SAMPLE JOB DUTIES

Do not copy any of these duties

Job Duties:

1. **Software Development and Maintenance (25%)**
    - Develop and maintain robust, high-performance applications using **Java, Python, C++, or JavaScript** for both back-end and front-end development.
    - Ensure software scalability and maintainability by utilizing **design patterns**, **object-oriented programming (OOP)** principles, and **version control** (e.g., Git).
    - Continuously optimize and improve existing codebase through **refactoring**, modularization, and integration of **third-party libraries** and services.
2. **Requirement Gathering and Collaboration (15%)**
    - Collaborate with **cross-functional teams** (product managers, UI/UX designers, and other engineers) to gather business and technical requirements, ensuring alignment with project goals.
    - Translate high-level requirements into **technical documentation** (e.g., use cases, user stories) and break down complex problems into actionable tasks and solutions.
    - Ensure effective communication between teams using **Agile/Scrum** methodologies, participating in **sprints**, **standups**, and planning meetings.
3. **Designing and Architecting Software Solutions (10%)**
    - Design and implement scalable software solutions using **microservices architecture**, **RESTful APIs**, and **cloud technologies** (AWS, Azure, or GCP).
    - Create **system architecture diagrams**, database schemas, and data flow diagrams to ensure efficient data management, integration, and system performance.
    - Apply **SOLID principles** and **design patterns** (e.g., MVC, Singleton, Factory) to ensure clean, reusable, and maintainable code.
4. **Testing and Quality Assurance (10%)**
    - Write and execute **unit tests** using frameworks such as **JUnit**, **PyTest**, or **Mocha**, and **integration tests** to ensure proper functionality of individual components.

- Implement **Test-Driven Development (TDD)** practices and ensure code coverage meets a minimum threshold using tools like **SonarQube** or **JaCoCo**.
- Perform **end-to-end testing** with continuous integration tools like **Jenkins**, ensuring a high-quality user experience.

5. **Code Reviews and Mentorship (8%)**
   - Conduct **peer code reviews** to ensure adherence to coding standards, best practices, and security protocols, using tools like **GitHub** or **GitLab** for version control and code collaboration.
   - Mentor junior developers on best practices, design patterns, and debugging techniques, and encourage **pair programming** for hands-on learning.
   - Provide **feedback** to ensure code is optimized, readable, and aligned with the team's architecture principles.
6. **Performance Optimization (8%)**
   - Analyze and improve application performance by identifying bottlenecks in the **code**, **database queries**, and **network requests**.
   - Implement **caching strategies** (e.g., **Redis**, **Memcached**) to optimize database load and response times for high-traffic applications.
   - Use **profiling tools** (e.g., **New Relic**, **JProfiler**) to monitor application performance in real-time and address inefficiencies.
7. **Security Implementation (5%)**
   - Follow **OWASP** (Open Web Application Security Project) best practices to safeguard applications from **SQL injection**, **cross-site scripting (XSS)**, **cross-site request forgery (CSRF)**, and **authentication vulnerabilities**.
   - Implement **data encryption** (e.g., **AES**, **TLS**) to ensure the confidentiality of sensitive information both in transit and at rest.
   - Regularly update dependencies and patch security vulnerabilities to maintain secure coding practices and protect the application from cyber threats.
8. **Documentation and Knowledge Sharing (5%)**
   - Maintain clear and concise documentation for codebases, APIs, deployment processes, and software components, using tools like **Swagger** for API documentation.

    - Ensure proper documentation of **database models**, **data structures**, and **system configurations** for future reference and easier onboarding of new team members.
    - Contribute to **internal knowledge-sharing sessions**, documenting lessons learned, technical challenges, and solutions to foster team collaboration.
9. **Continuous Integration and Deployment (5%)**
    - Set up and maintain **CI/CD pipelines** using tools such as **Jenkins**, **CircleCI**, or **GitLab CI** to automate builds, testing, and deployments, ensuring faster release cycles and reducing manual errors.
    - Integrate **containerization** technologies such as **Docker** and **Kubernetes** for application deployment, ensuring consistent environments across all stages of development and production.
    - Ensure the application can be deployed seamlessly across multiple environments (development, staging, production) with minimal downtime using **blue-green deployments** or **canary releases**.
10. **Client and Stakeholder Communication (5%)**

- Act as a technical liaison between clients, product managers, and development teams, providing clear explanations of technical concepts and development progress.
- Collaborate with stakeholders to gather feedback, refine requirements, and adjust project goals based on changing business needs and technical constraints.
- Provide regular project updates and demos to non-technical stakeholders, showcasing new features, system performance, and upcoming releases.

11. **Research and Adoption of New Technologies (5%)**

- Stay current with the latest **development frameworks**, **programming languages**, and **cloud technologies** to enhance software capabilities and team productivity.
- Evaluate new tools, libraries, and software development practices to improve application efficiency, user experience, and development speed.
- Experiment with emerging technologies like **machine learning**, **AI**, or **blockchain** to explore opportunities for innovation within the software.

12. **Debugging and Problem Solving (5%)**

- Debug and resolve complex issues using **debugging tools** (e.g., **GDB**, **Xcode Debugger**) and by analyzing **log files** and **stack traces**.
- Perform **root cause analysis** of critical issues, ensuring that the problem is identified and fixed while implementing measures to prevent recurrence.
- Work closely with the QA and support teams to ensure issues are addressed in a timely manner and are thoroughly tested before going live.

13. **Client Support and Troubleshooting (5%)**

- Provide technical support to clients, addressing urgent issues, identifying root causes, and ensuring the resolution of production incidents.
- Assist in troubleshooting and resolving **post-deployment bugs**, **performance issues**, and client-reported problems.
- Work with the support team to monitor application health and availability in the live environment, ensuring minimal downtime and high service availability.