Identification of User Interface Components for a Smart Application from Wireframe Designs

Technical Architecture Specification

1. Objective of this document

Based on the Project Proposal document, this document defines the technical, architecture and infrastructure requirements of the application.

2. Architecture Views

2.1. Logical View

The wireframe[1] created during the design phase are uploaded in a prototyping solution like build.me (https://www.build.me). **The entire module to identify different user interface components in a wireframe are performed as a micro service in the BUILD[2] environment.** This will enable in quick implementation of the idea as we can use the BUILD platform to manage the wireframe prototypes[3] uploaded by the user. And after identification of the User Interface components[4] we can use the metadata to generate the code using services provided by BUILD. As this is an independent service this can also be integrated to any other web development tools.

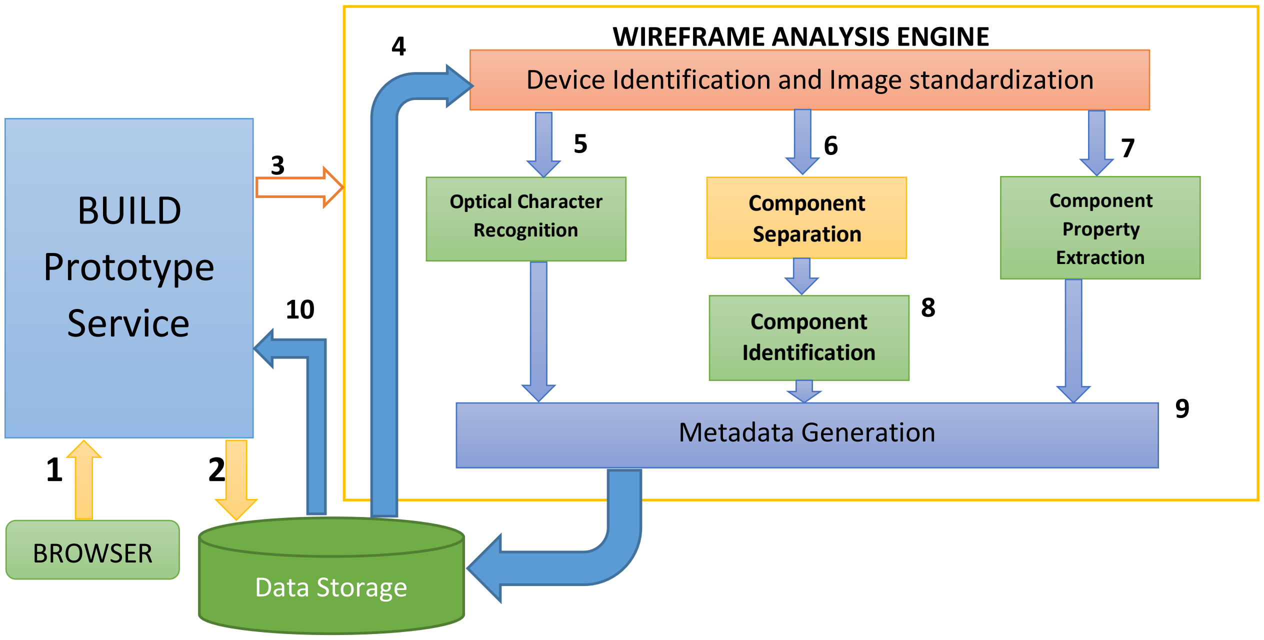

WIREFRAME ANALYSIS ENGINE
Device Identification and Image standardization
Optical Character Recognition
Component Separation
Component Property Extraction
Component Identification
Metadata Generation
BUILD Prototype Service
BROWSER
Data Storage
1
2
3
4
5
6
7
8
9
10

Figure 1: Logical view of different modules

Steps:

1. The user uploads a set of wireframe into a prototype object in BUILD.
2. For each wireframe uploaded a UI representation called image page is created and the image is stored in database as an asset marked with (Project Id, Page Id and Asset Id).
3. After the image is saved in database a trigger is initiated with the wireframe on the analysis engine.

4. The entire image is analyzed to identify the suitable device layout (Desktop, Tablet, Phone). **Then the corresponding grayscale image with noise reduction in fixed resolution is generated. This grayscale image is provided as input to next three steps processed parallel.**
5. **The image is analyzed using Optical Character Recognition system to identify the text and their position in the image.**
6. **The image is analyzed to break the wireframe into small segments each containing a User Component.**
7. **As the image is broken down different property associated with a component like position in the page, width, height etc. are deducted and passed to the metadata generator.**
8. **After the components are separated they are passed through an identifier to identify the type of the component.**
9. **The metadata generator receives all data and combines them to create a sterilized object that is persisted in database.**
10. After the metadata is made available in database a user gets the option to view the generated page in the prototype editor of BUILD, and using BUILD tools a user can even generate the code.

2.2. Implementation

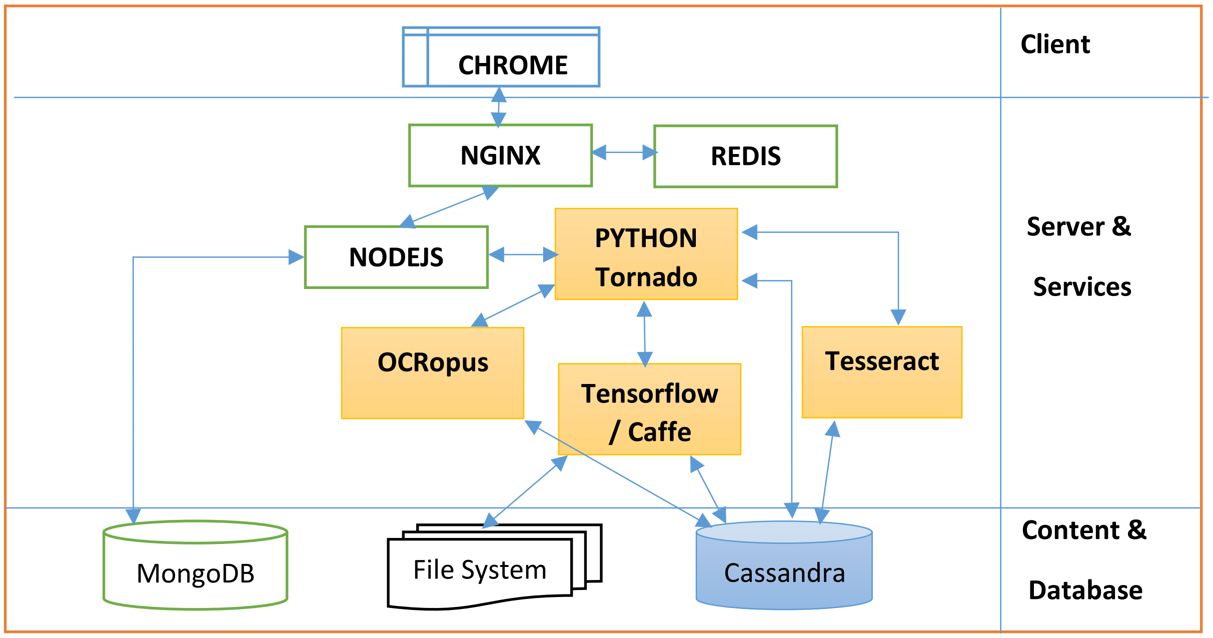


Figure 2:

A code repo is setup in GitHub as a private repository:

https://github.com/SubhasisDutta/WireframeTagging

2.2.1. Assumption

For the initial Proof of Concept it is assumed that:

1. All wireframe are for desktop (1280 X 1024) layout. Similar machine learning method using neural network that we will use for Component Identification can be used for device layout identification.
2. To keep analysis simple all wireframes even belonging to the same project are separately analyzed. The similarity of wireframes in same project are not exploited, which can give better result but will also increase the complexity of the system.

2.2.2. Image Standardization – Step 4

As a first step all input image is converted to gray image and to a resolution of 1280 X 1024 px. This is to remove the complexity of dynamic wireframe size. To do this using python packages in OpenCV.

2.2.3. Optical Character Recognition – Step 5

Currently exploring how open source solution like **OCRopus** (https://github.com/tmbdev/ocropy)
And **Tesseract** (https://github.com/tesseract-ocr/tesseract) can be modified to get the desired output.

2.2.4. Component Separation (Image Segmentation) – Step 6

Identified two image processing methods **canny edge detection with contour identification** and **watershed** method using OpenCV. Still verifying which will be better for our user case.

2.2.5. Component Property Extraction – Step 7

Yet to identify a good method to extract this information.

2.2.6. Component Identification – Step 8

Have identified two open source projects that are used for image recognition using machine learning techniques like neural networks. Am working of exploring them and building wireframe specific model to verify their accuracy.

a. **Tensor flow** - https://github.com/tensorflow/tensorflow
b. **Caffe** - https://github.com/BVLC/caffe

2.2.7. Metadata generation – Step 9

After step 4 to 8 work reliably the information can be used to create the metadata and stored for consumption in BUILD.

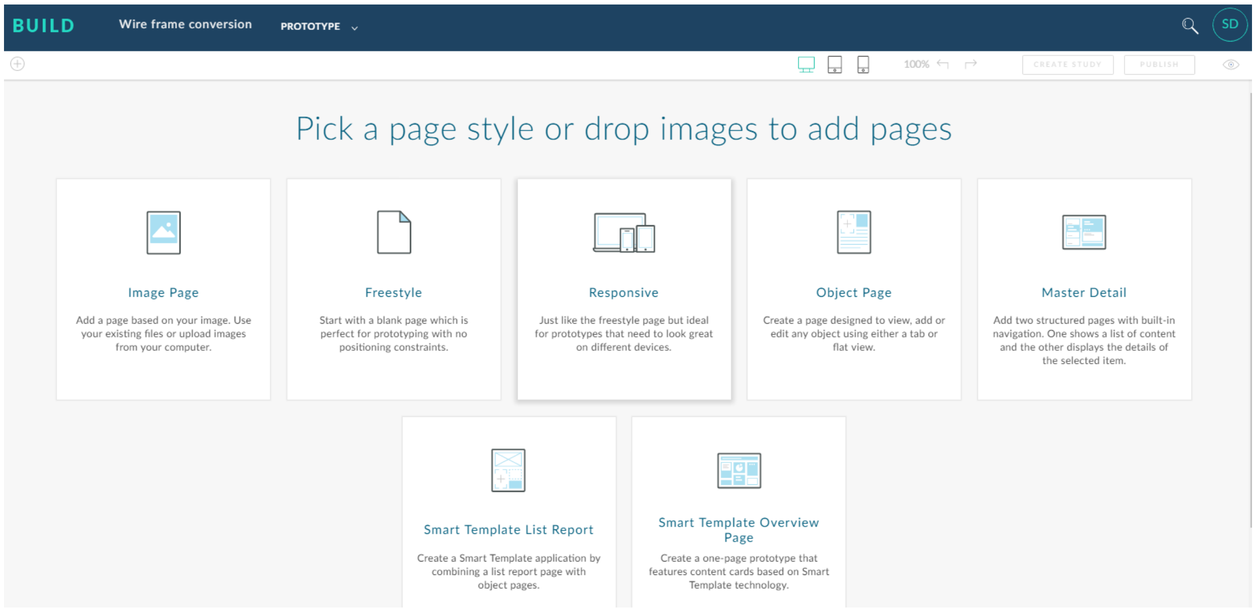
3. Infrastructure requirements

For the current prototype development a standard PC meets the infrastructure hardware requirements. After development the system can be setup in any cloud provided Virtual Machines.

Most of the software required are open source ex. Python, OpenCV, Tensorflow, etc.

4. User Interface

4.1. User Uploads Wireframe into BUILD



Pick a page style or drop images to add pages

Image Page
Add a page based on your image. Use your existing files or upload images from your computer.

Freestyle
Start with a blank page which is perfect for prototyping with no positioning constraints.

Responsive
Just like the freestyle page but ideal for prototypes that need to look great on different devices.

Object Page
Create a page designed to view, add or edit any object using either a tab or flat view.

Master Detail
Add two structured pages with built-in navigation. One shows a list of content and the other displays the details of the selected item.

Smart Template List Report
Create a Smart Template application by combining a list report page with object pages.

Smart Template Overview Page
Create a one-page prototype that features content cards based on Smart Template technology.

Figure 3: The wireframe image is uploaded into a BUILD project

4.2. BUILD generates a Page Container and renders it in UI for user with different options

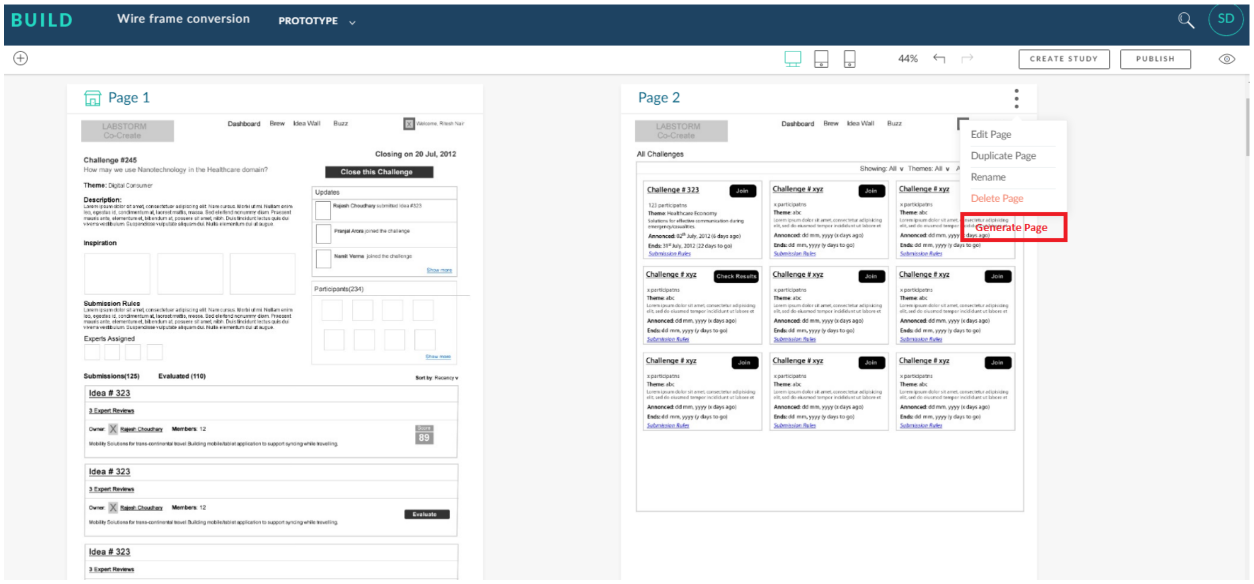


Figure 4: A new option Generate Page will be added, this option is enabled when an image page can be converted to an Object Page by the Wireframe Analytics Engine

4.3. In background all processing is done and if an Object Page can be successfully created an option in UI is provided.

As all process will be done in the backend no UI is required. However the Wireframe Analytics Engine will expose the following API.

1. **POST – Trigger service (/startprocess)** - This takes project ID, page ID and asset ID and obtains the Image from the database. And initiates the processing as a background process.

2. **GET – Check service (/checkpage)** – This verifies if an image page can be converted to an Object page. This process returns true id the Wireframe Analysis Engine was successful in generating the metadata else returns false.
3. **POST – Generate Page (/generatepage)** – This service builds on top of Create Page service of BUILD and uses the metadata generated by Wireframe Analysis Engine and creates a new page in the same project.

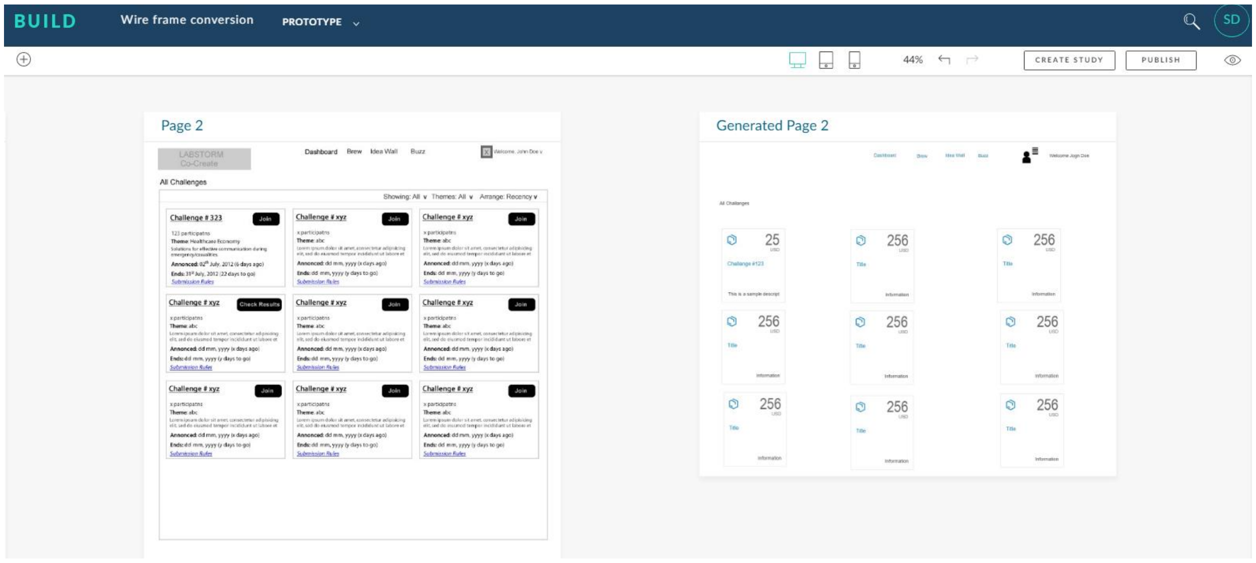


Figure 5: A new generated object page is added which can be converted to code by the BUILD platform