Ex 2: Demonstration of Growth MindSet

Aim:

Problem Statement: A company wants to setup the online platform help them with the design process

Ans)

The best possible design methodology/Approach to solve the problem and develop our product giving the best output is following the design thinking process.The design procedure presented in the preceding section is for simply-supported beams. Application of the design procedure to cantilever beams is straightforward as a cantilever beam can be taken as half of a simply-supported beam.

An important issue for cantilever beams is that the FRP plate must be adequately anchored to the clamped. This design procedure can also be applied to indeterminate beams, by simply treating each part of the beam between points of contra flexure as a simply-supported beam. In addition, if a clamped support exists, then the portion of the beam near the clamped support needs to be treated as a cantilever beam. One interesting and important situation of this kind arises in the strengthening of coupling beams in coupled shear walls.

Here, the coupling beams are subject to double bending so, for effective strengthening, the beam needs to be dealt with as two cantilever beams. Of course, such coupling beams are subject to reversed cyclic loading from either wind or earthquake, so flexural strengthening is required for bending in both directions. Design thinking is a non-linear, iterative framework that can be used in order to tackle big, complicated or even largely unknown problems in product development.

Steps to solve such problems

1) Empathize
2) Design
3) Ideative
4) Prototype
5) Test

Result:

Ex 3: User-Centered Design

Aim:-

Problem Statement:- By 2020, everyone had commas in their resumes and portfolios, and some even had full stops. And now, in the aftermath of the pandemic, we are devoid of connections. We are rarely in touch with one another, and as a result, innovators are unable to put their game-changing ideas into action without collaborations.

Ans)

The Best Possible Design Methodology/Approach to Solve the problem & and develop our product giving the best output is following the Design Thinking Process.

Design thinking is a non-linear, iterative framework that can be used in order to tackle big, complicated or even largely unknown problems in product development.Understanding the user and context of use - researching who my users are, what problems they have, and contextual considerations that motivate or influence their use of the product.

Specification of User & Business Requirements - establishes why the design is beneficial for both the user and the business and the problems the design is solving. At this point, designers and stakeholders are finding what metrics to use for measuring business success (ie. expected revenue, ROI) and what success for the user looks like.

Creation of Design Solutions - creation of potential solutions for the established problems. Creation of Storyboards, Journey Mapping, Wireframing, Designing Mockups, and User flows, Testing out different UI elements etc.Evaluation of designs – Usability Testing Is the most vital tool in this phase to determine how well the designs are performing.

Procedure:-

Found out who’s the end user for my product.Now to understand the end user, I’ve performed Qualitative & Quantitative Research using a short online survey.Created User Personas to represent a real target audience.Created an Empathy Map to understand more about the user persona.Now, Created Wireframes followed by the User flow of the product.Worked on the UI.

Conclusions:-

Understood that my target audience is going to be Employees and Students.After getting the survey responses, Came to a conclusion that most of my audience would prefer working onsite as their projects are being inactive & non interactive.After Gathering feedback from the testing phase, I've solved the feedbacks by making changes in the UI.In the 2nd testing phase, The Test users are able to perform all the actions/tasks within lesser time.

Result:-

Ex 4:- Demonstration of web-site design

Aim: -

Procedure:-

Step 1: Determine Your Site's Objective

Step 2: Pinpoint Your Domain Name

Step 3: Choose Your Website Platform

Step 4: Select Your Theme/Template

Step 5: Create Your Site Map

Step 6: Write Your Website Copy

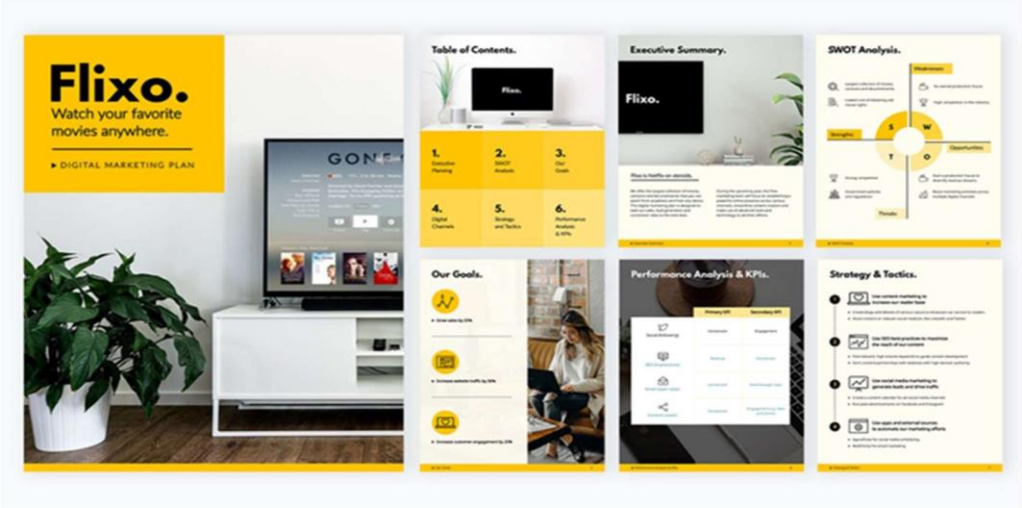
Step 7: Design Your Website Graphics

Step 8: Build Your WebPages

Step 9: Add Interactive Experiences

Step 10: Launch Your Website

Output:



Result:-

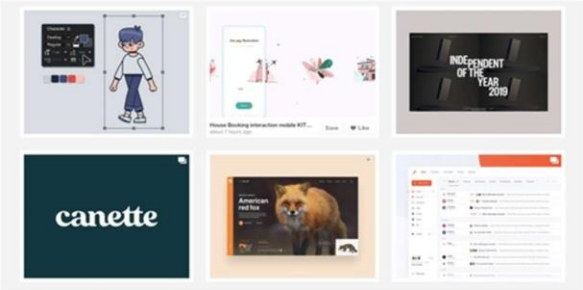
Ex 5:- Demonstration of User Interface

Aim: -

Procedure:-

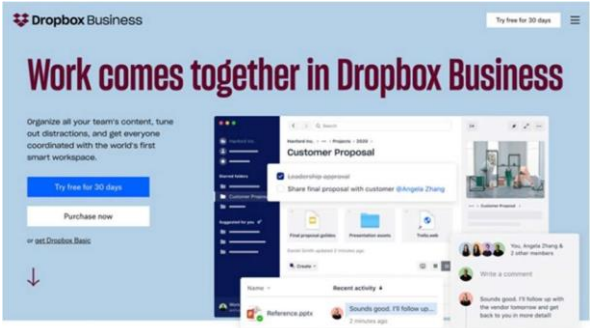
1. Dribbble's card design
2. Mailchimp's usability
3. Dropbox's responsive color system
4. Pinterest's waterfall effect
5. Hello Monday's white space
6. Current app's color palette
7. Rally's dynamism
8. Cognito's custom animation
9. Spotify's color gradients

1. Dribbble's card design

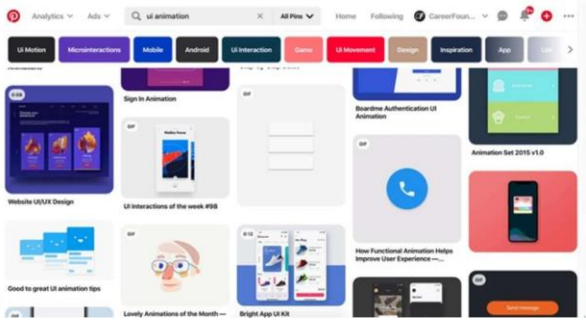


2.Mailchimp's usability

3. Dropbox's responsive color system



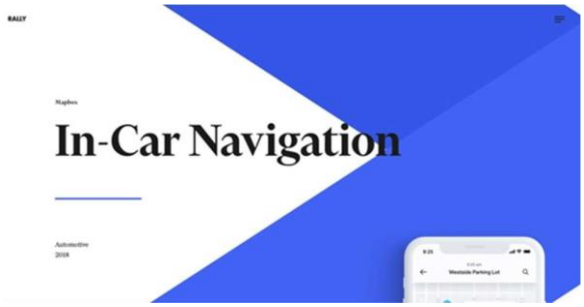
4.Pinterest's waterfall effect



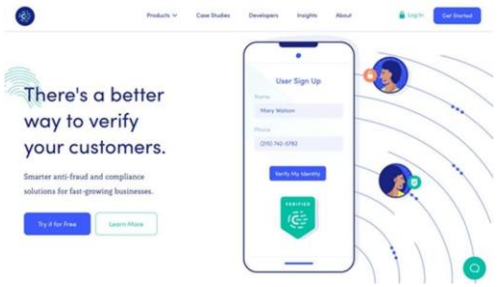
5. Hello Monday's white space

6. Current app's color palette

7. Rally's dynamism



8. Cognito's custom animation



9. Spotify's color gradients

Output :

Result:-

Ex 6:- Demonstration of URL Design

Aim: -

Procedure:-

1. Why do we need URL shortening?
URL shortening is used to create shorter aliases for long URLs. We call these shortened aliases “short links.” Users are redirected to the original URL when they hit these short links. Short links save a lot of space when displayed, printed, messaged, or tweeted. Additionally, users are less likely to mistype shorter URLs.The shortened URL is nearly one-third the size of the actual URL.
2. Requirements and Goals of the System
You should always clarify requirements at the beginning of the interview. Be sure to ask questions to find the exact scope of the system that the interviewer has in mind.
Our URL shortening system should meet the following requirements:

- Functional Requirements
- Non-Functional Requirements
- Extended Requirements

3. Capacity Estimation and Constraints
Our system will be read-heavy. There will be lots of redirection requests compared to new URL shortenings. Let’s assume a 100:1 ratio between read and write.
Traffic estimates:
Storage estimates:
Bandwidth estimates:
Memory estimates:
High-level estimates:
4. System APIs
Once we’ve finalized the requirements, it’s always a good idea to define the system APIs. This should explicitly state what is expected from the system.
We can have SOAP or REST APIs to expose the functionality of our service.
Following could be the definitions of the APIs for creating and deleting URLs:
createURL(api_dev_key,original_url,custom_alias=None,user_name=None,expire_date=None)
api_dev_key (string): The API developer key of a registered account. This will be used to, among other things, throttle users based on their allocated quota.
original_url (string): Original URL to be shortened.

custom_alias (string): Optional custom key for the URL.
user_name (string): Optional user name to be used in the encoding.
expire_date (string): Optional expiration date for the shortened URL.
Returns: (string)
A successful insertion returns the shortened URL; otherwise, it returns an error code.
deleteURL(api_dev_key,url_key)
Where "url_key" is a string representing the shortened URL to be retrieved; a successful deletion returns 'URL Removed'.
How do we detect and prevent abuse? A malicious user can put us out of business by consuming all URL keys in the current design. To prevent abuse, we can limit users via their api_dev_key. Each api_dev_key can be limited to a certain number of URL creations and redirections per some time period (which may be set to a different duration per developer key).
5. Database Design
Defining the DB schema in the early stages of the interview would help to understand the data flow among various components and later would guide towards data partitioning.
Database Schema:
We would need two tables: one for storing information about the URL mappings and one for the user's data who created the short link.
6. Basic System Design and Algorithm
The problem we are solving here is how to generate a short and unique key for a given URL.

a. Encoding actual URL
b. Generating keys offline

7. Data Partitioning and Replication

a. Range Based Partitioning:
b. Hash-Based Partitioning:

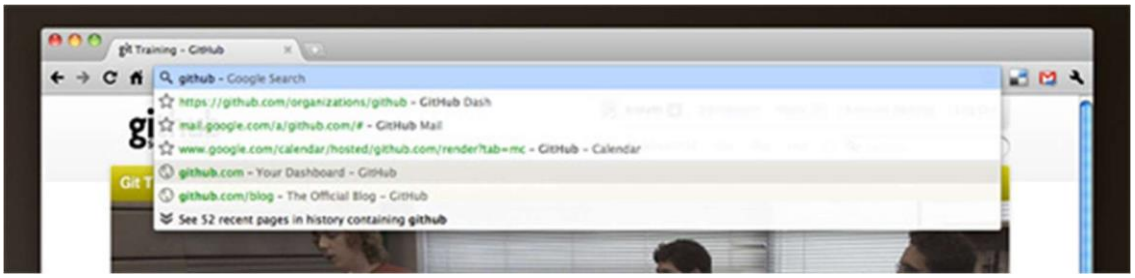
8. Cache
We can cache URLs that are frequently accessed. We can use any off-the-shelf solution like Memcached, which can store full URLs with their respective hashes. Thus, the application servers, before hitting the backend storage, can quickly check if the cache has the desired URL.
9. Load Balancer (LB)
We can add a Load balancing layer at three places in our system:

1. Between Clients and Application servers
2. Between Application Servers and database servers
3. Between Application Servers and Cache servers

10. Purging or DB cleanup
11. Telemetry.

12. Security and Permissions

Output:



1. ASCII-only user generated URL parts (defunkt, resque).
2. “pull” is a short version of “pull request” — single word, easily associated to the origin word.
3. The pull request number is scoped to defunkt/resque (starts at **one** there).
4. Anchor points to a scrolling position, not hidden content.

Result:-

Ex 7:- Demonstration of Web Browser design

Aim:

Description: Design in the browser is basically a concept using HTML and CSS as your primary design tools. Basically the code is writing right from the scratch during each phase of the project. From the clients brief on to the first design draft, to an rudimentary prototype to a finished product. Everything (or nearly everything) takes place in the browser. Rather than spending hours designing pixel-perfect design drafts in Photoshop, designing in the browser allows you to jump directly into the text editor and start shaping your code.

Demonstration:

Tools for Designing in the Browser

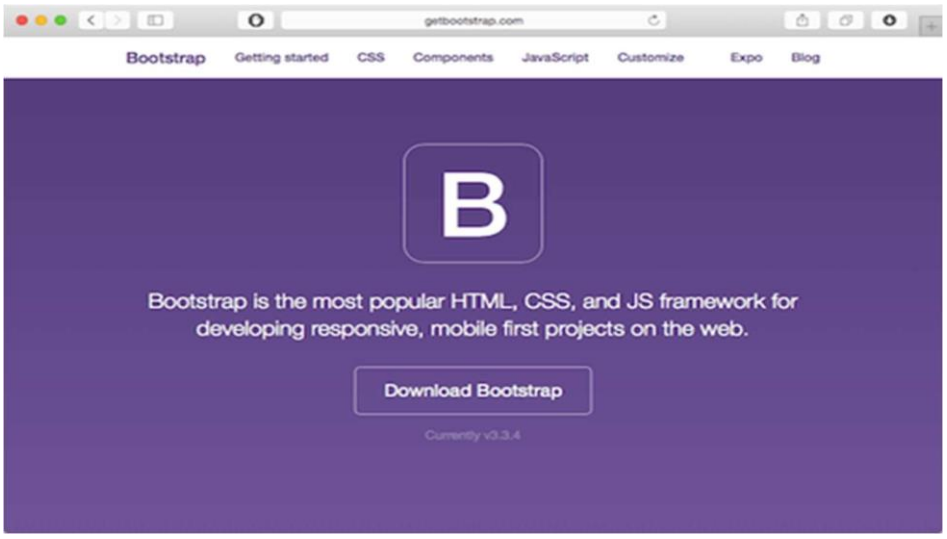
Convinced of the "design in the browser" concept? Want to get right into it? Perfect, here are a couple of great tools which are making your life easier when designing in the browser.

- Get a programming friendly editor

  This might sound a bit strange. But designing in the browser basically means working a lot on the code of your site. The editor will be your friend and go-to tool in many cases. Choose your editor wisely.

- Bootstrap – your front-end framework

  Bootstrap is probably the best known front-end prototyping framework available. Originally designed by Twitter, it's now available to everyone for free. Packed with some great functionality, it supports typography, forms, buttons and some great JavaScript options.



· Style guide

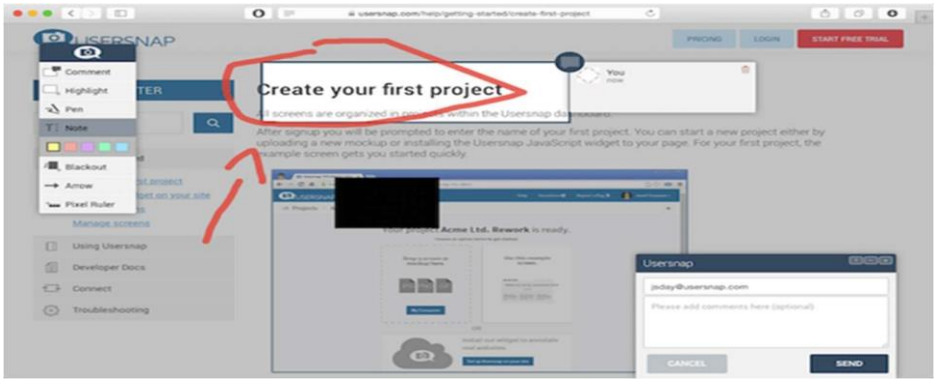
Next, it's about the style guide. It's super important to keep your design and style elements organized. With a style guide in place, design changes are super easy as they will come. And trust me: they will come.

· Chrome Developer Tools

After you created your first prototype in your browser, it's time to review, test and tweak. The best tools therefore are available for free in every Chrome browser (or Firefox if you prefer). With a right click on your site you can start the Chrome developer tools which offer you a broad range of features. You can play around with your code, move styles, edit content and much more.

· Keep track & manage tasks with Usersnap

Since designing in the browser is quite an agile approach, you might wonder how you keep track of change requests, bugs, ideas, etc. I therefore recommend Usersnap, which does a great job making collaboration on website or web app projects much easier. By adding the Usersnap widget to your prototype, every single piece of comment and feedback will be stored in your project overview where you can discuss feedback and track bugs.



Conclusion:

Designing in the browser still requires great design skills. No tool in the world will make up for your lack in design know how. However, the browser becomes more and more your design and development environment. And because most of the design process takes place in the browser, doesn't mean you should give up Photoshop.

Result:

Ex 8 :- Demonstration of Navigate design

Aim:

Description:

Navigation design is the discipline of creating, analyzing and implementing ways for users to navigate through a website or app.

Navigation plays an integral role in how users interact with and use your products. It is how your user can get from point A to point B and even point C in the least frustrating way possible.

To make these delightful interactions, designers employ a combination of UI patterns including links, labels and other UI elements. These patterns provide relevant information and make interacting with products easier.

Good navigation design can:

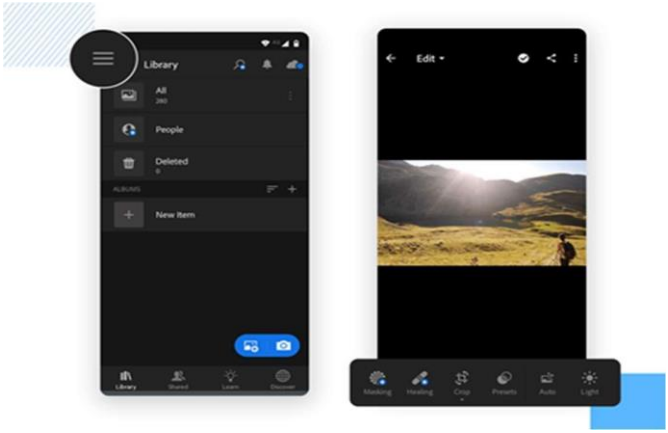
- Enhance a user's understanding
- Give them confidence using your product
- Provide credibility to a product

Implementing common navigation design patterns in UX design:

Many products will use a combination of these mechanisms in their designs because some patterns work better depending on the circumstances at hand.

Hamburger menu

The hamburger menu is often found on mobile, although it is increasingly becoming popular with desktop. The hamburger menu icon is 3 lines and can be clicked or tapped to reveal more navigation options.



Hamburger menu (left) and Tabs (right)

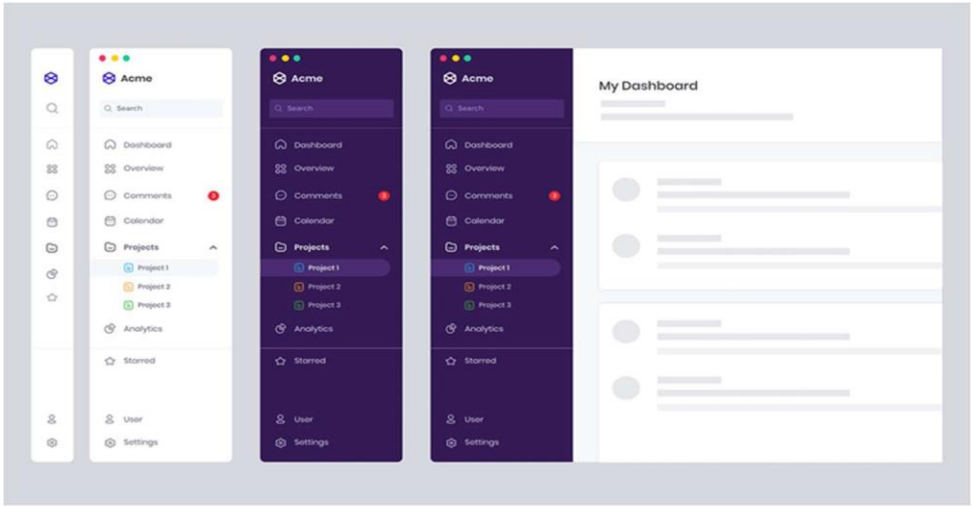
Tabs

Tabs are a popular navigation pattern and are commonly found on mobile devices. They're can be found on the bottom or top of the screen. Because you can only fit so many tabs at the bottom of your screen, you'll usually find the most important screens in a tabbed navigation.

Vertical navigation

Usually found on the left-hand side of screens, vertical navigation takes up generous screen space but displays a list of global navigation links and can include primary, secondary and tertiary navigation levels.

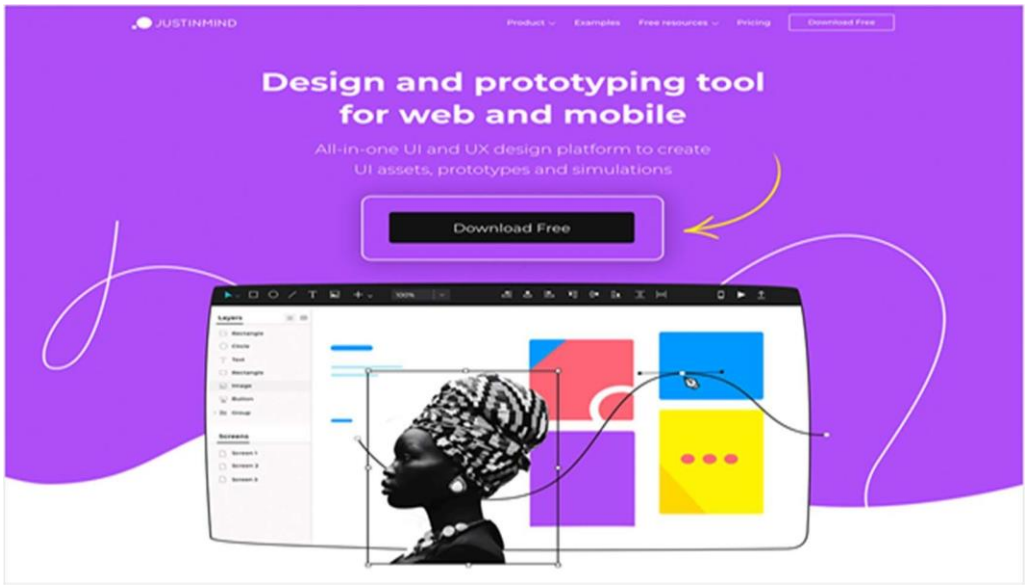
These are handy when the user already has a lot to digest in the screen. That means that in products that involve a lot of data, such as dashboard design, hiding the navigation options can be the best way to lower cognitive effort.



Vertical navigation menu

Call-to-action button

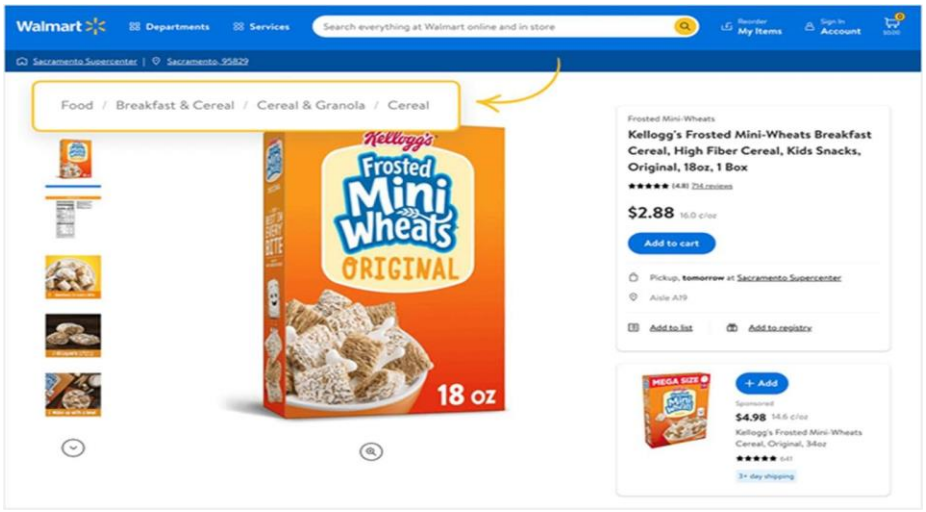
Call to action buttons are used to persuade, motivate and move your audience into an action whether it’s a sign up, a purchase or a download. They are usually given prime of place on websites and must be noticeable. Discover how picking the right UI colors can boost your conversions through the roof.



A prominent call to action button

Breadcrumbs

Inspired by the story of Hansel and Gretel, breadcrumb navigation (or breadcrumb trail), is a secondary navigation system that shows the user where they are in the system.



Conclusion:

There is a lot that goes into perfecting navigation design and getting it right means it usually has to go unnoticed. By aligning your user goals, content strategy and navigation design, you'll be able to create a cohesive and consistent user experience that your users will love.

Result:

Ex 9 :- Demonstration of Accidental & Essential Problems Excise Tasks

Aim:

Description:

It's when a UX design feature has been executed poorly that it sticks out like a fox in a hen-house. Sometimes UX design is so bad that it becomes counterproductive. There are lots of reasons for bad UX: perhaps the designer prioritized aesthetics over usability—the classic form-over-function problem—or perhaps the designers based their designs on their own assumptions rather than rigorous user research. Perhaps they integrated an intrusive feature, such as a pop-up, in an attempt to push a user to carry out a particular action. Typically, if the user has to dig deep to achieve what they want to achieve, feels forced to do something they don't want to, or gets lost in your user flow, the UX design needs to be improved.

Demonstration:

Essential and Accidental Problems

- Essential problems : Difficulties that are inherent in the nature of software
- Accidental problems : Difficulties related to the production of software
- 80 / 20 rule (one version): Immature engineering fields spend 80% of the time on accidental problems

  Mature engineering fields spend 20%

  Software engineering is probably approaching 50 / 50
- Most software engineering advances try to solve accidental problems
- Be skeptical, not pessimistic

  "Magical tools" claim will not solve this

  Buzzword processes will not solve this (spiral, structured, OO, agile, TDD, …)
- We've made slow steady progress in 20 years

  By basic research, solid engineering, and technology transition

  Strong typing, modularity, inheritance, modeling, IDEs, test criteria, education

Excise Tasks

- Overhead relates to solving problems:
    - Revenue Tasks : Sub-tasks that work to solve the problem directly
        - Studying
        - Doing homework
        - Listening to lectures
        - Taking tests
    - Excise Tasks : Sub-tasks that must be done but that are not really part of the problem
        - Driving to school
        - Parking !
        - Doing homework that does not reinforce concepts
- Excise tasks satisfy the needs of the tools or process, not the users

Conclusion: Understanding the Accidental & Essential Problems Excise Tasks on design UX to improve the skills on designing.

Result:-

Lab 10 :- Demonstration of design for user for GUI

Aim: -

Procedure:-

Step 1: Determine Your Site's Objective

Step 2: Pinpoint Your Domain Name

Step 3: Choose Your Website Platform

Step 4: Select Your Theme/Template

Step 5: Create Your Site Map

Step 6: Write Your Website Copy

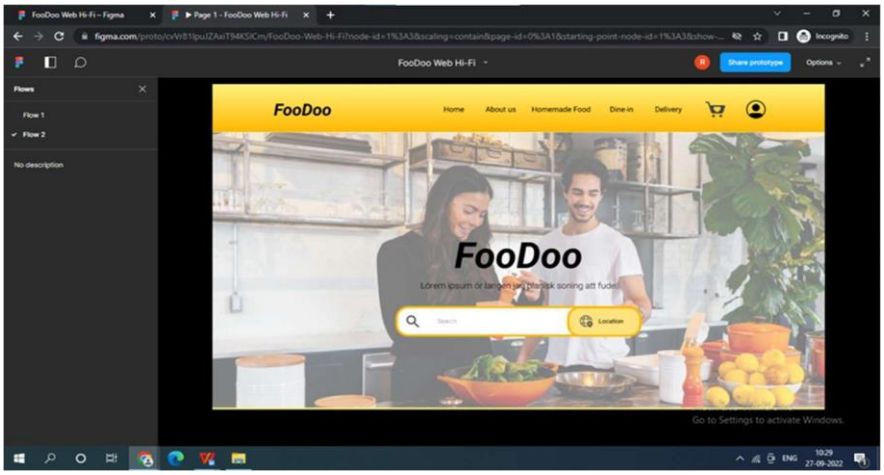
Step 7: Design Your Website Graphics

Step 8: Build Your WebPages

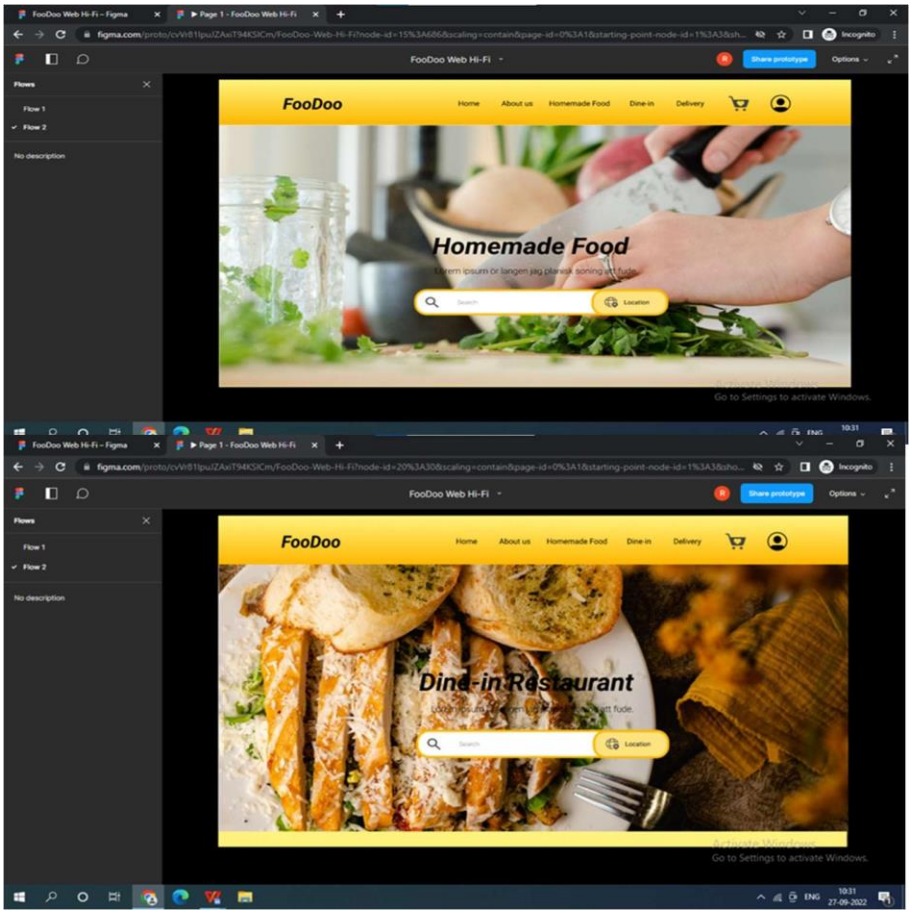
Step 9: Add Interactive Experiences

Step 10: Launch Your Website

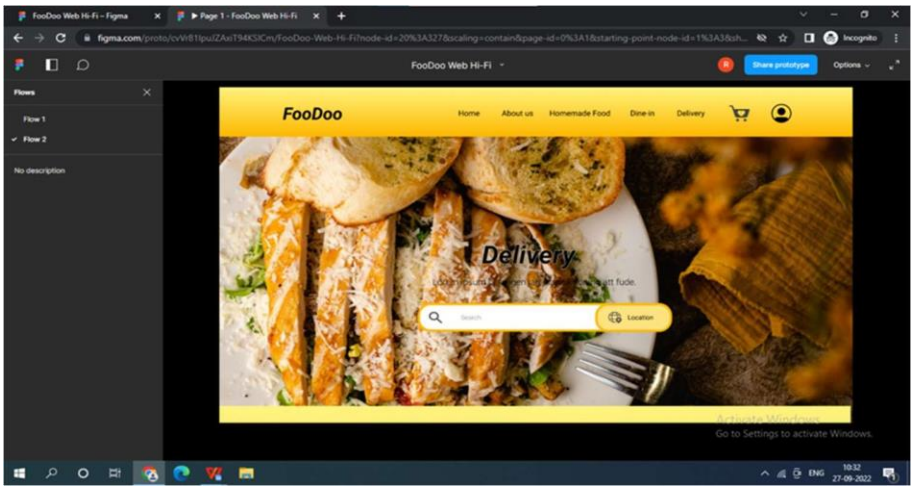
Output:

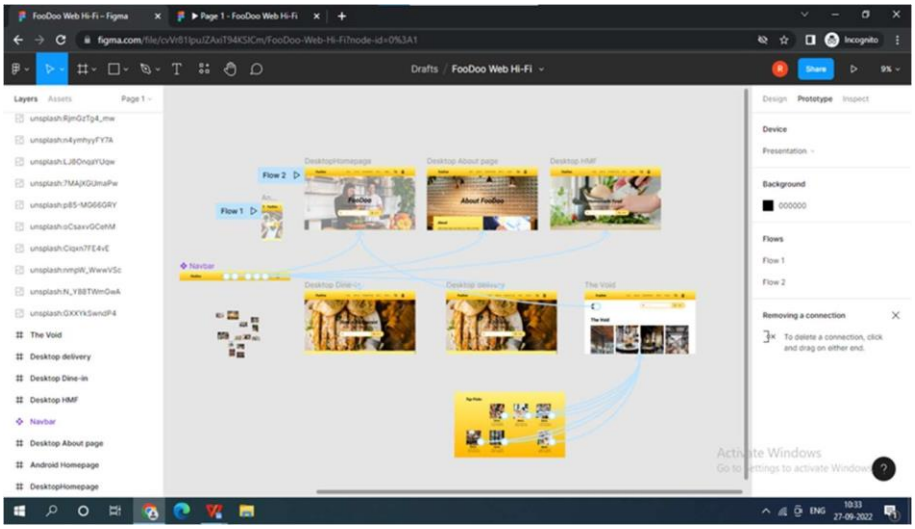

FooDoo Web Hi-Fi – Figma
Page 1 - FooDoo Web Hi-Fi
FooDoo Web Hi-Fi
Share prototype
Options
Flows
Flow 1
Flow 2
No description
FooDoo
Home
About us
Homemade Food
Dine-in
Delivery
FooDoo
Location
Go to Settings to activate Windows.
ENG
10:29
27-09-2022


FooDoo Web Hi-Fi – Figma
Page 1 - FooDoo Web Hi-Fi
FooDoo Web Hi-Fi
Share prototype
Options
Flows
Flow 1
Flow 2
No description
FooDoo
Home
About us
Homemade Food
Dine-in
Delivery
About FooDoo
About
Go to Settings to activate Windows.
ENG
10:30
27-09-2022


FooDoo Web Hi-Fi - Figma
Page 1 - FooDoo Web Hi-Fi
figma.com/proto/cvVr81IpuIZAxiT94KSICm/FooDoo-Web-Hi-Fi?node-id=15%3A686&scaling=contain&page-id=0%3A1&starting-point-node-id=1%3A3&sh...
Incognito
FooDoo Web Hi-Fi
Share prototype
Options
Flows
Flow 1
Flow 2
No description
FooDoo
Home
About us
Homemade Food
Dine-in
Delivery
Homemade Food
Location
Activate Windows
Go to Settings to activate Windows.
ENG
10:31

FooDoo Web Hi-Fi - Figma
Page 1 - FooDoo Web Hi-Fi
figma.com/proto/cvVr81IpuIZAxiT94KSICm/FooDoo-Web-Hi-Fi?node-id=20%3A30&scaling=contain&page-id=0%3A1&starting-point-node-id=1%3A3&sho...
Incognito
FooDoo Web Hi-Fi
Share prototype
Options
Flows
Flow 1
Flow 2
No description
FooDoo
Home
About us
Homemade Food
Dine-in
Delivery
Dine-in Restaurant
Location
Activate Windows
Go to Settings to activate Windows.
ENG
10:31
27-09-2022


FooDoo Web Hi-Fi - Figma
Page 1 - FooDoo Web Hi-Fi
figma.com/proto/cvVr81IpuIZAxiT94KSICm/FooDoo-Web-Hi-Fi?node-id=20%3A327&scaling=contain&page-id=0%3A1&starting-point-node-id=1%3A3&sh...
Incognito
FooDoo Web Hi-Fi
Share prototype
Options
Flows
Flow 1
Flow 2
No description
FooDoo
Home
About us
Homemade Food
Dine-in
Delivery
Delivery
Location
Activate Windows
Go to Settings to activate Windows.
ENG
10:32
27-09-2022


FooDoo Web Hi-Fi - Figma
Page 1 - FooDoo Web Hi-Fi
figma.com/file/cvVr81IpuIZAxiT94KSICm/FooDoo-Web-Hi-Fi?node-id=0%3A1
Drafts / FooDoo Web Hi-Fi
Share
Layers
Assets
Page 1
The Void
Desktop delivery
Desktop Dine-in
Desktop HMF
Navbar
Desktop About page
Android Homepage
DesktopHomepage
Flow 1
Flow 2
Design
Prototype
Inspect
Device
Presentation
Background
000000
Flows
Flow 1
Flow 2
Removing a connection
To delete a connection, click and drag on either end.
Activate Windows
ENG
10:33
27-09-2022

Result:

Lab 11 :- To demonstrate myths of usable security

Aim:

Procedure:

Password asking is a classic example of the security usability tradeoff

1. SECURITY AND UX: ALLIES NOT ENEMIES

Leading technology companies are rapidly implementing systems that increase security and user experience simultaneously. For example, the latest iPhone X uses their Face ID facial recognition technology in place of a PIN code or a fingerprint. Apple touts that this technology is secure enough to be your authorization when making purchases. Both usable and secure, this new IOS feature naturally accelerates purchasing time

Companies can create a symbiosis between security and usability

- The first step to making security and usability harmonize is co-creation.
- In addition to creating a culture of co-creation, security teams should embrace agile methodologies like most UX teams have.
- Finally, and perhaps most critically, is to use the techniques of UX in order to improve security.

The best example of this is choice architecture. Given options in the interface, the default or easy path should be the safer or more secure choice. We can also use UX techniques of analyzing behavior to find out if people are spending time and effort on risky behaviors – and if so, do research and analysis to figure out why.

Result:-

Lab 12 :- To implement demonstration of dialog box and toolbar

Aim:

Procedure:

Designing Dialog box

The Dialog class is the base class for dialogs, but you should avoid instantiating Dialog directly. Instead, use one of the following subclasses:

AlertDialog:A dialog that can show a title, up to three buttons, a list of selectable items, or a custom layout.

DatePickerDialog or TimePickerDialog: A dialog with a pre-defined UI that allows the user to select a date or time.

Creating a Dialog Fragment
You can accomplish a wide variety of dialog designs—including custom layouts and those described in the Dialogs design guide—by extending DialogFragment and creating an AlertDialog in the onCreateDialog() callback method.
For example, here's a basic AlertDialog that's managed within a DialogFragment:

```
class StartGameDialogFragment : DialogFragment() {

  override fun onCreateDialog(savedInstanceState: Bundle): Dialog {
  return activity?.let {
  // Use the Builder class for convenient dialog construction
   val builder = AlertDialog.Builder(it)
       builder.setMessage(R.string.dialog_start_game)
       .setPositiveButton(R.string.start,
       DialogInterface.OnClickListener { dialog, id  ->
        // START THE GAME!
       })
       .setNegativeButton(R.string.cancel,
       DialogInterface.OnClickListener { dialog, id ->
        // User cancelled the dialog
        })
         // Create the AlertDialog object and return it
         builder.create()
        } ?: throw IllegalStateException("Activity cannot be null")
    }
}
```

Figure 1. A dialog with a message and two action buttons.

Now, when you create an instance of this class and call show() on that object, the dialog appears as shown in figure 1.

Building an Alert Dialog

The AlertDialog class allows you to build a variety of dialog designs and is often the only dialog class you'll need. As shown in figure 2, there are three regions of an alert dialog:

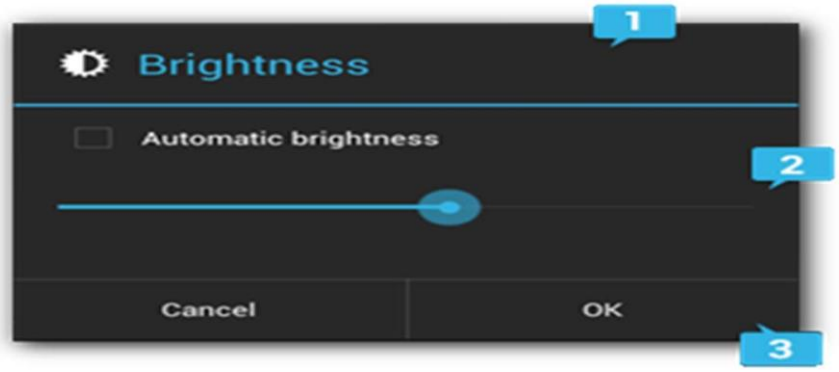


Figure 2. The layout of a dialog.

1. Title
2. Content area
3. Action buttons

The AlertDialog.Builder class provides APIs that allow you to create an AlertDialog with these kinds of content, including a custom layout.

To build an AlertDialog:

// 1. Instantiate an <code>

<a
href="/reference/android/app/AlertDialog.Builder.html">AlertDialog.Builder</
a></code> with its constructor

```
val builder: AlertDialog.Builder? = activity?.let {

   AlertDialog.Builder(it)

}


// 2. Chain together various setter methods to set the dialog characteristics

builder?.setMessage(R.string.dialog_message)

        .setTitle(R.string.dialog_title)


// 3. Get the <code><a
href="/reference/android/app/AlertDialog.html">AlertDialog</a></code>  from
<code><a
href="/reference/android/app/AlertDialog.Builder.html#create()">create()</a><
/code>

val dialog: AlertDialog? = builder?.create()
```

Adding buttons

To add action buttons like those in figure 2, call the setPositiveButton() and setNegativeButton() methods:

```
val alertDialog: AlertDialog? = activity?.let {

    val builder = AlertDialog.Builder(it)

    builder.apply {

        setPositiveButton(R.string.ok,

                DialogInterface.OnClickListener { dialog, id ->

                    // User clicked OK button

                })

        setNegativeButton(R.string.cancel,

                DialogInterface.OnClickListener { dialog, id ->
```

```
                // User cancelled the dialog
            })
    }
    // Set other dialog properties
    // Create the AlertDialog
    builder.create()
}
```

The set...Button() methods require a title for the button (supplied by a string resource) and a DialogInterface.OnClickListener that defines the action to take when the user presses the button.

There are three different action buttons you can add:

Positive

You should use this to accept and continue with the action (the "OK" action).

Negative

You should use this to cancel the action.

Neutral

You should use this when the user may not want to proceed with the action, but doesn't necessarily want to cancel. It appears between the positive and negative buttons. For example, the action might be "Remind me later."

You can add only one of each button type to an AlertDialog. That is, you cannot have more than one "positive" button.

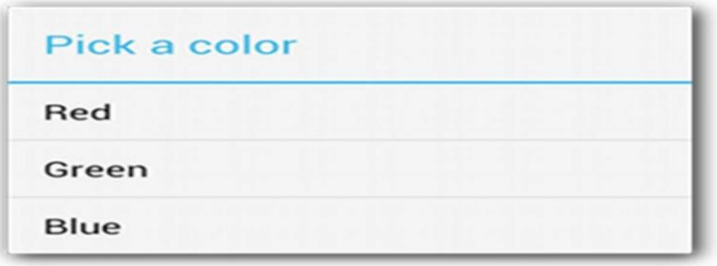


Figure 3. A dialog with a title and list.

Adding a list

There are three kinds of lists available with the AlertDialog APIs:

- A traditional single-choice list
- A persistent single-choice list (radio buttons)
- A persistent multiple-choice list (checkboxes)

To create a single-choice list like the one in figure 3, use the setItems() method:

```
override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
    return activity?.let {
        val builder = AlertDialog.Builder(it)
        builder.setTitle(R.string.pick_color)
                .setItems(R.array.colors_array,
                        DialogInterface.OnClickListener { dialog, which ->
                            // The 'which' argument contains the index position
                            // of the selected item
                        })
        builder.create()
    } ?: throw IllegalStateException("Activity cannot be null")
}
```

Because the list appears in the dialog's content area, the dialog cannot show both a message and a list and you should set a title for the dialog with setTitle(). To specify the items for the list, call setItems(), passing an array. Alternatively, you can specify a list using setAdapter(). This allows you to back the list with dynamic data (such as from a database) using a ListAdapter.

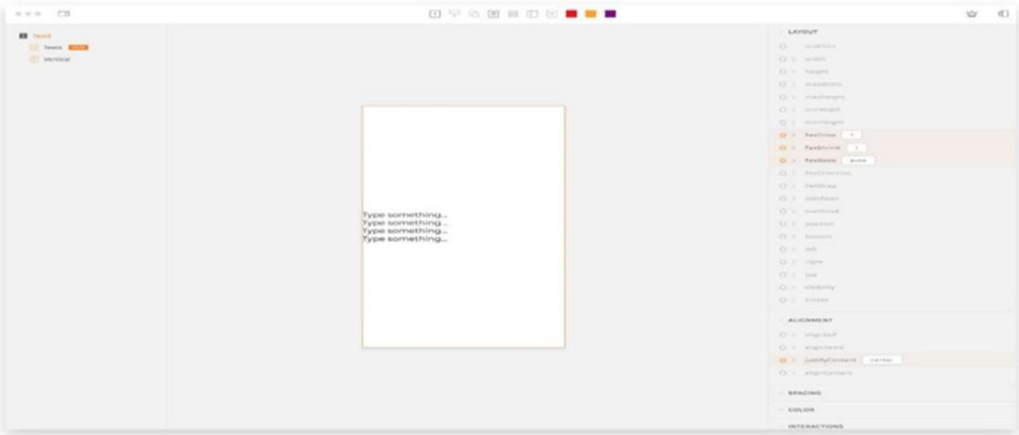
If you choose to back your list with a ListAdapter, always use a Loader so that the content loads asynchronously. This is described further in Building Layouts with an Adapter and the Loaders guide.

2. Designing Toolbar

Stage 1. Priming

Priming stands for the prime (first) version of the interface, like prototyping, but in Views Tools, the result is production quality code.
we move all the actions to the Top Bar.view component. Here's the result of our Priming work. At this point, the functionality is in with a default interface representation.

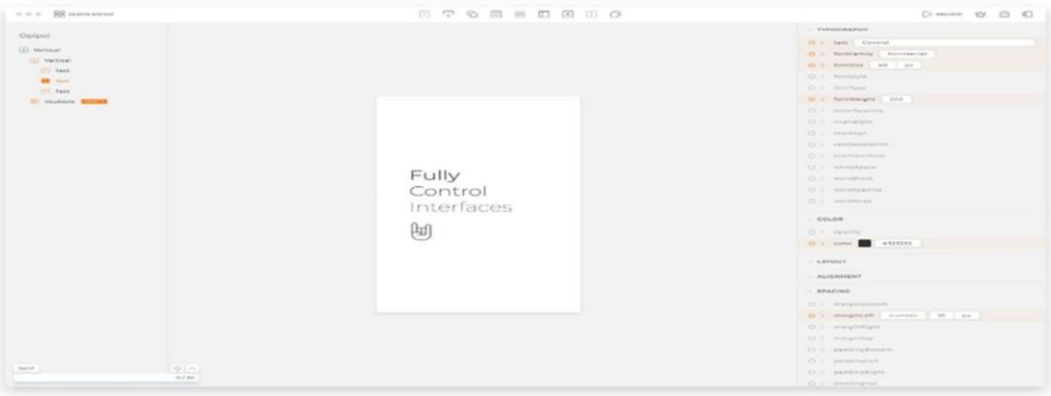
The result after Priming stage
Stage 2. Composition
We need to move buttons around, put back the window controls where they belong, center the block related tools, and remove the label from the Share Feedback button to make everything clean and consistent.



The result after playing with composition

Stage 3. Styling

On to the beauty! First, we need to replace the Views logo with a bit more obvious icon symbolizing the Design System window. There are few icons we need to make from scratch.



The result after playing with styles

Result:-

Lab 13 :- To demonstrate games and customizing controller buttons

Aim:

Procedure :

Demonstration of games:

Game UI designs refer to the user interface that game players first see when they enter a web or mobile app game. They work as the bridge to connect the game and the players together, making it easy for players to quickly understand the gameplay mechanics, find the right information, and start playing as soon as possible.

The better the design your game has, including better storytelling, high-quality animations, characters graphics, gameplay mechanics and user experience, the longer players are likely to play. The more immersive your game's UI is, the more likely players will make a purchase.

In total, a game's UI is a major factor for a potential customer and the better the UI, the more likely your revenue will increase.

10 of the best practices that can make your game UI designs shine:

i. Make players understand your game naturally

ii. Guide players using the right method at the right time

iii. Allow players to skip things

iv. Make your rewards and in-app purchases stand out

v. Never clutter game UI

vi. Create a clear visual hierarchy

vii. Make accessible navigation, like sidebar

viii. Make your game responsive

ix. Enable gamers to give feedback

x. Test your game UIs

2. Customizing controller buttons:

Customizing game's buttons is an important feature that can make gaming sessions more comfortable, make you a better gamer, and open the experience to new players.

i. In the toolbar, select Button.

ii. Drag the button images onto the Editor Canvas.

iii. Prepare the canvas (optional): Change the Grid size:

- Deselect any elements in the tree.
- In the Properties section, Canvas panel, adjust the Grid size to 10 pixels

iv. Prepare the canvas (optional): Set the Zoom to 200%.

v. Next, move the buttons into an arrangement in the canvas

vi. Once they are arranged, you might find it handy to be able to move them all at once. This is the Container's job. Draw a Container around all the buttons:

- Select the Container button in the Toolbar.
- In the canvas, draw around the buttons, until all have a red border.
- The tree will have a container with all the buttons a child of it.
- Select the container in the tree and then move the buttons in the canvas and it will move the buttons as one.

vii. Anchor the Container to the bottom center of the window (or wherever you plan for your buttons to be positioned on the skin):

- With the Container selected in the Tree, open on Position panel in the Properties section.
- Select the Anchor position.

viii. Add a second button state or more

ix. Add the appropriate action to each button image:

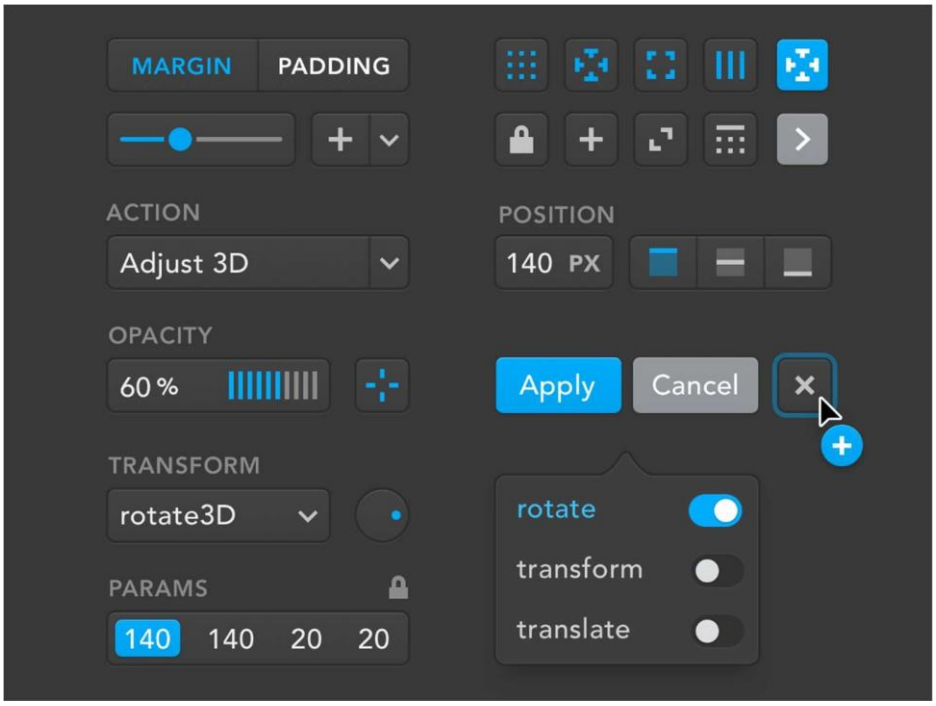
- Select a button in the Tree or Editor.
- In the Properties section, click on Actions.

- Double-click or click on the plus to the right of the table, to open the Action Settings.
- For a Tilt Down button, Choose Mouse Click for the Source and Tilt Down for the Action. Click, OK, when finished.
- Repeat this process for each button.

x. Test the skin with the Live Preview.

xi. Finally, click Close and save the skin (to the Skins Directory).

Output:-


MARGIN
PADDING
ACTION
Adjust 3D
POSITION
140 PX
OPACITY
60 %
Apply
Cancel
TRANSFORM
rotate3D
rotate
transform
translate
PARAMS
140
140
20
20

Result:-