# Projects I have worked on:

### **Debugger:**

A tool developed from scratch for providing the users with the reason why a particular issue is happening by troubleshooting various services in the pipeline and aggregating the result.

Sample Use case: (Identifies if audible content is playable in browser) <https://sef.amazon.com/debug/view/cloud?useCase=cloud&asin=B078YBNFP6&marketPlaceId=AU&customerId=A2LSZBAKTAMW3Q>

It reduces the development effort of the developer by more than 70%. Currently there are 5 use cases written on top of this platform

**Architecture:** [**https://w.amazon.com/bin/view/Digital\_SE/SEF/Debugger/**](https://w.amazon.com/bin/view/Digital_SE/SEF/Debugger/)

**Code Package:** [**https://code.amazon.com/packages/SEFDebugger/trees/mainline**](https://code.amazon.com/packages/SEFDebugger/trees/mainline)

**CR Phase 1:** [**https://cr.amazon.com/r/5395181/**](https://cr.amazon.com/r/5395181/)

**CR Website:** [**https://cr.amazon.com/r/5561873**](https://cr.amazon.com/r/5561873)

In phase 2, the platform was improved that we can write checks as child parent, such that only if the parent check passes, the child check will get executed. This improved the accuracy of the issue identification dramatically. Also added caching locally to cahce service calls.

**CR Phase 2:** [**https://cr.amazon.com/r/6455147/**](https://cr.amazon.com/r/6455147/)

### **SEFManagement Service Integration Tests:**

Summary: SEFManagementService is a coral service owned by SE Team. But there were no integration test cases available. Hence wrote integration test cases for top two use cases in a separate package.

Package: **Code** **package:**[**https://code.amazon.com/packages/SEFManagementServiceTestClient/trees/mainline**](https://code.amazon.com/packages/SEFManagementServiceTestClient/trees/mainline)

**CR:** [**https://code.amazon.com/reviews/CR-2326787/revisions/2#/details**](https://code.amazon.com/reviews/CR-2326787/revisions/2#/details)

### **BetaAudienceService:**

Currently working on development for BetaAudienceService. The CR is for listening to PeopleAPI events and update BetaAudienceService DB.

**CR:** [**https://code.amazon.com/reviews/CR-2911237**](https://code.amazon.com/reviews/CR-2911237)

### **BetaManagementWebsite:**

The CR is to improve uploading of Bulk uploader for BetaManagementWebsite (Acts as the front end for BetaAudienceService). It has small changes for addressing a lot of small bugs present in BetaManagementWebsite bulk uploader.

**CR:** [**https://code.amazon.com/reviews/CR-2305629**](https://code.amazon.com/reviews/CR-2305629)

### **OEDashboard:**

Stakeholders did not have a way to identify what is the time taken to acknowledge a ticket and resolve it. This made it hard to drive Operational excellence on top of it. In phase 1, worked on development of backend for fetching the data,

For phase 2, worked on adding month over month aggregation, and aggregation of data at the very top level (At audible level)

Sample URL: <https://sef.amazon.com/OEDashboard/AudibleTicketMetrics/ProgramLevel?impactType=ALL&aggregateType=week&vpId=0&resolverGroupId=0>

**CR Phase 1:** [**https://cr.amazon.com/r/7149148/**](https://cr.amazon.com/r/7149148/)

**CR Phase 2:** [**https://cr.amazon.com/r/7837718/**](https://cr.amazon.com/r/7837718/)

**CR Website:** [**https://cr.amazon.com/r/8068709/**](https://cr.amazon.com/r/8068709/)

### **Sparks:**

Sparks, a platform for users to propose ideas to the higher management team, which then will be reviewed by the management, and based on the value of the idea, The higher management will reach out to the individual user for making the idea to a product

We also introduced new features like a dashboard for Admins, where they can see aggregation of all the ideas submitted, and hourly aggregation of when the idea is submitted.

Sample URL: <https://sparks.amazon.com/>

**Code:** [**https://code.amazon.com/packages/SparksV2WebsiteContent/trees/mainline**](https://code.amazon.com/packages/SparksV2WebsiteContent/trees/mainline)

More details on

Debugger:

What?

A tool for debugging common issues that happen in a particular flow like "Searching for an asin in retail site" or "Playing the content online". This involves looking up of various services in the pipeline, identifying the probable cause, displaying the user with the list of checks with the status of each check, and providing an option to the user to reach out to the respective team.

We identified multiple use cases which need a similar tool (Search issues, Player issues, Sync related issues). Instead of developing multiple pages for each use case, I developed a framework, which the developer can leverage and add his use case to it.

Implementation:

Instead of writing a number of files (HTML, CSS, JS for UI, and controllers for UI and back end services, and proxies), The user writes what to be displayed in UI, and what service to be called in backend and proxies for calling them). The framework dynamically renders the page from the user data, dynamically calls the backend services(Use java reflection), aggregates the result and returns the page back to the user.

Whats’ it special about:

It reduces the developer effort by more than 70%. Instead of writing 10 classes, the user has to write one class and add some data in two more files. This also reduces the code reviewers’ effort to review only the bare minimum, instead of having to review an end to end flow.

Currently there are four use cases that are present in the debugger.

