# Internship Report

**Name: Sanjit Kumar**

**Internship Period: May 2025**

**Company Name and Location: Cognizant Technology Solutions, Chennai**

**Submitted to: Lovely Professional University, Phagwara**

## 1. Introduction

Software development relies on version control systems to track changes, collaborate, and manage multiple versions of code efficiently. Git, a widely used Distributed Version Control System (DVCS), enhances project management by allowing seamless branching, merging, and collaboration.

Similarly, Angular is a powerful front-end framework for building dynamic web applications, while Node.js provides a runtime environment for executing JavaScript outside the browser. This report explores these technologies, covering their essential concepts, applications, and best practices.

## 2. Version Control and Git

### 2.1 Definition and Purpose

Version control systems (VCS) track modifications in source code and allow developers to revert to previous versions when needed. They facilitate team collaboration and ensure smooth development workflows.

### 2.2 Benefits of Version Control

- Maintains historical versions of files.
- Enables efficient team collaboration.
- Prevents accidental loss of work.
- Simplifies debugging and rollback.

### 2.3 Types of Version Control Systems

- Centralized Version Control Systems (CVCS): Uses a single central repository.
- Distributed Version Control Systems (DVCS): Each user has a complete copy of the repository.

### 2.4 Understanding Git

**Git is a DVCS widely used for source code management. It allows multiple developers to work on projects simultaneously without being dependent on a central server.**

**Git Components**

- **Working Directory: The area where files are edited.**
- **Staging Area: Prepares changes before committing.**
- **Repository: Stores committed versions of files.**

## 2.5 Setting Up Git

**Installing Git**

1. **Download Git from git-scm.com.**
2. **Follow installation steps based on the operating system.**
3. **Configure Git with:**

## 2.6 Basic Git Commands

- **git add <file> - Stages changes.**
- **git commit -m "message" - Saves changes.**
- **git status - Displays current repository status.**
- **git log - Shows commit history.**

## 2.7 Branching and Merging

**Branching helps developers work on different features simultaneously.**

- **Creating a Branch**

## 2.8 Remote Repositories and Collaboration

- **Adding a Remote Repository**
- **Pushing Changes**

**3. Introduction to Angular**

**3.1 Overview**

**Angular is a front-end framework developed by Google to create dynamic, scalable web applications.**

**3.2 Setting Up the Angular Environment**

- **Install Angular CLI:**

**3.3 TypeScript Essentials for Angular**

**Angular uses TypeScript for enhanced features such as:**

- **Interfaces and type aliases.**
- **Classes and object-oriented programming.**
- **Asynchronous programming using Promises and RxJS Observables.**

**3.4 Angular Components**

**Angular applications are built using components.**

- **Component Lifecycle Hooks: ngOnInit, ngOnChanges**
- **Data Binding: Property binding, event binding, two-way binding (ngModel).**

**3.5 Directives and Pipes**

- **Built-in Directives: *ngIf, *ngFor, *ngSwitch, ngClass, ngStyle**
- **Custom Directives: Extend Angular's functionality.**
- **Pipes: Used for data transformation (e.g., date, uppercase).**

**3.6 Forms in Angular**

**Forms handle user input efficiently.**

- **Template-driven forms: Uses ngModel.**
- **Reactive forms: Uses FormGroup, FormControl.**

**3.7 Angular Routing**

**Routing allows navigation between components.**

- **Defining Routes: Set up in app-routing.module.ts.**
- **Lazy Loading: Optimizes performance.**

**3.8 HTTP Client and APIs**

**Using HttpClientModule for API interaction:**

- **GET, POST, PUT, DELETE requests.**
- **Error Handling using catchError and retry.**

**3.9 State Management in Angular**

**State management ensures data consistency.**

- **Using Services for local state.**
- **NgRx for advanced state management.**

**4. Introduction to Node.js**

**4.1 Overview**

**Node.js is a JavaScript runtime designed for server-side applications with non-blocking asynchronous operations.**

**4.2 Node.js Architecture**

- **Single-threaded, event-driven model.**
- **Non-blocking I/O for high scalability.**

**4.3 Node.js Module System**

- **Using require() to import modules.**
- **Exporting and importing functions across files.**

**4.4 Core Node.js Modules**

- File System (fs) for reading and writing files.
- HTTP Module for creating and handling web servers.

## 4.5 Asynchronous Programming

Node.js supports callbacks, promises, and async/await for efficient execution.

## 4.6 Debugging Angular Applications

- Chrome DevTools for real-time inspection.
- Visual Studio Code Debugger for breakpoints and watches.

## 5. Conclusion

Version control, Git, Angular, and Node.js are essential tools for modern web development. Git ensures efficient collaboration, Angular provides a robust framework for building dynamic web applications, and Node.js enables scalable backend solutions. Mastering these technologies improves development workflows and ensures efficient project management.