iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VII **Month of publication:** July 2022

DOI: https://doi.org/10.22214/ijraset.2022.45577

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VII July 2022- Available at www.ijraset.com

Development of Client-Specific Web Service for Manipulating Real-Time Contextual Data

Dhruva K R
ISE, RVCE, Mysore Rd, RV Vidyaniketan, Post, Bengaluru, Karnataka, India 560059

Abstract: ***The aim of this work was to create a fully functional client-specific solution for accessing real-time context specific data. These kinds of interfaces have been all the buzz in the domain of banking and can help in achieving a faster turn-around-time for the queries being asked by the internal users spread all over the globe. These kinds of interfaces provide an effective way to communicate with a user and offer helpful support in a more economical way- saving both time and efforts being spent on accessing real-time data. The application built aims to provide more space for decision making as real-time access is enabled. The study below mainly concentrates on the various techniques used in developing the system for accessing real-time context-specific data.***
Keywords: Real-time data, context-specific, client-specific

I. INTRODUCTION

Financial objectives are usually coupled with technical solutions to facilitate various pathways to ensure capital raising. Majority of technical solutions are web-oriented, to ensure wide reach of the financial services to the client with real-time data access and manipulation. Real-time data access plays a major role in these financial pathways and techniques such as real-time trade prices, real-time deal acquisitions, real-time service auctions etc. Development of appropriate web service for displaying real-time contextual data is carried out with the usage of clean architecture principles.

II. LITERATURE REVIEW

In paper [1], it is explained how testing can help lower the cost of maintaining a specific software application. A brief review of the several XUnit test patterns and smells that can help to increase the ROI of the test code is provided in the paper by the authors. The research paper also contained fundamental test methods, services offered, and code coverage approaches. According to this survey, many financial institutions are focusing the majority of their investments on raising the test code's return on investment.
In [2], the summary of the newest web technology features is covered in the paper. The new version of HTML and other tools described in this paper are intended to formalize and standardize solutions for technologies and functions that have already been implemented through a variety of web developers' hacks and plug-ins. Many of these features will soon be added to browsers. Application programming interfaces that have recently been defined allow the programs to access these functions.
The research, [3], a model utilized for quick web application development is investigated and analyzed. This model is built on the Model-View-Controller architecture (MVC), and it also includes routing, form generation and validation, security, database access, and other helpful elements. The same techniques that were used to construct this model in the PHP programming language can also be used to implement it in other settings and development languages.
In [4], the usage of.NET Core in the creation of web apps has been covered by authors. The authors' paper provides a brief summary of the various analyses that include tests carried out on a test application built using the.NET Core framework.
In paper [5], study analyses the top 4000 most visited websites on Alexa.com to pinpoint 500 specific websites that make the claim to offer a REST web service API. Key technical characteristics, degree of REST architectural conformance, and adherence to best practices are all subject to analysis.

III. METHODOLOGY

A. Architecture

The Fig. 1 describes the overall High-level Architecture of the real-time data access system that was implemented. The system comprises of mainly three sub-components which are inter-dependent on each other for overall functioning of the system. The Application Core Project comprises of the interfaces, domain events, specifications etc. which acts as a foundation for the business logic. The Infrastructure project comprises of the database context module, cache service, SMS service etc. which collectively groups the external services required for the functioning of the system.

©IJRASET: All Rights are Reserved

The .NET Core web app covers the User Interface layer of the system which contains controllers, view models, views etc. which play a major role in the functioning of the user interface of the system. These three sub-components function hand-in-hand for the overall functionality of the system.

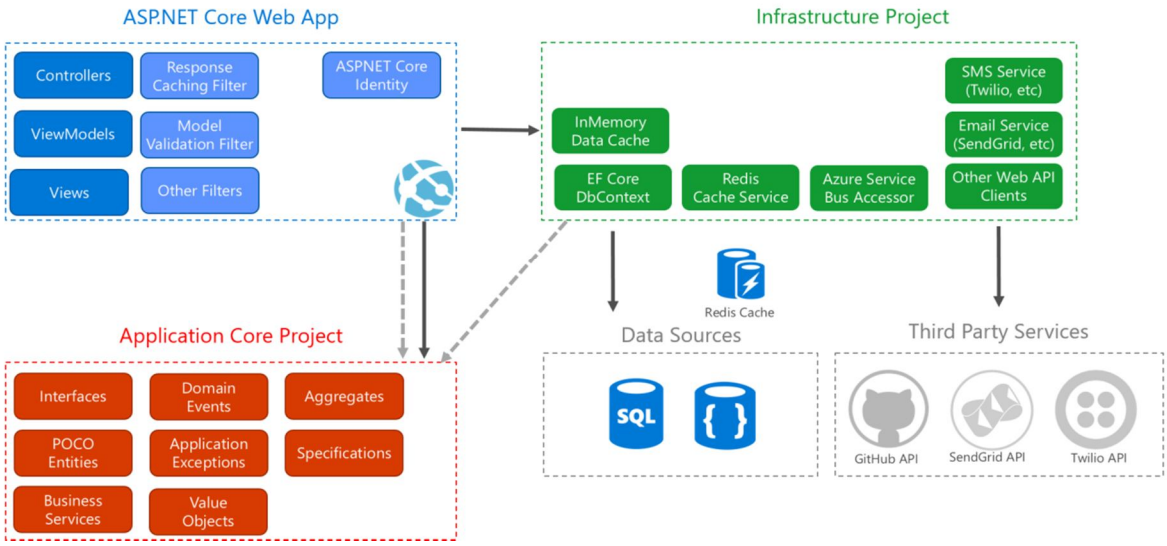


Fig. 1. Architecture diagram of the real-time data access system

B. Methodology

1) *Designing* - First, a suitable system structure is offered along with suitable documentation and requirements, which are explained using class diagrams, sequence diagrams, etc.
2) *Object Relational Modeling* – In this step, the application's business logic is properly mapped using the EFCore library to establish the fundamental framework for data source consumption.
3) *Backend* - In this step, the RESTful API is built using the.NET Core framework and includes tools for performing CRUD operations on the database's data.
4) *Frontend* - Ag-Grid is an open-source library that is used to build the framework of the frontend and offers pre-built React components that may be altered to meet our needs.
5) *Testing* - xUnit testing framework is used to test the CRUD operations in this step.

IV. RESULTS

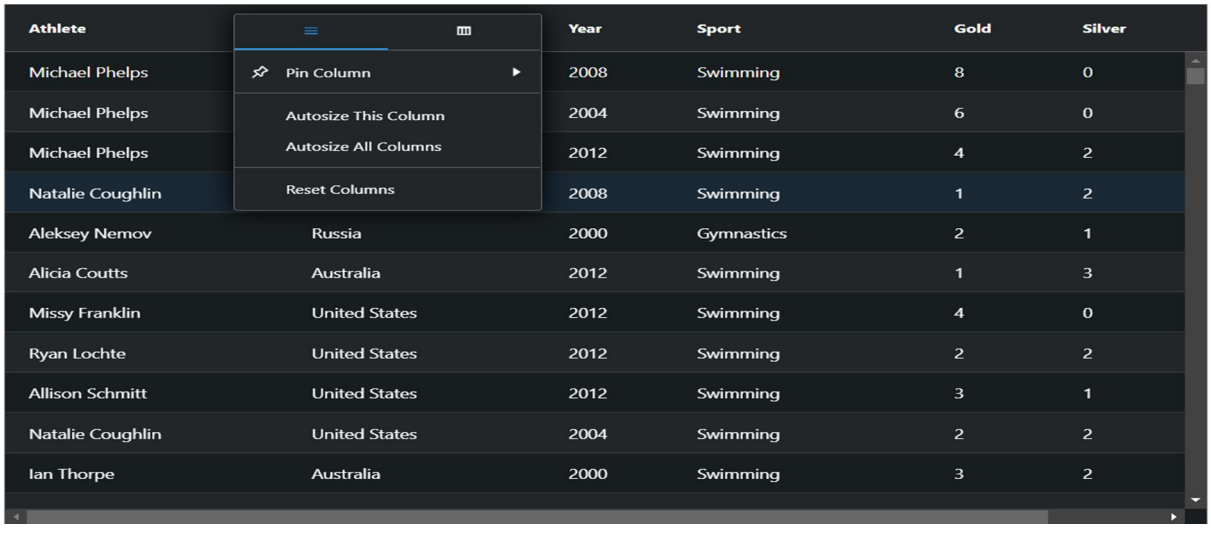


Fig. 2. Dashboard for accessing real-time data

The framework and structure of the system are designed to be adaptable and expandable for future requirements and use-case scenarios. The system has the necessary tools to give users immediate access to contextual data. A user-friendly and automated environment for engaging with clients is provided by the system's integration of the necessary notification systems, such as mailing services and SMS notification services. The system is designed to have a clean architecture, which separates the business logic from the system's infrastructure and user interface. To avoid issues like server outages or server inaccessibility, the system has a backup server installed, and it is programmed to balance the load appropriately.

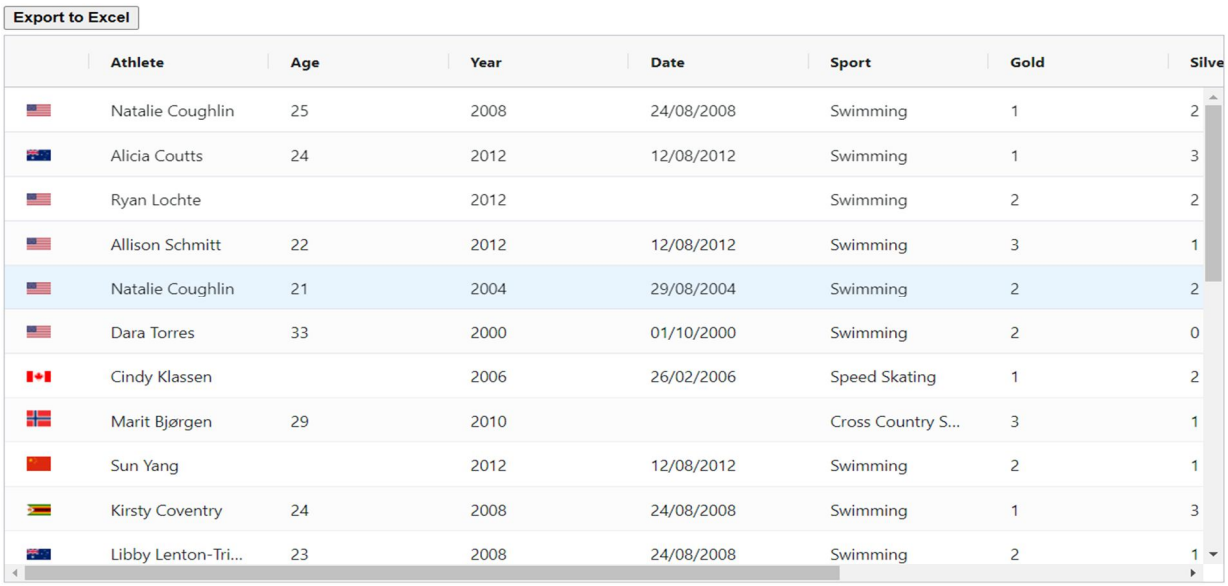


Fig. 3. Light-themed dashboard equipped with export functionality

V. FUTURE DISCUSSIONS

Integration with AI-powered bots to address FAQs and related questions about the application's process can be carried out. To boost the backend connectivity's performance, refactoring of the access-related approach is a way of practical implementation. Integration with user-specific logging techniques offers a quicker and simpler means of locating and resolving potential access-related user-specific faults. Integration with decentralized server systems aids in load balancing and increases the system's overall stability. Integration of database services in the cloud aids in resolving issues with database accessibility.

VI. CONCLUSION

Dependence on data has grown dramatically in today's environment. Access to constantly changing real-time data is therefore crucial for hassle-free decision making. Real-time access is crucial, but so is user-friendly filtering and customization. This project consists of a framework that offers a streamlined, user-friendly interface so that the user may manipulate data in real time without experiencing any lag. Also, this framework is intrinsically scalable to handle context-specific data because it may be defined in different situations. With very minor changes to the background process, this architecture can be scaled to meet the needs of the client. Integration with load balancing features make this framework reliable and easy for the connected clients to access.

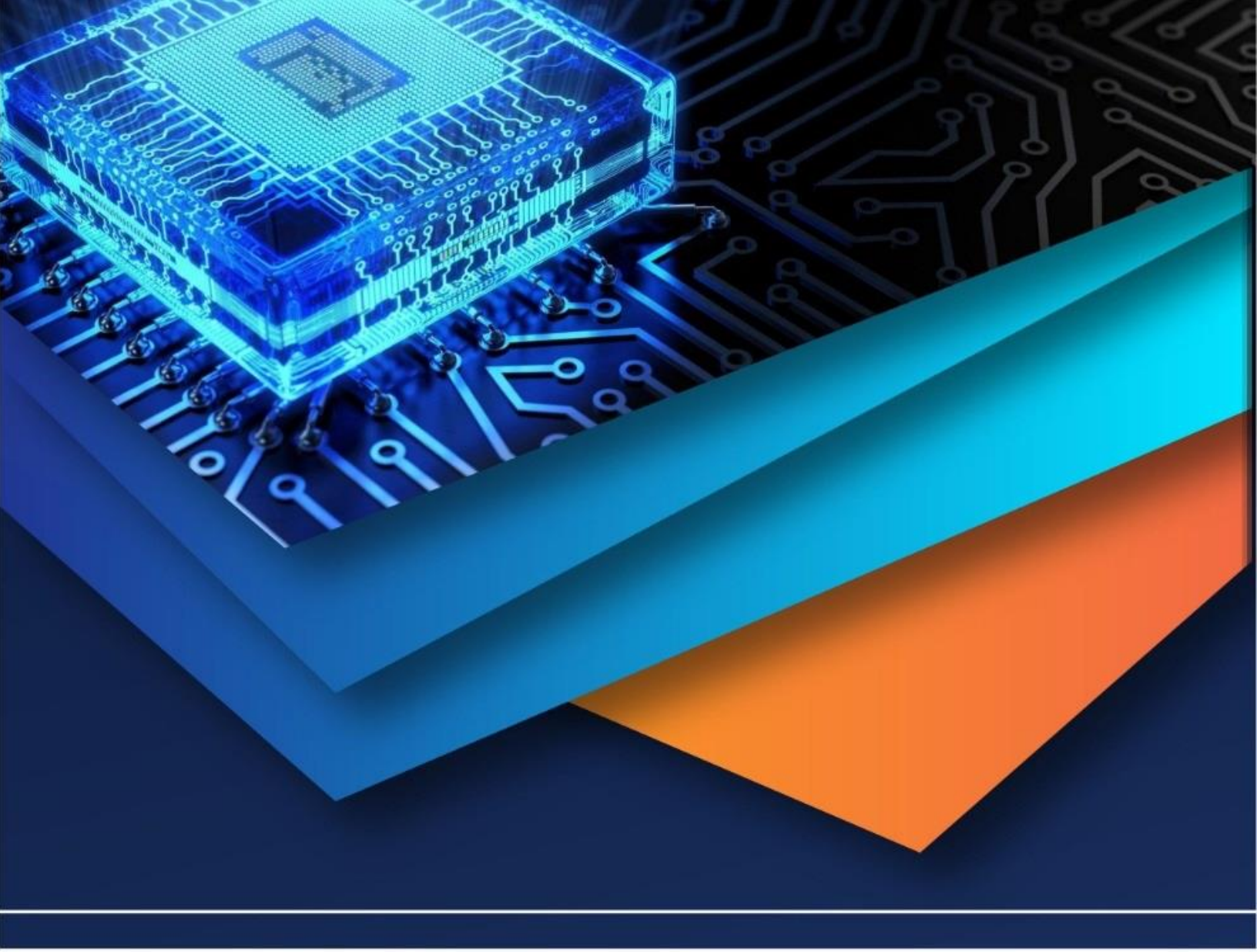
VII. ACKNOWLEDGMENT

Prof. Rekha B.S (Assistant Professor) was instrumental in the effective completion of this study, and the author would like to express their gratitude to them.

REFERENCES

[1] Gerard G Meszaros. XUnit test patterns and smells: improving the ROI of test code. In Companion to the 25th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, 2010
[2] Grega Jakus, Matija Jekovec, Saso Tomazic and Jaka Sodnik. New technologies for Web Development. Journal of Ljubljana University, 2010
[3] Dragos Paul-Pop, Adam Nelu Altar Samuel. Designing an MVC Model for Rapid Web Application Development. Journal of Romanian-American University, 2014
[4] Ewelina Piatkowska, Katarzyna Wasik The use of .NET Core in web applications development. Journal of Computer Science Institute, 2018
[5] Andy Neumann, Nuno Laranjeiro, Jorge Bernardino. An analysis of Public REST Web Service APIs. Journal of Computer Science Institute, 2018

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)