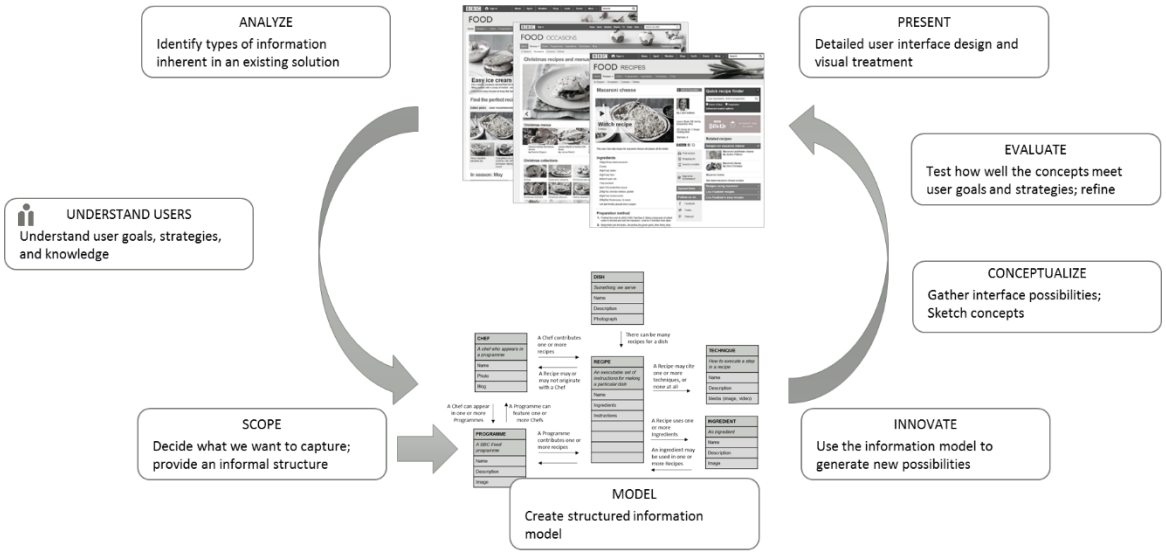
THE INFORMATION ARTICHOKE .COM

Information Access

A UNICORN BRIEFING NOTE



Martin Stares

June 2015

Information Acess

A UNICORN BRIEFING NOTE

Series introduction

This Briefing Note is part of the series "Experiencing + Architecting Information". The series provides intermediate solutions designers with polished and practical insights into designing information rich systems.

It draws upon insights from information architecture and user centered design. As such, it will be useful to front end designers who need to know more about information, and information architects and analysts who need to know more about users.

The series is introduced in the Briefing Note entitled "Experiencing + Architecting Information", where we detail the approach and coverage. It might help to read this first. The remaining briefing notes drill more deeply into each topic, and for the most part can be read in any order.

We sincerely hope this series will help you gain in skill and confidence in our wonderful profession.

Martin Stares
The Information Artichoke

June 2015

Information Acess

A UNICORN BRIEFING NOTE

Introduction

This Briefing Note explains how to design methods for accessing information that have a high degree of utility and usability.

In the series introduction "Experiencing + Architecting Information", we introduced two views of a user-facing solution, the user view and the architect view.

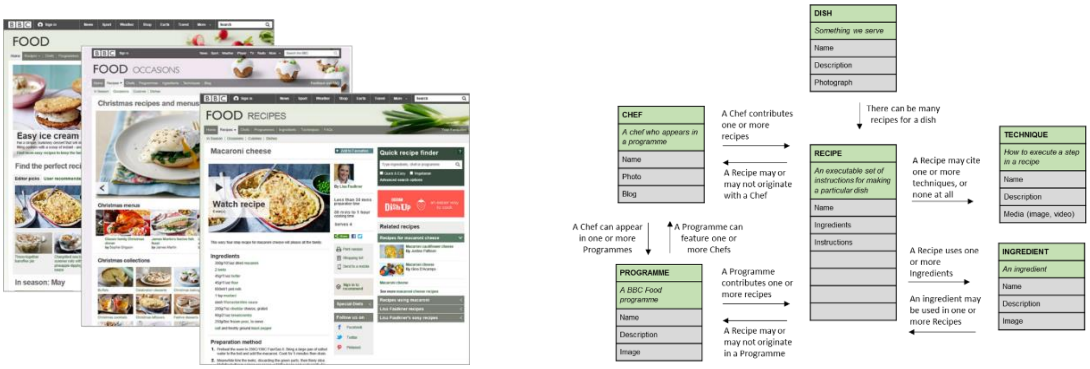


Figure 1: User and architect views of a food site

The architect view abstracts away the details of a particular user-facing solution, leaving a compact representation of what's going on from an information point of view. The resulting information model is a convenient tool for matching user goals to information structure and generating ideas for new functionality.

The same is true of access methods. The user view incorporates navigation structure, query refiners, filters, etc. allowing the user to slice and dice information, but is not a convenient way to see quickly what's going on from an information access point of view.

We need something like an architect's view of access methods, complementing the information model. Let us call it the **access model**. Creating an access model is the topic of this Briefing Note.

You should be comfortable creating and reading information models. See "Information Analysis" and "Information Modelling" for details.

First look at an access model

The access model is a structured way of describing access to the information in an information model.

Consider a consulting practice that contains information about its projects, practices and locations. Here is one possible access model for its public facing site.

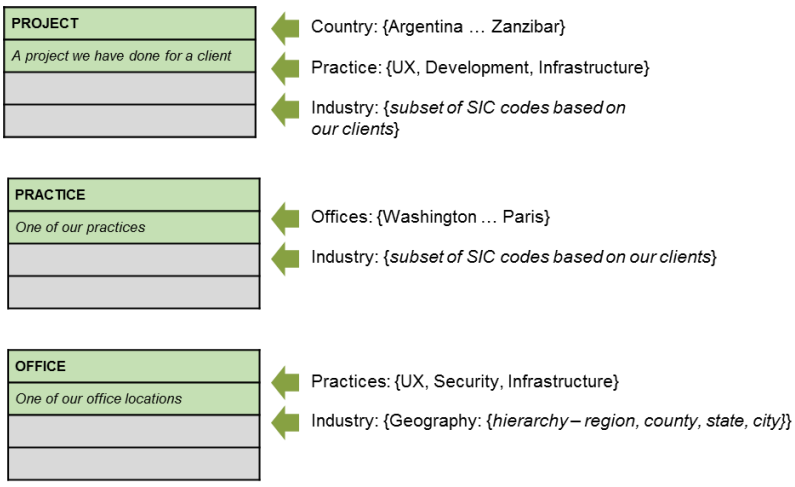


Figure 2: Access model for a consulting group with projects, practices and office locations

Like the information model, it is an abstract representation with a simple visual vocabulary, which you will learn in the next sections, if it is not apparent from the example.

It shows the ways we propose to access information in the system. It is definitive – it is the complete set of ways that this solution will allow the information to be accessed; no other access methods are proposed. On the other hand, it is only one of many possible access models for this information set. A different set of users may want to access the information in quite different ways, yielding a different access model.

Relationship to the user view

There is a one-to-many relationship between the access model and the user interface. An access model can be implemented as custom views, search refiners, filtering, etc. For example, the access methods for Project allow many UI implementations.

- Pre-built views
  - Users can view projects by country, projects by industry
- Advanced query
  - User can look for Infrastructure projects done in Argentina for Oil and Gas Extraction
- Filtering
  - Users can narrow down large result sets by “refiners” in the left navigation pane.

The user interface choices depend on the user goals and size of the information set.

The pieces of an access model

The basic unit of an access model is an **access method**. Here is an example.

Figure 3: Example access method

It consists of three elements:

- The information component being accessed (Project)
- The access method or access scheme (access by Country)
- The allowed values of the access method (a named list of countries).

This three-part model allows a systematic and structured exploration of access methods.

- Information component: look at all components in an information model in turn to ensure we explore all opportunities
- Access method: use a combination of information structure and user goals to explore access methods that might be **useful**
- Allowed values: use a combination of information structure and user goals to make the access method **usable**.

We explore these more deeply in the following sections.

Getting access ideas from the information model

(a) Use component attributes

Sometimes attributes of an information component provide a basis for access schemes. For example, consider the information component for an invoice coming into to our company.

INVOICE SUMMARY	***Illustrative Instance***
Summary of an invoice	
Vendor name	*Interactive Office Supplies*
Invoice number	*1274*
Invoice date	*2014-08-26*
Description	*Digital whiteboard*
Total amount	*$2,600*

Figure 4: Information component for invoice summary

Our accounts payable clerks might like to see Invoices organized by Vendor and Invoice Date, or search for them by Invoice Number. They might not be as interested in organizing them by Total Amount. So in this case there are already attributes that can be used for organizing or filtering information.

This specification is pretty wordy. We can document it visually as follows.

INVOICE SUMMARY	*Access Methods*
Summary of an invoice	
Vendor name	*Organize / search for invoices by vendor*
Invoice number	*Search for an invoice by its number*
Invoice date	*Organize / search for invoices by date*
Description	-
Total amount	-

Figure 5: Access possibilities for invoice summaries

With this in hand, we can consider all attributes, to see if other possibilities come to mind. In this case, we didn't have any ideas for structured access by Description; we used the notation "-" to show that we at least considered it.

The diagram is a more compact representation than the verbal description, and also facilitates systematic exploration with stakeholders. Once we have finished our explorations, we can document the decisions as a formal access method diagram.

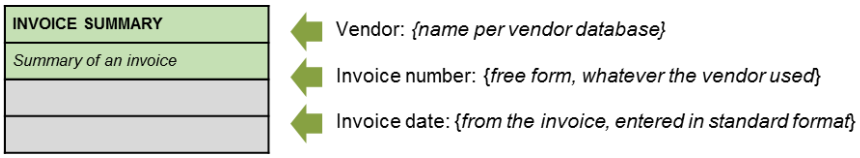


Figure 6: Access methods for invoice summary

We have also specified the allowed values.

Note that we do not list the attributes. For one thing, the full set of attributes is in the information model. For another, there is a tendency to waste time trying to line up access methods with attributes. Finally, not all access methods correspond to attributes of the component being accessed.

What about search?

For environments that support full text search, we can always provide search based solutions, for example letting the users search for invoices whose description contains "whiteboard". We regard this simple type of search as a platform capability rather than an access method we design, and will not document it in our access methods. Some platforms such as SharePoint 2013 encourage search-based solutions that the principles described in these Unicorn Briefing Notes to scope, refine, and present search results in smarter ways that a simple list of unconsidered search results.

(b) Use related components in the information model

The relationships between components in an information model indicate other access opportunities.

Here is the information model for the consulting group that we have used in other briefing notes

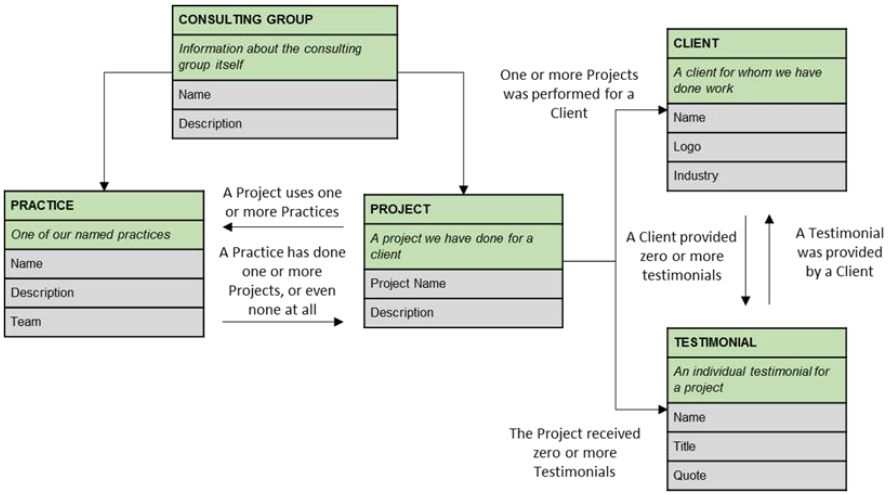


Figure 7: Information model for consulting group

Looking at Project, we can see access methods based information components that are related to it.

- By Client - " list all projects we have done for Greyfriars Consulting Group"
- By Practice - "list all projects that used the User Experience practice".

We needn't restrict ourselves to the name attribute of the other related components; you can use any attribute that makes sense, and even include processing too.

- By Industry – "list all the projects we have done for aerospace"; we can do this because all projects were done for a single client, and all clients belong to a single (let's assume) industry, so we know the industry associated with a project.
- By Number of Testimonials – "group projects by number of Testimonials received"; "show management which projects received no testimonials"; we can do this because of the direct relationship between Project and Testimonial.

We can document these in access method format.

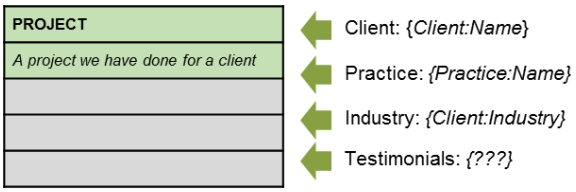


Figure 8: Access method based on information model relationships

Note how we used qualified names (Component: Attribute) for the allowed values. The allowed values for client's name, practice name, and client's industry get designed when we define the Client and Practice components. This part of the design work does not need to be done again here, and we simply refer to it using the qualified names Client:Name, Practice:Name and Client:Industry respectively.

(c) When attributes and relationships are not enough

In many cases, the attributes and relationships in an information model are not sufficient to provide all the access and filtering that might be useful, and additional attributes will need to be added.

Here is the information component for a Recipe as it might appear in your personal recipe card index.

RECIPE	***Illustrative Instance***
An executable set of instructions on how to make a particular dish	
Name	*Yorkshire Pudding*
List of ingredients and quantities	*3 eggs* *1 cup milk* *1 cup all-purpose flour* *2 tablespoons butter*
Instructions	*Preheat oven to 375 degrees F (190 degrees C). In a medium bowl, …*

Figure 9: Basic component for a recipe

This is the minimum information needed to make a Yorkshire pudding; without these attributes, you could not make the dish.

The only attribute that could be used for organization is Name, giving an alphabetical list of recipes. This may be workable for a personal card index, but not so if you are creating a competitive Food Site with hundreds of recipes and hoping to attract thousands of visitors with different goals.

While some visitors might be looking for Yorkshire Pudding by name, others might remember a tasty British accompaniment to roast beef, but don't know the name Yorkshire Pudding.

To make it easier for people to find a recipe, additional information needs to be added, not strictly part of the recipe.

Here is the recipe with additional information added, typical of what you find in online food sites.

RECIPE	
An executable set of instructions on how to make a particular dish	
Name	
List of ingredients and quantities	
Instructions	
Metadata	***Illustrative Instance***
Description	*A tasty accompaniment to roast beef*
Cuisine	*British*
Type of dish	*Accompaniment*
Course	*Main*
Special occasion	*Everyday*
Dietary notes	*Contains wheat, dairy*

Figure 10: Recipe with metadata

This additional information is called METADATA. It is information ABOUT the recipe rather than PART OF the recipe. In fact, metadata is often explained as being "data about data".

The metadata (Cuisine, Dietary Notes) doesn't change the way that we prepare the dish, but exists to provide access methods to the set of recipes.

That said, attributes and metadata both appear as rows in an information component, and in practice we often omit the label "Metadata" in the information component.

Getting access ideas from the user

Just as our understanding of users' goals, strategies and knowledge shapes the information model, so it also shapes the access model. To help frame our considerations, here is a simple model of user information access behaviour.

- User has a goal
- User accesses information through an access method (could be a pre-built view or search)
- User operates on results
    - Manageable number of results – user scans list for item(s) meeting goal
    - Too many results – user filters (subsets) the results
    - Too few results – user broadens search
    - No results – user abandons this access, looks for another.

To make this concrete, you could imagine the scenario of a user of a large food site looking for a nice British-themed Sunday dinner.

For each of these, we can imagine design objectives for information access.

- Provide access methods that allow users to start meeting their goals
- Provide information scent to help users scan efficiently
- Provide methods so that users can narrow down a set of results
- Provide methods so that users can broaden a set of results.

More powerfully, they correspond to questions that you can ask users of the proposed solution. For example, if we were building a corporate employee directory, we might ask our HR informants questions like this:

- Q: “If we pre-build views, how would you like to access employees?”
- A: “By name, by hire date, by date of last review, by office location”

- Q: “If we display a list of employees for you, what would you like to see for each one in the list (so that you don’t have to open the record)?”
- A: “Name, contact information, office location, photo, most recent project …”

- Q: “If your query gives a large number of employees, how would you like to narrow the list?”
- A: “Search by name is large: filter by “Employee/Consultant”, filter by office, filter by department”
- A: “Search by hire date is large: filter by starting month”

- Q: “If your search result is too small, how would you broaden your search?
- A: “If I am searching /filtering by office, and I don’t find who I am looking for in the San Francisco office, it would be nice to extend the search to Southern California, then the West Coast”
- A: “If I am searching /filtering by hire date, and I don’t find who I am looking for in the month I entered, say January 2015, I would like to extend my search a month either side of that, and be able to repeat as needed”

The questions work well as a set; the answers to later questions might refine the answers to earlier questions.

The answers shape the access model needed for the solution. But it is a good idea to do some lightweight prototyping with the user to validate their responses.

Aligning user requirements with the information model

We always have to cross-check user requirements against the information model.

Sometimes an information model does not support a desired access method. Here's part of the consulting group information model.

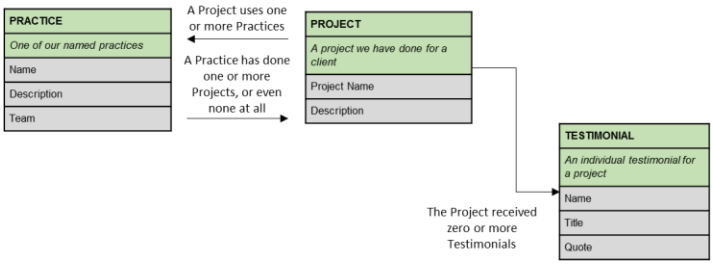


Figure 11: Part of the consulting group information model

Suppose we want to answer the questions.

- "Show me which practices have received the most testimonials"
- "Which practice received this testimonial"

We cannot tell this from the information model. Can you see where the problem is? Things look promising to start with, as there are paths between Practice and Testimonial, via Project. The problem is in the multiplicity "A Project uses **one or more** practices". Although the project as a whole received a testimonial, we cannot tell which practice earned the testimonial.

If we need to support this requirement, we will have to change the information model to include a direct relationship between Testimonial and Practice.

Allowed values

Now let us turn to the allowed values for an access method.

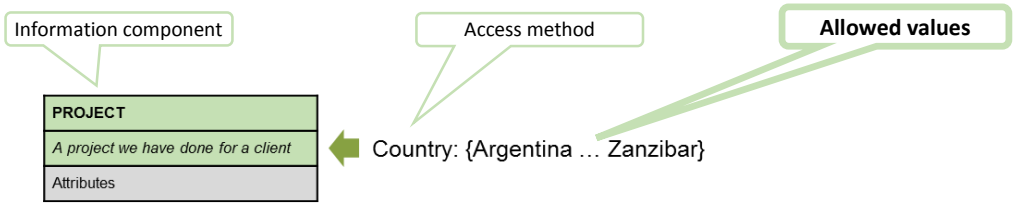


Figure 12: Example access method

The allowed values are a specification that provides consistent query and filtering terms for users as they follow their information seeking strategies. It helps to consider two distinct areas.

- Structural factors – how well the terms logically organize or filter information.
- Usability factors – how well the terms are, or could be, understood by the user.

Without considering both, the allowed values could be great for dividing up the information space, but that are unrecognizable to the user; conversely, we could have a set of terms that the user understands clearly, but that do not do a good job of dividing up the information space.

There is no definitive list of principles or best practices, but rather a set of considerations that might be more or less valuable in particular situations.

Structural factors

Structural factors refer to the logical structure of the set of terms and how well it divides up the information space.

(a) The allowed terms should be of the same type

For example

> Country: {Argentina … Zimbabwe}; the terms are all names of countries
> Practice: {UX … Infrastructure}; the terms are all names of practices

Sometimes we see top level site navigation like this:

> Area of Website: {Corporate, Operations, Financial, Legal, Contact Us}; the terms do not form a homogeneous set

From an information architecture point of view, this is something different than we have discussed. It is an example of Curated Content, where somebody pulls together useful subsets of a large body of content, sometimes of the same type, sometimes not, and provides a new level of classification.

(b) The allowed terms should form a complete set

For example:

> Practice: {UX … Infrastructure}; the terms are names of **all** of our practices

For countries, the word “complete” could mean different things.

> Country: {Argentina … Zimbabwe}; all the countries in the world
> Country: {Australia … United States}; countries where we do business

We often create flags to identify a special subset, with asymmetrical access

> SpecialPeople: {VIP}

In this case, we want to explicitly identify VIPS for special handling, but never need to select non-VIPs.

(c) Avoid or refactor structured terms

The allowed terms in

Type of Recipe: {Chinese Main Course … Chinese Dessert … Indian Main Course … Indian Dessert}

are in fact a combination of two other access methods, each of which should be part of the core design

Cuisine: {Chinese … Indian}
Course: {Main … Dessert}

You see this a lot in flat structures such as paper files or file folders.

Filing Category: {2010 - North America - USA – Financial – Banking … 2015 - South America – Brazil – Engineering - Civil}

Organizations often call this their "Hierarchy" but this is incorrect; in a true hierarchy, each level is a strict subset of the one above. When moving to a platform that supports metadata, you have the option of factoring this into multiple access methods.

Year: {2010 …}
Location: {*hierarchy: Region > Country*}
Industry: {*hierarchy: Sector > Industry*}

(d) Avoid or manage overlap

The allowed terms in

Cuisine: {Chinese … Lebanese … Middle Eastern … Sichuan}

overlap but in a structured (hierarchical) way that can be exploited in the user interface if we redesign the interface.

Cuisine: {*hierarchy: Region > Country*}

Take another look at the top level site navigation.

Area of Website: {Corporate, Operations, Financial, Legal, Contact Us}

The terms overlap, and so probably does the content. This is not necessarily a problem; rather than help a single user navigate between areas, we are here designing for multiple personas and providing a way for them to choose their preferred view of the site.

(e) Avoid or manage duplication

The allowed terms in

Institution: {Bank of Montreal, Royal Bank of Canada, BMO, RBC}

contain duplicates.

In this case, the duplication is in the form of an abbreviation. We can consider two access methods that do not contain duplicates:

Institution: {Bank of Montreal (BMO) …}
Abbreviation: {BMO …}

User testing will determine the best approach for the particular access method.

Usability Factors

Usability factors refer to how well the terms are, or could be, understood by the user.

There is often a gap between what the user knows and how you organize information; you will have to consider whether the user will understand your terminology, and whether it matters. Here are some examples:

- The HR department sees a clear distinction between policies and procedures and practices; end users regard them all as documents which they consume in the same way, so we don't present the distinction to end users.

- The HR department sees a clear distinction between policies and procedures; end users tend to regard them all as documents which they consume in the same way; HR wants them to learn the difference, as there are different requirements on the two types (policies have to read and accepted); we present the distinction

- A food site for the general public wants to organize recipes by country; it knows that the general public doesn't know all countries of the world, and are not prepared to make the effort, so buckets some of the less familiar ones into groups (West African, Scandinavian)

- A food site for foodies wants to organize recipes by country; it knows that foodies don't know all countries of the world, but are prepared to make the effort, so lists them all anyway; it provides a hierarchy and maps to help the users understand where a country is located

- A DIY store lists all of its products by names such as "2 inch zinc corner brace"; the beginning DIY person describes it to the floor staff as a "a right angle bracket, about this big, to strengthen the corner of a drawer"; the store will not use the users descriptions but provide a hierarchical structure with terms such as Hardware > Brackets to provide orientation.

There are also perceptual considerations, such as being consistent with formats (not mixing “John Smith” and “Smith, John” in a same list), using similar linguistic constructions (not mixing “Getting help with ABC", "How to get help with DEF”).

What’s next?

This material has introduced some ideas about information access. In particular, it shows how knowledge of the solution’s information structure and its intended users can shape the access provided.

The visual notation of an access model complements that of the information model, and provides a structured and systematic way to explore the quality of a solution’s access methods.

As part of the “Experiencing + Architecting Information” series, this Briefing Note “Information Modelling” has many touchpoints to other documents in the series.

- Information Modelling – a pre-requisite, showing how to generate the information model which the access model is built upon
- Understanding Users – explains the notions of user goals, strategies and knowledge, which get considered when designing access methods
- Innovation – shows how to use an information to explore user access possibilities; explores relaxing constraints in the model to generate brand new possibilities.
- Conceptualizing – shows how the various aspects of an access model are reflected (one:many) in user interface constructs, and demonstrates this with wireframes.

Good designing!