**DEVELOPMENT MANUAL**

We created a GitHub repository named **/rachitdalal/cs414-f18-001-Sparkans** where it provides access control and with collaborative code review, project management, documentation support and hosting all in one place. This makes our communication within the team simpler in an efficient manner.

The front end/client-side programming is what happens in the browser. It’s everything the users sees and interacts with. The back end/server-side programming, happens on the server and the database. It’s the machinery that works behind the scenes to power those fancy features users interact with on the client side.

**Front End Development:**

* A site is loaded in a browser from the server.
* Client- side scripts run in the browser and process requests without call-backs to the server
* When a call to the database is required JavaScript send requests to the back end.
* The back-end server-side scripts process the requests, pull what they need from the database then send it back.
* Server-Side scripts process the data, then update the site -populating drop down menus, loading products to a page, updating a user profile and more.

Some of the technologies we used in front-end web development include:

HTML5, CSS, JavaScript, Angular2, Json, Nodejs

**Back End Development:**

* Server-side scripts process requests and pull what they need from the database
* Framework are the libraries od server-side programming languages that construct the back-end structure of a site.
* There also exists STACK that comprises the database, server-side framework server and operating system (OS).
* APIs structure how data is exchanged between a database and any software accessing it.

**Servers: Machinery**

- When we type a URL into a browser, its broken down into parts that make it a very specific address. This is translated into an IP address, telling the server exactly where to look for a file.

- For static content, The HTTP server send an HTML file back to the browser, which is read and displayed.

- For dynamic content, a server first executes then returns the file.

**Databases: The Brains**

- Relational DB organizes structured data into columns where as non-relational database stores data in a single document file.

- Any time we request something from a website like searching for a product in an online store or searching locations, the database is responsible for accepting that query, fetching the data, and returning it to the website.

- Databases can also accept new and edited data when users of a website or application interact with them. The client can change information in a database from the browser, whether a user is posting articles to a CMS, uploading photos to a social media profile, or updating their customer information.

**Middleware: The Plumbing**

- Middleware essentially describes any software on the server that connects an application’s front end to its back end, it pipes any communication, like requests and responses, back and forth between your application and your server/database.

- Middleware (server-side software) facilitates client-server connectivity, forming a middle layer between the app(s) and the network: the server, the database, the operating system, and more. - --- Middleware can be multi-layered, organized into different layers of a site, whether it’s the presentation layer or the business layer.

Technologies used in the backend:

Java

**CLIENT:**

1. Install IDE like IntelliJ, eclipse (<https://www.eclipse.org/downloads/>) for the code to run in the tool platforms.

2. Install Node js from (<https://nodejs.org/en/download/>) based on your system version.

3. Download the Git from (<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>) and store it in specified location.

4. Open the IDE as a Maven project and specify the git path under File→settings →version control→ git → Path to git executable and add the git part here.

**CLONING:**

Repository: **rachitdalal/cs414-f18-001-Sparkans**

Copy the URL from the repository under clone or download it.

**IDE: Intelli J**

Under VCS→ check out from version control→Git. Enter the copied URL, test and clone it.

**Terminal:**

Run the following commands:

1. npm install

2. npm install -g @angular/cli

3. “npx ng serve” (for lab machines) or “ng serve” for other machines

BanqiUI

This project was generated with Angular CLI version 8.3.8.

Development server

Run ng serve for a dev server. Navigate to http://localhost:4200/. The app will automatically reload if you change any of the source files.

Code scaffolding

Run ng generate component component-name to generate a new component. You can also use ng generate directive|pipe|service|class|guard|interface|enum|module.

Build

Run ng build to build the project. The build artifacts will be stored in the dist/ directory. Use the --prod flag for a production build.

Running unit tests

Run ng test to execute the unit tests via Karma.

Running end-to-end tests

Run ng e2e to execute the end-to-end tests via Protractor.

Further help

To get more help on the Angular CLI use ng help or go check out the Angular CLI README.

**SERVER:**

**Building and running the server:**

Building

1. To build the server you must have Maven installed (<https://maven.apache.org/install.html>)

2. Run "mvn package" under project root(/cs414-f18-001-Sparkans) will run tests and build the server-\*.jar file.

3. Run "mvn clean" followed by "mvn package" to be sure you have only the latest version of the server built.

Running

1. To run the server use "java -jar server-\*.jar" on the .jar file located in /server/target/

2. If you're in a bash environment the "run" script at the project root will build and run the server.

3. The server runs on localhost:31406