UDC 004.438.52:004.415.3:004.421.2
DOI https://doi.org/10.32689/maup.it.2025.2.26

Oleh SYPIAHIN
Master's Degree, Information Security,
Full Stack Engineer, floLive (Israel),
fed4wet@gmail.com
***ORCID:** 0009-0008-7565-3221*

Dmytro POLTAVSKYI
Bachelor's Degree,
Engineering Team Lead, Uplandme Inc. (USA),
poltavskyi.dmytro@gmail.com
***ORCID:** 0009-0009-0387-6677*

Roman MARTYNENKO
Master's Degree,
Senior Software Engineer, Henry AI, Inc. (USA),
worldofwbdesign@gmail.com
***ORCID:** 0009-0005-4663-8530*

EXPLORING THE ADVANTAGES AND LIMITATIONS OF THE FEATURE-SLICED DESIGN ARCHITECTURAL METHODOLOGY IN COMMERCIAL FRONTEND PROJECTS

Abstract. The purpose of this study *is to identify the advantages and limitations of the Feature-Sliced Design (FSD) architectural methodology in the development of commercial frontend applications, particularly using Angular-based projects that require a high degree of modularity, adaptability to change, and preservation of structural integrity throughout the software product life cycle. Special attention is given to assessing the feasibility of implementing this approach within small and medium-sized development teams operating in fast-paced environments with short release cycles and high demands for maintainable code.*

The research methodology *is based on a structural analysis of the theoretical foundations of FSD, practical modeling of software architecture using an open-source ToDo application implemented according to the full hierarchical structure–app, processes, pages, widgets, features, entities, and shared–and a comparative analysis with leading alternative architectural approaches such as Atomic Design, MVVM, and MVC. The comparison was conducted based on criteria such as scalability, cohesion, module decoupling, reusability of components, structural transparency, and support for continuous integration.*

The scientific novelty *lies in a comprehensive description of the core advantages of FSD, including business logic encapsulation, minimized inter-module dependencies, predictable structural hierarchy, and transparent component interaction logic. For the first time, the study systematizes the practical constraints associated with FSD implementation: the complexity of initial configuration, fragmented documentation, high architectural discipline requirements, and the onboarding difficulties for new developers.*

Conclusions. *The study confirms the effectiveness of FSD as an architectural paradigm for building stable, scalable, and long-term maintainable frontend systems. However, successful adoption requires a well-prepared development team, comprehensive internal documentation, standardized structuring practices, and strict adherence to established architectural principles. The practical significance of the research lies in offering recommendations for integrating FSD into commercial web applications with elevated requirements for architectural manageability, maintainability, and scalability.*

Key words: *Feature-Sliced Design, Angular, frontend architecture, scalability, layered structure, modularity, ToDo application, architectural methodology, technical debt, UX composition, project structure.*

Олег СИПЯГІН, Дмитро ПОЛТАВСЬКИЙ, Роман МАРТИНЕНКО. ДОСЛІДЖЕННЯ ПЕРЕВАГ ТА ОБМЕЖЕНЬ АРХІТЕКТУРНОЇ МЕТОДОЛОГІЇ FEATURE-SLICED DESIGN У КОМЕРЦІЙНИХ FRONTEND-ПРОЄКТАХ

Анотація. Метою дослідження *є виявлення переваг та обмежень архітектурної методології Feature-Sliced Design (FSD) у розробці комерційних frontend-застосунків, зокрема на прикладі Angular-проєктів, які вимагають високого ступеня модульності, адаптивності до змін і підтримки структурної цілісності на всіх етапах життєвого циклу програмного продукту. Особливу увагу приділено визначенню доцільності впровадження даного підходу у командах малого та середнього масштабу, що функціонують у середовищі інтенсивної розробки, з короткими релізними циклами та високими вимогами до підтримуваності коду.*

Методологія дослідження *базується на структурному аналізі теоретичних засад FSD, практичному моделюванні архітектури програмного забезпечення на прикладі ToDo-додатку з відкритим кодом, реалізованого*

© O. Sypiahin, D. Poltavskyi, R. Martynenko, 2025

Стаття поширюється на умовах ліцензії CC BY 4.0

відповідно до всієї ієрархії рівнів – app, processes, pages, widgets, features, entities та shared. Також здійснено порівняльний аналіз із провідними альтернативними архітектурними підходами – Atomic Design, MVVM і MVC – на основі критеріїв масштабованості, когезії, рівня зв'язаності модулів, можливості повторного використання компонентів, структурної прозорості та підтримки безперервної інтеграції.

***Наукова новизна** полягає в комплексному описі ключових переваг FSD, серед яких – інкапсуляція бізнес-функціоналу, мінімізація міжмодульних залежностей, передбачувана ієрархія структури та прозора логіка компонентної взаємодії. Вперше узагальнено обмеження, характерні для практичного застосування методології: складність початкового впровадження, фрагментарність документації, високі вимоги до архітектурної дисципліни та труднощі адаптації нових розробників у команду.*

***Висновки.** Результати дослідження підтверджують ефективність FSD як архітектурної парадигми для побудови стабільних, масштабованих і довготривало підтримуваних frontend-систем. Водночас успішне застосування підходу вимагає високої підготовленості команди, внутрішньої документації, уніфікованих підходів до структурування та чіткого дотримання прийнятих принципів. Практична значущість дослідження полягає у формулюванні рекомендацій щодо інтеграції FSD у комерційні вебзастосунки з підвищеними вимогами до архітектурної керованості, підтримуваності й масштабованості.*

***Ключові слова:** Feature-Sliced Design, Angular, frontend-архітектура, масштабованість, шарова структура, модульність, ToDo-додаток, архітектурна методологія, технічний борг, UX-композиція, структура проєкту.*

Problem Statement. In the context of increasing complexity in modern frontend systems and growing demands for scalability, modularity, and maintainability of source code, there is an urgent need for architectural approaches that support the effective organization of web application structure. One such approach is Feature-Sliced Design (FSD), an architectural methodology that involves dividing a software product into hierarchical layers (app, processes, pages, widgets, features, entities, shared) with clearly defined responsibilities. Despite the rising popularity of this approach within open-source communities, its practical implementation in commercial environments raises several concerns related to initial implementation costs, scalability challenges, onboarding of new team members, and alignment with business requirements. The problem addressed in this study lies in the critical analysis of the strengths and weaknesses of the FSD methodology when applied in real-world frontend development projects. The aim of the study is to identify the architectural, technical, and organizational conditions under which the use of FSD ensures maximum development efficiency and to determine the limitations that may arise in a commercial setting.

Analysis of Recent Studies and Publications. In the current environment of rapid frontend technology development, there is increasing interest in optimizing architectural solutions, particularly those that are modular and scalable, enabling effective management of complexity in large-scale projects. A significant position in this context is occupied by the Feature-Sliced Design (FSD) architectural methodology, which proposes structural separation of code across domain, functional, and interface levels. The relevance of this topic is confirmed by the emergence of studies analyzing both its potential and limitations in practical application. One of the conceptually closest approaches is HOFA, presented in the monograph by H. Ben Khalfallah. This approach emphasizes the creation of clean architecture in JavaScript and React applications, where maintaining isolation, separation of responsibilities, and modular organization is critically important for code maintainability [2]. The HOFA methodology closely corresponds to the principles of FSD, particularly in terms of component autonomy and the structuring of business functionality. Similarly, the study by M. Kolomoyets and Y. Kynash focuses directly on the architectural design of frontend applications. It emphasizes the importance of maintaining a clear hierarchy in building web interfaces, which largely aligns with the core logic of FSD. The authors highlight the need for implementing structural decomposition that enables functionality to scale without compromising the integrity of the system [8]. An interesting interdisciplinary example of structured architecture is demonstrated by the team of X. Kong et al., who developed the RFSD-YOLO model for detecting prohibited items in X-ray images. Although this system belongs to the field of computer vision, its name (Refined Feature Sliced Design) and structural principles reflect the adaptation of modularity and independent component processing concepts to machine learning tasks [9].

The study by N. Rašović in the field of additive manufacturing also demonstrates the application of multi-attribute analysis methods and architectural structuring for technical decision-making. Although not directly related to web development, the approach to parameter formalization and modular task structuring indicates a broader trend of transferring architectural thinking into adjacent technical domains [10].

S. Gao et al., in their article, describe the architecture of an inspection neural network with a dynamic feature extraction module and a task alignment mechanism. The structure of this model reflects an intention to isolate subtasks and build efficient, predominantly independent modules, which fully aligns with the compositional principles of the FSD approach [5].

Therefore, recent publications indicate a growing demand for architectural methodologies characterized by clear decomposition and control over inter-component dependencies. The Feature-Sliced Design methodology

is increasingly viewed as a promising standard for structuring frontend applications; however, its practical implementation in commercial environments requires further investigation.

Formulation of the Research Objective. The primary objective of this study is to comprehensively examine the architectural methodology known as Feature-Sliced Design (FSD) within the context of commercial frontend development, with a focus on its practical feasibility, advantages, and structural limitations. The central emphasis is placed on evaluating the potential of FSD to enhance scalability, modularity, and maintainability of web applications, taking into account the specifics of business logic, team management, and the project lifecycle.

The technological goals of the study are as follows: to analyze the architectural principles underlying FSD and compare them with traditional models (MVC, MVVM, Atomic Design); to model the structural framework of a frontend application using FSD stratification (app / processes / pages / widgets / features / entities / shared); to implement a demonstration project (ToDo Application) in Angular, incorporating the core principles of FSD to assess the effectiveness of the approach; to verify technical performance, structural flexibility, and scalability potential using business logic scenarios.

The practical goals of the study are as follows: to identify the conditions under which the application of Feature-Sliced Design is optimal in commercial environments; to assess the impact of the FSD methodology on reducing technical debt, accelerating the onboarding of new developers, and improving team collaboration quality; to develop recommendations for implementing FSD in small and medium-sized IT teams.

The expected technical outcomes of the study are as follows: structured documentation and a visual model of the frontend application architecture built using the FSD methodology; a functional prototype of an Angular application implemented in accordance with FSD stratification; an analytical conclusion regarding the advantages and limitations of FSD compared to alternative approaches.

The study holds significant importance for the advancement of architectural design practices in the field of frontend development. It enables the evaluation not only of the effectiveness of FSD as a methodology but also of its relevance to real-world business environments, making the findings valuable for teams seeking structural clarity, flexibility, and scalability of software interfaces. The innovative aspect of this work lies in the practical application of FSD principles within the Angular environment, followed by an analysis of its performance in a business context.

Presentation of the main research material. The Feature-Sliced Design (FSD) architectural methodology, which is formed at the intersection of structural engineering and a domain-oriented approach to frontend development, is gradually gaining popularity among teams working with Angular, React, and other modern frameworks. The main idea is to divide an application not by technical characteristics (e.g., components, services, or templates), but by functional scenarios and business logic, which allows you to maintain the scalability and manageability of the project [4].

The methodology is based on the principle of a multi-level hierarchy: the application is structured into layers (app, processes, pages, widgets, features, entities, shared), each of which performs its own autonomous role. Lower layers are unaware of the existence of higher layers – for example, shared components have no idea how they are used at the page or feature level. This approach ensures one-way encapsulation of dependencies, i.e., direct links are formed only down the structure, which reduces the number of non-obvious links and facilitates maintenance [3]. Technically, this means that a lower layer (e.g., shared) does not have access to layers above or next to it. Higher layers (e.g., entities, features, pages) can use everything below them, but not vice versa – similar to rivers that flow into the sea but do not flow in the opposite direction. This logic of relationships between FSD layers is reflected in (Tab. 1).

Table 1

Rules for interaction between layers in Feature-Sliced Design architecture

Layer	Can be used	Can be used
app	shared, entities, features, widgets, pages, processes	–
processes	shared, entities, features, widgets, pages	app
pages	shared, entities, features, widgets	processes, app
widgets	shared, entities, features	pages, processes, app
features	shared, entities	widgets, pages, processes, app
entities	shared	features, widgets, pages, processes, app
shared	–	entities, features, widgets, pages, processes, app

The methodology pays special attention to the so-called “removability” of modules: each functional block must be implemented in such a way that it can be completely removed without compromising the integrity of

the system. This is achieved through high component cohesion and minimal interdependence between them. By analogy with a water system, each fragment (slice) is an independent "river" that flows into the common "ocean" of the application logic, but does not intersect with other direct flows [4].

As part of the research, a prototype ToDo application was implemented on Angular, built according to FSD principles. The corresponding structure looked as follows:

```
bash
src/
├── app/          # глобальні налаштування маршрутизації та сховища
├── pages/        # окремі сторінки (tasks-list, task-details, not-found)
├── features/     # сценарії користувача (tasks-filter, toggle-task)
├── entities/     # бізнес-сутності (task-card, task-row)
├── shared/       # UI-бібліотеки, API, спільні сервіси
```

Each page contains only the composition of components, while the interaction logic is moved to separate "features." For example, interaction with the task list is implemented as a separate function tasks-filter, and the change in execution status is implemented in toggle-task. The essence of a task is formed as a data structure with its own visual components, implemented as task-card and task-row [3]. Figure 1 shows how the application structure is built according to the FSD methodology: from pages to shared components. Each layer is clearly separated, and the arrows illustrate the permissible direction of interaction between layers.

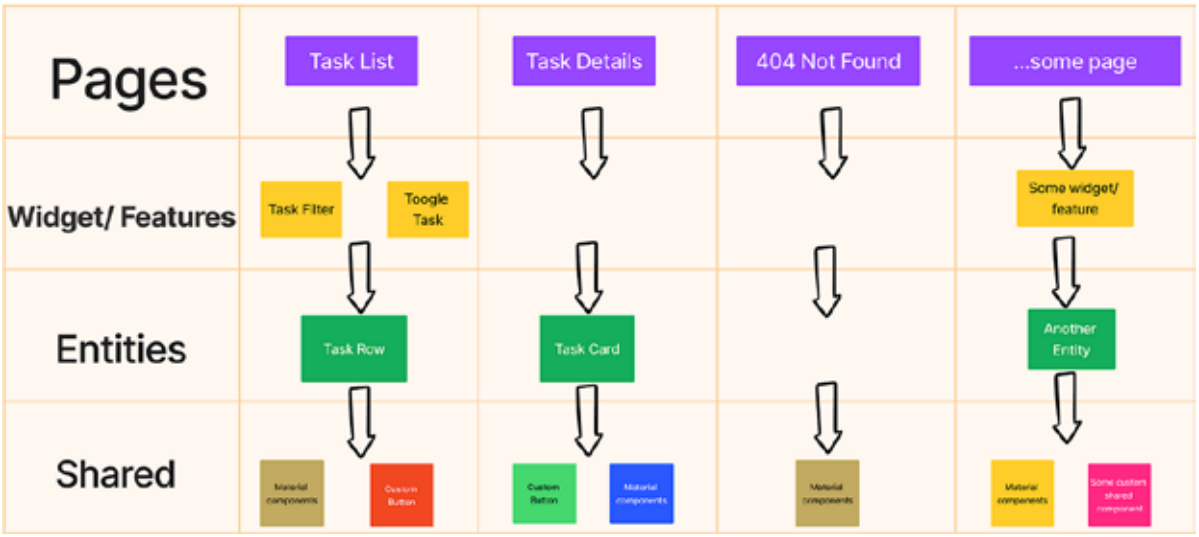

Fig. 1. Layer hierarchy in Feature-Sliced Design based on the ToDo application example

It is important to note that the implementation of the ToDo application provided an opportunity to empirically validate the key advantages of FSD. First, the project's structural organization facilitated the distribution of roles within the development team. Second, during the testing phase, a reduction in git merge conflicts was recorded, attributed to the separation of responsibilities across distinct project segments. Third, it became possible to easily remove or replace any functional block without disrupting the overall application logic, which is a critical factor for projects with frequent releases and evolving requirements [6, 7]. Thus, the application of Feature-Sliced Design in commercial frontend projects enhances the manageability of software architecture, although it requires the team to have a clear understanding of the paradigm and to maintain strict discipline in adhering to structural rules. While the methodology is not a universal solution, its benefits become particularly evident in medium- to large-scale development environments, where the cost of architectural debt is especially high.

Conclusions. The conducted study provided a theoretical and applied analysis of the Feature-Sliced Design (FSD) architectural methodology in the context of its application to commercial frontend projects. Based on the client's materials, the architecture of a ToDo application developed in Angular was reconstructed in accordance with FSD principles, which enabled the demonstration of real implementation mechanisms.

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

The main focus of the study was on the structural organization of the project through the stratification of logic: app, processes, pages, widgets, features, entities, and shared. Each layer was assigned specific functional responsibilities, permitted interdependencies (according to the rules of unidirectional encapsulation), and a defined role in ensuring modularity. Visual diagrams and integrated tables illustrate the direction of dependency flows and the logic underlying the construction of the interface layer of the product.

The research concluded that FSD ensures high structural flexibility, simplifies the distribution of responsibilities among developers, and facilitates the refactoring of individual code segments without compromising architectural integrity. Additional advantages of the methodology include reduced technical debt and logical isolation of functional modules. At the same time, the study identified several limitations, including the need for preliminary architectural planning, standardization of approaches within the team, and appropriate developer competencies.

The innovative contribution of this work lies in the analytical adaptation of the modern FSD methodology to a concrete application example, making the results applicable as a conceptual model for implementation in small and medium-sized projects. The resulting structure demonstrates potential for scalability and flexible development without compromising manageability.

Future research directions include the formalization of criteria for evaluating architectural efficiency in comparison with alternative approaches (such as MVVM or Atomic Design), as well as the extension of FSD methodology for use in multifunctional applications with complex business logic.

Bibliography:

1. Ben Khalfallah H. CRISP: Clean, Reliable, Integrated Software Process. *Crafting Clean Code with JavaScript and React*. Berkeley, CA : Apress, 2024. URL: https://doi.org/10.1007/979-8-8688-1004-6_6. (date of access: 08.06.2025).

2. Ben Khalfallah H. HOFA: The Path Toward Clean Architecture. *Crafting Clean Code with JavaScript and React: A Practical Guide to Sustainable Front-End Development*. Berkeley, CA : Apress, 2024. P. 233–287. URL: https://doi.org/10.1007/979-8-8688-1004-6_4. (date of access: 08.06.2025).

3. Feature-Sliced Design – modern Front End Architectural Methodology on Angular. *Medium*. URL: https://medium.com/@fed4wet/feature-sliced-design-modern-architectural-methodology-on-angular-d0ef705ef598. (date of access: 08.06.2025).

4. Feature-Sliced Design (FSD). URL: https://medium.com/@jstify.community/feature-slice-design-%D1%89%D0%BE-%D1%86%D0%B5-%D1%82%D0%B0%D0%BA%D0%B5-8cb86d059c16. (date of access: 08.06.2025).

5. Gao S., Xia T., Hong G., Zhu Y., Chen Z., Pan E., Xi L. An inspection network with dynamic feature extractor and task alignment head for steel surface defect. *Measurement* 2024. Vol. 224. P. 113957. URL: https://doi.org/10.1016/j.measurement.2023.113957. (date of access: 08.06.2025).

6. Goh H. A., Ho C. K., Abas F. S. Front-end deep learning web apps development and deployment: a review. *Applied Intelligence* 2023. Vol. 53, No. 12. P. 15923–15945. URL: https://doi.org/10.1007/s10489-022-04278-6. (date of access: 08.06.2025).

7. Hidayat D. C., Atmaja I. K. J., Sarasvananda I. B. G. Analysis and Comparison of Micro Frontend and Monolithic Architecture for Web Applications. *Jurnal Galaksi* 2024. Vol. 1, No. 2. P. 92–100. URL: https://doi.org/10.70103/galaksi.v1i2.19. (date of access: 08.06.2025).

8. Kolomoyets M., Kynash Y. Front-End web development project architecture design. *2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*. 2023. P. 1–5. URL: https://doi.org/10.1109/CSIT61576.2023.10324238. (date of access: 08.06.2025).

9. Kong X., Li A., Li W., Li Z., Zhang Y. RFSD-YOLO: An Enhanced X-Ray Object Detection Model for Prohibited Items. *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. October 2024. P. 4412–4418. URL: https://doi.org/10.1109/SMC54092.2024.10831942. (date of access: 08.06.2025).

10. Rašović N. Recommended layer thickness to the powder-based additive manufacturing using multi-attribute decision support. *International Journal of Computer Integrated Manufacturing* 2021. Vol. 34, No. 5. P. 455–469. URL: https://doi.org/10.1080/0951192X.2021.1891574. (date of access: 08.06.2025).

Дата надходження статті: 25.06.2025
Дата прийняття статті: 30.06.2025
Опубліковано: 23.09.2025