Hyper Personalization with AI based Dynamic UX

Srikanth BR[1] Jithesh Sathyan[2] Sanil Das[3] Arvind Dinakar[4]
[1,2,3,4]Digital Experience Unit, Infosys Limited

Abstract— Current web and mobile applications are designed to follow a specific screen flow to complete an activity. The screen design is done with a workflow in mind, that satisfies a business requirement. Hence in design, a fixed set of wireframes are modelled and implemented as screens in the application. Each customer is unique and follows a pattern in using the application. The next generation user experience should be based on hyper personalization. The generic app should dynamically adapt the screen and information presented to specific user needs and changes based on user interactions by collating and processing the relevant data to perform a real time experience design. This should be done with due consideration on data privacy. This paper proposes a technique to dynamically update user experience. The dynamic update will be based on triggers for quick dynamic changes and also does intelligent learning and feedback loop, to slowly adapt experience for the specific user. This paper defines the logical architecture for the next generation dynamic user experience based on several patterns that leverage Analytics, AI and IoT technologies. Use cases are presented from enterprise and consumer context. The paper also details the interactions with enterprise systems and gives an overview on the development of features. The paper concludes with a discussion on the scope of future work, after detailing key observations from actual implementation.

Keywords: User Experience, Analytics, Machine Learning

I. Introduction

Most of the current web and mobile apps, have analytics driven content update. However, analytics data is used for improving the content feed, and not to change the frontend experience. Dynamic adaptation of frontend experience is not being done to improve efficiency of end user to perform a routine operation. The current experience design is to have "One static application", to cater to many customers who are unique in their pattern of using the application. This results in reduced efficiency and sub-optimal experience, when the app is unable to adapt to user. This problem can be addressed by creating an application that adapts itself to specific context and user needs offering hyper personalization.

The next generation dynamic user experience focuses on a single app that adapts itself to specific user needs and changes based on user behavior by collating and processing the analytics data to perform a computational real time experience design. Experience change is based on Usage pattern (Rearrange dashboard and items in dashboard, and create shortcuts based on browse like frequent visit, order placed, price choice, etc.), Experience pattern (Continue from where user left with switch of device, load template for weekly shopping, user rearrange app look-n-feel etc.), Location pattern (Load capabilities, change look-n-feel, and load relevant content based on location), Context pattern (Proactive alerts like price drop, offers on items of interest, etc.), Domain pattern (like season/ daily weather conditions for products displayed, role based like prime vs regular customer on offers, etc. for retail app) and Learning (based on machine learning analysis of user activity over a period of time) .

II. Current Solution

The analytics data which are captured from the mobile and web apps, are generally used for identifying user's behavior, application performance, and to change the content which is to be displayed to the user. Though application monitoring and usage pattern tracking is done, it done not make any dynamic change in the overall application experience. There are no shorts cuts created for frequent used functionality, no new widgets to simplify an operation or other user interface elements introduced that is adapted to user needs. The current solutions are focused on offering a single app with a predefined experience to the user.

- The adaptation available in current solutions is limited to:
- Changing content that is customized to the user shown on a UI element.
- Selectively hide or display UI elements based on pre-defined rules.

Customized content is offered by most marketing solutions. Selective hide and display would require planning the screen and workflow and loading a lot more UI elements than what is required. Both these solutions are not satisfying the next gen experience requirements of dynamic user experience.

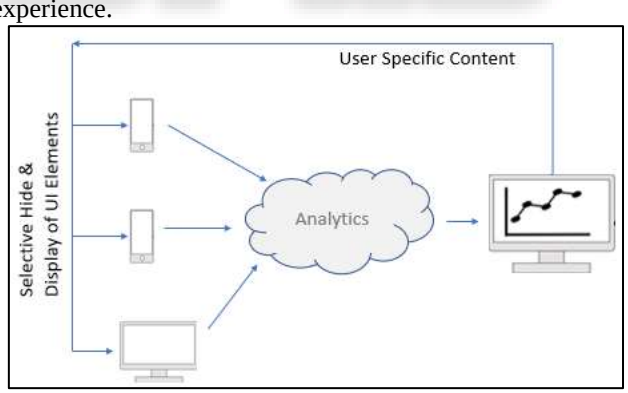

Fig. 1: Current solution data usage.

III. PROPOSED NEXT GEN DYNAMIC UX SOLUTION

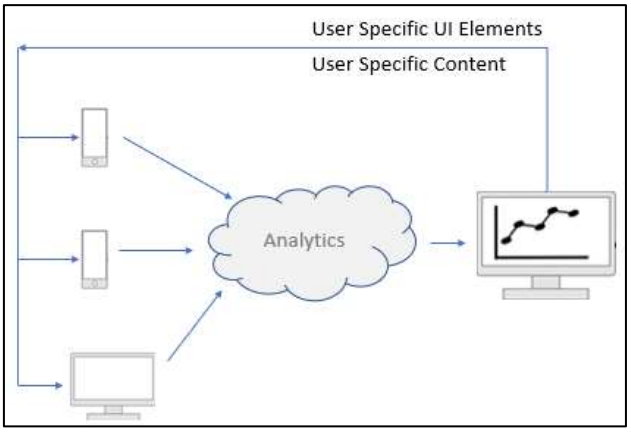

Fig. 2: Proposed next generation dynamic UX solution.

In the proposed next generation dynamic UX solution the analytics data which is collected from all the applications is processed and used for changing the application flow based on the user behavior and preferences. This will improve the user's experience and the user will find that the app adapts to the user with usage.

The high-level logical view of the proposed solution can be divided into four layers

- Experience layer
- Platform layer
- Project custom layer
- External system layer

A. Experience layer

This layer mainly has two modules, one is the client app and the other is the dynamic UX studio. Client app – to the client app the front-end developer has to add an adaptor or a container to which the dynamic component gets created and added dynamically. Dynamic UX studio – this is the portal for the admin or the front-end developer to create the triggers for each screen.

The Studio will specify the trigger and action which is used to model a screen on the fly based on computational design. This will have two interface one for adding the triggers which will be done by the admin and one for mapping the trigger to the screens which will be done by the business team. The basic information needed for an entry are Screens, Triggers, Actions, UI Elements and Containers. The table shows a typical entry in the Admin portal for displaying work orders assigned to a field agent. Based on the condition satisfied, the appropriate trigger, will result in dynamic loading of the mapped UI elements.

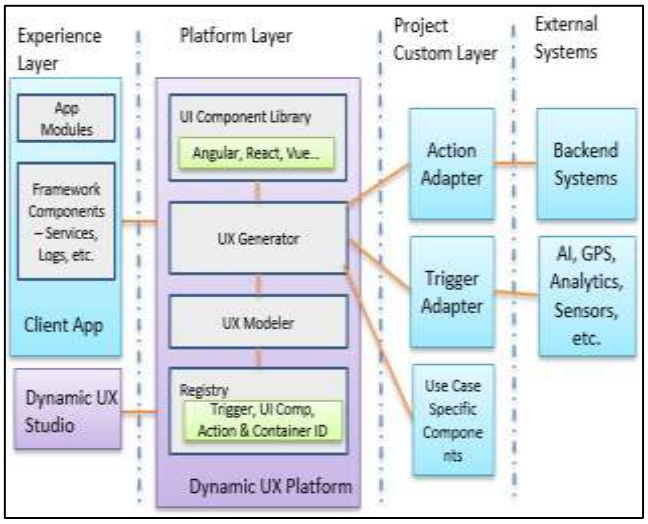

Fig. 3: Solution logical view

UC	Triggers	UI elements	Actions	Screen & Container
Work Orders Priority	Work Order less than 6	Cards in grid view	Show the work orders in grid view sort by the priority of the orders	Work Order Screen – TopContainer1
	Work Order Screen opened with the tasks more than 6	Cards in list view	Show the work orders in list view sort by the priority of the orders	Work Order Screen – TopContainer1
	High Priority Tasks	Alert box with Button to get to details screen	Show the work order details in the main dashboard	Home Screen – TopContainer2

The use of Studio is required only when the application does not have self-learning capability which can be implemented using machine learning.

B. Platform layer

This is the platform that we create for bringing the dynamicity to a new or existing web application.

This layer mainly has four modules,

- UI Component library
- UX Generator
- UX modeler
- Registry

1) UI component library –

All the generic custom (React or Angular or Vue) components are available here for the front-end developer to use. Custom modules will be created using these components and these are imported at the frontend and thus importing only the components that we want to use for each screen.

2) UX Generator –
This return the JSON which has the information to dynamically create the UX for a specific screen. It receives the request from the screen, the adapter process the request and gets the data from the DB.
3) UX modeler –
Based on the data entered in the dynamic UX studio, a file is generated which defines the dynamic experience to be created and the triggers that initiates the change. This file will be used by the UX generator.
4) Registry –
Has a unique identifier for each trigger UI component, action and container in which the dynamic change is to be loaded as a result of trigger.

C. Project custom layer

This layer will have all the business logic which will set or reset the trigger based on the criteria that the admin or the frontend developer provides. The adapters will be used to interact with the analytics systems and also with the customer backend systems. Based on the functionality the adaptor can be classified as action adaptor and trigger adaptor.

D. External systems

The external systems like the existing backend system and the AI, Analytics, Sensors, GPS systems interact to the dynamic UX generator platform with the help of the adaptors

IV. ADDING DYNAMICITY IN A MODULAR LEVEL

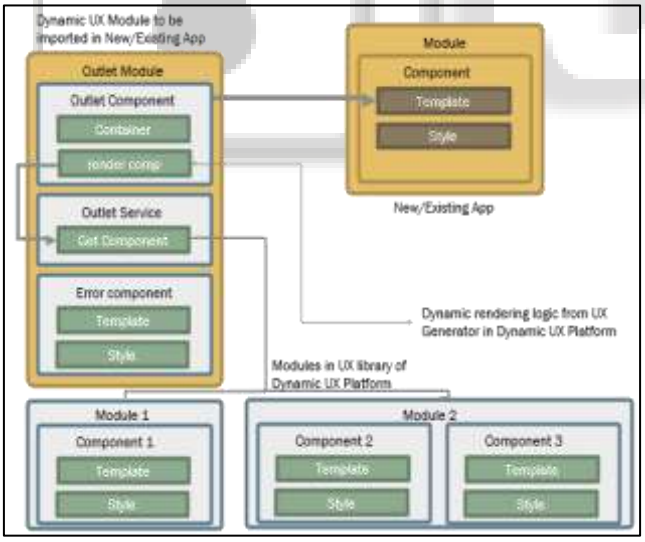

Fig. 4: Bringing dynamicity to the app

The existing app or the new app can be made dynamic by introducing an outlet module to each screen module. The outlet module has outlet component which is nothing but a container to hold the dynamic component the generator makes. The outlet component will render the component by making use of the outlet service which generates the component instances which are available in each module. The JSON or any other format of output from the generator, which can be parsed by the client module, will have the information about which all component to be used to generate the dynamic component which needs to be placed in the container.

V. BRAIN MODULE FOR SELF-LEARNING

The brain module does not collect user events to create a personality map, or get into model that can infringe data privacy. Instead the events are converted to weightage scores that is given to a machine learning module. The weightage inputs are mapped to UI category outputs. The weightage and UI category relation for the specific user is obtained using training data that comes from the use of the app. With more training data, the inter correlation between weightages is auto corrected, to match generic output UI element categories. When predefined thresholds for the UI categories are crossed, the corresponding UI elements are dynamically updated in the app. With change in weightage and UI category output values, the app keeps adapting to the user needs. In this way, rather than having predefined triggers for adapting the experience, a brain module can be trained to perform this function.

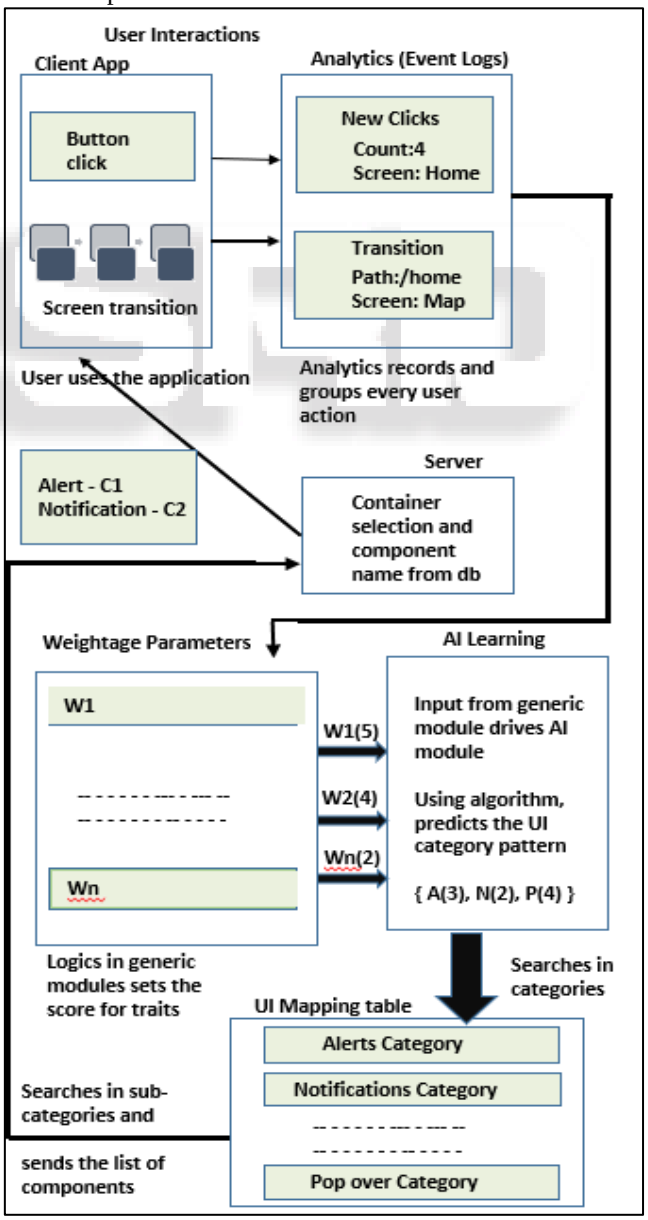

Fig. 5: Sequence diagram for implementing Brain module

VI. EXECUTION SEQUENCE

The high-level execution sequence that enables dynamic UX is shown in figure.

- Trigger Adapter will expose only a single API, which responds with current status of triggers.
- All rules, intelligent patterns, learning and analytics will act as feeds for the various triggers.
- Action adapter will respond with data, for setting attributes of UI Element. For example, for button it would be color change to set, and for a List view it would be the data to show in the List.
- UX Generator in the server invoke actions to get data for attributes of UI Elements, based on active triggers.
- A file is generated with details on rendering to be done. The screen render uses the generated file to load the appropriate UI element in the container, with the attributes and data defined in the file.

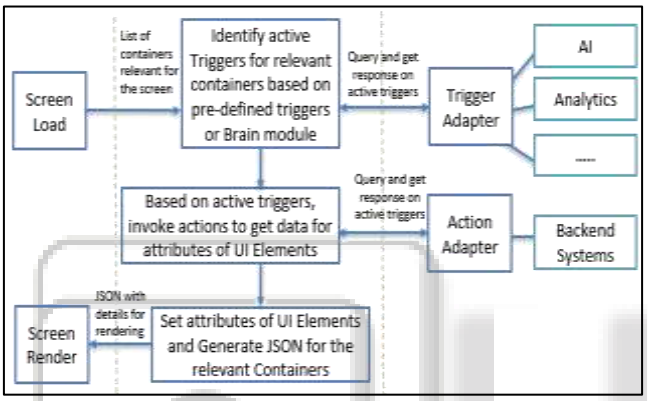

Fig. 6: Execution sequence of the app

VII. LEARNINGS FROM IMPLEMENTATION

- Dynamic User Experience module can be applied on Existing Apps and New Apps. The server framework being the same.
- Implementation was done with Angular, React, React Native – iOS & Android, and Kotlin. This was to validate the use of same server framework for use with SPA frameworks, cross platform frameworks and native frameworks for frontend development.
- Web component-based architecture supports re-use of UI Components across frameworks
- Any UI Component or set of components can be dynamically updated in the Containers
- Easy to introduce new triggers and experience changes at run time without re-deploying
- Focus is on true dynamic UX. Not on content change or selective hide/display of UI Components

Fig. 7: Dynamic UX Studio

Fig. 8: Same screen in app after adapting to three different users

VIII. CONCLUSION

Personalization of content in web and mobile apps, is mostly driven by predefined rules in content management systems (CMS), experience management solutions and predictive analytics solutions. These rules relying on user interaction events, history of user events and other logs captured on backend solutions. The data captured by these analytics tools in Stream Analytics, Big Data Analytics and Edge Analytics, are being leveraged to offer Hyper Personalization. AI driven Dynamic UX in Hyper Personalization, will encompass a machine learning algorithm that can offer personalized content and dynamic screens that adapt with usage. Automation and process bursting will also be offered in the next gen experience to optimize efficiency in every user interaction. AI based dynamic UX will offer a digital experience that is unique for the specific user.

REFERENCES

[1] H. Jetter and J. Gerken (2006), A Simplified Model of User Experience for Practical Application, pp. 106-111.
[2] V. Roto and E. Kaasinen (2008), The second international workshop on mobile internet user experience, pp. 571-573.
[3] Law, E., Roto, V., Hassenzahl, M., Vermeeren, A., and Kort, J. (2009), Understanding, Scoping and Defining User eXperience: A Survey Approach. Proc. CHI'09, ACM SIGCHI conference on Human Factors in Computing Systems.
[4] Evangelos Karapanos, Marc Hassenzahl, Jean-Bernard Martens (2008), User Experience Over Time, CHI 2008, April 5 – April 10, 2008, Florence, Italy ACM 978-1-60558-012-8/08/04.
[5] De Angeli, A., Sutcliffe, A., and Hartmann, J. Interaction, usability and aesthetics: what influences users' preferences. In Proceedings of the 6th conference on Designing Interactive systems, ACM (2006), 271–280

