



CSCI 5308 – Advanced Topics in Software Development

Group 18 – Budget Surfing

Deployment Document

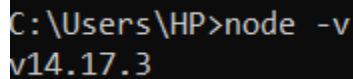
Work done by,

- 1. Parth Dhirubhai Gondaliya, B00893694**
- 2. GRM Vishnu, B00871849**
- 3. Hardee Rameshchandra Garala, B00869195**
- 4. Tuan Hamid, B00877065**
- 5. Neelansh Gulati, B00892946**

Frontend Deployment:

Prerequisite

1. Ensure that the machine has Node installed. It can be checked using the command `node -v` in command prompt.



```
C:\Users\HP>node -v
v14.17.3
```

Figure 1 Check NodeJS installation

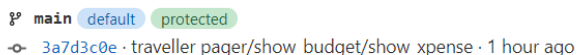
If no version is found, Node will need to be installed from

<https://nodejs.org/en/download/>

2. Check Git is installed. Use the command `git --version`. If git is not found, it will have to be installed on the machine <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Codebase

Repository link: https://git.cs.dal.ca/pgondaliya/budget_surfing_frontend



main default protected
3a7d3c0e · traveller pager/show_budget/show_xpense · 1 hour ago

Figure 2 Frontend branches

Download or clone the codebase from the “main” branch

`git clone https://git.cs.dal.ca/pgondaliya/budget_surfing_frontend.git`

Package Installation

Go into the cloned folder and execute the command *npm i* in command prompt or terminal. This will install all the packages needed to start the project.

```
PS E:\frontend\budget_surfing_frontend> npm i
npm WARN fork-ts-checker-webpack-plugin@6.5.0 requires a peer of typescript@>= 2.7 but none is installed. You must install peer dependencies yourself.
npm WARN tsutils@3.21.0 requires a peer of typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0-dev || >= 3.4.0-dev || >= 3.5.0-dev || >= 3.6.0-dev || >= 3.6.0-beta || >= 3.6.0-beta.0 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

audited 1468 packages in 47.663s

179 packages are looking for funding
  run `npm fund` for details

found 15 vulnerabilities (2 moderate, 2 high, 11 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
```

Figure 3 Successful installation of packages

Starting React Server

Execute the command *npm start* in command prompt or terminal. The server will be deployed on port 3000.

```
Compiled successfully!

You can now view frontend in the browser.

  http://localhost:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

assets by path static/js/*.js 2.2 MiB
  asset static/js/bundle.js 2.2 MiB [emitted] (name: main) 1 related asset
  asset static/js/node_modules_web-vitals_dist_web-vitals_js.chunk.js 6.92 KiB [emitted] 1 related asset
asset index.html 1.95 KiB [emitted]
asset asset-manifest.json 458 bytes [emitted]
cached modules 18.1 MiB (javascript) 31.3 KiB (runtime) [cached] 243 modules
webpack 5.69.0 compiled successfully in 27546 ms
```

Figure 4 React server start

Backend Deployment:

Prerequisite

1. Ensure that the machine has java installed. Java 8+ is required. It can be checked using the command `java -version` in command prompt. If no version is found, java will need to be installed from <https://adoptopenjdk.net/>

```
C:\Users\HP>java -version
openjdk version "11.0.13" 2021-10-19
OpenJDK Runtime Environment Temurin-11.0.13+8 (build 11.0.13+8)
OpenJDK 64-Bit Server VM Temurin-11.0.13+8 (build 11.0.13+8, mixed mode)
```

Figure 5 Check java installed

2. Check that maven is installed in the machine. It can be checked using the command `mvn -v` in command prompt. If no version is found, maven will need to be installed from <https://maven.apache.org/install.html>

```
C:\Users\HP>mvn -v
Apache Maven 3.8.4 (9b656c72d54e5baced989b64718c159fe39b537)
Maven home: C:\apache-maven-3.8.4
Java version: 11.0.13, vendor: Eclipse Adoptium, runtime: C:\Program Files\Eclipse Adoptium\jdk-11.0.13.8-hotspot
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Figure 6 Check maven installed

3. Check Git is installed. Use the command `git --version`. If git is not found, it will have to be installed on the machine <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Codebase

Repository link: <https://git.cs.dal.ca/courses/2022-winter/csci-5308/group18>

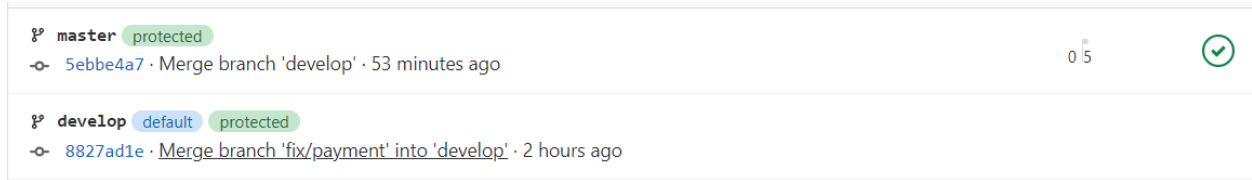


Figure 7 Main branches

Download or clone the codebase from the “master” branch

git clone https://git.cs.dal.ca/courses/2022-winter/csci-5308/group18.git

Setup Database:

Database Configuration with Spring

Update database property inside *application.properties* located inside resources to required database if needed (default Dalhousie remote server).

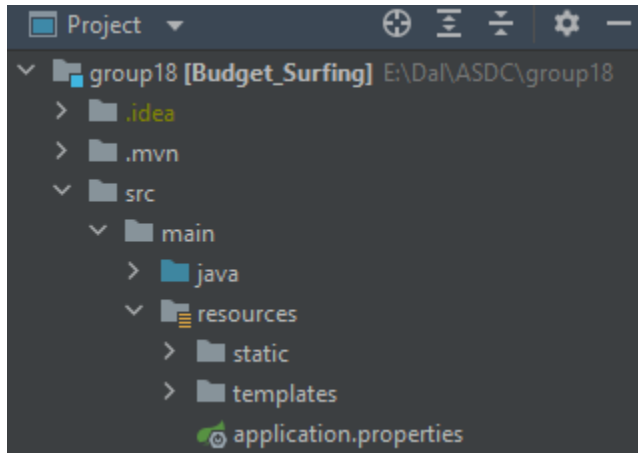


Figure 8 Properties file location

```
spring.datasource.url=jdbc:mysql://db-5308.cs.dal.ca:3306/CSCI5308_18_DEVINT?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
spring.datasource.username=CSCI5308_18_DEVINT_USER
spring.datasource.password=Hugoodaehe1800bo
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
spring.jpa.hibernate.ddl-auto=update
```

Figure 9 Dalhousie MySQL server config

For a first start of new database via spring, the following property should be used *spring.jpa.hibernate.ddl-auto=create* . Any subsequent starts should change the property to “update” instead of “create”.

External Dependencies Configuration:

Email Sending

Currently, the project uses “spring-boot-starter-mail” to send emails. This requires a valid smtp configuration to act as the email server. We are using the free SMTP service by Gmail for this purpose. Configuration for settings are stored in application.properties as seen in the following screenshot.

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=gondaliyaparth12345@gmail.com
spring.mail.password=tzowjxmrnzudwxqi
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

Figure 10 Email configuration

Payments (Stripe)

This project uses stripe-java to act as the external payment processor. As not actual payments are carried out at the moment, we have configured to Stripe’s test environment to prevent actual charges occurring. Configurations for the Stripe public key and secret key are stored in application.properties as shown below.

```
STRIPE_PUBLIC_KEY=pk_test_51KgDpZGWjFP75A0d8NsU2RVzqb0GKAVY7hok6faZ0Jw8ldStktP48bbnWT103A1I9U7jFPmvY6IAsDPXzPZN9
STRIPE_SECRET_KEY=sk_test_51KgDpZGWjFP75A0dRSJvfCkeZ46Go1Rhpr0fGNSW1xodoc96y9B4cNepbXDPtV3IMhPk93DPHMQsARAjgDKfs
```

Figure 11 Stripe configuration

Starting Spring Boot Server

Server port will be a default 8080, it can be changed in `application.properties` if required. Once all preceding steps have been completed, the server can be started successfully. Start the server using the command `mvn spring-boot:run` in terminal.

```
2022-04-05 07:47:41.577 INFO 15332 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-04-05 07:47:41.586 INFO 15332 --- [ restartedMain] com.group18.BudgetSurfingApplication : Started BudgetSurfingApplication in 4.719 seconds (JVM running for 5.192)
2022-04-05 07:48:39.782 INFO 15332 --- [nio-8080-exec-5] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-04-05 07:48:39.784 INFO 15332 --- [nio-8080-exec-5] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-04-05 07:48:39.789 INFO 15332 --- [nio-8080-exec-5] o.s.web.servlet.DispatcherServlet : Completed initialization in 4 ms
```

Figure 12 Server running on port 8080

Database Seeding

The following SQL queries need to be run on a fresh database server.

INSERT INTO ROLES VALUES (1,'ROLE_HOST');

INSERT INTO ROLES VALUES (1, 'ROLE_TRAVELLER');

Packaging JAR for Manual Deployment

The following command needs to be run in terminal to create a JAR of the entire project.

mvn clean package

```
INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ Budget_Surfing ---
INFO] Building jar: E:\Dal\ASDC\group18\target\Budget_Surfing-0.0.1-SNAPSHOT.jar
INFO]
INFO] --- spring-boot-maven-plugin:2.6.3:repackage (repackage) @ Budget_Surfing ---
INFO] Replacing main artifact with repackaged archive
INFO] -----
INFO] BUILD SUCCESS
INFO] -----
INFO] Total time: 32.247 s
```

Figure 13 Successfully packaging JAR

The packaged JAR can be run using the command `java -jar target/Budget_Surfing-0.0.1-SNAPSHOT.jar` in terminal

```
E:\Dell\ASDC\group18> java -jar target/Budget_Surfing-0.0.1-SNAPSHOT.jar
```

```
PV /-----C-----VVVV  
COX---|---|---|---|VVVV  
AV ---| | | | | C | ) ) )  
' |---| |---| |---| V---| / / / /  
=====|=====|_/_/_/_  
:: Spring Boot ::      (v2.6.3)
```

```
[2022-04-05 08:09:32.061 INFO 1968 ...] main com.group18.BudgetSurfingApplication  
Budget Surfing-0.0.1-SNAPSHOT.jar started by HP In E:\Dell\ASDC\group18
```

```
: Starting BudgetSurfingApplication v0.0.1-SNAPSHOT using Java 11.0.13 on DESKTOP-RQ267RH with PID 1968 [E:\Dell\ASDC\group18\target\Bu
```

Figure 14 Launching from JAR

Automatic Deployment via Pipeline

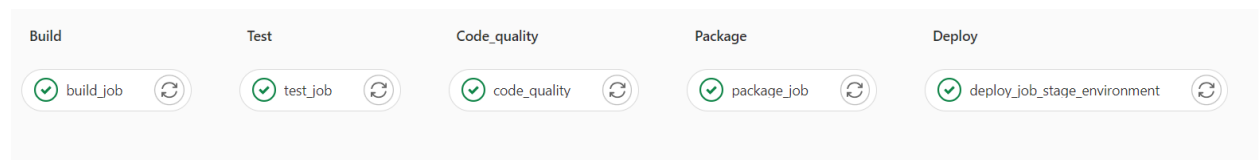


Figure 15 Gitlab pipeline