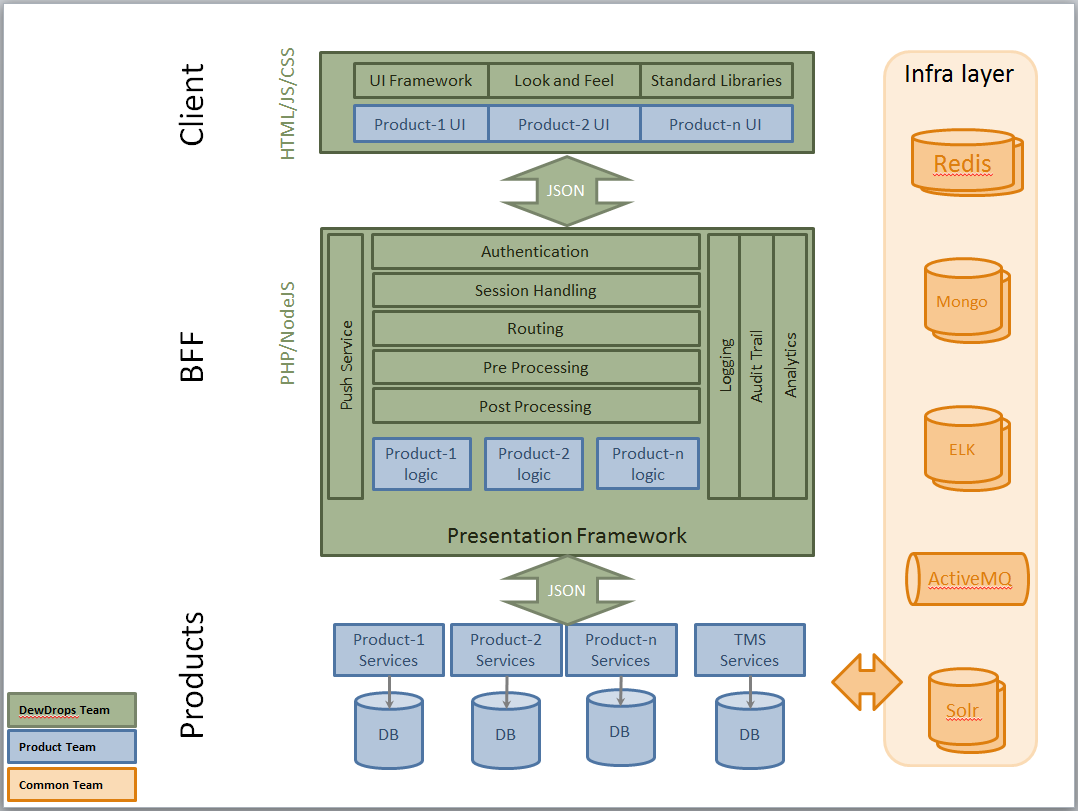
Introduction

DD is an initiative to build a unified user-interface and user-experience spanning across entire product suite. DD would be a single frontend to interact with all the individual Products. DD intends to take the user-experiences to next level after Rainbow.

# System Architecture

In this section we would discuss the overall architecture of the system. Detailed design of each layer would beseparately documented. The below sections gives you a high level view and pointer to certain need to be taken care in detailed desings. 

## UI layer

The UI layer would be comprised of the following for web-applications:

1. UI Framework: AngularJS, DewDrops JS files
2. Look and Feel: Stylesheets, images, SVGs, Fonts
3. Standard Libraries: 3rd party UI Plugins and widgets

Whereas for Mobile it would be:

1. Native UI developed for Android and iOS
2. Responsive pages from web application (future use)

This layer would be implemented as a hybrid combination of SPA (Single Page Application) and MPA (Multi Page Application). All screens from a single product should be clubbed together as a SPA. All these SPAs would collectively form a MPA i.e. switch over across products would be full page transformation. This would help in arranging and managing the products for customers who do not subscribe to the entire Zycus Suite.UI Layer should be HTML5 supportive and should make use of *sessionStorage* and *localStorage* wherever deemed appropriate.

Diligent decision should be taken about the data being stored locally, strongly considering the data security aspects.

#### UI components

DewDrops should attempt to provide common UI components for use in product specific pages. We may use libraries like Polymer (<https://www.polymer-project.org/1.0/>) for developing the components. This would help in facilitate reuse and standardisation. This layer talks to the [BFF layer](#_Backend_for_Frontend) using JSON.

## Backend for Frontend layer (BFF)

The DewDrops BFF layer would facade for the various products that would expose their business functionality over ReSTful webservices.

This layer should provide the below features:

1. Authentication: against TMS
2. Session Handling: Using Redis cluster
3. Provision for Pre Processing
4. Routing to product pages
5. Provision for Post Processing

This layer should provide a framework with additional features like:

1. Logging: using the ELK stack
2. Audit Trails
3. Analytics (using ZyTrack or integrate with any 3rd party tool)
4. Push Service

For logging we can use the ELK stack (Elastic Search, Logstash, Kibana) to help us with logging monitoring and quick search of logs during issue resolution. All the request and response could be logged in ELK. We should also add a unique Request-ID to every request coming in and correlate it to its response. This would help in quick fetching of related statements from ELK.

**NodeJS** would be a good choice considering that this layer would not have any business logic. Also since the preferred format of data exchange would be JSON, NodeJS would be a better choice considering it being designed for bring more suitable for systems having simple and large number of request. **ExpressJS/sails** would provide the web application framework required for the BFF.

Highlights for NodeJS:

1. **Non-blocking I/O**: required to call multiple products in parallel, if required. Additional features like Logging, Audit trails and Analytics can be done without impact on response time
2. **JSON**: NodeJS having inherent support for JSON would be added advantage to work on incoming and outgoing JSON objects to [UI Layer](#_UI_layer) and [Business Layer](#_Business_Layer)

However, choosing NodeJS would add a learning curve. NodeJS could be better suited for employees migrating from PHP to Angular/NodeJS.

**TODO**: All the above mentioned features need to be elaborated further.

The BFF framework would provide a modular structure so as to enable us to implement product & version specific logic.

The mobile AppServer would act as the BFF for Mobile Apps.

## Business Layer

Business layer for DewDrops is to be provided by utilising the existing business layer of the Zycus’ products. The layer should expose **stateless** ReST-ful web services to satisfy various data and functionality needs if the Presentation Layer. The Request-ID from the presentation layer should be carried forward throughout this layer and used for every logging. This would further help in deep-diving to resolve issues.

Business-layer (products) should be load-balanced using HA-Proxy, so that multiple nodes of product servers could be used for High-Availability and failover. The HA-Proxy should be configured to appropriately redirect the request to particular version of the specific product being used by the tenant. The section [Deployment](#_Deployment) sheds more light on this topic.

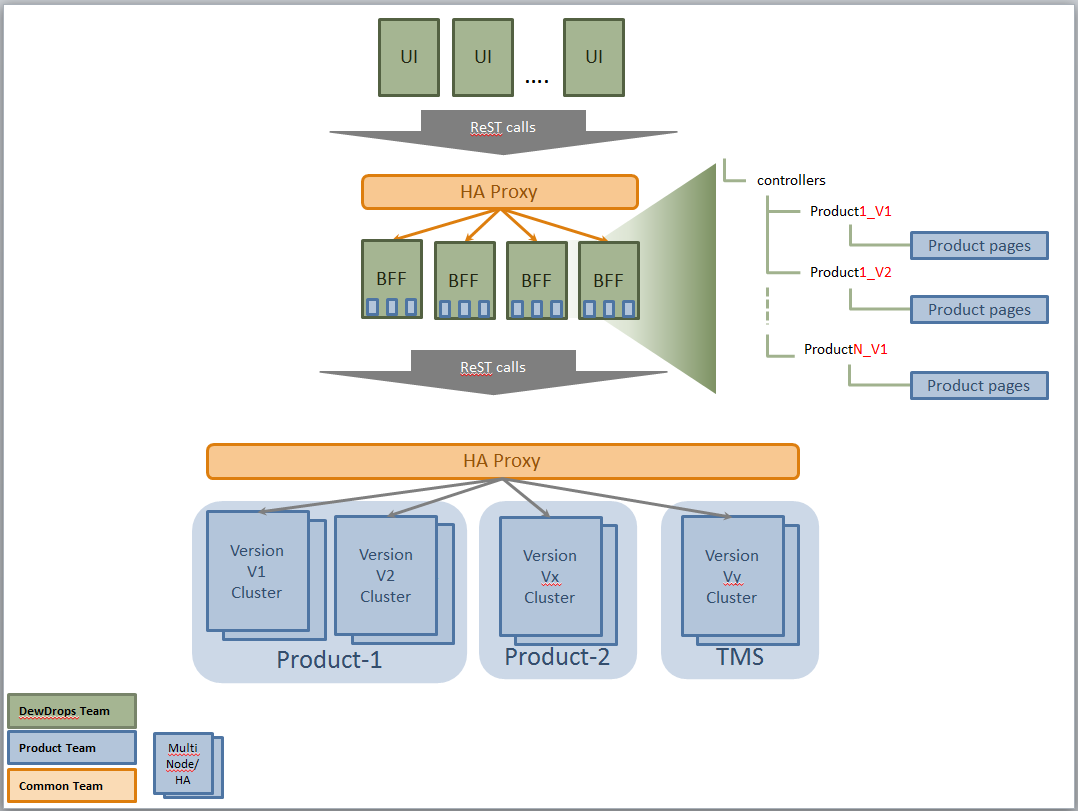
#### ReST-Ful webservices

This section describes the standards and guidelines to be followed by product specific teams. The interface should comply with guidelines published by iConsole to avoid rework. It is currently being handle in a separate document and would be merged here in future

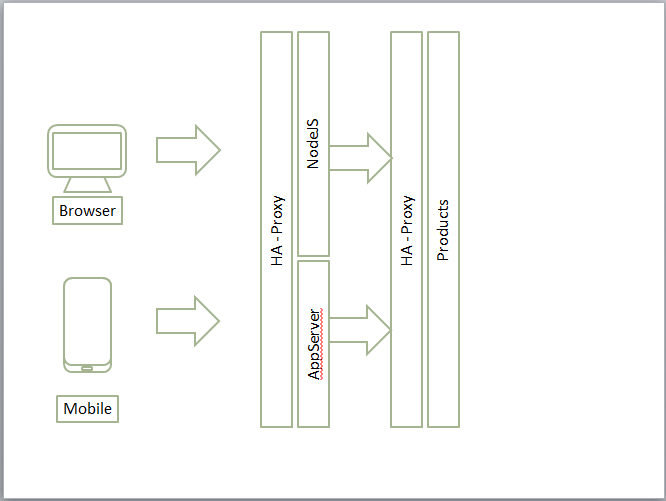
# Deployment

Some of the products have tenant specific deployments. We need to take care that these versions are correctly accessed and maintained. The [BFF](#_Backend_for_Frontend) frame work should have product and version specific logic so that the routing feature of the framework correctly forwards the request to its handlers. The product and version specific handlers would be having version specific URLs configured of its back end Java services. Hence the routing would be correctly handled.

Load testing has to be performed to find out the number of DD installations (Nodes) required. Diagram below depicts 4 nodes, but could be more actually. The DD nodes are load-balanced by HA-Proxy.



Mobile Apps would also be part of the DewDrops initiative.



# Challenges and Mitigation

The overall ideology and scope of DewDrops brings to us certain challenges.

|  |  |
| --- | --- |
| Challenges | Mitigation |
| 1. **Load**: As all the requests that were earlier handled by various installations of products would now need to be handled by DewDrops BFF | DD need to be cluster enabled for loadbalancing and scalability. DD installations should be done on VMs/Containers to enable adding more nodes to the cluster as an when required without any downtime |
| 1. **Releases**: Every product would have its own roadmap and commitments. Deployments should be smooth without having a need for downtime | Server Farming: New releases should be deployed on few nodes. Only after satisfactory verifications and sanity it should be rolled out to other nodes. Also, We could have the older version running for 15 days on some of the nodes to handle extreme cases of emergency rollbacks. Here the traffic could be routed to older version for the affected tenants. |
| 1. **Infrastructure**: Infrastructure support is imperative to handle the smooth functioning and uninterrupted client experience | Management needs to allocated budget for the infrastructure. DevOps practice should be implemented in line with the business demands. Support teams need to be well trained to handle |
| 1. **Development**: Development of new release and DB alterations need to be consider the points above and enable smooth fallback to older versions | RM process to be amended accordingly to verify these capabilities. Setup a control board to verify changes or publish guidelines |

# Testing

All the communication between layers (Client-BFF-Business) is based on JSON structure. Each layer should maintain these service contracts in Swagger (<http://swagger.io/>). This would help in testing each layer individually and also facilitate in further automation of the test cases. Test Strategy is being documented separately.

# Team Structure

teams would be a multi-location team spanning all three development centres: Mumbai, Pune and Bengaluru. People from existing products’ presentation layer would be working on product specific screens in DD. This would help in utilising their functional knowledge and technical knowhow about the product.