Bachelor's thesis

DEVELOPMENT OF REPORTING TOOL IN THE EXPERTS.AI PLATFORM

Lev Popov

Faculty of Information Technology
Department of Software Engineering
Supervisor: Ing. Stanislav Kuznetsov, Ph.D.
October 24, 2024

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Development of Reporting Tool in the Experts.ai Platform
Student:	Lev Popov
Supervisor:	Ing. Stanislav Kuznetsov, Ph.D.
Study program:	Informatics
Branch / specialization:	Web and Software Engineering, specialization Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2024/2025

Instructions

The goal of this work is to create a reporting module for displaying user activity statistics on widgets within the Experts.ai platform. The work will be divided into two parts and will be developed in cooperation with Roman Chertishchev. This part will include requirement analysis, UI/UX design, mockup testing, and will focus on the design and implementation of the user interface using modern frontend technologies. The development will take place directly in the Experts.ai platform and will be tested in a real environment.

1. Familiarize yourself with the issue and define business requirements.
2. Conduct requirement analysis for displaying user activity statistics on widgets.
3. Design and implement the user interface using techniques typical for web applications.
4. Perform integration with the backend part and create a functional MVP.
5. Test your design with real data.

Electronically approved by Ing. Michal Valenta, Ph.D. on 8 February 2024 in Prague.

Citation of this thesis: Popov Lev. *Development of Reporting Tool in the Experts.ai Platform.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

First of all, I would like to thank my supervisor, Ing. Stanislav Kuznetsov, Ph.D., for providing me with this topic for my bachelor's thesis and for his dedicated time and consultations that helped me correct mistakes in my work. Secondly, I want to express my deepest thanks to my family and friends for their support throughout my studies. I would also like to thank all the developers of the Experts.ai platform who shared their knowledge of implementation and architecture details.

Declaration

In Prague on October 24, 2024

Abstract

The subject of this bachelor's thesis is the design and implementation of the Reporting Tool frontend. This module expands the Experts.ai web platform by visualizing user activity on widgets in the form of charts and tables. The module was implemented using the Angular framework and TypeScript programming language as the rest of the platform frontend to extend its web application with the new components and web pages. The main contribution of this thesis was a thoroughly made user interface design with the most crucial statistics to see valuable insights about user behavior that leads to data-driven business decisions. This design was backed by the modern frontend libraries applied in implementing the Reporting Tool, which significantly improved the user experience and maintainability of the platform's code base.

Keywords web application, frontend, UI, UX, Experts.ai, user interactions, statistics, TypeScript, Angular

Abstrakt

Předmětem této bakalářské práce je návrh a implementace frontendu nástroje pro reportování. Tento modul rozšiřuje webovou platformu Experts.ai vizualizací uživatelské aktivity na widgetech ve formě grafů a tabulek. Modul byl implementován pomocí frameworku Angular a programovacího jazyka TypeScript, stejně jako zbytek frontendu platformy, aby rozšířil webovou aplikaci o nové komponenty a webové stránky. Hlavním přínosem této práce bylo pečlivě navržené uživatelské rozhraní, které umožňuje zobrazit klíčové statistiky vedoucí k cenným poznatkům o chování uživatelů, co napomáhá k rozhodnutím založeným na datech. Tento návrh byl podpořen moderními frontendovými knihovnami použitými při implementaci nástroje pro reportování, a to významně zlepšilo uživatelský zážitek a udržovatelnost kódu platformy.

Klíčová slova webová aplikace, frontend, UI, UX, Experts.ai, uživatelské interakce, statistika, TypeScript, Angular

Contents

List of Figures

List of Tables

List of code listings

List of abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
REST	Representational State Transfer
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UX	User Experience

Introduction

In recent years, many web platforms have desired to enhance user experience by collecting and analyzing how users interact with the product. Companies specializing in Software as a Service (next SaaS) products are continually looking for the best solutions to track activity and improve the usability of their applications. The company EDUMATCH, developers of the product Experts.ai, is no exception.

Experts.ai is a web platform designed to create connections between research organizations and business companies. It allows universities to upload their cooperation offers to the database and generate customizable web widgets for integration into university websites. Potential partners and future colleagues can then explore uploaded opportunities inside these widgets and apply for collaboration within the widget. Collecting and visualizing statistics on visitor activity in widgets was one of the most requested features, so EDUMATCH decided to extend the web platform with the new Reporting Tool to let customers and partners inspect what to improve in their offers or which intellectual assets are the most popular among widget guests.

The result of this thesis is intended not only for the Experts.ai users to view activity trends and make data-driven decisions based on them but also for the Experts.ai platform administrators and developers to track system status and sustainability if all the widgets are active and working correctly.

This topic was chosen because there is a great need to solve this problem. Developing the Reporting Tool user interface is also an opportunity to apply all the learned software engineering methodologies and deliver a valuable and reliable product.

The Reporting Tool will be developed in cooperation with Roman Chertishchev, whose bachelor thesis focuses on designing and implementing the backend part of the module. This thesis aims to analyze business requirements, design user interface wireframes, implement the frontend part of the new module using these designs, and test it with the end users.

This thesis is divided into two parts. In the theoretical part, firstly, the current Experts.ai platform architecture and how it can be expanded with

Reporting Tool functionalities will be reviewed. Possible features to inspire and improve from already existing solutions in web traffic analytics will also be researched in this part. Then, the most popular libraries for making web-based charts will be compared, and the chosen one will be utilized in the Reporting Tool implementation.

In the practical part, functional and non-functional requirements will be stated for the frontend of the Reporting Tool. Then, interactive wireframes of the user interface will be designed based on the resulting analysis. Finally, development decisions made through the implementation phase will be thoroughly described, as well as the testing process of the finished product with the end users.

Goals

The primary goal of this thesis is to design and implement the frontend part of the Reporting Tool module in the Experts.ai platform. The following subgoals must be completed to achieve this goal.

The theoretical part's goals are to overview the current structure and technologies used in the Experts.ai project, research how existing reporting services are solving the problem of visualizing user activity statistics, and choose the best suitable web charting technology for the purposes of Reporting Tool implementation.

The practical part's goals are to define business requirements, analyze the problem domain and to design prototypes of the planned new module according to the UI/UX approaches and conducted research. The final goals of this thesis are to implement the Reporting Tool module using the best modern practices of frontend development, and to test the resulted module with the end users. All of the stated goals will be controlled and approved in tight communication with the EDUMATCH organization.

Part I

Theoretical part

Chapter 1

Experts.ai overview

This chapter provides an overview of the Experts.ai platform interface and functionality, as well as an analysis of the technologies used in its development. This research will allow for a better understanding of the current state of the application, its internal processes, and the domain model of the platform. It will also help identify a list of key performance indicators (next KPI) that users will rely on for making business decisions. The technological analysis will explore possible ways to extend the platform with a Reporting Tool.

First, the user interface of Experts.ai, the contents of the web pages, and the operations that users can perform on them will be presented. Then, the platform's technology analysis will be examined, including the application's module architecture and how they communicate with each other.

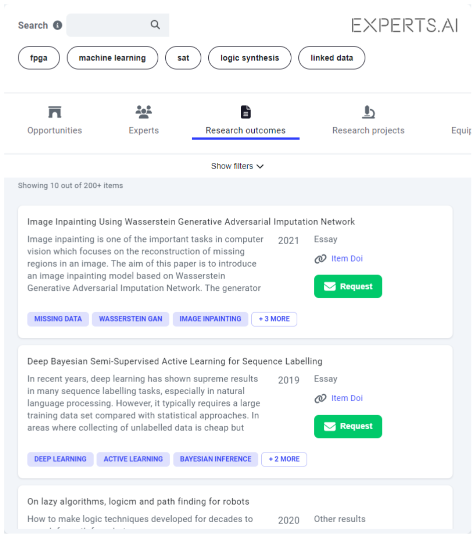
1.1 User interface

Experts.ai consists of two user interfaces to interact with the system: individual widgets at each university and an evidence platform. The individual widgets are designed for visitors of organizations' web pages to explore all uploaded offers and submit collaboration requests regarding the offers they are interested in. The evidence platform provides organization editors with an interface for configuring and generating the described widgets, as well as managing the intellectual assets they have uploaded.

1.1.1 Widget

A web widget is a small interactive third-party application embedded within a Hypertext Markup Language (next HTML) document that extends the functionality of a web page. Widgets simplify the integration of additional services and can significantly improve the user experience by providing useful tools and information in the user's current context.

The Experts.ai platform utilizes *inline frame* widget technology using the HTML element `<iframe>`, which allows rendering another web page in a box within the parent web page by a given URL. [1] On the platform, users can generate `<iframe>` code with the URL of an Experts.ai web page and embed it into the HTML document of their website, making the Experts.ai platform's content available on the user's page.



Showing 10 out of 200+ items

Image Inpainting Using Wasserstein Generative Adversarial Imputation Network

Image inpainting is one of the important tasks in computer vision which focuses on the reconstruction of missing regions in an image. The aim of this paper is to introduce an image inpainting model based on Wasserstein Generative Adversarial Imputation Network. The generator

2021 Essay

Item Doi

Request

MISSING DATA | WASSERSTEIN GAN | IMAGE INPAINTING | + 3 MORE

Deep Bayesian Semi-Supervised Active Learning for Sequence Labelling

In recent years, deep learning has shown supreme results in many sequence labelling tasks, especially in natural language processing. However, it typically requires a large training data set compared with statistical approaches. In areas where collecting of unlabelled data is cheap but

2019 Essay

Item Doi

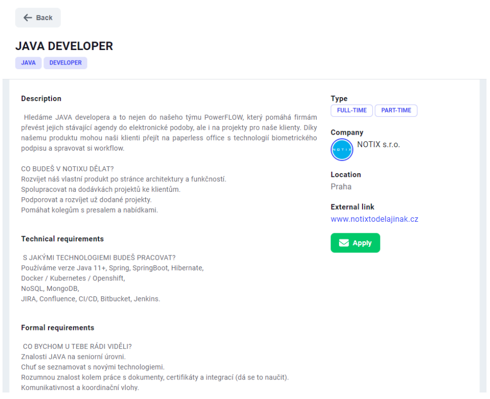
Request

DEEP LEARNING | ACTIVE LEARNING | BAYESIAN INFERENCE | + 2 MORE

On lazy algorithms, logicm and path finding for robots

How to make logic techniques developed for decades to

2020 Other results



JAVA DEVELOPER

JAVA DEVELOPER

Description

Hledáme JAVA developera a to nejen do našeho týmu PowerFLOW, který pomáhá firmám převést jejich stávající agendy do elektronické podoby, ale i na projekty pro naše klienty. Díky našemu produktu mohou naši klienti přejít na paperless office s technologií biometrického podpisu a spravovat si workflow.

CO BUDEŠ V NOTIXU DĚLAT?
Rozvíjet náš vlastní produkt po stránce architektury a funkčností.
Spolupracovat na dodávkách projektů ke klientům.
Podporovat a rozvíjet už dodané projekty.
Pomáhat kolegům s presalem a nabídkami.

Technical requirements

S JAKÝMI TECHNOLOGIEMI BUDEŠ PRACOVAT?
Používáme verze Java 11+, Spring, SpringBoot, Hibernate,
Docker / Kubernetes / Openshift,
NoSQL, MongoDB,
JIRA, Confluence, CI/CD, Bitbucket, Jenkins.

Formal requirements

CO BYCHOM U TEBE RÁDI VIDĚLI?
Znalosti JAVA na seniorní úrovni.
Chuť se seznamovat s novými technologiemi.
Rozumnou znalost kolem práce s dokumenty, certifikáty a integrací (dá se to naučit).
Komunikativnost a koordinační vlohy.

Type
FULL-TIME PART-TIME

Company
NOTIX s.r.o.

Location
Praha

External link
www.notixtodelajinak.cz

Figure 1.2 Widget item details

Figure 1.1 Widget item catalog

Integrating Experts.ai widgets on a web page allows the organization to showcase intellectual assets to potential partners through an interactive visual catalog. An example of such a catalog is presented in Figure 1.1. The catalog consists of a search panel with keywords, an option select of the asset type to be displayed in the list, and the list of offer cards, which are gradually loaded as the widget is scrolled down. Experts.ai platform widgets are most commonly utilized by current clients in the following use cases:

- A **university** organization embeds a widget on its website with a catalog of intellectual assets belonging to the **university** (for example, academic papers and their outcomes).
  Expected widget visitors: Business companies that want to order a consultation or commercialize the research.
- A **university** organization embeds a widget on its website with a catalog of ongoing research projects conducted by **laboratory** organizations partners.
  Expected widget visitors: Business companies looking to sponsor research or potential project partners.

- A **university** organization embeds a widget on its website with a catalog of current job opportunities posted by **company** organization partners of the university.
  Expected widget visitors: Students of the university who wish to collaborate with companies.

It is important to clarify that the functionality of the Experts.ai platform is not limited to these types of collaborations between organizations. Organizations can form any partnerships and install widgets regardless of the types of organizations. From these collaborations, the main entities of the Experts.ai domain can be identified:

- **Organization** The owner of intellectual assets, collaboration offers, or an organization capable of attracting a new audience for other asset holders by embedding a widget on their web pages.
- **Organization type**
  - **University**
  - **Faculty**
  - **Laboratory**
  - **Company**
- **Item** An organization's offer with detailed collaboration description, which can be responded to by opening it on the widget and contacting the holder.
- **Item type**
  - **Expert** Specialist in academic work and various scientific fields, who can be contacted for consultation or collaboration.
  - **Research outcome** Result of experts' work to be applied in commercialization and business innovations.
  - **Research project** Collaboration of scientific experts to discover new research outcomes.
  - **Success story** Case study of the research organizations' collaboration with business partners.
  - **Equipment or Service** Paid asset that business organizations can order from a university for temporary use.
  - **Opportunity** Job position, internship, or thesis topic.
- **Widget** An organization's embedded catalog on a website, allowing the promotion of both the organization's own items and the items belonging to partner organizations.

The following is a list of the main interactions that widget visitors can perform on items displayed in widgets:

View Widget visitor scrolls through the list of items on a widget and sees the newly shown item preview card.

View detail Widget visitor clicks on the preview card to open a new widget page with more details about the offer.

Apply Widget visitor clicks the contact button, fills in a message with their contact details, and sends it.

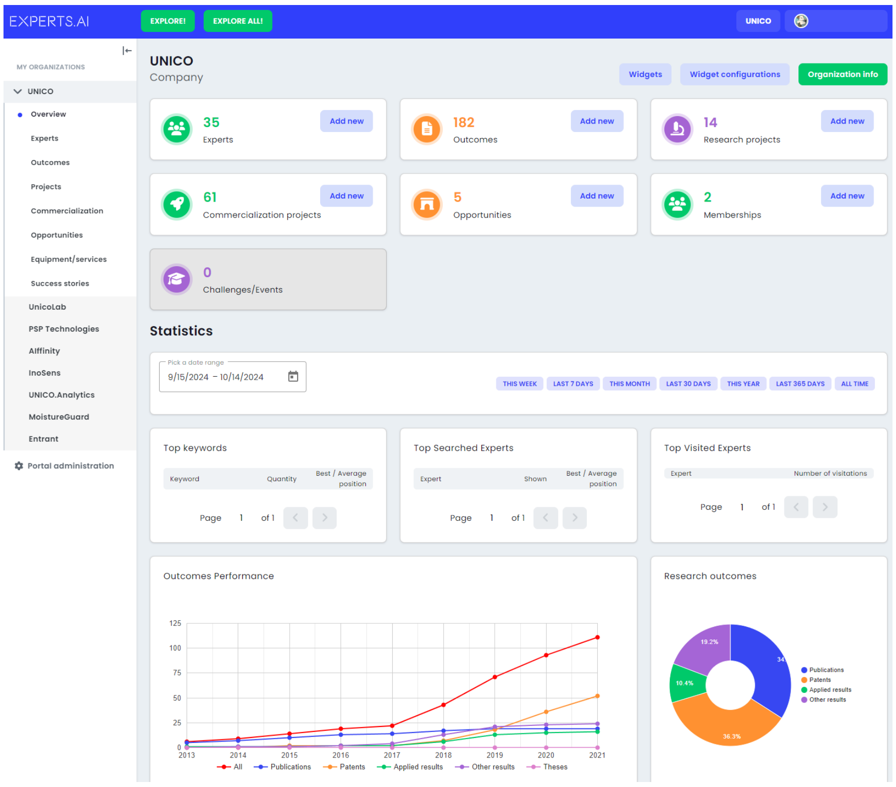
1.1.2 Evidence portal

The Evidence portal is available at the URL `https://experts.ai/`. To access it, a registered profile in the Experts.ai system is needed. After logging in, the user can navigate to the main page of their organization. Unlike the evidence portal, viewing and interacting with widgets does not require authentication and is available to any anonymous visitor of the web pages where the widgets have been integrated.

1.1.2.1 Organization overview

The user interface of the evidence portal is presented in Figure 1.3. The side navigation panel displays a list of all the organizations the current user is a part of. With it, the user can navigate not only to the main pages of each organization but also to the management pages for items categorized by each item type available on the platform. The organization's main page is a dashboard that displays on its cards the number of items of each type, charts showing the performance of research activities, and an interface for managing the user roles and settings of the organization.

At the bottom of this page, there are charts that visualize the organization's performance according to the intellectual assets' successes from the data uploaded to the platform. These charts are implemented using the *Google Charts* library [2]. The charts have visual errors and flickering when hovering the cursor over their sections; some of the charts disrupt the layout of the page and extend beyond their card, and the line chart's tooltip does not follow the user's cursor unless the cursor is moved very carefully and precisely. All of these issues negatively affect the user experience of using the portal, so these interface shortcomings will be considered in the next chapters of this thesis.


EXPERTS.AI
EXPLORE! EXPLORE ALL!
UNICO

MY ORGANIZATIONS
UNICO
- Overview
- Experts
- Outcomes
- Projects
- Commercialization
- Opportunities
- Equipment/services
- Success stories

UnicoLab
PSP Technologies
Affinity
InoSens
UNICO.Analytics
MoistureGuard
Entrant
Portal administration

UNICO
Company
Widgets | Widget configurations | Organization info

35 Experts — Add new
182 Outcomes — Add new
14 Research projects — Add new
61 Commercialization projects — Add new
5 Opportunities — Add new
2 Memberships — Add new
0 Challenges/Events

Statistics

Pick a date range: 9/15/2024 – 10/14/2024
THIS WEEK | LAST 7 DAYS | THIS MONTH | LAST 30 DAYS | THIS YEAR | LAST 365 DAYS | ALL TIME

Top keywords

Keyword	Quantity	Best / Average position

Page 1 of 1

Top Searched Experts

Expert	Shown	Best / Average position

Page 1 of 1

Top Visited Experts

Expert	Number of visitations

Page 1 of 1

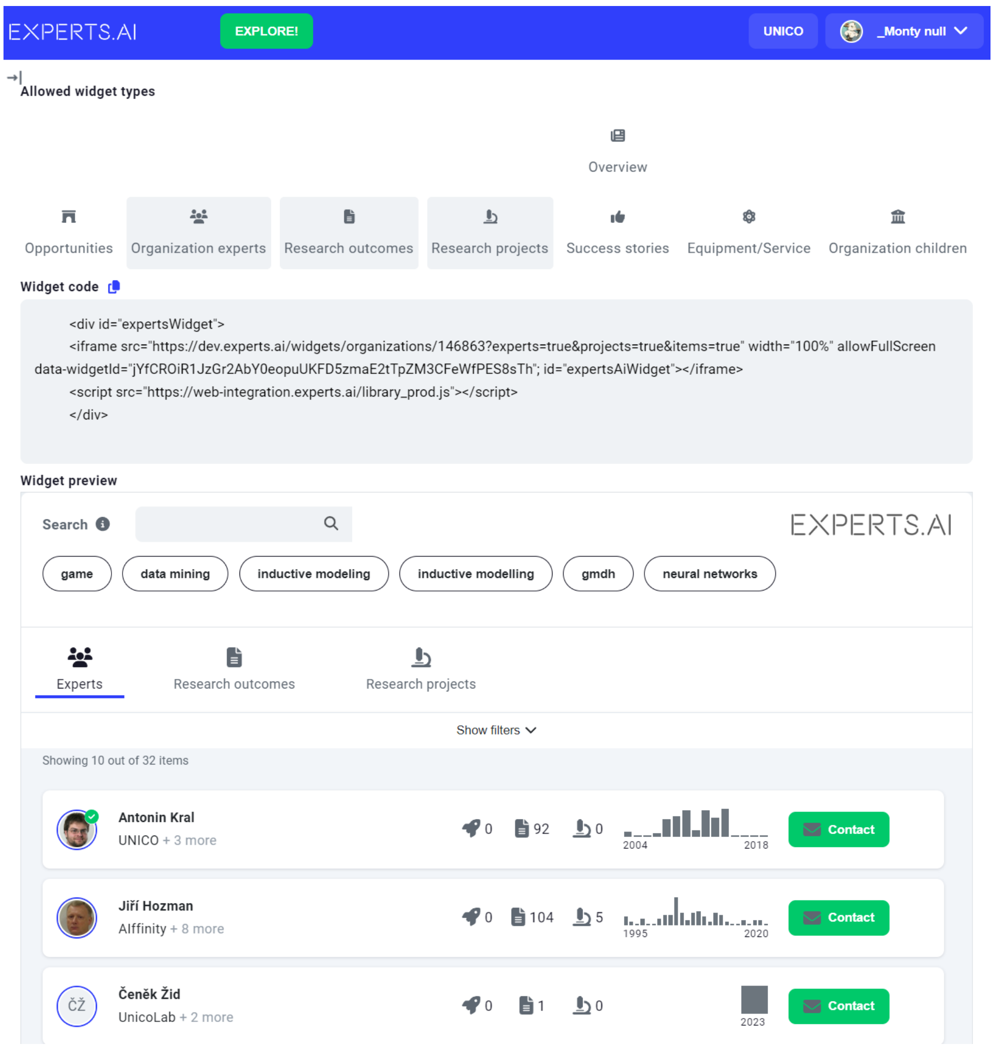
Outcomes Performance

Research outcomes

Figure 1.3 Evidence portal, organization overview [3]

1.1.2.2 Widget configuration

The organization overview page features a “Widget Configurations” button, which directs the user to the widget configurations page. This page is divided into two sections (see Figure 1.4): the upper section, where the organization's editor selects which components will be displayed on the widget, and the lower section, which presents the final embeddable code, based on the HTML `<iframe>` technology mentioned above. After the user reviews the resulting widget visualization preview, this code can be integrated into the organization's website HTML document.



```
        <div id="expertsWidget">
        <iframe src="https://dev.experts.ai/widgets/organizations/146863?experts=true&projects=true&items=true" width="100%" allowFullScreen
data-widgetId="jYfCROiR1JzGr2AbY0eopuUKFD5zmaE2tTpZM3CFeWfPES8sTh"; id="expertsAiWidget"></iframe>
        <script src="https://web-integration.experts.ai/library_prod.js"></script>
        </div>
```

Figure 1.4 Widget configurations page [3]

Organizations usually generate several widgets with different configurations, displaying various types of offers. The current solution does not distinguish widgets from each other or save organizations' widgets and configurations to the platform's database. The widget configuration is embedded directly in the URL that `<iframe>` accepts as a resource to render in another web page. The example of the `<iframe>` URL is:

```
https://experts.ai/widgets/organizations/{organizationId}?experts=true&
↪   projects=true&successStories=true}
```

All the URLs that need to be accessed from the external websites in a form of a `<iframe>` widget start with the path segment `/widgets`, while most of the evidence portal pages start with the `/evidence` path segment:

```
https://experts.ai/evidence/organizations/{organizationId}
```

The lack of widget instance management doesn't allow the system to distinguish organization widgets from each other, which will complicate the analysis of user interaction sources. This problem will be addressed later in this thesis.

1.1.2.3 Portal administration

The portal provides a web interface for managing the platform's global data. These private pages are available only to authorized users with an administrator role. The administrator tools section lacks functionality that would enable convenient monitoring of widget diagnostics and user activity, so it is necessary to check the status of the widgets manually.

1.2 Technology analysis

The architecture of Experts.ai is structured around a three-tier design containing the database (data tier), backend (application tier), and frontend (presentation tier). This division is intended to create a clear separation of each part's concerns, which enhances the platform's scalability and maintainability.

Three-tier architecture is a widely used software application structure, and it allows each tier to function independently while seamlessly interacting with the others. One of the main advantages of three-tier architecture is allowing developers to update or scale each tier simultaneously without impacting the rest of the system. [4]

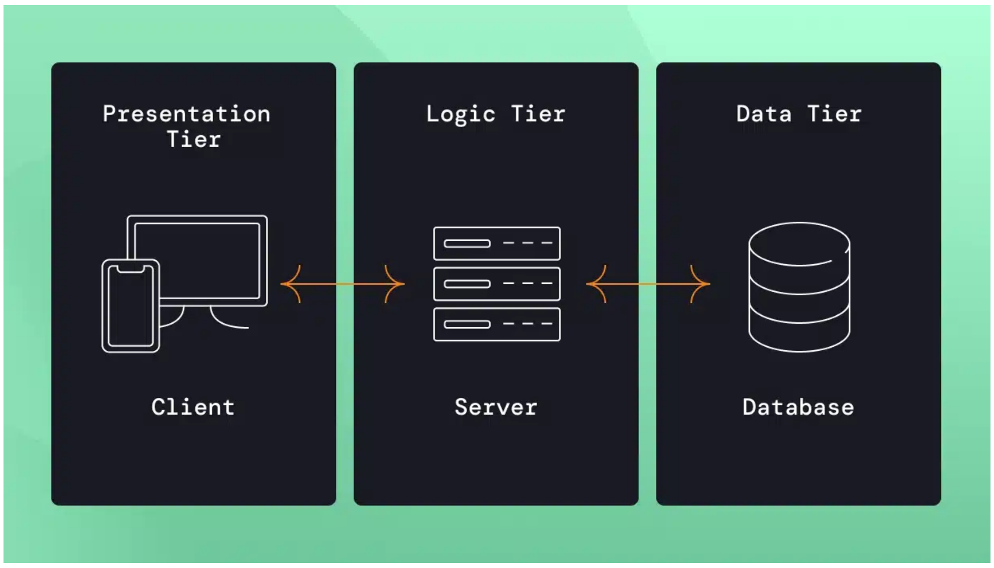


Figure 1.5 Three-tier architecture [4]

1.2.1 Database

The data tier, also known as the database or storage tier, is the place where the application's processed information is stored and managed. Software and systems designed to function as the data tier in the architecture have qualities such as scalability, reliability, and security protection. [4]

In the context of the Experts.ai platform, PostgreSQL [5] is used as the data tier technology, which utilizes the SQL relational data structure. Relational databases consist of a fixed schema that requires data to be organized into predefined tables with columns and relationships between them. This structure supports complex queries and maintains strong data integrity, but its architecture must be well-designed in advance through detailed analysis. [6]

1.2.2 Backend

In the three-tier architecture, the application tier, often called the business logic or server tier, serves the role of a communication channel between data and presentation tiers. It functions as the core where business logic is processed. This tier takes the user inputs from the presentation tier, processes them through its business rules, and interacts with the data tier to add, update, or delete the data as needed. [4]

Experts.ai's backend server is written in Java, a powerful and widely used programming language. Java offers many advantages for completing software engineering tasks, such as its object-oriented nature for modular design and reusable code or its platform independence for web applications that operate on various devices and operating systems. [7] Platform's backend is based on the Spring framework, which incorporates Dependency Injection and Inversion of Control principles. [8] Spring framework suits Experts.ai platform architecture so that all new functionality requirements can be quickly fulfilled without increasing the complexity of the backend server with each new development iteration.

Another important responsibility of the application tier is authorization, the process of verifying whether a user has the necessary permissions to access private resources or data. The backend contains a role system, with which it checks the roles of the current user interacting with it, and if the user does not have the necessary role, access to the resource is denied, returning an error. On the Experts.ai platform, the role system is structured so that in each organization, its participant can have one of three roles: *ORGANIZATION_VIEWER*, *ORGANIZATION_EDITOR*, or *ORGANIZATION_ADMIN*. Additionally, the platform has the *PORTAL_ADMIN* role, which gives users access to manage all organizations and resources through the interface described in Section 1.1.2.3.

1.2.3 REST API

Application Programming Interface (next API) is a collection of rules and protocols that enables one software application to interact with another. It specifies the methods and data formats that applications utilize for communication with one another. APIs allow developers to access features or information from external systems without the need to comprehend the underlying code or concepts. [9]

The data and presentation tiers in the three-tier architecture cannot communicate directly for security reasons. The backend serves all the connections in the architecture, handling user input from the frontend and managing data operations with the database. There are many different types of APIs, incorporating different protocols and data formats. Application tiers provide a specification of API called REST API (also known as a RESTful API or RESTful Web API) to act as an intermediary for the presentation tier to access secured data in the data tier. REST API uses standard HTTP protocol methods such as GET, POST, PUT, and DELETE to interact with resources (identified by unique URI endpoints) on a server and returns standard HTTP responses with status codes. REST APIs are stateless, meaning each request from a client to the server must contain all the information needed to fulfill that request. [9]

These qualities allow REST APIs to be highly scalable, which will be useful for expanding the backend REST API with new endpoints for the Reporting Tool module. New resources must be introduced, with which the presentation tier will communicate to obtain aggregated user activity statistics, and an interface for managing the resource of unified widget instances.

1.2.4 Frontend

The presentation tier serves as the user interface and the communication layer of the application, allowing the user to interact with it. Its primary function is to present information to users and gather information from them. [4]

In the Experts.ai platform, the presentation tier is represented by a web application developed using the Angular framework. This framework allows the development of dynamic web single-page applications (next SPA) that users launch through a browser. Angular is built with TypeScript, a superset of JavaScript developed by Microsoft that adds static typing to catch type-related issues and must be compiled to JavaScript to run in the browser. [10] Angular is based on a Component-Based Architecture (next CBA), which relies on components — reusable and extensible pieces of the user interface. [11] This architecture allows the code to remain modular, making it possible to expand the application with a new Reporting Tool module by introducing new components and utilizing other supporting Angular development tools without significant changes to the system.

Chapter 2

Angular framework

This chapter is dedicated to a detailed analysis of Angular, a modern framework for fast and efficient web application development. As mentioned in the previous chapter (see Section 1.2.4), the frontend application of Experts.ai is built using this framework, so in order to describe the extension of the platform with the Reporting Tool interface, it is necessary to present the development structures and tools of Angular in more detail.

2.1 Architecture

In the past, web clients were actively developed based on the architectural patterns Model-View-Controller and its extension Model-View-ViewModel, which aimed to address the drawbacks of interface and data synchronization present in its predecessor. These patterns strictly distribute responsibilities within the application system, simplifying maintainability; however, unfortunately, the tight coupling between the three main elements in both patterns caused issues with extending applications with new features, testing, and performance. [12]

To address the increasing complexity in web applications, a fundamentally different and innovative architectural pattern appeared — CBA. In this pattern, applications consist of reusable and extendable components, each representing a part of the user interface. The fragmentation of code has improved its efficiency in responding to user input and changes, as well as its readability and maintainability during team development. CBA is now widely applied in all frameworks for developing web applications, including Angular. [12]

A reusable component in Angular consists of the following main parts: [11]

- **TypeScript class** which contains the logic of the component, such as handling events or managing data.
- **Template** controls what is rendered in the HTML Document Object Model (next DOM), the tree structure of HTML elements created by the browser after loading a web page.

- **CSS selector** that defines how the component is named in HTML to be used in another components' templates.
- Optional **list of CSS styles** that control the look and layout of the component.

Other foundational parts of the framework are as follows: [11]

- **Services** are used to separate reusable business logic and data that can be shared among multiple components simultaneously.
- **Directives** are additional template instructions for manipulating the DOM. In Angular, they are divided into two categories:
  - **Structural directives** modify the structure of the DOM by adding or removing elements based on specified conditions (like `*ngIf` and `*ngFor`).
  - **Attribute directives** are used to modify the appearance or behavior of elements (like `ngClass` and `ngStyle`).
- **Forms** are applied for handling and validating user input before sending it to the backend.

2.2 Routing

As mentioned in Section 1.2.4, Angular is a framework for building SPA web frontends. In the past, many applications were built on the MPA (Multi-Page Application) architecture, a traditional web application model where any user interaction or page content viewing resulted in a new request to the server and a full page reload. In this architecture, web applications rely on server-side rendering, where the server generates the full HTML document for each page and sends it to the client, which has its drawbacks, such as slow transitions between web pages and increased backend load.

SPAs, on the other hand, rely on JavaScript to dynamically change the content of a single current page. The initial loading of a SPA page takes more time, but afterward, interactions, page transitions, and user experience become much smoother due to dynamic content changes with JavaScript rather than full page reloads as in MPA. The load on the backend decreases, and page transitions become faster because SPAs fetch only data from the API, not entire HTML documents. SPAs are often implemented with asynchronous data loading, which allows sending backend requests in parallel rather than resource-intensive sequential requests as in MPA. [13]

SPAs implement client-side routing to simulate navigation between different views or components within the same page for the following advantages:

Application state structurization Routing allows SPAs to manage and maintain different application states using distinct URLs. Each route represents a specific view, making the structure of the application easier to understand, manage, and scale. For example, URLs like `/organizations`, `/widgets`, or `/experts` can map to different views of the frontend.

Nested routes A route hierarchy can be established in SPA routing, allowing for navigation patterns within the app. This is valuable when there is a parent and child components layout. Each child component can have its own route, which gets displayed within the parent component's router outlet. The example of such URL is `organizations/:id/widgets/widget-constructor`.

Simulating traditional MPAs Although SPAs only load one HTML page, routing enables developers to modify the URL in the browser and dynamically update the view, creating the perception of navigating between different pages. This approach maintains an experience similar to conventional websites, where users anticipate a URL change as they explore various sections.

Browser features SPA routing utilizes the browser's native navigation tools, such as the back and forward buttons, history, bookmarking, URL sharing, or revisiting particular states of the application later on. These browser capabilities are crucial for a seamless web experience.

Routing is one of Angular's core features, and it can be implemented and configured using the following framework tools: [11]

- **Router module** is responsible for configuring and managing the routing setup in Angular. It defines how different routes map to components and can be imported into the root or feature modules using `RouterModule.forRoot()` or `RouterModule.forChild()`, respectively.
- **Route configuration** is a set of TypeScript Route objects defining the paths for different views, and specifying which component should be rendered when a certain path is visited. It supports dynamic parameters, redirection, and nested routes.
- **Router outlet** acts as a placeholder in the template where the routed components are displayed. It is dynamically filled with the appropriate component depending on the current route and allows for nested routing by placing additional RouterOutlets within child components.
- **RouterLink** is a directive used to create links for navigation within an Angular application. It binds a path to an HTML element, which when clicked, triggers the Angular Router to change the URL and render the associated component, providing SPA behavior without page reloads.

2.3 Reactive programming

Reactive programming in Angular is an asynchronous programming paradigm centered around data streams and the propagation of changes. It allows developers to work conveniently and quickly with the flow of application data and events as users interact with the interface. Angular utilizes the Reactive Extensions for JavaScript library (next RxJS) to represent this data as streams, where some components can emit values (for example, those receiving user input), while others subscribe to changes in that stream (interface components that need to update based on the input). [11]

RxJS provides the type `Observable`, which acts as a data stream that emits its value to everyone who is subscribed to its changes. If a subscription appears after the `Observable` was emitted its value, the subscriber will not receive any value until the stream emits the next one. In cases where it is necessary for the subscriber to receive the last emitted value upon subscription, regardless of the data stream's behavior, the type `BehaviorSubject` is used. [11]

In addition to accepting user input, asynchronous handling is also actively used in frontend communication with external REST APIs through HTTP protocol. Angular has a built-in service called `HttpClient`, designed for sending such requests. Web applications communicate with APIs asynchronously, so the methods for sending HTTP requests return a response of type `Observable` after successful communication. Additionally, `HttpClient` automatically maps the JSON responses returned from the REST API into TypeScript objects using predefined interfaces, eliminating the need to implement additional response parsers. [11]

Chapter 3

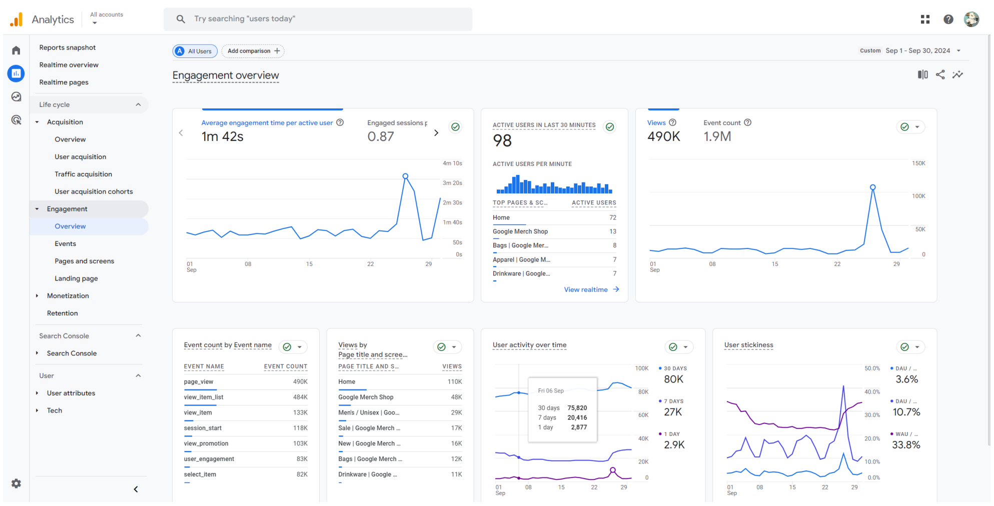
Existing solutions

This chapter examines existing solutions in web analytics and user behavior tracking. Services will be researched from the perspective of user experience and available functionalities for the end user, including which statistics these platforms consider most important and which of them lead to valuable data-driven decisions.

This analysis provides conclusions about what was already explored and defined in the field of user activity tracking and highlights the pros and cons of modern solutions that can be addressed during this thesis and discussed with the EDUMATCH.

3.1 Google Analytics

Google Analytics is one of the most popular tools for web traffic analytics of user behavior on websites and mobile applications. Its primary task is the automatic collection of data about website visitors, such as which pages they view and how their activity converts into key actions on the client's website. Then, on the analytics portal, visualized aggregations of user activity can be thoroughly explored in an intuitive user interface. [14]



ACTIVE USERS IN LAST 30 MINUTES
98

ACTIVE USERS PER MINUTE

TOP PAGES & SC…	ACTIVE USERS
Home	72
Google Merch Shop	13
Bags | Google Mer…	8
Apparel | Google M…	7
Drinkware | Google…	7

Event count by Event name

EVENT NAME	EVENT COUNT
page_view	490K
view_item_list	484K
view_item	133K
session_start	118K
view_promotion	103K
user_engagement	83K
select_item	82K

Views by Page title and scree…

PAGE TITLE AND S…	VIEWS
Home	110K
Google Merch Shop	48K
Men's / Unisex | Goo…	29K
Sale | Google Merch …	17K
New | Google Merch …	16K
Bags | Google Merch …	12K
Drinkware | Google …	11K

Fri 06 Sep

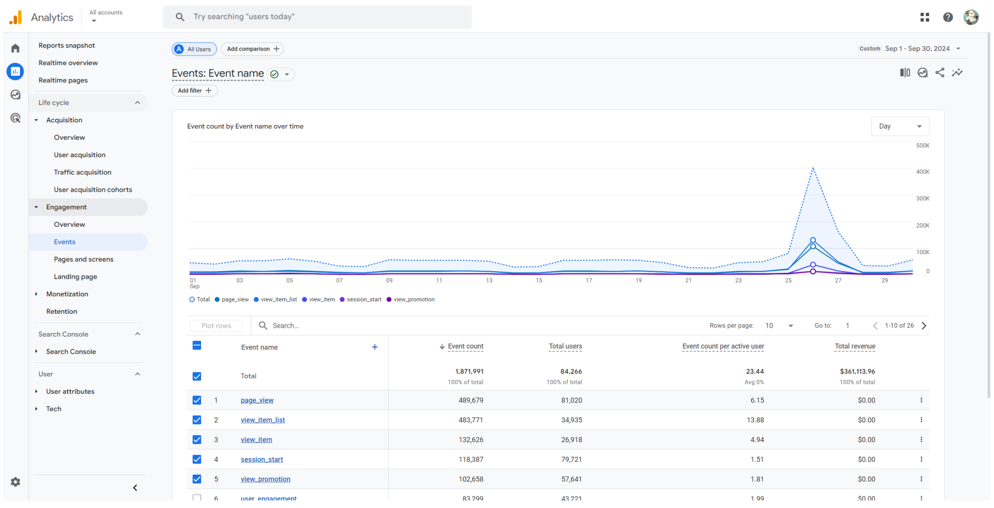
30 days	75,820
7 days	20,416
1 day	2,877

Figure 3.1 Google Analytics dashboard [14]

The interface of the analytics portal in Figure 3.1 consists of a list of premade reports as a template — dashboards made up of report cards with aggregations of activity data. In the header of the portal, there is a time filter that allows users to select a range for which the statistics will be displayed. In Google Analytics, users can manually create custom dashboards from cards according to their preferences. However, the default collection of reports has a quality structure containing everything essential for useful statistics about user behavior:

- **Acquisition report** shows where new users come from so the organization can identify the main sources of audience inflow and continue to work in those areas.
- **Engagement report** describes how the website's content engages users. It visualizes metrics on visitor interest trends, how they interact, and what they like most about the analyzed product.
- **Retention report** reflects how well the website maintains user interest and how many users utilize the website on a regular basis.

The frontend of Google Analytics, like Experts.ai, is implemented using the Angular framework. Therefore, the implementation details and styling can be considered as inspiration for the Reporting Tool. For example, Google Analytics uses sortable tables from the *Angular Material* library (see Figure 3.2) in addition to various charts and lists to display aggregations. [15] It is also worth noting that the graphs on the platform are quite different from those in the *Google Charts* library or are an extreme custom modification of it. From this fact, it can be concluded that it is not advisable to use a library that even its creators do not utilize in the form it is provided to developers in the context of the Experts.ai portal.



		Event name	Event count	Total users	Event count per active user	Total revenue
✓		Total	1,871,991 100% of total	84,266 100% of total	23.44 Avg 0%	$361,113.96 100% of total
✓	1	page_view	489,679	81,020	6.15	$0.00
✓	2	view_item_list	483,771	34,935	13.88	$0.00
✓	3	view_item	132,626	26,918	4.94	$0.00
✓	4	session_start	118,387	79,721	1.51	$0.00
✓	5	view_promotion	102,658	57,641	1.81	$0.00
☐	6	user_engagement	83,299	43,221	1.99	$0.00

Figure 3.2 Google Analytics table statistics [14]

- **Strengths:**
  - Great default template for reports, good categorization of statistics, and only the most valuable visualizations for business decisions.
  - Provides a broad set of tools for tracking traffic sources, user behavior, and conversion metrics.
- **Limitations:**
  - Focuses only on high-level metrics, not many detailed statistics.
  - Difficult dashboard customization, requires good knowledge in data analytics.

3.2 Mixpanel

Mixpanel is a service that tracks user behavior on web portals and platforms. The Mixpanel solution consists of two main parts: a service for integrating monitoring into a customer's web application and an analytical portal where customers can construct their own personalized dashboard from report charts. Each report chart reflects various trends, compositions, or comparisons of user activity metrics. [16]

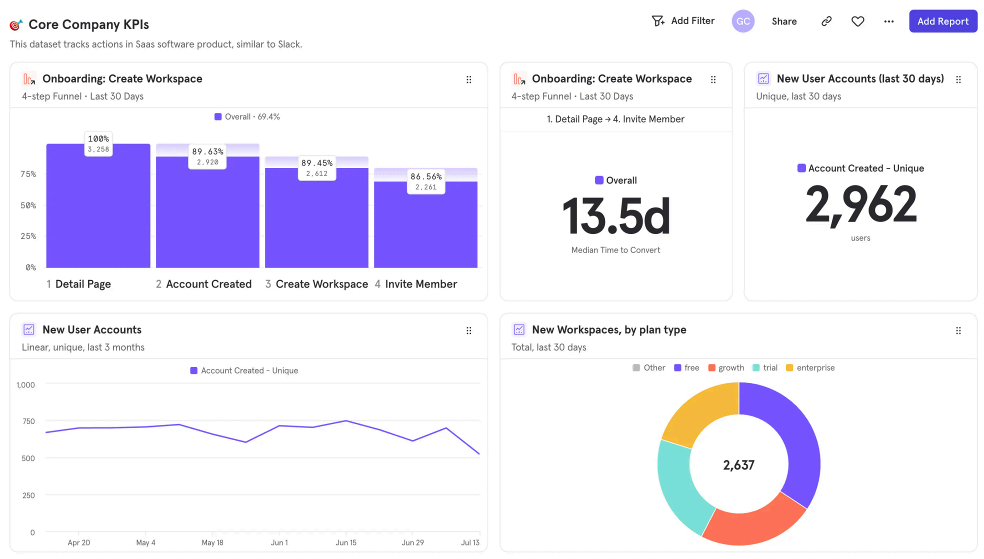


Figure 3.3 Mixpanel dashboard [16]

Mixpanel groups the report cards into the following main categories of visualizations [16]:

- **Insights** are a group of reports that display trends, rapid changes, comparisons and compositions of various metrics as a whole.
  - Time-segmented line chart
  - Time-segmented column chart
  - Comparison bar chart
  - Composition pie chart
  - Table chart
- **Funnels** examine how users perform events in an interaction series. Funnels evaluate the number of users who convert and drop-off from one interaction to another and show these amounts as a vertical bar chart with percentage values.

One of the features of the platform's interface is the time filter selection panel for displaying reports. This menu includes quick-select buttons for date ranges, such as the last 7, 30, or 90 days, or the current week and month. Additionally, by clicking on the time period, it can be manually set using a pop-up calendar. Another noteworthy feature is the ability to export any report card as a PDF or PNG file.

- **Strengths:**

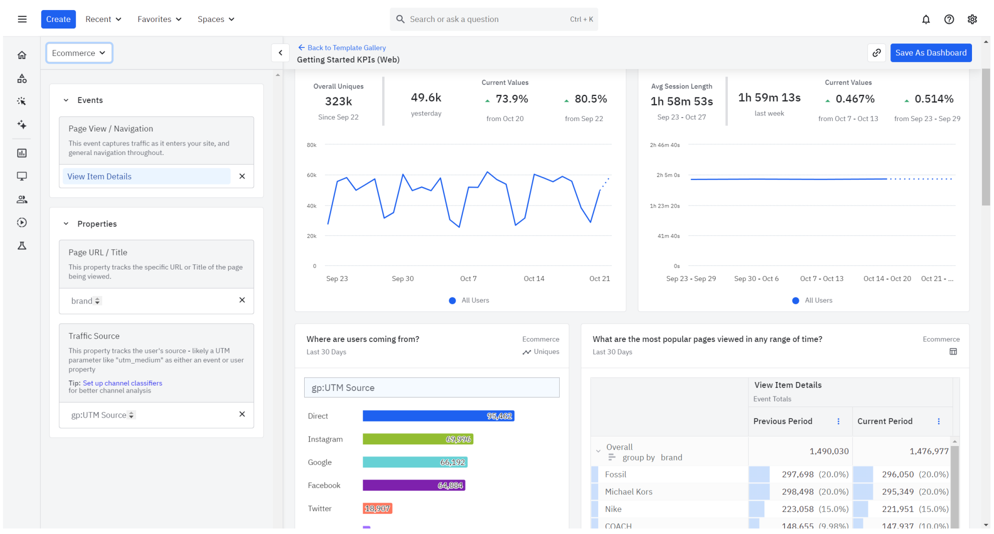
- User-friendly interface, easier to use for the target audience of product managers and marketers.
- The quality categorization of statistics was thoroughly analyzed by the development team within the business domain of analytical tools.

- **Limitations:**
  - Limited free version and one of the most expensive paid plans from all of the analyzed solutions.
  - Not many traditional metrics for user activity analysis.

3.3 Amplitude

Amplitude is another product analytics platform that helps teams understand user behavior with digital products. Amplitude emphasizes event-based analytics, providing insights into how users interact with various features of a product. [17]

One of Amplitude's starter templates provides visualizations of key KPIs for web platforms. Platform developers define web KPIs as measurable values that indicate how effectively a website achieves its objectives. Some of the most commonly used indicators are page views, unique visitors, conversion rates, and traffic sources. The platform also provides a filter that segregates user interactions by their interaction type before displaying the KPIs report.



Page View / Navigation
This event captures traffic as it enters your site, and general navigation throughout.
View Item Details

Page URL / Title
This property tracks the specific URL or Title of the page being viewed.
brand

Traffic Source
This property tracks the user's source - likely a UTM parameter like "utm_medium" as either an event or user property
Tip: Set up channel classifiers for better channel analysis
gp:UTM Source

	View Item Details Event Totals Previous Period	 Current Period
Overall group by brand	1,490,030	1,476,977
Fossil	297,698 (20.0%)	296,050 (20.0%)
Michael Kors	298,498 (20.0%)	295,349 (20.0%)
Nike	223,058 (15.0%)	221,951 (15.0%)
COACH	148,655 (9.98%)	147,937 (10.0%)

■ **Figure 3.4** Amplitude dashboard [17]

- **Strengths:**
  - Scalable for large data sets of user activities.

  - Useful reports that provide fundamental metrics for business decisions.
  - The switchable layout of the dashboard is immediately available when viewing statistics, rather than during its configuration and building.

- **Limitations:**
  - Requires a clear strategy on what events to track, can be steep for teams unfamiliar with product analytics.
  - Poor performance on the report pages, long loading times, and lack of smoothness when viewing the page.

3.4 Conclusion

The main reason why none of the ready-made solutions on the market were chosen for the implementation of the Experts.ai analytics portal is that such services are most often designed for large-scale applications with high traffic volume and a wide range of interaction sequences users can perform. The Experts.ai platform does not yet have a sufficiently large scope of user activities on the widgets, so most statistics on ready-made services cannot even be filled with data or visualized.

In addition to the fact that the cost of ready-made solutions is excessive for the activity level of the Experts.ai platform, the pricing plans of such services do not align with the vision of the Reporting Tool by EDUMATCH management. Access to an organization's statistics is planned for all its participants, meaning there will be no limit on the number of Reporting Tool users depending on the number of active organizations on the platform. At the same time, the pricing plans of all the analytics services listed in this research are designed for a limited number of users with access to the analytics portal, typically only for the data analysts or marketing department, and the statistics are provided globally for the entire system, not for each organization individually.

Due to this mismatch with the wishes of the Experts.ai clients regarding the final functionality of the statistics, and since EDUMATCH does not want to transfer data about organizations and user activities to third parties, it was approved that the Reporting Tool will become an extension of the current platform rather than an external third-party service. However, many of the advantages and designs of existing solutions will be adopted and serve as inspiration for the design of the Reporting Tool user interface. The available functionality will be considered when analyzing and drafting the requirements for the internal solution.

Chapter 4

Charting libraries

As mentioned in Section 1.1.2.1, charts from the Google Charts library cause numerous issues in the Experts.ai portal interface and do not meet the client's design requirements. This chapter contains an analysis and comparison of other libraries for rendering charts on web pages, to select one that will be actively applied in the development of the Reporting Tool module.

4.1 ApexCharts

ApexCharts is a modern library designed for developers to make interactive web visualizations with minimal configuration. It provides a variety of predefined chart types and offers lots of powerful interactivity customization, like chart zooming, time panning, and tooltip customization. ApexCharts has official wrapper for Angular, making it easy to integrate with Experts.ai frontend. ApexCharts has a feature for exporting charts to a PDF file but it requires a commercial license. [18]

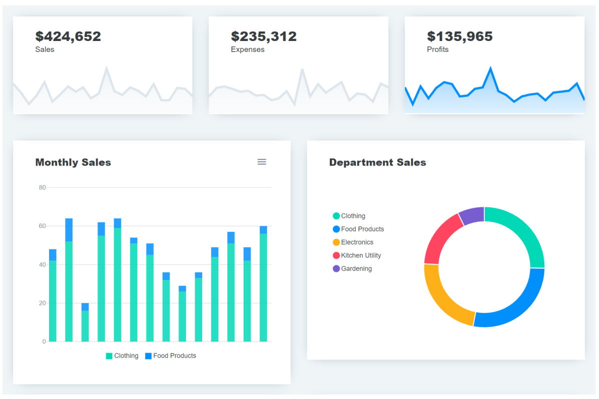


Figure 4.1 ApexCharts [18]

4.2 Apache ECharts

ECharts is a powerful library, especially for large-scale data visualizations. It supports various advanced features such as heatmaps, tree diagrams, and 3D charts, making its primary use case for complex visualizations. [19] The integration with Angular, while possible, is not as seamless as with other libraries. Charts customization can be difficult for simple use cases, where more straightforward libraries might be more efficient.

Figure 4.2 Apache ECharts [19]

4.3 Chart.js

Chart.js is a popular open-source charting library known for its simplicity and ease of use. This library is lightweight and performance-friendly, which are important factors for the platform infrastructure and smoothness of user experience. Integration in Angular is performed by using the wrapper library `ng2-charts`, which, in addition to providing convenient prebuilt chart components, also utilizes RxJS reactivity, which is very important for modern Angular applications. Additional advantages of Chart.js include extensive documentation and significant community support through contributions to the repository and the development of plugins with additional functionality. [20] Chart.js provides a wide range of fundamental chart types, making it suitable for implementing a statistics dashboard.

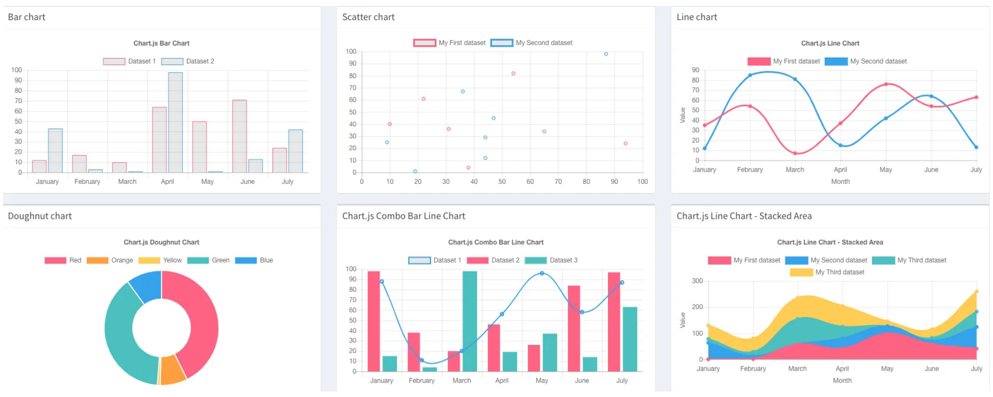


Figure 4.3 Chart.js [20]

4.4 Conclusion

As a result of comparing chart visualization libraries in Table 4.1, Chart.js is the library chosen for chart visualization of the Reporting Tool, which will be added to the Experts.ai frontend project along with the `ng2-charts` wrapper.

Criteria	ApexCharts	ECharts	Chart.js
Angular Integration	+	∼	+
Customization	∼	+	+
Performance	+	∼	+
Interactivity	+	+	∼
Documentation	∼	∼	+
Community support	∼	∼	+
Responsive design	+	+	+
Ease of use	∼	—	+
Licensing	—	+	+

Table 4.1 Comparison of ApexCharts, ECharts, and Chart.js

Part II

Practical part

Chapter 5
Analysis

This chapter is dedicated to defining the requirements of the planned frontend, following the needs of Experts.ai management. Analyzing customers' requirements at the start of development is essential to preventing potential issues and planning errors that may occur later.

To analyze the requirements, it was necessary to discuss the needs for the functionality and technical specifications of the Reporting Tool's frontend with EDUMATCH's management. Discussions were held in person at the company's office and via online calls using Microsoft Teams.

During these discussions, the management was informed about the research performed on existing solutions in Chapter 3 to explore ideas proven by experience that can be adapted to the context of the Experts.ai platform. Conclusions from each discussion were adjusted to fit the standard methods of software engineering analysis, such as requirement analysis and use case modeling.

5.1 Requirements analysis

Requirements analysis is a vital stage in the software development life cycle, where the functional and non-functional requirements of the project are identified, documented, and confirmed with the client. While functional requirements define the specific tasks and features the system must perform, non-functional requirements describe the system's quality attributes, such as performance, supportability, and usability. [21]

The documented requirements will later help to define the conditions under which the project will be considered complete and at the Minimal Viable Product (next MVP) state. In addition, this analysis will help determine the list of functionalities for the final frontend, which will be agreed upon with the client, represented by EDUMATCH.

5.1.1 Functional requirements

Based on the analysis of the platform's functionality, the analysis of existing solutions, and consultations with the client, the following functional requirements were defined for the Reporting Tool's frontend application:

- **F1 — Item statistics overview**
  The reporting module must allow users to view visualized statistics on widget visitors' interactions with a specific item.
  Priority: High
  Complexity: High
  Type: Functionality
- **F2 — Widget statistics overview**
  The module must allow users to view visualized statistics on how widget visitors interact with items on a specific widget.
  Priority: High
  Complexity: High
  Type: Functionality
- **F3 — Organization statistics overview**
  The module must allow users to view visualized statistics on how widget visitors interact with the items or widgets of a specific organization.
  Priority: High
  Complexity: High
  Type: Functionality
- **F4 — Platform global statistics overview**
  The module must allow users to view visualized statistics on how many interactions visitors perform with the items or widgets of a specific organization.
  Priority: High
  Complexity: High
  Type: Functionality
- **F5 — Navigation between statistics**
  The module will have a clear and intuitive navigation functionality between the statistics pages of related entities (organizations, widgets, or items). Users will also be able to switch between the evidence and statistics pages of the current entity, allowing them to immediately make changes to the entity after analyzing its statistics.
  Priority: High
  Complexity: Low
  Type: Functionality
- **F6 — Statistics filtering**
  Users will have the ability to filter which interactions will be counted and

displayed in the statistics and which will not. Module will provide filters by date range, interaction types, specified organization owners of interacted items or widgets where the interaction occured. Both global filters for the entire statistics page and individual filters for visualizations will be available.
Priority: High
Complexity: Middle
Type: Functionality

- **F7 — Statistics resizing**
  On the statistics pages, users will be able to control the visualization cards layout on the web page. Layouts will differ by resizing visualizations, hiding less important ones, and expanding more detailed visualizations to full screen.
  Priority: Low
  Complexity: Low
  Type: Functionality

- **F8 — Exporting statistics**
  Users will have the ability to export visualized statistics in PDF format or export statistical data in CSV format. It will be possible to export each card individually or the entire statistics page.
  Priority: Middle
  Complexity: Middle
  Type: Functionality

- **F9 — Widgets evidence**
  The Experts.ai platform will be expanded with new functionality for creating unified widgets, and each organization will have an administration page for managing its widgets' instances.
  Priority: High
  Complexity: Middle
  Type: Functionality

5.1.2 Non-functional requirements

The EDUMATCH company, based on the conducted research, expected the following non-functional requirements from the final Reporting Tool module:

- **NF1 — Extension of Experts.ai frontend**
  The resulting Reporting Tool module must be an extension of the current Experts.ai web application, written in the TypeScript programming language using Angular framework.
  Priority: High
  Complexity: High
  Type: Supportability

- **NF2 — Visualization of charts using Chart.js**
  In the previous chapter (see Chapter 4), libraries for web-based chart rendering were analyzed. Due to its advantages, the Chart.js library was selected, and the Reporting Tool will primarily be implemented using this library.
  Priority: High
  Complexity: Middle
  Type: Usability

- **NF3 — Module extensibility**
  The module must have a structure that can easily be extended in the future with new reports, visualizations, and filters.
  Priority: High
  Complexity: High
  Type: Supportability

- **NF4 — REST API**
  The module should be integrated with the Experts.ai backend and receive statistical data for rendering through REST API communication. The implementation of new backend endpoints with business logic and data aggregation was developed by Roman Chertishchev as part of his bachelor thesis. [22]
  Priority: High
  Complexity: High
  Type: Performance

- **NF5 — Responsiveness**
  The module's interface must be responsive so that the visualizations and content of the Reporting Tool are always displayed correctly and in proper proportions, regardless of the user's browser resolution.
  Priority: Middle
  Complexity: Low
  Type: Usability

- **NF6 — User Experience**
  The graphical user interface of the module must be intuitive and self-explanatory for users who are just starting to use the Reporting Tool. No detailed instructions or explanations from Experts.ai developers should be required for users to work with it.
  Priority: High
  Complexity: Middle
  Type: Usability

- **NF7 — Testing**
  User Acceptance Testing must be conducted on the final solution to verify the intuitiveness and user-friendliness of the interface. This testing will

help identify design issues and confirm whether the module meets all the stated requirements.
Priority: High
Complexity: High
Type: Reliability

5.2 Use case modeling

Unified Modeling Language (next UML) diagrams, an industry standard actively employed across development methodologies, are used to specify, visualize, and document software systems. One of these types of diagrams was created to affirm and describe the purposes for which end users will utilize the Reporting Tool.

UML Use Case Diagrams illustrate how users (referred to as *actors*) interact with the system by engaging in specific functions or *use cases*. Each use case represents a particular functionality or task the system performs in response to an actor's actions. The main objective of a use case diagram is to capture how a system interacts with external entities, providing a high-level view of the system's functionality. [23] An Additional way to document use cases is to write scenarios of user actions for each use case.

5.2.1 Actors

An actor represents the role of a user who interacts with the modeled system. The list of actors for this use case diagram is based on the role and authorization system of the Experts.ai backend, described in Section 1.2.2.

Organization viewer A participant of an organization who can view its statistics as well as the statistics of its widgets and items. Additionally, they can access the organization's widget section and copy the HTML iframe code to embed the widget catalog on their website.

Organization editor Can perform all the same actions on the entities of their organization as the Organization viewer. In addition, they are granted permission to create, edit, and delete instances of widgets belonging to the organization in which they hold this role.

Portal administrator Has access rights to all organizations on the platform, and can therefore perform any action that a viewer or editor can for any organization. Moreover, in the portal administration menu, the administrator can access global statistics for the entire platform rather than for each organization individually.

5.2.2 Use cases

The final UML Use Case model is presented in Figure 5.1, and below is the textual documentation of all the use cases scenarios for this project.

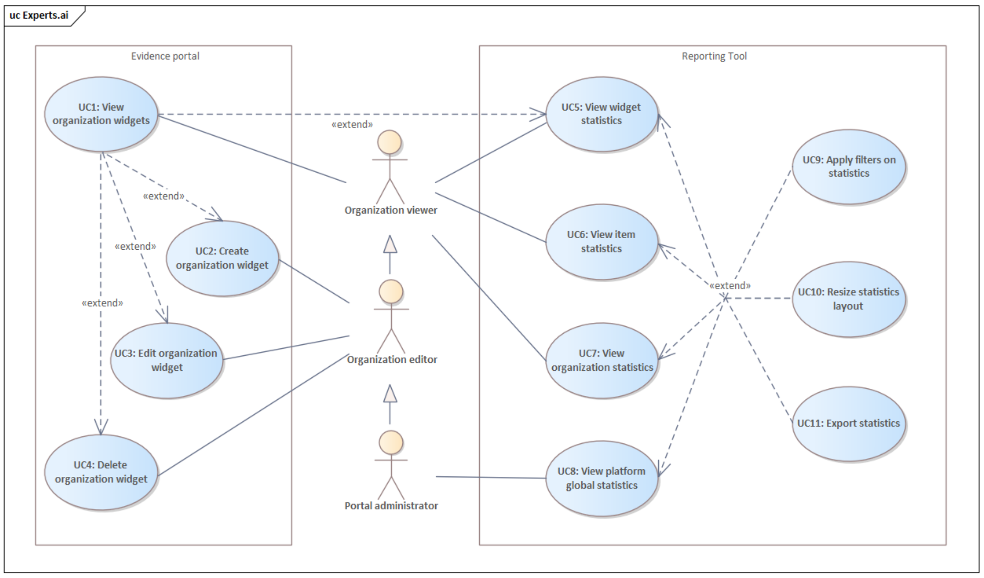

uc Experts.ai
Evidence portal
Reporting Tool
UC1: View organization widgets
UC2: Create organization widget
UC3: Edit organization widget
UC4: Delete organization widget
UC5: View widget statistics
UC6: View item statistics
UC7: View organization statistics
UC8: View platform global statistics
UC9: Apply filters on statistics
UC10: Resize statistics layout
UC11: Export statistics
«extend»
Organization viewer
Organization editor
Portal administrator

Figure 5.1 UML Use Case Diagram

UC1: View organization widgets

1. Viewer enters the overview page of the organization by the portal's sidebar.
2. Viewer clicks the "Widgets" button to access the evidence page of the organization's widgets.

UC2: Create organization widget

1. On the organization's widgets page, the editor clicks the button for new widget creation to access the widget constructor page.
2. Editor selects a subset of item type components that will be available in the widget and moves to the next constructor step.
3. Editor chooses whether the widget will have an organization header or not and moves to the next step.
4. Editor sets the name of the widget and then saves it to generate the HTML code for it.

UC3: Edit organization widget

1. On the organization's widgets page, the editor clicks the "Edit" button on the card of the desired widget instance.
2. System shows the modal screen with the form of widget configurations.
3. Editor changes the options in the modal screen and clicks the "Save" button.
4. System edits the widget's configuration.

- **UC4: Delete organization widget**

1. On the organization's widgets page, the editor clicks the "Delete" button on the card of the desired widget.
2. System erase the widget instance from the platform'sdatabase and deactivate all its `<iframe>` codes.

- **UC5: View widget statistics**

1. On the organization's widgets page, the viewer clicks the "Statistics" button on the card of the desired widget to access the widget's statistics.

Alternative scenario:

1. On the related entity's statistics page, the viewer clicks on the desired widget's name to access its statistics.

- **UC6: View item statistics**

1. Viewer enters the evidence page of the desired item type belonging to the organization that owns the desired item by the portal's sidebar.
2. Viewer finds the desired item in the list of items (including using the search filter).
3. Viewer opens the item's evidence window.
4. Viewer clicks the button in the lower right corner of the screen to access the item's statistics.

Alternative scenario:

1. On the related entity's statistics page, the viewer clicks on the desired widget's name to access its statistics.

- **UC7: View organization statistics**

1. Viewer enters the organization overview page by the portal's sidebar.
2. Viewer clicks the button in the lower right corner of the screen to access the organization's statistics.

- **UC8: View platform global statistics**

  1. Administrator enters the portal administration page by the portal's side-bar.
  2. Administrator clicks the "Platform global statistics" option to access the platform's statistics.

- **UC9: Apply filters**

  1. Viewer on the report page sets the desired filters for the statistics report.
  2. System updates all the visualizations in the report according to the new filter.

- **UC10: Resize statistics layout**

  1. Viewer on the report page chooses one of the layout options for the report section.
  2. System resizes and rearranges report cards according to the user's chosen layout.

- **UC11: Export statistics**

  1. Viewer on the report page clicks the "Export" icon on the report card header.
  2. Viewer chooses a file format to export statistics, if the chosen format was:
     a. PDF: system lets the viewer download a file with saved visualization as an image
     b. CSV: system lets the viewer download a file with plain data of the statistics before visualization.

5.2.3 Functional requirements coverage

When analyzing the planned system, it is important to ensure that, upon full implementation, the final solution will meet all requirements established with the client. For this purpose, Table 5.1 was created, reflecting that each functional requirement is covered by at least one use case of the Reporting Tool. So, it can be said that if the Reporting Tool allows end users to complete all the listed use cases, then the final solution will meet all functional requirements.

	F1	F2	F3	F4	F5	F6	F7	F8	F9
UC1	+				+				
UC2		+			+				
UC3			+		+				
UC4				+	+				
UC5						+			
UC6							+		
UC7								+	
UC8									+
UC9									+
UC10									+
UC11									+

Table 5.1 Functional requirements coverage

Chapter 6

Design

This chapter contains all activities aimed at describing the structure of the future application. First, it establishes a conceptual model of the problem domain according to the research of the Experts.ai platform. Then, it outlines the design of the interface, with a strong emphasis on user experience. Lastly, the chapter contains details of the REST API documentation, which the frontend will communicate with.

6.1 Conceptual model

The main entities with which the platform's end users interact were described in Section 1.1.1, and the technologies for their storage and administration in Section 1.2.1. During the analysis of the Experts.ai challenges and the design of the Reporting Tool module for collecting user activity on widgets, a conceptualization was created using the domain model.

The domain model is a visual representation of the problem domain, its purpose in this project was to accurately depict the main entities, their attributes, and relationships between them. [24] The final version of the domain model is presented in Figure 6.1.

This domain model was subsequently applied in the architectural design to extend the Experts.ai database as part of Roman Chertishchev's bachelor thesis. The data storage structure for interactions on widgets was considered in the further design of all available visualizations and statistics of the Reporting Tool frontend, to assess how aggregable and possible they are only from the data that will be available from the backend part.

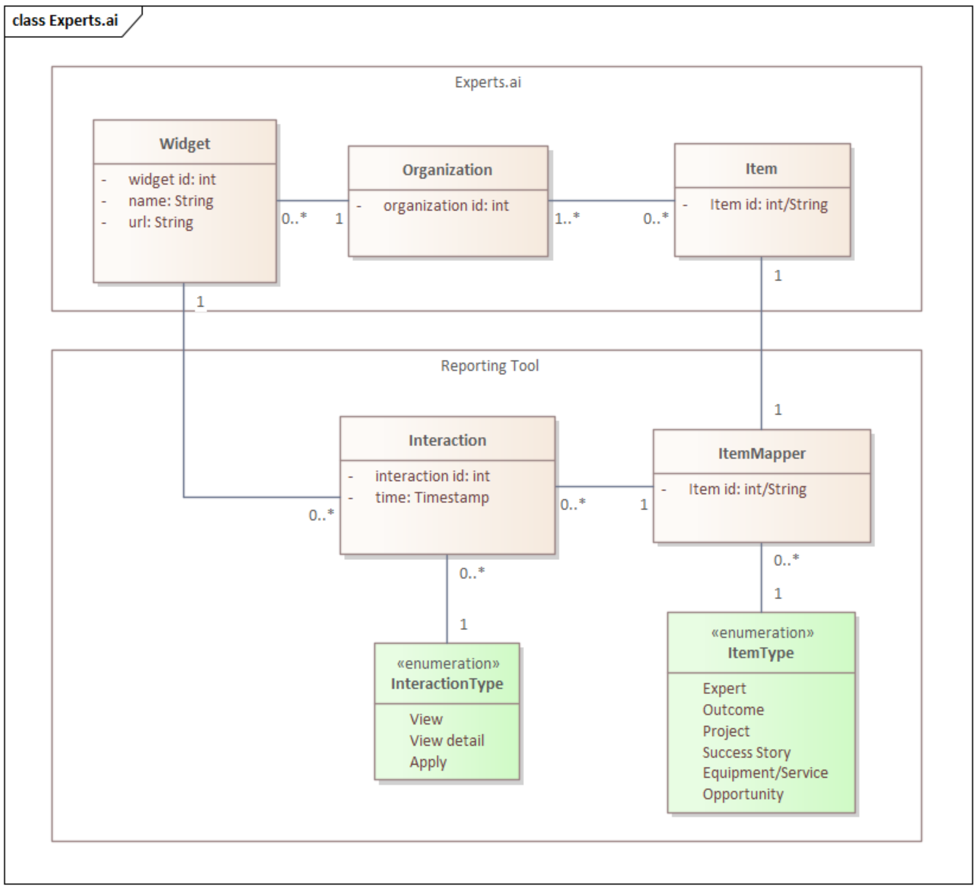


Figure 6.1 Domain Model

It was agreed with the EDUMATCH organization that, as part of the collaborative work for the MVP project state, it will be sufficient for the Reporting Tool module only to monitor three types of user interactions with widgets, that were stated in Section 1.1.1. These interaction types are “View”, “View detail” and “Apply”.

That said, the following information about user interaction is available in the project database: [22]

- **Interaction id** for identifying interactions from each other.
- **Timestamp** when the interaction was performed.
- **Interaction type** enumeration value according to the abovementioned description (can be “View”, “View detail” or “Apply”).
- **Item type** enumeration value as one of the intellectual assets’ categories provided on the platform (can be “Expert”, “Research outcome”, “Research project”, “Success story”, “Equipment/Service” or “Opportunity”).

- **Item id** with which the user interacted. The set of organizations that own this item can be found through this value in the platform's database.
- **Widget id** on which the interaction occurred. An organization that owns this widget can be found through this value in the platform's database.

On the backend part of the Reporting Tool module, these interaction records must be aggregated and transformed into values that can be conveniently visualized on various charts and tables.

6.2 Interface prototyping

The next important part of frontend software development is user interface design (UI). UI design focuses on the visual aspect (appearance and styling) and interactive elements (ease of use) of the final product. A related but broader concept is user experience design (UX), which aims to understand the needs and behaviors of the end user. The main tools of UX design are user research, information architecture, and product prototyping. [25]

UI/UX prototyping is the process of creating simplified digital mockups of the product or application being developed. Application prototypes do not include the final functionality required for the planned product nor reflect the final design. Prototyping allows for feedback from end users and testing of the application's design intuitiveness before beginning the full implementation of the interface. [26]

Prototypes are divided into two main types:

Low-fidelity (lo-fi) These prototypes, also known as wireframes, only generally reflect the design and visual aspects of the final application, focusing on functionality and the approximate layout of the future interface. Lo-fi prototypes are styled as sketches and consist of text, buttons, and black-and-white rectangles that show the approximate content placement in the interface. These prototypes are very quick to create and help identify interface issues early in the design phase of the future application.

High-fidelity (hi-fi) More detailed and colored representations that closely resemble the final look of the interface. The goal of a hi-fi prototype is to provide a realistic preview of the interface and evaluate the design's appearance, appeal, and branding.

Both types of prototypes can be interactive or not. Interactive prototypes simulate working with the final interface, for example, by adding navigation between interface screens when clicking on interactive buttons. Interactive prototypes allow usability testing with end users, who can then provide feedback on any interactions they find confusing or unclear.

The goal of this section is to provide a textual description of the completed UI design of the Reporting Tool. The final design of the user interface must meet all the documented requirements in Section 5.1, including being intuitive enough for first-time users, in line with requirement NF6. All visualizations set on the statistics pages must be verified to ensure they can be aggregated from user interaction records in the Reporting Tool database (see Section 6.1).

To support the textual documentation, low-fidelity interactive prototypes of the interface were designed using *Balsamiq* [27], a tool for sketching and simplistic visualizing of user interfaces. This application contains not only a large library of pre-made interface elements in lo-fi style (buttons, labels, input fields, menus, option selectors, layout elements) but also functionality to switch interface screens when clicking on hyperlinks (buttons, menu items), making these wireframes interactive. This will allow for interactive UI testing with end users to validate the design with EDUMATCH management. All created wireframes can be viewed in the attachment of this thesis.

6.2.1 Statistics page header

Each statistics page in the Reporting Tool module starts with a header containing the name of the current entity and a menu for configuring filters. The header can be seen in Figure 6.2 and allows managing several display options for statistics, which cover the functional requirement F6 from documentation in Section 5.1.1.

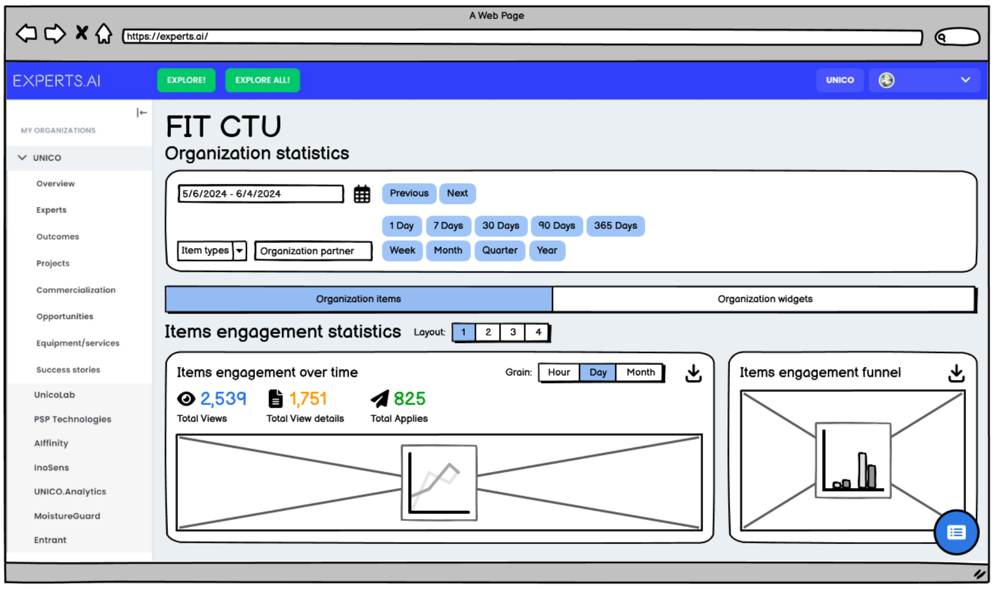


Figure 6.2 Statistics page header

Filters are located on the left side of the menu bar. They are designed to calculate statistics by only considering interactions corresponding to the user-defined properties.

- **Date range filter** allows selecting a time range for which interaction records in the database will be considered. The interval's start and end dates can be set manually or by using quick selection buttons. The quick selection buttons are divided into two types: options to select the last number of days from the current day (1, 7, 30, 90, or 365 days) or options to select the period in which the current day falls (week, month, quarter or year). Navigation between intervals is possible using the "Previous" and "Next" buttons, which move the current date range by its length back or forth. The filter is mandatory for use, with the default selection being the last 30 days from the current day.
- **Item types filter** is a dropdown list of checkbox elements, allowing the user to select a subset of intellectual asset types from the platform that have been interacted with. The filter is mandatory for use, and by default, if no checkbox is selected in the list, it is the same as considering interactions for all item types in the resulting statistics. The filter is not available on the item statistics page, as all interactions displayed in this report are obviously performed on a single item type — the type of the current item.
- **Partnership filter** is a text input field for the organization's name, which allows filtering interactions that occurred with the assistance of the specified organization. For example, in widget statistics, partners are organizations that provide items on the widget, and for item statistics, partners are the organizations owning widgets where interactions with the item occur. This filtering can help evaluate the development of business relationships with a collaborating organization. As the name is entered, auto-complete suggestions help the user avoid typing the entire organization name. The filter is optional for use.

6.2.2 Organization items statistics

When a user is on any page of the Evidence portal, a round Floating Action Button (next FAB) with a chart icon appears and hovers in the bottom right corner of the screen. Upon clicking it, the user is taken from the current organization's evidence page to the statistics page, which is presented in Figure 6.2. The FAB changes icon and clicking it again will redirect to the evidence page of the current entity (if the user is on the organization statistics page, by clicking it again he will redirect back to the organization evidence page). The page consists of the described header and a tab selection for viewing either the statistics of the organization's items or widgets. Depending on the selected tab, a long scrollable dashboard report follows, consisting of report cards grouped by named sections.

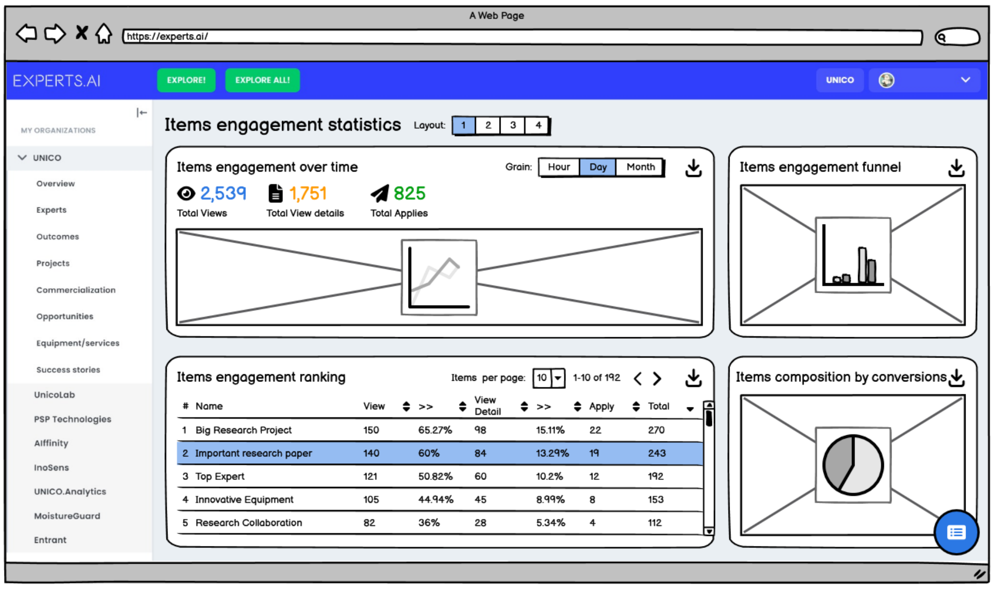
In the statistics tab selection, the organization items tab is set as the default option because the vast majority of organizations on the platform have uploaded items but no widgets, and organizations with widgets are likely to have their own items as well. Therefore, the primary view of the organization statistics page initially displays the statistics of its items, while promoter organizations can use the tab to switch to the widget statistics they need.

6.2.2.1 Engagement section

From the research of available interactions on widgets in Section 1.1.1, the following KPIs were identified by which organizations will assess performance in terms of how effectively they are achieving certain business objectives:

- Numeric amount of “View” interactions.
- Numeric amount of “View detail” interactions.
- Numeric amount of “Apply” interactions.
- Conversion percentage rate from “View” to “View detail” interaction.
- Conversion percentage rate from “View detail” to “Apply” interaction.

Using these indicators, Experts.ai clients can determine the results of their cooperation with EDUMATCH, the benefits they gain from uploading their offers to the platform, and how changes in offers or widgets affect activity growth.



#	Name	View	>>	View Detail	>>	Apply	Total
1	Big Research Project	150	65.27%	98	15.11%	22	270
2	Important research paper	140	60%	84	13.29%	19	243
3	Top Expert	121	50.82%	60	10.2%	12	192
4	Innovative Equipment	105	44.94%	45	8.99%	8	153
5	Research Collaboration	82	36%	28	5.34%	4	112

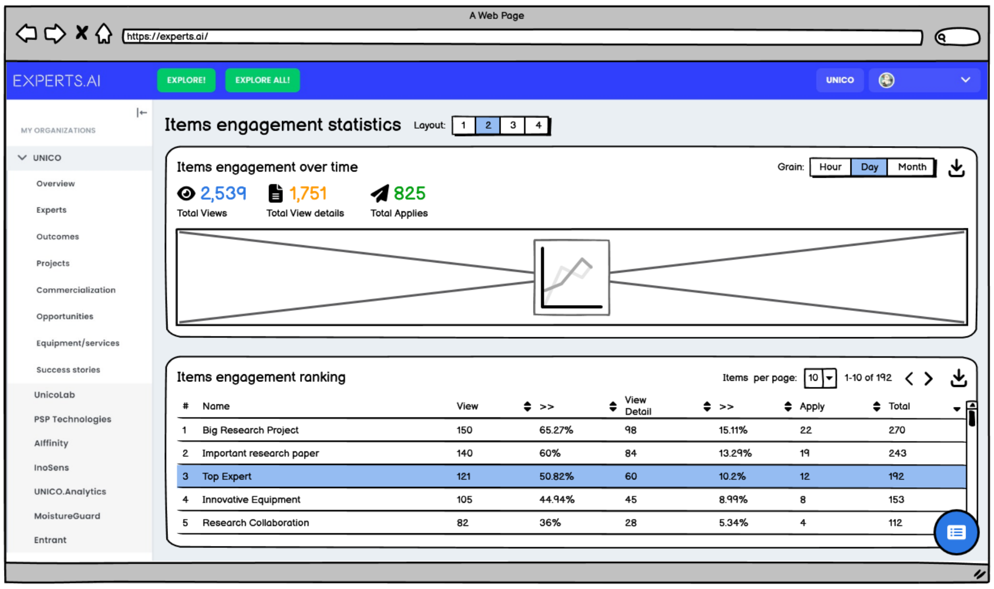
Figure 6.3 Organization items engagement statistics

All these metrics are visualized primarily in the first section called “Engagement”, which is presented in Figure 6.3 and consists of four report cards.

The structure of each report card includes the report card title, personal options that only affect the report card itself, and an export button (covering functional requirement F8). The engagement section consists of the following report cards:

- **Items engagement over time** displays the totals of three interaction types over a given date range and a line over time chart, reflecting trends in how the number of earned interactions changed over each unit of time. This report card has its own option allowing the selection of the chart's granularity, which determines the time step unit on the line chart. Possible granularity options are: "Hour", "Day", and "Month" (with "Day" selected by default).
- **Items engagement funnel** uses a vertical bar chart to show the totals of interaction types, but it also displays the percentage of users who convert from one interaction stage with the item to the next, and the percentage of drop-offs. This visualization allows organizations to identify at which stage of conversion most difficulties arise. For example, if "Apply" interaction rarely follows "View detail", the organization can identify and address this problem.
- **Items engagement ranking** sorts all the organization's items by a user-specified type of sorting (by default, sorting by the total sum of interactions in descending order is selected; sorting can only be done by numeric values) and displays their KPIs in table form. The ranking does not display all the organization's items at once but shows them in a scrollable window, divided into pages that can be navigated using a paginator in the upper right corner. Using this report card, organization representatives can identify their best and worst items to make data-driven decisions for their improvement. In addition to the ranking functionality, all rows in the table are clickable and redirect to the page of the item that was clicked.
- **Items composition by conversions** uses a pie/doughnut chart and divides all the organization's items into 4 categories: items with conversion rates to View detail and to Apply below 10%, only to View detail above 10%, only to Apply above 10%, and both conversions above 10%. The totals of these categories are then presented as a composition of a whole to determine which item categories dominate and which are in the minority. This visualization provides an overview of all the organization's items' conversions, showing how many items have reasonable user conversion rates and how many need improvement to get users to the most important "Apply" interaction.

In this same section, the functionality covering functional requirement F7 regarding resizing statistics for user convenience is demonstrated. To achieve this, at the beginning of each section, a button toggle group is provided, allowing the user to select one of several layout settings for the section. Each layout changes the sizes of the cards, increasing the size of important cards that benefit from more space and hiding secondary cards. Figure 6.4 presents the second layout configuration for the section, which hides the secondary "Funnel" and "Composition" statistics in favor of "Over time" and "Ranking".

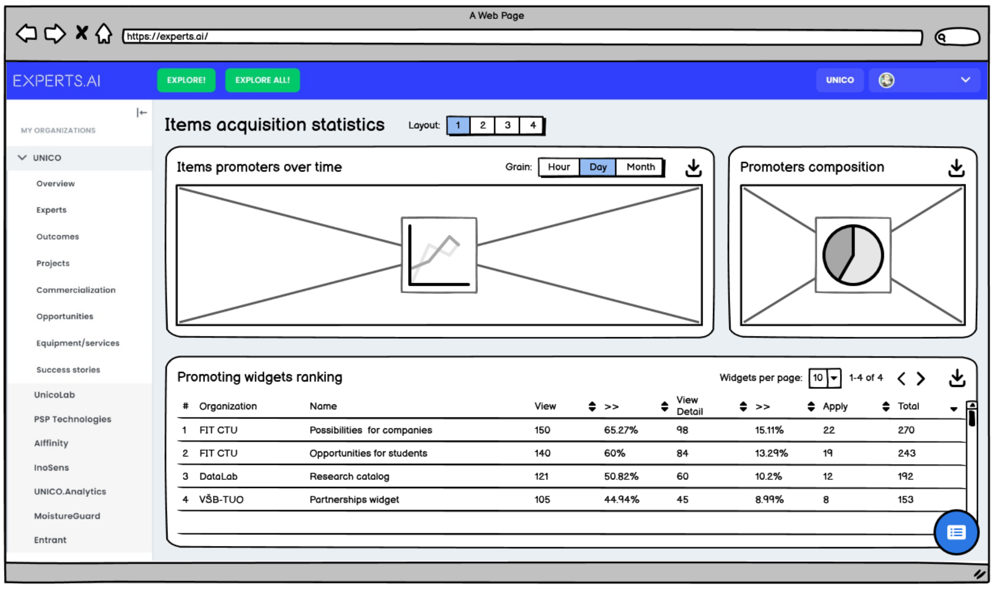


#	Name	View	>>	View Detail	>>	Apply	Total
1	Big Research Project	150	65.27%	98	15.11%	22	270
2	Important research paper	140	60%	84	13.29%	19	243
3	Top Expert	121	50.82%	60	10.2%	12	192
4	Innovative Equipment	105	44.94%	45	8.99%	8	153
5	Research Collaboration	82	36%	28	5.34%	4	112

Figure 6.4 Organization items engagement statistics, second layout

6.2.2.2 Acquisition section

This section is dedicated to visualizing the sources from which the organization acquires traffic and interactions with its items and is depicted in Figure 6.5. Since interactions with the items occur through widgets, these sources are the active widgets of promoting organizations. It is important to clarify that in this section, the organization to which this report belongs may also be displayed, as organizations can promote their own items on their widgets as well as those of others. These statistics will help organizations compare which of their partners brings the most traffic and user activity, with whom it is worth strengthening partnerships and establishing new campaigns to attract an audience.



#	Organization	Name	View	>>	View Detail	>>	Apply	Total
1	FIT CTU	Possibilities for companies	150	65.27%	98	15.11%	22	270
2	FIT CTU	Opportunities for students	140	60%	84	13.29%	19	243
3	DataLab	Research catalog	121	50.82%	60	10.2%	12	192
4	VŠB-TUO	Partnerships widget	105	44.94%	45	8.99%	8	153

Figure 6.5 Organization items acquisition statistics

In this section, the following reports are provided:

- **Items promoters over time** on the line over time chart displays trends on how organizations, through widgets, generate interactions with items over a unit of time. Each promoting organization is assigned its own color line, and the report card has the same grain selection option as "Items engagement over time".
- **Promoters composition** collects the number of interactions provided by organizations and displays each promoting organization as part of a whole on a pie/doughnut chart.
- **Promoting widgets ranking** sorts all widgets that bring interactions with the current organization's items in a table. Since the ranking may include widgets from different organizations, a column for "Organization" is added for each item, unlike the "Items engagement ranking" table. The rows are also clickable and link to the selected widget's statistics page.

6.2.2.3 Prime time and item types engagement sections

The organization items report concludes with two sections presented in Figure 6.6.

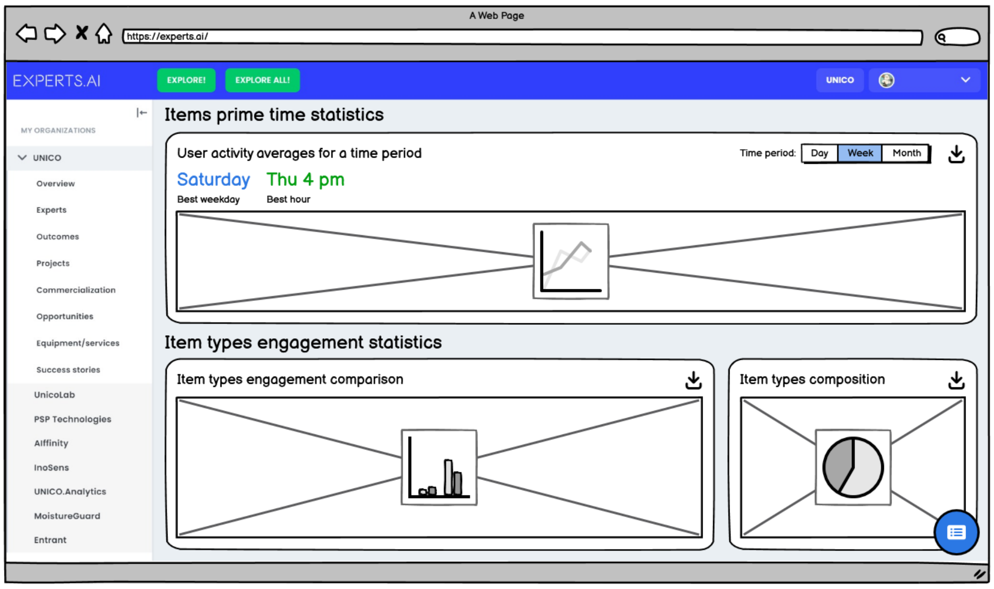


Figure 6.6 Organization items prime time and item types statistics

The concluding report cards on the organization items include:

- **User activity averages for a time period** is the only report card in the ”Prime time” section, which uses a line over time chart to show the average user activity over a unit of time for a specific time period. The time period is a personal option of this report card, with available options including “Day”, “Week”, and “Month”. For example, when the “Week” period is selected, the most popular day of the week and hour within the week are also displayed at the top of the chart. This statistic will allow organizational representatives to choose the time frame during the day, week, or month when they can reach the most users, such as scheduling a weekly newsletter or a monthly upload of new items.
- **Item types engagement comparison** consists of a vertical bar chart made up of six pairs of bars, with each pair representing one of the types of items provided on the platform. The left bar in each pair shows the number of interactions for that type of item, while the right bar displays the number of items of that type. The left bar corresponds to the metrics on the left vertical axis, and the right bar corresponds to the right vertical axis. The chart allows for a clear comparison of which types of items get the most activity based on the quantity provided by the organization on the platform.
- **Item types composition** shows the number of interactions for each type of item relative to the whole on a pie/doughnut chart.

6.2.3 Organization widgets statistics

The second tab of organization statistics is presented in Figure 6.7 and displays a report on the interactions that occurred on all widgets created by the organization. Its structure mirrors that of the organization items statistics, with a few changes listed below:

- All visualizations related to the three types of interactions are colored in three different colors so that users can intuitively distinguish between interactions conducted on the item and interactions conducted on the widget.
- The “Engagement” section consists of the same four report cards, but the statistics now reflect interactions with all widgets of the organization. The ranking table displays a sorted list of the organization’s widgets rather than items.
- The “Item types” filter and the “Item types engagement” section reflect which types of offers attract users most on the organization’s widgets.
- The “Acquisition” section was replaced by the “Promotion” section.

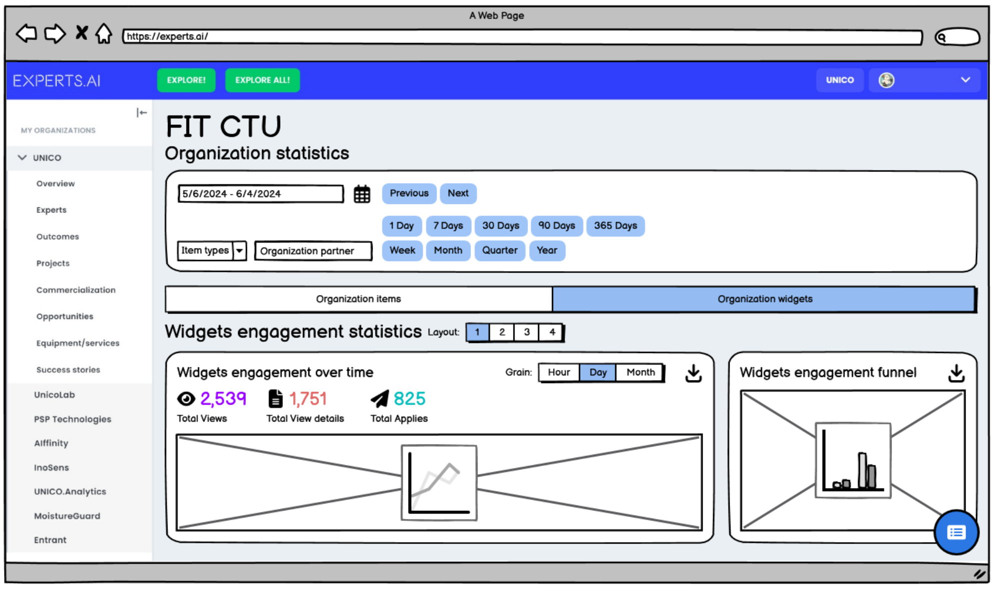
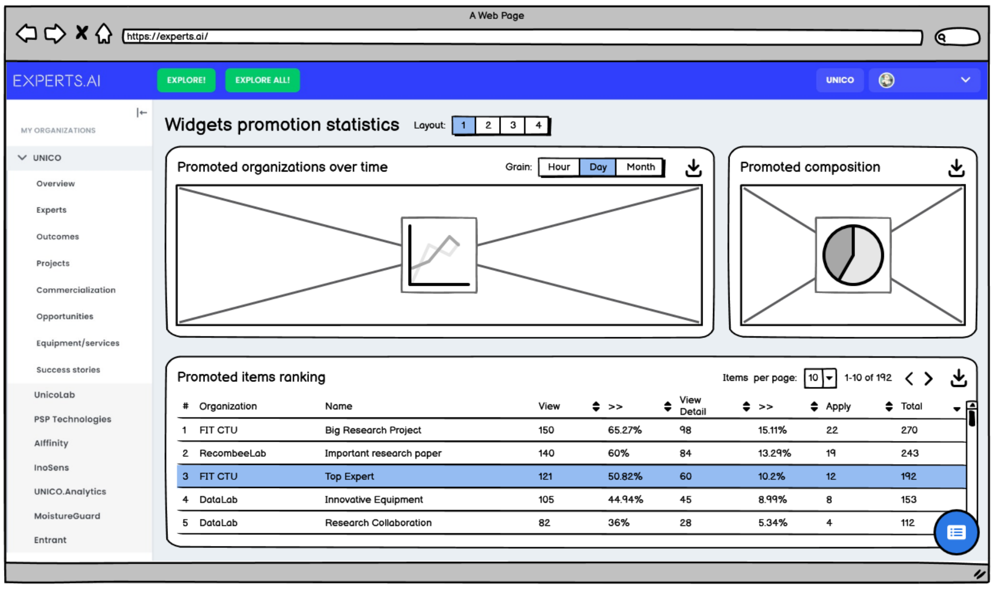


Figure 6.7 Organization widgets statistics

6.2.3.1 Promotion section

This section reflects how the organization’s widgets fulfill their role in promoting the offers of the organization or partner organizations. It allows for identifying which partners and offers are in greater demand among the audience of the current organization, and which partners the organization promotes more effectively through its widgets. Similarly to the “Acquisition” section and serving as its replacement in the widget report, this section is designed for the organization to highlight potential partners in advertising and strengthen business relationships.

- **Promoted organizations over time** shows how widgets provide partner organizations with user activity on a line over time chart, to highlight how trends and demand for partners change over time among the audience of the organization. Each partner organization has its own line on the graph, displaying the total of all types of interactions with that partner’s offers for each unit of time. This chart has an option for selecting granularity.
- **Promoted composition** on the pie/doughnut chart provides statistics on which organizations dominate demand among widget visitors and collectively receive the most interactions, and which do not earn enough interactions relative to the whole.
- **Promoted items ranking** sorts all items displayed on the organization’s widgets by activity and presents them in a table. Similar to the “Promoting widgets ranking,” an additional column labeled “Organization” appears in the table.



#	Organization	Name	View	>>	View Detail	>>	Apply	Total
1	FIT CTU	Big Research Project	150	65.27%	98	15.11%	22	270
2	RecombeeLab	Important research paper	140	60%	84	13.29%	19	243
3	FIT CTU	Top Expert	121	50.82%	60	10.2%	12	192
4	DataLab	Innovative Equipment	105	44.94%	45	8.99%	8	153
5	DataLab	Research Collaboration	82	36%	28	5.34%	4	112

Figure 6.8 Organization widgets promotion statistics

6.2.4 Widget statistics

The page with the statistics report for a specific widget can be accessed through the widgets evidence menu or by clicking on the line with that widget in the widget ranking table. This page, in addition to the changed header name and entity type, differs from the organization's widget statistics only in that the data aggregations are presented for a single widget rather than multiple widgets, and that in the "Engagement" section, there are only two report cards out of four displaying the main KPIs of the current widget —"Engagement over time" and "Engagement funnel". The report cards "Engagement ranking" and "Engagement composition" are not relevant in the context of a single entity rather than a collection of entities. All other sections remain unchanged.

6.2.5 Item statistics

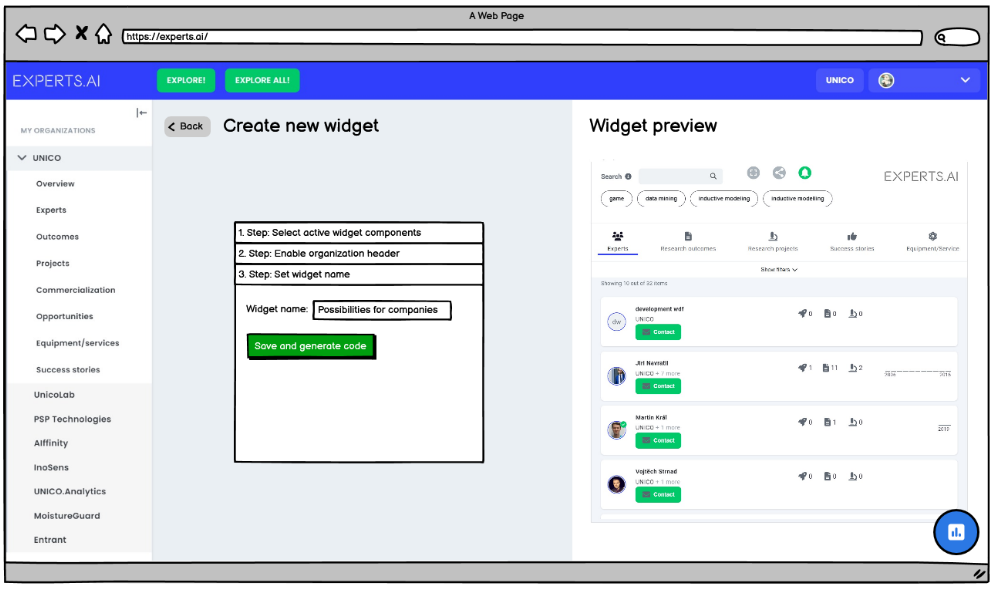
The page with the report for a specific item is accessible just like the organization by pressing the FAB button while on the item evidence URL, and similarly, from the statistics page, you can return to the item evidence page by pressing the button again. Another option to access the item statistics page is to click on its row in the item rankings. The structure of this report does not differ from the organization items report except for two points — the report cards "Engagement ranking" and "Engagement composition" are absent for the same reason as in the specific widget report, and the section "Item types engagement" is absent because interactions with the specific item occur only with its type.

6.2.6 Platform global statistics

The statistics of all organizations and all widgets are only accessible to platform administrators through the portal administration page. The structure of this statistics dashboard mirrors the layout of the organization statistics with two tabs, but the context of aggregations is now not limited to items and widgets of a single organization, but includes all items or all widgets available on the platform. This report allows for assessing the overall success and productivity of the platform, as well as tracking sudden drops in activity in case there is an error with access to resources or widgets on the platform.

6.2.7 Widgets evidence

As described in Section 1.1.2.2, the platform needs to be enhanced by unifying widgets so that each has its own identifier for tracking statistics. To achieve this, a new user interface for creating instances of widgets with different configurations is required. A new solution was designed consisting of two pages: one displays a list of the organization's active widgets and control buttons for them, while the second presents a visual flow for creating a new instance in which the user goes through several configuration steps. The final step allows the user to save the instance on the server, copy the `<iframe>` code, and insert it on their website.



■ **Figure 6.9** Widgets constructor

6.2.8 Prototypes evaluation

The user interface prototypes were provided to the EDUMATCH management, who also shared them with several end users of the evidence portal to gather user feedback on the intuitiveness and usefulness of the interface. The testers were representatives from partner organizations that are interested in the statistics of their entities. Additionally, the lo-fi prototypes were tested by the Experts.ai development team, specifically the global statistics page of the platform that is only available to portal administrators.

Testers feedback was positive; the interface was clear in terms of user experience and did not lead to misunderstandings, despite the fact that the lo-fi prototypes lacked detailed self-explanatory visualizations of the charts. The EDUMATCH management approved this interface design and the implementation will proceed according to this documentation.

6.3 REST API

The final phase in the design of the Reporting Tool frontend application is close collaboration with Roman Chertishchev, who is working on the backend part of the module. The primary task during this stage is to create comprehensive and well-structured documentation for the REST API that connects the client-side application with the server. This documentation will ensure stable parallel development of both parts of the module so that sudden, frequent changes in the frontend or backend do not become necessary due to corrections in the

API. This part of the application design was completed in accordance with the NF4 requirement.

6.3.1 Endpoints

During several design revisions, the final solution arranged the endpoints as nested resources. This is a hierarchical structure where each statistics page resource has sub-resources for sections, and each section resource has sub-resources for report cards. Each endpoint returns the aggregated data needed for a specific report card to ensure its visualization displays the required statistics.

- URIs of report pages:
  - Organization items statistics:
    /stats/organizations/{organizationId}/items
  - Organization widgets statistics:
    /stats/organizations/{organizationId}/widgets
  - Widget statistics:
    /stats/organizations/{organizationId}/widgets/{widgetId}
  - Item statistics:
    /stats/{itemType}/{itemId}
  - Platform global statistics:
    /stats/platform
- URIs of report sections:
  - "Engagement" section: **/engagement**
  - "Acquisition" or "Promotion" section: **/promotion**
  - "Prime time" section: **/prime-time**
  - "Item types engagement" section: **/item-types**
- URIs of report cards:
  - Line over time chart: **/over-time**
  - Funnel vertical bar chart: **/funnel**
  - Pie/doughnut composition chart: **/composition**
  - Comparison vertical bar chart: **/comparison**
  - Sortable table with KPIs: **/ranking**

In such a structure, only those endpoints are available that correspond to the layout of sections and report cards described earlier in Section 6.2. For example, a group of URIs is presented to which the frontend will send requests to gather aggregated data for visualizing the section ”Items engagement statistics” according to wireframe Figure 6.3:

```
GET /stats/organizations/{organizationId}/items/engagement/over-time

GET /stats/organizations/{organizationId}/items/engagement/funnel

GET /stats/organizations/{organizationId}/items/engagement/ranking

GET /stats/organizations/{organizationId}/items/engagement/composition
```

This structure will allow adding optional personal parameters to each individual visualization that apply only to it. Each endpoint for retrieving aggregated data for the report card requires specifying query parameters in the request with the structure presented in Table 6.1. These parameters are appended to the end of each request URI and correspond to the options in the filter header described in Section 6.2.1.

Name	**Type**	**Description**
startDate	ISO 8601 Timestamp	The initial time point from which interactions are considered in the report
endDate	ISO 8601 Timestamp	The final time point after which interactions are not considered in the report
itemTypes	String array	Types of items for which interactions will be counted in the report (not available in the item report)
partnerId	Integer	Identifier of the organization; only those interactions that benefit the current entity (promote or provide items) will be counted

Table 6.1 Statistics filters query parameters

To these mandatory query parameters, optional ones are added depending on the report card, for example `grain` or `timePeriod`. “Ranking” report cards have the largest number of personal parameters in order to implement server-side pagination and sorting. Organizations on the Experts.ai platform have a large number of items, and if the frontend manages pagination and sorting, it will affect the performance of the Reporting Tool web pages and the user experience. Introducing the parameters described in Table 6.2 allows the backend to sort and paginate the data instead of the frontend, returning only the prepared short page by the REST API.

Name	Type	Description
column	String	The value by which the elements will be sorted
direction	asc/desc	Selection of sorting in ascending or descending order
pageIndex	Integer	The current page of the table
pageSize	Integer	The maximum number of elements that the server will return

Table 6.2 Ranking options query parameters

The REST API backend of the server was also expanded with endpoints for constructing and managing widget instances, a new functionality that allows creating, reading, updating, and deleting unique widget records in the Experts.ai database.

`POST /internal/organizations/{organizationId}/widgets`

`GET /internal/organizations/{organizationId}/widgets`

`PUT /internal/organizations/{organizationId}/widgets/{widgetId}`

`DELETE /internal/organizations/{organizationId}/widgets/{widgetId}`

And to obtain the configuration of the current widget from the backend to render it, a new endpoint was allocated:

`GET /common/widget-instances/{widgetId}`

6.3.2 Security

During communication, the REST API returns responses with HTTP status codes; for example, any successful request is followed by a response with the code `200 OK`. If a user attempts to navigate to the organization's statistics page on the frontend without having the necessary roles to view the reports (as described in Section 5.2.1), the frontend will send REST API requests to a forbidden resource and receive an error in response with the code `403 Forbidden`. The frontend will handle these errors by displaying to the user an already existing modal window in the project code that informs them they are trying to access a page for which the external organization has not granted access, with an option to contact the organization's administrators to request access.

6.3.3 API Mocking

Since the frontend part of the Reporting Tool was developed in parallel with the backend part in Roman Chertishchev's bachelor's thesis, at the time of frontend application development, the documented API endpoints with statistics aggregations were not yet developed and available on the backend. In such cases, developers resort to a software development technique called API Mocking when backend functionality is not yet available. This approach allows developers to simulate the behavior of a future planned API by returning example service responses via HTTP communication with a mock server at specified endpoints with request bodies. [28]

The REST API of the Reporting Tool was documented with all the necessary endpoints, query params, path variables, request bodies and example responses, after which this documentation was uploaded to *Postman*, an API platform with tools for building, documenting, and using APIs. [29] *Postman* has native support for API mocking. For this, it was necessary to fill in a collection of requests with expected responses, after which the collection could easily be used to launch a mock server that runs on the *Postman* service host and is accessible via an internet connection at a dedicated URL (i.e., this is not a local solution). During development, the Reporting Tool interface communicated with the *Postman* mock API. However, over time, as the Experts.ai backend was extended with newly implemented endpoints, the *Postman* mock URL in the frontend source code was gradually replaced with the Experts.ai backend host URL. This way, the Reporting Tool frontend was integrated with the backend part.

Chapter 7

Implementation

This chapter is dedicated to the details of implementing the Reporting Tool frontend application. It includes descriptions of the Angular framework tools and libraries and the decisions made to implement the frontend according to the interface design described in Section 6.2. Code snippets of crucial parts of the implementation will support all these points. The screenshots of the implemented frontend can be viewed in the attachment of this thesis.

7.1 Routing

The routing of the Experts.ai application is implemented using Angular configuration methods described in Section 2.2. The configuration of the frontend application was expanded with the following views and URLs:

- **/statistics/organizations/{organizationId}:** Organization statistics
- **/statistics/organizations/{organizationId}/widgets/{widgetId}:** Widget statistics
- **/statistics/{itemType}/{itemId}:** Item statistics
- **/statistics/platform**: Platform global statistics
- **/evidence/organizations/{organizationId}/widgets:** Widgets evidence
- **/evidence/organizations/{organizationId}/widgets/constructor:** Multistep widget creation

Additionally, the routing in the widget module was also changed so that access to widget pages is now by its instance identifier. Now the URL for `<iframe>` looks as follows:

```
https://experts.ai/widgets/organizations/{organizationId}/{widgetId}
```

These URLs were added to the router module's route configuration in Listing 7.1. Navigation to the new views was implemented using `routerLink` directive attribute or `Router.navigate()` method.

```
this.router.navigate([`/statistics/organizations/\${baseId}`]);
```

```
<a class="btn btn--light-blue me-3" [routerLink]="'widgetEvidence' |
↪  path: (organizationId\$ | async)">
```

Code listing 7.1 Adding routes

```
{
  path: PathFragment.STATISTICS,
  children: [
    {
      path: PathFragment.ORGANIZATIONS + '/:' +
      ↪  UrlParams.ORGANIZATION_ID,
      component: OrganizationStatisticsComponent,
      children: [
        {
          path: PathFragment.WIDGETS + '/:' + UrlParams.WIDGET_ID,
          component: WidgetStatisticsComponent,
        }
      ]
    }, {
      path: ':' + UrlParams.ITEM_TYPE_STATISTICS + '/:' +
      ↪  UrlParams.ITEM_ID,
      component: ItemStatisticsComponent,
    }, {
      path: PathFragment.PLATFORM,
      component: PlatformStatisticsComponent,
    }]
},{
  path: PathFragment.EVIDENCE,
  children: [{
    path: PathFragment.ORGANIZATIONS,
    children: [{
      path: ':' + UrlParams.ORGANIZATION_ID,
      children: [
      ...
        }, {
          path: PathFragment.WIDGETS,
          component: EvidenceWidgetComponent,
          children: [
            {
              path: PathFragment.CONSTRUCTOR,
              component: WidgetConstructorComponent
```

7.2 Components structure

The future extensibility of the module in accordance with the non-functional requirement NF3 and the minimization of code duplication were among the important factors in development. The placement of Reporting Tool components in the Experts.ai frontend files is presented in Figure 7.1.

```
/components
  /reporting-tool
    report-card.component
    /report-card-contents
      chart-engage-over-time.component
      chart-promo-over-time.component
      chart-prime-over-time.component
      chart-funnel.component
      chart-composition.component
      chart-comparison.component
      table-ranking.component
    report-filters.component
    organization-statistics.component
    widget-statistics.component
    item-statistics.component
    platform-statistics.component
  evidence-widget.component
  widget-edit-modal.component
  evidence-widget-constructor.component
  ...
/services
  report-statistics.service.ts
  evidence-widget.service.ts
  ...
```

Figure 7.1 Components structure

To minimize code repetition for each report card, a unified interface was created for use in the component of each page. This component was implemented using the *Content projection* pattern [30] in Angular, which allows for the creation of complex components capable of accepting DOM elements as input. The area where the component accepts content is marked with the `<ng-content>` placeholder, and if it is necessary to add several such insertion points for content, the placeholders can be unified using the `select` attributes with different labels.

For example, in the Reporting Tool, the `report-card.component` consists of two placeholders, `report-card-options` and `report-card-content`. The first placeholder optionally contains personal options for each visualization in the report card header, while the second placeholder contains one of the visualizations stored in the `/report-card-contents` folder. An example of using this pattern is presented in Listing 7.2.

Code listing 7.2 Content projection example

```
<app-report-card [name]="'Items engagement over time'" class="col-12
↪  mb-4">

  <div report-card-options class="d-flex align-items-center">
    <div class="me-2">Grain:</div>
    <mat-button-toggle-group hideSingleSelectionIndicator=true
    ↪  (change)="updateEngagementGrain($event.value)">
      <mat-button-toggle [value]="Grain.HOUR">Hour</mat-button-toggle>
      <mat-button-toggle [value]="Grain.DAY">Day</mat-button-toggle>
      <mat-button-toggle
      ↪  [value]="Grain.MONTH">Month</mat-button-toggle>
    </mat-button-toggle-group>
  </div>

  <app-chart-engage-over-time report-card-content style="display:
  ↪  contents" [pathVariables]="pathVariables"
  ↪  [grain$]="itemsEngagementGrain$"></app-chart-engage-over-time>
</app-report-card>

<app-report-card [name]="'Items engagement funnel'" class="col-12
↪  mb-4">
  <app-chart-funnel report-card-content style="display: contents"
  ↪  [pathVariables]="pathVariables"></app-chart-funnel>
</app-report-card>
```

In the structure of the report cards, it was also decided that each visualization calls the communication service method with the backend independently to obtain the data that it needs to visualize (in accordance with the single responsibility principle for each component). To call this method, the component must gather all parameters for sending a REST API request according to the user's input.

The architectural solution was to implement all personal request parameters as component inputs using @Input() decorator, while the parameters for global filters would be obtained by each component through a common injectable service. In most cases, personal options will represent a single communication format of *Parent-Child components* or *Siblings components*, for which the @Input() decorator is applicable. The header component `report-filters.component` will update the service's `BehaviorSubject` from the report filters after each user activity, to which every `report-card-content` will be subscribed. This approach will help avoid large networks of @Input()

and @Output() decorators between components and simplify the report component templates because all visualizations require the values of the main filters.

■ **Code listing 7.3** Collecting parameters and getting data from REST API example

```
@Input() grain$: BehaviorSubject<Grain>;

engagementOverTimeStatistics$: Observable<EngagementOverTimeResponse>;

constructor(private reportStatisticsService: ReportStatisticsService)
↪  {
  }

ngOnInit(): void {
    const filtersParams$ =
    ↪  this.reportStatisticsService.filtersParams$;

    this.engagementOverTimeStatistics$ =
    ↪  combineLatest([filtersParams$, this.grain$]).pipe(
      tap(() => this.loading = true),
      switchMap(([filtersParams, grain]) =>
        this.reportStatisticsService.
        getEngagementOverTimeStatistics(this.pathVariables,
        ↪  filtersParams, grain)),
      tap(() => this.loading = false),
    );

    this.engagementOverTimeStatistics$
      .pipe(
        map(data => this.transformEngagementData(data))
      )
      .subscribe(transformedData => {
        this.updateLineChartData(transformedData);
      });
}
```

Thanks to the use of `BehaviorSubjects` as data streams, all report cards are subscribed to events in the interface that the user will interact with. At the moment of changes, all components will immediately start asynchronously sending multiple requests to the backend module to retrieve new data and visualize it right away. This application of RxJs classes corresponds to the use of the reactive programming paradigm described in Section 2.3.

7.3 Angular Material

During the development of the frontend, the UI component library *Angular Material* [15] was actively utilized. The Angular team at Google and the community actively support this library. Its distinctive features include close integration with the architecture of the Angular framework and adherence to Material Design guidelines. Thanks to this library, the Reporting Tool interface is consistent in its design and aligns with the design of other pages on the Experts.ai portal. The pre-built UI components allowed for not having to implement many controls from scratch.

Some complex components of this library used in the Reporting Tool and details of their configuration will be described next.

7.3.1 Datepicker

The `mat-date-range-input` is located in the main filters header as shown in Figure 7.2 and allows for manual selection of the time period for which the report will be generated. In the date picker, dates can be entered both as text input from the keyboard and via a pop-up calendar where the start and end dates can be selected. This component can be used together with the FormGroup directive, collecting both dates into a form and in the component code, subscribing to the `Observable` data stream `FormGroup.valueChanges` to update service filters value on changes.

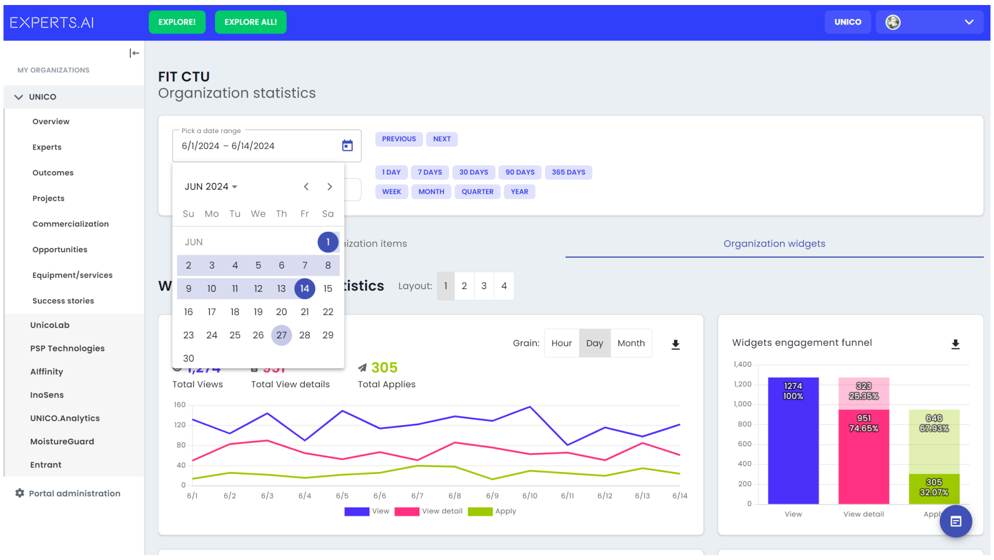


Figure 7.2 Datepicker interface

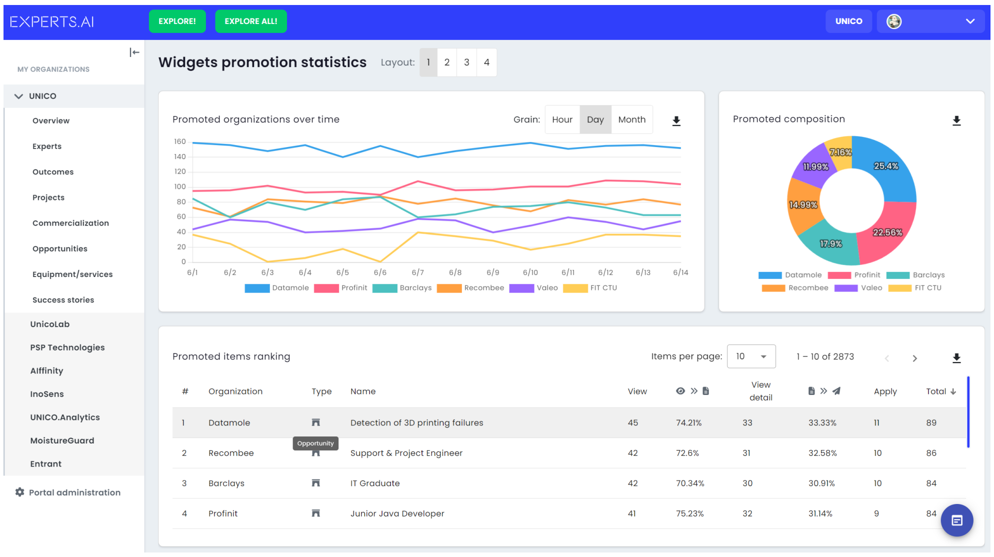
Code listing 7.4 Datepicker configuration with FormGroup

```
<form [formGroup]="filtersFormGroup" class="col-12 col-lg-3">

<mat-form-field appearance="outline" subscriptSizing="dynamic"
↪  class="col-12 mb-2">
  <mat-label class="pe-1">Pick a date range</mat-label>
  <mat-date-range-input [rangePicker]="picker">
    <input matStartDate formControlName="startDate" placeholder="Start
    ↪  date">
    <input matEndDate formControlName="endDate" placeholder="End
    ↪  date">
  </mat-date-range-input>
  <mat-datepicker-toggle matIconSuffix
  ↪  [for]="picker"></mat-datepicker-toggle>
  <mat-date-range-picker #picker></mat-date-range-picker>

  @if
  ↪  (filtersFormGroup.controls.startDate.hasError('matStartDateInvalid'))
  ↪  {
    <mat-error>Invalid start date</mat-error>
  }
  @if
  ↪  (filtersFormGroup.controls.endDate.hasError('matEndDateInvalid'))
  ↪  {
    <mat-error>Invalid end date</mat-error>
  }
</mat-form-field>
...
```

7.3.2 Table

Component `mat-table` was used as the foundation for the "ranking" report cards. Data is provided to the table through the `dataSource` input, and all possible columns are defined in the template, while the active columns to be rendered are supplied through the row definitions in the template using an array of column names. The total number of possible columns in the table may exceed the current number of columns displayed, which was applied to differentiate rankings in the reports (in the "acquisition/promotion" sections, each row indicates the organization, while there is no such column in the "engagement" section). Instead of stretching the table across the entire report page, the tables were designed to be limited in height and scrollable. While scrolling, it was important for the table header to remain visible to the user, which was achieved by adding the property `sticky: true;` to the header row.

For the F5 requirement, each row of the table is clickable and directs to the page of the selected entity. This functionality was implemented using the `routerLink` directive on all rows, as demonstrated in Listing 7.5 and Listing 7.6. The application of this configuration is demonstrated in Figure 7.3 in the ”Promoted items ranking” report card, where hovering over a table row changes its color and clicking redirects to the statistics page for that entity.



Promoted items ranking

Items per page: 10 1 – 10 of 2873

#	Organization	Type	Name	View	👁 » 🗎	View detail	🗎 » ➤	Apply	Total ↓
1	Datamole		Detection of 3D printing failures	45	74.21%	33	33.33%	11	89
2	Recombee	Opportunity	Support & Project Engineer	42	72.6%	31	32.58%	10	86
3	Barclays		IT Graduate	42	70.34%	30	30.91%	10	84
4	Profinit		Junior Java Developer	41	75.23%	32	31.14%	9	84

Figure 7.3 Widgets promotion statistics

Code listing 7.5 Clickable table rows implementation

```
<tr mat-header-row *matHeaderRowDef="displayedColumns; sticky:
↪  true"></tr>
<tr mat-row *matRowDef="let element; columns: displayedColumns;"
  [routerLink]="'itemStatistics' | path: element.type :
  ↪  element.id"></tr>
```

Code listing 7.6 Clickable table rows styling

```
.mat-mdc-row .mat-mdc-cell {
  transition: background-color 0.2s ease;
  cursor: pointer;
}

.mat-mdc-row:hover .mat-mdc-cell {
  background-color: #f0f0f0;
}
```

7.3.3 Sort header and paginator

Each ranking table has its personal options for page navigation and sorting, described in Section 6.3.1. The implementation of this interface utilized the `mat-paginator` component and the following sorting directives applied to column definitions.

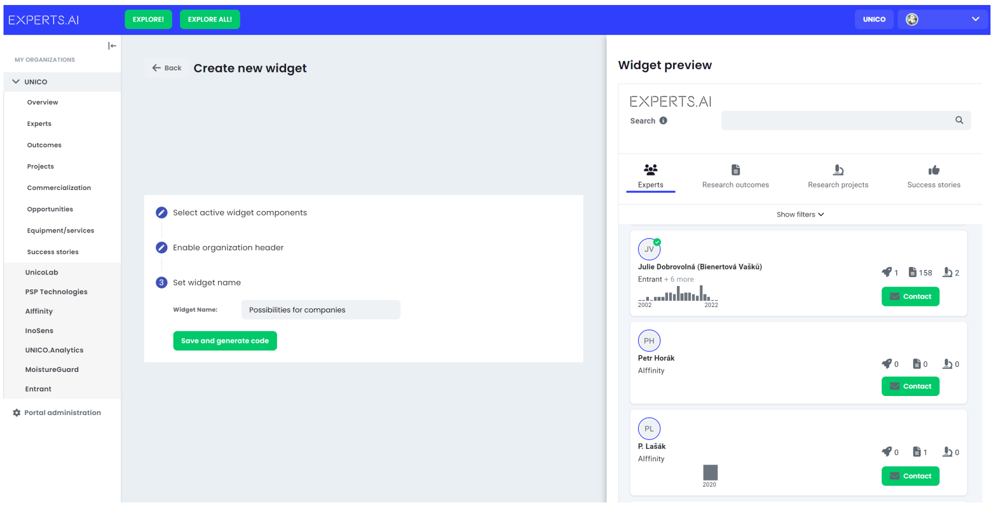
- `matSort matSortDisableClear` are directives for the table element that add sorting behavior and ensure that there is always at least one sorting option selected (one sorting method is always strictly active).
- `matSortActive="total" matSortDirection="desc" matSortStart="desc"` are directives for the entire table, which specify that the default option is ”Total” in descending order, and when clicking on any other column, the first direction will be descending.
- `mat-sort-header disableClear` are added to each column definition that can be manipulated for sorting. These directives are not applied to columns such as “Name”, “Type”, “Organization”.

An important detail was to return the paginator to the first page when the sorting method or global filter is changed, which was also considered in the implementation of the “ranking” report card.

```
this.sort.sortChange.subscribe(() => (this.paginator.pageIndex = 0));
```

7.3.4 Stepper

The multistep interface for creating a new widget instance to match the prototype from Section 6.2.7 utilizes the `mat-stepper` component. In the context of the widget constructor, it is configured to be vertical and linear (steps cannot be skipped until all previous ones are completed), and it also features a convenient template and flexible configuration for each step. This meets the NF3 requirement for application extensibility, anticipating future custom widget options beyond those approved with EDUMATCH during this thesis. An example of the implementation is presented in Figure 7.4.



Widget Name: Possibilities for companies

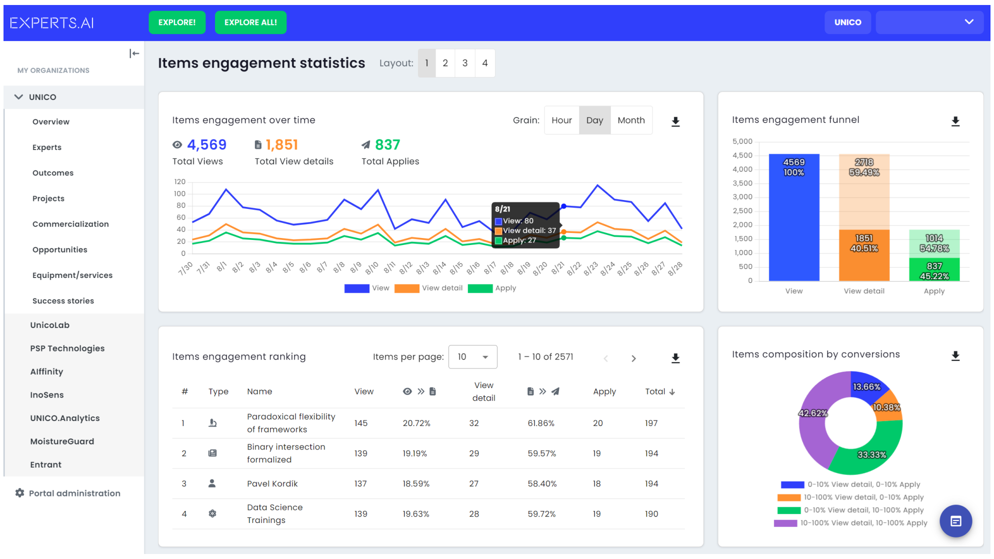
Figure 7.4 Stepper in widget constructor

7.4 Responsive design

The frontend of the Experts.ai platform is built on the *Bootstrap* CSS framework for creating responsive application designs that adapt to any user resolution. [31] It provides a set of pre-designed classes for HTML elements that manipulate their dimensions based on the current viewport size.

The most utilized feature of this framework in the frontend of Experts.ai is the layout toolkit, consisting of a responsive grid system made up of rows and 12 columns. The number of columns an HTML element occupies is determined by its `col` class, and the amount of columns may change depending on the width of the viewport, so that in the case of a small screen, the cells on the page adjust their positions and sizes to remain visible and readable. Another framework utility is easy access to flexbox, padding and margin styles using predefined CSS classes. This feature allows many components not even to have their own defined CSS file, as *Bootstrap* classes cover all necessary styles.

Using this library, the requirement F7 was implemented. A `mat-toggle-button-group` option selector from *Angular Material* was added to each statistics section, which switches the layout of the *Bootstrap* grid elements, their `col` class values and `height` style to change dimensions, and a `hidden` input component to hide or unhide the report card. This functionality is presented in Figure 7.5 and Figure 7.6.

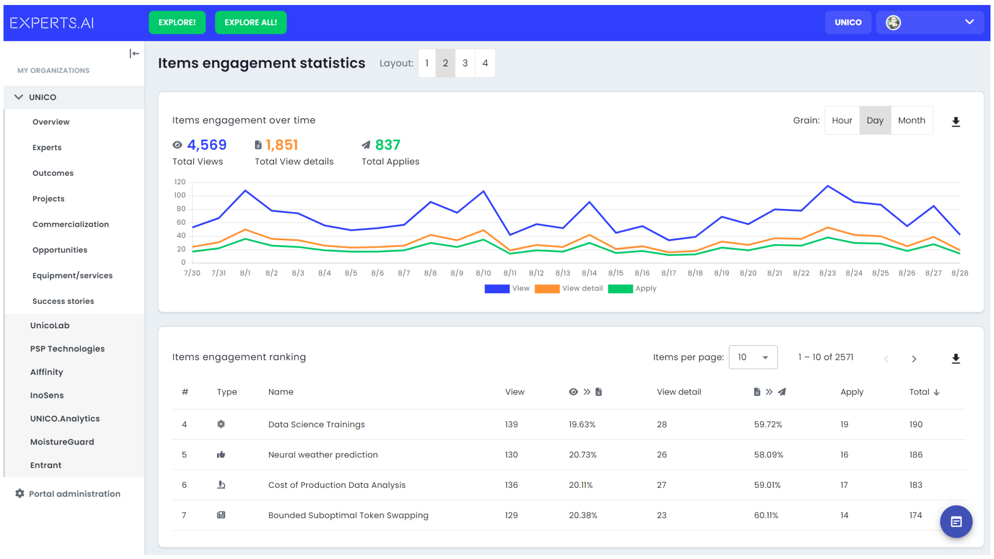


4,569 Total Views · 1,851 Total View details · 837 Total Applies

Items engagement ranking — Items per page: 10 — 1 – 10 of 2571

#	Type	Name	View	»	View detail	»	Apply	Total ↓
1		Paradoxical flexibility of frameworks	145	20.72%	32	61.86%	20	197
2		Binary intersection formalized	139	19.19%	29	59.57%	19	194
3		Pavel Kordík	137	18.59%	27	58.40%	18	194
4		Data Science Trainings	139	19.63%	28	59.72%	19	190

Figure 7.5 Items engagement statistics, first layout



4,569 Total Views · 1,851 Total View details · 837 Total Applies

Items engagement ranking — Items per page: 10 — 1 – 10 of 2571

#	Type	Name	View	»	View detail	»	Apply	Total ↓
4		Data Science Trainings	139	19.63%	28	59.72%	19	190
5		Neural weather prediction	130	20.73%	26	58.09%	16	186
6		Cost of Production Data Analysis	136	20.11%	27	59.01%	17	183
7		Bounded Suboptimal Token Swapping	129	20.38%	23	60.11%	14	174

Figure 7.6 Items engagement statistics, second layout

The *Chart.js* charts and *Angular Material* tables often encountered issues with responsive settings. The content extended beyond the grid system, did not stretch to the full width or height of the parent element, or expanded infinitely beyond the screen. The final responsive configuration, which does not cause issues and conflicts between these libraries and *Bootstrap*, is presented in Listing 7.7.

Code listing 7.7 Responsive charts and tables

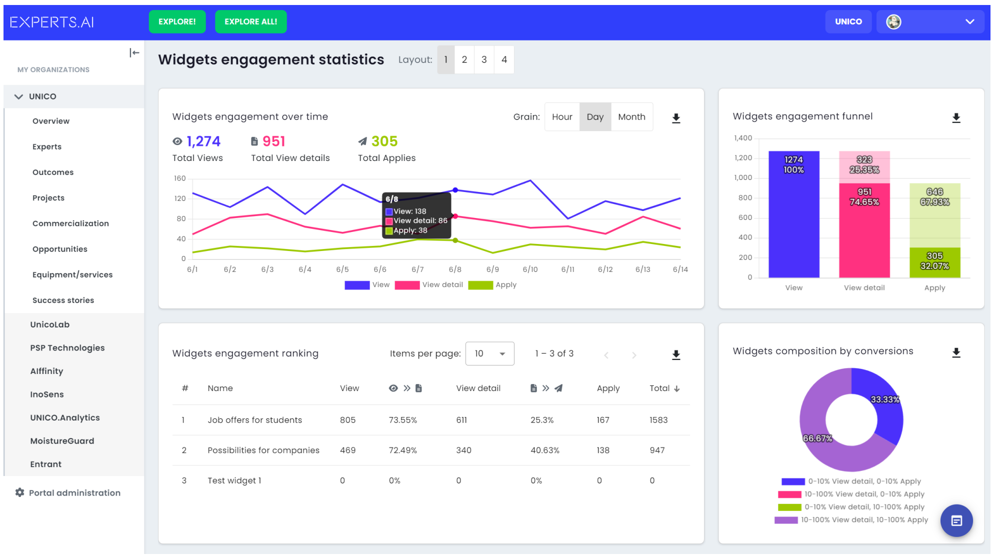
```
<div class="row">
    <div class="col-12 col-lg-8 mb-4">
        <article class="card d-flex flex-column align-items-stretch">
            <div class="flex-grow-1 position-relative">
                <canvas class="chartjs-render-monitor"
                    baseChart
                    [data]="lineChartData"
                    [options]="lineChartOptions"
                [type]="lineChartType">
                </canvas>
            </div>
        </article>
    </div>
    ...
</div>

<div class="row">
    <div class="col-12 col-lg-8 mb-4">
        <article class="card d-flex flex-column align-items-stretch">
            <div class="flex-grow-1 overflow-auto">
                <table mat-table ...
            </div>
        </article>
    </div>
    ...
</div>
```

7.5 Chart.js plugins

As stated in Chapter 4, the Chart.js library was chosen for the implementation of the Reporting Tool to render charts with statistics on the frontend, and one of the reasons for its selection was the active community support with a large ecosystem of plugins and extensions. The plugins used and their applications will be described in the following subsections.

7.5.1 Datalabels

The *chartjs-plugin-datalabels* plugin was one of the most important in the implementation, as it displays labels on chart data without the user needing to hover over them. This was a crucial feature for implementing all funnel and composition report cards, as the percentage values in them are much more important than the visualization itself. Additionally, *Chart.js* by default does not calculate percentage ratios of data values, which was added through this plugin. This functionality will also be necessary for exporting to PDF, as visualizations in PDF format lose their interactivity when the mouse cursor is hovered over them. Therefore, thanks to data labels, the export of such statistics will be clearer and more useful. An example of data labels can be seen on the funnel and composition charts in Figure 7.7.



Widgets engagement ranking

Items per page: 10 1 – 3 of 3

#	Name	View	👁 » 📄	View detail	📄 » ➤	Apply	Total ↓
1	Job offers for students	805	73.55%	611	25.3%	167	1583
2	Possibilities for companies	469	72.49%	340	40.63%	138	947
3	Test widget 1	0	0%	0	0%	0	0

Figure 7.7 Widgets engagement statistics

Code listing 7.8 Funnel datalabels configuration

```
plugins: {
  legend: {
    display: false
  },
  datalabels: {
    display: 'auto',
    textStrokeColor: 'rgba(0,0,0,0.6)',
    textStrokeWidth: 4,
    color: 'white',
    align: 'bottom',
    anchor: 'end',
    textAlign: 'center',
    font: {
      size: 14,
    },
    formatter: (value, context) => {
      if (context.dataIndex === 0) {
        return value + '\n' + `100%`;
      } else {
        const previousValue =
        ↪  context.chart.data.datasets[0].data[context.dataIndex - 1]
        ↪  as number;
        const percentage = Number(((value / previousValue) *
        ↪  100).toFixed(2));
        return value + '\n' + `${percentage}%`;
      }
    },
  }
},
```

Code listing 7.9 Composition datalabels configuration

```
formatter: (value, context) => {
          let sum = 0;
          const dataArr = context.chart.data.datasets[0].data;
          dataArr.map(data => {
            sum += data as number;
          });
          const percentage = Number((value * 100 / sum).toFixed(2));
          return `${percentage}%`;
        },
```

7.5.2 Annotations

The *chartjs-plugin-annotation* plugin allows drawing lines, boxes, labels, points, and other shapes on the chart area. This plugin was used for prime time statistics to display the most active areas on average during the selected time period. The most active intervals longer than the grain of the chart are represented on the plot using box annotations, while the intervals of the same length as the grain are represented using line annotations. An example of the plugin usage can be seen in Figure 7.8 on the "User activity averages for a time period" report card.

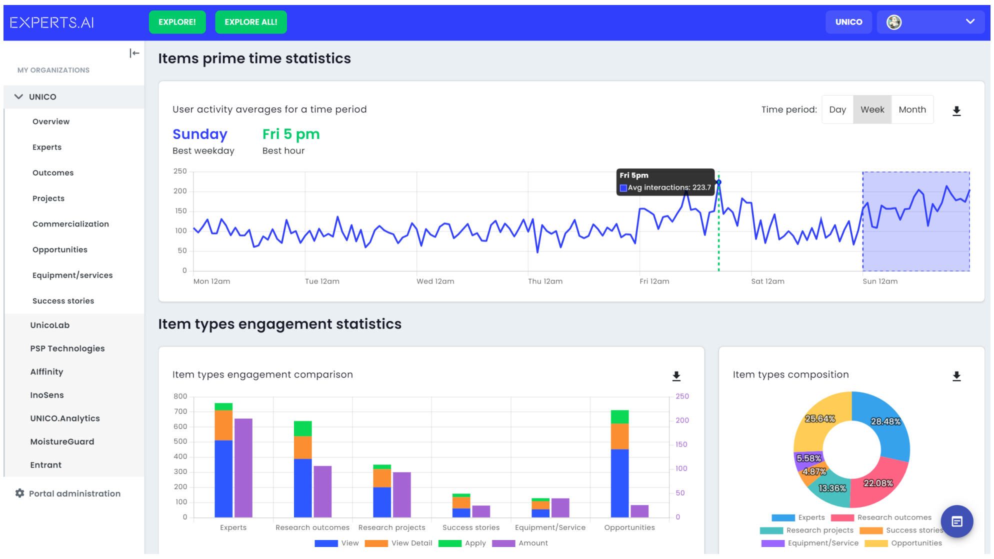


Figure 7.8 Prime time and item types engagement sections

7.5.3 Deferred

Since the statistics pages are presented as long scrollable reports, users often miss all the animations of the charts initialized at the bottom of the screen. The *chartjs-plugin-deferred* plugin was configured so that the animation of each visualization plays only when the user reaches the chart within their viewport. A global setting was implemented on the frontend to ensure that any chart starts playing its information only when half of the chart area is reached with the option `yOffset: '50%'`.

7.6 PDF export

To implement the exporting feature under functional requirement F8, established libraries among developers, *html2canvas* [32] and *jsPDF* [33], were applied. The first library is designed to take a screenshot of a DOM element in any of the available image formats. The `html2canvas` function captures the HTML content and converts it to a canvas. The canvas is then converted into a selected image format. To capture a visualization DOM element, the decorator `@ViewChild` was used, which allows finding an element with a specified reference in the HTML document structure, for example, `#contentWrapper` reference.

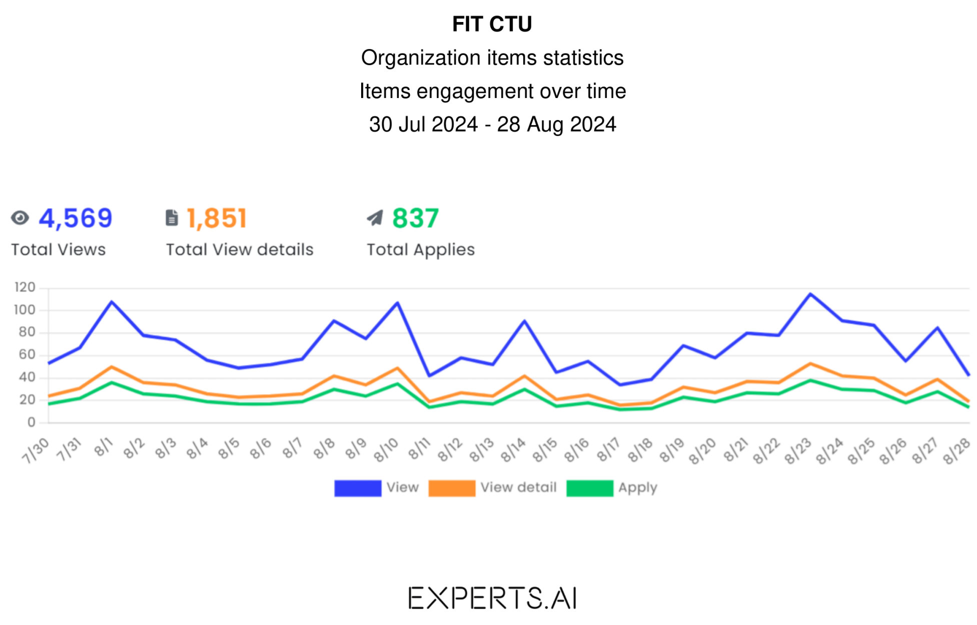
Afterward, the final image file of the visualization can be inserted into a PDF document created using the *jsPDF* library (`pdf.addImage`). In addition, the library was used to create a small template with the context of the report card, including which filter was applied and which entity the visualization was obtained from, so that the final PDF document could be shared with those who did not use the Reporting Tool.

Code listing 7.10 Exporting visualization to PDF

```
export class ReportCardComponent {

@Input() name: string;

@ViewChild('contentWrapper') content: ElementRef;

exportAsPdf(): void {
this.reportStatisticsService.filtersParams$.pipe(take(1)).subscribe(
    filtersParams => {
      const startDateFormatted =
      ↪  moment(filtersParams.startDate).format('DD MMM YYYY');
      const endDateFormatted =
      ↪  moment(filtersParams.endDate).format('DD MMM YYYY');
      const dateRangeText = `${startDateFormatted} -
      ↪  ${endDateFormatted}`;

      const organizationName = filtersParams.name;
      const reportType = filtersParams.reportType;
      const reportCardTitle = this.name;

      html2canvas(this.content.nativeElement).then(canvas => {
        const fileName =
${organizationName}_${reportCardTitle}_${startDateSimple}_${endDateSimple}`
            .replace(/\s+/g, '_')
            .replace(/[^\w\-]+/g, '');

          const contentDataURL = canvas.toDataURL('image/png');
          const pdf = new jsPDF('l', 'mm', 'a4');

          ...

          pdf.save(fileName);
```

The code presented in Listing 7.10 renders a new PDF file on the client side (an example of the file is presented in Figure 7.9), which the user can download and save to their computer. The method's code saves the PDF files with names that allow the exported visualizations to be distinguished from one another. An example of the exported file name looks like this:

```
FIT_CTU_Items_engagement_over_time_07-30-2024_08-28-2024.pdf
```



FIT CTU
Organization items statistics
Items engagement over time
30 Jul 2024 - 28 Aug 2024

4,569 Total Views
1,851 Total View details
837 Total Applies

EXPERTS.AI

Figure 7.9 PDF exported file example

Chapter 8

Testing

This chapter describes the methodologies used to test the implementation of the Reporting Tool's frontend. The testing allows for evaluating the product's readiness and characteristics that define its quality, highlighting deficiencies for correction and suggestions on how to expand the project beyond the established requirements for the final product.

8.1 Static code analysis

Static analysis is the automated scanning and testing of the source code for programmatic and stylistic errors. This check is performed by linters, tools that help developers ensure that their code adheres to a set of coding standards and follows best practices, which improves the readability, maintainability, and overall quality of the code. Linting is performed without executing or running the code, which is why this analysis is called static. [34]

There are many language-specific and general linters, but for the Reporting Tool, *ESLint* [35] was chosen, a linter for JavaScript and TypeScript that is supported by most Integrated Development Environments, including the one used to develop the module's frontend. Thanks to the static detection of issues and warnings in the module's implementation, the resulted code has maintained its consistency across the entire Experts.ai project, does not contain duplicates and formatting errors, and does not accumulate technical debt that could lead to future rework or breakdown of the frontend.

8.2 User acceptance testing

User acceptance testing (next UAT) is the verification that the final application meets all requirements and needs of the client, which occurs as the final stage of software development. During this manual testing, actual end users test the software to determine if it performs as intended in a real-life context, and testers do not encounter difficulties or obstacles when using the product as intended. The main purpose of acceptance testing is to validate a flawless end-to-end user flow. [36]

Before conducting UAT, it is necessary to prepare test scenarios in advance that cover all requirements and use cases of the project. Then, based on the developed scenarios, testers from user groups involved in these scenarios are invited. These participants are provided with a testing environment in which they perform actions according to the user flow outlined in the scenarios. The test organizer monitors how the end users navigate the new functionality and notes any problematic areas where testers encounter issues. At the end of the testing, participants leave their feedback on what they liked and what was lacking in the final solution. [36]

Five potential users of the Reporting Tool participated in this testing. Two users represented a group of organization administrators who upload their offers to the portal, configure widgets, and are interested in understanding how their entities perform and how much new audience their organizations attract. Three users represented a group of portal administrators or developers of the Experts.ai platform, who have access to the "Portal administration" menu and can monitor the status of the entire platform, including the level of activity occurring on it and whether any widgets or items were stopped working.

The testing was conducted in person at the EDUMATCH company office on a separate personal computer with the operating system Windows 10 Education 22H2 and through the browser Google Chrome version 129.0.6668.101. Testing was conducted in a local testing environment to ensure that the testers' actions did not alter important real data from the production server. End users tested the functionality of the new module according to the following test scenarios:

- **Managing widgets by user activity**
  Covered use cases: UC3, UC8, UC9, UC10

1. Open your organization's overview page.
2. Click on the button to go to the organization's statistics.
3. Specify the statistics date range manually from March 5, 2024, to May 5, 2024.
4. Switch to the organization's widgets statistics tab.
5. Determine which item type is the most popular on the organization's widgets.
6. Enable the filter so that the statistics display only for interactions with this item type.

7. Identify which widget of the organization brings the highest conversion to the "Apply" interaction and remember its name.
8. Click on the button to go back to the organization evidence page.
9. Click on the "Widgets" button.
10. Create a new widget with any name that includes only the component with the most popular item type.
11. Edit the widget with the highest conversion of this item type to "Apply", turn off the component responsible for the most popular item type.
12. Copy the HTML code of the new widget with one component of the most popular item type.

- **Data-driven improvement of organization items by statistics**
  Covered use cases: UC1, UC3, UC5, UC7

1. Open your organization's overview page.
2. Click on the button to go to the organization's statistics.
3. Specify the statistics date range for the last 90 days.
4. Find the item in the organization that earned the fewest "Apply" interactions in that time period.
5. Go to the statistics page for that item.
6. Set the "Item promoters over time" chart grain as "Month".
7. Export "Item promoters over time" as PDF file.
8. Click on the button to go to the evidence page for that item.
9. Edit its content to engage better the audience of organizations represented in the exported chart.

- **Removing redundant widgets**
  Covered use cases: UC2, UC3, UC6, UC11

1. Open your organization's overview page.
2. Click on the button to go to the organization's statistics.
3. Switch to the organization's widgets statistics tab.
4. Choose the layout of the report section so that the widgets' ranking is the biggest possible size of the screen.

5. Find the widget with the fewest "View" interactions of all the widgets and remember its name.
6. Click on the button to go back to the organization evidence page.
7. Click on the "Widgets" button.
8. Delete the widget found in the statistics.

- **Viewing platform's conversion changes (only for portal administrators)**
  Covered use cases: UC4, UC5, UC7

1. Open portal administration page.
2. Choose "Platform global statistics" menu.
3. Specify the statistics date range for this month.
4. Export conversion rates from items funnel statistics as CSV file.
5. Move the statistics date range one month back.
6. Export items funnel statistics as CSV file again.

- **Viewing platform's growth evaluation (only for portal administrators)**
  Covered use cases: UC4, UC5, UC6

1. Open portal administration page.
2. Choose "Platform global statistics" menu.
3. Specify the statistics range for this year.
4. Switch to the platform's widgets statistics tab.
5. Choose the layout of the report section so that the widgets' engagement over time chart is the biggest possible size of the screen.
6. Set the chart grain as "Month".

8.2.1 Testing results

During the testing, the Reporting Tool functioned properly and in accordance with the expectations of EDUMATCH. Participants quickly familiarized themselves with the interface and even in complex steps of the scenarios, they instinctively understood what action needed to be taken next. Positive feedback was received from the testers regarding the functionality of the module. It was suggested to add visual icons for selecting the layout of the report cards, and to implement that the filter header remains at the top of the screen while scrolling down the report, so that it does not require returning to the beginning of the report.

In Section 5.2.3 it was stated that use cases cover all functional requirements. The test scenarios cover all established use cases, and based on the fact that end users successfully passed all scenarios, it can be concluded that all planned functionality of the Reporting Tool was implemented and requirements were met.

Conclusion

The main goal of this thesis was to design and implement the Reporting Tool module for the Experts.ai platform that visualizes user activity statistics in widgets. To achieve this goal, it was necessary to accomplish several sub-goals in each part of this thesis.

In the theoretical part, research was conducted on the Experts.ai platform, including an overview of its user interface structure and technological architecture. Then, several of the most popular existing solutions for viewing web traffic statistics were analyzed, highlighting their advantages and key concepts in the field of user activity reports. In the end, after the comparison of web visualization libraries was made, the Chart.js library was chosen, which was later used to implement all the charts on the frontend of the module.

In the practical part, functional and non-functional requirements for the module's frontend were established with the EDUMATCH, which served as the foundation for all subsequent stages of software development. After that, prototypes of the user interface were designed, focusing on smooth and intuitive user experience, which were tested with end users before the beginning of the implementation. The final completed sub-goals were the finished implementation of the reporting tool and an accomplished user acceptance testing, where end users did not encounter difficulties in its usage.

Since all the sub-goals of the thesis were completed, it can be stated that the resulting frontend was successfully developed to an MVP status. The final solution fulfilled all the stated EDUMATCH requirements and offers a robust interface for visualizing user activities in widgets, enhancing the overall functionality of the Experts.ai platform. The application allows Experts.ai clients to understand user engagement with their offers better. It provides valuable insights into user behavior patterns and trends, making it a significant contribution to the platform's collection of tools. Representatives of organizations that upload their offers to the platform but do not have their own widgets can now use the new module to identify which partners bring them more activity and new users. This can lead to new collaborations and growth opportunities for both organizations inside the platform's context. Testing with end-users confirmed the new module's effectiveness and user-friendliness while also highlighting new potential desired features to expand the Reporting Tool.

8.3 Future work

- **Visualizations for A/B Testing** To validate results of A/B testing, two items or two widgets with different configurations (for example, variant A and variant B) are provided to different user groups for interaction. Then, a difference in engagement arises between one variation of the tested object and the other. In A/B testing, analytical tools are used to make data-driven conclusions from visualizations of user activity about which object configuration leads to higher user conversions. The final Reporting Tool is one of those solutions for analyzing A/B test results. Currently, this can be done by opening reports of different variations of a tested object in two separate web pages. In the future, it is planned to extend the reporting tool with functionality that allows comparing statistics directly on one web page with specified A/B test filters, which will also improve the visibility of the difference in user behavior depending on the object variant.
- **Discord bot statistics** One of the platform's new features, currently under development, will allow students to view job opportunities uploaded to the platform through Discord chatbots. Discord bots will perform the same function of displaying and distributing offers in a user-friendly format, just like the widgets, so that the Reporting Tool can be expanded with new statistics pages for the chatbots as well. Organization editors will be able to conduct data-driven analysis on which tools better promote items and generate more interactions: widgets or Discord bots.
- **Data personalization** All collected statistics in the final Reporting Tool Module are anonymized, and no unified information is stored in the database about which user performed the recorded interaction. If a legislative analysis of personal data retention is conducted, and the option to request user permission to unify their session in the widget is introduced, then the module can be expanded with new user data statistics (For example, average session time, average time spent between certain interactions, or statistics on which cities or countries generate the most interactions).
- **Report dashboard builder** The structure of statistics dashboards in the Reporting Tool is currently easily extendable for Experts.ai frontend developers. In the future, the module's functionality can be expanded by allowing platform users to select aggregations independently, configure visualizations, and customize the structure of the report dashboard from cards with visualizations. User-configured dashboards will be stored in the platform's database, but template dashboards, including the current

configuration in the final module of this thesis, will remain available on the platform.

Bibliography

1. BANERJEE, Shubrodeep. *HTML Iframes* [online]. GeeksforGeeks, 2024 [visited on 2024-10-13]. Available from: `https://www.geeksforgeeks.org/html-iframes/`.
2. *Google Charts* [online]. Google, 2024 [visited on 2024-10-13]. Available from: `https://developers.google.com/chart`.
3. *Experts.ai* [online]. Unico, 2024 [visited on 2024-10-13]. Available from: `https://experts.ai/`.
4. *What is three-tier architecture* [online]. IBM Corporation, 2024 [visited on 2024-10-13]. Available from: `https://www.ibm.com/topics/three-tier-architecture`.
5. *About* [online]. The PostgreSQL Global Development Group, 2024 [visited on 2024-10-13]. Available from: `https://www.postgresql.org/about/`.
6. *What is a relational database?* [online]. IBM Corporation, 2024 [visited on 2024-10-13]. Available from: `https://www.ibm.com/topics/relational-databases`.
7. *What is Java?* [online]. IBM Corporation, 2024 [visited on 2024-10-13]. Available from: `https://www.ibm.com/topics/java`.
8. *Introduction to Spring Framework* [online]. Broadcom, 2024 [visited on 2024-10-13]. Available from: `https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overview.html`.
9. *What is a REST API?* [online]. IBM, 2020 [visited on 2024-10-13]. Available from: `https://www.ibm.com/topics/rest-apis`.
10. *What is TypeScript?* [online]. Microsoft, 2024 [visited on 2024-10-13]. Available from: `https://www.typescriptlang.org/`.
11. *Angular Docs* [online]. Google, 2024 [visited on 2024-10-13]. Available from: `https://v17.angular.io/docs`.

12. ŻURAWSKI, Paweł. *Angular MVVM, MVC, CBA – A look at different approaches to application architecture* [online]. Pretius Ltd., 2024 [visited on 2024-10-13]. Available from: `https://www.panaya.com/blog/testing/what-is-uat-testing/`.
13. *SPA vs MPA: Which Web Architecture is Right for You?* [online]. Ramotion, 2024 [visited on 2024-10-13]. Available from: `https://www.ramotion.com/blog/spa-vs-mpa/`.
14. *Google Analytics* [online]. Google, 2024 [visited on 2024-10-13]. Available from: `https://marketingplatform.google.com/about/analytics/`.
15. *Angular Material* [online]. Google LLC, 2024 [visited on 2024-10-13]. Available from: `https://material.angular.io/`.
16. *What is Mixpanel* [online]. Mixpanel, 2024 [visited on 2024-10-13]. Available from: `https://docs.mixpanel.com/docs/what-is-mixpanel`.
17. *Amplitude* [online]. Amplitude, Inc., 2024 [visited on 2024-10-13]. Available from: `https://amplitude.com/`.
18. *ApexCharts* [online]. ApexCharts, 2024 [visited on 2024-10-13]. Available from: `https://apexcharts.com/`.
19. *Apache ECharts* [online]. The Apache Software Foundation, 2024 [visited on 2024-10-13]. Available from: `https://echarts.apache.org/en/index.html/`.
20. *Chart.js* [online]. Chart.js Contributors, 2024 [visited on 2024-10-13]. Available from: `https://www.chartjs.org/`.
21. *Functional and Nonfunctional Requirements: Specification and Types* [online]. AltexSoft, 2023 [visited on 2024-10-13]. Available from: `https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/`.
22. CHERTISHCHEV, Roman. *Vývoj reportovacího nástroje v platformě Experts.ai [Development of Reporting Tool in the Experts.ai Platform]*. 2024. Available also from: `https://dspace.cvut.cz/handle/10467/115867`. Bachelor's thesis. České vysoké učení technické v Praze, Fakulta informačních technologií.
23. *Use-case diagrams* [online]. IBM Corporation, 2023 [visited on 2024-10-13]. Available from: `https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-use-case`.
24. *Domain Modeling: What you need to know before coding* [online]. Thoughtworks, Inc., 2021 [visited on 2024-10-13]. Available from: `https://www.thoughtworks.com/insights/blog/agile-project-management/domain-modeling-what-you-need-to-know-before-coding`.

25. LAMPRECHT, Emil. *The Difference Between UX and UI Design: A Beginner's Guide* [online]. CareerFoundry, 2023 [visited on 2024-10-13]. Available from: `https://careerfoundry.com/en/blog/ux-design/the-difference-between-ux-and-ui-design-a-laymans-guide/`.
26. *An introduction to UI prototyping* [online]. Hotjar Ltd., 2023 [visited on 2024-10-13]. Available from: `https://www.hotjar.com/ui-design/glossary/prototype/`.
27. *Balsamiq* [online]. Balsamiq Studios, LLC, 2024 [visited on 2024-10-13]. Available from: `https://balsamiq.com/`.
28. HELTON, Allen. *What is API mocking?* [online]. Postman, Inc., 2024 [visited on 2024-10-13]. Available from: `https://blog.postman.com/what-is-api-mocking/`.
29. *What is Postman?* [online]. Postman, Inc., 2024 [visited on 2024-10-13]. Available from: `https://www.postman.com/product/what-is-postman/`.
30. *Content projection with ng-content* [online]. Google, 2024 [visited on 2024-10-13]. Available from: `https://v17.angular.io/guide/content-projection`.
31. *Bootstrap* [online]. The Bootstrap Authors, 2024 [visited on 2024-10-13]. Available from: `https://getbootstrap.com/`.
32. *html2canvas* [online]. Niklas von Hertzen, 2024 [visited on 2024-10-13]. Available from: `https://html2canvas.hertzen.com/`.
33. *jsPDF* [online]. npm, Inc., 2024 [visited on 2024-10-13]. Available from: `https://www.npmjs.com/package/jspdf`.
34. *What is Static Analysis & How Does it Work?* [online]. Datadog, 2024 [visited on 2024-10-13]. Available from: `https://www.datadoghq.com/knowledge-center/static-analysis/`.
35. *ESLint* [online]. OpenJS Foundation and ESLint contributors, 2024 [visited on 2024-10-13]. Available from: `https://eslint.org/`.
36. *User Acceptance Testing (UAT) Process Explained* [online]. Panaya, 2018 [visited on 2024-10-13]. Available from: `https://www.panaya.com/blog/testing/what-is-uat-testing/`.

Contents of the attachment

```
readme.txt..........................contents of the attachment description
examples
├── screenshots....................images demonstrating the application
└── wireframes...........................designed application wireframes
src
└── thesis.zip .............. archive with LaTeX source code of the thesis
text
└── thesis.pdf ................................ thesis text in PDF format
```