cjXJFAzurrAngular 8+, Typescript

  

 **C Monish**

**Sr. Java Full Stack Developer**

**Email:** monishchintha17@gmail.com

**Phone: 913-348-0450**



**Chintha Monishkumar Reddy**

**Java Full Stack Developer**

**ABOUT ME:**

**Overall, 11+** *years of experience in Full Software Development Life Cycle (****SDLC****) which includes* ***Analysis****,* ***Design****,* ***Development****, Implementation, and Testing of web-based applications using* ***Java****/****JEE*** *technologies. Expertise in core* ***Java*** *concepts like* ***OOPs*** *concepts* ***Collections****,* ***Exception******Handling****, and* ***Multithreading****.*

**Java 17 | Angular | MySQL | JavaScript| Node.js | Spring**

**PROFESSIONAL SUMMARY**

* Highly skilled and results-driven **Full Stack Java Developer** with **11+** years of expertise in **OOPs, Java/J2EE technologies, React, Angular, NodeJS,** and all phases of the **Software Development Life Cycle (SDLC**), including design, development, testing, and deployment of web-applications.
* Proficient in **Agile** and **Waterfall** methodologies, ensuring alignment between business requirements and technical implementations. Implemented security features using **LDAP**, **OAuth 2.0, SSL, AWS IAM**, and **JWT** with Spring Security.
* Experienced in **MEAN Stack** Development, including **MongoDB**, **Express**, **Node.js**, and **Angular,** for creating SPAs.
* Skilled in React components, **Redux architecture**, and **Flux concept**, enhancing website functionality.
* Strong grasp of Core Java concepts and **data structures**, with expertise in messaging modules using **Spring JMS**, **Apache Camel**, and **RabbitMQ**. Managed code repositories using **GIT and Bitbucket.**
* Proficient in **Spring framework** (MVC, IOC, ORM, JDBC, AOP, BATCH, WEBFLOW) and **Microservices** development with **Spring Boot.** Experienced in **J2EE** Testing with **JUnit, MOCKITO**, and logging with Log4j and Splunk.
* Extensive experience in Core **Java, J2EE, JSP, Servlets**, **JDBC, Spring**, MuleSoft ESB, Maven, Hibernate, Web Services, and **SOAP**. Established **CI/CD** pipeline using **Jenkins with GitHub** and **Docker** container services.
* Skilled in Amazon Web Services (**AWS**), including **S3, EC2, SQS, DynamoDB, Lambda**, and Elastic Load Balancing.
* Implemented serverless architecture using **AWS services** such as **API Gateway, Lambda**, and **DynamoDB**.
* Proficient in Integrated Development Environments (IDE)s such as NetBeans, Eclipse, and SQL Developer.
* Experienced in building and deploying **EAR, WAR, and JAR files** on various servers, including **Apache Tomcat**.
* Utilized **WebSphere and JBoss** for implementing business applications.
* **Education Details:**

I completed my bachelor’s from Karunya Institute of technology and Sciences from Coimbatore in the year 2012.

**TECHNICAL SKILLS**

|  |  |
| --- | --- |
| **Languages/Utilities** | Java 17, C, C++, Python, PL/SQL, XML. |
| **JAVA Technologies** | Core Java, Multithreading, JDBC, JNDI, Java beans, J2EE, Junit, Maven,Mockito, Ant, Log4j. |
| **Frame Works** | Spring, Spring Boot, Spring Batch Struts, Kafka, JSF and Hibernate, Sw.  |
| **Web Services** | WSDL, SOAP & REST. |
| **Web Technologies** | Applets, JSP, HTML, CSS3, JavaScript, JSON, AJAX, jQuery, Angular, React, Typescript. |
| **Databases** | MYSQL, Oracle, SQL server, MongoDB, Couchbase, PostgreSQL. |
| **Middleware** | Servlets, Java Beans, RMI, EJB, JMS. |
| **IDEs and Tools** | Eclipse, NetBeans, Text pad, Maven, UML, Log4j, ANT. |
| **Version Control Tools** | Subversion, GitHub, CVS. |
| **Web/Application Servers** | Tomcat, IBM WebSphere, JBoss, Apache. |
| **Methodologies** | Waterfall, Agile. |
| **Operating Systems** | Windows 7/8/XP, Linux, UNIX. |
| **Cloud Technologies** | AWS Cloud Components S3, EC2, Lambda, Elastic Bean Stalk, ECS, RDS. |

**PREFESSIONAL EXPERIENCE**

**Rockefeller Capital, NY July 2021 to Till Date**

**Full Stack Java Developer**

**Responsibilities:**

* Developed frontend components using **Angular CLI** and **TypeScript,** creating UI components for web pages.
* Implemented **Angular directives**, expressions, and routing capabilities for bookmarking, filtering, and validation, achieved optimized performance with **Angular Lazy** Loading technique.
* Leveraged **AWS Lambda** for serverless computing, enabling code execution **without server management**.
* Using **OAuth 2.0, OIDC** (OpenID Connect) makes user identity and authentication possible in online programs.
* Utilized **Spring Boot 3** for rapid development of cloud **RESTful Microservices**, minimizing configuration overhead.
* Leveraged **Swagger/Open API** specifications to visually represent **RESTful APIs**, automating documentation, testing.
* Experienced in installing, configuring, and monitoring the Datastax Cassandra Cluster, **DevCenter** and **OpsCenter**.
* Excellent understanding of **Cassandra Architecture** and management tools like **OpsCenter**.
* Experience in creating and managing **multi-threads** concurrently in Java 17 applications.
* Proficiency in **synchronizing threads** using synchronization mechanisms like locks, synchronized blocks, and volatile.
* Experience working with **Angular** to setup **Web API’s** using **Javascript.**
* **AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS), enabling you to run your code without provisioning or managing servers.**
* **Sybase is a relational database management system (RDBMS) developed by Sybase Inc., now owned by SAP.**
* **Redis has a large and active community, which contributes to its ongoing development and ecosystem of libraries and tools.**
* **The Netflix stack is a collection of technologies and tools used by Netflix to build and operate their streaming platform.**
* Provided full range of production DB2 support for all mission critical databases, Planning for **VSAM to DB2** conversion**.**
* **TypeScript** supports gradual typing, allowing you to gradually introduce type annotations to existing JavaScript.
* Knowledge of **multi-thread safety** issues and techniques for avoiding race conditions and deadlock situations.
* Implemented a dynamic image rendering web service that is consumed by **JSP, jQuery, JavaScript, and AJAX.**
* Object / Data Migration from one **DB2** subsystem to another; develop REST APIs using **MuleSoft Any point API Platform** Support and Administer SQL replication for **DB2** on Z/OS environment.
* Building/Maintaining Docker container clusters managed by **Fnetes Linux**, Bash, GIT, Docker, on GCP
* Worked with **OpenShift** platform in managing Docker containers and **Kubernetes Clusters**.
* **Lambda functions are event-driven and execute in response to events or triggers, making them highly scalable and efficient.** Utilized various utilities like JSP, JSTL, JavaScript, HTML, & CSS, and jQuery.
* **The Netflix stack includes a variety of open-source software and proprietary technologies tailored to meet Netflix's specific needs.**
* Refactored existing project to make it more **RESTful** and **thread-safe** usin**g** Spring Boot.
* **Sybase provides a robust and scalable solution for managing structured data.**
* **Redis is widely used as a caching layer, message broker, and real-time data streaming platform.**
* Configured AWS **Multi Factor Authentication**in **IAM**to implement 2 step authentication of user's access using Google Authenticator and AWS Virtual MFA; Involved in the creation of dynamic web pages with the use of **JSP and HTML.**
* Implemented the application using **Spring Boot 3** Framework and handled the security using **Spring Security**.
* Implemented **Kafka producer** and consumer applications on **Kafka cluster** setup with help of Zookeeper.
* **You can write AWS Lambda functions in various programming languages, including Python, Node.js, Java, and more, offering flexibility and developer choice. MongoDB** is a popular open-source NoSQL database management system.
* It is designed to store and manage large volumes of structured, semi-structured, and unstructured data using **MongoDB**.
* **Node.js's** single-threaded event loop for efficient handling of multiple connections, optimizing **I/O-bound**.
* Integrated **PL/SQL** with Java 17 applications on Unix or Windows platforms to harness **robust** database capabilities.
* Implemented **JSON web-based** tokens for authentication and authorization security configurations using **Node.js**.
* Developers can define **GraphQL** schemas using Java code, specifying types, fields, and operations.
* Used Micro service architecture with **Spring Boot** **3** based services interacting through a combination of REST and **Apache Kafka** message brokers and worked with Kafka Cluster using **ZooKeeper**.
* **Netflix's primary programming language within the stack is Java, which is used extensively for backend services and microservices.**
* **Sybase supports SQL (Structured Query Language) for querying and manipulating data.**
* **Redis also provides Lua scripting capabilities, allowing developers to execute custom logic on the server side.**
* **MongoDB** uses a document-oriented data model, where data is stored in flexible, JSON-like documents.
* Documents in **MongoDB** can have varying structures, allowing for dynamic schema and easy updates.
* Automating **Git workflows in Java applications** for tasks like code review, testing, and deployment.
* Integrating Git with issue tracking systems like Jira or GitHub Issues through **Java 17 APIs.**
* Implementing **Git-based authentication** and authorization mechanisms in Java 17 applications.
* **AWS Lambda functions are stateless and can be easily scaled horizontally to handle varying workloads, ensuring optimal performance and resource utilization.** Generating Git commit statistics and reports using **Java-based Git APIs.**
* **Setting up CI/CD** pipelines to automatically build, test, and validate code changes upon each commit using DevOps.
* Tuned **MuleSoft batch** applications to obtain optimum performance and **CPU utilization**.
* Implemented **Single Sign-On** solutions for seamless authentication across multiple applications using **Spring Boot**.
* Developed **RESTful services** with **Node.js** to facilitate communication between frontend and backend systems.
* Utilized **Angular UI** (Angular Bootstrap) for frontend development, creating **UI components** and directives.
* Leveraged **JDBC API** in Java applications for seamless interaction with **MySQL** databases.
* Experienced in Automating, Configuring, and deploying instances on AWS, Azure environments and Data centers, also familiar with **EC2**, **Cloud watch**, **Cloud Formation** and managing security groups on AWS.
* **Netflix uses the Spring Framework, specifically Spring Boot, as the foundation for building and deploying Java-based microservices in the Netflix stack.**
* **AWS Lambda automatically manages the underlying infrastructure, including compute resources and scaling, freeing you from operational overhead.** **Jenkins** server for automated builds and deployments, integrating seamlessly with **GIT**.
* Utilized **Spring Boot** to develop microservices, deployed with **Docker containers** for scalability.
* **Integrating CI/CD** practices into the DevOps workflow to automate software delivery processes.
* **Sybase offers features such as transaction management, data integrity enforcement, and multi-user concurrency control.**
* **Redis has built-in replication and persistence options to ensure data durability.**
* Utilized Java libraries for **NFC** communication to enable secure contactless payments in POS systems.
* Conducted comprehensive **SOAP and web services** testing using **SOA UI** for robust application validation.
* **Netflix leverages Apache Cassandra, a highly scalable and distributed NoSQL database, for storing and managing large volumes of data with high availability.**
* Employed **XSLT** for transforming **XML** documents, enhancing data presentation in Java applications.
* Embraced **JSON** as the preferred data format for **RESTful web services** due to its simplicity and readability.

**Environment:** NestJS, HTML5, CSS3-SASS, JSTL, Twitter Bootstrap4.0, OIDC, TypeScript, ES6 JavaScript, EKS, HTTP Service, Spring boot 3,Selenium**,** GraphQL, JPA, Docker, Java 17, ORM, Lambda, DB2, AWS CDK, AWS API Gateway, **AWS Cognito,** AWS RDS, ECS, **AWS Kinesis, AWS Glue**, SOAP, SPA, Netflix, Salesforce, MuleSoft, KS, Struts, Kubernetes**,** Kafka, Node.js, Shell scripting, JSON, XSLT, DevOps, SQL, Elastic Beanstalk, Cassandra**,** Angular, CI/CD, Git, UNIX, IAM, SSO, Elastic Search, Redis, XML, NPM, Web pack, RESTful, EC2, JSTL, CSS, React, JavaScript,MongoDB**,** TypeScript, MySQL, Spring CLI.

**State of Utah, UT. January 2019 – May 2021**

**Full Stack Java Developer**

**Responsibilities:**

* Developed **RESTful web** services with **Spring Boot 3** , integrating **Hibernate** for efficient database interaction.
* Integrated **third-party APIs** and libraries to extend functionality and streamline development.
* Documented **REST APIs** with **Swagger** and **orchestrated Docker containers** for deployment.
* Utilized **JSP** for scalable web application development with **MVC** architecture.
* Used **Hibernate** for efficient database object retrieval, **Configured Jenkins** for continuous **integration and deployment**.
* Modified Cassandra. yaml and Cassandra-env.sh files to set the **configuration** properties like node addresses, **Memtables size** and **flush times** etc. Integrated **OAuth 2.0 and OIDC** for user identity and authentication in online programs.
* **AWS Lambda functions can be triggered by various AWS services, such as API Gateway, S3, DynamoDB, and more, enabling seamless integration with your AWS ecosystem.**
* **The Netflix stack includes Apache Kafka, a distributed streaming platform, for building real-time data pipelines and event-driven architectures.**
* **Sybase provides high availability and fault tolerance through features like replication and clustering.**
* **Redis offers various features such as caching, pub/sub messaging, and geospatial indexing.**
* **CDK** provides a command-line interface (CLI) and APIs for deploying and managing **CDK** applications.
* **CDK** integrates with **AWS SDKs** and services, allowing developers to interact with **AWS** resources and services directly from their **CDK code. MongoDB** supports **ACID** transactions at the document level, ensuring data consistency.
* **Selenium supports cross-browser testing,** allowing tests to be run on different browsers like Chrome, Firefox, Edge, Safari, and Internet Explorer. Implemented **React-based** presentation layers to enhance user interface and experience.
* Utilized **Spring Security** and transactions in applications, **integrating Spring Framework** components into Microservices architecture. Involved in developing custom talabs, JSP, EJB's, Portlets and DB2 - stored procedures.
* **MySQL JDBC** driver is required to establish a connection between Java 17, applications, and MySQL databases, which can be included in the project's dependencies.
* **Netflix uses Apache ZooKeeper for distributed coordination and maintaining configuration information across the Netflix**
* **You only pay for the actual compute time consumed by your AWS Lambda functions, with no upfront costs or idle resources, leading to cost-effective and efficient resource utilization.**
* **AWS Lambda functions can be integrated with other AWS services, enabling serverless architectures and event-driven workflows for building scalable and resilient applications.**
* Java 17 applications can connect to **MySQL** databases either locally or over a network, depending on the deployment scenario; MongoDB has high availability and scalability through features like replica sets and sharing using **MongoDB**.
* **MongoDB** provides horizontal scalability, allowing you to distribute data across multiple servers or clusters.
* Good functional knowledge on Oracle Confidential XStore Suite of applications (Xstore XOffice, XPay).
* Deployed and managed containerized applications using **Azure Kubernetes Service** (AKS).
* **Netflix relies on the Netflix OSS (Open Source Software) suite, which includes various libraries and tools developed by Netflix to address specific challenges in building distributed systems at scale.**
* **The Netflix stack incorporates the concept of microservices architecture, where applications are divided into smaller, loosely coupled services that can be developed and deployed independently.**
* Leveraged **Quarkus** for streamlined development tasks and integrations with technologies like **Hibernate ORM**, **REST Easy, Kafka, and MongoDB;** Utilized **NoSQL** databases for handling large volumes of unstructured data.
* Utilized **EKS** with standard **Kubernetes** tooling **APIs** for seamless integration and management.
* Employed **SOAP UI** for testing **SOAP** services and **REST APIs**, incorporating various testing methodologies into the framework using **Spring Boot 3**; **DAO layer** with Hibernate standards, to access data from IBM DB2.
* **Selenium offers robust handling** of dynamic web elements and **AJAX-based** applications, ensuring reliable test automation for modern web technologies.
* **Sybase has been widely used in enterprise applications for its performance, reliability, and support for large-scale data management.**
* **Redis supports a wide range of data structures, including strings, lists, sets, sorted sets, and hashes.**
* **Setup/Optimise** ELK {Elasticsearch, Logstash, Kibana} Stack and Integrated **Apache Kafka** for data ingestion.
* Develop Nifi workflow to pick up the data from rest **API** server, from data lake as well as from SFTP server and send that to **Kafka broker**; Extending Git functionality using **Git plugins or extensions** developed in Java.
* Configuring alerts and notifications to promptly notify teams about any issues or anomalies detected in the **CI/CD** pipeline or deployed applications using **DevOps**.
* **AWS Lambda provides built-in logging and monitoring capabilities to help you troubleshoot and monitor the execution of your functions, ensuring operational visibility and control.**
* Establishing collaborative and cross-functional teams that work together to continuously improve the **CI/CD** process and drive innovation using **DevOps**. Implementing Git submodule functionality in **Java for managing dependencies**.
* Static typing in **TypeScript** provides type safety by allowing developers to specify types for variables, function parameters, and return values, Creating **Git aliases and custom commands** programmatically in Java.
* Integrating Git with continuous integration (CI) pipelines using Java-based tools like **GitLab CI.**
* Building custom **Git clients or GUI** applications using Java 17 frameworks like JavaFX or Swing.
* **TypeScript's type** inference automatically infers types based on usage, reducing the need for explicit type annotations.
* **Selenium** can be integrated with various testing frameworks such as **TestNG, JUnit, NUnit, and Mocha**, providing additional functionalities like assertions and test reporting.
* **Netflix leverages containerization technologies like Docker and orchestration platforms like Kubernetes to manage and scale their microservices within the Netflix stack.**
* **Redis is an open-source, in-memory data structure store.**
* **Redis is known for its high performance and low latency, making it suitable for use cases that require fast data access.**
* Utilized **Angular TypeScript** for frontend development, enhancing user experience in single-page applications.
* Implemented backend services for state management using **Amazon S3 and DynamoDB.**
* **AWS Lambda functions can be deployed and managed using AWS Lambda Console, AWS CLI (Command Line Interface), or AWS SDKs (Software Development Kits), offering multiple deployment options to suit your workflow.**
* Managed **Docker orchestration and Docker containerization** using **Kubernetes.**
* **TypeScript's async/await** syntax simplifies asynchronous programming, making it more readable and maintainable.
* Used **Kubernetes**to orchestrate the deployment, scaling and management of Docker Containers.
* Using **JDBC**, developers can perform CRUD (Create, Read, Update, Delete) operations on **MySQL** databases, manipulate data, and manage transactions efficiently.
* Utilized **Spring Boot 3 auto-configuration** feature to streamline setup and reduce boilerplate code in application development. Implemented service mesh using Istio for advanced traffic management and security in OpenShift.

**Environment:** Java/J2ee, Spring Boot 3 , REST, EDBC, SQL, HTML5, CSS, JavaScript, jQuery, AJAX, JSON, React, Eclipse, JBoss, Maven, Azure, DB2, Kafka, Micro Services, XSLT, DB2, Postman, Angular, API, CI/CD, Salesforce, TypeScript, MuleSoft, AWS CDK, Selenium, Node.js, Swagger, Unix, Docker, Netflix, Java 17, Spring CLI, Lambda , AKS, ICAM, Spark, Terraform, SOAP, JSTL, JMeter, Barcode4J, RESTful, MongoDB, SNAP, Couchbase UML, Quarkus, Cassandra**,** Agile, XP, Xstore, Open Shift, DevOps, Kubernetes, XML, PL/SQL, ORM, MySQL.

# **FM Global –Rhode Island, NY January 2017 – December 2018 Java Full Stack Developer**

**Responsibilities:**

* Developed and Implemented **Micro Services using Java/J2EE**, **Spring Boot 3,** and **Spring Framework**.
* Used **JSF** & **RFID** framework in developing a user interface using Components, Validators, Events, and Listeners for handling request processing; Involved in writing the test cases for the application using **JUnit**.
* Have knowledge on partition of **Kafka messages** and setting up the replication factors in Kafka Cluster.
* Handled the security and Authorization by **Spring Security** and **OAuth authentication Protocol**.
* Involved in the design and development of UI components using **Angular**, **JavaScript**, HTML, CSS, and **Bootstrap**.
* Implemented **MongoDB** database concepts such as locking, transactions, indexes, and replication.
* Implemented **JSON** web-based tokens for authentication and authorization security configurations using **NodeJS.**
* Worked with modules like **MongoDB** and **Mongoose** for database persistence using **Nodejs** for the interactions with MongoDB; Used Kafka for actively tracking real-time data processing, messaging, and log aggregation.
* Prepared statements and connection pooling are commonly used techniques in Java applications to optimize database interactions with **MySQL**, improving performance and security.
* **The Netflix stack is designed to be highly available, fault-tolerant, and scalable, allowing Netflix to serve millions of customers worldwide with uninterrupted streaming services.**
* **Redis is often referred to as a key-value database, as it stores data as key-value pairs.**
* Deployed **Spring Boot 3** applications with microservice architecture using **REST** and **Apache Kafka**; **BPMN platforms** typically expose REST APIs for interacting with process instances, tasks, and variables programmatically.
* Worked on **Big Data Integration & Analytics based** on Hadoop, SOLR, Spark, Kafka, Storm and web Methods.
* **Selenium provides** support for headless browser testing, allowing tests to be executed without a graphical user interface, which can improve test speed and efficiency, **particularly in continuous integration pipelines**.
* Involved in development of test environment on Docker containers and configuring the Docker containers using **Kubernetes.** BPMN workflows can model the end-to-end order fulfillment process in a warehouse.
* Developed microservice on boarding tools leveraging java and Jenkins allowing for easy creation and maintenance of build jobs and **Kubernetes deploy** and services.
* **JSTL** is an essential tool for developers working with JSP to improve productivity and code quality in web applications.
* worked on creating **serverless Microservices** by integrating **AWS Lambda**, **S3**, **cloud** **watch**, and **API gateway**.
* Message broker services are achieved in the application using Active MQ with the JMS framework.
* Implemented **Spring Dependency Injection (IOC)** feature of spring framework to inject beans into the User Interface and **AOP** for **Logging**. Integration with identity providers and directory services is common in **Java ICAM**.
* Ability to collaborate effectively with cross-functional teams like **web applications** to **analyze requirements**, design solutions, and deliver successful projects.
* **Netflix utilizes a range of monitoring, observability, and logging tools, such as Atlas, Spectator, and Eureka, to ensure operational visibility and diagnose issues within the Netflix stack.**
* **RFID data** enabling features such as inventory management, asset tracking, and supply chain visibility.
* Used **Spring Security framework** for login authentication, and password hashing. Worked on **Java Message Service (JMS) API** for developing message-oriented middleware (MOM) layer for handling various asynchronous requests.
* **XStore’s** POS terminals can be deployed within warehouse facilities to facilitate sales transactions, order processing, and customer interactions.
* **Selenium** has a large and active community, with **extensive documentation**, forums, and online resources available for support and learning. Utilized **Kubernetes**for the runtime environment of the **CI/CD**system to build, test deploy.
* Used **SOAP UI tool** to test SOA based architecture application to test **SOAP** services & **REST API**. Used Page Object Model, UI Mapping, Selenium Manager, Data Driven using **CSV, CSS** Selectors as a part of Framework.
* Performed **SOAP / web services** testing using **SOAP UI**, wrote adaptors to test / invoke web services.
* Implemented **Quartz scheduler** to execute batch programming using **Spring Batch**.
* Configured Jenkins to handle application deployment in the **Pivotal Cloud Foundry (PCF) server** and to integrate with **Git Hub version control**.
* Build a **CI/CD pipeline** using **Jenkins automation** and **GitHub repositories** using the webhooks plugin.
* **JIRA** tool for project management used in the application to track bugs, issues, and project management functions.
* Set up **Jenkins server** and build jobs to provide automated builds based on polling the **GIT source control system**.

**Environment:** Java, J2EE, Spring Boot 3, Microservices, AWS, JSF Framework, Spring Security, Angular, Node.JS, Kafka, JavaScript, HTML5, Docker, Java 17, CSS and Bootstrap, XStore, REST API, Unix, ORM, Micro Services, Kubernetes, RESTful, CI/CD, RFID, NoSQL Mongo DB, Barcode, JUnit, Netflix, Jenkins, Swagger, Log4j, JIRA, AKS, Git, BPMN Amazon Web Services, Selenium, MySQL, MongoDB, JPA Framework, SPA, Cassandra. Spring Rest, SOAP API, Spring MVC, Spring Security, Hibernate, IAM**.**

**Cigna, Plano, Texas. November 2013 – November 2016**

**Java Developer**

**Responsibilities:**

* Involved in System Analysis and Design methodology as well as Object-Oriented Design and development using **OOAD** methodology to capture and model business requirements.
* Expertise in version control systems like **Git** for collaborative development and code management in both **Windows forms and web application projects**. Implemented the service layer using Spring 5 with transaction and logging interceptors.
* **Access Management (IAM)** in Java projects involves integrating **authentication, authorization, and other security-related functionalities**. Worked in **TDD** and performed testing using Mocha/Chai and Cypress.
* Responsible for **JAVA 17, J2EE, JSF, JBPM, EL, Groovy, and Java FX** coding/development.
* Implemented the web-based application following the MVC II architecture using the **Struts** **Framework**.
* Developed front-end components using **JSP, and Struts Tiles** layout. Involved in unit testing using **JUnit**.
* Worked in using **React JS components, Forms, Events, Keys, Routers, Animations, and Flux** concepts.
* Used React JS for templating for faster compilation and developing reusable components.
* Build Servers using AWS, importing volumes, launching EC2, RDS, and creating security groups load balancers (ELBs) in the defined virtual private connection using Swagger. Used JBPM to manage workflow, and schedule Jobs.
* **Spring JPA** can automatically generate queries based on method names defined in repository interfaces developed.
* back-end web services using **Node JS** and stored dependencies **using NPM**.
* Implemented **Java 17 J2EE** patterns like Data Access Object, Session Facade, Business Delegate, and Value Object.
* Used spring framework for **middle-tier and spring-hibernate** for data access.
* Used Spring AOP to implement various features of the application such as security, where crosscutting concerns.
* Developed SPA project using **Angular, TypeScript for frontend, NodeJS**.
* Worked on node Web API for consuming the **REST** Endpoint using **AWS**.
* Worked on developing applications, and automated scripts leveraging MongoDB.
* Used **Spring Boot** for building applications with spring initializers and **bootstrapping** the build configuration and developing custom solutions that act upon **Jenkin's** information in authentic time.
* Recommend and develop a strategy for integrating the internal security model into new projects with **Spring** **Security and Spring Boot**. Developed client-side Interface using React JS.
* Used the Mozilla Firefox extension, Firebug, to view and debug **HTML, DOM, and JavaScript**.
* Project coordination with other **Development teams, System managers**, and webmasters and developed a good working environment.

**Environment:** Java JEE 17, 11, JSP, MongoDB, Servlets, Node JS,API, JSF, Spring DI/IOC, Hibernate, Angular, Swagger, Groovy, XML, HTML, JDBC, Web services, CI/CD, AWS, Selenium, Git, Web Services (SOAP/WSDL and REST), MySQL, React.js Jasper, ORM, Web Logic Application Server, Junit, Maven, Log4J, SVN, Oracle 10g, JavaScript, CSS, UNIX.

**Atom Technologies, Mumbai, India July 2012 - September 2013**

**Java Developer**

**Responsibilities:**

* Used **Spring framework MVC** in the development of new modules based on requirements using J2EE.
* Utilized various utilities like JSP, JSTL, JavaScript, HTML, & CSS, and jQuery.
* Refactored existing project to make it more **RESTful** and **thread-safe** uin**g** Spring Boot.
* Involved in the creation of dynamic web pages with the use of JSP and HTML.
* **Conﬁgured and set up Java Workspace** which included setting up server and theme installation and configuration. Became familiar with using and navigating through Java Workspace (Eclipse).
* **Debugged Java classes** used to modify web pages such as Business Objects and Data Access Objects.
* Implemented JDBC to perform the database calls from the Java layer.
* Used JDBC to persist Java Objects into the database.
* Employed **Spring Boot's auto-configuration** feature to streamline setup and reduce boilerplate code in application development. Implemented logging for the whole application using **Log4j**.
* For **XML marshaling** and **unmarshalling used JAXB**. Worked with databases like **Oracle**.
* Used Servlets as an interface between the front and backend to handle **HTTP requests** and send the response back to the front end. Tested many components of web applications and documented my observations.
* Utilized **Spring Boot's** robust ecosystem of starter dependencies for accelerated development of production-ready applications. Participated in **unit testing**, **integration testing,** and installation of the system.
* Participated in analyzing the requirements and depicted them through use cases and scenarios.
* Contributed to the decision-making of the software and hardware requirements and tool evaluations.
* Involved in daily scrum meetings and worked on the sprint tasks in the **Agile Scrum development**.

**Environment:** Java, J2EE, Spring, Core Java, SOAP/Rest, Web Services, API, JavaScript, HTML5, CSS, SASS, JIRA, Linux, Selenium NOSQL(MongoDB), AJAX, Apache JMeter, RESTful, Spring Boot.