MicroFrontend Architecture for Mobile

Pitfalls of traditional approach

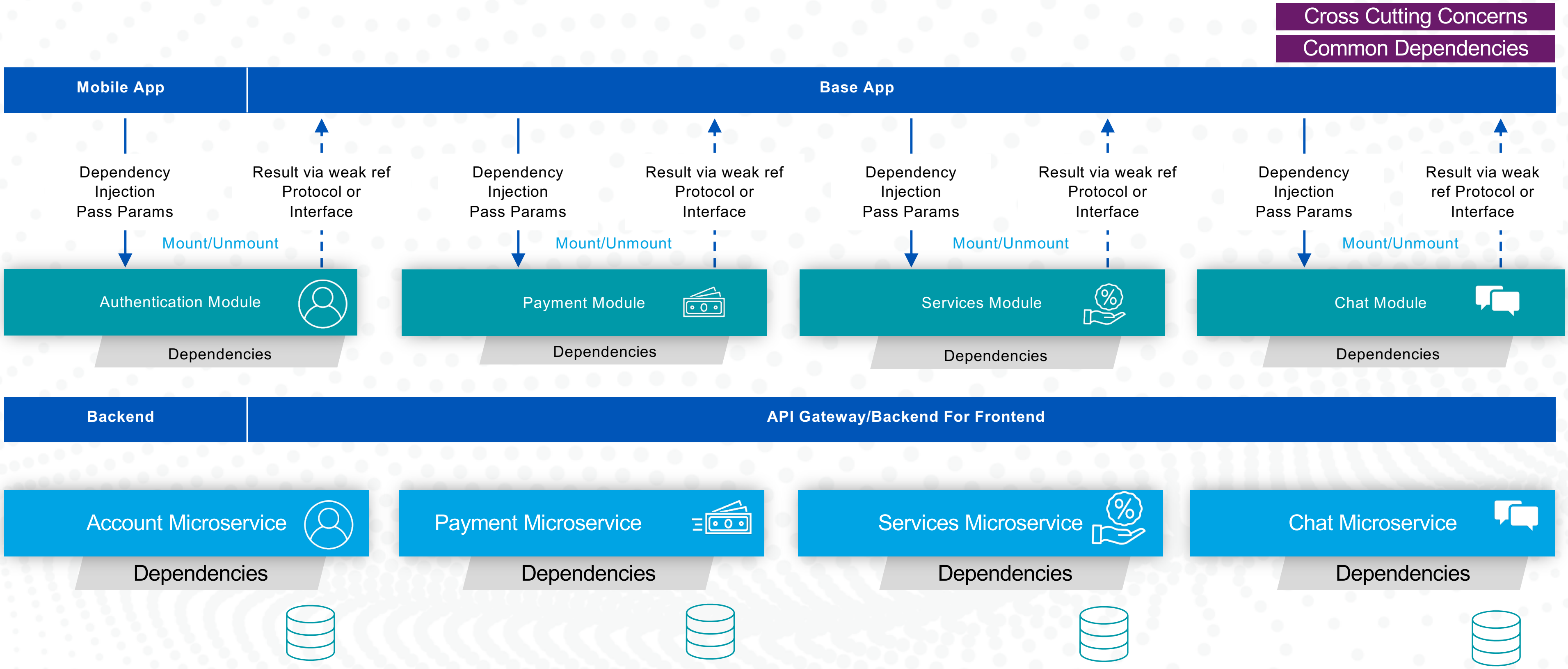
- Apps is slow and not responsive – silent failures/crashes as a result of monolith loading for larger apps.
- Native Modularization is not completely leveraged
- Scaling the teams will be an issue – two team, two platforms, two architectures
- Large code base; tough for developers and QA to understand the code and business knowledge.
- Tight coupling between components, as everything is in one namespace.
- Less Scalable; Less Reusability

Benefits of MicroFrontend

- Smaller, reusable, more cohesive and maintainable codebases
- More scalable organizations with decoupled, autonomous teams
- Ability to upgrade, update, or even rewrite parts of the frontend in a more incremental fashion than was previously possible
- Faster build time
- Being technology agnostic
- Testable compliant code
- Time to Market

Confidential: By Saad Shams

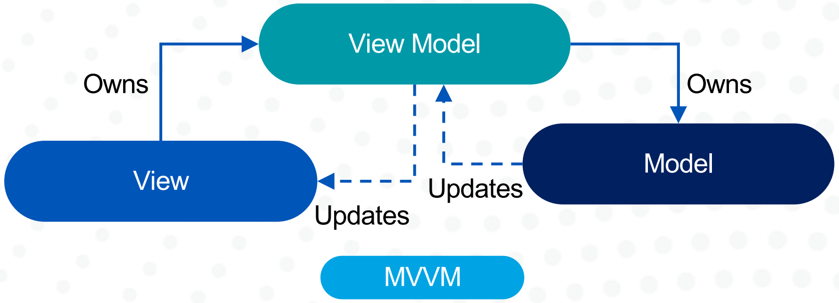
End-to-End Architecture – MicroFrontend for Mobile



Confidential: By Saad Shams

Module Reference Architecture – MicroFrontend for Mobile

Mobile Application User Interface/Experience Layer



Base App: Cross Cutting Concerns

- Notification Management
- Network Management
- Log Management
- Exception Handling
- Analytics
- Security

Micro Front-end Module

Minimal Client-Side Code

- Validators
- Client-Side Logic
- Parsers

Reusable Components

- Widgets
- Templates

Data Persistence & Data Access

- Keychain/KeyStore
- External Data/Content
- Data Store

3rd Party Libraries & Capabilities

- Lib
- SDKs

Native Platform

- Native APIs
- SDK

Confidential: By Saad Shams