Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

Курсова робота

за спеціальністю „Інженерія програмного забезпечення”

Architectural approach in Software development. Research and technology stack decision.

Керівник курсової роботи

Ас. Корнійчук М.А.

_____
(підпис)

“____” _____ 2021 р.

Виконав

студент 4 курсу

факультету інформатики

Кузьменко Д. О.

Work Schedule

Number	Coursework writing stages	Date
1	Team collaboration and research of the judicial domain	10.09.2020-17.09.2020
2	Technological stack decisions	25.09.2020
3	Development stage	01.10.2020 - 28.02.2021
4	Product testing	01.03.2021 - 08.03.2021
5	Completed coursework draft	29.03.2021
6	Completed final coursework draft	11.04.2021

Table of Contents

Introduction

The topic of architectural approach in software engineering is chosen for the research in this work. The topic is undoubtedly relevant at the moment. The further we progress technologically, the more requirements are set for new projects centered around the software. Also, more projects are being created, more products are being designed, and more diversified approaches can be assessed and chosen for many different tasks.
Domain research and technology stack decisions do play a vital role in the potential risk avoidance and success of the project.

This work will comprise of different segments, such as the **theoretical part** of software development architectural approach - architectural patterns, different methodologies to the development itself, server and client sides of the application as necessary parts of any product, core solutions to the set tasks, and solutions to continuous integration and delivery - are to be explored, assessed, compared and explained; **the practical part** - in which the team project is evaluated, the development process inside the team is shown, and the general structure of the project, as well as the decision per every point mentioned in the theoretical part, is explained; **the demo part** - involves demonstration snapshots

from the working application, deployed at https://themida-devs-kma.herokuapp.com/.

Our team consisted of 5 members:

Dmytro Kuzmenko (Software Engineering 4) - Backend developer/Product manager

Tetiana Meronyk (Software Engineering 4) - Backend developer

Kryvosheienko Julia (Software Engineering 4) - Frontend developer

Simon Boytcov (Software Engineering 4) - DevOps engineer

Mykyta Kozhevnykov (Marketing 4) - Market analyst and researcher

Software development lifecycle

Seven main stages of SDLC:

1. Planning
2. Analysis
3. Design and Prototyping
    a. Architecture
    b. User interface / User experience
    c. Platforms
    d. Communications
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Planning

The process of establishing goals and tasks to be achieved during a set amount of time (sprint, for instance).

Analysis

Analysis's main task is to figure out the main issue while trying to fix the system. Furthermore, different stages are present in the analysis, such as:

- Subdividing the project/system into smaller pieces
- Analyzing and setting goals
- Coming up with what has to be done
- Researching and creating questionnaires for end-users to create definite requirements.

Design & Prototyping

During this stage, the predefined set of prerequisites has to be made:

- Certain architecture, namely the programming languages, frameworks, techniques, have to be specified
- A user interface including wireframes, prototype, website/application design, etc
- User experience involving user stories, customer journey maps, and other useful tools
- Platforms to be able to get your application working
- Communications have to be defined as measures for the application to be able to establish a connection to the server or perhaps other copies of the application.

Implementation

The hands-on experience with code. All the implementations of the designed architecture are written here.

Testing

Quite a crucial step that has to be done in order to present the Minimal Value Product, start the deployment process and begin to receive feedback from end-users.

Deployment

In the deployment step, the application is being made available to the target audience, which itself starts to use it, test it, and give the very necessary and important feedback.

Maintenance

The maintenance step involves a complete, deep evaluation of the project or the system to make sure it stays up-to-date. Additionally, all the different changes could be implemented in the software part of the project here.

SDLC Methodologies

Waterfall

The Waterfall model is the basic methodology of development pertaining to the step-by-step analysis and development of the software.

Agile

The Agile is much more flexible of an approach. It has some core principles, such as:

- people are more important than process;
- the product is more important than overwhelming documentation;

- readiness to change is more important than following the set plan. [9]

Iterative

In the Iterative development model, developers create a demo version of the software in a matter of a couple of iterations. It is a methodology based on cycling through the multiple different parts of the project, i.e., analyzing, testing, implementing, prototyping, and fine-tuning the product.

Without a shadow of a doubt, it is vital to use a Version Control System in any of the aforementioned methodologies in order to achieve success.

Architectural patterns and styles:

- Client-server
- Data-centric
- Monolithic app
- Microservices architecture
- Layered architecture
- Representational State Transfer (REST)

Our decision for the team project - RESTful API and Agile approach (Scrum).

Advantages of REST architecture

Here are a couple of upper sides of a REST architecture:

- **Simplicity:** The RESTful architecture is much easier and more convenient to develop than SOAP.
- **Scalability:** The applications can be scaled quite widely via adding additional server nodes behind a load balancer. [8]

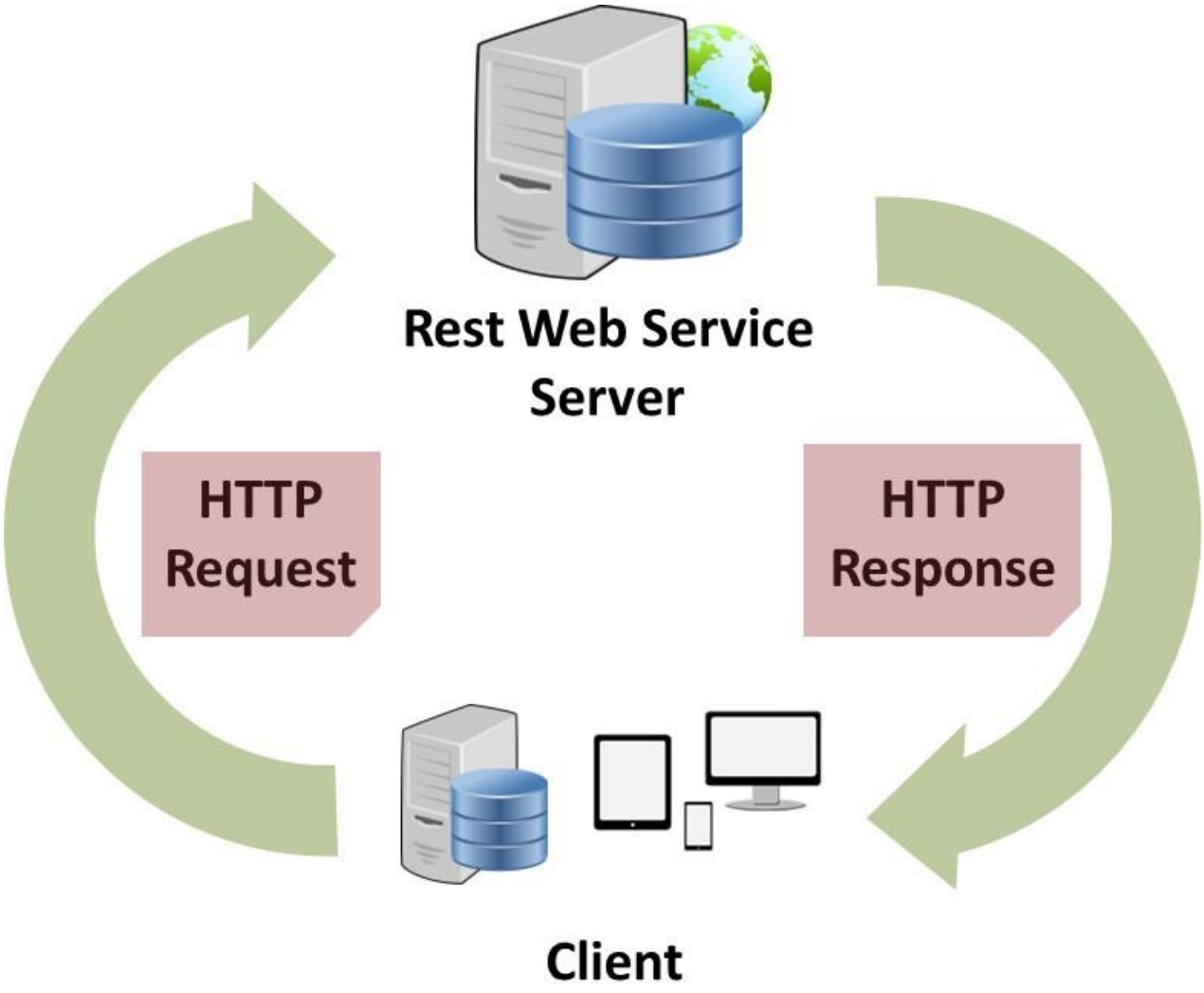


Image 1.1. REST Architecture Scheme

Approaches to system architecture design.

Product management, research, and technology stack decision

Product management

During every application/product development project, there is always a need for proper coordination inside the team, as well as the task delegation. The product manager comes in handy in this situation.

Who are product managers?
A product manager is a bridge between the business objective and the customer requirements. This person comes up with an idea of what a solution to a problem looks like and proposes a vision to implement with the help of an assembled team.

PM responsibilities:

- Understanding what an end-user requires.
- Developing analyses by analyzing certain market trends.
- Helping define product's vision.
- Helping stakeholders understand this vision
- Giving priority product features and capabilities.

Research

A lot of research has to be done in order to correctly assess the target audience, the need for your product, the number of resources needed to get to the MVP stage, and many other things.

Thanks to preemptive and right research, a lot of pitfalls can be avoided due to the fact that Cone of Uncertainty (an agile methodology term used to describe the confidence in how the project is going to transpose from one stage to another) begins to narrow down, and the potential losses are fortunately minimized.

In order to sum up, research analysis and discovery of the target domain are all vital and necessary for the risk diminishing. The risk being the potential chance that the product won't be needed, or will be perceived as useless, or will lack some key components/features.

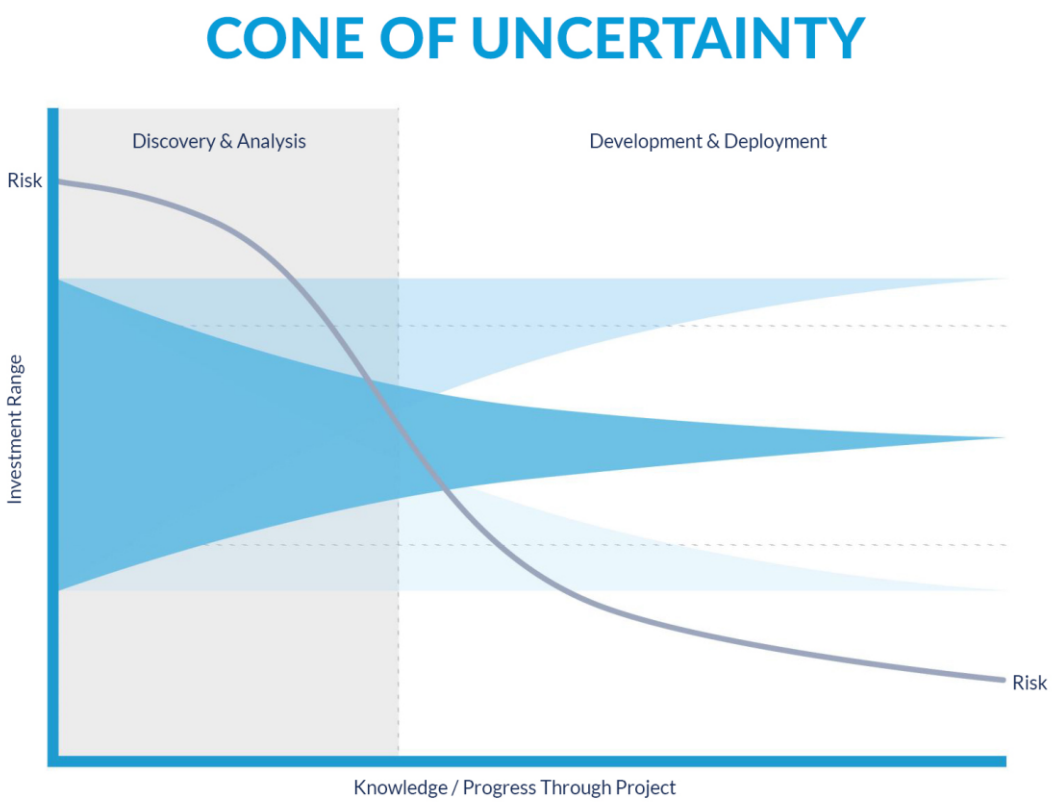


Image 1.2. The cone of uncertainty

Technology stack decision

Last but not least, we do require a proper comparison of the wide range of tools, technologies, frameworks, and libraries before committing to a certain decision. This is definitely a part of the proper research. However it's not a business-, but rather a tech-oriented one. Different parts of the product are to be considered:

- Server support
- UI/UX support
- Core features, i.e., engines/approaches to a solution for a certain task

- CI/CD support

Backend backbone. Frameworks, potential options

Main programming language: JavaScript
Frameworks: Node.js

Main programming language: Python
Frameworks: Flask, Django

Main programming language: Java
Frameworks: Spring Boot

1. Node.js

Node.js is a server platform, which is a part of the technological stack that encompasses all the needs in web development, based on JavaScript. Node.js uses JavaScript-V8-Engine, the same one that is used in Chromium-based browsers. Quite a large amount of popular frameworks are based on Node.js, such as Express.

Node.js advantages

- The appearance of Node.js made JavaScript full-stack web-project development possible. This resulted in the fact that developers of different server sides of the applications got their hands on not only the powerful JavaScript potential but also the whole JS ecosystem, that is, libraries that could be used in the server environment.

- JavaScript code is much more compact than, for instance, C code. The performance of JavaScript code is quite high to be used in projects which require fast computation.
- Node.js platform and its frameworks are not heavily resource- and scalability-dependent. That's why the platform is often used by those who use microservice architectures.
- Node.js JavaScript code is compiled into machine code, which allows better performance in comparison to code interpreting. The community of JS developers foresees the constant development of Node.js as Google is always trying to improve the V8 engine.
- Node.js — is an open-source project that hosts a lot of developers. This helps people who struggle with some issues find a fast solution from the community.
- Node.js is planned to become a heavy-compute platform, which will be decent for such tasks as machine learning, deep learning, etc. [4]

2. Django

Django is a backend Python framework. It follows the model view controller (MVC) approach. It is great for the development of advanced web applications.

The framework improves coding greatly, is prone to be very reusable, and will definitely lead to a faster development time. Django is used by many popular products, such as Instagram and Pinterest.

Django Advantages

- Decent speed – Django is supposed to be used even by novice developers, which helps developers speed up the entire process.

- Wide variety of features– Django has quite a wide arsenal of tricks and features which help users achieve certain project requirements. Such tasks as authentication, sitemaps, security, and many others are solved with the help of these features.

- Secure – Django is without a doubt a secure framework that helps its users prevent security breaches, i.e., SQL injection, packet-sniffing, and others.

- Scalable – Django provides very decent scalability. A lot of well-known enterprises and websites are reliant upon the scalability of Django to maintain the working flow of their products.

- Versatile – Django can be used for the development of many different application types. Some of these include social networks, CRMs, high-compute platforms, and others. [4]

3. Flask

Flask can be easily set up with classic HTML or with bootstrap pieces. It is a modular framework that is capable of high performance (Our choice for the project)

Flask Advantages

- Simplicity – Developers that have a grasp of Python can easily adapt to working with Flask. The framework can be easily understood as there is no requirement to learn multiple different nuances some frameworks possess.
- Flexibility – Flask's components can be easily modified. The application can be configured without a problem
- Decent performance – Flask is designed to provide high performance for quite a large amount of end-users. Flask is not dependent on an abstraction between the user and requests.
- Modularity – with the help of modular structure, the developer can write software much more freely and structured. [3]

4. Spring Framework is a Java-based framework. Developers can use a wide variety of extensions inherited from Java EE.

Spring Boot Advantages

- Dependency injection management is made simple
- The application and its properties are easily customized
- SpringBoot tutorials are easy to understand for even junior developers. [3]

FrontEnd backbone. Frameworks, potential options

Angular

Angular is a frontend framework written in TypeScript, developed by Google

Data binding. Angular was built with Model-View-Controller architecture. The Model and View parts are easily synchronized. Data binding provided opportunities for engineers to diminish the time needed for development.

Dependency injection. Dependencies figure out the interaction of many code snippets and distinguish the influence of changes being made in the components. Quite commonly, the dependencies are components' built-in parts. With Angular, there is an option to use injectors that define dependencies in a different manner, helping separate the components from the dependencies.

Community. From the very start, Angular surged in popularity among the best of engineers. Enough support, discussions around the problematic topic as well as an ability to find a solution to nearly every potential problem or issue have made Angular attractive even to inexperienced users. [1]

Vue.js

Vue.js is another option to consider when choosing the frontend framework for the project. It is currently one of the most emerging frontend technologies.

Vue.js Advantages:

Simplicity

Vue.js philosophy is centered around the Pareto principle - 20% of commitment and 80% of the result. Vue.js is also great for working with components because usually, all the file types (html, css, js) can be stored in a single file.

Integration

Developers can integrate Vue.js into other frameworks such as React or Angular, making the project much more customizable. Due to the ease of integration, Vue.js is rising in popularity as a choice for web development. What is more, with the help of many components, engineers managed to create different solutions for the applications and to switch between the preferable frameworks.

User-Friendly

Vue.js can be described as an easy-to-learn framework. It doesn't require prior knowledge of multiple programming languages. Another advantage is the ability of the files to be used in a multitude of editors without causing too many issues.

Customization

Thanks to all of its functions being accurately accessible, Vue is a good choice for any engineer. The functions are supposed to be named according to personal

preferences. Every submodule of the app can have secluded functions, making it seizable to get the application correctly customized.

Very Few Restrictions

The design of Vue.js offers not as many restrictions and greater flexibility to complete the project. The core library focuses on the View part, which, combined with the modular approach and the use of various libraries, allows programmers to get any issues resolved quickly. Even though the community of Vue is only developing and evolving, it still offers quite good support. [7]

React

ReactJS is another framework that makes use of fast page rendering and JavaScript, making the application very user-friendly and high-performant.

Large community- The community is quite enormous and provides free access to documentation and individual experiences, which is only an upper hand for developers.

Code reusability- React gives any engineer an option to use the already written code. Engineers are not required to rewrite code for different platforms, as ~85% of React's code can be reused for multiple platforms.

Pre-built components- React development transpires in a fast manner due to the components that have already been made present in React. There is a possibility that a feature or a function that you require to use in your application

may have already been sketched by another developer, which you can use without getting ‘charged.’

Simple UI - While building an application, it is important that every action happens in a systematic manner. The UI designed in React is more user-exposed and loads quite faster.

Modularity and declarative programming - with the modular programming aspect of React, engineers can transform segregate any programming code snippets into certain unit structures called modules. The applications can be easily reconfigured and often updated, with the help of constant comparison to similar solutions. Declarative programming improves React development quite nicely. [2]

Classic JavaScript, JQuery, and templating engine (our choice)
Due to the lack of knowledge and certain experience in our team, we chose the option of classic pure JavaScript, JQuery, and built-in template engine Jinja2 in order to develop the frontend part of the application, which turned out to be a decent option.

Search Engine solutions - potential options

Two main options that were considered for the project and the information retrieval, indexing, storage - were Solr and ElasticSearch

Data Sources
Both tools offer an opportunity to work with a wide range of data sources.

Use Cases

While both products are document-oriented search engines, Solr is focused on text searches with advanced information retrieval.

Elasticsearch is a good choice when the task lies within scaling, data analysis, and processing time-series data to find out some hidden patterns in the data. Its integration with Kibana makes it a really popular tool to use for analysis. Elasticsearch is the best choice when it comes to more or less modern web applications, where data is transported in JSON format.

Searching

Both Solr and Elasticsearch support near real-time searches with the help of Lucene's power. They both have additional search-related features.

Indexing

Because both Frameworks are schemaless, it is not a problem to index the data that is unstructured without defining the schema of the index beforehand. Both search engines support custom analyzers, synonym-based indexing, stemming, and lemmatization and tokenizer improvements.

Clusters, Sharding, and Rebalancing

Both Elasticsearch and SolrCloud provide support for sharding. ES has better support for scaling and cluster management due to the fact it was designed to scale horizontally. Its disadvantage lies within the statement that the shards cannot be increased in size once they've been created.

Comparison table 1.1

	Solr	Elasticsearch
Installation and Configuration	Easy to get up and running with and very supportive documentation	Easy to get up and running with with very supportive documentation. Several packages are available for various platforms.
Searching and Indexing	Optimal for text search and enterprise applications close to the big data ecosystem	Useful as both a text search and an analytical engine because of its powerful aggregation module
Scalability and Clustering	Support from Solr Cloud and Apache Zookeeper dependence for cluster coordination	Better inherent scalability; design optimal for cloud deployments
Community	A historically large ecosystem	A thriving ecosystem for the FOSS version of Elasticsearch and the ELK Stack
Documentation	Patchy, out-of-date	Well-documented

The choice was made in favor of ElasticSearch, because it has decent documentation, a lot of tutorials, further potential in analytics via Kibana integration, and is the best choice when data is transported in JSON format. [6]

Continuous Integration and Delivery (CI/CD)

Continuous integration is a software development discipline, which is centered around a certain set of practices that encourage teams of developers to commit changes to repositories as often as possible due to the ability to track down issues and bugs easier.

In causality to the fact that almost all of the applications do require developing code in different platforms and tools, the developers need a segregated platform or tool to track changes and integrate within those different platforms.

A typical CD pipeline has build, test, and deploy stages.

CI/CD features allow setting/changing/removing these variables, masking variables such as passwords and user keys, and having them configured during the deployment.

CI/CD is a really good practice because it addresses the issue between developers who are willing to commit changes often with operations that require stable operations and constant control. With the help of CI/CD, there will be no problem for the team to make commits quite frequently. [5]

Heroku.

Heroku is a platform as a Service that lets you build, test, scale and deploy applications. We used Gitlab integration with Heroku in order to set up the CI/CD process in our team project. We chose Heroku and Gitlab because of the relative ease of setup, not requiring any prior experience with CI/CD. [5]

My experience - project in-depth

While being engaged in a student project in Kyiv-Mohyla Academy, our team was tasked with the following scenario: create a Minimal Value Product in the domain of judicial cases. The estimated time window for the project was about half a year. It was, however, quite challenging to decide on the problem my team was going to address in order to help the lawful representatives, i.e., lawyers (even law journalists, judges, and interested residents were considered as the target audience at first).

The following ideas emerged while we were discussing the problematics of the current state of information retrieval and other issues centered around the Law system of Ukraine:

- Anomaly detection and analysis for the court cases
- Autocompletion of the text snippets for the lawyers
- **Ranged smart search on the collection of documents provided by the hosting company.**

A big discussion took place, a couple of times even. Finally, the third idea was selected and decided to be worked on.

Our team consisted of two frontend developers, a backend developer, a product manager, and a marketing analyst. I was responsible for the product and initialized and monitored all the coordination inside the team. What is more, we would not have been able to achieve success without the help of our scrum master.

We were iterating by sprints. Every 1 to 2 weeks, every team member presented their completed tasks, described the plans for the next sprint.

General project structure

RESTful API lies in the middle of the development and implementation of the client-server part of the application. The main server/route interaction file - app.py - is a main Flask application file, which contains all the necessary imports (libraries), routes, functions, dictionaries i/o files.
The following modules are present in the project, namely: the parser.py module, which is an independent module responsible for parsing all the needed documents, i.e., formatting them, splitting them into parts, removing stop words, etc.; the elastic_insert.py module, which helps easily, in a matter of tens of minutes, to reupload all the necessary parsed documents in the elastic search indices (this was the case a couple of time, due to the fact the ElasticSearch free trial version only lasted a few weeks or so. Upon the end of the trial, we had to set up a new cluster and reintegrate it within our server. Last but not least, we have an important mapping storage dictionary, which has the key of the document name, and the value of a tuple, containing the index name and document id inside that index inside the ElasticSearch.

The client part of the application consists of the templates' module that encompasses all the templates for the rendering and the static module - the one having the submodules of .css, .js files, and images.

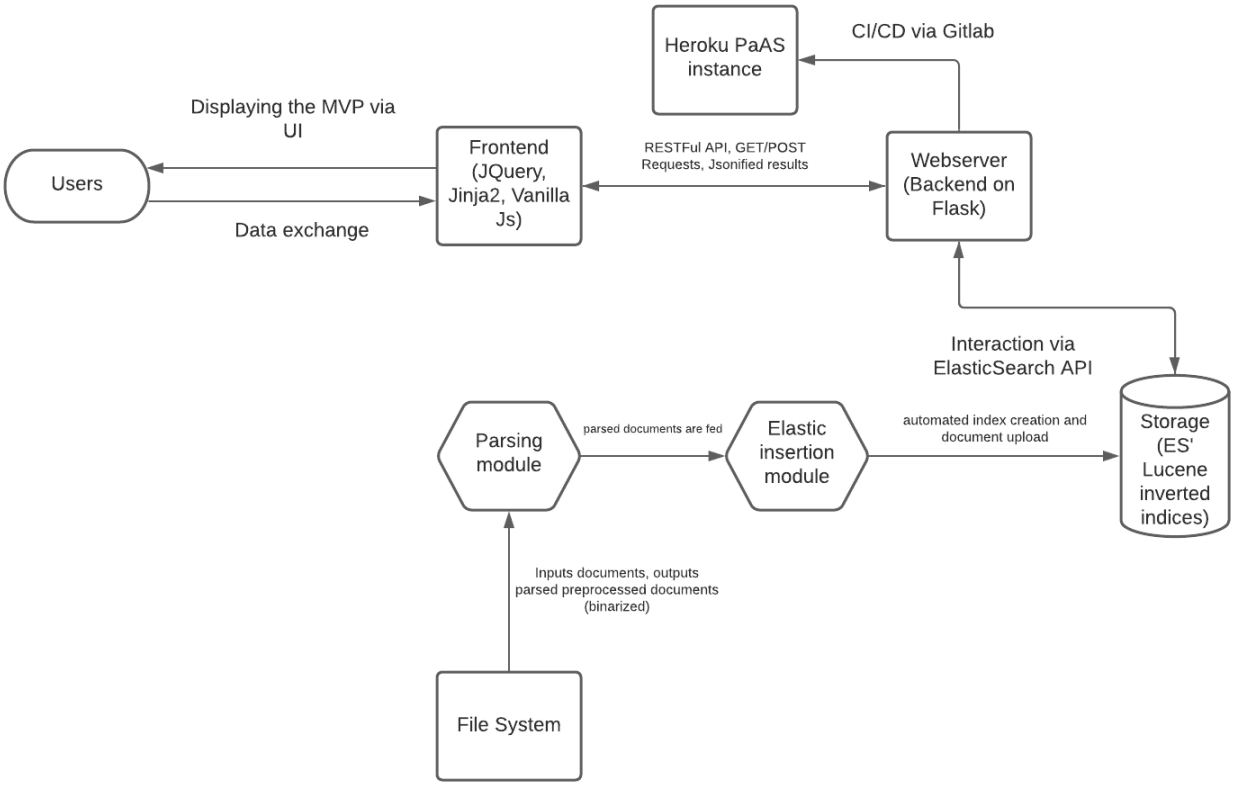
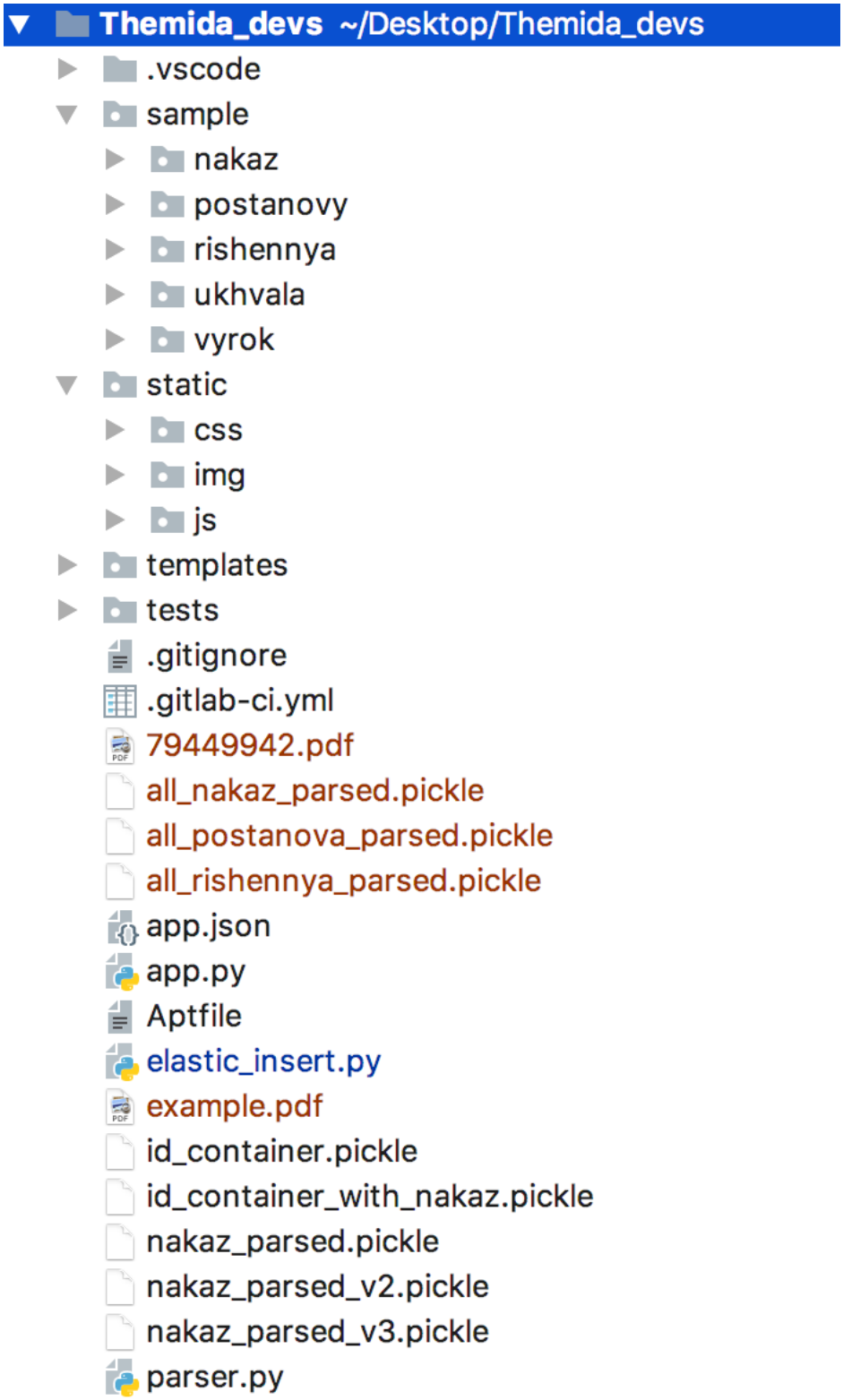


Diagram 1. The designed architecture of the application. It has all the necessary components, i.e., Backend, Front-end, Search Engine Core, and CI/CD.



Snapshot 1.1. The file and directory structure within the PyCharm project.

```
with open('all_nakaz_parsed.pickle', 'rb') as f:
    temp_dict = pickle.load(f)


# https://them-iders-dev.es.europe-west4.gcp.elastic-cloud.com:9243

es = Elasticsearch(
    ['them-iders-dev.es.europe-west4.gcp.elastic-cloud.com'],
    http_auth=('elastic', 'II8naw65mco3gGfKHXp7jOY0'),
    scheme="https",
    port=9243,
)

with open('id_container_with_nakaz.pickle', 'rb') as f2:
    id_container = pickle.load(f2)

# change this to relevant doc_type
index_name = 'nakaz' # nakaz # rishennya
i = 0

for i, (k, v) in enumerate(temp_dict.items()):
    i += 1
    #print(v)
    temp_body = {

        'doc_name': k,
        'doc_text_corpus': v,
        'case_num': v[0],
        'doc_date': v[1],
        'first_part': v[2],
        'main_part': v[3],
        #'main_part_2': v[4],
        'judge_part': v[4],
        'timestamp': datetime.now()
    }
    res = es.index(index=index_name, id=i, body=temp_body)
    print("DBG {} out of {}, {} inserted".format(i, len(temp_dict.items()), k))
    id_container[k] = (index_name, i)

with open('id_container_with_nakaz.pickle', 'wb') as f1:
    pickle.dump(id_container, f1)
```

Snapshot 1.2. Example of the developed python module, namely the elastic_insert.py module, responsible for transferring the documents into ElasticSearch indices.

Challenges & Problems

During the planning, analysis, and research stages of our product development, we interviewed a lot of different potential end-users of our product. We heard many ideas from lawyers, law journalists, and just people who had certain judicial expertise. We came up with a list of points of the biggest ‘pain’:

Lack of a proper search system over the documents with all the conveniences, snippets of the query inside the document.

Snippets, as in Google, were one of the first cool features we came up with. Allowing the users to quickly take a look at the document’s inner part without opening it and searching and scrolling through the whole page is quite impactful. We received mostly positive feedback regarding the feature during our final pitch presentation in the host company.

Lack of useful document grouping (by case ID etc.)

The feature we introduced allowed users to interact with grouped documents and was highly useful

Lack of improved search.

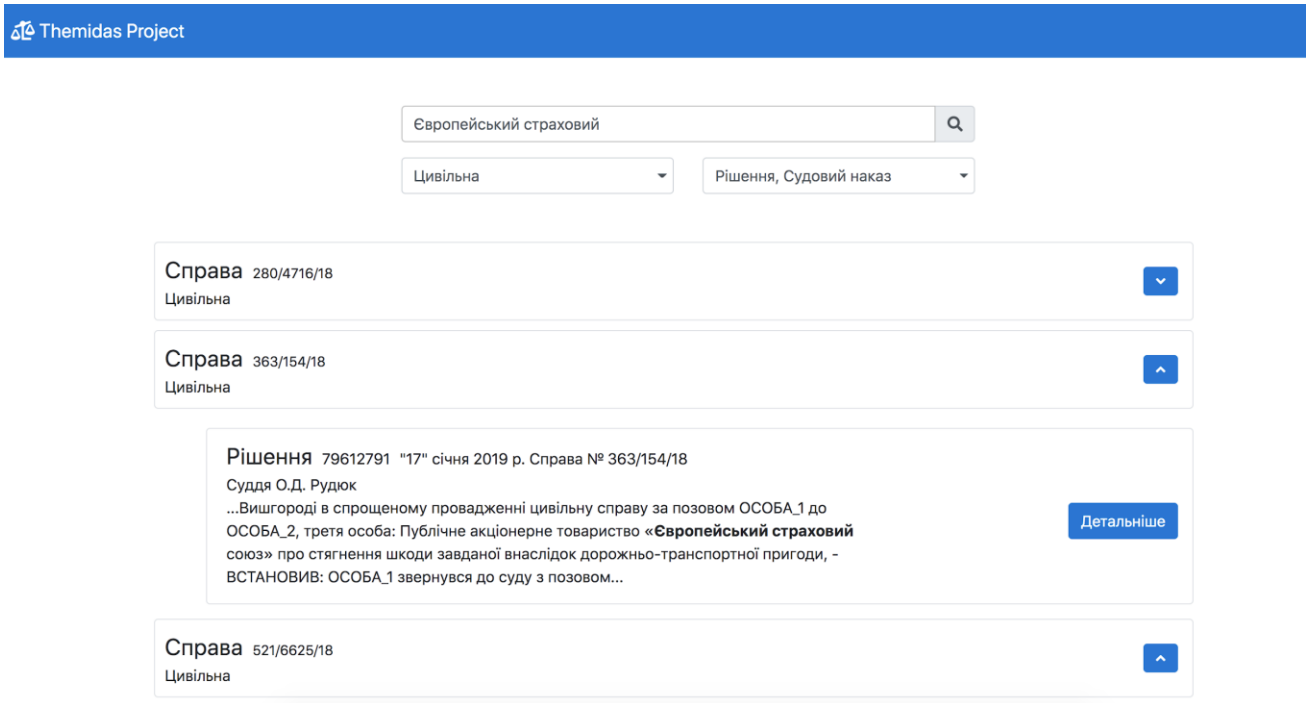
We used ElasticSearch, which allows multiple different query formats & search with priority over parsed document areas.

This was definitely among the first ideas we checked, and it resonated quite nicely with the feedback of our interviewees.

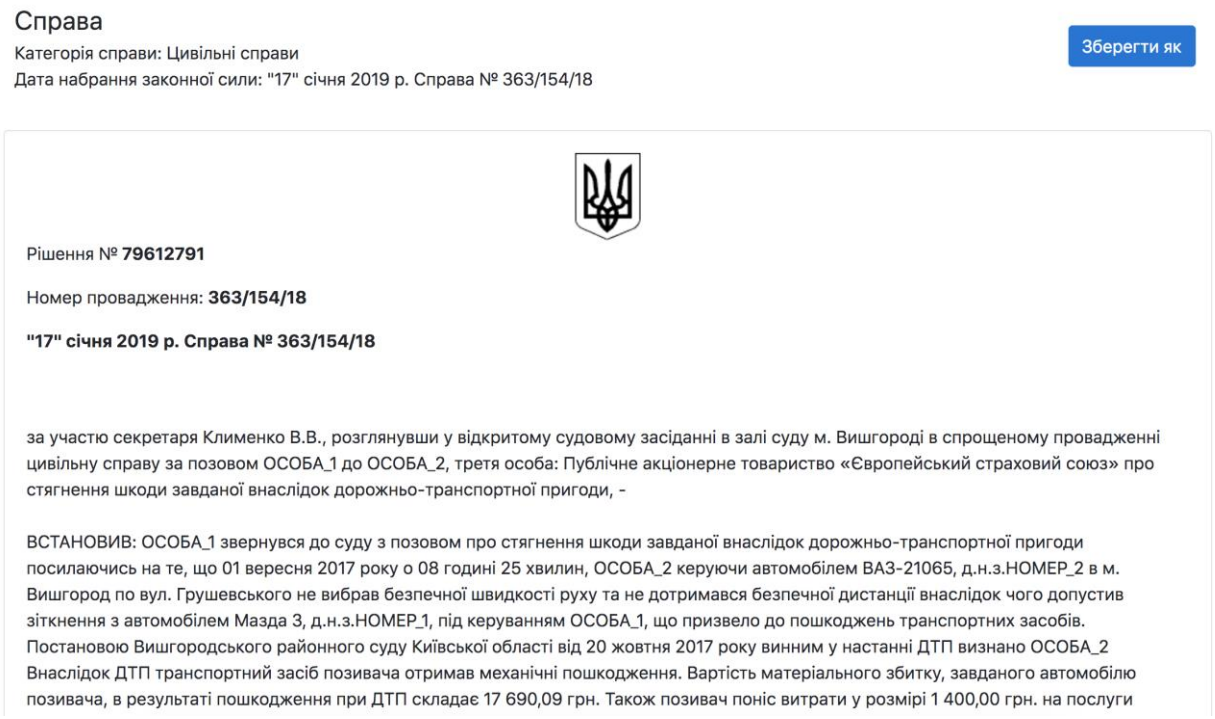
Absence of parsed documents areas, such as case ID, date, intro part, resolution part, the final part, judge, etc.

Without a doubt creating a parser was a decisive part of the project - this was a completely separate, self-written module that achieved decent accuracy in parsing the judicial documents of different types

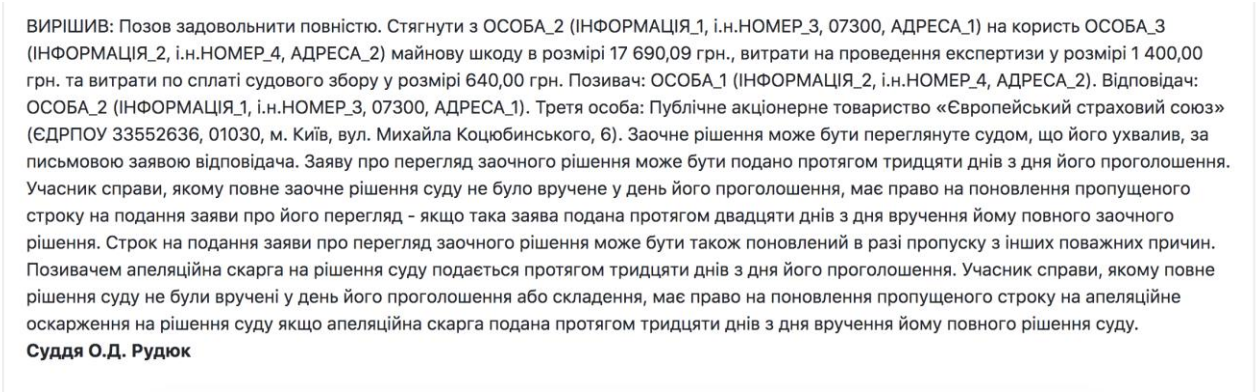
Product demo



Snapshot 2. The “Європейський страховий” query and the results. The snippet displays the most relevant appearance (defined by ElasticSearch) of the query, ranged by the predefined, preparsed zones of the document (initial, motivational, resolutional, final), and displays the query itself in bold.

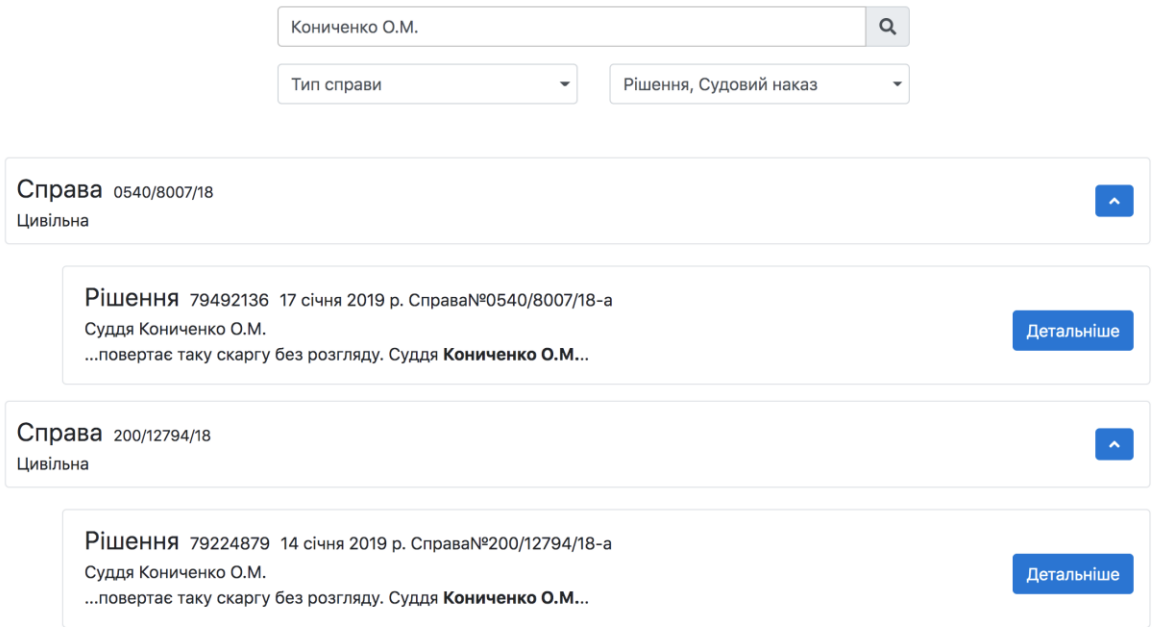


Snapshot 3.1. Demonstration of the document page, segmented by the document zones. The decision number, case ID, date, and the motivational part are clearly displayed.

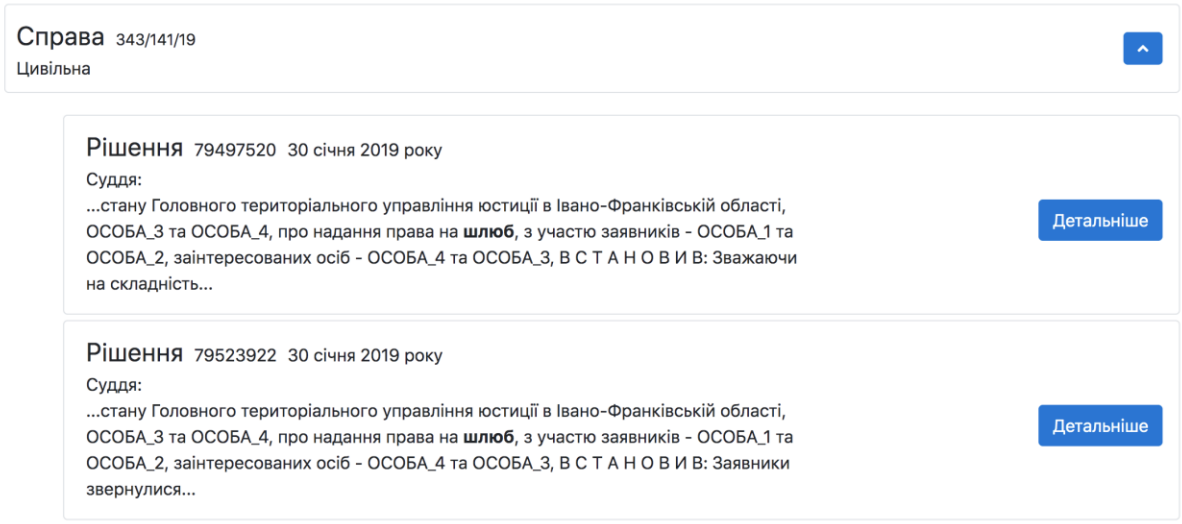


ВИРІШИВ: Позов задовольнити повністю. Стягнути з ОСОБА_2 (ІНФОРМАЦІЯ_1, і.н.НОМЕР_3, 07300, АДРЕСА_1) на користь ОСОБА_3 (ІНФОРМАЦІЯ_2, і.н.НОМЕР_4, АДРЕСА_2) майнову шкоду в розмірі 17 690,09 грн., витрати на проведення експертизи у розмірі 1 400,00 грн. та витрати по сплаті судового збору у розмірі 640,00 грн. Позивач: ОСОБА_1 (ІНФОРМАЦІЯ_2, і.н.НОМЕР_4, АДРЕСА_2). Відповідач: ОСОБА_2 (ІНФОРМАЦІЯ_1, і.н.НОМЕР_3, 07300, АДРЕСА_1). Третя особа: Публічне акціонерне товариство «Європейський страховий союз» (ЄДРПОУ 33552636, 01030, м. Київ, вул. Михайла Коцюбинського, 6). Заочне рішення може бути переглянуте судом, що його ухвалив, за письмовою заявою відповідача. Заяву про перегляд заочного рішення може бути подано протягом тридцяти днів з дня його проголошення. Учасник справи, якому повне заочне рішення суду не було вручене у день його проголошення, має право на поновлення пропущеного строку на подання заяви про його перегляд - якщо така заява подана протягом двадцяти днів з дня вручення йому повного заочного рішення. Строк на подання заяви про перегляд заочного рішення може бути також поновлений в разі пропуску з інших поважних причин. Позивачем апеляційна скарга на рішення суду подається протягом тридцяти днів з дня його проголошення. Учасник справи, якому повне рішення суду не були вручені у день його проголошення або складення, має право на поновлення пропущеного строку на апеляційне оскарження на рішення суду якщо апеляційна скарга подана протягом тридцяти днів з дня вручення йому повного рішення суду.
Суддя О.Д. Рудюк

Snapshot 3.2. Additional fields, such as the resolution part and the judge, are displayed as well.



Snapshot 3.3. Finding by the judge who was responsible for the case.



Snapshot 3.4. The key feature - grouping by the case ID, an important addition to our MVP.

Summary

To sum up, the following areas of software development were researched and thoroughly described in this work:

Software Development Life Cycle (SDLC) - it plays a crucial role in defining what product requires, what must be done, and how it has to be done.

Software development methodologies and approaches - the three most common ones were mentioned, namely waterfall, agile and iterative.

Architectural patterns and styles - quite a few were named, and the chosen pattern (REST) was described in detail.

Product management - was described as an important component in the lifecycle of every product. The importance of timely **research** was mentioned. With the help of research, analysis, and discovery of the target domain, risks can be severely diminished.

Server-side (backend) and **UI/UX side (frontend)** and their respective most popular frameworks, as well as **Search Engine** and **CI/CD solutions,** were mentioned, described, and compared. Furthermore, to every single mentioned part of the technology - **the decision made** for our **core project** was named and explained.

The **general project structure** was explained in detail, and the **architecture diagram** of the whole system was shown.

Last but not least, **the demo** of the product, as well as some code snippets, were shown via the snapshots.

References

[1] Altexsoft: [Website]. URL: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/ (viewed on: 24.03.2021).

[2] Altexsoft: [Website]. URL: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/ (viewed on: 24.03.2021).

[3] Slant.co: [Website]. URL: https://www.slant.co/versus/158/1398/~spring-boot_vs_flask (viewed on: 27.03.2021).

[4] Back4app blog : [Internet portal]. URL: https://blog.back4app.com/backend-frameworks/ (viewed on: 27.03.2021).

[5] Створене посилання: Inforworld: [Website]. URL: https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html (viewed on: 11.04.2021).

[6] Logz.io: [Website]. URL: https://logz.io/blog/solr-vs-elasticsearch/ (viewed on: 29.03.2021).

[7] Towardsdatascience: [Website]. URL: https://towardsdatascience.com/what-are-the-pros-and-cons-of-using-vue-js-3689d00d87b0 (viewed on: 29.03.2021).

[8] Richardson L. RESTful Web APIs: Services for a Changing World. California. 361 p. (viewed on: 22.03.2021).

[9] Schwaber K. , Beedle M. Agile Software Development with Scrum. New York. 158 p. (viewed on: 22.03.2021).