Assignment-4

Session: 2021 – 2025

Submitted by:

Bisma Muhammad Ali

2021-CS-170

Supervised by:

Mr. Atif

Department of Computer Science

University of Engineering and Technology Lahore

Pakistan

Contents

1 Introduction

NPM (Node Package Manager) is a command-line tool for managing packages and dependencies in Node.js projects. It simplifies the process of installing, updating, and removing packages, as well as managing versioning and dependency resolution. NPM works by connecting to the npm registry, where packages are stored, allowing developers to easily download and integrate third-party code into their projects.
NPM (Node Package Manager) serves as a central hub for accessing, sharing, and distributing JavaScript packages. It operates by leveraging a vast repository known as the npm registry, where developers publish packages for public or private use. NPM facilitates package installation, enabling developers to specify dependencies within their projects via a package.json file.

2 Installation Process

The installation process of Node.js and NPM is straightforward. Downloade the installer from the official Node.js website and followed the installation instructions for operating system. After installation, verified the installation by running the following commands in the terminal:

```
node -v
npm -v
```

3 Background Research

3.1 Purpose and Significance of NPM

NPM plays a vital role in contemporary web development by simplifying the management of project dependencies, version control, and package publishing. It streamlines the process of incorporating external libraries and tools into projects, thereby enhancing the efficiency of development workflows.

3.2 Key Features of NPM

1. **Dependency Management**: NPM allows developers to define project dependencies in a package.json file and automatically install the required packages. It resolves dependencies recursively and ensures that the correct versions are installed.

2. **Package Installation**: Developers can install packages globally or locally using the npm install command. Global installation makes packages available system-wide, while local installation installs packages within the project directory.

3. **Version Control**: NPM employs semantic versioning to manage package versions effectively. Developers can specify version ranges in the package.json file, allowing for flexibility in upgrading dependencies while maintaining compatibility.

4. **Package Publishing**: NPM enables developers to publish their own packages to the NPM registry, making them accessible to the wider community. Publishing packages involves adhering to guidelines and best practices to ensure quality and usability.

4 Basic Usage

4.1 Initializing a New Node.js Project

Initialized a new Node.js project using the npm init command and followed the prompts to create a package.json file.

4.2 Installing Packages

Commonly used packages such as lodash and axios were installed using the npm install command. Different options and flags like –save and –save-dev were explored during package installation.

4.3 Exploring Options and Flags:

Explore different options and flags available with the npm install command, such as –save, –save-dev, and –global, to manage dependencies and save them to the appropriate sections in package.json.

5 Managing Dependencies

5.1 Manual Management:

Manually added, updated, and removed dependencies in the package.json file to understand how dependency management works.

5.2 Utilizing Commands:

Utilized npm install and npm uninstall commands to manage dependencies effectively, ensuring that dependencies are installed or removed as required.

6 Package Scripts

6.1 Defining Custom Scripts:

I defined custom scripts in the scripts section of the package.json file to automate common tasks such as compiling, testing, and deploying my project.

6.2 Executing Scripts:

Executed the defined scripts using the npm run command and observed the output to verify that the automation tasks were performed correctly.

7 Challenges Faced

1. **Understanding Semantic Versioning:** Initially, I faced challenges in understanding semantic versioning and its impact on dependency management. However, after reading the documentation and experimenting with version ranges, I gained clarity on how to specify version requirements effectively.

2. **Network Connectivity**: NPM relies on internet connectivity to download packages from the registry. Poor network conditions or firewall restrictions may hinder package installation and updates.

3. **Slow Installation**: Installing large packages or a multitude of dependencies can be time-consuming, especially on slower network connections or less powerful hardware.

8 Conclusion

This assignment provided valuable hands-on experience with NPM, enabling me to gain proficiency in managing dependencies, automating tasks, and enhancing web development workflows. Through experimentation, exploration of NPM packages, and engagement with the community, I deepened my understanding of NPM fundamentals and advanced features. Overall, this assignment has significantly contributed to my skills in web development tooling and automation.

9 Additional Resources

- Node.js Official Website
- NPM Documentation
- NPM Registry