|  |  |  |
| --- | --- | --- |
| RAZDA Co. | | |
| **Filename: [order\_queue\_manager.py]** | | |
| **Summary***:*  Automates order queuing and management, processing batches based on priority, peak hours, and load balancing. Interfaces with the database for order tracking and alerts admins of critical issues. | | |
| ***Processes*** | | |
| * **Fetch Orders** | **fetch\_orders retrieves orders based on status and priority (e.g., pending orders) from the database.** | |
| * **Prioritize Orders** | **prioritize\_orders pulls high-priority orders into the queue, ensuring urgent orders are processed first.** | |
| * **Process Orders in Batches** | **process\_orders\_in\_batches handles orders in batches, adjusting load based on batch size and priority.** | |
| * **Process Single Order** | **process\_order attempts to process each order individually, retrying if it fails up to the RETRY\_LIMIT specified in configuration.** | |
| * **Mark Order as Submitted** | **mark\_order\_as\_submitted updates the order status to "Submitted" in the database upon successful processing.** | |
| * **Mark Order as Failed** | **mark\_order\_as\_failed sets an order to "Failed" if retries are exhausted, notifying the admin if needed.** | |
| * **Alert Admin** | **alert\_admin sends a warning notification to the admin in case of critical issues with any order.** | |
| * **Monitor Peak Hours** | **monitor\_peak\_hours dynamically adjusts processing frequency during high-demand hours.** | |
| * **Periodic Order Queue Check** | **periodic\_order\_queue\_check continuously checks and processes orders based on a defined schedule.** | |
| **Files it Gets Information From:** | | **Files it Sends too:** |
| * **MySQL Database**: Pulls and updates order statuses in the Razda order database. | * **Alerts Module**: Sends alerts to admins if an order repeatedly fails, or any issues occur in processing. | |
| * **Environment Variables**: Loads configuration settings, e.g., retry limits, peak hours, batch size. |  | |
| **Expected input into file:** | | **Expected output from file:** |
| * **Orders Data**: Fetches orders from the database with specific statuses or priorities for processing. * **Order Retry Limits and Batch Sizes**: Configurations for handling the number of retries and batch processing frequency. | | * **Updated Order Status**: Orders marked as "Submitted" or "Failed" depending on processing outcomes. * **Order Processing Logs**: Tracks all orders processed, including those that fail or succeed. * **Admin Alerts**: Sends alerts if critical issues or repeated failures occur. |
| **Things that need to be taking place:** | | |
| |  | | --- | | **- Batch Processing for Efficiency: process\_orders\_in\_batches batches orders, balancing load while prioritizing urgent orders.** |  |  | | --- | | **- Retry Mechanism with Exponential Backoff: process\_order retries failed orders with an exponential backoff, reducing network strain and improving success rates.** |  |  | | --- | | **- Peak Hour Adjustments: monitor\_peak\_hours adapts processing speed and frequency during peak hours, ensuring timely handling of high-demand periods.** |  |  | | --- | | **- Admin Notification on Repeated Failures: alert\_admin triggers when orders exceed retry limits, enabling prompt intervention by admins.** |  |  | | --- | | **- Database Connection Management: close\_connection ensures database connections are safely closed, preventing potential data loss or security risks.** |  |  | | --- | |  | | | |
| Edit log (udate each time you make changes to doc or file). | | |
| | **Possible Enhancements:** | | --- |  |  | | --- | | - **Dynamic Load Balancing**: Adjust batch size or processing frequency based on live server load. |  |  | | --- | | - **Enhanced Alerting Mechanism**: Integrate with real-time notifications like SMS or email for critical failures in order processing. |  |  | | --- | | - **Data Insights and Monitoring**: Add detailed analytics on order processing times and error rates, providing insights into operational efficiency and failure patterns. |  |  | | --- | | - **Automated Fallback Systems**: Implement a secondary fallback or alternate server in cases of critical database issues, ensuring continuity in order processing. | | | |
| * Oliver Smith (Razda Admin) Nov 8, 2024: | | |