**Слайд 2: L**

Is it a pleasure for you when you after difficult day have to think what product and where you have to buy in order to make supper? Definitely not.

However, you do not have exit. You have to do it and do not have any chances that this problem can be solved by itself.

**Слайд 3: P**

Our team consists of 4 members:

**Pavlo Reva** – software engineer and tester. He developed components according to requirements and covered them with unit tests.

**Denys Saukh** – system architector.He designed the structure of the system, came up with the relationship between the components.

**Слайд 4: D**

**Lesia Volobuieva -** software engineer and tester. She developed components according to requirements and covered them with unit tests.

**Olha Bahno –** business analyst. She collected all the requirements, broke them down by tasks, she was an intermediary between the customer and the developers.

**Слайд 5: O**

***Problem:*** high cost of food, not the quality of food (or lack of quality).

***Affects:***

* people of all age categories whose interest is product quality
* people of all age categories whose interest is the cost of the product
* people of all ages who buy food from supermarkets.

***The result of which is***: overpayment for the food product, wasting time on finding a quality and affordable product.

***Winnings from:***

create a mobile application that allows you to quickly find information about a product, study the pricing policy in supermarkets regarding this product, and also determine in which nearest (relative to the user) supermarket of the city you can buy this product.

***May consist of the following:***

* The ability to quickly find out all the necessary information about the product
* Reducing the amount of time spent on finding a quality and affordable product

Simplicity, ease, efficiency and availability of use of the product by a large number of users.

**Слайд 6: L**

ProFinder is a conceptually new product on the food area in Ukraine. It really has no analogs in Ukraine and usage of this will like the step in future. This document provides basic knowledge why you should review our product as one that worth working with.

**Слайд 7: P**

Every day people face with problems of food with quality that does not match to its price. It is too hard to find really food, which will have good quality and corresponding price in the same time. Our product can solve the problem.

**Слайд 8: D**

ProFinder consists of three modules: user part, market part and admin panel. User will be able to find what he really wants. In addition, he has opportunity to give feedback to some market and it will have influence for quality of the products. Market owner has to regulate prices and discount offers via ProFinder. Moreover, admin from admin panel can regulate relationships of buyer with a market. User will have never seen markets with bad goods and high prices. All types of the users will have benefit – usual users will have good-quality products for a corresponding price.

**Слайд 9: O**

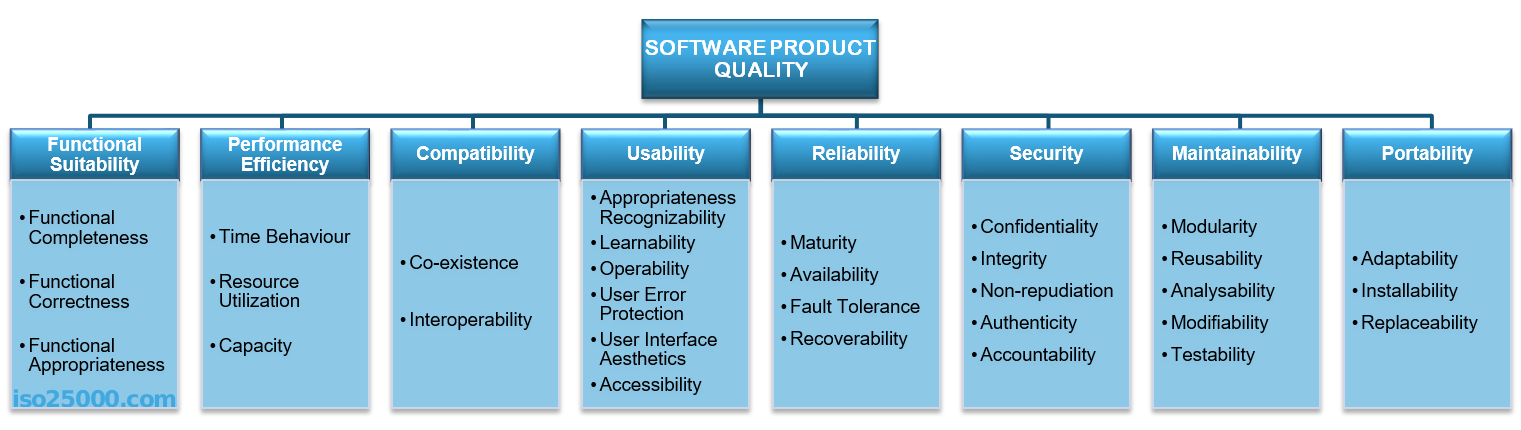
The store is one of the main actors of the system, as the developed application assumes the creation by the buyer of the rating of stores. In addition, the store itself provides the necessary information about the goods in case of interest by this application.

The buyer is the main actor of the system. It is for its use that this project is being developed. The buyer will be available all the functions of the application, which will improve and accelerate the process of buying the desired goods in the store.

The programmer is the creator of the application. It interacts with the application most often: both at design time, and as a potential user, and after the release of the project in order to improve it.

**Слайд 10: L**

To consider the types of quality attributes, we can use a diagram from ISO 25010:



This standard describes the quality attributes for a software product. Next, we’ll look at what exactly each attribute means individually.

**Слайд 11: P**

**Performance** shows the response of the system to performing certain actions for a certain period of time.

There **are two ways how to measure performance**:

* **Latency:** Time spent on responding to an event
* **Channel capacity**. The number of events that occur at a certain point in time.

In practice, **the possible performance indicators** include, for example:

* Average/maximum number of system users per time unit.
* Average page load time.
* Average method execution time.

Performance issues very often grow into problems that can affect everything, from the server’s capacity or the ways in which you develop your front-end to the efficiency of database queries or the capacity of communication channels.

Performance is almost always included in the list of key quality attributes that need to be considered by the architect, since it affects the entire system and can affect many parts of the architectural solution. Therefore, on the internet, you can find a large number of examples of how to deal with performance problems.

**Слайд 12: D**

**Usability** is one of the most important attributes, because, unlike in cases with other attributes, users can see directly how well this attribute of the system is worked out. One of the key problems of usability is too much interaction or too many actions necessary to accomplish a task. Incorrect sequences of steps in multistage interfaces are also a problem of usability. Data elements and controls may be designed not according to the accepted patterns of user experience, which also complicates the interaction. For example, if you are developing an iOS application, then it is important to use [the guidelines from Apple](https://developer.apple.com/ios/human-interface-guidelines/overview/themes/), or [the guidelines from Microsoft](https://msdn.microsoft.com/ru-ru/library/windows/desktop/dn688964%28v=vs.85%29.aspx) — for Windows desktop applications.

Examples of important indicators for this attribute are:

* List of supported devices, OS versions, screen resolutions, and browsers and their versions.
* Elements that accelerate user interaction, such as “hot keys”, “lists of suggestions”, and so on.
* Average time a user needs to perform individual actions.

Support of accessibility for people with disabilities.

**Слайд 13: O**

Reliability is an attribute of the system responsible for the ability to continue to operate under predefined conditions. Most often, the system fails due to the inaccessibility of external elements, such as databases, systems, and network connections.

**Слайд 14: L**

Availability is part of reliability and is expressed as the ratio of the available system time to the total working time. Important indicators for this attribute are:

* Availability.
* Planned downtime.
* Time needed to update the software, and so on.

**Слайд 15: P**

Security is responsible for the ability of the system to reduce the likelihood of malicious or accidental actions as well as the possibility of theft or loss of information. There are a number of measures that are used to protect systems: authentication, encryption, audit, and others.

Examples of this attribute in the work of the system are:

* The ability of the system to detect DDoS attacks and respond to them.
* Restrictions of user access in accordance with authentication/authorization.
* Prevention of SQL injection.
* Encryption of passwords and content.
* Secure connection.

**Слайд 16: O**

Supportability is the ability of the system to provide useful information for identifying and solving problems. The main problems in ensuring supportability can be addressed with the following means:

* No diagnosis: How the activity and performance of the system are controlled. This includes various types of logging.
* No tools for troubleshooting: This includes backups, various systems for creating snapshots of the system, and tools for auditing the system. When the system fails, it is always more pleasant to wait for an automatic restart than to solve the issue manually.
* No health checking: This includes a variety of systems for measuring compilation time, deployment time, database size, or mobile application size.

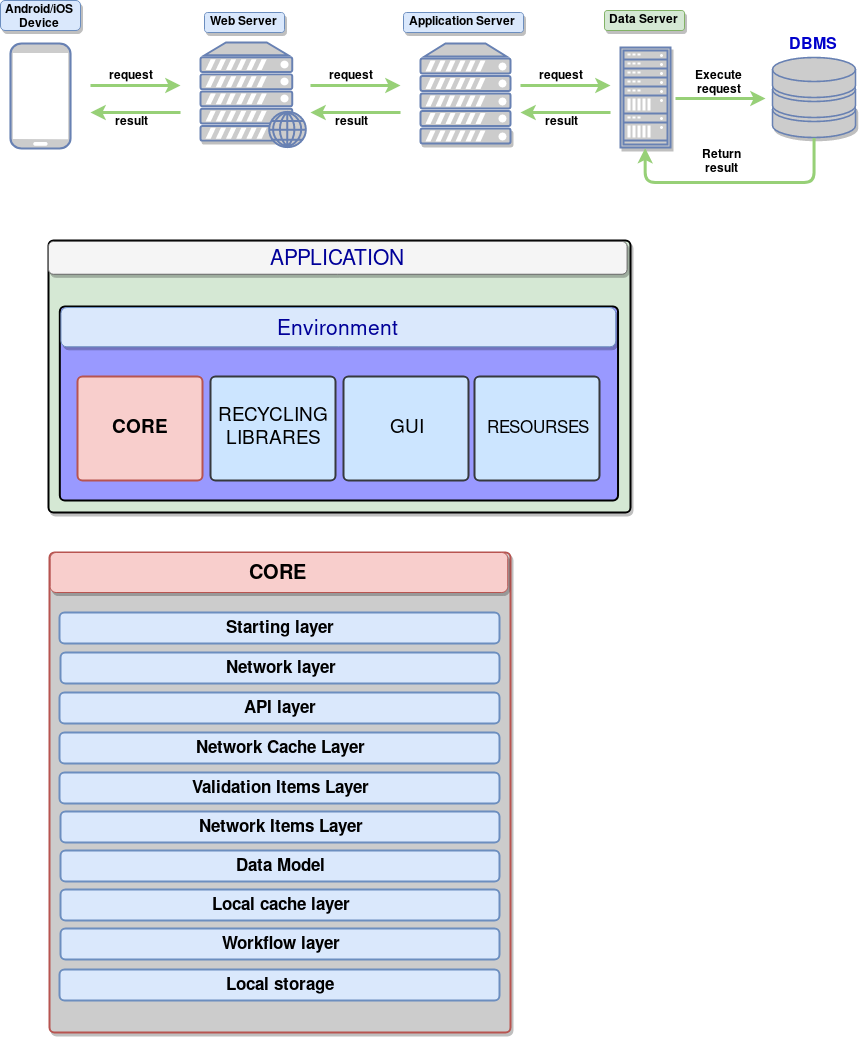
Most often these are not considered in start-ups or small projects initially. The cost of maintaining the supportability attribute is high, and the result is only visible on a large scale. However, with the growth of the team and the product, this attribute becomes one of the key ones.

**Слайд 17: D**

Utility trees are a way to organize these quality attributes. In regard to ATAM they serve as a way to prioritize quality attributes and later to evaluate the suitability of a candidate architecture vs. the requirements. I also like to use them as reference requirements and stories (or themes depending on size) as part of a backlog in agile development.

**Слайд 18: L**

Based on the requirements of the system, it was decided to use two-tier client-server architecture.



The advantages of this architecture, in our system are:

* Significant reduction of network traffic.
* Reduces the complexity of the client applications (most of the burden falls on the server side), and hence, reduced requirements for hardware facilities of the client machines.
* A special software tool - SQL-Server - leads to the fact that a substantial part of the design and programming tasks is already solved.
* Significantly increases the integrity and security of the database.
* A two-tier architecture is simpler, since all requests are serviced by the same server.

**Слайд 19: P**

There is a structure of mobile application.

The core of application includes the following:

1. Starting layer. Defining workflow, program execution begins.
2. Network layer. Provides the transport mechanism for interaction.
3. API layer. Providing a unified command system of interaction between client and server.
4. Validation Layer. Validation data, received from the network.
5. Data Model. The data model for the interconnection of data entities.
6. Local cache layer. Local caching layer that provides local access to network resources that are already received.
7. Workflow layer. Including classes and algorithms specific to the application.
8. Local storage.

**Слайд 20: O**

As we said earlier, we have independent layers in our architecture. There are 5 main layers:

**View layer** – user interface layer, all frontend part of the project, different ui controls, consist of 3 parts – web-based client, ios client, android client.

**Service layer –** interlayer between view and business logic layers. It has one main function – to transfer data between client and server.

**Business logic layer –** part of system, which encapsulate all hidden business logic, it is responsible for all calculations; prepare different statistic data and other hard work, which require large computing power. This function executes remote server.

**Data access layer –** interlayer between business logic layer and directly DB. It transfers data from DB to business logic layer to execute some calculations and operations.

**DB –** storage of user’s data and information about markets.

**Слайд 21-22: D**

There’s one important document - Vision doc. The purpose of creating this mobile application is the need for qualitative products and the solution of the problem of minimizing the time of searching for quality food products. The version of the software product at the time of writing the concept document is 1.0.1.

Main users of the system:  
• Stores products with good reputation and product quality.  
• A variety of plate-based buyers with a mobile device.

**Слайд 23-24: L**

This document includes changes in comparison with the previous version of both the concept document and the software product itself. It was added features such as feedback from the store via chat, translation of messages inside the application for foreign buyers / shops, a map with the location of stores, adding a rating for the store. The version of the software product at the time of writing the concept document is 1.1.0.

**Слайд 25: P**

What do we use in order to bring your idea into life?

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

Swift is a robust and intuitive programming language created by Apple for building apps for iOS, Mac, Apple TV, and Apple Watch. It’s designed to give developers more freedom than ever. Swift is easy to use and open source, so anyone with an idea can create something incredible.

React is a library for helping developers build user interfaces (UIs) as a tree of small pieces called components.

**Git** is a [version-control](https://en.wikipedia.org/wiki/Version-control) system for tracking changes in [computer files](https://en.wikipedia.org/wiki/Computer_file) and coordinating work on those files among multiple people. It is primarily used for source-code management in [software development](https://en.wikipedia.org/wiki/Software_development), but it can be used to keep track of changes in any set of files.

**Слайд 26: O**

The Architecture Tradeoff Analysis Method (ATAM) is a method for evaluating software architectures relative to quality attribute goals. ATAM evaluations expose architectural risks that potentially inhibit the achievement of an organization's business goals. The ATAM gets its name because it not only reveals how well an architecture satisfies particular quality goals, but it also provides insight into how those quality goals interact with each other—how they trade off against each other.

The ATAM is the leading method in the area of software architecture evaluation. An evaluation using the ATAM typically takes three to four days and gathers together a trained evaluation team, architects, and representatives of the architecture's various stakeholders.

**Слайд 27: D**

**What are the outputs of ATAM?**

– A set of **architectural approaches** identified and or applied: Sometimes we can identify architectural approaches that cannot be applied on our architecture

– **A Utility Tree**: a top-down mechanism for directly and efficiently translating the business drivers of a system into concrete quality attribute scenarios.

– A **set of scenarios** identified and the subset that had been effectively mapped in the architecture.

– A **set of questions about the quality attributes** in the architecture and the answers to these questions. In our case our questions are a set of metrics and the answers are the values measured.

– The **risks identified**: risks that the architecture is able to mitigate and the risks that threaten the system and the business goals.

**Слайд 28: L**

This is a prototype of our system. Сказать типо, что это наброски. Это поможет сделать наш дизайн лучше и интереснее. А также, облегчит работу фронтенд разработчикам.

**Слайд 29: P**

Friendly design could help you to not only enjoy and get a pleasure, but also to feel comfortable during working with our application.

**Слайд 30: D**

Сказать, что есть гит и по всем по вопросам, если они есть Вы можете обращаться на гит – смотреть его. Искать, скачивать

**Слайд 31: O**

Если есть вопросы – вот все контакты