Vinspect

Executive Summary

Building a compliance and audit system for restaurants. To enable ease of use so that employees at the restaurants find it simpler to use and adapt to the system. The system reduced regulatory paper work amongst the restaurants and kept the records all digital.

The challenges

The initial technical non-functional requirements we collected were as below:

Permission driven

Security Aspect

Multi-tenant system with data segregated between users of the system

To ensure that we meet the required expectations, along with meeting the non functional technical needs, we also needed to understand the domain and the way GRC projects are handled.

Building a compliance and audit system for restaurants.

GRC – Governance, Risk and Compliance

The solution

GRC – Governance, Risk and Compliance. GRC projects generally have a list of assessment questions, Risk statements and continuous reviews of the risk statements. These reviews will then provide a report of the current status and the way forward to mitigate these risks for an organization. The reviews also mandate the collection of various documents as evidence of a particular risk mitigated. This data is sensitive and confidential data to the company that gets assessed. This nature of GRC project execution leads to the non functional technical requirements stated above.

Execution

We ran a month of discovery phase where engineers from Aritha attended several discovery sessions with the client. These sessions were focused on understanding the domain, understanding the current ways of handling a project. Understanding the problems with current execution. This lead to the creation of an initial set of requirement documents, wire frames and an overall project plan of when different modules of the solution could be delivered.

Pegging the high-level project plan provides a way to timebox the whole development cycle. Without such a project plan these from the scratch projects tend to have a long tail finishes. These long tail finishes allow for scope creep and budget creep.
During the phase of discovery a develop team was built and was on standby to start the development work immediately. This team consisted of Project Manager, Architect, UI and API developers, Testers and DevOps Engineers.

To arrive at an initial speed, the sprints were cut short to one-week sprints where the development team got into the rigor of developing and delivering quickly. This set the right velocity, quality and allowed the team to have very quick feedback cycles. Once the team settled down, the sprints were changed to two-week sprints.

Technical Solution

A strong user onboarding flow was set up which provided clear demarcation of user roles and responsibility. This allowed the technical team to build the product keeping in context the various users who will act on the product.

To satisfy the strong security needs, the below were implemented from the start so that security is not an afterthought

- File MIME type checks for file uploads
- Token expiry
- Password expiry policy
- Permission based login
- No plain text password
- Multi-tenant system segregating data of customers

File Mime type checks

File MIME type checks were implemented using standard libraries Apache Tika and Apache Commons

TAGs

Multi tenancy

The Multi tenant system allows the clients on the platform to keep their data segregated. Multi tenancy is an architecture in which single instance of the software application serves multiple customers. There are many ways in which multitenancy can be achieved

1. **Database per tenant**
   Each tenant has its own database and is isolated from other tenants

2. **Shared Database, Shared Schema**
   All Tenants share a database and tables. Every table has a Column with the Tenant Identifier, that shows the owner of the row.

3. **Shared Database Separate Schema**
   All Tenants share a database, but have their own database schemas and tables

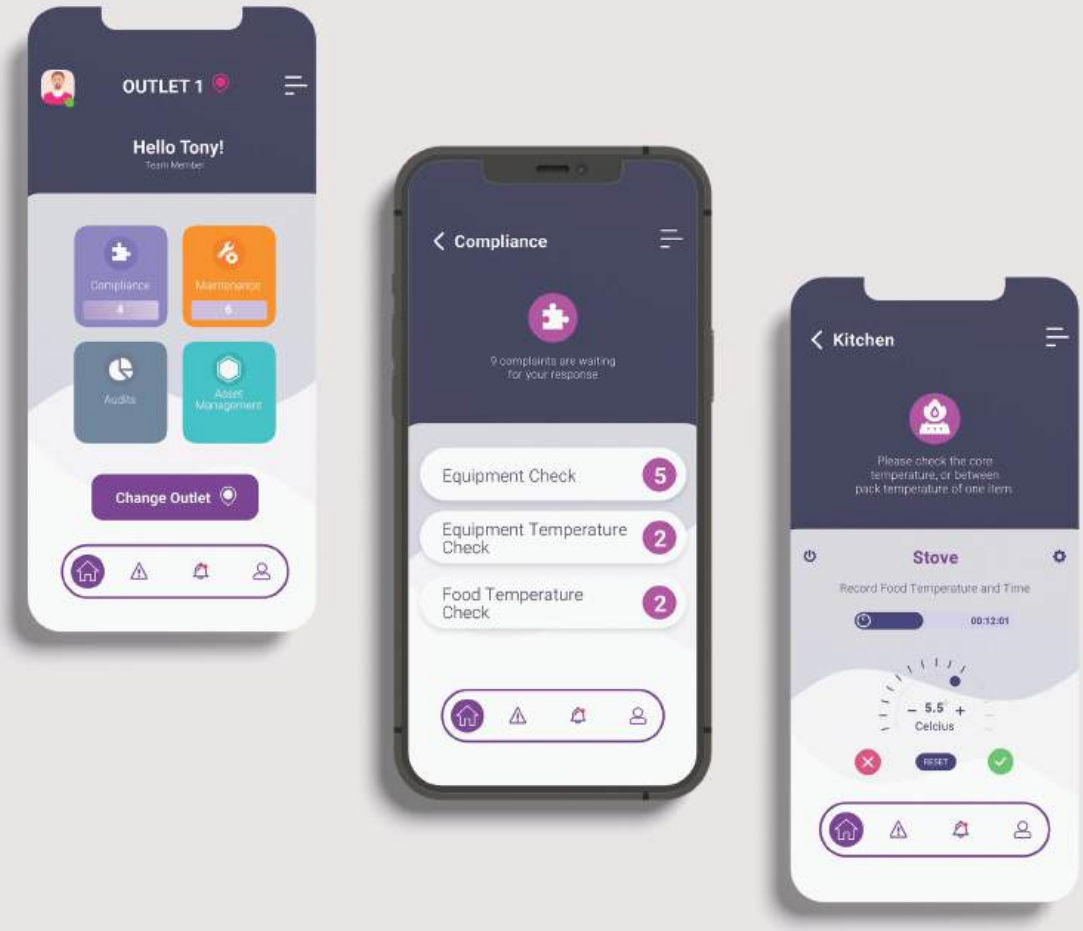
To handle a lot of administration purposes, the architectural decision made was to use “Shared Database Separate Schema”.

Excellent Report generation

The success of GRC project depends on creating a readable and comprehendible report at the end of the execution. To facilitate an innovative report generation module was build which is configurable at various levels. Users of the system can create templates of the reports and use the same while generating the end report. This allows organizations to standardize the language and format of the report while keeping the template dynamic enough to create reports for different projects.

Delivery

One of the needs of Cyraacs was to build a delivery pipeline which is agnostic of the cloud. Meaning, Cyraacs wanted to have the freedom of deploying this product either on on-prem servers or on the cloud.
A unique engine was built to automate and facilitate the build and seamless release of features. Given that the releases were being implemented in realtime it was imperative that a robust testing framework was established to meet the pre-defined quality parameters. Leveraging the vast in-house product design and testing experience, Aritha ensured high-quality releases by pre-determining the critical paths



The ability to roll back releases if situation warrants, is enabled by the design and technology adopted by Aritha. This helps safeguards data integrity, minimizes data loss and provides for consistent user experience.

Technology Stack

Development	ReactJS, Java, MySQL, Kafka
Third party integrations	Stripe, Razorpay, Twilio
DevOps	AWS, Jenkins