PROJECTS

- **Pickup Tab Experience in Amazon Detail Page**
    - Implemented a *"Collection" Tab* in the Product Page where Pickup Options are shown to customers if eligible.
    - Created an Event-driven, Pre-Compute system in AWS (CDK-based) that maintains all reqd. Pickup Location Metadata. This system processes 2K+ CRUD events per second.
    - Implemented the delivery date messaging (*FREE pickup Tuesday, July 2*) for the ***Pickup Tab*** in the Amazon Product Page.
        - It was a complex problem which could have introduced additional latency to the entire Product Page.
        - Worked with Principal Engineers across Retail org. and arrived at a solution which did NOT add any Latency/Traffic overhead.
    - Worked with other core-Amazon teams to integrate with their stack to compute and render necessary data for Pickup Offers(Price, Item Availability, Merchant Details etc.).
    - The *Tab* is rendered upto 40K times in the core Product Page and uplifted orders to Pickup Locations by an estimated 10% (~8K orders per day on average).

- **DHL Postnumber Validation**
    - Amazon has a tie up with DHL in Germany where customers can order to DHL Pickup Locations from amazon.de.
    - To order to DHL pickup locations, customers required a mandatory id called the "*postnumber*".
    - However, customers sometimes entered incorrect/garbage values for their *postnumber* (ex - 123). This was leading to almost ~5K failed deliveries everyday across Germany.
    - Reached out and collaborated with DHL's engineers to implement DHL's internal validation algorithm.
    - Launched the validations which now entirely prevents incorrect postnumbers from being used.

- **Accurate Estimated Delivery Date Messaging for Pickup Locations in Poland**
    - There was a *long existing gap* in the Delivery date service that caused a 1-2 day delay in the estimated delivery date on the Product Page. If the order could arrive by *Tuesday*, the estimated date would be on a *Thursday*.
    - A good portion (~20%) of amazon.pl 's overall orders are delivered to Pickup Locations.
    - This was marked as a critical blocker to launch *Amazon Prime* in amazon.pl (due to high usage of Pickup Locations and Amazon Prime's requirement to provide faster shipping).
    - Identified the gap and spearheaded the development of the solution across *5 core Amazon teams*. The primary challenge was to ensure the solution met the diverse requirements and constraints of all teams, including service tenets, latency/TPS, and operational overhead. Successfully aligned all teams to reach a consensus on a solution that satisfied these constraints.
    - Worked with the amazon.pl Vice President(s) to escalate and prioritize the progress for the fix. Delivered the fix before the expected date and enabled the launch of Prime in Poland.

- **Pickup Checkout Experience WW**
    - Implemented a low-latency (<10 ms) API used in Checkout to "upsell" customers about Amazon Hub Locations near their addresses. This API serves upto ~7K customer requests per second.
    - Owned the primary discoverability microservice across design reviews, office hours, New Feature Requests, etc.
    - Made multiple optimizations and improvements to the microservice -
        - Identified a redundant call pattern reducing ~300 TPS in downstreams traffic for Pickup Location Eligibility.
        - Built Canaries ("OneBoxes") to catch potential issues early, during deployments.

- **Operational Excellence**
    - Enabled dynamic distributed throttling to eliminate bot-related service brownouts. Implemented comprehensive testing and configured dashboards and alarms. Successfully managed peak readiness for Black Friday and Prime Day from 2022-2024, ensuring optimal

scaling and performance while coordinating traffic expectations with upstream and downstream teams.

- **Pickup Config Centralization and Onboarding Automation**
    - There are multiple teams under the Pickup Charter with distinct responsibilities like - Metadata, Discoverability, Ranking, Post-Order, Notifications, Eligibility/Capacity Checks, etc.
    - Multiple teams were using their own data stores to maintain *Partner specific configuration*. As a result, if a configuration had to be added/deleted, it was a cross-team manual effort that took up to 4 developer days.
    - Worked with all involved teams to identify the Configuration requirements and came up with an approach to centralize the configuration to a single data-store. Created a common access library for the centralized data-store which allows each team to read the configurations.
    - Since the configuration was critical to the proper functioning of multiple services/teams, gated the UPDATE / DELETE operations via a *new microservice*. Added human-specific and service-specific authorization and authentication, enabling teams to modify their respective data.
    - Setup a call-chain across all Microservices to automate the configuration onboarding process
    - With this solution, the Partner was only required to be onboarded at a single place.

- **Return to Pickup Point Optimization**
    - This was a pilot program whose initial design had a huge scaling bottleneck.
    - With every expansion to additional zip codes, we required scaling across 3 different microservices and 2 different teams.
    - It was a tedious process as traffic varied from zip code to zipcode based on customer behavior (which couldn't be predicted). We were always either over-scaled or under-scaled with each expansion and had to correct our scaling with time, costing a lot of developer effort and infrastructure costs.
    - Proposed a modification to the existing design by introducing a tradeoff that slightly reduced accuracy for a significant reduction in the overall load to our downstream services.
    - Derived metrics from the existing customer experience to quantify the potential impact (3% drop in Accuracy). Used this to back up the proposal and got the alignment of all Product Managers involved. Implemented the same.
    - Since then, scaling was only needed 25% of the time when compared to earlier and was also only limited to a single microservice.