# DEPENDIFY: THE DYNAMIC DEPENDENCY HUB

by Sidhant Bansal

CSS 533: Distributed Computing

Professor Robert Palmer

University of Washington Bothell

Bothell, WA

05/18/2022

Table of Contents

## I. Introduction

The idea of this project is that it will be an online repository from which people can download and upload different types of dependencies as per their requirements. It will behave like a marketplace for finding all the available dependencies out there and uploading one if it is not there.

The main components would be downloading an available dependency and uploading a dependency to the central repository. A user can also search the central repository to find the specific dependency.

The application would be hosted on an AWS server which will provide me with steady infrastructure that will help my development, and also provide me public IP address so that it is accessible to my clients from anywhere. The front end would be hosted on an AWS server, application logic would be hosted on 2 different AWS servers with an application load balancer and the persistence tier would be on a separate AWS server to keep my application fail-safe.

The proposed system's goal is to create a centralized repository where users can find or upload the latest versions of dependencies that are essential for a particular part of the code to work. For the user, it will make it easier to search in a centralized space instead of going through different links on the internet which can be time-consuming and misleading sometimes.

## II. Assumptions

I am going to assume that the user is aware of the dependency that they require when they are searching for it on the centralized system.

## III. Components

### *Search*

The interface of this component will allow the user to search for a dependency from the User Interface. The search button on the user interface will interact with the Java framework that in turn will communicate with the database to check the availability of the searched dependency. The dependency files would be stored on the Amazon S3 server and the database will have a path to those files. The possible usage and error conditions would be the following:

1. If the dependency is present in the database then it will be displayed to the user on the user interface with the details like “Group Id”, “Artifact Id”, “Latest Version”, “Last Update Date”, and a button to “Download” the same.
2. If the dependency is not present in the database then the user would be displayed a screen with a text saying “No results found”.

### *Upload*

The interface of this component will provide an option for the user to upload a dependency. The upload button on the user interface will interact with the Java framework which will interact with the database to make entries in the database and upload the file to the Amazon S3 server. While uploading a file the user would be required to fill out the “Group id”, “Artifact id”, and “Version number” of the dependency. It will have several possible usage and error conditions:

1. All the above three fields would be required and the user would not be able to upload the dependency without entering valid entries in those fields.
2. If the dependency is not there in the database then a new entry would be added to the database and server.

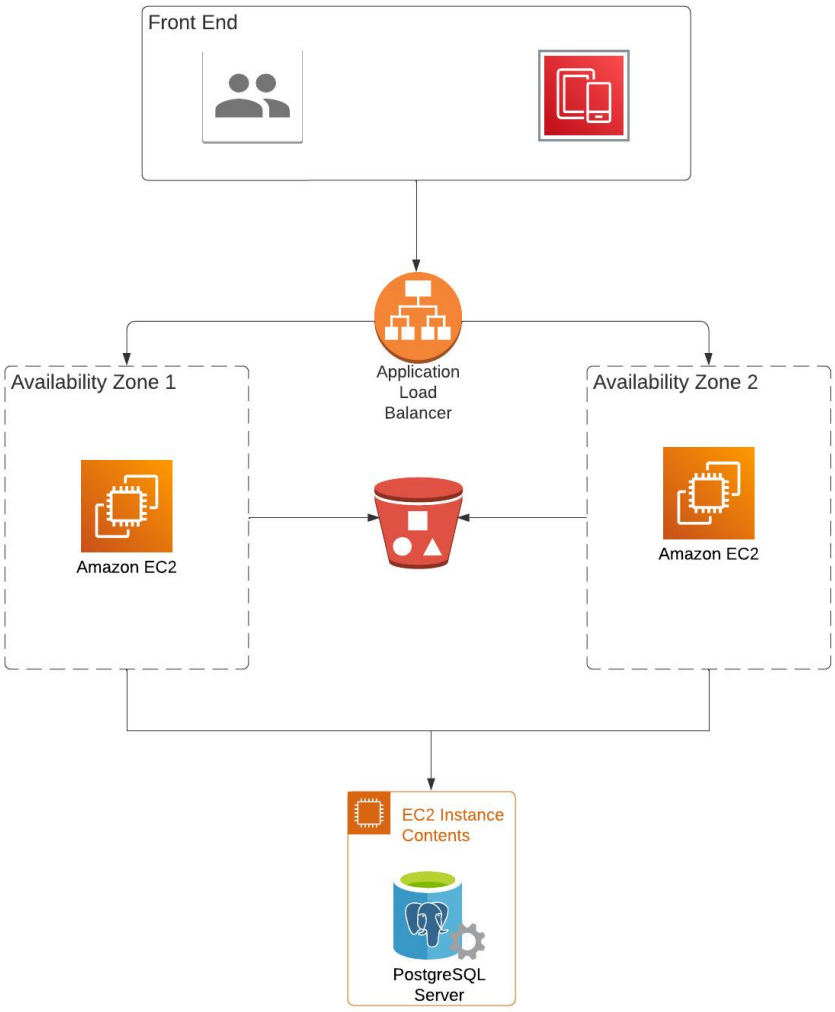
3. If the dependency is already present in the database and the version is the same as the one already present then no changes would be made.
4. If the dependency is already present but the version is different then both the versions of that dependency would be maintained in the database and the server. The next user searching for that dependency will have the option to choose from all the versions.

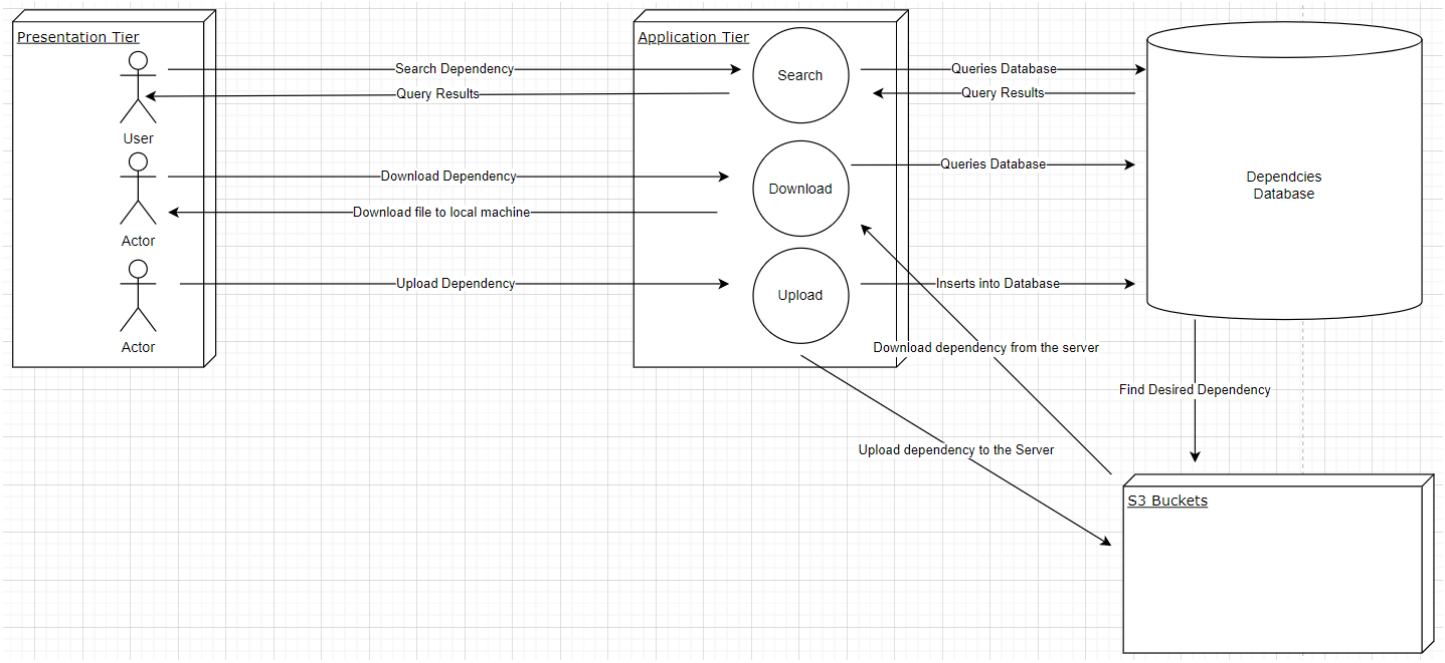
*Download*

The interface of this component will allow the user to download the desired version of the dependency. Once the user clicks on download then the java service will query the database to download the desired dependency from the Amazon S3 server. It can have these usage and error conditions:

1. The download would be successful.
2. If the download is unsuccessful because of an interruption in the internet connection or for any other reason, then the user would be notified that the download was unsuccessful.

## IV. Architecture Diagrams






## V. Languages and Frameworks

### *Presentation Tier*

The User Interface was developed using the Angular framework which would basically include HTML, CSS for designing web pages, and Javascript for functionalities.

### *Application Tier*

The middle tier was developed using the Java language. The framework used here was Spring.

### *Data Tier*

PostgreSQL was used for the data tier since it is an object-relational database.

## VI. Database Design

The operations required on the database were:

1. SELECT for fetching details of an existing dependency during the search and download operation.

2. INSERT for adding a new dependency during the upload operation.
3. UPDATE for updating the version of a dependency during the upload operation.

My planned system had a single table only for managing dependencies information and the schema of the same was:

| | |
|---|---|
| UNIQUE_ID | INTEGER PK |
| GROUP_ID | VARCHAR |
| ARTIFACT_ID | VARCHAR |
| VERSION | DECIMAL |
| LAST_UPDATE | DATE |
| RETRIEVAL_KEYWORD | VARCHAR |

- The UNIQUE_ID was used as a primary key.
- The GROUP_ID was the id of the organization or group that created/uploaded the dependency.
- The ARTIFACT_ID was the name of the jar file without the version number.
- The VERSION was the version number of the specific dependency
- The LAST_UPDATE was the date on which the dependency was last updated on the central repository.
- The RETRIEVAL_KEYWORD was the keyword that was used for the search operation and to retrieve the file from the S3 bucket.

## VII. Testing

- Unit Tests were written to determine if each component (Search, Upload, and Download) is working as expected.
- Integration Tests to determine if each layer/tier is communicating with the other layer/tier as expected.
- System Tests/Human Testing to determine if the application as a whole is working as desired.