Product Requirements Document

Tempflix WeatherApp –
Road Warrior Weather Tool

Product Requirements Document

Key Stakeholders

Document Status	Draft
Engineering	Rahul Khandalkar
Design	Rahul K, Ray K, Julie C
Product	Rahul K, Ray Kwong
Marketing / TME	Rahul K
Supportability / Telemetry	Rahul K

1. Objective: **[Mock-up objective/Assumption]**

Vision	Most accurate live temperature of hottest state capitals
Goals	Timeframe: 1 week – MVP Primary goal is simplifying the operations and management function that a website administrator has to perform for API integration of weather tool. Secondary goal of this API is for customer to adopt spring framework solution and usage of library for OpenWeather API in an existing environment. Spring use JSONParser libraries to extract the data from json file. RESTTemplate to access the Open Weather API. Storing data convenient in H2 in-memory database.
Initiatives	Incorporated organization stylings and wireframes
Persona(s)	Weather Forecast Support Engineer, Technical Marketing Engineer, Sales

2. Release

Release	weatherapp-0.0.1
Date	May 26, 2021
Initiative	Tempflix WeatherApp 1
Milestones	1. Minimum Valuable Product (MVP) 2. Scalable solution
Features	1. Live streaming of temperature of hottest US State capital 2. Deployable solution as web application and API interface
Dependencies	1. Maven 3.8.1 2. RedHat Open JDK 1.8 3. Supports all OS. Preferred Windows.

3. Features

1. **[Mock-up Story/Assumption]**

Feature	Live stream of temperature of hottest US state capitals
Description	The feature will enable the product support engineer and door to door marketeer to provide product feature in Tempflix tablet and satellite internet. The live stream data will be updated on tablet from any location to show API data.
Purpose	One-click application to show live weather stream
User problem	The challenge to open browser for purpose of this feature is tedious
User value	One-click application provide easy access and better satisfaction
Assumptions	The end user have knowledge on US State Capitals, internet usage and OpenWeather API will provide required output
Not doing	None at the moment
Acceptance criteria	OS update version above x.x

2.

Feature	Deployable solution as web application and API interface
Description	The solution can be deployed as client-server architecture. The code works as deployable solution on server. The spring framework capability to provide API interface can provide further development of MVP for frontend Angular.js to interface with middle layer Java application and H2 database
Purpose	Scalable and readable solution of weatherapp-0.0.1
User problem	Existing solution did not follow best practices of SLDC. The developer left the company and provided no documentation to cause frequent hotfixes. Thus, user did not receive reliable update on their earlier version of Tempflix.
User value	A properly followed SLDC best practice will provide product reliability
Assumptions	The end user has cleanly uninstalled the previous version of the application. The newly installed version is from TempFlix official repository.
Not doing	The web application is not developed to have Angular.js framework. The MVP is limited to running application 15 times in a day because free version of Open Weather OneCall API allows 1000 request per day.
Acceptance criteria	1. Windows 10 (preferred) 2. RedHat Open JDK 1.8 - jdk-8u292-x64 (Login Required) https://developers.redhat.com/content-gateway/file/java-1.8.0-openjdk-1.8.0.292-2.b10.dev.redhat.windows.x86_64.msi 3. Maven 3.8.1 - apache-maven-3.8.1-bin.zip https://maven.apache.org/download.cgi https://mirrors.sonic.net/apache/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.zip

4. Implementation

Limitation: Wireflow/Front End not implemented yet. Only Model, Controller implemented. View feature is limited to CLI operations.

The application is broken in four layers:

1. Controller - External request are received by Controller and provided to Service layer.
2. Service - Logic implementation. Example: if-else, avg calculation, etc are provided in this layer
3. Repository - Data storage to in-memory H2 database and retrieving from repository.
4. Model - interfaces with all the above layer, provides medium to implement future getter/setter functionality.

User Flow: (-> arrow denotes call function)

Start application main() -> WeatherController.getTopSevenHottestCities()
WeatherController -> Parse resources-us-state-capitals.json and store in
WeatherController -> WeatherController.parseCapitol()
WeatherController.parseCapitol() stores in List usStateCapitalList
WeatherController.parseCapitol() -> model-USStateCapital
model-USStateCapital -> model-City
model-City.setName(),model-City.getName()
model-USStateCapital

WeatherController -> Service-WeatherService
Service-WeatherService -> Service-WeatherServiceImpl
Service-WeatherServiceImpl.deleteFromTable()
WeatherController

WeatherController -> Service-WeatherService (Interface)
Service-WeatherService -> Service-WeatherServiceImpl populate data from github JSON
Service-WeatherServiceImpl.saveAvgTemperatureByLatLong() saves avg of 7daytime temperature of all US capitals
Service-WeatherServiceImpl.saveAvgTemperatureByLatLong() -> repo-weatherRepository.storeAvgTempOfCity
repo-weatherRepository.storeAvgTempOfCity -> repo-TopTenHottestCityMapper()
WeatherController

WeatherController -> Service-WeatherService
Service-WeatherService -> model-TopTenHottestCity
model-TopTenHottestCity.TopTenHottestCity() retreives city and avg temp of top 10 hottest US state capitals
WeatherController.printTopTenHottestCity() print the previous returned value.

5. Pre-requisite, Working and Test

Download:

Windows 10 (preferred):

1. RedHat Open JDK 1.8 - jdk-8u292-x64 (Login Required)

https://developers.redhat.com/content-gateway/file/java-1.8.0-openjdk-1.8.0.292-2.b10.dev.redhat.windows.x86_64.msi

2. Maven 3.8.1 - apache-maven-3.8.1-bin.zip

https://maven.apache.org/download.cgi

https://mirrors.sonic.net/apache/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.zip

Mac:

1. Java JDK16

https://docs.oracle.com/en/java/javase/15/install/installation-jdk-macos.html

https://www.oracle.com/java/technologies/javase-jdk16-downloads.html

2. Maven (Same as Windows)

https://maven.apache.org/install.html

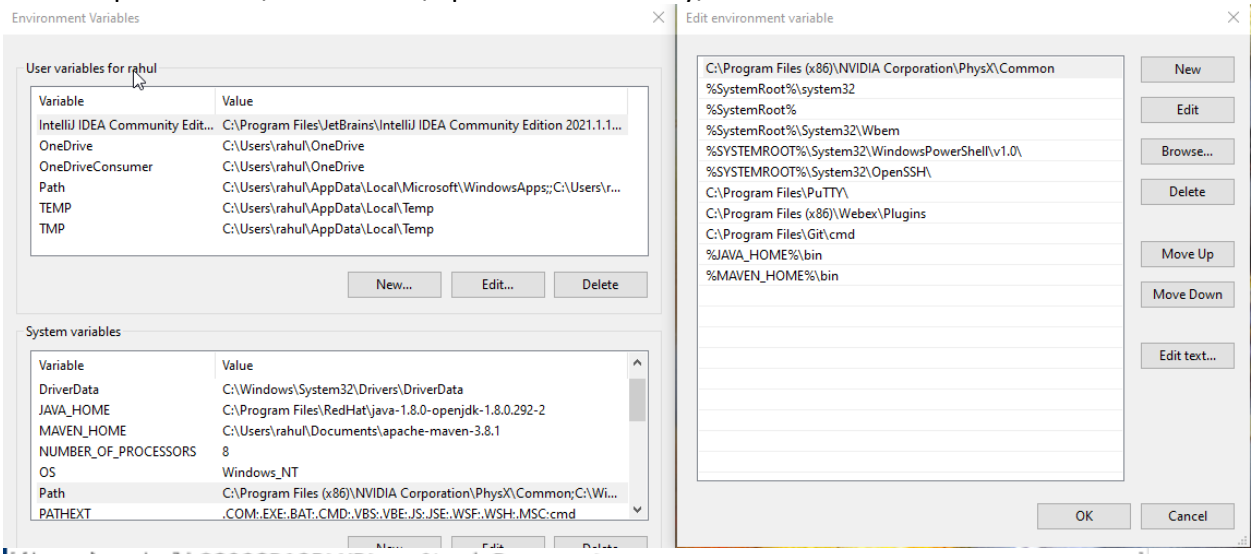
Ensure System has internet connectivity to download any required Java library dependency

Install instruction:

1. Install OpenJDK and Maven, ensure path environment variables to correct Java and Maven installation.

   Windows - %JAVA_HOME%/bin and %MAVEN_HOME%/bin

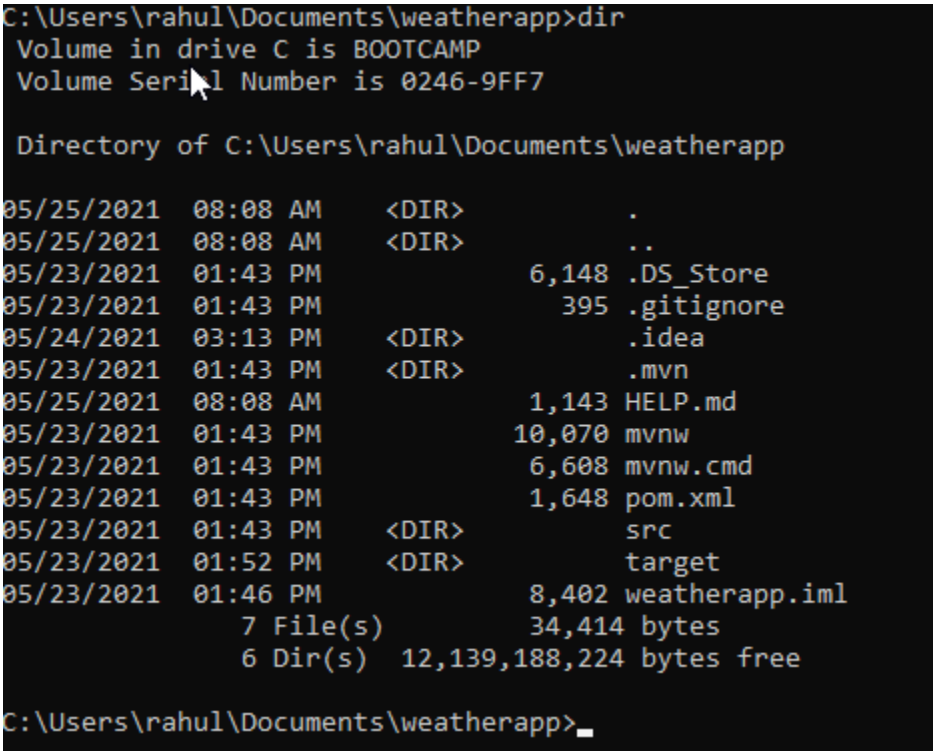
   Mac - export PATH=/Documents/apache-maven-3.x.y/bin:$PATH




```
(base) rahulk@C02CD13BLVDL ~ % cd Documents
(base) rahulk@C02CD13BLVDL Documents % cd apache-maven-3.8.1
(base) rahulk@C02CD13BLVDL apache-maven-3.8.1 % cd bin
(base) rahulk@C02CD13BLVDL bin % pwd
/Users/rahulk/Documents/apache-maven-3.8.1/bin
(base) rahulk@C02CD13BLVDL bin % export PATH=/Users/rahulk/Documents/apache-maven-3.8.1/bin:$PATH
```

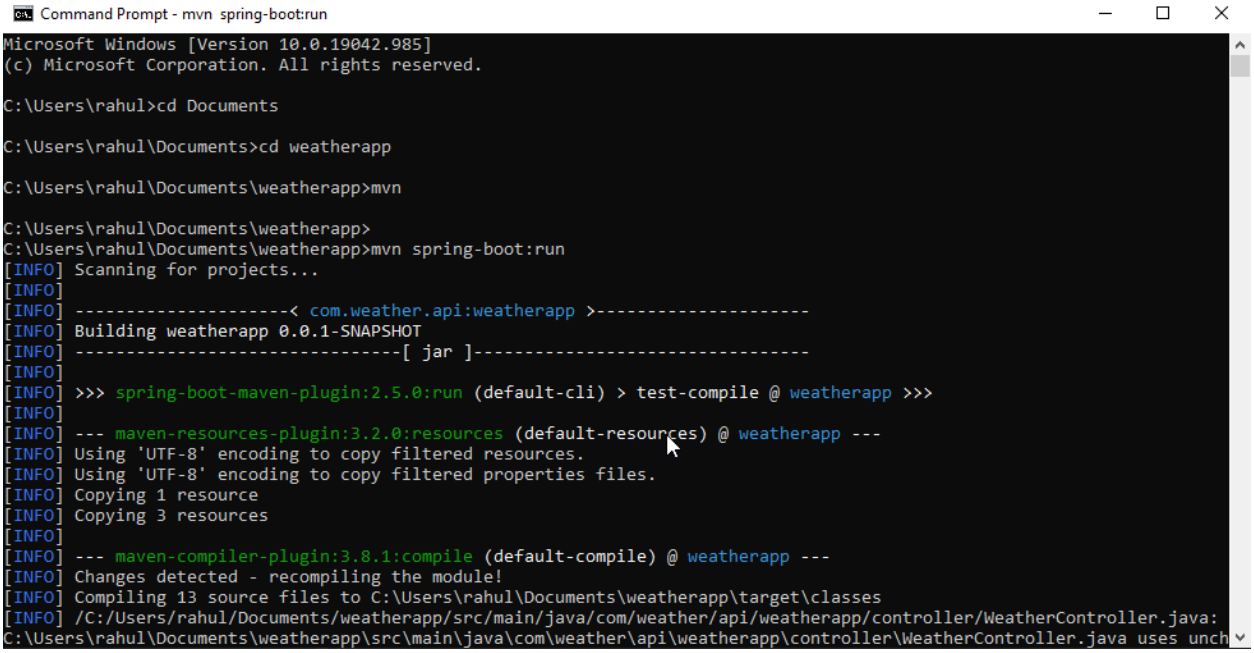
2. Unzip the application in Documents

3. Launch command prompt
   Change directory to unzip file, ensure pom.xml reside

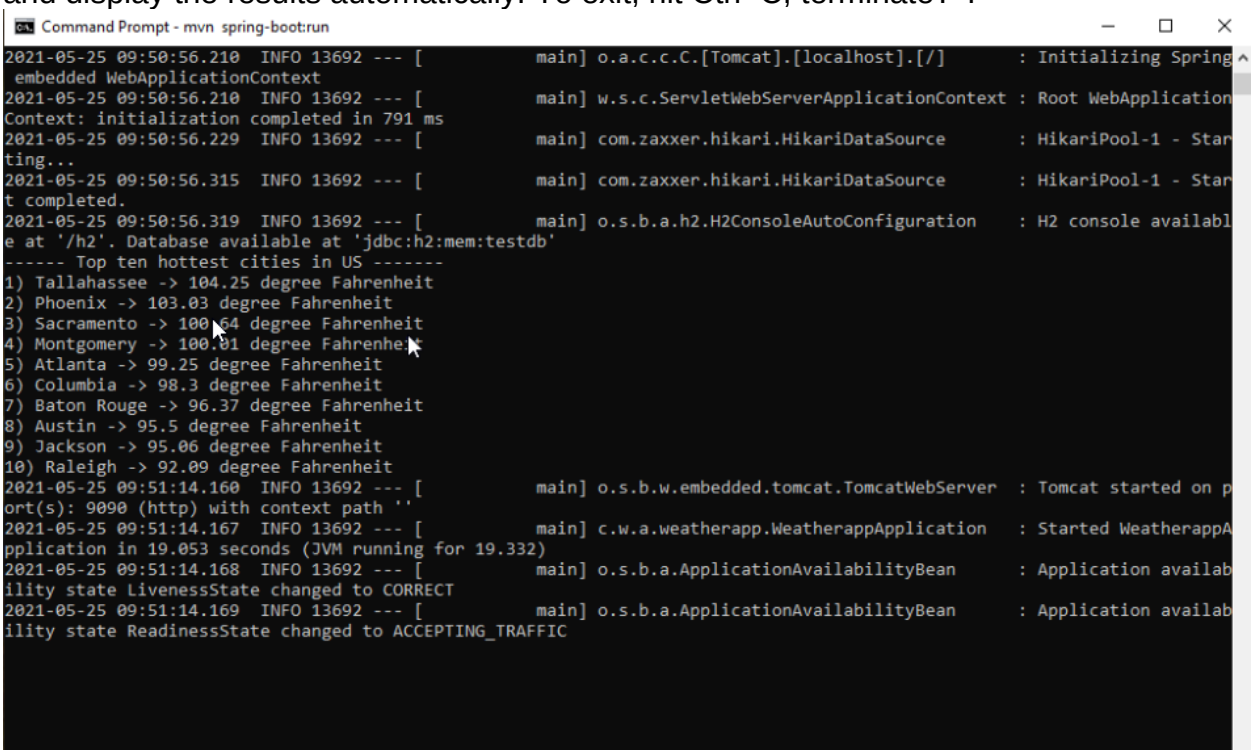


The app is ready to run. Apple Mac experience should be similar.

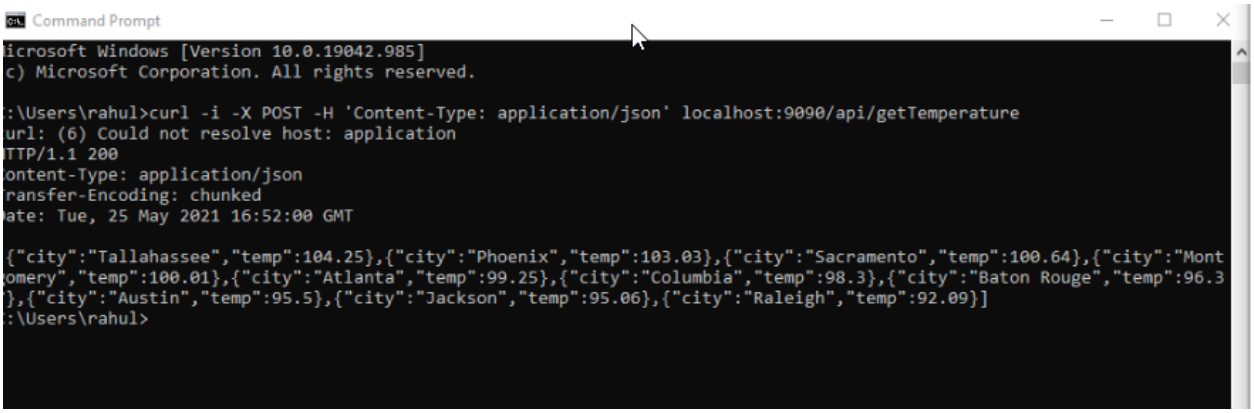
4. Run the application from this directory, command: mvn spring-boot:run
   First time running the application can take 5 minutes to download dependency



5. The application is now running. It should take around 20-30 seconds to complete and display the results automatically. To exit, hit Ctrl+C, terminate? Y



6. To re-run the application from #5, please open another command prompt window and run command to send POST request to our internal web server.
   curl -i -X POST -H 'Content-Type: application/json' localhost:9090/api/getTemperature
   It can take another 20 seconds to show results



6. Analytics

Hypothesis: We believe web application will achieve to provide output in 20 seconds for version weatherapp-0.0.1 and not limited to amount of OpenWeather API Call

As we are retrieving, computing, storing database and then print top 10 hottest US state capitals in one call. Thus, it takes time about 20 seconds.

Key performance indicator	Baseline	Target	Timeframe
Retrieval, storing database and then print top 10 hottest US state capitals	21	20 s	-

7. Future work

Future features	Purpose	Priority	Timeframe
Separate API call interface for each retrieval, storing database and then print top 10 hottest US state capitals	Better management for future implementation to separate functionality	P3	1 week
Frontend angular.js solution	End-user comfort, lesser dependency on CLI functionality, one click interface	P1	3 weeks