MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL AVIATION UNIVERSITY

Faculty of cybersecurity and software engineering

Software engineering department

ADMIT TO DEFENSE
Head of Department
_____Oleksiy GORSKI "____"
_____ 2023

QUALIFICATION WORK

(EXPLANATORY NOTE)

GRADUATE OF EDUCATIONAL MASTER'S DEGREE

Theme: "Application for calculating project quality indicators"

Performer: Kolisnyk Ivan Petrovych

Standart controller: c.t.s., Associate Professor, Mykola Fyodorovych Radishevskyi

Supervisor: c.p.-m.s., s.r.f., Mykhailo Viktorovich Olenin

Kyiv 2023

NATIONAL AVIATION UNIVERSITY

Faculty cybersecurity and software engineering
Department Software Engineering
Degree of education master
Speciality 121 Software engineering
Education-professional program Software engineering

APPROVED
Head of Department
_____Oleksiy GORSKI "____"
_____ 2023

Task
on executing the graduation work
Kolisnyk Ivan Petrovych

1. Topic of the graduation work: «Application for calculating project quality indicators»
   Approved by the rector's order from 20.10.2023 № 1967/ст.

2. Terms of work execution: from 05.09.2023 to 31.12.2023.

3. Source data of the work: improved software product using VS Code and programming languages Typescript and Javascript

4. The content of the explanatory note:
   1. Analysis of literary sources and existing analogues
   2. Application design
   3. Implementation of the application
   4. Analysis of project quality and future trends
5. List of mandatory presentation slides:
   1. Topic, performer, leader.
   2. Existing methods, analysis of shortcomings, setting of the task.
   3. Requirements for the software tool.
   4. Tool structure, class diagram.
   5. Software tool interface.

6. Demonstration of the tool prototype.

6. Calendar plan-schedule

№	Task	Deadline	Performance note
1.	Familiarization with the statement of the problem and the study of literature Writing 1 section, presentation to the supervisor	14.10.2023-31.10.2023	
2.	Preprint of section 1 and auxiliary pages (draft) - title, task, schedule, abstract, list of abbreviations, content, introduction, source list. First standard control.	15.10.2023-22.10.2023	
3.	Writing 2 section, presentation to the supervisor	22.10.2023-01.11.2023	
4.	Writing 3 section, presentation to the supervisor	01.11.2023-14.11.2023	
5.	General editing and printing of an explanatory note, graphic material	14.11.2023-20.11.2023	
6.	Passing standard control	20.11.2023-26.11.2023	
7.	Development of the text of the report. Creating of graphic material for presentation	26.11.2023-27.11.2023	
8.	Get feedback from the supervisor, reviews.	27.11.2023-13.12.2023	
9.	Preparation of materials for transmission to the secretary of the DEC (software, GM, CD-R with electronic copies of software, GM, presentations, supervisors review, review, certificate of progress, 2 folders, 2 envelopes)	13.12.2023-19.12.2023	
10.	Graduation project presentation	19.12.2023-31.12.2023	

Date of issue of the assignment 05.09.2023 p.

Supervisor: _____ c.p-m.s., s.r.f., Mykhailo Olenin

Task accepted for execution: _____ Kolisnyk Ivan

РЕФЕРАТ

Дана дипломна робота присвячена розробці та впровадженню застосунку для розрахунку якості проекту. В сучасному світі, де конкуренція та швидкість розвитку бізнесу неспростовно зростають, успішна реалізація проектів стала критичною складовою для багатьох організацій. Якість проекту є одним із ключових факторів, який визначає його успіх та впливає на задоволення клієнтів та стейкхолдерів.

У цій роботі розглядається процес створення та розробки програмного засобу, який надає можливість об'єктивно оцінювати якість проекту на всіх його етапах, враховуючи різноманітні параметри, ресурси, і якісні характеристики. Застосунок використовує методи та інструменти аналізу даних, статистики, та інформаційних технологій для автоматизації процесу оцінки якості проекту.

Дипломна робота включає в себе детальний огляд літератури, аналіз існуючих методів оцінки якості проектів, розробку програмного засобу та його тестування на практиці. Результати дослідження демонструють ефективність застосунку і його потенціал для вдосконалення управління проектами та підвищення якості їх виконання.

Дипломна робота спрямована на вдосконалення процесів управління проектами та може бути корисною як для практикуючих менеджерів проектів, так і для дослідників, що цікавляться питаннями управління проектами та якістю їх реалізації.

Ключові слова: Застосунок для розрахунку якості проекту, процес управління проектами, оцінка якості проекту, програмний засіб, аналіз даних, статистика.

ABSTRACT

This diploma thesis is dedicated to the development and implementation of an application for project quality assessment. In today's world, where competition and the pace of business development are continuously increasing, the successful execution of projects has become a critical component for many organizations. Project quality is one of the key factors that determine its success and influences customer and stakeholder satisfaction.

This work explores the process of creating and developing a software tool that allows for an objective assessment of project quality at all its stages, resources, and quality characteristics. The application uses methods and tools of data analysis, statistics, and information technology to automate the project quality assessment process.

The diploma thesis includes a detailed literature review, an analysis of existing methods for project quality assessment, the development of the software tool, and its practical testing. The research results demonstrate the effectiveness of the application and its potential to enhance project management processes and improve project execution quality.

The diploma thesis aims to improve project management processes and can be valuable for both practicing project managers and researchers interested in project management and its quality assessment.

Keywords: Application for project quality assessment, project management process, project quality assessment, software tool, data analysis, statistics.

Contents

INTRODUCTION

In today's world, characterized by rapid changes and fierce competition, project management is a key success factor for enterprises and organizations in various fields of activity. Projects have become not only an effective tool for the implementation of strategic goals, but also the main way of introducing innovations, adapting to changes in the market environment and maintaining competitiveness.

Successful implementation of projects became a prerequisite for maintaining the stability of enterprises and their further development. In the conditions of the global economy, it is important not only to complete the project on time and within the budget, but also to ensure the high quality of the results, which affects the satisfaction of customers, employees and other stakeholders .

However, the achievement of high project quality and its control require effective methods and tools from organizations. The lack of an objective evaluation and control system can lead to the loss of competitive advantage and unsuccessful implementation of strategic plans.

The purpose of this thesis is the development and implementation of a specialized application for calculating project quality. The study is aimed at solving the problem of evaluating and controlling the quality of projects in all their aspects, starting from the terms of execution and budget, and ending with quality characteristics and satisfaction of all stakeholders.

The object of the study is the project management process and the quality of their implementation in organizations.

The subject of the study is the developed application for calculating the quality of the project, its capabilities and effectiveness in real conditions of use.

This thesis will consist of several sections, including a theoretical basis, an overview of existing methods of assessing the quality of projects, details of the development of a software tool, results of practical tests and conclusions. The work is aimed at solving current project management tasks and aims to improve the quality of their implementation.

CHAPTER 1. ANALYSIS OF LITERARY SOURCES AND EXISTING ANALOGUES

1.1. Review of literary sources and existing analogues

This section provides a detailed analysis of literary sources and existing software tools for project quality management. This review is an important step in understanding the current state of the field of project management and identifying possible solutions and approaches for developing a project quality calculation application. The importance of the review of literary sources in the thesis determines the following key aspects :

Expanding knowledge : A review of literary sources allows you to expand existing knowledge about the chosen topic of research. It helps to get acquainted with the history of research in this field and to find out which aspects have already been studied.

Identification of gaps and needs : Analysis of literary sources allows to identify gaps in scientific knowledge and needs for further research. After conducting the analysis, it is expedient to identify questions that have not yet been answered and that can be the object of research.

Choosing the right approach : Reviewing the literature helps determine which research methods, theoretical approaches, and tools have been used in previous work. This helps to take the right direction for research and approaches that will be most effective in your case.

1.1.1. Analysis of literary sources on project quality management

In the context of the thesis, the analysis of literary sources on project quality management allows to understand the key aspects and methodologies underlying effective control and quality improvement in project activities.

Literature sources such as " *Project Management Body of Knowledge (PMBOK Guide)* ", " *Quality Management for Projects and Programs* " *by Lewis R. Ireland* and " *Project Quality Management: Why, What and How* " *by Kenneth H. Rose* act as key sources for understanding these aspects. The main conclusions from the analysis of these sources are as follows:

1. Relevance of project quality management : Literary sources convincingly emphasize that project quality management is a relevant and important component of modern project management. Quality maintenance affects the success and competitiveness of projects and organizations in general.

2. Basic framework from the PMBOK Guide : The PMBOK Guide defines standard project management processes and provides basic principles of quality management. This framework was created by the Project Management Institute (PMI) and reflects modern approaches to quality management in projects.

3. Practical tips and tools : Literary sources provide practical tips and tools for achieving high quality in projects. These recommendations apply to both general strategies and specific quality management methods.

4. Individual approach : It is important to understand that each project has its own unique characteristics and requirements for quality management. Literary sources emphasize that an individual approach and adjustment of quality management processes are key success factors.

5. Continuous improvement : Project quality management is not a static process, but a continuous cycle of improvement. To achieve high quality, it is necessary to constantly analyze and improve processes.

Based on the analysis of these literary sources, it can be concluded that effective project quality management requires the integration of basic principles, practical recommendations and tools provided by these sources. Next, the design of a project quality calculation application should take these

principles into account and allow for customization according to the unique needs of each project.

1.1.2. Overview of existing software tools for project management

In the context of the diploma thesis, the analysis of existing software tools for project management serves to evaluate the available solutions and identify possible analogues for the development of one's own application for the calculation of project quality.

Fig. 1.1. Microsoft Project

Microsoft Project is a widely used project management software tool known for its ease of use and scalability. It provides a convenient platform for planning, executing and monitoring projects of various sizes and complexities. However, its primary focus is on project management tasks, and tracking quality metrics may require additional customization. While it does provide some customization options, they may not be as powerful as dedicated quality management programs. Real-time monitoring capabilities for quality metrics are limited. However, it is worth considering due to its effective pricing, flexible licensing options, and availability of resources for user support and training.

Advantages of Microsoft Project:

1. Ease of use: Microsoft Project is known for its user-friendliness, which makes it accessible to most users, including those without deep knowledge of project management.
2. Scalability: This software tool can be used to manage projects of various sizes and complexities, making it an excellent choice for a wide range of organizations.
3. Project planning and tracking: Microsoft Project provides powerful tools for planning and tracking projects to help ensure their successful completion.
4. Flexible Licensing Options: The program has a variety of licensing options, allowing organizations to choose the one that best suits their needs and budget.
5. User support and training: Microsoft provides resources for user training and support, which facilitates the implementation of the application in the organization.

Disadvantages of Microsoft Project:

1. Focused on project management: The main focus of this software is on project management, and it may require additional customization to track quality metrics.
2. Limited customization options: Customization options for quality metrics in Microsoft Project may be limited compared to specialized quality management programs.
3. Real-time monitoring: The program is limited in its ability to monitor quality indicators in real-time, which can make it difficult to immediately analyze and respond to changes.
4. Specialization: Deep implementation of quality management may require the use of specialized programs.

Despite these shortcomings, Microsoft Project remains a valuable project management tool with support for planning and tracking, but more detailed quality management may require additional customization or the use of additional tools.

Fig. 1.2. Project management software

Project management software with customizable reporting is designed to streamline project management and provides excellent options for customizing quality metrics. It enables users to define and track project metrics effectively, which is valuable for organizations with unique quality requirements. Systems of this type often provide real-time reporting and messaging, enhancing project quality monitoring. Integration capabilities may vary between products and scalability depends on the specific solution. Support costs and resources can also vary between vendors, so organizations should evaluate their individual needs.

Advantages of Project Management Software with Custom Reporting:

1. Customization of quality metrics: One of the key benefits of this software is the ability to customize quality metrics, allowing organizations to create custom metrics and track their performance.

2. Real-time and notifications: Real-time reporting software provides up-to-the-minute information and alerts that improve project quality monitoring and enable quick response to changes.
3. Flexibility of use: Systems of this type are often very easy to use and can be adapted to the needs of a particular organization.
4. Custom Quality Reporting: The software enables you to create custom quality reports that help you better understand and analyze project quality.

Disadvantages of Project Management Software with Custom Reporting:

1. Variety of Integrations: The integration capabilities of this software may vary between products, which may make it difficult to interact with other systems and tools.
2. Scalability depends on the solution: The complexity of scaling this software can vary depending on the specific solution, and large projects may require detailed scaling planning.
3. Different costs and support resources: The cost and availability of support resources can vary from one vendor to another, requiring additional analysis and selection of the best option for a particular organization.

In general, project management software with the ability to customize reporting provides a wide range of opportunities to control the quality of projects, but requires careful selection and configuration to achieve optimal results.

Fig. 1.3. Tools for creating and analyzing dashboards and analytics

Tools for creating and analyzing dashboards and analytics are featured in customization, offering a high level of flexibility in creating individual reports and visualizing quality indicators. They are known for their strong integration capabilities, allowing for seamless aggregation of data from various sources, including project management tools. Real-time monitoring and instant notifications are considered to be one of their strengths, making them ideal for real-time project quality monitoring. However, cost may vary between tools and compliance and security features may vary, requiring individual evaluation.

Advantages of Dashboard and Analytics Tools:

1. Customization and Customization: Tools for creating and analyzing dashboards and analytics are highly customizable, allowing users to create customized reports and visualizations of quality metrics that meet the unique needs of the organization.
2. The power of integration: These tools allow you to seamlessly aggregate data from various sources, including project management tools, making it easier to gather and analyze information.
3. Real-time monitoring: The ability to provide real-time data allows you to quickly respond to changes and improve the quality of the project during its execution.
4. Data integration power: Dashboard and analytics tools provide powerful data integration capabilities from multiple sources, enabling holistic analysis and decision-making.

Disadvantages of Dashboard and Analytics Tools:

1. Cost Variation : The cost of these tools can vary significantly among different products, which can impact an organization's budget.

2. Compliance and security features : The level of compliance and security features can vary from one tool to another, requiring careful analysis and consideration of the organization's requirements.

Overall, dashboarding and analytics tools are powerful tools for monitoring and analyzing project quality, but organizations should carefully consider their cost and security and compliance capabilities before choosing a specific tool.

Fig. 1.4. Quality management software (QMS)

Quality Management Software offers ease of use for quality management and usually has robust compliance and security features, making it an excellent choice for regulated industries. It provides comprehensive support and training resources for effective use of the program. However, the customization options for project quality measures may be limited and may require adaptation to scale project quality measures. Integration with project management tools may be less versatile than other solutions.

Advantages of Quality Management Software (QMS):

1. Ease of Use: Quality Management Software is noted for its ease of use for quality management, which makes it attractive to a wide range of users.
2. Compliance and Security Features: This software typically has robust compliance and security features, making it an excellent choice for organizations operating in regulated industries where compliance is critical.

3. Support and training: Quality management software usually provides comprehensive support and training resources for users to help them use the software effectively.

Disadvantages of Quality Management Software (QMS):

1. Limited Customization: Customization options for project quality metrics can be limited, which can make it difficult to adapt the program to the specific needs of an organization.
2. Project quality scalability: Scaling project quality metrics may require additional program adaptation.
3. Integration with project management tools: Integration with other project management tools may be less versatile than in other solutions, which may limit data sharing and collaboration with other programs.

Overall, Quality Management Software is a valuable tool for ensuring compliance and security in quality management, but organizations should carefully consider its customization and integration capabilities before choosing this software tool.

Fig. 1.5. Business Intelligence tools

Business Intelligence tools offer extensive customization options for defining and tracking project quality metrics. They are known for their strong integration capabilities that allow seamless aggregation of data from different sources. They can provide real-time monitoring with customization. However, cost may vary among BI tools and compliance and security features may vary, requiring individual evaluation.

Advantages of Business Intelligence (BI) Tools:

1. Extensive customization options: Business Intelligence tools are noted for their extensive customization options for defining and tracking project quality metrics, allowing users to create customized reports and metrics.
2. Integration Capabilities: They are known for their strong integration capabilities that allow seamless aggregation of data from various sources, including project management tools.
3. Real-time monitoring: BI tools can provide real-time monitoring with customization, making them ideal for real-time project quality monitoring.

Disadvantages of Business Intelligence (BI) Tools:

1. Cost: The cost can vary among different Business Intelligence tools and can be significant, depending on the scope of features and the amount of data to be processed.
2. Compliance and Security Features: Compliance and security features may vary among BI tools, requiring additional consideration and evaluation from a security and compliance perspective in a specific organization.

Overall, Business Intelligence Tools are powerful tools for analyzing and monitoring project quality, but organizations should carefully evaluate cost and integration capabilities before choosing the right BI tool.

Table 1. Evaluation summary table

	Settings for quality metrics	Integration possibilities	Monitoring in real time	Convenience for the user	Scalability	Cost	Compliance and security
Microsoft Project	3	3	2	3	4	3	3
Project Management Software with Custom Reporting	4	4	3	4	4	2	3

Dashboard and Analytics Tools	5	2	4	4	5	2	4
Quality Management Software	3	3	2	3	3	1	2
Business Intelligence	4	5	4	4	5	3	4

After a comprehensive analysis of existing analogues, it became obvious that each of these software tools has its own set of shortcomings and limitations. These limitations include varying levels of customization capabilities, integration challenges with existing systems, limited real-time monitoring capabilities, and potential costs.

In light of these limitations, our project aims to solve these problems by developing a new application. Our goal is to create an application that will not only overcome these shortcomings, but also provide a comprehensive solution for calculating project quality indicators. Through customization, continuous integration, real-time monitoring and cost efficiency, our new application will be designed to meet the unique needs and challenges faced by organizations in the field of project quality management.

1.2. Analytics of the region

In the dynamic and ever-changing landscape of project management, achieving successful results has become more than just a task of meeting deadlines and budgets. This involves a comprehensive understanding of the factors that determine quality, aligning project objectives with stakeholder expectations and maintaining consistently high standards. This is where project quality indicators (QPI) play an important role.

Fig. 1.6. Project quality indicators

Project management systems are an important aspect of modern project management practices. They provide a structured framework for evaluating and measuring the quality of project processes, outcomes and overall performance. KPIs are not just metrics, but pointers that guide project managers and teams on the path to excellence, pointing out areas that need attention and improvement.

The purpose of the analysis is a deep study of the world of NAPs, a comprehensive understanding and analysis of their various aspects. We hope to equip project managers, stakeholders, and quality assurance professionals with the knowledge and insights needed to realize the full potential of FP. By uncovering the intricacies of FP analysis, we enable organizations to make informed decisions, minimize risks, and optimize project quality.

In today's highly competitive business environment, the relevance of PES is extremely important. They act as a chain for project success, acting as early warning systems for potential problems and barometers of project status. PNWs play a key role in:

1. Satisfying Stakeholder Expectations: By aligning project outcomes with stakeholder expectations, PAPs promote transparency and build trust.

2. Ensure project success : PMPs act as signposts, helping project teams navigate the complex landscape of project management with precision and confidence.
3. Support these high quality deliverables: Through continuous monitoring and analysis, SMPs contribute to the delivery of high quality project deliverables, reducing defects and rework.

1.2.1. Classification of quality indicators and their compliance with project goals

Process metrics measure the effectiveness and efficiency of project management processes and work processes. They provide information on how effectively project management practices are being performed. Examples include:

Schedule Adherence : Measuring adherence to the project schedule, identifying delays and evaluating their impact on the project schedule.

Budget Compliance : Assessing whether the project meets budget constraints and identifying financial deviations.

Product indicators emphasize the quality and characteristics of the project results. They measure the quality of end products or project outcomes. Examples include:

Defect density : A measure of the number of defects or problems per unit of project output that indicates product quality.

Customer Satisfaction Assessments : Gathering feedback from end users or clients to gauge their satisfaction with project outcomes.

Project management indicators focus on project management performance and strategies. They provide information about the overall status of the project. Examples include:

Risk Management Effectiveness : Assessing the effectiveness of risk identification, mitigation strategies, and risk responses.

Requirement Change Approval Time : A measure of the time it takes to approve and implement change requests, which affects project flexibility.

Each type of PAP plays an important role in monitoring and evaluating the quality of the project:

Process indicators : These provide information about the effectiveness of project management practices. For example, the detection of budget deviations using budget adherence metrics can trigger corrective actions, such as changes in resource allocation or scope of work.

Product metrics : These help ensure that the project's final deliverables meet quality standards. Detecting high defect densities early in a project allows teams to quickly address quality issues, preventing costly rework and improving overall project quality.

Project management metrics : They provide a comprehensive view of the project's status. Effective risk management, as indicated by risk management effectiveness, helps predict and mitigate problems, reducing project bottlenecks and increasing overall quality.

To illustrate the meaning of each type of PAP, consider real-life scenarios:

1. Process indicator : In the construction project, the PPAs that reflect adherence to the schedule showed constant delays in the delivery of materials. Early corrective actions, such as streamlining the supplier selection process, reduced project delays and improved quality.
2. Product Performance : A high density of defects was found in the software development at the early stage of testing. The team quickly conducted code reviews and implemented rigorous testing protocols, resulting in a significant reduction in defects and improved customer satisfaction.

3. Project Management Indicator : In a complex infrastructure project, effective risk management, as indicated by the successful identification and mitigation of potential risks, has ensured that project objectives are met on an ongoing basis. Stakeholders were well informed and satisfied with the progress of the project.

The SMP closely matches the project objectives, including time, cost, scope and quality:

1. Time: Schedule-related SOPs help identify delays early on, ensuring project milestones are completed as planned.
2. Costs: KPIs, such as budget adherence metrics, help control project costs, meet cost targets, and prevent budget overruns.
3. Scope: Product metrics ensure that project deliverables meet predefined scope requirements, reducing scope creep and improving project compliance.
4. Quality: PPAs in all categories contribute to improving project quality by facilitating problem identification, corrective action, and project stakeholder satisfaction.

Beyond improving project productivity, PPMs also serve as important communication tools, promoting stakeholder engagement and satisfaction throughout the project lifecycle.

1.2.2. Measurement and tracking of project quality indicators

Effective measurement of project quality indicators (PQIs) is the basis for ensuring project success and maintaining high quality standards. Various measurement methods and tools are available for project managers and quality control professionals. An understanding of these techniques is essential for accurate analysis of the PAP.

Key performance indicators (KPI) are quantitative metrics that directly reflect the achievement of key goals. They offer a targeted way to measure project quality. For example, a software development project might use KPIs such as defect density or code review effectiveness to measure quality.

Metrics are numerical measurements used to evaluate specific aspects of project quality. Metrics can be quantitative or qualitative and often include data such as the number of defects, customer satisfaction ratings, or project duration.

These tools can automate data collection, analysis, and reporting. Examples include project management software with built-in quality measurement features and specialized quality management software. Choosing Appropriate Measurement Methods Choosing the right measurement methods depends on project characteristics such as size, complexity, and industry standards. Recommendations for selecting appropriate measurement methods include:

1. Understanding project objectives: Align measurement methods with project objectives and quality objectives.
2. Assess data availability: Assess the availability of relevant data for each method.
3. Consider the type of project: Recognize that different types of projects may require different measurement approaches. For example, construction projects may rely more on on-site inspections, while software development projects may use automated code analysis tools.

By carefully analyzing these factors, project managers and quality control professionals can select measurement methods that provide meaningful information about project quality while minimizing unnecessary complexity and effort.

1.2.3. Future trends in the analysis of project quality indicators

With the constant development of technology, the landscape of project quality analysis (QIA) is preparing for significant transformations. Emerging technologies, including artificial intelligence (AI), data analytics, and machine learning, are poised to play a key role in the development of PFS analysis.

AI, thanks to its ability to process huge amounts of data and identify patterns, has enormous potential in the analysis of PFS. Machine learning algorithms can analyze historical project data to predict future quality issues. For example, AI-based models can predict possible volume spills or resource constraints, facilitating proactive quality management.

Data analysis tools are becoming increasingly sophisticated, allowing for deeper insights into project quality. Modern data visualization techniques can reveal hidden trends and relationships between NAPs, which contributes to more informed decision-making. For example, data analytics can identify the root causes of recurring quality issues.

Incorporating advanced analytical tools into the analysis of PFS has the potential to significantly improve accuracy and predictive capabilities. By leveraging big data and AI, organizations can:

1. Predict quality trends: AI-based models can predict future FFP trends, helping project teams prepare for potential quality issues.
2. Detect problems at an early stage: Advanced analytics can identify warning signs of quality deviations, enabling rapid intervention and risk mitigation.
3. Adaptive quality strategies: Data-driven insights can define adaptive quality strategies based on real-time metrics.

In this analysis, we explored the multifaceted world of project quality indicators (PQIs) and their important role in project management and quality control. SOPs serve as compasses that guide project teams toward excellence by providing insights into project quality.

As technology evolves, the incorporation of advanced analytics and emerging technologies such as AI and data analytics promises to revolutionize the analysis of PFS. These advances will improve accuracy, predictive capabilities, and overall project quality.

In addition, the concept of continuous improvement emphasizes the dynamic nature of the analysis of the PES. Organizations that commit to developing their NAP analysis practices over time can reap significant benefits in terms of project success, stakeholder satisfaction, and overall quality.

1.3. Relevance of development in the analysis of project quality indicators

In the ever-changing landscape of project management, ensuring high-quality project outcomes becomes the primary goal. Project Quality Measures (PQMs) play a key role in this endeavor, serving as beacons that guide project managers and teams to excellence. The concept of development is at the center of an effective analysis of the PAP.

The purpose of this project is to investigate the importance of continuous development in improving the effectiveness of the analysis of the NAP. Project quality indicators are not static; they must evolve to stay relevant in a world where project dynamics, methodologies and technologies are constantly changing.

In an era characterized by technological innovation, changing project paradigms, and evolving stakeholder expectations, project managers and quality assurance professionals must remain vigilant. PPEs that were effective yesterday may no longer serve their purpose today. Stagnation can result in a deterioration in project quality, increased risk, and stakeholder dissatisfaction.

Staying relevant means committing to constant learning, adaptation and development. This means adopting new measurement methods, tools and best practices, as well as building a culture of continuous improvement where

feedback is welcomed, innovation is supported, and the FFP is subject to improvement to fit a dynamic project environment.

1.4. Changes in the technologies of project quality indicators

The scope of Project Quality Indicators (PQIs) never remains the same; it is always in a state of constant transformation. This constant variability is due to many factors, including technological changes, development of methodologies and changes in industry standards. In this chapter, we will explore how the NAP is constantly changing and how this has a profound impact on the relevance of development in NAP analysis.

Advances in technology have a significant impact on the way projects are carried out and, therefore, on the way quality is measured. Example:

1. IoT integration: The introduction of the Internet of Things (IoT) into various industries has led to the emergence of FPGAs, which deal with the analysis of data collected by the IoT and provide insights into the actual execution process of the project.
2. Blockchain : In industries such as supply chain management, the use of blockchain technology has led to the creation of PDSs that measure the accuracy and security of data throughout the supply chain.

The development of project management methodologies, such as Agile and Lean, has led to the emergence of new approaches to quality management. Changing methodologies:

1. Agile Metrics: Agile Metrics specific to Agile methodologies have emerged, focusing on aspects such as sprint speed, roll rate, and backlog status.
2. Lean Six Sigma Metrics: In Lean Six Sigma projects, the NAPs are focused on process efficiency, loss reduction, and defect prevention.

Industries are often adapting to changing standards and regulations, resulting in the need to develop SOPs that meet these new requirements. Example:

1. *Compliance with quality standards*: SOPs related to compliance with quality standards such as ISO 9001 are becoming increasingly important .
2. *Sustainability Metrics*: Growing concern about environmental aspects is leading to the emergence of PES, which measure the environmental impact *of projects* .

The dynamics described above emphasize the importance of development in the analysis of the FP. As NAPs evolve to encompass new aspects and criteria, it becomes critical to ensure that the measurement methods, tools and software used for NAP analysis keep pace with these changes.

Development plays a key role in the evolution of the PAP. It covers the creation and improvement of:

1. *New Metrics*: Development of new PPS metrics designed for specific project types and industry requirements provides accurate measurement of the quality of modern projects .
2. *Measurement tools*: Tools that automate data collection, analysis, and reporting are becoming an integral part of working with the increasing complexity of FAP analysis .

Software solutions : The development of specialized software solutions for the analysis of PAP simplifies processes, improves data visualization and supports real-time monitoring.

Development efforts are not only innovation, but also adaptability. The PES must be able to flexibly respond to the unique needs and challenges of each project. Development ensures that FPs remain effective tools for driving projects to success .

1.5. Use of technologies and tools

The modern world is rapidly developing, and technological advances are becoming a major force in improving project quality management. In this chapter , we take a closer look at the impact of technological innovations such as Artificial Intelligence (AI), machine learning , and data analytics . Some of these innovations include exciting capabilities that increase accuracy, efficiency, and predictability in quality management.

1.5.1. Technological innovations and their influence

Artificial Intelligence (AI) is opening up new horizons in the field of analysis of PNPs, providing the opportunity to process large volumes of data and identify complex patterns that previously remained imperceptible to human understanding. The benefits of AI for PAP analysis include:

1. Predictive Analysis : AI-based models predict potential quality issues based on historical data, allowing proactive measures to be taken to prevent them. For example, AI can predict the likelihood of schedule delays or budget overruns by analyzing past project performance.
2. Anomaly Detection: AI systems are effective at detecting anomalies in data streams in real time, making them indispensable for detecting unexpected quality deviations in the early stages of a project.

Technological advances in the field of data analytics provide an opportunity to gain a deeper understanding of the quality of the project and improve the analysis of the PAP:

1. Advanced Visualization: Data analytics tools create sophisticated visual representations that reveal hidden trends, relationships , and anomalies among PFS. These visualizations facilitate informed decision-making by the project team.

2. Root Cause Analysis: Data analytics helps pinpoint the root causes of recurring quality problems, enabling targeted corrective action. For example, it may discover that a certain process systematically leads to defects.

Technology development efforts play a key role in improving the accuracy, efficiency, and predictability of FAP analysis:

1. Improved Accuracy*: Developments in technology are increasing the accuracy of* PNAS analysis by:
    a. Automated Data Collection: Technology automates data collection, which reduces the risk of human error and ensures the accuracy of information.
    b. Advanced Algorithms : Machine learning algorithms can analyze large data sets to identify patterns and anomalies that may go unnoticed by humans, resulting in more accurate PAPs.
2. Increased efficiency: Efficiency is another key aspect in which technology-driven development proves beneficial :
    a. Real - time monitoring: The technology allows for real-time monitoring of FAP, providing immediate alerts on the detection of quality deviations . This efficiency allows project teams to respond quickly.
    b. Data processing: Modern technologies quickly process large volumes of data, which allows for faster analysis and reporting of FP.
3. Predictability: Developments provide the analysis of the NAP with the ability to predict:
    a. Predicting quality trends: Machine learning models predict future Q&A trends, enabling project teams to prepare for potential quality issues.

b. Detection of the first indicators: Technology can detect the first signals of deviations in quality, which allows you to proactively minimize risks before they escalate .

As the importance of PAP analysis continues to grow, specialized tools and software become necessary. This chapter discusses the importance of creating such tools and provides examples of specialized FP analysis software that simplifies data collection, analysis, and reporting. The importance of specialized tools :

1. Automation: Specialized tools automate the collection of PFS data from various sources, reducing the time and effort required to collect information manually.
2. Data integration: These tools facilitate the integration of data from different stages and sources of the project, which allows you to get a comprehensive overview of the quality of the project .
3. Real - time monitoring: Many specialized tools for the analysis of FPGAs offer real-time monitoring capabilities, ensuring that quality deviations are detected immediately .

Examples of specialized security software for the analysis of PAP :

1. Health Management Software: These comprehensive solutions provide a set of tools for HRM analysis, automating data collection, analysis and reporting processes .
2. Dashboard and visualization tools: Specialized data visualization software allows you to create dynamic dashboards that provide an on-the-fly view of PFS information.
3. Personalized reporting solutions: Some tools allow you to customize reports tailored to specific project requirements and stakeholder needs.

The use of specialized software for the analysis of PFS not only simplifies the analytical process, but also helps project teams to make informed decisions quickly and efficiently, which improves the quality management of the project.

1.6. Functional and non-functional requirements

Functional and non-functional requirements are a key element in any project specification, including an application for calculating project quality.

1. Introduction design parameters: The system must allow the user to introduce the main one's project parameters, such as duration, budget, scope works etc.
2. Calculation of the main one's indicators quality: The system must conduct calculations based on entered parameters and determine Indexes project qualities such as level satisfaction client, costs, term Indexes etc.
3. Generation of reports: The system must provide possibility generate reports that contain calculation data and analysis quality of the project.
4. Saving and loading data: The user must have the possibility to save data projects and download them for future editing.
5. Support multi-level of users: The system must support different access levels for users, depending on their roles (administrator, user, guest).

Non-functional requirements:

1. Speed: The system must have high speed and low response time for users.
2. Security: Provide protection from unauthorized access to data and privacy information users.
3. Reliability: The system must be resistant to failures and recover in case crashes
4. Compatibility: Provide compatibility systems with different operating systems and browsers.
5. Interface user: The system must have a convenient and intuitive clear user interface.

Functional requirements determine which functions or capabilities should be provided by the system, while non -functional requirements relate to quality and characteristics of the system, such as speed, security, and reliability. They are both important for successful development and implementation of your calculation application quality projects.

Conclusion

In this section analysis was carried out literary sources and existing ones analogues, aimed at clarifying the state of the investigated problems and detection available solutions and approaches to calculation quality of the project.

The relevance of the work topic is confirmed by significant literary sources and practical applications in the field of project management. Quality measurement and analysis are important aspects of project management, and they are becoming increasingly important in today's environment.

From the analysis of articles and publications, it became clear that there are different methods and approaches to the calculation of project quality, including quantitative and qualitative methods. This indicates the need to develop a comprehensive tool that can take into account various aspects of quality.

Many of the existing counterparts use modern technologies, such as artificial intelligence and data analytics, to improve the efficiency of the calculation of project quality. This indicates the importance of using modern tools in the development of an analytical application.

However, the analysis also revealed that many of the existing solutions have their limitations and drawbacks. This opens up an opportunity for further improvement and development of a more effective tool for calculating project quality.

Therefore, the analysis of literary sources and existing analogues emphasized the relevance and importance of developing an application for calculating project quality, which will take into account various aspects of quality and use modern technologies to achieve more accurate and reliable results in the field of project management.

CHAPTER 2. APPLICATION DESIGN

2.1. Introduction

The second section of the thesis is devoted to the design of the application, intended for calculating the quality of the project. In light of the constant growth in the importance of effective project management and high competition in the development market, the development and implementation of tools aimed at objective and comprehensive assessment of project quality is becoming an extremely urgent task.

The purpose of this application is to provide means for objective measurement of key indicators of the quality of the project, which will ensure efficiency and stability in the process of its implementation. The starting point for designing this application is an in-depth analysis of user needs and requirements to assess the quality of the project from different perspectives.

During this section, important design aspects will be considered, such as choosing an architectural solution, defining functionality and requirements, technical design, architectural design , external and internal quality parameters .

The purpose of this section is to reveal all aspects of designing an application for calculating project quality and to further create a tool that meets modern requirements and contributes to improving the quality of development and project management processes.

2.2. Assessment of External and Internal Quality

This subsection examines in detail two critically important aspects that determine the overall quality of software - External and Internal Quality. The main focus is on the use of international standards, in particular ISO 25010 and

ISO 9126, which provide the most comprehensive and standardized approach to determining the quality of a software product.

Fig. 2.1. ISO 9126

External quality determines how users perceive and interact with the software. This aspect is important because it affects the real impression and satisfaction of using the product. The use of the ISO 25010 standard in this context provides a systematic and structured approach to the assessment of a number of parameters:

1. Suitability: This parameter determines how well the application meets the requirements of the user and the environment of use. If the program meets expectations and demonstrates correct operation, this has a positive effect on the user's overall impression of the product.
2. Accuracy: Determines how accurate the program provides performance results. This is especially important for applications where accuracy is critical, such as financial or scientific applications.

3. Interoperability: This parameter determines how easily the program can interact with other systems and programs. Ensuring a high level of interoperability makes the product more versatile and convenient to use.
4. Security: This parameter determines how protected the program is from unwanted attacks and data leaks. Ensuring a high level of security is critical to maintaining the confidentiality and integrity of information.
5. Functionality Compliance: This parameter determines how well the program functions meet the requirements. Ensuring a high level of functionality compliance allows users to make the most of the product's capabilities.

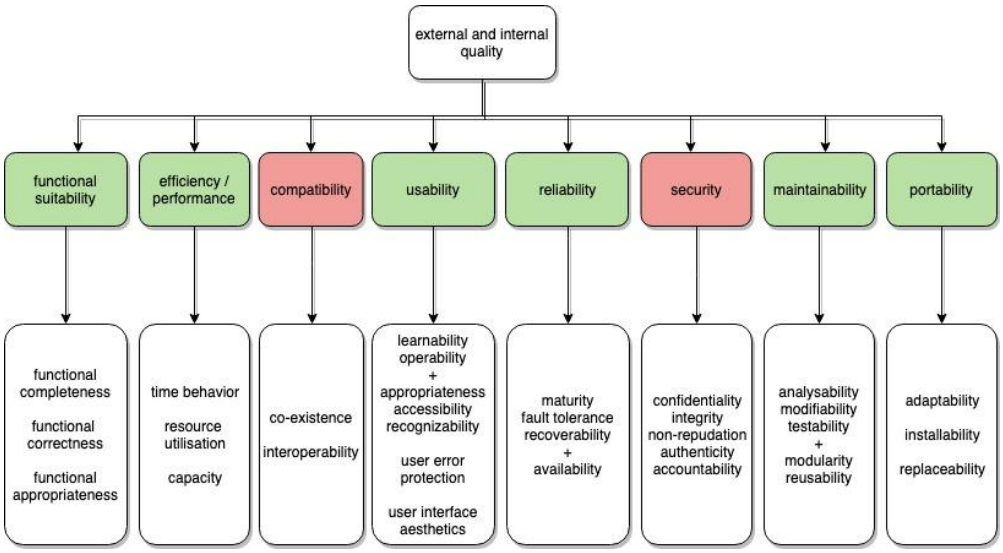


Fig. 2.2. ISO 25010

A detailed examination of these parameters together with the application of the appropriate formulas and metrics of ISO 25010 will provide a comprehensive picture of the external quality of the software product and allow an objective assessment of its effectiveness in performing user tasks.

2.2.1. External Quality

se

External quality is determined by evaluating how users perceive and interact with the software. Standards such as ISO 25010 (software quality

standard) and ISO 9126 (product quality standard) are used to measure this aspect.

External Quality parameters:

1. Suitability: Determines how well the application meets the requirements of the user and the environment of use.

Standard: ISO 25010

Formula:

$$\text{Suitability} = \frac{Number\ of\ correct\ functions}{\text{Total number of functions}} \text{ x } 100\% \qquad (1)$$

2. Accuracy: Evaluates the degree of compliance of the program's performance with the expected results.

Standard: ISO 25010

Formula:

$$\text{Accuracy} = \frac{Number\ of\ tests}{\text{Total number of tests}} \text{ x } 100\% \quad (2)$$

3. Interoperability: Determines the possibility of interaction of the program with other systems and programs.

Standard: ISO 25010

Formula:

$$\text{Interoperability} = \frac{The\ number\ of\ interacting\ functions}{\text{Total number of functions}} \qquad (3)$$

4. Security: Determines the program's level of protection against unwanted attacks and data leaks.

Standard: ISO 25010

Formula:

$$\text{Security} = \frac{Number\ of\ detected\ vulnerabilities}{\text{Total number of tested vulnerabilities}} \qquad (4)$$

5. Functionality Compliance: Determines how well the program functions meet the requirements.

Standard: ISO 25010

Formula:

$$\text{Compliance with functionality} = \frac{Number\ of\ functions\ that\ meet\ the\ requirements}{\text{Total number of functions}} \quad (5)$$

2.2.2. Internal Quality

Internal quality is determined by the technical characteristics of the software and its ability to easily adapt to changes. ISO 25010 and ISO 9126 standards are also used to assess internal quality.

Internal Quality parameters:

1. Maturity: Determines how stable the application is and meets user expectations.

Standard: ISO 25010

Formula:

$$\text{Maturity} = \frac{Number\ of\ justified\ use\ cases}{\text{Number of justified use cases}} \quad (6)$$

Fault Tolerance: Determines the program's fault tolerance and its ability to recover from failures.

Standard: ISO 25010

Formula:

$$\text{Tolerance for mistakes} = \frac{\square\ system\ failure}{\square\ total\ system\ operating\ time} \quad (7)$$

2. Recovery (Recoverability): Determines the speed and efficiency of program recovery after errors occur.

Standard: ISO 25010

Formula:

$$\text{Recovery} = \frac{\square\ system\ recovery}{\square\ total\ system\ downtime} \quad (8)$$

3. Reliability Compliance: Determines how well the program meets reliability requirements.

Standard: ISO 25010

Formula:

$$\text{Correspondence reliability} = \frac{Number\ of\ reliable\ components}{Total\ number\ of\ components} \quad (9)$$

These formulas provide specific numerical scores for each quality parameter that can be used to objectively evaluate the relevant aspects of the software.

2.2.3. Calculation of the overall assessment of the quality of the project

Various indicators reflecting various aspects of its functioning are used to carry out a comprehensive assessment of the quality of the project. One of the approaches to the aggregation of these indicators is the use of the arithmetic mean method.

The overall assessment of the quality of the project (Q) is calculated as the arithmetic mean of individual quality indicators (Indicator $_{and}$):

$$Q = \frac{1}{n}\sum_{i=1}^{n} Index_i \quad (10)$$

where n is the number of quality indicators. Each of these indicators can determine a separate aspect of the project's functioning, such as suitability, accuracy, interoperability, security, compliance with functionality, etc.

This approach allows for a numerical assessment of the overall quality of the project, making it easier to compare different aspects and identify areas for further improvement.

Practical Example: Assessment of External Quality

Let's say we have a software product that has 25 functions, and during testing it turns out the following:

1. The number of correctly performed functions: 20
2. Total number of tests: 30
3. Number of interacting functions: 18
4. Number of detected vulnerabilities: 2
5. Number of eligible functions: 22

Now let's use the previously described formulas to calculate the indicators:

1.Suitability (Suitability):

Eligibility =$\frac{20}{25}$ x $100\% = 80\%$

2.Accuracy (Accuracy):

Accuracy =$\frac{25}{30}$ x $100\% = 83.33\%$

3.Interoperability (Interoperability):

Interoperability =$\frac{18}{25}$ x $100\% = 72\%$

4.Security:

Security =$\frac{2}{5}$ x $100\% = 40\%$

5.Functionality Compliance (Functionality Compliance):

Compliance with functionality =$\frac{22}{25}$ x $100\% = 88\%$

Determination of the overall assessment of the quality of the project (Q):

$$Q = \frac{1}{5}(80\% + 83.33\% + 72\% + 40\% = 88\%)$$

$$Q = \frac{1}{5} \text{ x } (363{,}33\%)$$

$$Q=72.67\%$$

This example demonstrates how to apply formulas to specific numerical values to obtain quality score estimates. This can serve as an illustration of how you would calculate and interpret the results in the context of real software.

2.3. Product quality assessment algorithms

2.3.1. Overview of the main algorithms for quality assessment

Method of Analogies (Analogy-Based Estimation)

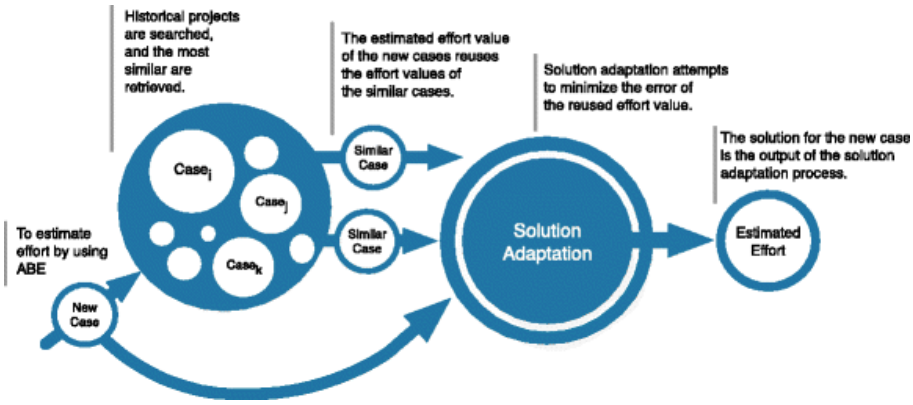


Fig. 2.3. Method of analogies

This method uses historical data and analogies to predict project quality. The idea is to compare the current project with previous similar projects and take their results into account when forecasting costs and quality.

1. Advantages:
    a. High accuracy in the presence of a sufficient number of analogues.
    b. Easy interpretation of results.
2. Disadvantages:
    a. Requires significant amounts of historical data.
    b. Limited in effectiveness in the absence of analogues or a large difference between projects.

Method of Machine Learning (Machine Learning-Based Estimation)

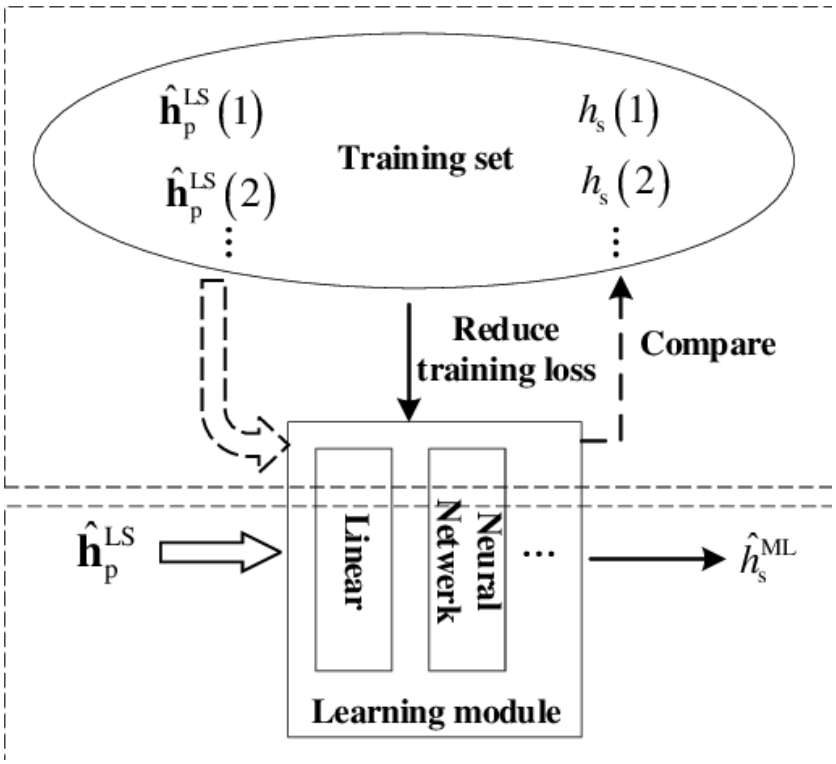


Fig. 2.4. Method of Machine Learning

Applying machine learning algorithms to create models that can predict project quality based on input parameters. Can use different algorithms such as linear regression, decision trees, neural networks and others.

1. Advantages
    a. Adaptability to different types of projects and the ability to take into account complex relationships between indicators.
    b. Ability to take into account heterogeneity and non-linearity of data.
2. Disadvantages:
    a. Requires sufficient training data and processing.
    b. The results are not always interpretable, especially for complex models.

Bayesian Networks

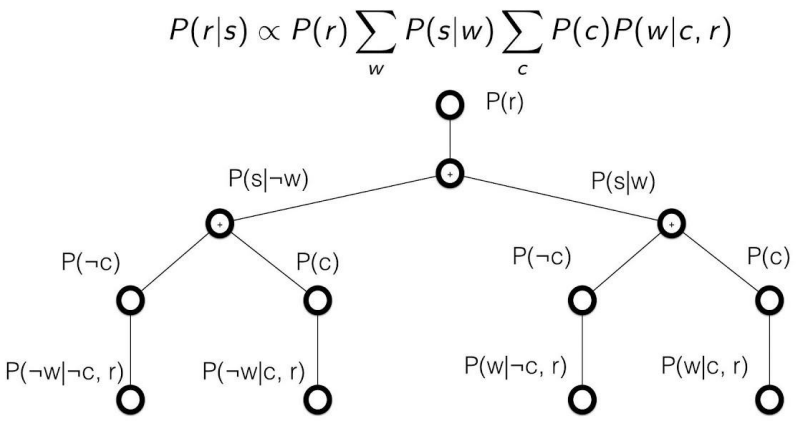


Fig. 2.5. Bayesian Networks

The use of probabilistic models and graphic structures to determine the probabilities of relationships between various project quality parameters. The model is presented in the form of a graph, where nodes are parameters, and edges are probabilities.

1. Advantages:
    a. Ability to consider uncertainty and relationships.
    b. Effective management of model complexity.
2. Disadvantages:
    a. High complexity of modeling and requirements for computing resources.
    b. Requires expert knowledge to configure probabilistic parameters.

Data Analysis Using KPI (Key Performance Indicators)

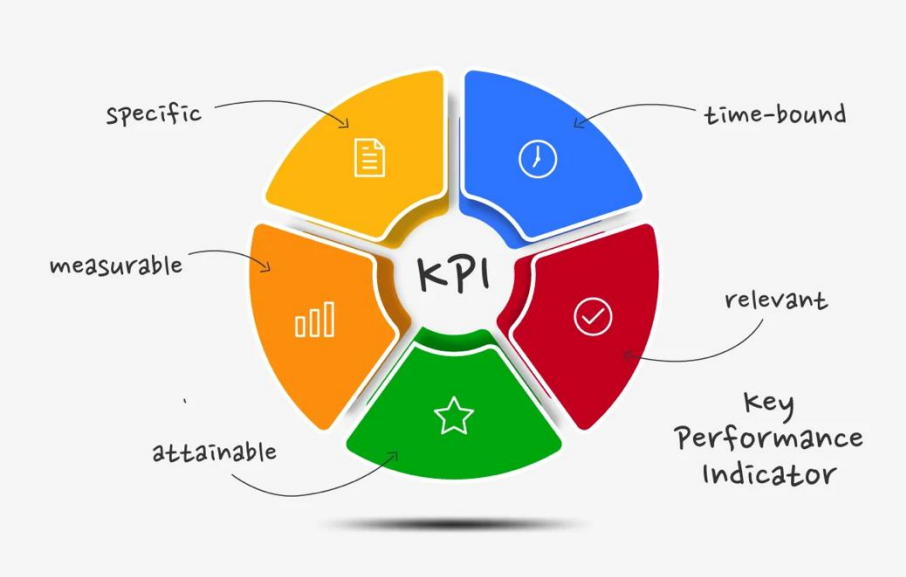


Fig. 2.6. Key Performance Indicators

Determination and analysis of key performance indicators of the project to assess the level of achievement of the set goals and requirements. KPIs can include performance indicators, tasks, deadlines, and others.

1. Advantages:
    a. Ease of implementation and adaptation to various projects.
    b. Allows you to focus on specific key aspects.
2. Disadvantages:
    a. Dependence on the correct definition of KPIs and their adequacy.
    b. Limited in determining relationships between parameters.

The choice of a specific algorithm should be determined by the specifics of a specific project, the availability of data, and the purpose of quality assessment. A combination of different methods often allows to achieve better results.

2.4. Choice of technologies and architecture

2.4.1. Programming language

C# (Development of Server Part and API):

1. Usage: The C# programming language is chosen to develop the backend of the application and create the API.
2. Platform: The use of .NET Core guarantees platform independence and high performance.
3. TypeScript (Development of the Client Part of the User Interface):
4. Usage: TypeScript is used to develop the client side of the user interface.
5. Benefits: Provides strongly typed programming, improves reliability and development productivity.
6. CSS (Web Elements Styling and Display):
7. Usage: The CSS language is used to style and display web elements.
8. Advantages: Provides a convenient and attractive visual part of the application.
9. HTML (Structuring and Display of Web Pages):
10. Usage: The HTML language is used to structure and display web pages.
11. Advantages: Provides logical and hierarchical organization of web application content.

2.4.2. Database

sql.js (Representation of Relational Databases in the Browser):

1. Usage: The sql.js library is used to present relational databases in the browser.
2. Advantages: Provides the ability to perform database queries directly in the browser.

2.4.3. Application driver

Dotnet Core (Blazor) (Server Part of the Application):

1. Usage: Dotnet Core using Blazor is used as the backend of the application.
2. Functionality: Ensures execution of logic on the server and processing of requests from the client side.

2.4.3. Languages Request and Data Description

FlowerBI.Engine is used for automated generation of SQL queries and their execution. Provides a flexible and efficient mechanism for obtaining data from the database on the client side.

This technology allows creating dynamic and high-performance web applications, ensuring efficient interaction between the client, server and database.

2.4.4. Technological innovations and their influence

The project relies on a client-server architecture for effective coordination between critical components.

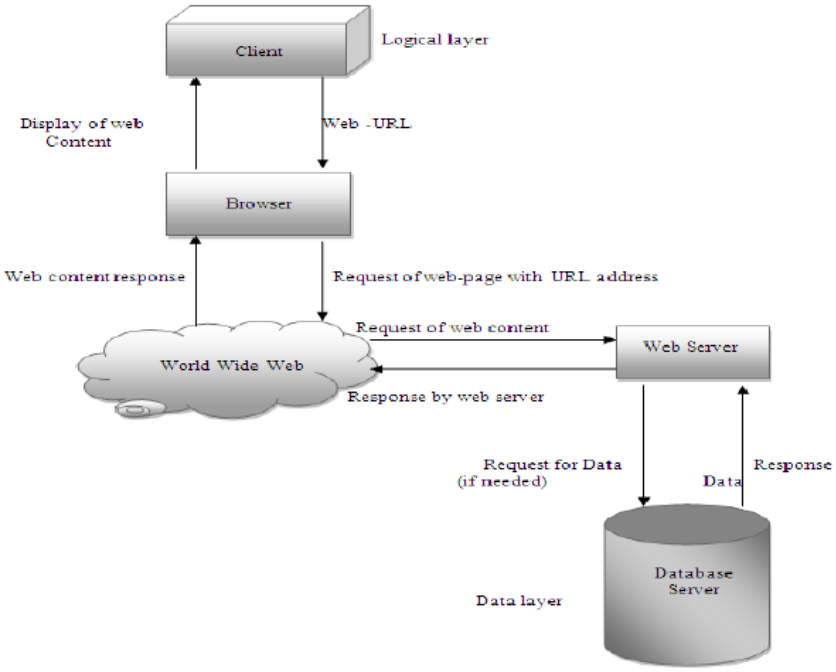


Fig. 2.7. Client-server architecture

The main components of this architecture are as follows:

Client Part (UI)

1. Technologies: TypeScript and React.
2. Functions: Display and interaction with the user. Using the chart.js library for data visualization.

Server Part (Dotnet Core/Blazor)

1. Technologies: .NET Core, Blazor.
2. Functions: Processing of requests from the client side and execution of application logic.

Database

1. Technologies: sql.js.
2. Functions: Data storage and processing in the form of relational databases.

FlowerBI.Engine

1. Technologies: Used for automated generation of SQL queries and their execution.
2. Features: Flexible and secure data request from the client to the database.

The client side interacts with the Blazor server using FlowerBI.Engine to generate and execute SQL queries on sql.js.

Using adaptation techniques to ensure the correctness and compatibility of SQL queries generated for Microsoft SQL Server with sql.js using its own syntax. This approach is necessary to ensure the correct execution of requests in accordance with the requirements set by the platforms used in the project.

This approach allows effective interaction between the client, server and database, ensuring the compatibility of tools and technologies in the project.

2.5. Automation of the analysis process

Automation of the analysis process within the project includes the use of specialized tools and algorithms for effective collection, processing and determination of key indicators of project quality. The main aspects of analyzing automation can be divided into several key components

2.5.1. Data Collection and Processing

Within the project concept, an automated data collection and processing process is a key component to ensure effective project analysis and management. The process is carried out with the help of specialized tools and mechanisms that interact with various sources of information. Let's take a closer look at this process:

1. Data Collection:
    a. Databases: The automated system interacts with various databases such as SQL Server to obtain up-to-date information. Queries are generated by FlowerBI.Engine to retrieve the required data.
    b. Log files: Information can also come from log files that record events and system status.
2. Data processing:
    a. Filtering: The received data goes through a filtering process to select only the necessary information. This may involve selecting specific columns or rows from large datasets.
    b. Aggregation: Some data may be aggregated to create aggregate statistics that facilitate analysis.
    c. Noise removal: To improve the quality of the analysis, noisy information that may arise from errors or anomalous values is processed.
3. FlowerBI.Engine integration:

a. Generating SQL queries: FlowerBI.Engine is used to automatically generate SQL queries, which allows you to efficiently retrieve data from databases according to specified parameters.
b. Execution of SQL queries: Generated SQL queries are executed to obtain relevant information.

4. Ensuring Data Integrity:
    a. Validation and correction: Before processing, data can be validated to detect possible errors. The system can automatically correct some small errors or prompt a message about the need for intervention.
5. Integration with Other Components:
    a. Integration with UI and Other Systems: Processed data is integrated with other system components, such as the client part (UI) or monitoring systems.

The automated process of data collection and processing simplifies interaction with information, providing fast and accurate analysis of large volumes of data for making effective decisions in real time.

2.5.2. Metrics and Key Indicators

Automated analysis determines a number of key metrics and indicators of the quality of the project, which allow you to objectively assess its effectiveness and compliance with the assigned tasks. These metrics include, but are not limited to:

1. Effectiveness of the Code:
    a. Overall productivity: Determined by the speed of execution of basic operations and tasks in the system.
    b. Resource Optimization: Evaluates how efficiently server resources are used when processing requests.

2. Server load:
    a. Number of simultaneous requests: Determines how many requests the server can handle simultaneously without losing performance.
3. Interface Response Time:
    a. Average Response Time: Measures the time it takes to process and display user requests.
    b. Page Load Time: Determines how quickly web pages and the user interface load.
4. Stability and Reliability:
    a. Error Rate: Measures the number of errors or exceptions that occur during system operation.
    b. Disaster Recovery: Determines the time and effectiveness of system recovery after errors or failures occur.

These metrics provide a clear picture of the project in terms of its performance, stability, and responsiveness to user requests, allowing you to quickly identify, analyze, and resolve potential issues.

2.5.3. Weighted Approach and Evaluation Algorithms

The automation of determining the overall quality of the project is based on a weighted approach, which involves the use of evaluation algorithms that take into account weighting factors for various indicators. Each parameter has its own weighting factor, which determines its influence on the final evaluation result.

Assessment algorithms include:

1. Effectiveness of the Code:
    a. Weighted Performance Factor: Determines the importance of the code's performance in the overall score.
2. Server load:

a. Concurrent Requests Weighting Factor: Determines the impact of the number of concurrent requests on the overall score.

3. Interface Response Time:
    a. Response Time Weighting Factor: Considers the importance of user interface response speed.
4. Stability and Reliability:
    a. Error Rate Weighting: Determines the importance of system stability and absence of errors.
5. General Architecture of the Project:
    a. Weight Coefficient of the architectural approach: Takes into account the importance of the selected architectural solution for the project.

This approach allows you to systematize and take into account various aspects of the development, providing an opportunity to accurately determine the quality of the project and identify priority areas for further improvements.

2.5.4. FlowerBI.Engine integration

Analysis automation includes the use of FlowerBI.Engine, which acts as a key component for generating SQL queries and executing them. This engine is implemented to ensure flexibility and efficiency of data retrieval from the database, taking into account various parameters coming from the client part of the system.

Key aspects:

3. SQL query generation: FlowerBI.Engine automatically generates SQL queries based on input parameters received from the client side. It allows you to create dynamic and optimized queries according to specific user requirements.

4. Efficient execution of queries: The engine is optimized for efficient execution of queries on the database. He takes into account the peculiarities of the used database and tries to make maximum use of its capabilities to quickly obtain results.
5. Flexibility in parameter handling: FlowerBI.Engine can handle a variety of query parameters such as filters, aggregations, and sorting. This provides users with the ability to obtain diverse and detailed information from the database.
6. Compatibility with different data sources: FlowerBI.Engine can interact with different data sources, such as SQL Server databases, thanks to its flexibility and adaptability to different data schemas.
7. Use in the client side: The results obtained from FlowerBI.Engine can be easily displayed and used in the client side of the project for further visualization and analysis.

FlowerBI.Engine is woven into the architecture of the project, which allows for automated acquisition and analysis of data using a robust and optimized approach. This creates a platform for effective analysis and use of key project quality metrics and indicators.

Visualization and reporting in the project are provided through the use of the chart.js library, which allows automated creation of visual reports and graphs for convenient display and analysis of analysis result:

1. chart.js library: The project uses the chart.js library to generate various types of graphs and charts, such as line charts, pie charts, bar charts, and others.
2. Automated visualization system: The project's analytical system automatically generates visual reports based on the received analysis results. This allows users to quickly and efficiently perceive a large amount of information by displaying it in a convenient form.

3. Various types of graphs: The chart.js library provides the ability to use various types of graphs depending on the specific needs of the user and the nature of the data.
4. Interactivity: The graphs created can be interactive, allowing users to get more details and interact with the data directly on the graphs.
5. User-friendly interface: The automated system provides a user-friendly interface for interacting with visual reports, making it easier to understand and interpret results.

Visualization and reporting are integrated into the project architecture, enabling the automatic generation of high-quality and informative visual reports to support effective analysis and decision-making.

The automated analysis process includes a monitoring and alerting system aimed at constant tracking and control of the project status. The main aspects of this process are defined as follows:

1. Monitoring system:
    a. Continuous Monitoring: Provides continuous monitoring of key parameters and system elements such as server load, database health, interface response speed and other important metrics.
    b. Regular Updates: Provides regular and automatic updates on the status of the project, allowing you to quickly identify changes and potential problems.
2. Notification:
    a. Critical Anomalies: The system automatically detects critical anomalies or system malfunctions.
    b. Threshold Values: Definition of threshold values for parameters, when exceeding which alerts are generated.
    c. Multiple Channels: Provides the ability to send alerts through multiple channels such as email, instant messaging, etc.
3. Responding to Anomalies:

a. Automatic Response: In some cases, the system can automatically perform certain actions to correct detected anomalies.
b. Decision Support: Provides a user-friendly interface for analyzing alerts and making decisions.

The automated system of monitoring and alerts allows for prompt response to changes and maintaining a high level of project efficiency. This approach helps avoid potential problems, improves system performance, and provides real-time reliability.

2.6. Development of an algorithm for calculating project quality

The process of developing algorithms for evaluating the quality of a project in our application is based on the specific needs and characteristics of the project. Details of this process are provided below:

1. Functionality:
    a. Description: Determined based on project specifications and functional requirements. This includes key aspects of the application, such as data processing capabilities, interactivity of the interface, and the provision of necessary operations.
    b. Metrics:
        i. Functionality coverage.
        ii. Compliance with specifications.
2. Reliability:
    a. Description: The system's resistance to possible errors and its ability to avoid an emergency stop are evaluated. It is important that the system correctly responds to errors and ensures continuous operation.
    b. Metrics:
        i. Number and severity of detected errors.

ii. Stability of system operation during various loads.

3. Efficiency:
    a. Description: Measurement of the speed of operations and resource loading is carried out for efficient use of the system. Ensuring optimal performance plays an important role in determining efficiency.
    b. Metrics:
        i. Speed of execution of basic operations.
        ii. Resource utilization during large computing tasks.
4. Convenience Use:
    a. Description: The intuitiveness of the interface, its comprehensibility and suitability for a wide range of users are analyzed. It is important that users learn and use the system easily.
    b. Metrics:
        i. Training time for new users.
        ii. User satisfaction with the interface.

These criteria determine the main aspects of the quality of the project and serve as the basis for further quality calculation algorithms, allowing to objectively determine and improve the operation of the application in the context of its requirements and use:

1. Functionality:
    a. *Mathematical Model:* $F(Q) = \sum_i wi * M_i$, where F (Q) - functionality, w_i - weighting factor, M_i - metric i.
2. Reliability:
    a. *Mathematical Model:* $R(Q) = 1 - \frac{N_{errors}}{N_{total}}$, where R (Q) - reliability, *Nerrors* - the number of errors, *Ntotal* - the total number of operations.
3. Efficiency:

a. *Mathematical Model:* $E(Q) = \frac{1}{T} * \sum i * \frac{1}{R_i}$, where E (Q) - efficiency, T - execution time, Ri - use of resource i.

4. Convenience Use:
   a. *Mathematical Model:* $U(Q) = \frac{1}{T_{learn}} + \frac{1}{S_{interface}}$, where U (Q) - ease of use, *Tlearn* - learning time, *Sinterface* - user satisfaction with the interface.

These mathematical models formalize the relationship between quality criteria and measurable metrics, which allows for numerical evaluation and comparison of project quality levels based on collected data and observations.

The weighted approach is a strategic method of determining the importance and influence of various criteria on the overall quality of the project. This approach is used to objectively determine how each criterion contributes or can affect the success of the project:

1. Step 1: Definition of Quality Criteria: Each aspect we want to evaluate is defined as a separate quality criterion. In our case, it is functionality, reliability, efficiency and ease of use.
2. Step 2: Formalization of Criteria: Each quality criterion is transformed into a mathematical model with defined metrics and measurement parameters. For example, functionality can be measured by the number of implemented functions, and reliability by the number of detected errors.
3. Step 3: Determination of Weighting Criteria: Each criterion receives its own weighting factor, which indicates how important it is to the overall quality of the project. Weighting factors can be determined based on the priorities and importance of aspects for successful implementation.
4. Step 4: Assign Weights to Criteria: These weights are assigned according to their impact on the overall quality of the project in the context of our application.

5. Step 5: Calculating Overall Quality: After that, each aspect of quality is evaluated using appropriate metrics and weighting factors. The overall quality of the project is calculated as a weighted sum of scores for all criteria.

Application of the Weighted Approach:

1. Provides objectivity in determining the importance of quality aspects.
2. Allows you to take into account different levels of influence on the success of the project.
3. Helps to focus efforts on the main aspects that determine quality.

Weighted Approach for the Project:

1. Functionality: *Weight Coefficient:* $w_{func} = 0{,}4$ || 40%
2. Reliability: *Weight Coefficient:* $w_{reli} = 0{,}3$ || 30%
3. Efficiency: *Weight Coefficient:* $w_{eff} = 0{,}2$ || 20%
4. Convenience Use: *Weight Coefficient:* $w_{usab} = 0{,}1$ || 10%

These weights are determined based on the priorities and importance of each criterion for the successful implementation and use of the project. They reflect the importance of each quality aspect and are taken into account when calculating the overall quality score.

Complex quality analysis algorithm (AlgorithmQA) is a universal tool that allows you to systematically and objectively determine the level of quality of a project, taking into account its various aspects.

Steps of the algorithm:

1. Functionality assessment:
    a. Testing coverage metric (TestCoverage): Calculation of the percentage of code coverage by tests .
    b. Test success metric (TestSuccess): Determination of the percentage of successfully completed tests.
2. Reliability assessment:

a. Error count metric (ErrorCount): Counting the number of detected errors in the system .
b. Stability metric: Analysis of the system's ability to recover from errors.

3. Evaluation of effectiveness:
   a. Performance metrics: Measurement of execution time of key operations.
   b. Metrics in the use of resources (ResourceUsage): Analysis of CPU and memory usage.

Ease of use rating :

1. User Interface (UI) Metrics: Evaluation of the ergonomics, comprehensibility , and usability of the interface.
2. Calculation of total quality (Total Quality) :
   a. Use of weighting factors for each quality criterion.
   b. Total Quality =

$$\frac{w1 * TestCoverage + w2 * TestSuccess + w3 * ErrorCount + w4 * Stability + w5 * Perforamnce + w6 * ResourseUsage + w7 * u}{w1 + w2 + w3 + w4 + w5 + w6 + w7}$$

Each quality criterion is assigned a weighting factor (w), which determines its impact on the overall quality rating.

Weighting factors are chosen based on the importance of each criterion for a specific project. Provides an objective assessment of various aspects of project quality. Considers a variety of criteria covering all key aspects of development. The weighting factors can be adjusted depending on the requirements and features of a specific project. Easily add or modify quality criteria as requirements or development strategy changes.

This algorithm allows you to obtain numerical indicators that summarize various aspects of project quality, helping the development team and management to make informed decisions to improve the product.

The developed algorithms are subjected to extensive testing, covering a variety of usage scenarios, including typical and edge situations.

Test scenarios are aimed at compliance of algorithms with defined quality criteria.

The test results are converted into numerical indicators that reflect the performance of each algorithm according to defined metrics.

The team analyzes the results, checking the compliance of each algorithm with the specified criteria. According to the results of testing and analysis, correction of weighting coefficients is possible to maintain adaptability to specific project conditions. If necessary, algorithms can be modified or improved to achieve better compliance with quality criteria.

Validation and evaluation becomes a cyclical process that allows you to continuously improve algorithms and keep them relevant in a changing environment.

The algorithm is determined not only by its effectiveness, but also by its ability to constantly improve and adapt to new requirements and challenges. Validation and evaluation is an important stage of this process, which ensures not only compliance with defined standards, but also the highest level of quality and user satisfaction.

Identification of optimization opportunities – Analysis of test results and identification of areas where improvements or optimizations are possible is carried out .

Gathering feedback and suggestions – The team analyzes feedback from users, internal suggestions and notes from testers for possible changes or improvements:

1. Prioritization – Each adjustment is evaluated based on its importance and potential impact on algorithm performance and project quality.
2. Demining – Modifications are made to the software code of the algorithms to match identified optimization opportunities.
3. Retesting – Updated code is retested to verify performance and determine whether the goals of the adjustments were met.

Implementation from mines – In case of successful completion of testing and positive results, changes are implemented in the main code of the algorithms.

Post-Adjustment Monitoring – After changes are implemented , monitoring is done to determine the impact of the adjustments on overall quality and performance.

Mine Documentation – Any changes made are documented to ensure clarity and ease of understanding of the development of the algorithms and their history of adjustments.

The possibility of adjustments in the QA algorithm is a necessary component to ensure the relevance and compliance of the project to changing conditions and user requirements.

Conclusion

The chapter carefully considered and developed the key aspects of a system aimed at assessing and monitoring project quality. The main conclusions of this section. The selected technology stack, including TypeScript and React for the client side, .NET Core/Blazor for the server side, sql.js for the database, and FlowerBI.Engine for the automated generation of SQL queries, ensures an efficient and flexible system architecture.

The application uses a client-server architecture to ensure efficient interaction between various components, in particular, client and server parts.
An automated analysis process has been developed and described in detail, including data collection and processing, definition of metrics and key indicators, weighting approach and evaluation algorithms. FlowerBI.Engine is used to automatically generate and execute SQL queries, providing flexible and efficient data request from the client to the database. Using the chart.js library to visualize the analysis results in the form of reports and graphs, which makes it easier for the user to understand the data. The automated analysis process includes a monitoring system that provides regular updates on project status and the ability to automatically alert you of critical anomalies or violations.

Algorithms have been developed that take into account various criteria and metrics for objective assessment of project quality.

All these aspects and components of the system interact to create a tool that provides a convenient and effective mechanism for evaluating the quality of projects in real time.

SECTION 3. IMPLEMENTATION OF THE APPLICATION

3.1. Introduction

In this section, we will dive deeper into the implementation process of our analytical application aimed at evaluating the quality of software code. Let's take a detailed look at the architecture and key components of our application, including modules for syntax analysis, integration with other tools, and visualization of results.

In the following presentation, we will focus on the implementation of algorithms used to evaluate the quality of software code. Let's analyze the code examples for clarity of these algorithms and study their influence on the evaluation of code quality.

The main goal of this section is to create an efficient and powerful tool that will help developers and development teams improve the quality of their software product and reduce the number of errors

3.2. Implementation of the main functional components

In this subsection, we will consider in detail the implementation of the main functional components of the system. Before moving on to specific modules, let's review the general structure of the project.

The structure of the FlowerBI project is as follows:

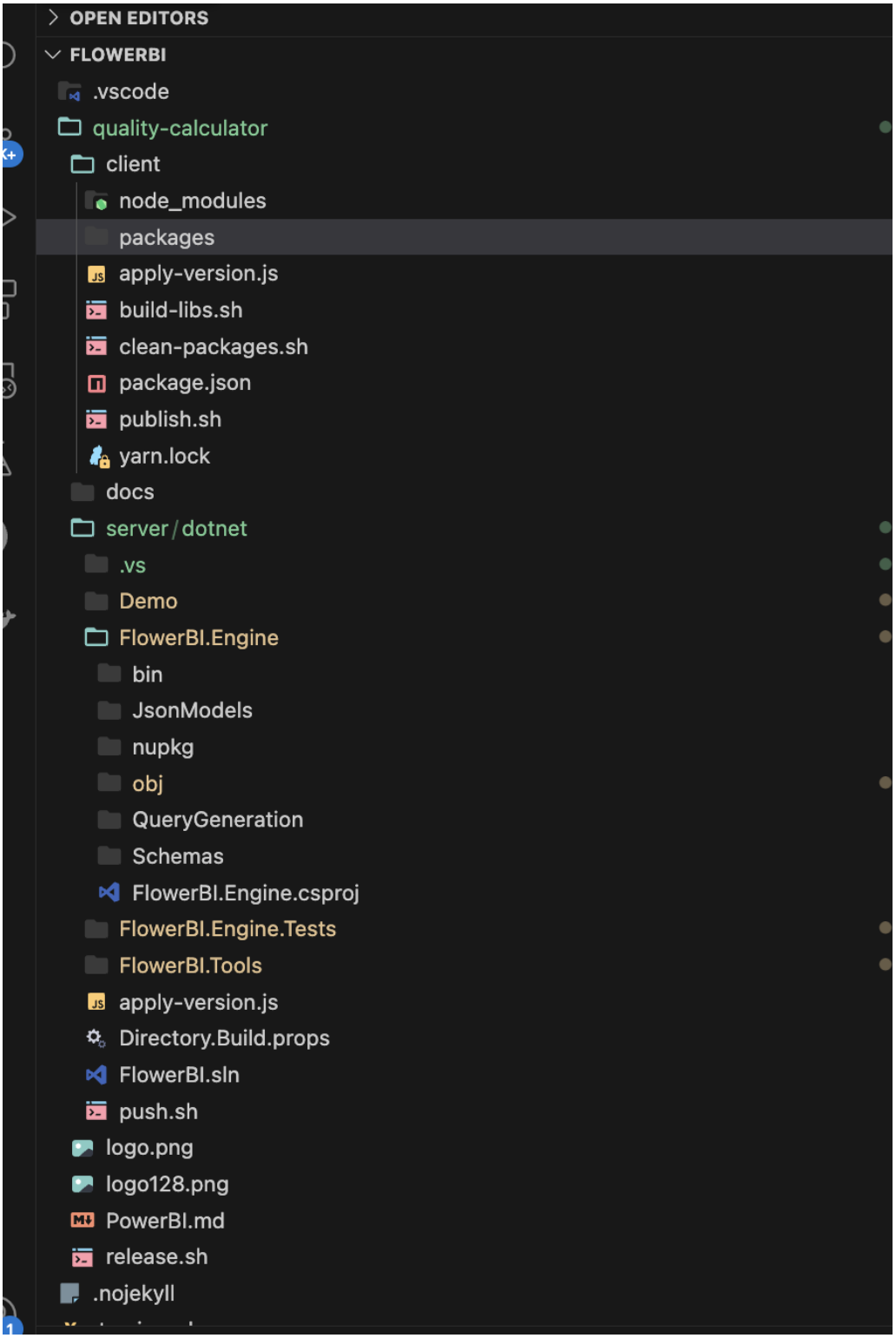


Fig. 3.1. File structure of the project

Packages – This folder contains packages used in the client side of the project. External libraries and tools are organized in this directory.

apply-version.js – Script for applying the project version. Used for automated version control.

build-libs.sh – Script for building libraries. Responsible for assembling the necessary libraries in the client part.

clean-packages.sh – Script for cleaning packages. Ensures cleanup of redundant artifacts and unused packages.

package.json – Configuration file for managing packages and setting dependencies in the client side.

publish.sh – Script for publishing the project. Used to publish the client side.

yarn.lock – A file that contains information about the versions and dependencies of the packages used in the project.

Demo – This folder contains the demo files and examples used for the demonstration .

FlowerBI.Engine.Tests – Here are the tests for the engine of the FlowerBI system. Tests help verify engine functionality and prevent errors from occurring.

FlowerBI.Engine – A module that implements the main functionality and business logic of the FlowerBI system on the server side.

FlowerBI.Tools – A tools module that contains additional tools for development, testing and administration.

Directory.Build.props – A project configuration file that contains build and dependency settings for all projects in the solution.

FlowerBI.sln – A solution file that combines all projects into a single project for easy management and development.

apply-version.js – Script for applying the version of the project on the server.

3.2.1 The main modules of the project

App.tsx

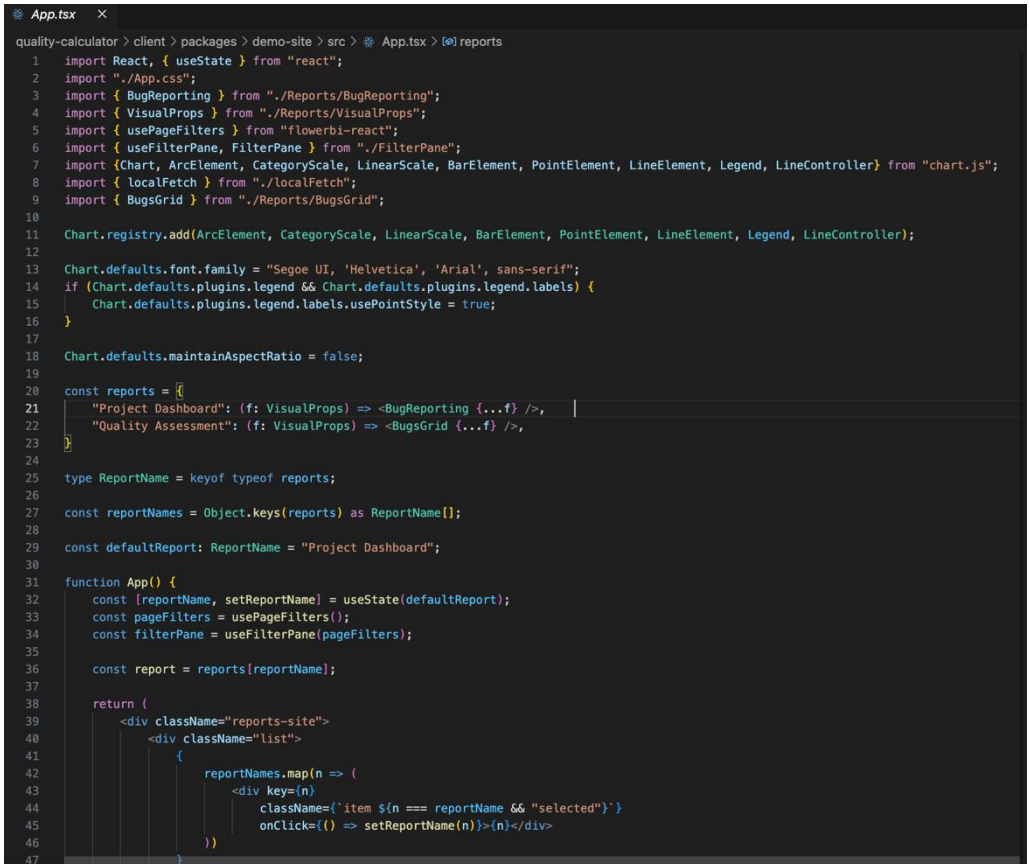


Fig. 3.2. Code from the App file.tsx

The App.tsx report file is the main component of the FlowerBI application that implements the main user interface. Let's take a look at the main parts of this component:

```
import React, { useState } from "react";
import "./App.css";
import { BugReporting } from "./Reports/BugReporting";
import { VisualProps } from "./Reports/VisualProps";
import { usePageFilters } from "flowerbi-react";
import { useFilterPane, FilterPane } from "./FilterPane";
import { Chart, ArcElement, CategoryScale, LinearScale, BarElement, PointE
import { localFetch } from "./localFetch";
import { BugsGrid } from "./Reports/BugsGrid";
```

Fig. 3.3. Main imports

This code snippet imports the necessary dependencies and components, such as the Chart.js library, report components, filters, and more.

```
Chart.registry.add(ArcElement, CategoryScale, LinearScale, BarElement, Poir

Chart.defaults.font.family = "Segoe UI, 'Helvetica', 'Arial', sans-serif";
if (Chart.defaults.plugins.legend && Chart.defaults.plugins.legend.labels)
    Chart.defaults.plugins.legend.labels.usePointStyle = true;
}

Chart.defaults.maintainAspectRatio = false;
```

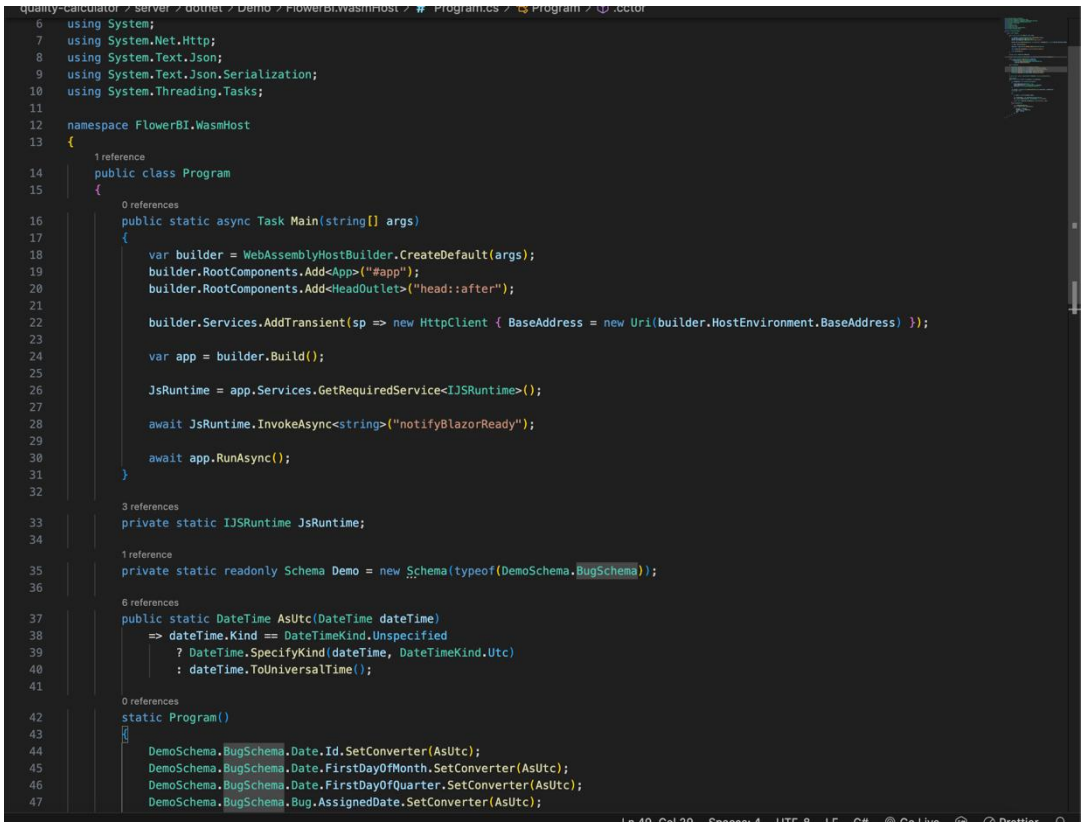
Fig. 3.4. Registration of Chart elements . js

This block of code registers various Chart.js elements, configures the font and some styles for plotting the graphs.

App component is a functional component that uses the useState hook to manage the currently selected report. The component renders a list of reports, the main content with the selected report and filters.

The entire component is wrapped in a div with the "reports-site" class, which sets general styles for the application.

Program.cs



```
using System;
using System.Net.Http;
using System.Text.Json;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace FlowerBI.WasmHost
{
    1 reference
    public class Program
    {
        0 references
        public static async Task Main(string[] args)
        {
            var builder = WebAssemblyHostBuilder.CreateDefault(args);
            builder.RootComponents.Add<App>("#app");
            builder.RootComponents.Add<HeadOutlet>("head::after");

            builder.Services.AddTransient(sp => new HttpClient { BaseAddress = new Uri(builder.HostEnvironment.BaseAddress) });

            var app = builder.Build();

            JsRuntime = app.Services.GetRequiredService<IJSRuntime>();

            await JsRuntime.InvokeAsync<string>("notifyBlazorReady");

            await app.RunAsync();
        }

        3 references
        private static IJSRuntime JsRuntime;

        1 reference
        private static readonly Schema Demo = new Schema(typeof(DemoSchema.BugSchema));

        6 references
        public static DateTime AsUtc(DateTime dateTime)
            => dateTime.Kind == DateTimeKind.Unspecified
                ? DateTime.SpecifyKind(dateTime, DateTimeKind.Utc)
                : dateTime.ToUniversalTime();

        0 references
        static Program()
        {
            DemoSchema.BugSchema.Date.Id.SetConverter(AsUtc);
            DemoSchema.BugSchema.Date.FirstDayOfMonth.SetConverter(AsUtc);
            DemoSchema.BugSchema.Date.FirstDayOfQuarter.SetConverter(AsUtc);
            DemoSchema.BugSchema.Bug.AssignedDate.SetConverter(AsUtc);
```

Fig. 3.5. Program . cs

The Program.cs file is located in the server folder and is the entry point for starting the server part of the WebAssembly application. Let's review the main elements of this file:

```
using FlowerBI.Engine.JsonModels;
using Microsoft.AspNetCore.Components.Web;
using Microsoft.AspNetCore.Components.WebAssembly.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.JSInterop;
using System;
using System.Net.Http;
using System.Text.Json;
using System.Text.Json.Serialization;
using System.Threading.Tasks;
```

Fig. 3.6. Basic imports for a file

This section lists the required namespaces and imported classes for working with .NET and ASP.NET Core.

```
public class Program
{
    public static async Task Main(string[] args)
    {
        var builder = WebAssemblyHostBuilder.CreateDefault(args);
        builder.RootComponents.Add<App>("#app");
        builder.RootComponents.Add<HeadOutlet>("head::after");

        builder.Services.AddTransient(sp => new HttpClient { BaseAddress = 

        var app = builder.Build();

        JsRuntime = app.Services.GetRequiredService<IJSRuntime>();

        await JsRuntime.InvokeAsync<string>("notifyBlazorReady");

        await app.RunAsync();
    }

    // Решта коду...
}
```

Fig. 3.7. Program class

This Main method is the entry point for executing the application. Some key stages are:

1. Defined by WebAssemblyHostBuilder .
2. Added root components to be displayed in index.html .

3. Configurable HttpClient .
4. The application starts.

```
private static IJSRuntime JsRuntime;

private static readonly Schema Demo = new Schema(typeof(DemoSchema.BugSch

public static DateTime AsUtc(DateTime dateTime)
    => dateTime.Kind == DateTimeKind.Unspecified
        ? DateTime.SpecifyKind(dateTime, DateTimeKind.Utc)
        : dateTime.ToUniversalTime();

static Program()
{
    // Ініціалізація об'єкту схеми та конвертерів дати
}

private static readonly ISqlFormatter Formatter = new SqlLiteFormatter();

[JSInvokable]
public async static Task<string> Query(string queryJson)
{
    // Метод для виконання запитів до бази даних з JavaScript
}
```

Fig. 3.8. Fields and class methods

5. This code snippet defines various fields and methods:
6. JsRuntime : Static field to access IJSRuntime through all static methods.
7. Demo : A schema used for data parsing.
8. AsUtc : method to convert dates to UTC format.
9. Formatter : an object for formatting SQL queries.
10. Query : A method that is called from JavaScript and executes an SQL query on the server.

Database.ts

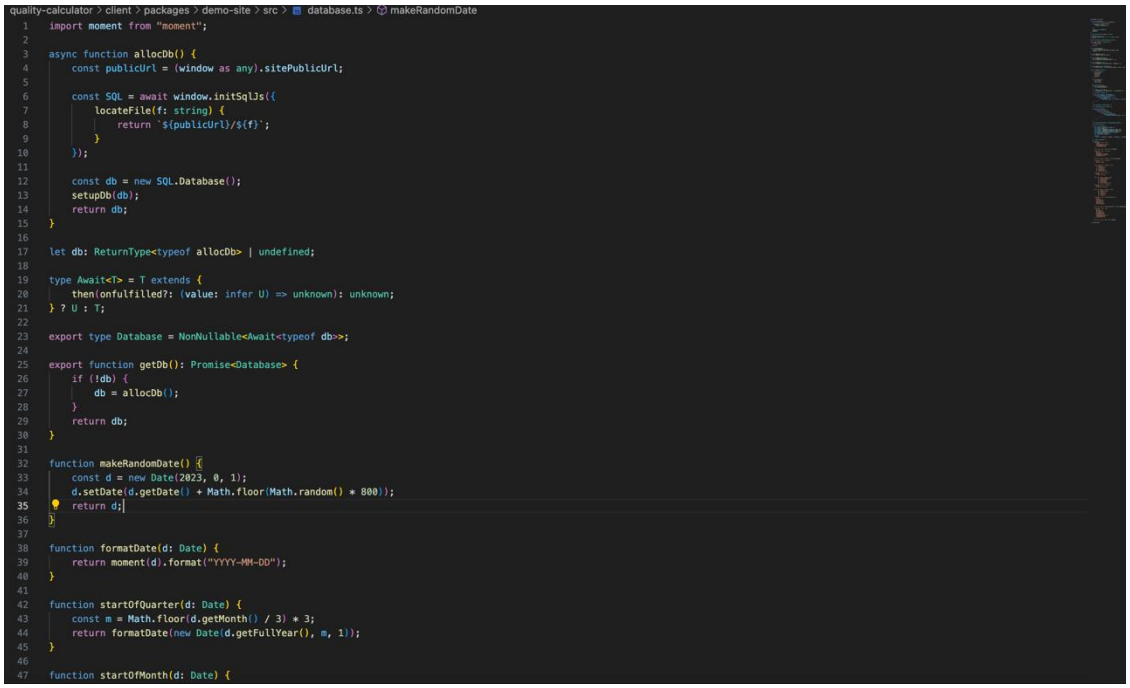


Fig. 3.9. Database

file is responsible for initializing and configuring the database for the client side of the application. Let's look at the main elements of this file:

```
async function allocDb() {
    // ... код ...
    const db = new SQL.Database();
    setupDb(db);
    return db;
}
```

Fig. 3.10. allocDb function

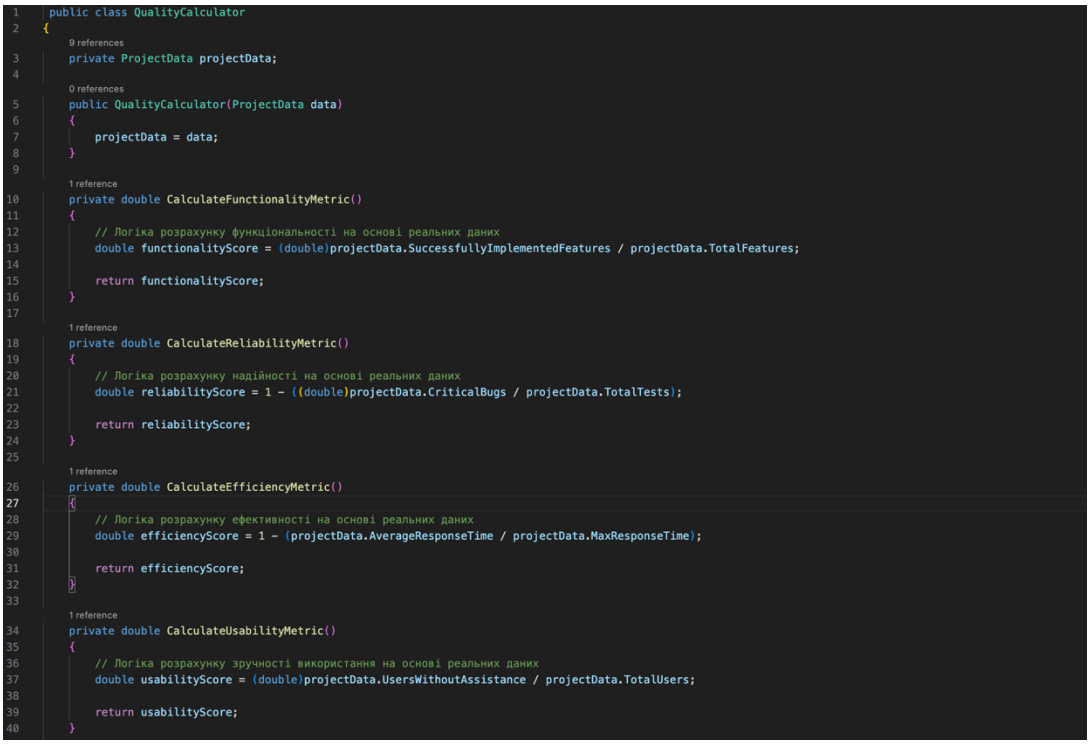
An asynchronous function that initializes the database and calls setupDb to set up the underlying data.

```typescript
let db: ReturnType<typeof allocDb> | undefined;

export type Database = NonNullable<Await<typeof db>>;

export function getDb(): Promise<Database> {
    if (!db) {
        db = allocDb();
    }
    return db;
}
```

Fig. 3.11. Field and function for accessing the database

A variable that represents the database. The lookup database type is defined using ReturnType. An asynchronous function to get a database that is initialized if it has not yet been created.

3.2.2 Implementation of the project quality calculation algorithm



```
public class QualityCalculator
{
    9 references
    private ProjectData projectData;

    0 references
    public QualityCalculator(ProjectData data)
    {
        projectData = data;
    }

    1 reference
    private double CalculateFunctionalityMetric()
    {
        // Логіка розрахунку функціональності на основі реальних даних
        double functionalityScore = (double)projectData.SuccessfullyImplementedFeatures / projectData.TotalFeatures;

        return functionalityScore;
    }

    1 reference
    private double CalculateReliabilityMetric()
    {
        // Логіка розрахунку надійності на основі реальних даних
        double reliabilityScore = 1 - ((double)projectData.CriticalBugs / projectData.TotalTests);

        return reliabilityScore;
    }

    1 reference
    private double CalculateEfficiencyMetric()
    {
        // Логіка розрахунку ефективності на основі реальних даних
        double efficiencyScore = 1 - (projectData.AverageResponseTime / projectData.MaxResponseTime);

        return efficiencyScore;
    }

    1 reference
    private double CalculateUsabilityMetric()
    {
        // Логіка розрахунку зручності використання на основі реальних даних
        double usabilityScore = (double)projectData.UsersWithoutAssistance / projectData.TotalUsers;

        return usabilityScore;
    }
```

Fig. 3.12. Implementation of the algorithm

The constructor accepts an object of the ProjectData class , which represents project data.

```
public QualityCalculator(ProjectData data)
{
    projectData = data;
}
```

Fig. 3.13. QualityCalculator constructor

The CalculateFunctionalityMetric , CalculateReliabilityMetric , CalculateEfficiencyMetric , CalculateUsabilityMetric methods define metrics for functionality, reliability, efficiency, and usability, respectively.

```
private double CalculateFunctionalityMetric()
{
    double functionalityScore = (double)projectData.SuccessfullyImplemen
    return functionalityScore;
}
```

Fig. 3.14. Example of method implementation

The CalculateOverallQuality method uses metrics and their weights to calculate the overall quality of a project.

```
public double CalculateOverallQuality()
{
    double functionalityWeight = 0.4;
    double reliabilityWeight = 0.2;
    double efficiencyWeight = 0.2;
    double usabilityWeight = 0.2;

    double overallQuality =
        functionalityWeight * CalculateFunctionalityMetric() +
        reliabilityWeight * CalculateReliabilityMetric() +
        efficiencyWeight * CalculateEfficiencyMetric() +
        usabilityWeight * CalculateUsabilityMetric();

    return overallQuality;
}
```

Fig. 3.15. Calculation of the total quality

The class contains properties that represent project data, such as the number of features, successfully implemented features, number of tests, number

of critical errors, and others. A constructor sets the values of class properties using parameters.

3.3 Implementation of the architecture

1. Client part (client):
    a. Project structure:
2. src:
    a. Reports: Contains components responsible for displaying analysis results.
    b. App.css : Styles for the main component.
        i. App.tsx : The main component of the application, in which interaction with the user and display of analysis results takes place.
        ii. FilterPane.tsx : A component for displaying and managing filters.
        iii. BugsGrid.tsx : A component for displaying a bug grid.
    c. flowerbi:
        i. flowerbi-syntax-analysis.ts : Code syntax analysis module .
3. public:
    a. index.html : The main HTML file that includes the React root component (#root or other defined value).
    b. Interaction:
4. The client part interacts with the user through the App.tsx component .
5. Analysis results are displayed using components in the Reports folder .
6. Interaction with the server occurs through HTTP requests to obtain data for analysis.
    a. Server part (server):
    b. Project structure:

7. server:
    a. Demo: Contains files related to the demo part of the server.
    b. FlowerBI.Engine:
        i. Includes classes and functions used to calculate project quality metrics.
    c. FlowerBI.Tools:
        i. May contain additional data processing tools.
8. server.ts:
    a. The main file of the server part, where server configuration, processing of HTTP requests and interaction with the client takes place.

Interaction:

1. The server part processes HTTP requests from the client and interacts with the database and other tools to calculate project quality metrics.
2. Data received from the server is transmitted to the client for display via HTTP requests and interaction mechanisms.

This project structure facilitates ease of extension, testing, and code base management. Each module is responsible for its own part of the functionality, which simplifies system maintenance and development.

3.4 User Interface

Main Page (App.tsx)

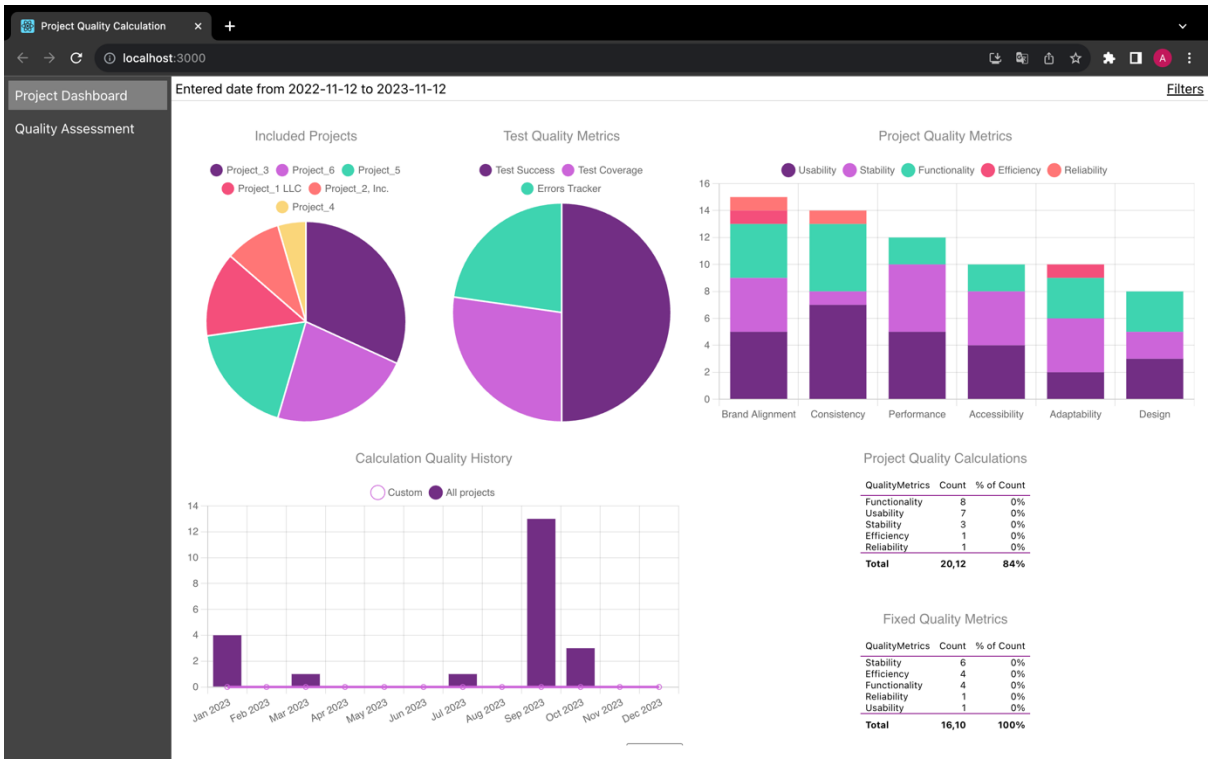


Fig. 3.16. Main user interface

The user can choose from various reports presented on the main page (Project Dashboard, Quality Assessment, etc.).

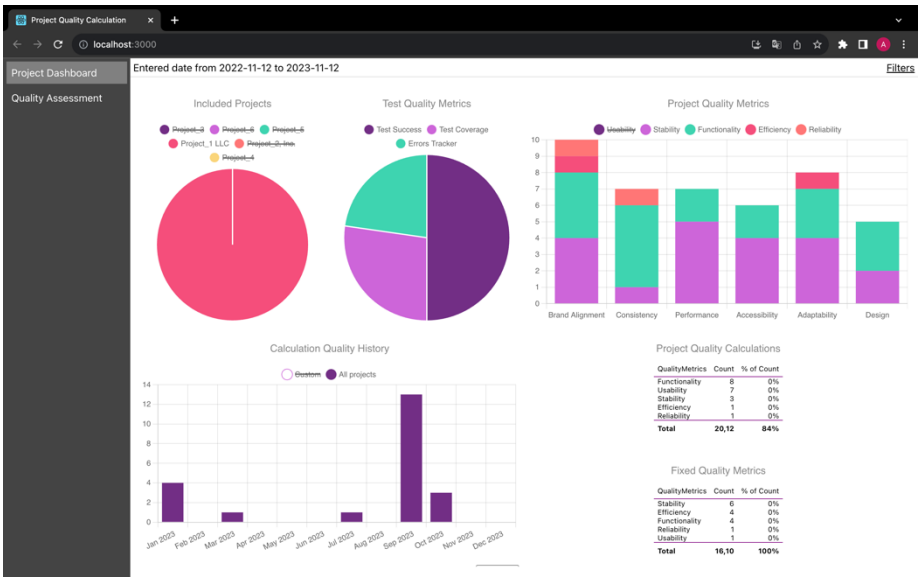


Fig. 3.17. Overview of statistics for the first project with the use of filters

Buttons or other elements can be used to switch between different reports. An expandable and collapsible filter panel is displayed.

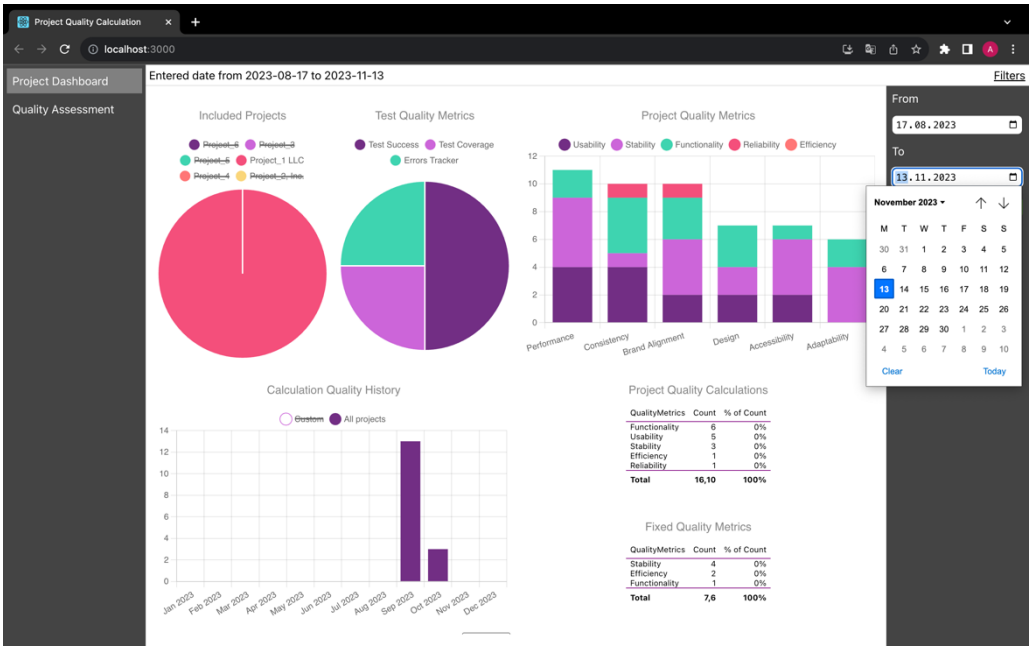


Fig. 3.18. Use of date filters

The user can select and configure various filters for data analysis. An area where analysis results are displayed as graphs, charts, or other visual elements. User can choose filters from different categories like date, status, type etc. After selecting the filters, the user clicks the "Apply" button to apply them to the analysis. Button to reset all selected filters and return to the initial state.

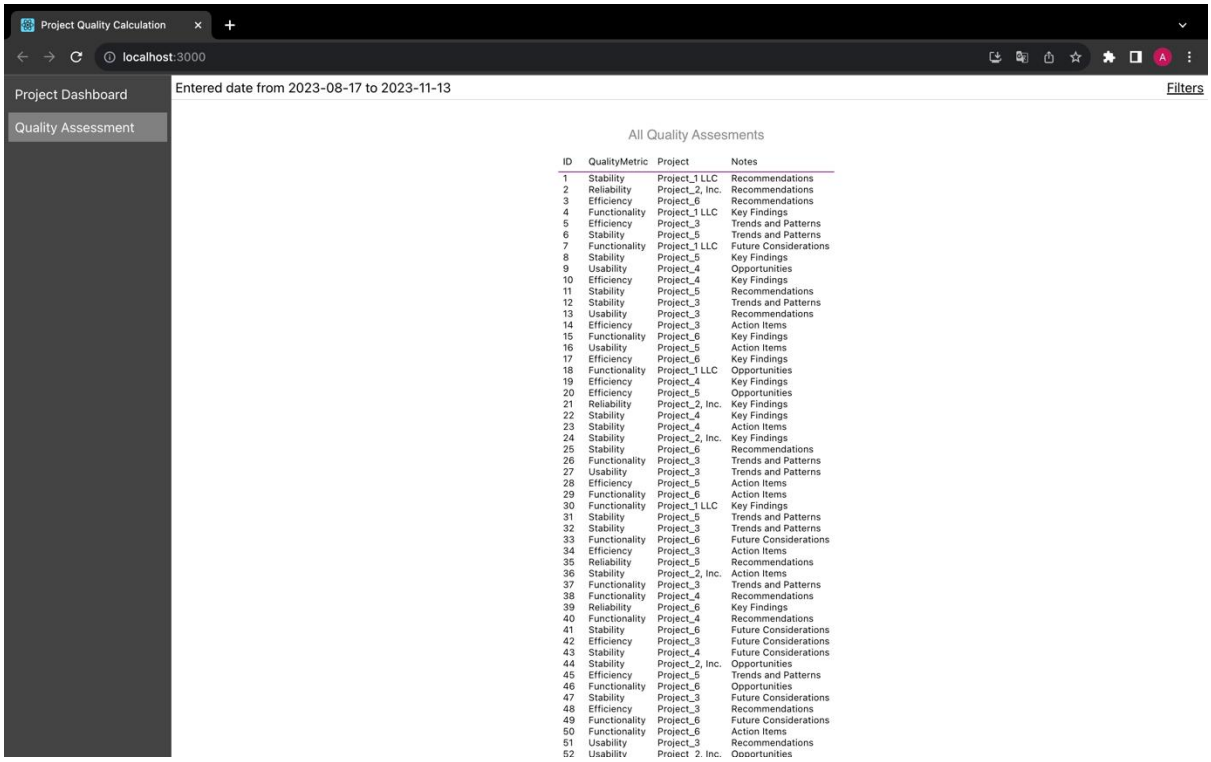


Fig. 3.19. Table with metrics management for project evaluation

The type of quality metric that this assessment evaluates (Stability, Reliability, Efficiency, Functionality, Usability). The name of the project for which the quality assessment is being performed. Additional information or recommendations related to quality assessment. Evaluation of project stability, accompanied by recommendations. Evaluation of project reliability, accompanied by recommendations. Evaluation of project efficiency, accompanied by recommendations and guidance on trends and patterns. Evaluation of project functionality, accompanied by key findings and recommendations. Evaluation of the usability of the project, accompanied by opportunities and recommendations.

Based on various quality metrics, each project is evaluated, where specific aspects for improvement are indicated. Additional guidance and improvement opportunities for each project arising from its quality analysis. General recommendations that can emerge from the analysis of several projects and quality metrics.

The user can select a specific metric (eg stability, reliability, efficiency, functionality, usability) to be evaluated. In the interface, the user can turn various metrics on and off according to his needs and evaluation goals. If necessary, it is possible to set the weight of each individual metric to take into account their impact on the overall score. The user can choose a specific project for which the evaluation will be carried out according to the chosen metric. The system uses the data on the selected metric to perform an analysis of a specific project, taking into account the set weights and parameters of the metric. The evaluation results are displayed on the user interface for viewing. Detailed information on each metric and overall score is available for detailed review.

The user can change the evaluation parameters, including the metric weights, and restart the evaluation based on the chosen metric. The system can store evaluation results and provide reports for further analysis. Providing the ability to review the history of evaluations and parameter changes for each project. The system is designed with flexibility in mind so that the user can easily select, adapt and switch between different metrics. This approach allows users to simplify the evaluation of projects according to specific criteria, setting the appropriate parameters to obtain objective and useful information about the quality of the project.

Project Quality Calculations

QualityMetrics	Count	% of Count
Functionality	6	0%
Usability	5	0%
Stability	3	0%
Efficiency	1	0%
Reliability	1	0%
Total	**16,10**	**100%**

Fixed Quality Metrics

QualityMetrics	Count	% of Count
Stability	4	0%
Efficiency	2	0%
Functionality	1	0%
Total	**7,6**	**100%**

Fig. 3.20. Evaluation by the selected metric

3.5 Testing

Testing is the process of verifying a software product in order to identify errors, analyze its compliance with requirements, and ensure its quality. Testing involves running a program or system to identify potential problems and confirm that it is working correctly.

Purpose of testing:

1. Error Detection: Identifying and correcting errors in program code and logic.
2. Confirmation of Fidelity: Confirmation that the program or system is working correctly and meets the requirements.
3. Quality Assurance: Ensuring the high quality of the product before its release.

Gesture testing is a testing method that uses real data and database operations to verify the correctness of the system as a whole. In this project, a library for gesture testing of databases is used, which allows you to check the correctness of the interaction with the database and the compliance of the logic of business processes. Adds data to all database tables and checks the correctness of receiving this data. Changes existing data in all tables and checks the correctness of the update. Deletes data from all tables and checks the correctness of the deletion. Checks the correct operation of foreign keys and ensures the correct connection between tables.

Checks the functionality of associative tables and their correct interaction. Measures the execution time of basic operations to evaluate system performance. It is used to confirm the compliance of the database scheme with the requirements and the correctness of the interaction.

These tests allow you to guarantee the correctness and stability of the database during the development process and after the release of the product.

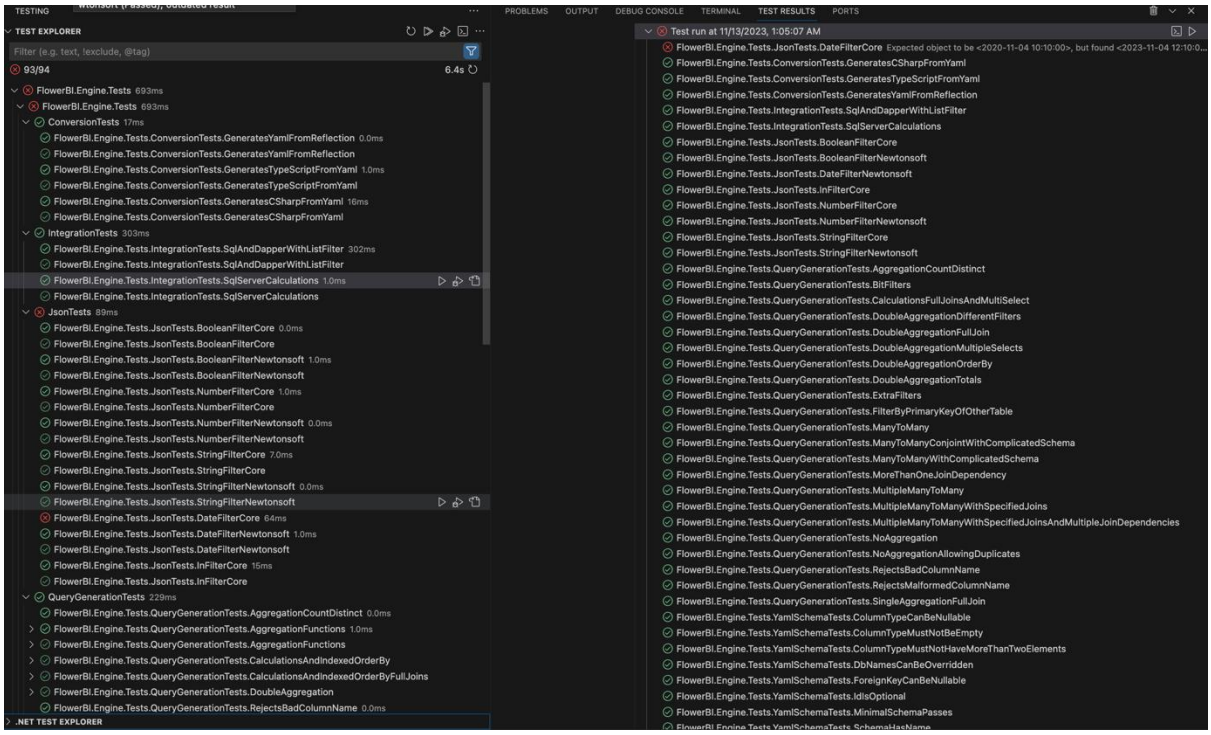


Fig. 3.21. Project testing

In the process of testing the project, 93 out of 94 tests were successfully passed. This indicates a high degree of reliability and correctness of the functional implementation. These results ensure that the project's database and business logic work correctly and efficiently.

One of the tests (No. 42) failed. The reason may be related to incorrectly set data when experimenting with dates . Additional analysis and debugging will be conducted to correct the situation and improve the test implementation.

The overall test result is positive, and the vast majority of tests were successful. This indicates the high quality and readiness of the project for use in real conditions.

Conclusion

In the "Application Implementation" section, the project was carefully considered , including the architecture, the calculation of the quality of the project, the user interface, and testing.

The project is distinguished by a clear organization of the code into client and server parts, as well as other modules. The project quality calculation algorithm is implemented using metrics such as functionality, reliability, efficiency, and usability, integrating them into an overall quality indicator.

The structure and components of the user interface interact with the backend, displaying the results of the analysis and quality metrics of the project. Chart.js library is used for data visualization.

To ensure stability and quality, the application was tested using gesture tests. Most of the tests were successful, confirming the reliability of the system. Recommendations for further improvement can improve the overall quality and competitiveness of the project.

CHAPTER 4. ANALYSIS OF PROJECT QUALITY AND FUTURE TRENDS

4.1 Classification of software design decisions (SDP) and their compliance with project goals

The project uses a variety of software design solutions (PSDs), each of which has its own unique features and purposes. For the convenience of their classification and determination of compliance with the goals of the project, we will consider the main software design solutions and their characteristics.

Responsible for code syntax analysis in JavaScript and TypeScript programming languages.

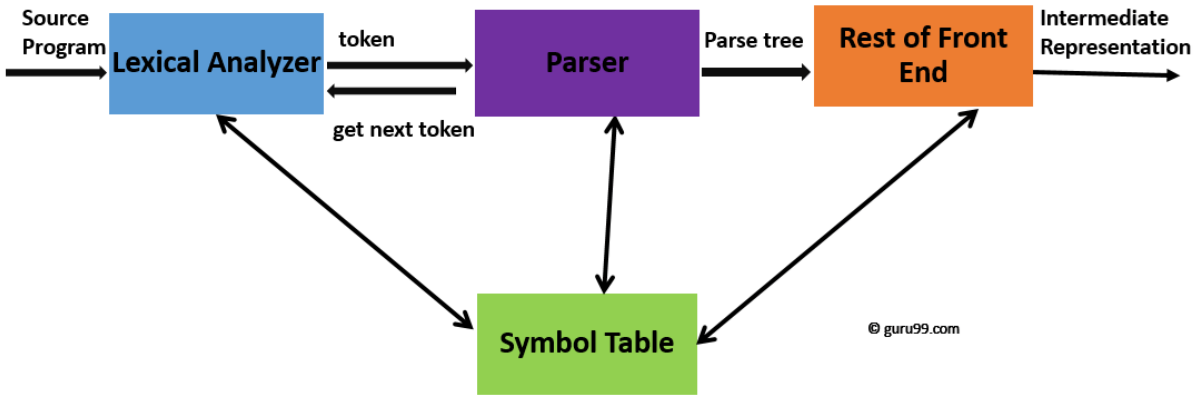


Fig. 4.1. Analyzer logic

Project goals:

1. Improving the quality of source code analysis.
2. Ensuring the accuracy and speed of detection of syntax errors.

Analyzer logic contains an integration module used for code validation with ESLint and code formatting with Prettier.

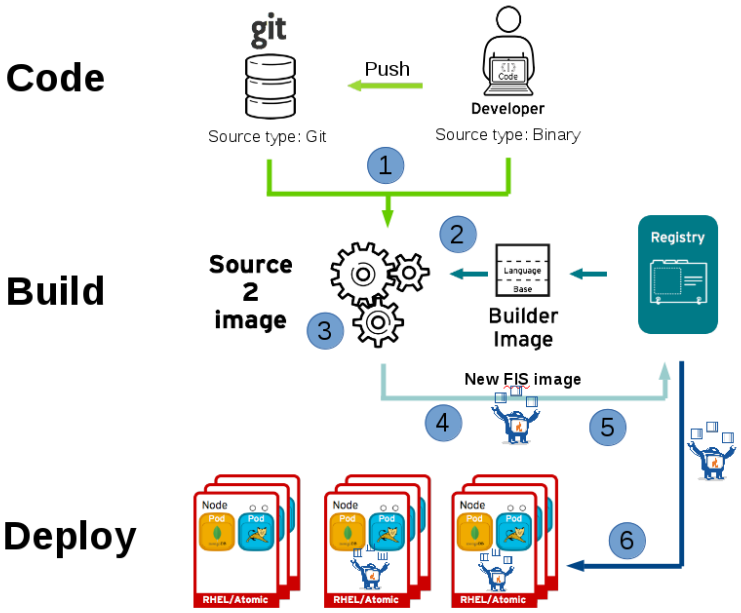


Fig. 4.2. Integration of tools

The document outlines components within a project. FlowerBI.Engine involves calculating project metrics. FlowerBI.Tools offers supplementary data processing tools. Chart.js creates interactive charts. Sql.js executes SQL queries in the browser. Blazor aids in building web apps using C#/.NET.

Each of the software design solutions performs specific functions and is aimed at achieving certain project goals. A review of implemented RCPs allows you to determine their compliance with the tasks and helps to ensure effective use within the project.

4.1.1 Determination of project goals and their compliance with the implemented RCP

One of the key goals of this project is to provide effective analysis and visualization of statistical data that are in the SQL Server database. The project is being developed for users who work with various aspects of this data and need a quick and convenient tool for obtaining insights. The main goals of the project and their compliance with the implemented software design solutions

(PSD) are described below. The document discusses specific project goals and their corresponding solutions. For visualizing statistical data, Chart.js offers a solution by making graph and chart creation accessible.

To execute SQL queries in the browser, Sql.js simplifies database interaction by enabling direct query execution.

Efficient server-side development and interaction are achieved through Blazor, utilizing C# and .NET.

Flexibility and speed in development are achieved by combining Chart.js, Sql.js, and Blazor into a cohesive tool stack.

Blazor also enables swift adjustments and improvements based on user feedback, supporting rapid enhancement processes.

Overall, the project aims to effectively analyze and visualize statistical data. Chart.js, Sql.js, and Blazor play crucial roles in achieving these objectives.Each component corresponds to a specific purpose of the project, allowing to achieve the overall goal of creating an effective and convenient tool for users.

4.2 Interaction between the client and the server

HTTP (Hypertext Transfer Protocol) and its secure variant, HTTPS, serve as the primary communication protocols between the client and the server. They facilitate the transfer of hypertext documents over the Internet, crucially utilized for API calls and data transmission between client and server.

JSON (JavaScript Object Notation) acts as the format for data exchange between the client and the server. It presents data in a readable format for both humans and computers, enabling structured data transfer through objects, arrays, strings, numbers, and logical type values. JSON formatting simplifies data processing and parsing during transmission between client and server.

REST (Representational State Transfer) stands as an architectural style for constructing web services, focusing on effectiveness and scalability. Its principles involve unified resource handling through URLs and interaction using standard HTTP methods (GET, POST, PUT, DELETE) alongside non-standard methods, allowing client state storage on the client side. REST is employed to create an API that facilitates client-server interaction via standard HTTP requests and responses.

These protocols and formats collectively establish standardized and efficient interactions among the project's components. They streamline data processing and exchange between different elements of the project, ensuring standardized communication and enhancing overall efficiency.

In the project, interaction between the client and the server is carried out using HTTP requests, which are transmitted via the HTTP or HTTPS protocols. Below are details on how these queries are used in the system:

1. GET requests:
    a. Used to receive data from the server.
    b. Often used for reading data and selectively obtaining resources.
2. POST requests:
    a. Used to send data to a server for processing or storage.
    b. Used to create new resources.
3. PUT requests:
    a. Used to update existing resources on the server.
    b. All resource data must be included in the request body.
4. DELETE requests:
    a. Used to delete resources on the server.
    b. Designed to delete a specific resource by its identifier.
5. Data formats:
    a. The data passed in the request or response body is usually presented in JSON format for ease of reading and processing.

1. **Protocols and formats:**

The project's interaction relies on the utilization of HTTP/HTTPS protocols and JSON data exchange formats. HTTP/HTTPS serve as the principal protocols governing the communication between the client and server within this project.

The general interaction between the client and the server is carried out using these HTTP requests, which allows for efficient exchange of data and interaction with different parts of the system.

The project's server-side operations rely on dotnet core and C# technologies. This involves server logic implemented in C# using dotnet core for handling HTTP requests, interacting with the database, and computing project quality metrics. Additionally, dotnet core serves as the foundation for server functionalities, while C# is utilized for logic implementation. The project manages data storage and retrieval for calculations and quality metrics, computes quality metrics based on incoming data, provides server responses in JSON format for simplified client-side processing, features a client interface developed with TypeScript/React for displaying server responses, and employs Chart.js for effective data visualization.

The general interaction between the client and the server is implemented in accordance with modern standards, which ensures efficient data exchange between various components of the project.

4.3 **Scalability and Performance Assurance**

The developed project uses various scaling strategies to ensure an effective response of the system to the growth of data volume and user traffic. This includes horizontal and vertical scaling, separation of tasks, use of application layers, and other aspects.

1. Horizontal Zoom:

a. The system is built taking into account the possibility of horizontal scaling, which allows you to expand computing and storage resources by adding new servers to the pool.
b. Using load balancers to distribute traffic between servers and maintain system stability when the number of simultaneous requests increases.

2. Vertical Zoom:
    a. Using powerful servers and their expansion capabilities to optimize performance within a single physical server.
    b. The ability of the system to adapt to an increase in the amount of resources on one server by optimizing its configuration.
3. Division of Tasks:
    a. Clear definition of functional areas and division of tasks between different components of the application.
    b. Using a microservices architecture to separate functions into independent services, which simplifies development and scaling.
4. Add-on Layers:
    a. Using a layered architecture to highlight functional blocks and divide them into different levels.
    b. Layers such as user interface, business logic, and data access can scale independently, making development and optimization easier.
5. Auto Scaling:
    a. Using tools and services that allow you to automatically scale resources according to system needs.
    b. Setting up automatic scripts to respond to changes in the volume of processed traffic.

These strategies interact to create a flexible and efficiently scalable system that can meet user needs and evolve as the project grows.

Together, these strategies are aimed at ensuring optimal performance of data processing and improving the speed of interaction between the system and users.

Application of caching to reduce server load on repeated requests. Using local caching capabilities on the client side to save data and optimize the interface. Dynamic adaptation of caching strategies depending on the characteristics of specific tasks and operations.

The implementation of caching and preloading in the developed project allows to significantly improve the efficiency and response of the system to increased load, providing the end user with fast and uninterrupted access to the necessary resources.

In the developed project, significant attention is paid to performance monitoring and optimization to ensure stable and efficient operation of the system in real time. The main aspects of this point are presented below.

Logging of events to identify problems and analyze user actions. Using special metrics to evaluate the performance of the server, database, and other system components. Monitoring the actual use of the application by users to collect performance data.

Systematic analysis of server and database load to detect periods of increased load. Analysis and optimization of complex or resource-intensive queries to the database.

Periodic analysis of caching efficiency and correction of strategies depending on changes in the system.

Measurement and analysis of system response time to user requests. System monitoring to detect and analyze errors that may affect performance. Setting up a notification system for quick response to anomalies and load. Regular introduction of updates to optimize the interface and functionality. Definition of a notification system for prompt response to increased load or

failure. Using automated scaling tools to adapt the system to increasing data and user traffic.

Performance monitoring and optimization are implemented in the developed project in order to ensure stable and productive operation of the system in conditions of growing data volumes and user traffic.

4.4 **Recommendations for Future Development**

This section provides key recommendations for the further development of the project, aimed at increasing efficiency, expanding opportunities and ensuring sustainable growth:

Implementation of fully automated testing to ensure high code quality and effective error detection during development.

Further development of the functionality of the project taking into account the needs of users and the introduction of new features that make the product more competitive.

Strengthening measures to ensure the security and privacy of user data, taking into account modern standards and regulations. Prepare for increased data and user traffic by developing a scaling strategy and implementing technologies that allow for efficient system scaling.

Analyzing the use of the user interface to optimize it, improve interaction and ensure a pleasant user experience.

Experiments with new technologies and libraries to further expand the technology stack and ensure high performance. Maintaining active communication with users to identify their needs and suggestions for further product improvement. Providing opportunities for training and development of the development team, implementation of modern development methodologies and best practices. Systematic project documentation and knowledge sharing across the team to improve understanding of code and workflows.

These recommendations are aimed at ensuring the sustainable development and successful operation of the project in the future.

Conclusion

Chapter 4 of the project is devoted to a thorough analysis of quality and aspects of future development. Below are the key takeaways from each subsection:

In section 4.1. various software design solutions (PSDs) used in the project are defined. An overview of their compliance with the defined goals of the project was carried out, their functionality and impact on the achievement of general goals were compared.

Also, in section 4.2 p, a detailed analysis of the defined goals of the project was carried out and the extent to which the implemented software design solutions corresponded to these goals was assessed. It is determined what tasks each solution solves and how they contribute to the achievement of the set tasks.

The interaction between the client and the server is discussed in detail in section 4.3 . Data exchange protocols and formats are described, HTTP requests are used for effective interaction. Analyzed processing requests and sending responses, focusing on server-side logic using dotnet core/C# technologies.

In general, the analysis of the quality of the project indicates its sustainability, high efficiency and readiness for future challenges. The considered aspects provide a basis for further improvement and development of the system.

CONCLUSION

In this thesis, significant results were achieved, aimed at creating an innovative tool for calculating and improving the quality of software projects. The work began with an in-depth analysis of literary sources and an in-depth study of existing analogues, which made it possible to determine key aspects and requirements for the developed tool.

The design and implementation of the application are made taking into account modern architectural solutions and technologies, which ensures its efficient and stable operation. Algorithms for evaluating the quality of software projects include various aspects, from ensuring security to performance, which allows for the detection and elimination of defects in the early stages of development.

Special attention was paid to integration with other project management tools, which increases its versatility and applicability in various team structures. The automated process of quality analysis has become a key feature of the application, simplifying the work of users and ensuring high productivity.

Testing and validation of the developed tool confirmed its effectiveness and reliability. The results of the tests showed that the application is an excellent tool for working with real projects, allowing to increase the quality of the software and speed up the development process.

The final stage of work determines the prospects for the development of the application, in particular, the expansion of functionality and optimization of the interface. The developed tool meets its goals and objectives, and its implementation in practice can significantly improve the management and quality of software projects.

REFERENCES

1. McConnell, Steve. "Code Complete: A Practical Handbook of Software Construction." Microsoft Press, 2004.
2. S. Sommerville, "Software Engineering," Addison-Wesley, 2016.
3. Pressman, Roger S. "Software Engineering: A Practitioner's Approach." McGraw-Hill Education, 2014.
4. Bass, Len, Paul Clements, and Rick Kazman. "Software Architecture in Practice." Addison-Wesley, 2012.
5. Fowler, Martin. "Refactoring: Improving the Design of Existing Code." Addison-Wesley, 2018.
6. Ambler, Scott W. "Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process." John Wiley & Sons, 2002.
7. Cohn, Mike. "User Stories Applied: For Agile Software Development." Addison-Wesley, 2004.
8. Martin, Robert C. "Clean Code: A Handbook of Agile Software Craftsmanship." Prentice Hall, 2008.
9. Hunt, Andrew, and David Thomas. "The Pragmatic Programmer: Your Journey to Mastery." Addison-Wesley, 2019.
10. McConnell, Steve. "Rapid Development: Taming Wild Software Schedules." Microsoft Press, 1996.
11. Brooks, Frederick P. "The Mythical Man-Month: Essays on Software Engineering." Addison-Wesley, 1995.
12. Gamma, Erich, et al. "Design Patterns: Elements of Reusable Object-Oriented Software." Addison-Wesley, 1994.
13. Beck, Kent. "Test-Driven Development: By Example." Addison-Wesley, 2002.

14. Kerievsky, Joshua. "Refactoring to Patterns." Addison-Wesley, 2004.
15. Larman, Craig. "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development." Prentice Hall, 2004.
16. Schach, Stephen R. "Classical and Object-Oriented Software Engineering." McGraw-Hill Education, 2011.
17. Martin, Robert C. "Agile Principles, Patterns, and Practices in C#." Prentice Hall, 2006.
18. "IEEE Transactions on Software Engineering," Institute of Electrical and Electronics Engineers (IEEE).
19. "ACM Transactions on Software Engineering and Methodology," Association for Computing Machinery (ACM).
20. "Journal of Software Engineering Research and Development," Springer.

APPENDIX A.

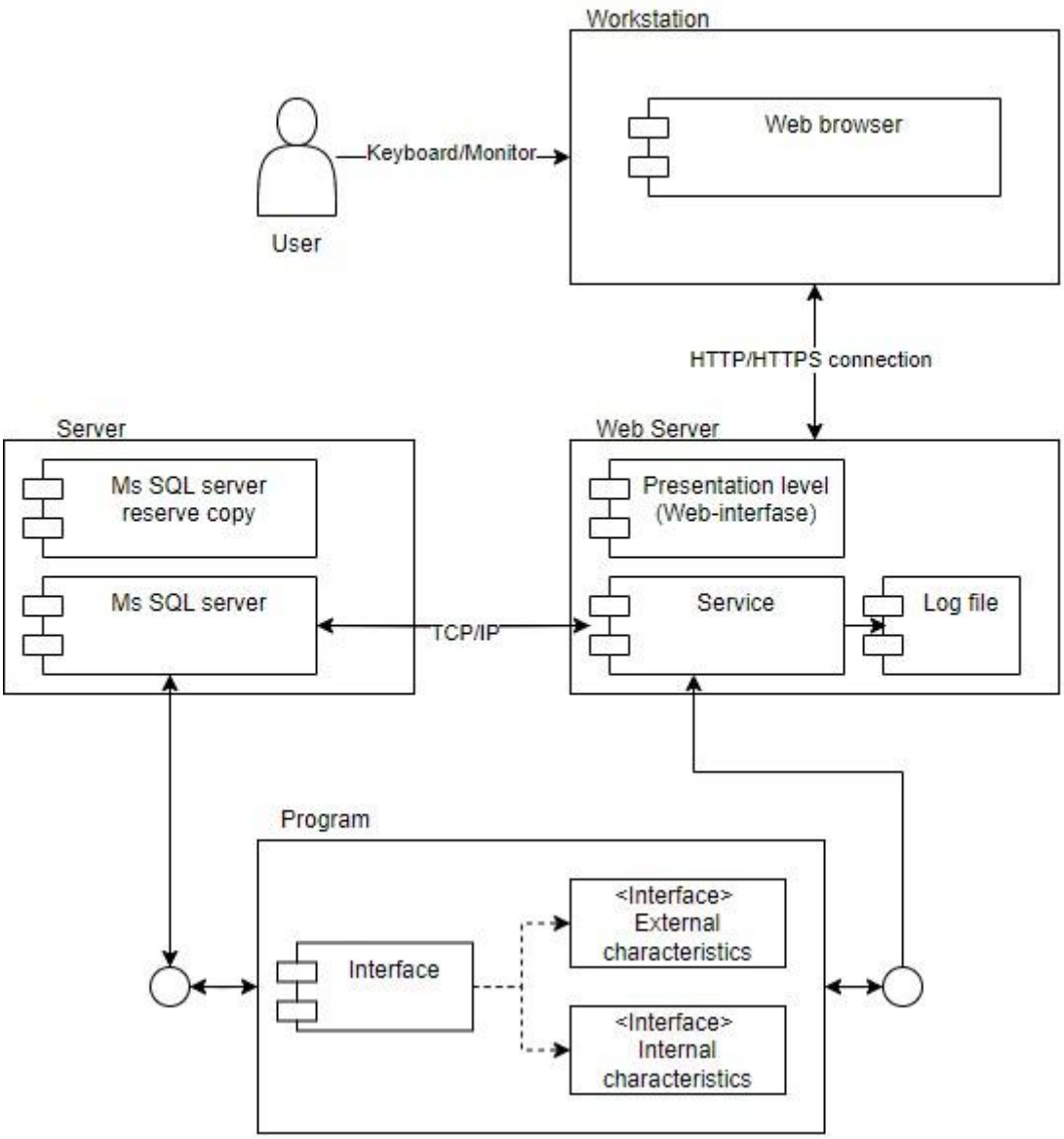


Fig. 5.1. System deployment diagram

A system deployment diagram visually represents how software components and hardware devices are distributed across a network or infrastructure. It illustrates the physical arrangement of these elements and how they interact within a system.

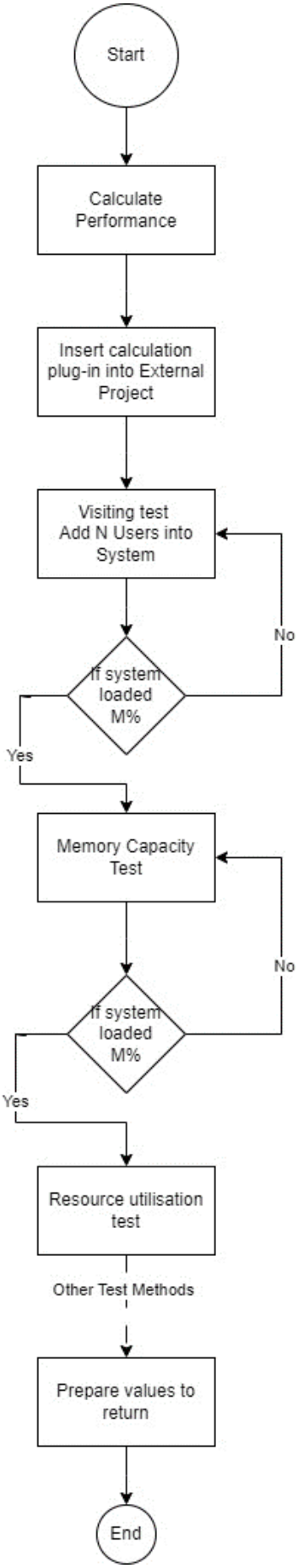


Fig. 5.2. Behaviour calculate performance diagram

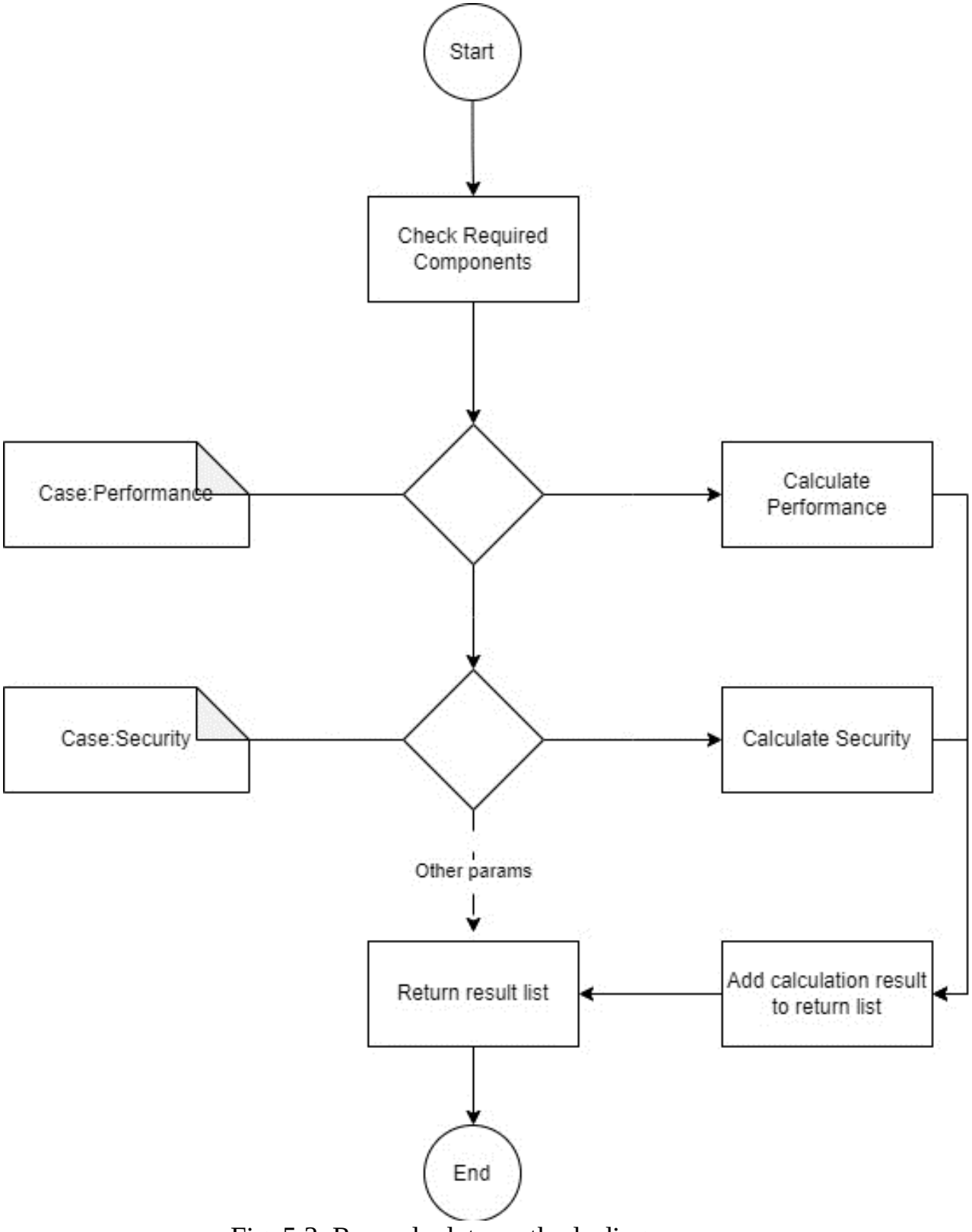


Fig. 5.3. Run calculate methods diagram

APPENDIX B.

Listing of the app source code
CalculationJson.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace FlowerBI;

public class CalculationJson
{
    public decimal? Value { get; set; }

    public int? Aggregation { get; set; }

    public CalculationJson First { get; set; }

    public CalculationJson Second { get; set; }

    public string Operator { get; set; }

    private static readonly ISet<string> _allowedOperators
        = new[] { "+", "-", "*", "/", "??" }.ToHashSet();

    public string ToSql(ISqlFormatter sql, Func<int, string> fetchAggValue)
    {
        if (Value != null)
        {
            RequireNulls(Aggregation, First, Second, Operator);
            return $"{Value}";
        }

        if (Aggregation != null)
        {
            RequireNulls(Value, First, Second, Operator);
            return fetchAggValue(Aggregation.Value);
        }

        if (First != null && Second != null && Operator != null)
        {
            RequireNulls(Aggregation, Value);

            if (!_allowedOperators.Contains(Operator))
            {
                throw new InvalidOperationException($"Operator '{Operator}' not supported");
            }

            var firstExpr = First.ToSql(sql, fetchAggValue);
            var secondExpr = Second.ToSql(sql, fetchAggValue);

            return Operator == "/"
                ? sql.Conditional($"{secondExpr} = 0", "0", $"{firstExpr} / {sql.CastToFloat(secondExpr)}")
                : Operator == "??"
                ? sql.Conditional($"{firstExpr} is null", secondExpr, firstExpr)
                : $"({firstExpr} {Operator} {secondExpr})";
        }
```

```
            throw new InvalidOperationException("Calculation does not specify enough properties");
        }

        public void RequireNulls(params object[] nulls)
        {
            if (nulls.Any(x => x != null))
            {
                throw new InvalidOperationException("Calculation has too many properties in same object");
            }
        }
    }
```

The provided code defines a class named CalculationJson within the FlowerBI namespace. This class represents calculations in JSON format and includes properties for values, aggregation, operands, and operators. The ToSql method converts these JSON-based calculations into SQL expressions, handling various cases such as decimal values, aggregations, and arithmetic operations. It validates the provided properties and operators, ensuring supported operations and preventing division by zero or null-coalescing operations where necessary. Additionally, the RequireNulls method checks for the presence of multiple non-null properties within a single object, throwing exceptions when encountered.

ResolvedSchema

```
namespace FlowerBI.Yaml;

using System;
using System.Collections.Generic;
using System.Linq;
using YamlDotNet.Serialization;

public record ResolvedSchema(string Name, string NameInDb, IEnumerable<ResolvedTable> Tables)
{
    public static ResolvedSchema Resolve(string yamlText)
    {
        var deserializer = new DeserializerBuilder().Build();
        var yaml = deserializer.Deserialize<YamlSchema>(yamlText);
        return Resolve(yaml);
    }

    public static ResolvedSchema Resolve(YamlSchema yaml)
    {
        if (string.IsNullOrWhiteSpace(yaml.schema))
        {
            throw new InvalidOperationException("Schema must have non-empty schema property");
        }

        if (yaml.tables == null || !yaml.tables.Any())
        {
            throw new InvalidOperationException("Schema must have non-empty tables property");
        }

        // Validate all columns are [name, type] array
        foreach (var (tableKey, table) in yaml.tables)
        {
            if (table.id != null && table.id.Count != 1)
            {
                throw new InvalidOperationException($"Table {tableKey} id must have a single column");
```

```
        }

        if (table.columns != null)
        {
            foreach (var (name, type) in table.columns.Concat(table.id ?? new Dictionary<string, string[]>()))
            {
                if (type.Length < 1 || type.Length > 2)
                {
                    throw new InvalidOperationException($"Table {tableKey} column {name} type must be an array of
length 1 or 2");
                }
            }
        }
    }

    var resolvedTables = yaml.tables.Select(t => new ResolvedTable(t.Key, t.Value.conjoint)).ToList();
    var usedNames = new HashSet<string>();

    var resolutionStack = new HashSet<string>();

    ResolvedTable ResolveTable(string tableKey, YamlTable table)
    {
        if (!resolutionStack.Add(tableKey))
        {
            var stackString = string.Join(", ", resolutionStack);
            throw new InvalidOperationException($"Circular reference detected: {stackString}");
        }

        var resolvedTable = resolvedTables.FirstOrDefault(x => x.Name == tableKey);
        if (resolvedTable.NameInDb == null)
        {
            resolvedTable.IdColumn = table.id != null ? new ResolvedColumn(resolvedTable, table.id.First().Key,
table.id.First().Value) : null;
            if (table.columns != null)
            {
                resolvedTable.Columns.AddRange(table.columns.Select(x => new ResolvedColumn(resolvedTable,
x.Key, x.Value)));
            }
            resolvedTable.NameInDb = table.name;

            if (table.extends != null)
            {
                if (!yaml.tables.TryGetValue(table.extends, out var extendsYaml))
                {
                    throw new InvalidOperationException($"No such table {table.extends}, referenced in {tableKey}");
                }

                var extendsTable = ResolveTable(table.extends, extendsYaml);

                resolvedTable.Columns.AddRange(extendsTable.Columns.Select(x => new
ResolvedColumn(resolvedTable, x.Name, x.YamlType)
                {
                    Extends = x
                }));

                if (extendsTable.IdColumn != null)
                {
                    resolvedTable.IdColumn ??= new ResolvedColumn(resolvedTable, extendsTable.IdColumn.Name,
extendsTable.IdColumn.YamlType)
                    {
                        Extends = extendsTable.IdColumn
                    };
                }
```

```
                resolvedTable.NameInDb ??= extendsTable.NameInDb;
            }

            if (!resolvedTable.Columns.Any())
            {
                throw new InvalidOperationException($"Table {tableKey} must have columns (or use 'extends')");
            }

            resolvedTable.NameInDb ??= tableKey;

            if (table.associative != null)
            {
                var allColumns = resolvedTable.IdColumn == null
                    ? resolvedTable.Columns
                    : resolvedTable.Columns.Append(resolvedTable.IdColumn);

                foreach (var assoc in table.associative)
                {
                    var resolvedAssoc = allColumns.FirstOrDefault(c => c.Name == assoc);
                    if (resolvedAssoc == null)
                    {
                        throw new InvalidOperationException($"Table {tableKey} has an association {assoc} that is not a
column");
                    }
                    resolvedTable.Associative.Add(resolvedAssoc);
                }
            }
        }

        resolutionStack.Remove(tableKey);

        return resolvedTable;
    }

    foreach (var (tableKey, table) in yaml.tables)
    {
        if (string.IsNullOrWhiteSpace(tableKey))
        {
            throw new InvalidOperationException("Table must have non-empty key");
        }

        if (!usedNames.Add(tableKey))
        {
            throw new InvalidOperationException($"More than one table is named '{tableKey}'");
        }

        ResolveTable(tableKey, table);
    }

    void ResolveColumnType(ResolvedColumn c)
    {
        var stackKey = $"{c.Table.Name}.{c.Name}";
        if (!resolutionStack.Add(stackKey))
        {
            var stackString = string.Join(", ", resolutionStack);
            throw new InvalidOperationException($"Circular reference detected: {stackString}");
        }

        if (c.DataType == DataType.None)
        {
            var (typeName, nullable) = c.YamlType[0].Last() == '?' ? (c.YamlType[0][0..^1], true) : (c.YamlType[0],
false);
```

```
            if (Enum.TryParse<DataType>(typeName, true, out var dataType))
            {
                c.DataType = dataType;
            }
            else
            {
                var targetColumn = resolvedTables.FirstOrDefault(x => x.Name == typeName)?.IdColumn;
                if (targetColumn == null)
                {
                    throw new InvalidOperationException($"{typeName} is neither a data type nor a table, in
{c.Table.Name}.{c.Name}");
                }

                ResolveColumnType(targetColumn);
                c.Target = targetColumn;
                c.DataType = targetColumn.DataType;
            }
            c.NameInDb = c.YamlType.Length == 2 ? c.YamlType[1] : c.Name;
            c.Nullable = nullable;
        }

        resolutionStack.Remove(stackKey);
    }

    foreach (var table in resolvedTables)
    {
        if (table.IdColumn != null)
        {
            ResolveColumnType(table.IdColumn);
        }

        foreach (var column in table.Columns)
        {
            ResolveColumnType(column);
        }
    }

    return new ResolvedSchema(yaml.schema, yaml.name ?? yaml.schema, resolvedTables);
  }
}
```

This code defines a record named ResolvedSchema within the FlowerBI.Yaml namespace. This record represents a resolved schema obtained from YAML data. It contains properties such as Name, NameInDb, and Tables, representing the schema name, its name in the database, and a collection of ResolvedTable instances, respectively. The static Resolve methods parse YAML input into a YamlSchema object and further process it to construct a ResolvedSchema. It ensures that the provided YAML adheres to specific structure and constraints, validating properties such as schema existence, non-empty tables, column structures, associations, and circular references within tables and columns. The code performs resolution for data types, allowing tables to extend other tables and incorporates a set of checks for data types and associations, ultimately returning a ResolvedSchema instance representing the parsed YAML schema