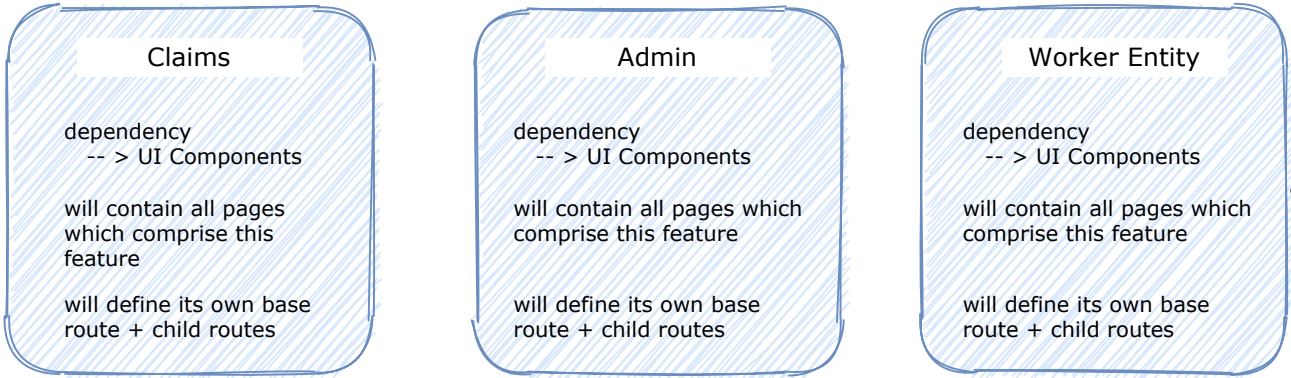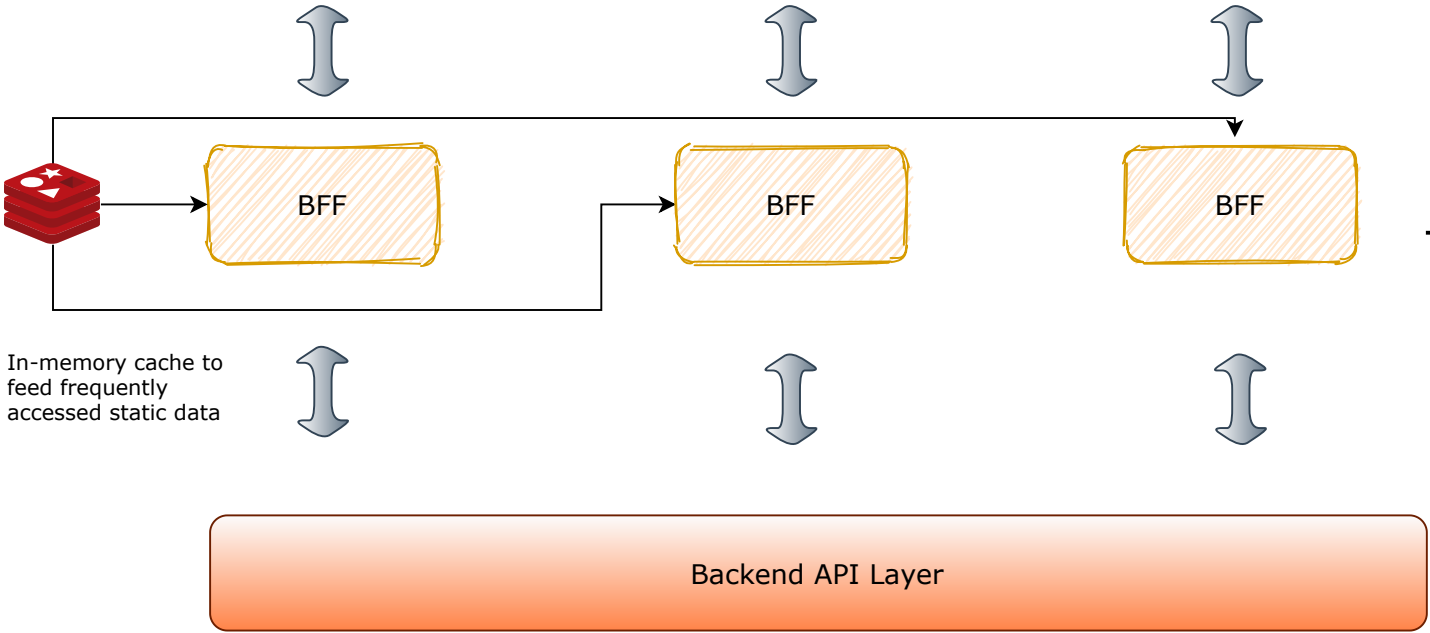**UI Components**

- button
- select
- card
- grid
- etc.

developed as a NPM package with its own dependencies (Angular library application)

version managed with its own package.json

not a deployable unit

will be imported by the layer below

will not communicate with backend or CMS. data will be fed in by consuming parents and events emitted out
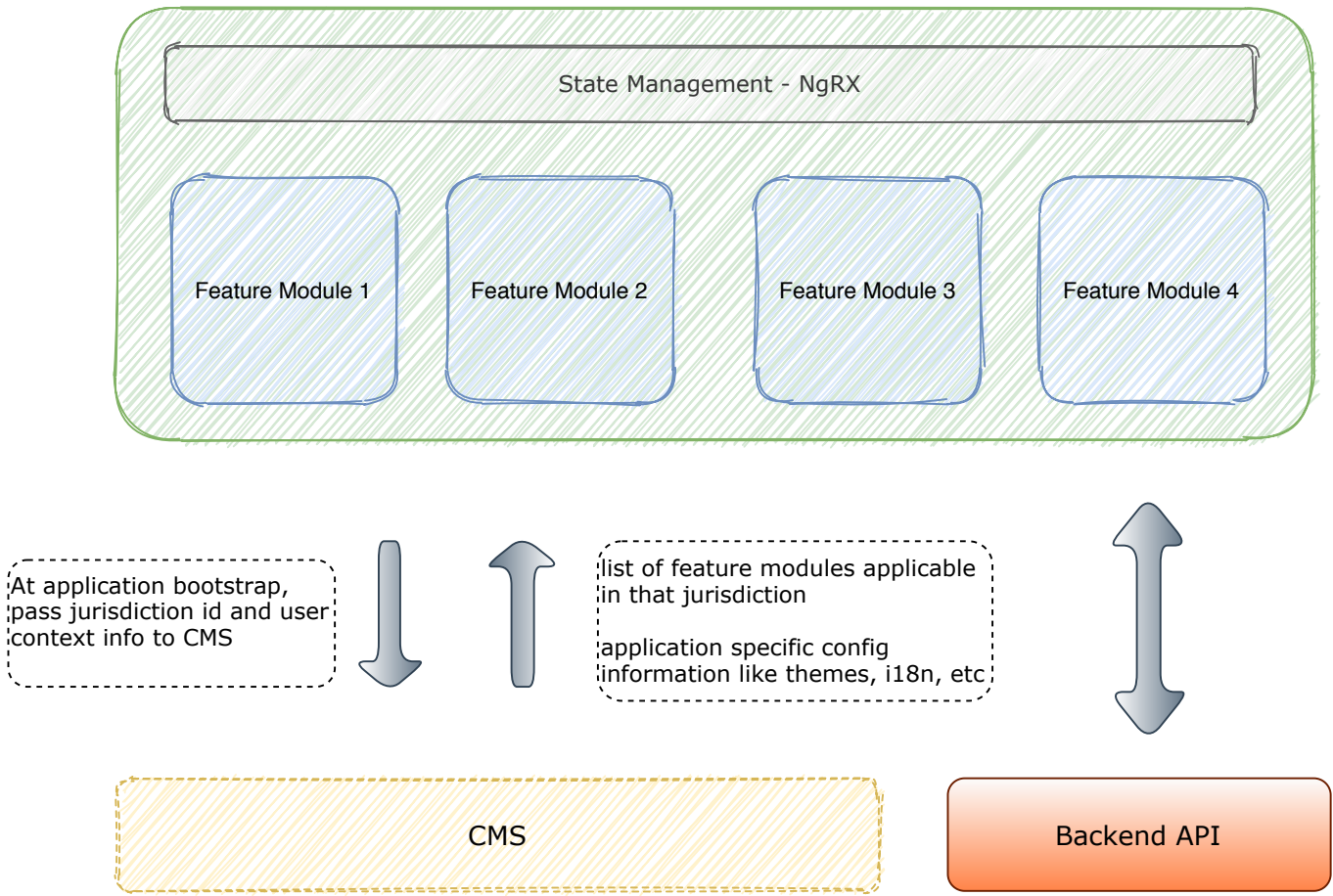
Feature Modules

Claims

dependency
-- > UI Components

will contain all pages which comprise this feature

will define its own base route + child routes

Admin

dependency
-- > UI Components

will contain all pages which comprise this feature

will define its own base route + child routes

Worker Entity

dependency
-- > UI Components

will contain all pages which comprise this feature

will define its own base route + child routes

there will be ONE Angular application for EACH feature catering to all jurisdictions

each feature will communicate independently with API layer

each feature module will fetch its own config info from CMS

app context info like jurisdiction and user roles will be available to each feature module via common application state layer (refer to main app below)

version managed; independently testable and deployable

BFF

BFF

BFF

each feature module can have its own BFF layer or there can be one common BFF layer feeding all feature modules

.NET Web API application

Since BFF layer will be hosted in close proximity of backend API server, any n/w induced delay will be minimal

In-memory cache to feed frequently accessed static data

Backend API Layer

Main UI Application

State Management - NgRX

Feature Module 1

Feature Module 2

Feature Module 3

Feature Module 4

main deployable UI application, single install, common across all jurisdictions

each feature module will be installed as a NPM package and will be imported by main app module

all feature modules will be lazily loaded, apart from those required at application bootstrap (example - core & shared)

any feature module can be individually upgraded without affecting overall application deployment

application routing will be setup at bootstrap to enable only enabled features

main UI application will communicate with backend API for cross - cutting concerns like security, logging

At application bootstrap, pass jurisdiction id and user context info to CMS

list of feature modules applicable in that jurisdiction

application specific config information like themes, i18n, etc

CMS

Backend API

Individual repos

- UI Component
  - each component will be a publishable NPM package scoped to this repo (i.e. @ui-component/button, @ui-component/grid)
- Feature Module
  - will have multiple publishable Angular apps within this repo
  - this will ensure that each feature module will be a scoped package (i.e. @feature-module/claim, @feature-module/admin)
- Main UI Application