Alignment of Architecture and Research

Alignment within the service team

1. Worked closely with service design and the research teams to fully understand the vision and end to end service.
2. Aligning the Reusability of the core components with the service including understanding the dependancies and features
3. Understanding where requirments fall out of the technology patterns being deployed.
4. Faciliate communication between teams
5. Reusable web component strategy
6. Raising with other delivery teams requirments to support our service
    - Example being the common component for order creation added to the JUI backlog to support Public Law.
    - Document generation

Solution Architect Role

1. Deliver architecture in an agile environment
2. Own the solution and create the HLSA containing key design decisions and from which the development team will build a solution
3. Produce logical software designs for the development team and work closely with them
4. Design APIs (typically RESTful) for public and internal consumption
5. Define how the service will integrate with legacy
6. Bring key design decisions to the Governance process
7. Define the NFRs for the service project
8. Produce the deployment architecture.
9. Support the development team (design patterns, test approaches, tooling and SDLC pipeline)
10. Contribute to central design work - particularly re-usable design patterns
11. Carry out reviews of compliance against platform architecture

Reform Arcitecture Landscape Reuse

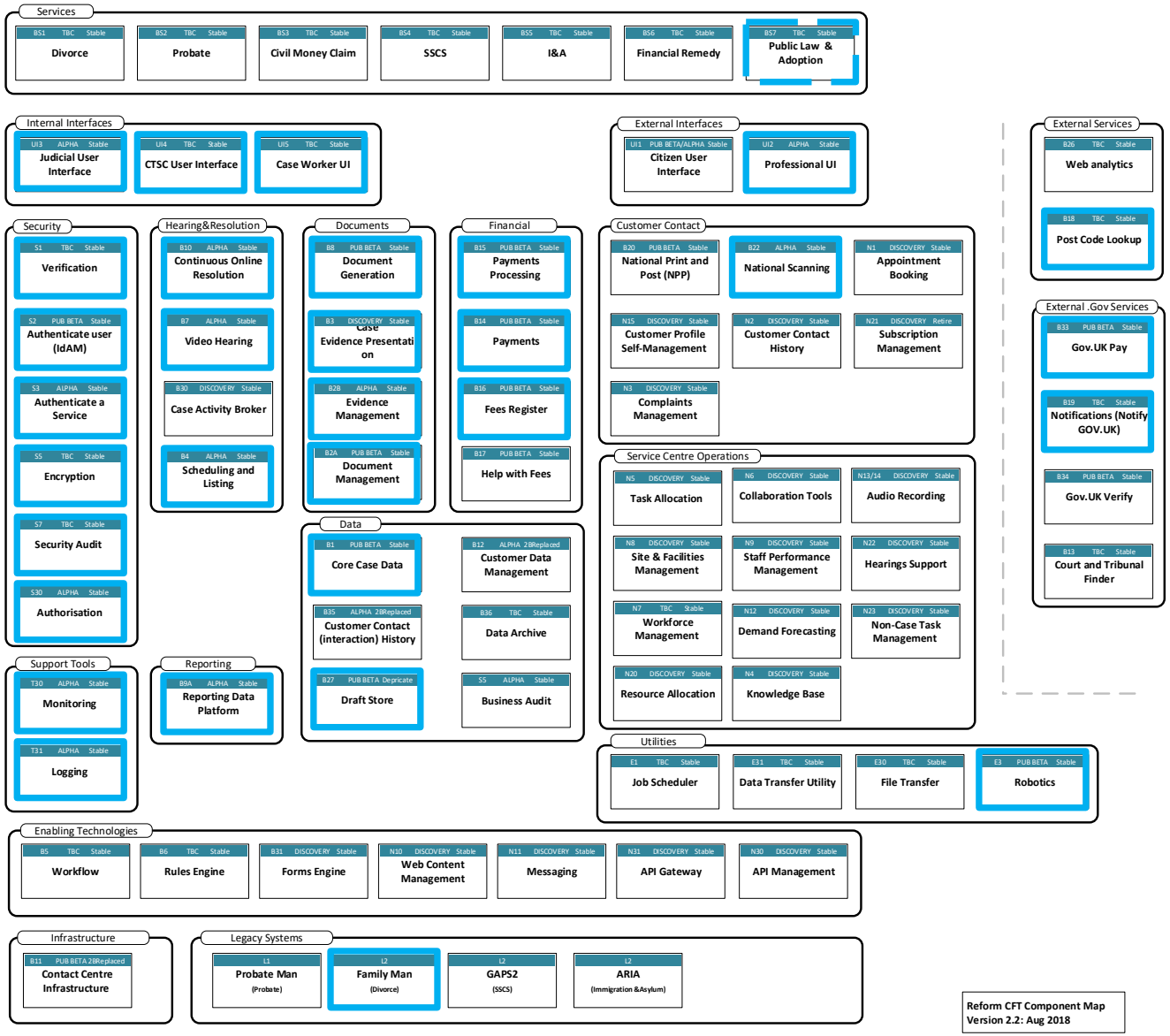
Key:

This

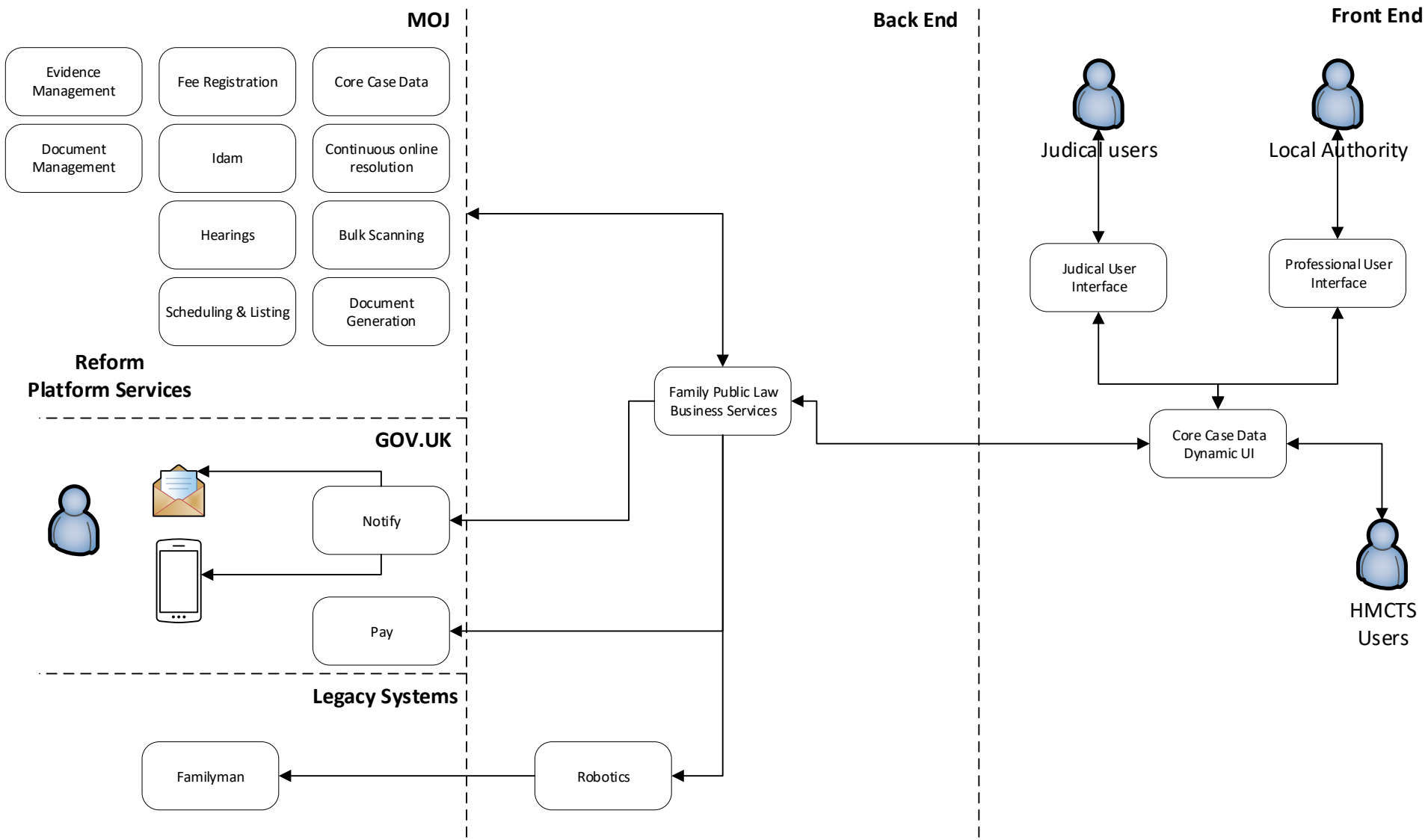
Used

Future

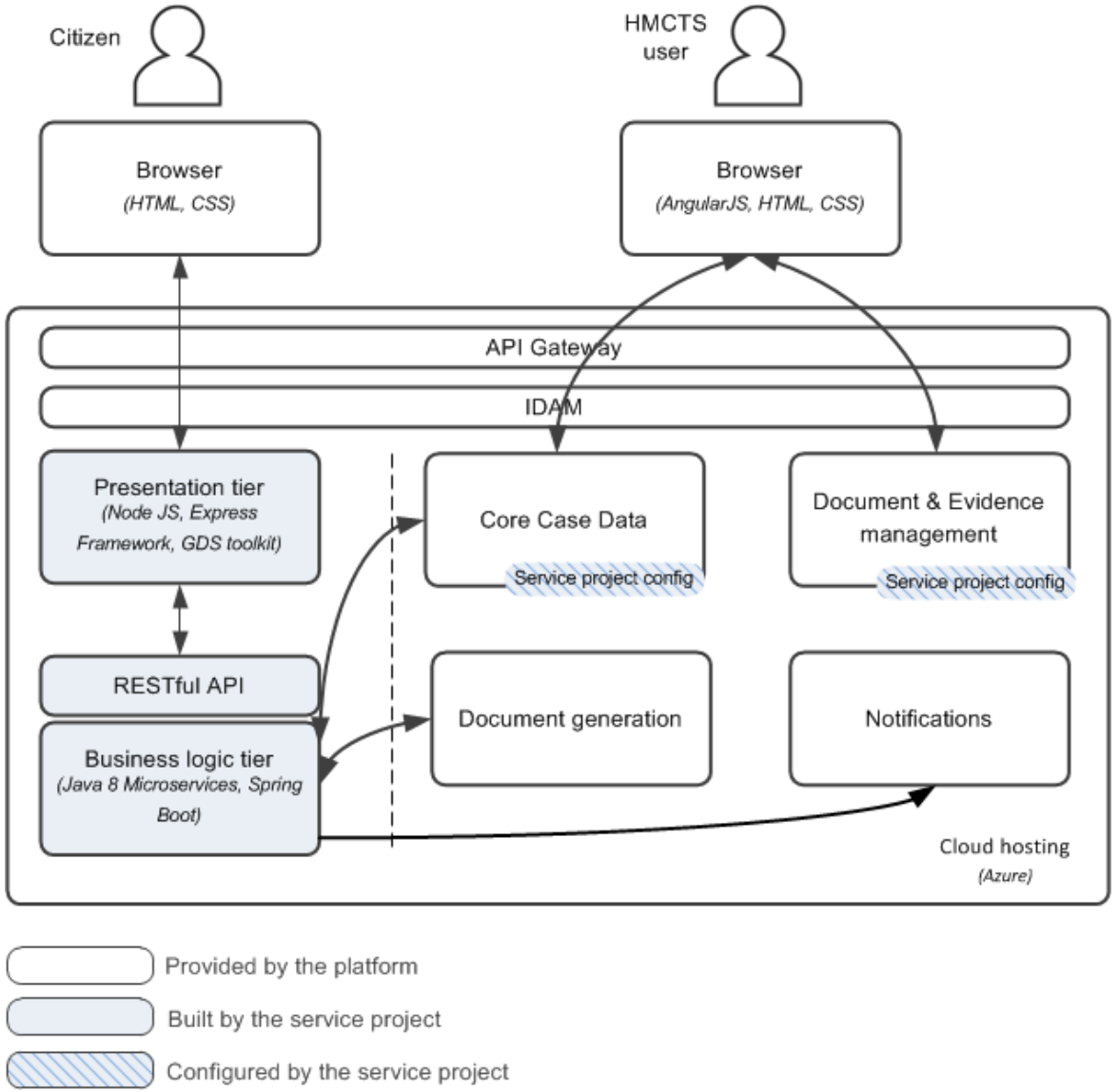
Removed



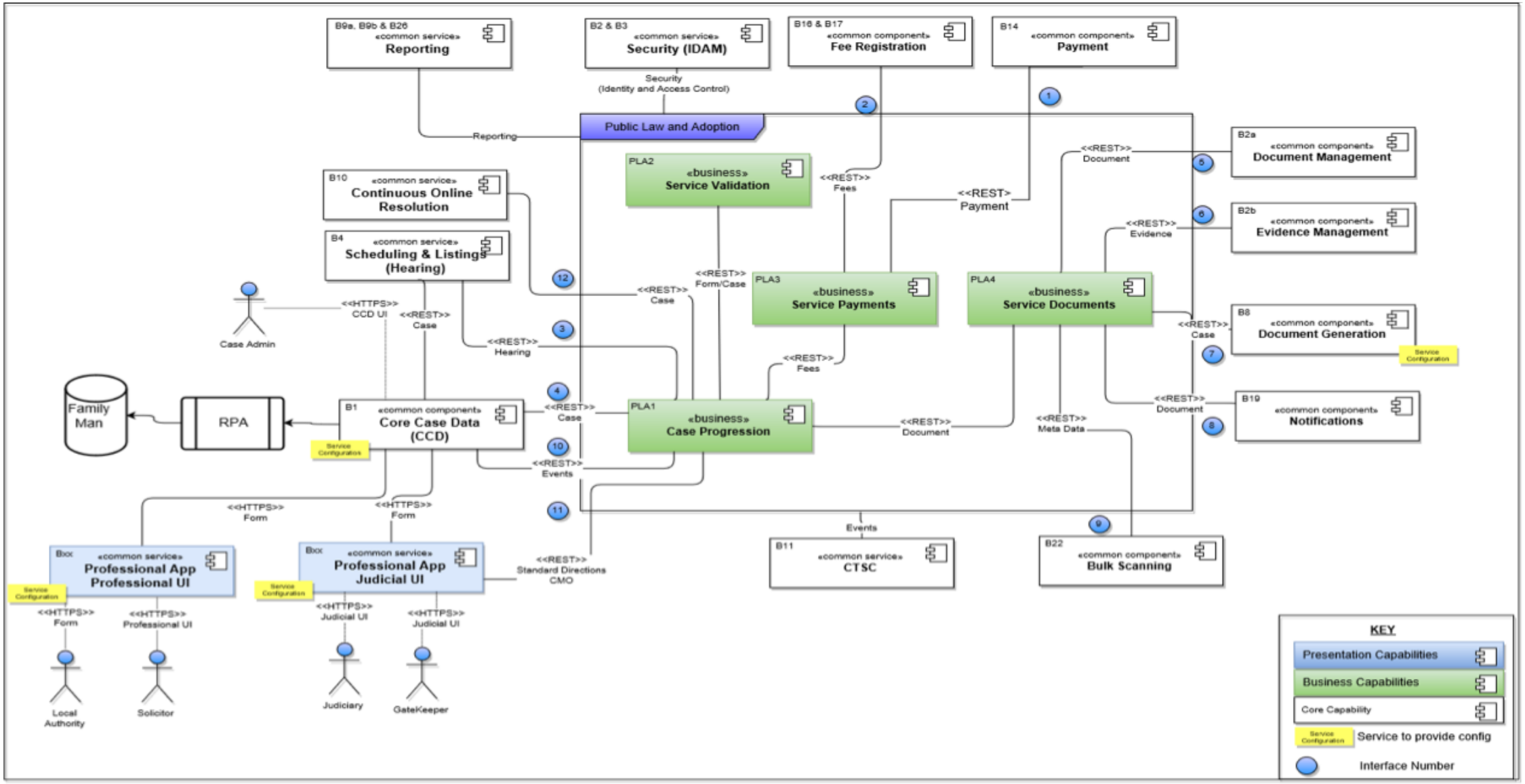
Architecture – Private Beta



Architecture – Reform Approach



Architecture – Overall Vision



Technology Risks

Show there's minimal risk associated with the technology you chose

The 2 primary programming languages chosen are Java 8 and NodeJs. Both are mature languages with an active support community and a large pool of resources to draw on. Spring Boot and Express.js are frameworks chosen for the respective languages and each are very popular and actively developed and maintained. Infrastructure runs on Microsoft Azure, and is created using HashiCorp Terraform, an OpenSource industry standard tool.

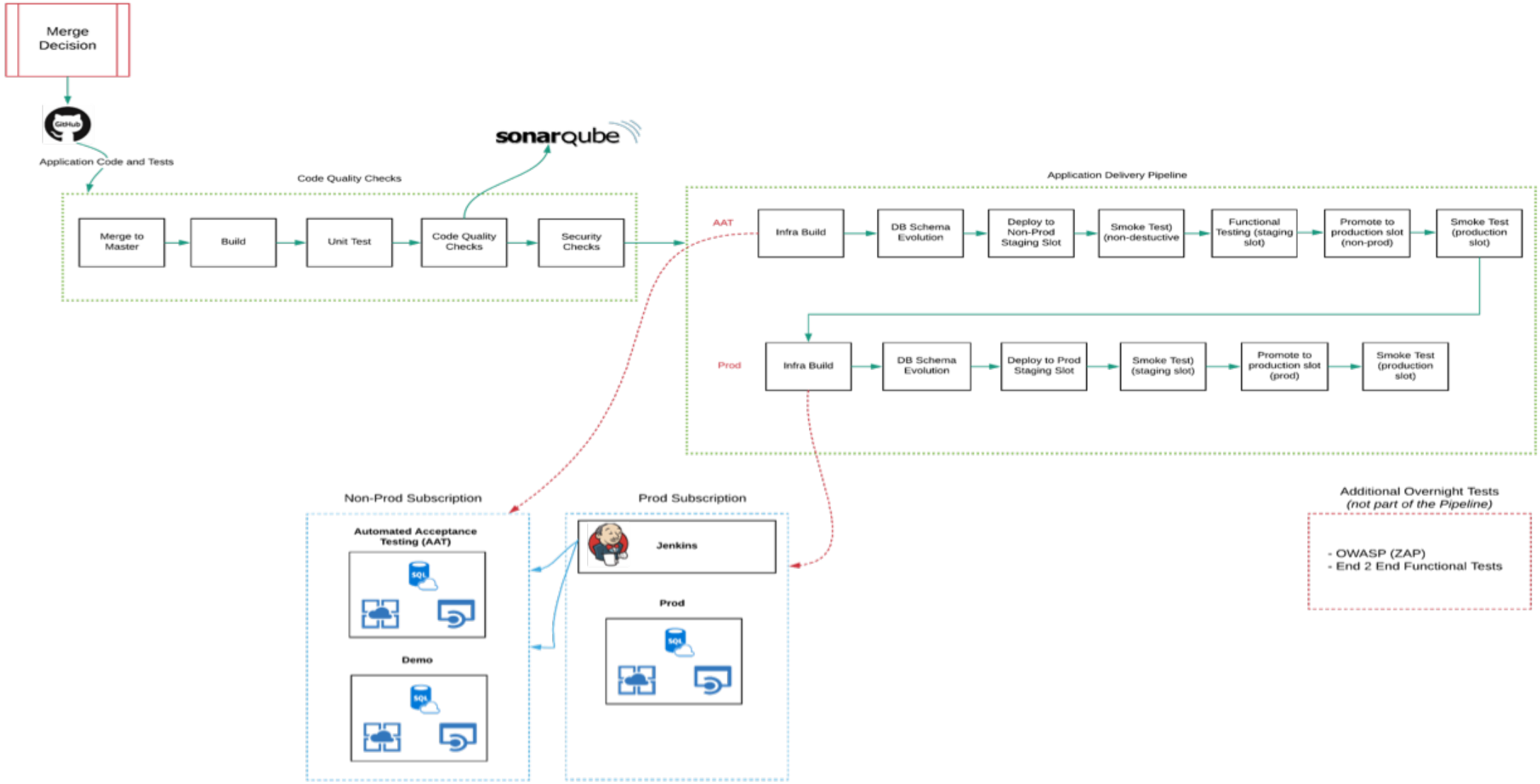
Prove you have the ability to deploy software frequently with minimal disruption to users

We will use a continuous integration and deployment pipeline which deploys code to production upon merge into master branches. The specific technology used to enable zero down time deployments is Azure Deployment Slots in the App Service Environments.

Describe your way of deploying software, i.e. how you can deploy frequently with minimum impact on users

Having performed all relevant testing and quality checks we will deploy using blue / green deployments switching between the environments and ensuring full user access. Should anything happen out of the normal controls a maintenance page will be displayed to ensure against data loss.

Deployment Pipeline



Deployment Pipeline

Demonstrate that you have an effective deployment environment

The programme has created a library of code (Terraform based) that builds consistent immutable environments on Azure. This ensures and enforces identical and consistent infrastructure throughout software development from sandbox, through testing, to Production itself. An automated toolchain (Jenkins) moves application code through a series of security, configuration, and smoke tests; builds infrastructure; and deploys. Pre-prod and Production are “zero touch” environments, eliminating errors and issues stemming from manual configuration and intervention.

Show that you can create new environments quickly and easily

Using Infrastructure as Code automation, the programme has developed the capability to create full environments from an empty Azure subscription in around 4 hours, and individual development environments in a little over an hour. this is orchestrated by HashiCorp Terraform, an open source language that defines the configuration of virtual clouds, and then builds an execution plan to create the infrastructure.

Tech stack

Frontend
NodeJS
ExpressJS
Nunjucks
Angular 6

APIs
Java 8
Spring Boot

Automation
Terriform
GitLab
Gerrit
Artifactory
Jenkins
Docker

Testing
REST-assured
Pally
Sinon / Mocha / Chai
JUnit / Mockito

Infrastructure
Azure
CNP platform

Databases
PostgreSQL 9.5