Intertext: User Evaluation

The simplicity of consuming data is lost within the complexity of modern-day applications; nowadays, something as simple as checking the weather, reading a news article, browsing an image gallery, buying a product or service and filling out a form can be frustrating and time-consuming. We identified fundamental problems that leads to user experience problems and the compromise of user privacy and security. To address these problems as much as possible, we created Intertext.

DISCLAIMER: Some of the Intertext applications and/or the features mentioned in this survey are not yet fully implemented, and should be thought as proof of concepts for the time being. As of now, only the core engine that powers the rendering process, and the Intertext web client is implemented. An Intertext command-line client is under the works.

* Required

1. Worker ID *

2. What is your age group ? *

Mark only one oval.

- 0 - 16
- 17 - 30
- 31 - 45
- Above 45

3. What is your gender (optional) ?

Mark only one oval.

- Female
- Male
- Other: _____

4. Your country of origin (optional) ?

5. Years of experience? *

Mark only one oval.

- 1 ~~or less~~ (less than 1)
- 1 to 3
- 3 to 5
- 5 to 10
- 10 or more

Evaluation

Intertext is a platform that provides developers with a set of building blocks to build their applications. These building blocks are merely definitions/descriptions of actions and UI components in a functional sense without any form of presentational and experiential concerns. The development output of applications built with Intertext are functional descriptions with no visual cues. One can think of this output as "headless" applications definitions.

Intertext also offers a set of client applications that can render these outputs into fully functional front-ends. These are to include clients for different platforms, optimised for the host environment given its constraints. For instance Intertext web client renders these application definitions using web technologies suitable for a Desktop experience, whereas Intertext client for mobile renders them as Native mobile application.

Intertext clients can be thought of as web browsers such as Google Chrome and Safari, and it works in a similar manner. A user visits a website through a web browser, and the browser renders the website for them. It is similar to how Intertext works, with some fundamental differences and key benefits.

Please see below link to preview Intertext web client, and some of these UI components:
https://inx.oguzgelal.com/?u=https%3A%2F%2Fintertext-backend-demo.herokuapp.com%2Fdemo

The "building blocks" mentioned above in essence is called IUIDL (Intertext User Interface Description Language), which is an XML-based markup language. The main idea behind building an Intertext application is to assemble IUIDL in a generic backend based on the business logic, and serve it through a public endpoint. Intertext clients are completely agnostic of how IUIDL is assembled, or what the backend technology stack is. An example could be found in the repository below, which includes a sample "Recipe App" built on a Node.js / express server, and uses Handlebars templating engine to assemble the XML code.

IUIDL gives developers tools required to build complete and fully functional front-end applications. What comes out of the box is not limited to only UI components, but it

also offers commands that allows developers to "instruct" Intertext clients to make a request, navigate to another page, manage inputs, manage front-end state and so on.

https://github.com/oguzgelal/intertext-backend-demo

Cross-platform support

Problem:
Cross-platform application development is a prominent topic even to this day. Even though several development techniques emerged attempting to solve this problem, building applications capable of running on multiple platforms is still a complex engineering challenge. Creating and maintaining multiple applications for different platforms is even more costly and time consuming.

Solution:
Once an Intertext application is being served from an endpoint, it works on any Intertext client on any platform. IUIDL is universal, device agnostic, thus applications built on IUIDL are also universal and device agnostic. It's components, commands, even the layout system runs predictably across all devices of all screen sizes and capabilities, lifting concerns arising from cross-platform development from developers plate.

6. I would prefer using IUIDL over building my own components from scratch with the cost of adding cross-platform support *

Mark only one oval.

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree

Accessibility support

Problem:
Creating fully accessible applications is not always the priority for many development projects due to its costs and efforts. Accessibility implementations are often complicated as there are many different ways of creating accessible UI elements for different kinds of accessibility needs.

Solution:
As we have already established, IUIDL is agnostic of the view layer. Using IUIDL, developers do not implement "actual UI elements", they implement descriptions of a UI. Intertext clients are responsible for interpreting and rendering these descriptions in an accessible manner. The accessibility implementations are already handled, without needing any further action from the developer.

7. I would prefer using IUIDL over building my own components from scratch with the cost of making them accessible *

Mark only one oval.

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

Maintenance

Problem:
Building a front-end application is a great effort, but maintaining and enhancing it over time could be even more troublesome. The world of front- end development is a fast-growing and evolving ecosystem. The changes are persistent, and at times breaking. There are thousands of tools, libraries, frameworks, SDKs available at the fingertips of developers at no cost. It is a common practice to make use of these libraries as they help with the development significantly. However, the diversity in the libraries used to build the software results in a diversity of maintenance problems. Even the most well-tested and maintained libraries could break after an update, causing a headache for the developers and hardship for the users.

Solution:
Building Intertext applications are merely a matter of assembling and serving XML code from server side. Developers do not need a separate front-end project, any existing backend can be used to serve IUIDL. This way, maintenance requirement of Intertext applications are eliminated.

8. I would prefer using IUIDL over building my own components from scratch with the cost of maintenance

Mark only one oval.

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly Agree