Design Document

Budgie

Ethan Alvi	etalvi@student.fullsail.edu
Saul Gerda	sgerdaramos@student.fullsail.edu
Steven Alexander	sealexander1@studen.fullsail.edu

Contents

1. Introduction

Purpose:

The purpose of the project is to develop a mobile application that provides a smart budgeting tool for users during grocery shopping. This application is designed to help users adhere to a predefined budget, ensuring they can manage their expenses effectively and avoid overspending. By offering a seamless and intuitive user experience, the app aims to make budgeting a straightforward and integral part of the shopping process.

The application allows users to input their desired budget and functions as both, a shopping list and a real-time budget tracker. As the user scans items with the camera it will cross the item from the list simultaneously adding the total price of the item, and subtracting it from the specified budget; providing instant feedback on the remaining funds through notifications. This feature allows users to make informed purchasing decisions, on the go, enhancing the user's ability to stay to their financial goal.

Overall, this budgeting application addresses a common problem for many consumers, staying within budget while shopping. By integrating budgeting directly into the shopping experience, the app not only helps users save money but also promotes better financial habits. This document will provide detailed information on the design, functionality, and implementation of the application, ensuring that every engineer understands its significance and can contribute effectively to its development.

Goal: What problem is this solving?

The goal of this project is to solve the problem of overspending during grocery shopping by providing a real-time budgeting tool that helps users track expenses, make informed purchasing decisions, and promote responsible spending habits.

Existing Solution:

No. This is a completely new project starting from the start.

Definitions:

N/A (Our Domain is a common definition that is not needed.)

2. Overall Description

2.1. Product Perspective

2.1.1. User interface Hardware

The application will use the touch screen to interact with the user and give the user feedback. Also, the application will utilize the device camera to read information on product prices and calculate a total that will be subtracted from the budget.

2.1.2. Hardware interfaces & Memory Constraints

A mobile device capable of running the Android system.
Mobile device with a camera and internal storage.
The storage will use the phone's internal system to store previous shopping lists and budgets. This will be stored as a text file we aim to use less storage.

2.1.3. Software interfaces

Android

2.1.4. Library Dependencies

.Net MAUI
IronPython

2.1.5. Database

Local data storage.
The phone will hold any data saved.

2.1.6. 3rd Party Data and APIs

ML Barcode Scan Kit (Android)
Camera2 API (Android)

3. Specific Requirements

3.1. Milestones

Feature A:

1. Users can input shopping items in a list format.
2. Users can intake multiple quantities of the same item.
3. Users can add the price of the same item inputted multiple times and get the total.
4. Users can input their desired budget.
5. Users can use the camera to read and take the price of items.
6. the app will automatically subtract item prices from the budget.
7. users can delete items from the shopping list.
8. When an item is deleted, its price will be added back to the budget.

Feature B:

9. Users can manually input the price of items if the scan can’t read the price.
10. Users will be notified when they are over budget.
11. Users can store previous shopping lists.
12. Users can change application settings.
13. Users can store previous budgets.
14. Users can reuse a list with a new budget.
15. Users can create multiple lists and budgets.
16. The app will use machine learning to enhance the shopping experience.
17. Users will receive suggestions for commonly inputted items while typing.

Feature C:

18. Users can share their shopping lists with other users.
19. Users can be reminded of the app when they are near a store

20. The app will include geo-locations features.
21. Users will be able to select the correct price if the scan reads multiple prices.

3.1.1. Minimum Viable Product (A Features)

1. Users can input shopping items in a list format.
2. Users can intake multiple quantities of the same item.
3. Users can add the price of the same item inputted multiple times and get the total.
4. Users can input their desired budget.
5. Users can use the camera to read and take the price of items.
6. the app will automatically subtract item prices from the budget.
7. users can delete items from the shopping list.
8. When an item is deleted, its price will be added back to the budget.

3.1.2. Alpha (B Features)

1. Users can manually input the price of items if the scan can't read the price.
2. Users will be notified when they are over budget.
3. Users can store previous shopping lists.
4. Users can change application settings.
5. Users can store previous budgets.
6. Users can reuse a list with a new budget.
7. Users can create multiple lists and budgets.
8. The app will use machine learning to enhance the shopping experience.
9. Users will receive suggestions for commonly inputted items while typing.

3.1.3. Beta (C Feature)

1. Users can share their shopping lists with other users.
2. The app will include geo-locations features.
3. Users will be able to select the correct price if the scan reads multiple prices.

4. User Experience

4.1. Use Cases

Main User Flows:

Flow 1: User wants to set their grocery shopping budget

1. Open app
2. Hit 'Create Budget' button
3. Enter budget name
4. Enter budget amount
5. Hit 'Create/Save' button

Flow 2: User wants to write their shopping list

1. Open app
2. Open budget (if no budget, refer to *Flow 1*)
3. Hit ‘Add Item’ button
4. Enter item name
5. Repeat steps 3 and 4 for each additional item in shopping list

Flow 3: User wants to check off an item in their list

1. Open app
2. Perform *Flow 1* and *Flow 2* if not done already
3. Enter price next to item (optional)
4. Hit checkbox next to item (if item has no price, user enters *Label Scanning Flow*)

Flow 4: User wants to uncheck an item and add it back to their shopping list

1. Open app
2. Assuming the user has already done *Flow 1, Flow 2,* and *Flow 3*, scroll down to bottom of Budget page
3. Below shopping list, under ‘In Cart’, uncheck box next to the item

Flow 5: User wants to add an item to their list from a price label

1. Open app
2. Open budget (if no budget, refer to *Flow 1*)
3. Hit ‘Add Item from Label’ button
4. User performs *Label Scanning Flow*
5. New item with default name ‘item 1’ appears with scanned price

Flow 6: User wants to remove an item from their list

1. Open app
2. Perform *Flow 1* and *Flow 2* if not done already
3. Hit the ‘X’ button next to the item

Flow 7: User wants to know if they are overbudget

1. Open app
2. Open budget (if no budget, refer to *Flow 1*)
3. Current budget progress is located at the top (will turn red if overbudget)
4. If performing *Flow 3* or *Flow 8* will cause you to be overbudget, a popup will inform you that you are now overbudget

Flow 8: User wants to change the quantity of an item in their shopping list

1. Open app
2. Perform *Flow 1* and *Flow 2* (or *Flow 5*) if not done already

3. Hit the up arrow next to the item (or down arrow) to increase (or decrease) the quantity

Flow 9: User wants to delete a budget

1. Open app
2. Assuming user has performed *Flow 1*, Hit the 'Delete Budget' button next to the budget icon
3. A popup will ask if you are sure. Hit yes (or no if you are not)

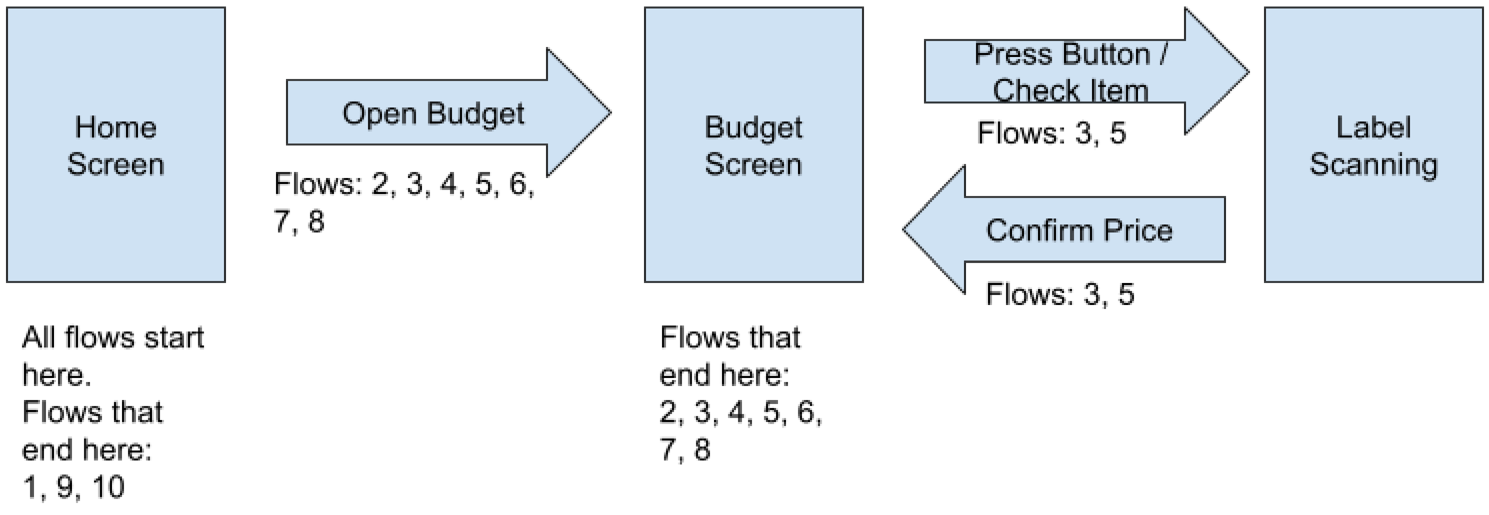
Flow 10: User wants to update the budget

1. Open app
2. Assuming user has performed *Flow 1*, Hit the 'Edit' button next to the budget icon
3. Enter new budget amount
4. Hit 'Save' button

Sub Flows:

Label Scanning Flow

1. Camera opens automatically. Aim camera at label's price
2. Hit 'scan' button when ready
3. Predicted number appears. If incorrect hit 'rescan' button and repeat steps 1 and 2 or change number manually.
4. Hit 'save' button


Home Screen
Open Budget
Flows: 2, 3, 4, 5, 6, 7, 8
Budget Screen
Press Button / Check Item
Flows: 3, 5
Label Scanning
Confirm Price
Flows: 3, 5
All flows start here. Flows that end here: 1, 9, 10
Flows that end here: 2, 3, 4, 5, 6, 7, 8

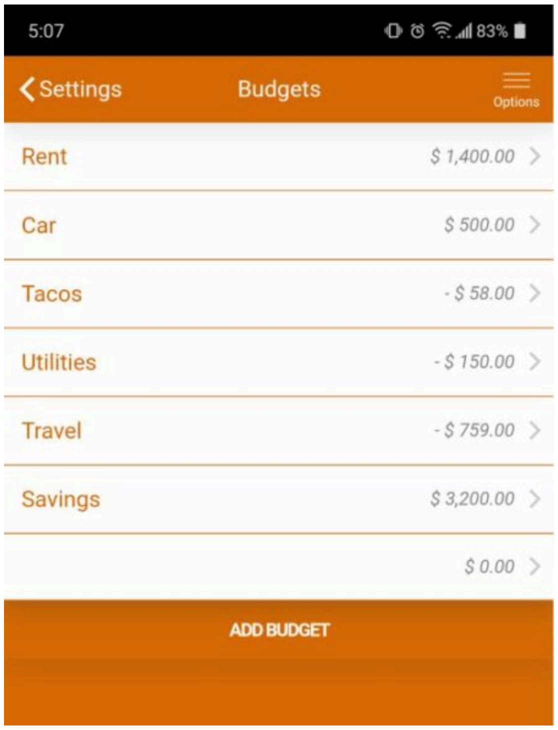
4.2. User characteristics

Our ideal users are diverse, each with unique needs and preferences. The four primary users that we're targeting are:

- Budget-Conscious Shoppers: People who are highly mindful of their spending actively seek tools to help them stay within a strict budget.
- Young Adults and Students: This group comprises college students and young professionals beginning to manage their finances independently.
- Busy Parents: Parents who juggle multiple responsibilities appreciate tools that simplify their shopping experience.
- Impulsive Shoppers: People who struggle with impulse buying and often exceed their planned budget fall into this category.

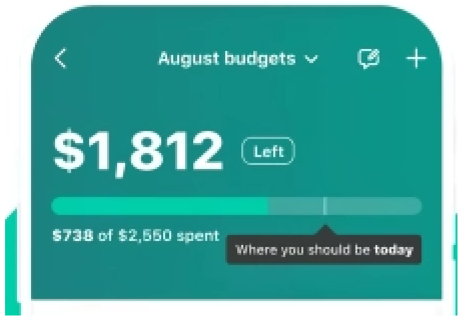
4.3. User interface

Home Page: This page will list all of the user's budgets. Each budget will include a name, a budget amount, an edit button, and a delete button. Below is a screenshot of another budgeting app that summarizes the purpose of our home page. As you can see, it displays a list of budgets for different purposes. In our app, the names would look more like "Grocery", "Game", "Book", etc. to indicate budgets for different stores/trips. Below the list, you can add a budget with the "Add Budget" button, which is also shown in the screenshot below.



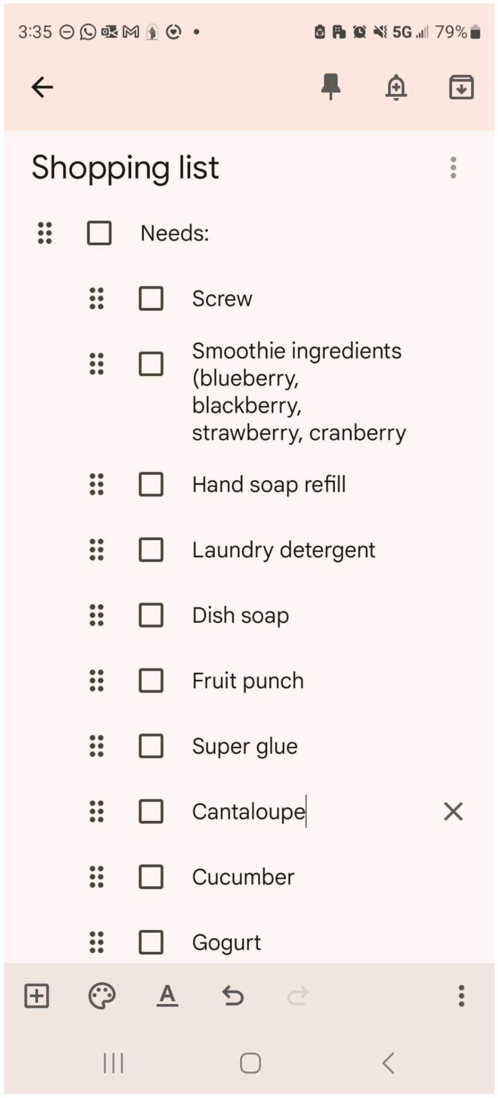
Rent	$ 1,400.00
Car	$ 500.00
Tacos	- $ 58.00
Utilities	- $ 150.00
Travel	- $ 759.00
Savings	$ 3,200.00
	$ 0.00

Budget Page: This page is where the core features happen. Here, the user can see their budget progress and their shopping list. Below is a screenshot to illustrate the budget portion. At the top, there will be the amount left in the budget. Next to it will have the total budget in smaller letters. This is shown nicely in the example below. There is also a progress bar in the example, which we may include if there is time.



The other half of the Budget Page will have the shopping list. This will be modeled very much like the Keep Notes checkbox lists, which a screenshot is shown below. This model is highly functional and intuitive. First, it is not shown below, but there will be a button to add an item to the list: one to add via the label scanner, and one to add yourself. On the list, you will be able to reorder the items with a “grip” bar next to each item. You can check off an item by hitting the checkbox next to the item. You can edit the text for each item. Finally, you can remove an item by hitting the X button that appears if you’re selecting it, each of which are shown below on the screenshot.

Key features that are not in the Keep Notes checkbox list include: 1. The price of each item. This will be editable, like the text. 2. A counter label for each item that shows the quantity of said item and a way to change it (either buttons or input box, probably the latter). All of these features for every item in the shopping list are beginning to stack up, so there may not be room to display every one of them on a single row in the list. It may be better to only display them when an item is selected, or even group them inside a menu button. We will see what solution we’ll need when we draw a wireframe sketch.

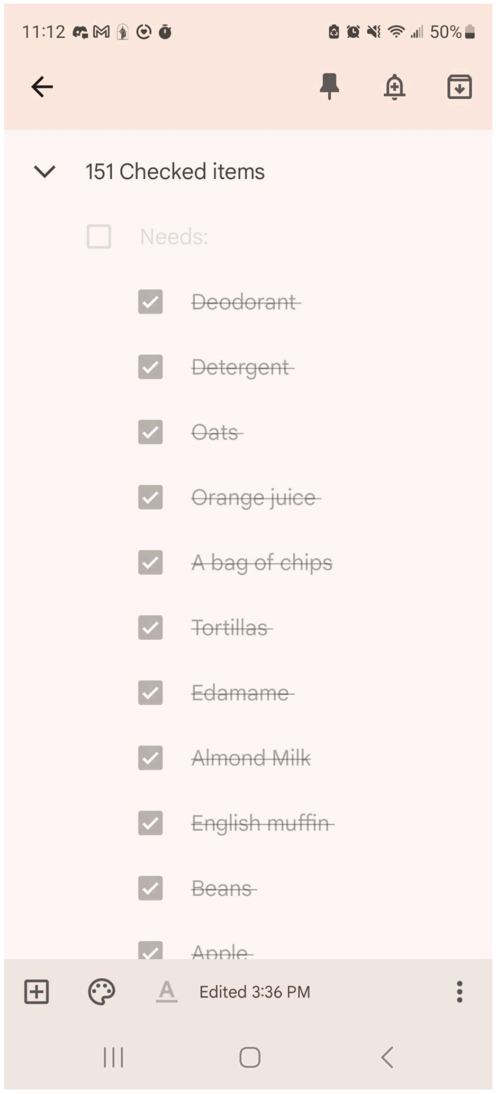


3:35 · 5G 79%

Shopping list

- [ ] Needs:
  - [ ] Screw
  - [ ] Smoothie ingredients (blueberry, blackberry, strawberry, cranberry
  - [ ] Hand soap refill
  - [ ] Laundry detergent
  - [ ] Dish soap
  - [ ] Fruit punch
  - [ ] Super glue
  - [ ] Cantaloupe
  - [ ] Cucumber
  - [ ] Gogurt

Every item in the shopping list has a checkbox. When you hit the checkbox, you’re saying the item is now in your cart, which means it will add the price of that item against your budget. A list of these items will be below your shopping list. An example of how this can be done is shown in the screenshot of Keep Notes below. The items are crossed and grayed out, so you don’t get confused with the items on your shopping list. From here, you can uncheck an item, and it will return to your shopping list and change your budget accordingly. You can also delete the items from this list, which will affect your budget, but the item won’t return to your shopping list; it will stay removed.



11:12 50%

151 Checked items

Needs:

- [x] ~~Deodorant~~
- [x] ~~Detergent~~
- [x] ~~Oats~~
- [x] ~~Orange juice~~
- [x] ~~A bag of chips~~
- [x] ~~Tortillas~~
- [x] ~~Edamame~~
- [x] ~~Almond Milk~~
- [x] ~~English muffin~~
- [x] ~~Beans~~
- [x] ~~Apple~~

Edited 3:36 PM

Label scanning: This will be handled via APIs, but if the option is there, we'd like to have a 'Scan' button while the camera is open to scan the label. When you hit 'Scan', it will take the photo, and predict the price of the item. If there are multiple prices on the photo, it could display each of them in a popup and let you choose which one to pick. Finally, if it predicts the wrong price, there will be a button to rescan the label, or if you wish, there will be a place to manually enter the price yourself.

5. Development Environment

Developer Tools:

The developmental tools that will be used during the development of this project are listed below.

- Visual Studio (version: 17.10.1)

    - .NET MAUI
    - IronPython
- Visual Studio Code (version 1.89.1)
    - Python

Content Generation Tools:

- Google Icon
- Figma

Data Tools:

- Local Data Storage

3rd Party Dependencies and APIs:

- .Net MAUI (version 4.7.2)
- IronPython (version 3.4.1)
- ML Barcode Scan Kit (Android) (version 4.2.1)
- Camera2 API (Android) (Support android 5.0 (API level 21) and higher)

Entity Relational Diagram

N/A

6. Production Plans

Code Review Plan

Requisites for Review

- All functionality is complete and working
    - Has been tested on end device
    - Needs no further revision
- Meets all coding standard requirements
- Functions are properly commented
- You will need to test as a team BEFORE Build Reviews.
- All completed work needs to be present on the Development branch unless there is a good reason for otherwise.

Integration Plan

Source Control

- When will branches be created?
        - Branches will be created per feature at the beginning of the project. One feature/one branch per developer and once the feature is complete a new branch will be created and assigned to the developer.
- When will they be merged?
    - Branches will be merged twice a week before the build review on Wednesdays and Sundays. The branch that will be used to merge into will be a testing branch.
- What are your major branches - Main/Production? Development? Staging?
    - There will be 5 major branches.
        - Main branch.
        - Test_Main_Branch.
        - MVP (minimum viable product Feature A).
            - This branch will have a testing branch and features will be a branch.
        - Alpha_branch (B Features).
            - This branch will have a testing branch and features will be a branch.
        - Beta_Branch (C Features).
            - This branch will have a testing branch and features will be a branch.

Testing Plans

Weekly Code Reviews

- The Day before Build Reviews with Mentor/Producer:
    - 1-2hrs - priority one
        - All team members will test all code completed during the work week
    - 15-30 min complete/incomplete discussion

Bug Database

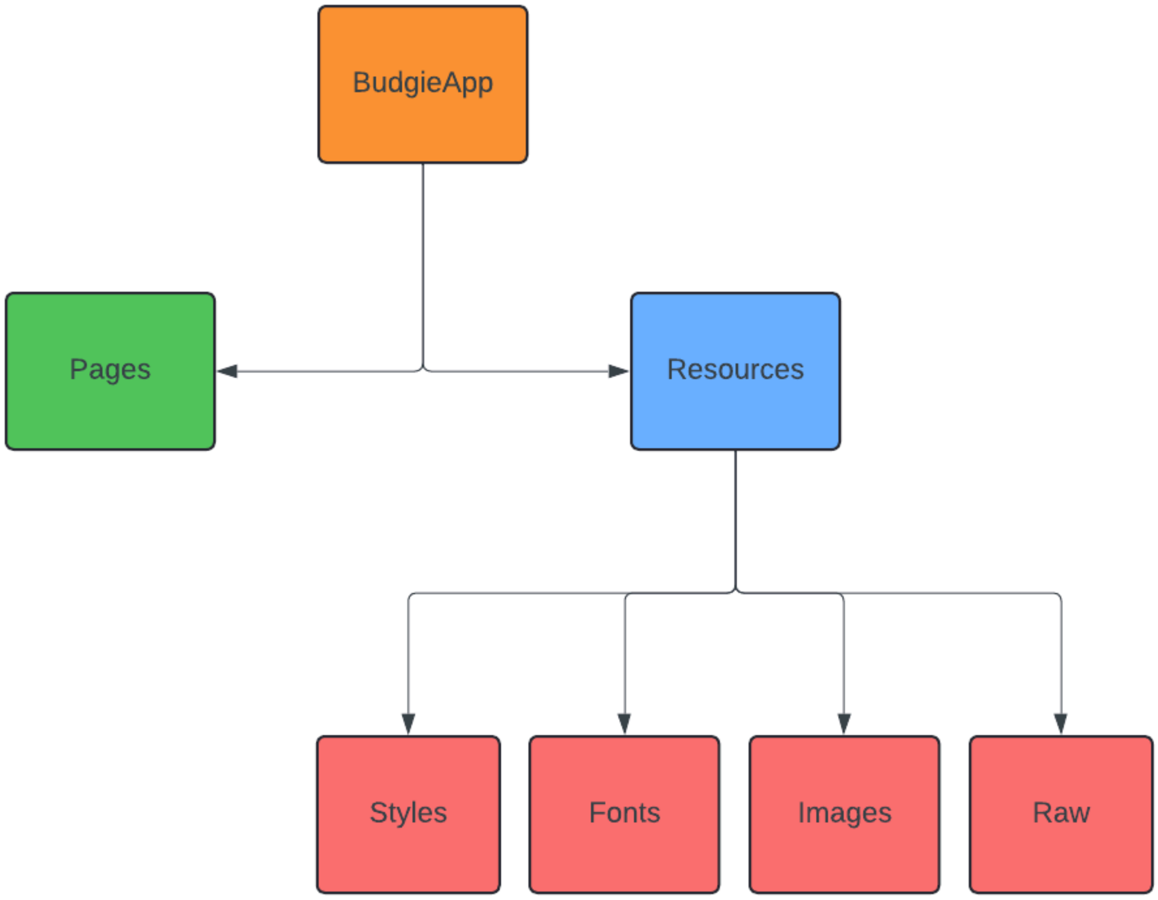
- Maintained and updated by the team members
- Software
    - Jira

Outside Testing

- Month 4 of Capstone
- 5 Testers selected
- Crosstesting will be expected on a monthly basis with other teams in CS and GD
- A Testing Report Compiled
    - Scenarios planned
    - Findings

- Surprises
- Planned Solutions

7. Folder Hierarchy


BudgieApp
Pages
Resources
Styles
Fonts
Images
Raw