

Cyprian Mitheme

Optiven Software Developer Interview Preparation

Question & Answer Revision Guide • 23 September 2026

How to use this: Read for structure and key points, not word-for-word memorization. Speak naturally, answer the exact question, and never claim experience you do not have.

1. INTRODUCTION & MOTIVATION

Q: Tell us about yourself.

Suggested answer: Good morning, and thank you for having me. My name is Cyprian Mitheme. I hold a Bachelor of Science in Information Technology, Second Class Honours, from the Technical University of Kenya, and I have 3 years of experience in software development. I currently work as a Full Stack Software Developer, working on frontend and backend systems, databases, APIs, debugging, testing and deployment. My main experience is with PHP, Laravel, JavaScript, jQuery, React, Next.js and MySQL, and I also have experience with Java, Spring Boot and Docker. I enjoy building systems that solve real business problems and I am looking for an opportunity where I can continue growing while contributing to the team.

Q: Why are you interested in this role?

Suggested answer: The role matches my practical experience in developing business systems and gives me an opportunity to grow technically. I enjoy understanding a business problem, translating it into software and making sure the final solution is reliable and useful.

Q: Why should we hire you?

Suggested answer: I have practical experience building and maintaining business systems, not only academic knowledge. I have worked across frontend, backend, databases, APIs, debugging, testing and deployment. I can understand business requirements and translate them into working software, and I am willing to learn, take feedback and improve.

Q: What are your strengths?

Suggested answer: My strengths are problem solving, practical software development, debugging and learning. I am comfortable moving between frontend, backend and database layers when investigating a problem.

Q: What is one area you are improving?

Suggested answer: I am improving the conciseness of my technical explanations. Because I have worked on systems with many connected components, I can sometimes give more detail than necessary. I am working on giving the direct answer first and adding detail when it is useful.

Q: Why are you looking for a new opportunity?

Suggested answer: I am looking for an opportunity where I can continue growing as a software developer, work on more structured and challenging systems and contribute my existing experience in backend, full-stack development, databases and APIs.

2. CURRENT ROLE & PROJECT EXPERIENCE

Q: What do you do in your current role?

Suggested answer: I work as a Full Stack Software Developer. My work involves understanding client and business requirements, developing frontend and backend functionality, working with databases and APIs, debugging, testing, deployment and maintenance.

Q: Tell us about Fleet Kambulus.

Suggested answer: Fleet Kambulus is a business and fleet management system that I helped build from scratch through deployment. It supports vehicles, fuel, spare parts, maintenance, requisitions, suppliers, operational expenses, payments and reports. My work covered frontend and backend development, database design, business logic, APIs, validation, debugging and maintenance.

Q: Explain the requisition module.

Suggested answer: The requisition process captures information such as supplier or fuel station, truck, quantity, price, payment mode and date. The system validates the information, saves the requisition and its details, and provides documentation such as printing for future reference.

Q: How did you design the requisition database?

Suggested answer: I separated the requisition header from its details. The requisition table stores the main information such as requisition ID, number, supplier, date, user and status. The details table stores the individual items or transactions and references the requisition through a foreign key. I also used a temporary requisition table while the requisition was being built before the final save.

Q: How do you prevent a requisition from being partially saved?

Suggested answer: I use a database transaction. The process begins with BEGIN, saves the header and details, and COMMITs if everything succeeds. If an error occurs, I ROLLBACK so the database is not left with only part of the transaction.

3. FRONTEND, BACKEND & ARCHITECTURE

Q: Frontend vs backend?

Suggested answer: The frontend is what the user interacts with, such as forms, tables and pages. The backend handles server-side business logic, authentication, authorization, database operations and APIs.

Q: Explain MVC.

Suggested answer: MVC means Model, View and Controller. The Model handles data and database-related operations, the View presents information, and the Controller handles requests and coordinates the application flow. In Laravel, a typical flow is Route to Controller, Controller to Model or database, then Controller returns a View or JSON response.

Q: Why use MVC?

Suggested answer: MVC separates responsibilities, making applications easier to maintain, debug, test and extend because presentation, request handling and data access are not all mixed together.

Q: Library vs framework?

Suggested answer: A library is reusable code that my application calls when needed. A framework provides application structure and controls much of the application flow. A simple phrase is: with a library, I call it; with a framework, the framework calls my code.

Q: What is Dependency Injection?

Suggested answer: Dependency Injection means a class receives the objects or services it depends on from outside instead of creating them itself. This reduces coupling and makes code easier to test and maintain. Spring's IoC container manages these dependencies.

4. REST APIs & WEB DEVELOPMENT

Q: What is a REST API?

Suggested answer: A REST API is an interface that allows applications to communicate with resources over HTTP. GET retrieves data, POST creates or submits data, PUT replaces or updates a resource, PATCH partially updates it, and DELETE removes it.

Q: Explain a Laravel GET endpoint.

Suggested answer: The request reaches the route, which points to a controller method. The controller handles authentication and authorization where required, calls the model or database, handles errors and returns JSON to the frontend.

Q: What status codes do you know?

Suggested answer: 200 means success, 201 means created, 400 means bad request, 401 means authentication is required or invalid, 403 means authenticated but not authorized, 404 means not found, and 500 means an unexpected server-side error.

Q: How would you secure an API endpoint?

Suggested answer: I would require authentication, check authorization for the specific action, validate backend input, use safe database access, protect sensitive data, use HTTPS and return appropriate errors without exposing internal details.

5. AUTHENTICATION, AUTHORIZATION & SECURITY

Q: Authentication vs authorization?

Suggested answer: Authentication answers "Who are you?" It verifies identity. Authorization answers "What are you allowed to do?" For example, logging in is authentication, while checking whether a user can edit or approve a voucher is authorization.

Q: What is RBAC?

Suggested answer: RBAC means Role-Based Access Control. Users are assigned roles and roles have permissions. The backend checks the user's permissions before allowing protected actions.

Q: Can frontend authorization secure an API?

Suggested answer: No. Frontend checks improve user experience but are not a security boundary because a user can call the API directly. The backend must enforce authentication and authorization.

Q: How should passwords be stored?

Suggested answer: User passwords should never be stored as plain text. They should be securely hashed using an appropriate password-hashing algorithm and verified against the stored hash during login.

Q: A database password was committed to Git. What do you do?

Suggested answer: I would treat it as compromised. First I would rotate or change the database password. Then I would remove it from source code and use environment variables or secure secrets management. I would also remove the exposed credential from Git history where necessary and review repository access. Hashing is for user passwords; a database credential must be protected and rotated.

Q: What is SQL injection?

Suggested answer: SQL injection occurs when untrusted input is incorporated into SQL unsafely. I prevent it with parameterized queries, prepared statements or safe framework database mechanisms, validation and by avoiding direct string concatenation of user input into SQL.

6. DATABASE & SQL

Q: Primary key vs foreign key?

Suggested answer: A primary key uniquely identifies each record. A foreign key references a primary or unique key in another table and establishes a relationship. For example, payments.tenantid can reference tenants.tenantid.

Q: A query with 500,000 rows is slow. What do you do?

Suggested answer: I would first use EXPLAIN to inspect the execution plan. I would look for full table scans, inefficient joins, missing indexes and unnecessary sorting or filtering. Then I would optimize WHERE, JOIN and ORDER BY parts, add appropriate indexes and test performance again.

Q: Are views automatically faster?

Suggested answer: No. A normal database view is mainly a reusable query definition and does not automatically improve performance. Performance depends on the underlying query, indexes, data volume and execution plan.

Q: What is a database transaction?

Suggested answer: A transaction groups related database operations into one unit. If everything succeeds, we COMMIT. If something fails, we ROLLBACK so the database remains consistent.

Q: What is an index?

Suggested answer: An index helps the database find rows more efficiently for suitable filtering, joining or sorting. Indexes also consume storage and can add write overhead, so they should be added based on actual query needs.

Q: What is an INNER JOIN?

Suggested answer: An INNER JOIN returns rows where the join condition matches in both tables. For example, joining requisitions to suppliers can return requisitions with a matching supplier.

7. LARAVEL / PHP LIVE CODING

Q: How would you create a POST employee endpoint?

Suggested answer: I would define a POST route pointing to a controller, validate the request, create the employee through Eloquent or another database layer, and return JSON with an appropriate status such as 201.

Q: How would you validate employee email?

Suggested answer: I would validate it as required and with a valid email format, and enforce uniqueness using Laravel validation. I would also add a UNIQUE database constraint so the database protects the rule even if two requests arrive at the same time.

Q: Why use both validation and a database constraint?

Suggested answer: Application validation gives the user a clear error message. The database constraint provides the final data-integrity guarantee and protects against race conditions or other code paths.

Q: How do you prevent sensitive fields from appearing in JSON?

Suggested answer: I return only the fields the client needs. In Laravel, sensitive model fields can also be hidden using the model's \$hidden property. Passwords and secrets should never be returned in normal API responses.

Q: How do you protect employee creation?

Suggested answer: I would require authentication middleware and then authorize the specific permission, such as employee.create. An unauthenticated request gets 401, an authenticated user without permission gets 403, and only then does the controller validate and save.

8. JAVA / SPRING BOOT

Q: What is Spring Boot?

Suggested answer: Spring Boot is a Java framework built on top of the Spring Framework and is commonly used to develop backend applications and REST APIs. It simplifies Spring development by reducing boilerplate configuration and integrates with Spring Security and Spring Data. Applications can also be containerized using Docker.

Q: Controller vs Service vs Repository?

Suggested answer: The Controller handles HTTP requests and responses. The Service contains business logic and coordinates operations. The Repository handles data access. Separating these responsibilities improves maintainability and testability.

Q: Why is Dependency Injection important in Spring?

Suggested answer: It reduces coupling because a class receives the services it needs rather than creating them directly. Spring's IoC container manages these dependencies, improving maintainability, flexibility and testability.

Q: Spring Boot vs Laravel?

Suggested answer: Both can build backend applications and APIs, but they belong to different ecosystems. Laravel is a PHP framework, while Spring Boot is part of the Java and Spring ecosystem. The choice depends on project requirements, team and existing technology stack.

9. GIT, DOCKER & TEAMWORK

Q: Describe your Git workflow.

Suggested answer: I update main, create a feature branch, make focused changes and meaningful commits, push the branch and create a Pull Request for review. After review and required changes, the branch can be merged.

Q: How do you resolve a merge conflict?

Suggested answer: I update my feature branch with the latest main changes, merge or rebase them, inspect conflicting files, resolve the conflicts carefully, run tests, commit the resolution and push the updated branch.

Q: What is Docker used for?

Suggested answer: Docker packages an application and its dependencies into a container so it can run consistently across environments. It makes deployment more repeatable and reduces differences between development, testing and production.

Q: How do you debug a production issue?

Suggested answer: I reproduce the issue, check browser console and network requests, review application and server logs, inspect the database or stored procedure involved, identify the root cause, fix it, test the complete flow and confirm the issue is resolved.

10. LIVE CODING & PROBLEM SOLVING

Q: What should you do before starting a live coding task?

Suggested answer: First understand the requirement and clarify ambiguity. Then explain the approach briefly, implement the smallest solution that satisfies the requirement, test normal and edge cases and explain decisions while coding.

Q: What if you get stuck?

Suggested answer: I would not go silent. I would explain what I have established, identify exactly where I am stuck and reason through possible approaches. I would simplify the problem and solve it step by step rather than guessing.

Q: How would you build a CRUD API?

Suggested answer: I would define the data structure, create routes, validate input, implement controller and model or service logic, connect the database, return appropriate HTTP responses and test create, read, update and delete operations including validation and error cases.

11. BEHAVIORAL & SITUATIONAL

Q: Tell us about a difficult bug you solved.

Suggested answer: I approach difficult bugs by reproducing them and narrowing down whether the problem is frontend, backend or database. I inspect console and network requests, logs and database or stored-procedure errors. Once I identify the root cause, I fix it, test the complete flow and verify the result.

Q: How do you handle pressure or a tight deadline?

Suggested answer: I clarify the required outcome, prioritize the important functionality, break the work into smaller tasks and communicate early about blockers. I focus on delivering a tested solution rather than rushing unverified changes.

Q: How do you handle feedback from a senior developer?

Suggested answer: I listen carefully, understand the reason behind the feedback and apply it. If I do not understand something, I ask questions. I see code review as an opportunity to improve both the current solution and my future development practices.

Q: What if you do not know an answer?

Suggested answer: I would be honest about what I know and what I have not worked with directly. I would explain related concepts I understand and describe how I would investigate or learn the missing part. I would rather be accurate than pretend to have experience I do not have.

Q: Where do you see yourself growing?

Suggested answer: I want to strengthen my backend and full-stack development skills, especially reliable APIs, database-backed systems and scalable applications. I also want to deepen my Java and Spring Boot knowledge while continuing to build on my PHP and Laravel experience.

12. FINAL INTERVIEW QUESTIONS

Q: What are your salary expectations?

Suggested answer: I am open to discussing the company's range for the role. My expectation would depend on the responsibilities, my three years of practical experience and the overall compensation package. I am open to a reasonable discussion.

Q: Do you have questions for us?

Suggested answer: Yes. I would like to understand more about the development team's current projects, the main technologies used, how developers collaborate and review code, and what success would look like for someone in this position during the first few months.

Q: Is there anything else you would like us to know?

Suggested answer: I am genuinely interested in the opportunity. I have practical experience building business systems from requirements through development, testing and deployment, and I am ready to learn, contribute and grow with the team.

QUICK REVISION SHEET

- **Authentication:** Who are you? **Authorization:** What are you allowed to do?
- **RBAC:** Users → Roles → Permissions; backend enforces permissions.
- **MVC:** Model, View, Controller.
- **REST:** GET retrieve, POST create, PUT replace, PATCH partial update, DELETE remove.
- **401:** unauthenticated. **403:** authenticated but not authorized. **404:** not found. **500:** server error.
- **Transaction:** BEGIN → operations → COMMIT, or ROLLBACK on failure.
- **SQL performance:** EXPLAIN → find bottleneck → optimize → appropriate indexes → retest.
- **DI:** dependencies are provided from outside the class.
- **Git:** update main → feature branch → commit → push → Pull Request → review → merge.
- **Live coding:** Understand → clarify → plan → code → test → explain.
- **Security:** validate input, protect secrets, hash user passwords, enforce backend authorization, use HTTPS and safe database access.

Final reminder: You do not need to know everything. If you know the concept, explain it clearly. If you do not know something, be honest and explain how you would investigate it. Stay calm, listen fully, and do not over-engineer a simple live-coding requirement.