Backend Documentation

Updated 11/10/2021

This backend service utilizes the Spring framework and will be used by our team to build application features.

Requirements

- Spring
- Maven
- MySQL Community Workbench
- IDE (I recommend IntelliJ)

Optional

- Postman (to test API requests/responses)

Database

This application will use MySQL for data storage.

I recommend developers use a combination of local and cloud databases for creating/testing features. In production, the application will use Amazon Relational Database Service (RDS) for data storage.

Amazon Relational Database Service (RDS)

The Amazon RDS database is currently live and will work with this service.

Amazon AWS Accounts

Instead of using IAM role accounts I think it would be best if developers used one account for the database. The account credentials should be used as environment variables on your computer. I will distribute the credentials later this week.

Your IP address is required so that I can add you to the RDS instance security group.

API Endpoints (*as of 11/10/2021*)

I have created some basic endpoints so that you can test queries to the database hosted on AWS.

I suggest using Postman and MySQL Workbench when making requests to the endpoints so that you can verify the data.

GET endpoints

/home	Returns HTML document
/get-users	Returns JSON array of all users
/get-bugs	Returns JSON array of all bugs

POST endpoints

/create-bug	Inserts a bug into the database

Request Body

title	string
bug_description	string
*due_date	string
assigned_to	string
created_by	string
severity	string
bug_status	string

*due_date requires the following formatting: YYYY-MM-DD

Request example (JSON)

```
{
   "title": "Error on page",
   "bug_description": "There is an error on the page",
   "due_date": "2021-07-11",
   "assigned_to": "1",
   "created_by": "1",
   "severity": "HIGH",
   "bug_status": "OPEN"
}
```

NOTE:
I added a value generator so that the id automatically increments when a user or bug entry is added to the database.

Additional Dependencies
Dependencies are included in the POM document and should download automatically.

Spring Security dependencies are not included in this template but will be added later.

Spring Security – User Authentication
Since password hashing will be done on the client-side, all the backend will do is store the hashed passwords in the database and use Spring Security to verify that the user entered the correct password.

I currently have this set up locally and will push it to the repository after the initial pull request is approved and merged to `main`

Spring Security handles logout automatically when the /logout endpoint is accessed by the user.

Setup
This service is technically ready to go with the only exception being the properties file which houses key/value pairs that Spring uses for application configuration. Without these pairs properly configured, the Spring application will not run.

Since this project is viewable by the public on Github, I decided to remove the values from the configuration file. I will provide configuration details to the team later this week.

Location of configuration file within the file hierarchy:

```
src → main → resources → application.properties
```

More information on application properties can be found here:
https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html

Getting Files From The Repository
Once the backend is approved it will be merged to the main branch on Github. From there you will be able to clone the repository to your local machine.

*****IMPORTANT*****
After cloning, please make sure that that you create a separate working branch for yourself. Only approved work should be merged to the main branch.