in `register.ejs` to "password" to achieve this.

Increasing the salt rounds

Try increasing the number of salt rounds, say to 12, 15, 20, 100. What happens?

Securing the database connection

In the lecture, you learned that using the root user to connect to your database poses a significant security risk. Can you change your code to use an application-specific user?

In MySQL, you can create a new database user like this:

```
CREATE USER IF NOT EXISTS 'my_bookshop_app_user'@'localhost' IDENTIFIED BY
'qwertyuiop';
```

You can then give that user access to your bookshop database (and only your bookshop database) using the following:

```
GRANT ALL PRIVILEGES ON myBookshop.* TO 'my_bookshop_app_user'@'localhost';
```

You can then change the connection in your application to use this new user:

```
// Define the database connection
const db = mysql.createConnection ({
    host: 'localhost',
    user: 'my_bookshop_app_user',
    password: 'qwertyuiop',
    database: 'myBookshop'
})
```

End of lab

Congratulations on completing this lab.

In the next lab activity, you will implement session management in your bookshop application.