Problem solving and algorithmic thinking answer (Core Coding excercise)

When importing a large CSV file with over 100,000 product records, several issues can cause slowdowns or failures:

1. **Large payload processing in memory**: Parsing and processing all records at once can lead to high memory usage and eventually cause out-of-memory errors or process crashes.
2. **Database locking and contention**: Writing many records in a single transaction or too many simultaneous writes can lock tables and slow down other queries.
3. **Long request timeouts**: Synchronous uploads that block the request-response cycle often exceed server timeouts.
4. **Validation bottlenecks**: Validating each record on the main thread or in a blocking manner can add significant delays.
5. **Error handling**: Lack of proper batching and error isolation can cause the entire import to fail if a single record is invalid.

Propose a Solution

To design an efficient and reliable import process:

1. Chunking and batching: Process the CSV in small batches (e.g., 500–1,000 records per batch). This reduces memory usage and allows partial progress without locking up resources.
2. Asynchronous or background jobs: Move the import task to a background queue (e.g., using a job queue system like Bull for Node.js, Sidekiq for Ruby, or Celery for Python). This prevents blocking HTTP requests and allows for retries.
3. Stream parsing: Use a streaming parser (e.g., csv-parser in Node.js) to process the file line by line instead of loading the entire file into memory.
4. Bulk insert operations: Use database-specific bulk insert features to reduce query overhead and improve performance.
5. Detailed error logging: Capture and log errors per batch, allowing the rest of the file to continue importing rather than failing completely.

Plan the Implementation

1. Validation layer: Validate data per batch asynchronously before inserting. Only save valid records, and log invalid ones for review.
2. Temporary staging table: Load raw data into a staging table first. After all data is in, perform consistency checks and move to production tables in a controlled transaction.
3. Progress tracking: Maintain an import progress status (e.g., “Pending”, “In Progress”, “Completed”, “Failed”) in the database so the user can see the current state.
4. Notification or webhook: Notify users when import is done (e.g., via email or dashboard alert), avoiding the need for them to wait synchronously.

Prepare for Growth

When scaling to millions of records:

1. Horizontal scalability: Use distributed processing workers to split the import job across multiple servers.
2. Database partitioning and indexing: Optimize database schema to handle large inserts, including using appropriate indexes and table partitioning strategies.
3. Rate limiting and throttling: Control the rate of batch inserts to avoid overwhelming the database or impacting other services.
4. Monitoring and auto-scaling: Integrate metrics (CPU, memory, DB load) and auto-scale the background job workers as needed.
5. Retry and idempotency: Ensure imports can be retried safely without duplicating records, by using unique constraints or checksums.

Part 2: Soft Skills and Teamwork

i. Workload Management

To ensure critical parts of the project are completed on time, I would first identify and prioritize tasks based on business impact and dependencies. The most crucial features required for launch (e.g., core procurement workflows, payment processing) should be completed first. Less critical enhancements or nice-to-have features can be deferred to future sprints or post-launch.

I would also break large tasks into smaller, more manageable sub-tasks to make progress more visible and measurable, which helps in reducing overwhelm.

ii. Team Support

For teammates struggling with their workload, I would offer support by redistributing tasks where possible and encouraging open communication about bottlenecks or blockers. Pair programming or peer assistance sessions can help solve challenging issues faster and build team confidence.

Additionally, I would focus on fostering a supportive environment by recognizing achievements, celebrating small wins, and maintaining regular check-ins to understand each team member's mental and emotional state.

iii. Communication with Stakeholders

If it becomes evident that we might miss the deadline, I would communicate this proactively and transparently to stakeholders or person that is more experience than myself as early as possible.

I would clearly explain the root causes of the delay, the current mitigation steps, and propose a realistic revised timeline. It is also important to present options (e.g., a phased rollout with core functionalities first) and emphasize our commitment to delivering a stable and high-quality product rather than rushing an incomplete solution.

This approach helps maintain trust and shows professionalism and ownership over the project outcomes.