International Journal of Management, IT & Engineering
Vol. 14 Issue 02, February 2024
ISSN: 2249-0558 Impact Factor: 7.119
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Enhancing Resilience: A Solution Framework for Handling Third-Party Service Disruptions in FinTech Mobile Applications

Amol Gote
Vikas Mendhe

Abstract

In the realm of Financial Technology (FinTech), mobile applications are pivotal for facilitating transactions, including loan applications. However, disruptions in critical third-party services, such as credit bureaus, can significantly impact the user experience. This paper introduces a solution framework for proactively managing these disruptions. During the service downtime, user transactions continue to be processed. These applications are designed to detect service interruptions through specific exceptions and are subsequently rerouted to dedicated queues, along with their request payloads. This process triggers the transmission of an indicator to the mobile application, facilitating appropriate user messaging regarding the service disruption. Post such a service down event, as soon as the first application is successfully processed without encountering third-party service exceptions, an event is triggered to process the remaining queued applications. As part of this processing for each application at the end, the user receives a notification to resume their application. This paper emphasizes the critical role of user-centricity, proactive communication, and strategic queuing in effectively managing disruptions, particularly those involving essential services like credit bureaus, within FinTech mobile applications.

Keywords:

FinTech;
Mobile Applications;
ServiceDowntime;
User Experience;
Queuing;
Loan Applications;
Third-party Services;
Credit Bureaus.

Author correspondence:

Amol Gote
Solution Architect, Plainsboro, New Jersey, USA.
LinkedIn: https://www.linkedin.com/in/aamolgote
Email: aamolgotewrites@gmail.com

Vikas Mendhe
Senior Consultant, Austin, Texas, USA
LinkedIn: https://www.linkedin.com/in/vikas-mendhe-69260012
Email: vikas.mendhe@gmail.com

1. Introduction

In the ever-evolving landscape of FinTech, mobile applications have become pivotal tools for users, offering unparalleled convenience and efficiency in conducting financial transactions. Among the most critical functions of these applications is the seamless processing of essential operations, particularly loan applications, which often rely on the integration of vital third-party services. However, in this intricate web of interconnected systems, vulnerabilities emerge when these third-party services encounter downtime, presenting a formidable challenge to both user experiences and the reliability of the FinTech ecosystem.

The reliance of FinTech mobile applications on external services, such as credit bureaus, underscores the paramount importance of the continuous availability of these services. Any disruption would impact the user experience of the applicant using the mobile application. Acknowledging the gravity of this challenge, this paper delves into a proactive solution framework designed to effectively manage these disruptions.

In this paper, we introduce a solution that not only ensures the uninterrupted flow of loan applications during service interruptions but also prioritizes user-centricity through clear and timely communication. Additionally, this framework is proficient at both identifying disruptions and implementing strategic queuing to ensure data integrity and streamline the process of resuming applications once normal service functionality is restored.

Our study transcends technical solutions, placing a robust emphasis on user experience, proactive communication, and strategic queuing as pivotal components in the effective management of disruptions. Particularly, we highlight the management of critical service disruptions involving entities like credit bureaus. Our aim is to contribute to the resilience of FinTech mobile applications, enhance the user experience, and strengthen the mobile application's resilience with third-party services.

2. Common Workflow Patterns in Mobile Applications.

In this section, we elucidate the existing flows of mobile applications designed to serve various business use cases, such as processing unsecured loan applications for purposes like home improvement and medical financing. This flow adheres to the conventional structure of mobile applications, where the mobile app initiates backend APIs (Application Programming Interface) requests. Subsequently, these backend APIs, in turn, invoke third-party external partner services, as illustrated in the diagram provided.

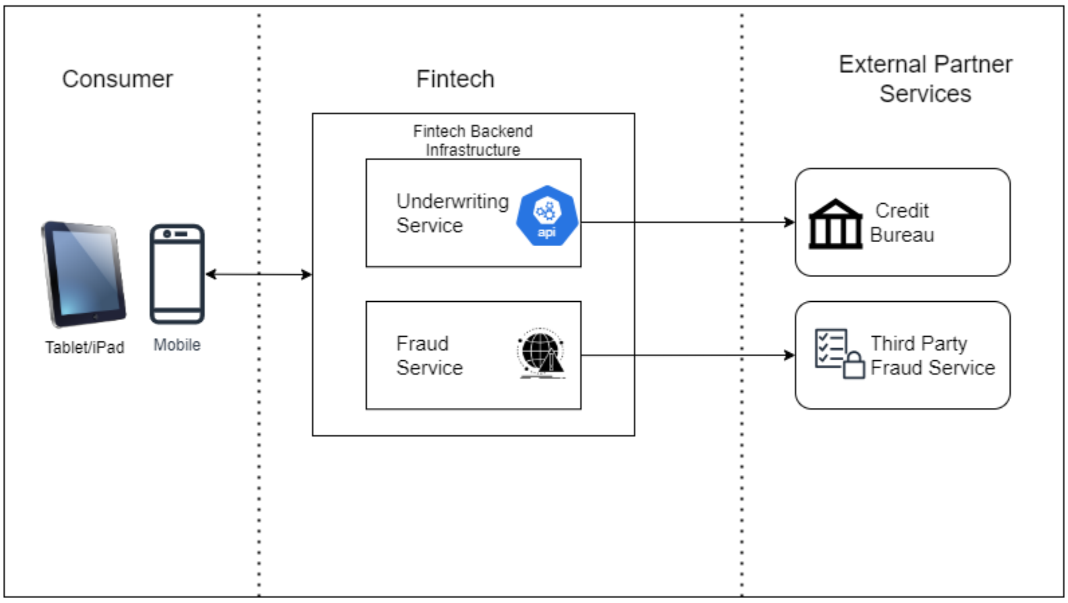


Figure 1.*Common Workflow Patterns in Mobile Applications.*

In this scenario, when the credit bureau service encounters downtime or begins to throw unexpected exceptions, these issues propagate upward, directly impacting the user experience of applicants seeking consumer loans. This can lead to suboptimal user experiences, customer abandonment, and ultimately, a significant loss of business revenue.

To address these challenges and ensure a seamless, positive user experience, it is crucial to effectively manage critical service downtime exceptions. The following section will delve into the proposed solutions to tackle these issues.

3. Handling Critical service down

In this scenario, APIs invoking third-party critical services, such as those with credit bureaus, incorporate resiliency measures. Calls to third-party APIs are designed to handle service downtime exceptions or unexpected errors. If the calling API detects such unexpected behavior, it duplicates the application requests and redirects them to a queue as messages. Once the requests are redirected to the queue, the APIs invoked by the mobile application send an indicator back as an API response, indicating that one of the business-critical services required to process the application is down. Based on this indicator, the mobile application seamlessly manages the user experience and informs the user that one of the critical services is currently unavailable. Users are assured that they will receive a notification once the service resumes and their application will be processed.Depending on the health of the third-party service, further applications may begin queuing up for processing.

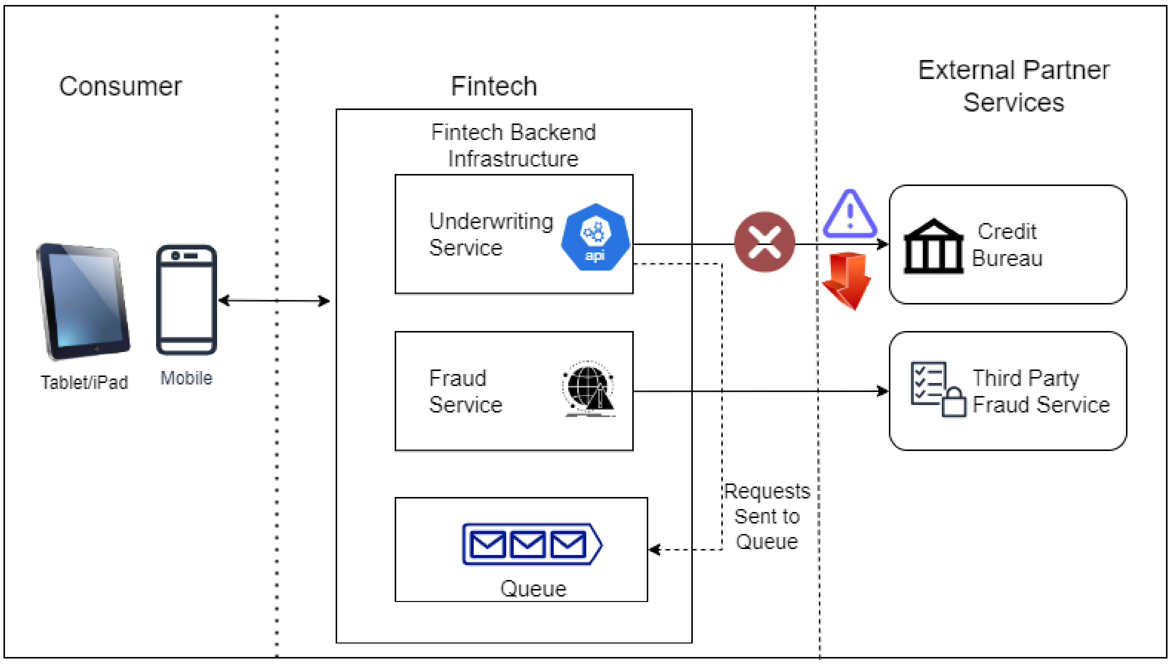


Figure 2. *Handling critical service down with a Queue.*

The queue messages contain the request payloads. For storing these intermittent messages, you have the flexibility to choose between using any type of queue or storing them in a database. If the decision is to store them in a database, it becomes essential to manage message states using indicators such as 'to process,' 'processing,' and 'processed.'

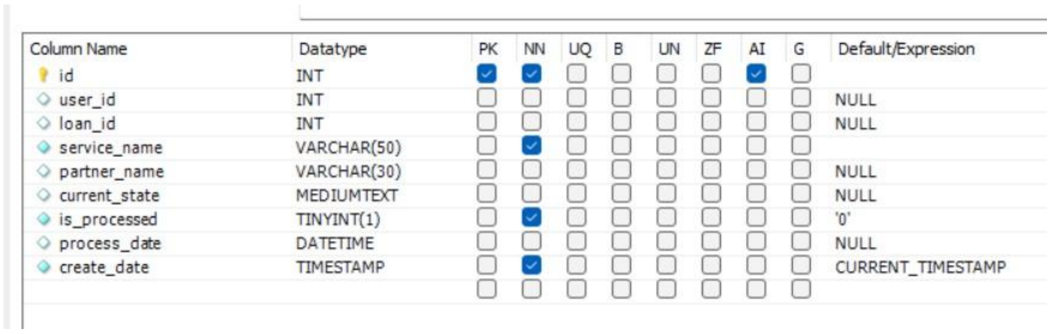
- For queue-based solutions in the cloud, you have options like:
- AWS (Amazon Web Services) Queues
- Microsoft Azure Service Bus
- Google Cloud Pub/Sub

If you prefer not to use cloud services, other queuing solutions include:

- RabbitMQ
- Apache Kafka
- Microsoft Message Queuing (MSMQ)

If the request payload is large, an alternative approach is to create a database record with a GUID (Globally Unique Identifier) identifier and push that identifier to the queue. In this scenario, when a message is dequeued from the queue, the processing backend job retrieves the request payload from the database using the associated identifier. This method efficiently manages large payloads while ensuring data integrity and retrieval when needed.For storing the intermittent state in the database, we can leverage the below table schema as a starting point and tweak it according to the application-specific business:

- partner_name – external third-party partner name e.g. TransUnion
- service_name – Third-party external service name e.g. Bureau_Report
- current_state – this stores the request payload.
- is_processed – this is an indicator to identify the processing stages like, already processing, processing, processed.
- The remaining columns are self-explanatory.

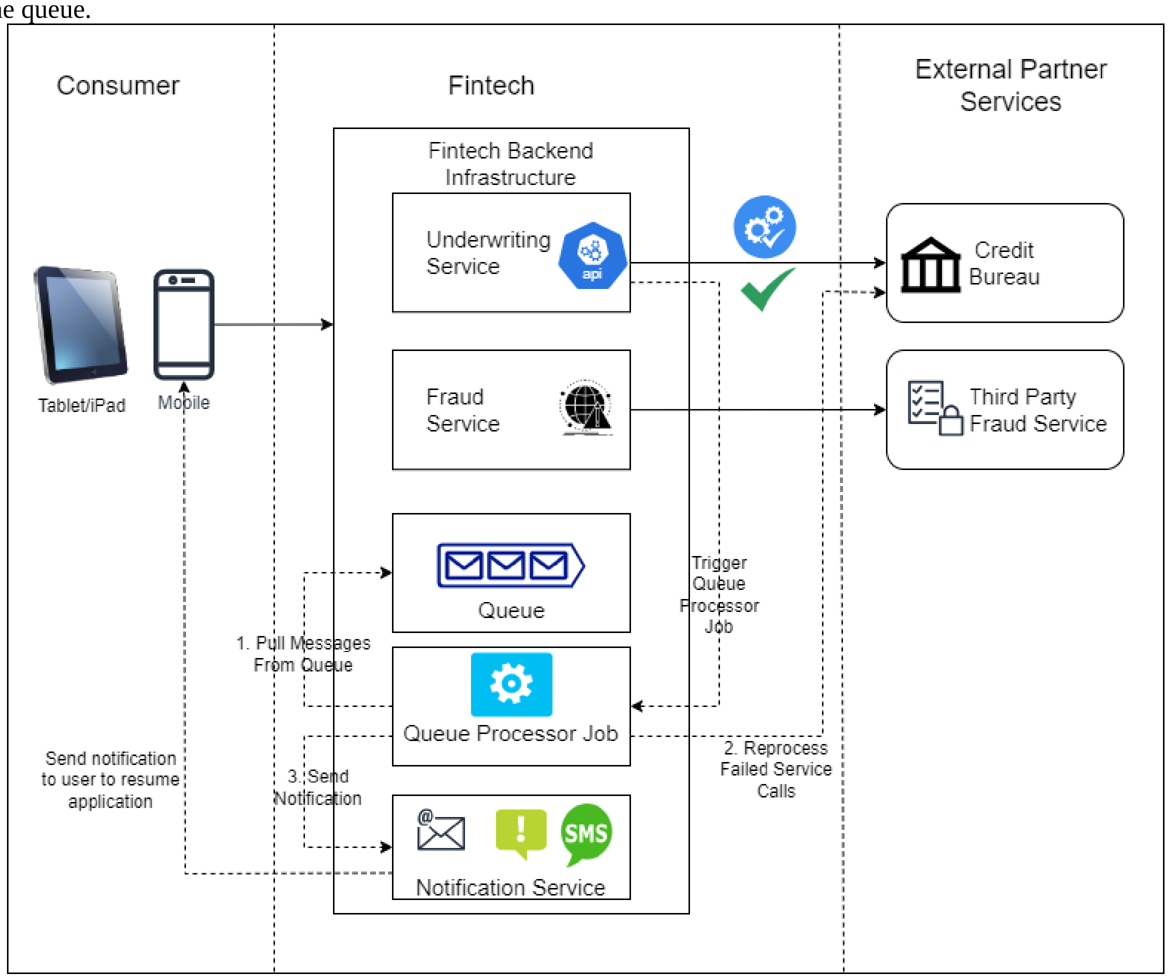


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	✓	✓					✓		
user_id	INT									NULL
loan_id	INT									NULL
service_name	VARCHAR(50)		✓							
partner_name	VARCHAR(30)									NULL
current_state	MEDIUMTEXT									NULL
is_processed	TINYINT(1)		✓							'0'
process_date	DATETIME									NULL
create_date	TIMESTAMP		✓							CURRENT_TIMESTAMP

Figure 3. *Database table for storing request state.*

4. Processing Queued Messages When the Service Recovers.

In the previous section, we discussed how messages are queued based on the health of the third-party external service. Once one of the requests to the service is completed successfully, it serves as an indicator that the service is on the path to recovery. To ensure accuracy, the decision to declare the service as healthy can be configured to wait for a certain number of successful requests to be processed. Relying solely on the result of a single request might not be optimal, especially if the third-party external service is in a recovery stage. Once the threshold for the successfully processed requests is reached, the system will declare the status of the service as healthy and then send a notification to the background job to start processing the messages in the queue.



Consumer
Fintech
External Partner Services
Fintech Backend Infrastructure
Underwriting Service
api
Credit Bureau
Tablet/iPad
Mobile
Fraud Service
Third Party Fraud Service
Queue
Trigger Queue Processor Job
1. Pull Messages From Queue
Queue Processor Job
2. Reprocess Failed Service Calls
Send notification to user to resume application
3. Send Notification
SMS
Notification Service

Figure 4. *Processing queued applications once service turns healthy.*

Once the backend job receives a notification to process the queued messages, it starts its job. As part of this, each message performs the following steps:

- Dequeue the message from the queue.
- Depending on the strategy implemented to persist the request payload (as discussed in Section 3), retrieve the payload.

- Call the external service with the retrieved payload and complete the remaining business processes required for processing the loan application. If the call fails at this stage, move the message to a separate exception queue for manual analysis.
- After completing the business process, send a notification to the end user using the notification service. The notification service is another microservice responsible for sending various types of notifications to end users. These notifications may include push notifications, SMS messages, and email notifications.

Once a user receives the notification, the user can resume the application process, for a better user experience message will contain a deep link for the mobile application, which will take the user to the exact next step in the loan application process.Deep links are URLs that are designed to take users to specific content or screen within a native mobile application, rather than just launching the app's home screen.

5. Benefits of the Resiliency Solutions Framework

Seamless User Experience: One of the primary advantages of this approach is its ability to maintain a seamless user experience, even in the face of service disruptions. By promptly notifying users about service unavailability and providing clear communication, the framework helps prevent frustration and abandonment, ultimately enhancing user satisfaction.

Data Integrity and Reliability: The use of queuing mechanisms and payload storage strategies ensures data integrity and reliability. Storing request payloads and managing message states in queues or databases ensures that critical data and states remain accessible and secure, even during service downtime.

Resilience to Third-Party Service Failures: The framework's resiliency measures enable it to adapt to third-party service failures efficiently. Duplicating and redirecting application requests to a queue, minimizes the impact of service downtime on ongoing processes, allowing applications to queue up for later processing when the service is restored.

In addition to the proactive resiliency measures outlined in this framework, it is essential to note that all third-party services integrated also have robust retry patterns in place. These retry patterns are configured to make multiple attempts, typically three times, with a 10-second delay between each attempt, to reestablish a connection and successfully process requests in the event of temporary service disruptions. After three retries, the approach of placing messages in the queue comes into play. By combining the retry patterns of third-party services with a resiliency framework, we create a multi-layered approach to handle service interruptions effectively.

6. Drawbacks for the Resiliency Solutions Framework

Complexity: Implementing the framework can introduce increased complexity to the application architecture. The need for queuing mechanisms, message state management, and database integration may require additional development efforts and maintenance.

7. Conclusion

In the FinTech world, where mobile applications serve as vital channels for critical transactions, the resilience of these applications in the face of third-party service downtime is paramount. The Proactive resiliency framework offers a solution to this challenge, prioritizing user-centricity, data integrity, and streamlined processing.

Through the diligent handling of service disruptions, including the duplication and queuing of application requests, the framework ensures that users experience minimal disruption and are promptly informed about service unavailability. This commitment to user satisfaction enhances trust and fosters uninterrupted financial operations. Furthermore, the framework's reliance on queuing mechanisms and payload storage strategies guarantees data integrity and reliability, safeguarding essential data during service downtime. Additionally, the system's adaptability to third-party service failures minimizes the impact on ongoing processes, allowing applications to queue up for processing when services are restored.

While resiliency solution brings benefits, it is essential to acknowledge the complexity it introduces to application architectures. Application development teams should be prepared for the additional effort and maintenance required for additional components.

In conclusion, the resiliency framework exemplifies the importance of proactive communication and strategic queuing in managing third-party service disruptions, especially those involving critical entities like credit bureaus, in FinTech mobile applications. By prioritizing user satisfaction and data integrity, we contribute to the resilience and reliability of these essential Fintech applications in an ever-evolving digital landscape.

References

[1] Silva, Marco António Rodrigues Oliveira, "Improving the resilience of microservices-based applications", *University ofMinho, Dissertation reports*, 19th Feb 2021.

[2] James Lewis, Martin Fowler, "Microservices: A Practitioner's Guide."
Retrieved from https://martinfowler.com/microservices, 21st Aug 2019

[3] Pethuru Raj, G. Sobers Smiles David, "Cloud Reliability Engineering: Technologies and Tools", 2021

[4] Roland Kuhn, Brian Hanafee, Jamie Allen, "Reactive Design Patterns", *Chapter 12. Fault tolerance and recovery patterns*, Feb 2017

[5] www.jrebel.com, "Guide to Microservices Resilience Patterns",
Retrieved from https://www.jrebel.com/blog/microservices-resilience-patterns, 1st April 2020

[6] IcePanel, "Top 6 message queues for distributed architectures" Retrieved from https://icepanel.medium.com/top-6-message-queues-for-distributed-architectures-a3cbabf08993, 18th May 2023