JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

FEATURE TOGGLES: THE EVOLUTION IN SOFTWARE DEVELOPMENT PROCESS AS SECRET WEAPON

[1]Hardik S Panchariya, [2]Hitesh V Hadole, [3]Prajwal A Patil, [4]Ritesh S Gaygawali, [5]Vinay A Rajgure

[1]Student, [2]Student, [3]Student, [4]Student, [5]Assistant Professor
Computer Science and Engineering,
Prof Ram Meghe College Of Engineering and Management Amravati, India

Abstract : Feature toggle management offers a dynamic way to handle various types of feature flags in a well-organized manner, aiding developers in the development process. Teams gain increased flexibility in deploying and testing features. Managing flags according to requirements is crucial, and the management system allows developer teams to view and control the state of each feature using toggle buttons, simplifying the management of multiple features through an intuitive GUI.
While our idea isn't entirely new, we conducted a review and researched existing feature flag management systems in the market. To innovate, we're implementing a feature that stores the history of each flag, tracking when it was used, which developer used it, and the duration the feature was turned on or off. In our project, we aim to introduce a dynamic interface for organizations to manage their flags. Additionally, our management tool will be accessible via API, allowing organizations to integrate the flag management tools directly into their code. They can use a generated API key to call the API, enabling easy access to toggle changes within the management tool through API calls.

IndexTerms **– Feature Toggle, Feature Toggle Management Tool, Continuous Delivery, Feature Flags, Software Development, Authentication and Authorization.**

I. INTRODUCTION

Before going in the deep about our feature toggle management tool we have firstly understand what is feature flag, why it is used and some more about them in most of the industry the feature flags is also know as feature toggles so don't be confuse. Now coming back to our main topic what is the Feature toggle it is a mechanism or simple conditional toggle that allow developer to turn a feature on or off in the running software. They are commonly used by large web companies including Google, Facebook and more enable and disable features with some conditional statement is used by the software practitioners using feature toggles may increase code complexity, decrease the down time of the running websites and also decrease the quality of codebase. Feature toggles added to the codebases to allow remote control of certain condition and logic at runtime. in the code the logic is already wrapped at some point it can be control by the simple feature toggle if the statue of the feature toggle in on the wrapped code condition is executed and if the status of the feature toggle is off then the wrapped logic is omitted. feature toggles are commonly used by the product developer for caring the A/B testing, authorization control in the application, the feature toggles have verity of used in application and infrastructure, in application the new features is roll out allowing teams to test on a small subset of the users before releasing them to everyone as this testing occur to mitigate the risk of the feature

How feature toggles work: feature toggles are typically added to codebases to allow the remote control of certain specific logic at the runtime in code as the logic is wrapped so that it can be controlled by the status of a feature toggle. if the feature toggle status is on then the wrapped logic is executed if the feature toggle status in off then the wrapped code is skipped. The returned status when an application wants to know the status of the feature toggle at runtime it makes a request to an external data source or managed service. the application can then decide whether or not to executed wrapped logic because the application gets the status of feature toggles from a remote source it allows wrapped logic to be controlled remotely, simply by toggling the flag on or off in the configuration file without requiring code deployment, if the status of a feature toggle is changed it also changed in the application at runtime. As many feature toggles that control different parts of the application so the feature toggles are assigned unique names or keys to be differentiable and descriptive of the logic that they control this key we use as one of the major attribute in our management tool to diffract the flags.
benefits of the feature toggles: the benefit of the feature toggle is that it mitigates the risk depending with the releasing in the changes to application. a new feature release or a small refactor there is always the inherent risk of releasing new regression. to mitigate this, changes to an application can be placed behind feature toggles, allowing them to be turned on or off in the event of an emergency. In many cases the practices allow teams to catch regressions early and roll back if necessary, mitigating the risk of bugs at runtime

Types of Feature Toggles: Depending on how long feature Toggles should remain on or off in the code and what is there purpose, they can be divided into the 4 categories
1. Release Toggle
2. Experiment Toggle
3. Ops Toggle
4. Permission Toggle
Let's know in brief about each of them
1. Release Toggle
Release toggles allow code deployment of untested and incomplete code to the production environment without activation of the code. this allows product managers, to release new features as needed
2. Experiment Toggle
Experiment toggles can be used to control user-specific behaviour within an application to this end, users are identified within the software system so that they can then be specifically controlled. In this way, new function of feature flag can be quickly tested on specific groups of people so that final conclusions can be drawn for further development
3. Ops Toggle
Ops toggles allow operations teams to control application operations via feature toggles, This enables quick rollback of features and testing the workload of new features on small groups. In addition, the Ops toggle allows functions that are not required to be disabled during operation and are known as kill switches
4. Permission Toggle
Permission toggles can be used to control the specific user behavior of known users within an application. you can use these toggles to enable functions for premium users. A/B tests are another possible application

There are three methods to implement feature toggles:
1. Direct Implementation in Code: Implementing feature toggles directly in the code involves incorporating conditional statements or flags within the codebase. This practice allows developers to control the activation or deactivation of specific features in a software application.
2. Integration through a Configuration File: Integration via a configuration file entails designing a software application to read and utilize configuration settings from an external file, commonly known as a configuration file or config file. This file, which can be in formats like JSON, YAML, XML, or a simple key-value pair format config.json or application. properties, contains parameters and settings configuring the behavior of the application.
External Configuration File: The application is programmed to read configuration settings from a designated external file, typically in JSON, YAML, XML, or key-value pair format (e.g., config.json or application.properties).
Configuration Loading: During application startup or runtime, the application reads the configuration file to load various settings, including parameters like database connection details, API endpoints, feature toggles, logging levels, and more.
Separation of Concerns: Utilizing a configuration file allows developers to separate configuration concerns from the application's logic. This separation facilitates easier maintenance, as changes to configuration parameters can be made without modifying the application code.
Dynamic Behavior: As the configuration file is external to the code, changes to configuration settings do not necessitate recompilation or redeployment of the application. This flexibility makes it easier to adapt the application to different environments or update settings without affecting the core.
3. Integration through Feature Flag Management Software: Integration via feature flag management software involves incorporating a specialized tool or service into a software application to handle the implementation and control of feature flags. Feature flag management tools provide a centralized platform for managing and controlling the activation or deactivation of specific features in an application. These tools often come with additional capabilities for fine-tuning feature rollout strategies, monitoring, and experimentation.

Feature Flag Management Software (FFMS) refers to a specialized tool or service created to assist developers in effectively managing feature flags. These tools typically provide a user interface or API for creating, updating, and controlling feature flags, often offering additional features to streamline the management of feature releases. Centralized Control is a key aspect of FFMS, as it consolidates the management of feature flags. This centralization allows developers and product managers to toggle features, set rollout percentages, and manage configurations through a web interface or API. This centralized approach is crucial for maintaining control and visibility over feature releases. Rollout Strategies are supported by FFMS, including percentage-based rollouts, targeting specific user segments, or gradually releasing a feature to different environments. These strategies facilitate controlled and phased deployments of new features, contributing to a smoother release process. Experimentation and A/B Testing capabilities are often provided by some FFMS. Developers can define different variations of a feature and measure their impact on user behavior or system performance. This functionality enhances the ability to experiment with features and make data-driven decisions. Monitoring and Analytics integration with FFMS allows teams to collect data on feature usage, performance, and user engagement. This data becomes instrumental in making informed decisions about feature releases and adjustments, enhancing the overall understanding of user interactions. The integration of FFMS into the development process provides teams with greater flexibility in deploying and testing features. It enables controlled releases, quick rollbacks in case of issues, and offers insights into user interactions with different feature variations. Popular FFMS tools include Launch Darkly, Splittail, Rollout, and others.
Feature Toggle Use Cases: 1. Feature Toggles and Continuous Delivery: Continuous Integration and Continuous Delivery (CI/CD) is a software development discipline championed by Martin Fowler. This approach allows software to be released to production at any time, emphasizing quick delivery of stable and bug-free software. 2. A/B Testing: A/B testing involves experimenting with new ideas and confirming hypotheses using real-world data. It segments traffic into two variations of a feature, capturing metrics for each variation side-by-side. Feature toggles can facilitate A/B testing by controlling the variation

in which each user is placed. If a feature toggle is off for a user, they see the A variation; if it is on, they see the B variation. 3. Optimizely and Feature Toggles: Optimizely, a leader in Progressive Delivery and Experimentation, offers various solutions for feature flag management. Their enterprise feature flagging and experimentation suite, Optimizely server-side experimentation, enables development teams to manage features and run tests on any internet-connected device. SDKs are available for popular languages and frameworks, including Node, Python, Ruby, Go, React, Swift, and C#.

A feature toggle management tool is a software solution specifically designed to assist development teams in implementing, controlling, and managing feature toggles within their applications. These tools play a crucial role in contemporary software development practices, providing teams with enhanced control over feature releases, minimizing risks associated with deployments, and facilitating a more iterative and adaptive development process. Feature toggles, which are conditional statements in the code, empower developers to manage the enabling and disabling of specific features without necessitating modifications to the underlying codebase. Feature toggle management tools offer a centralized platform to manage these toggles, simplifying the process of controlling feature releases, conducting experiments, and managing configurations. The tools typically provide a centralized interface or API, allowing developers and team members to easily create, modify, and control feature toggles. This centralized approach aids in maintaining visibility and control over feature releases. Many feature toggle management tools feature a user-friendly web interface, enabling users to toggle features, set rollout percentages, and configure various aspects of feature behavior without requiring code changes. These tools often support various rollout strategies, including percentage-based rollouts, targeted releases to specific user segments, or gradual feature releases to different environments. These strategies enable controlled and phased deployments of new features. Some tools go beyond basic feature toggle management and offer capabilities for conducting experiments and A/B testing. Developers can define different variations of a feature and measure their impact on user behavior or system performance. Integration with monitoring and analytics tools allows teams to gather valuable data on feature usage, performance, and user engagement. This data serves as a foundation for making informed decisions about feature releases and necessary adjustments. Feature toggle management tools frequently integrate seamlessly with Continuous Integration/Continuous Deployment (CI/CD) pipelines. This integration allows for automated feature releases, ensuring that toggles are appropriately managed throughout the entire development lifecycle. Additionally, these tools may include features for securing access to feature toggles, ensuring that only authorized team members can modify or control specific features.

A feature toggle management tool using an API key necessitates an understanding of what an API is, its benefits, and its functionality. API, an acronym for Application Programming Interface, comprises a set of rules and protocols enabling one software application to interact with another. In contemporary software development, APIs are foundational, fostering the integration of diverse services, applications, and platforms. They play a pivotal role in the development of web applications, mobile apps, and various software systems reliant on interconnected components. APIs define methods and data formats for applications to communicate seamlessly, promoting interoperability and allowing developers to integrate functionalities from diverse sources. APIs facilitate collaboration among different software systems by providing a standardized means of communication, simplifying the development process, and promoting modularity through a layer of abstraction. Operating on a request-response model, APIs involve a client application sending a request to a server application, which responds with the requested information or performs the desired action. Standardized data formats, such as JSON and XML, make it easy for different systems to understand and process data. APIs often expose specific endpoints representing various functionalities or resources, each corresponding to a specific URL or URI.
Authentication and authorization mechanisms are crucial components of APIs to ensure that only authorized users or applications access specific functionalities or data, enhancing security. Two common architectural styles for API design are Representational State Transfer (REST) and Simple Object Access Protocol (SOAP). RESTful APIs emphasize simplicity and scalability, while SOAP APIs adhere to a more rigid communication protocol. In the context of a feature toggle management tool using an API key, this refers to a system or service enabling developers to control and manage feature flags or toggles through an API. An API key serves as the authentication and authorization mechanism, ensuring that only authorized entities can modify feature toggles through the API. This specialized service provides a platform for creating, controlling, and managing feature toggles in an application, allowing developers to control feature activation or deactivation without modifying the underlying codebase. Examples of such tools include LaunchDarkly, Split.io, and Rollout.io. The feature toggle management tool exposes an API that developers can interact with programmatically. This API facilitates querying the status of feature toggles, updating toggle configurations, and performing other operations related to feature management. To enhance security and control access, the feature toggle management tool typically requires an API key for authentication. Developers or applications wishing to interact with the API must include a valid API key in their requests, commonly in headers or request parameters. This approach is prevalent in modern software development, especially when managing feature flags dynamically, conducting A/B testing, or gradually rolling out features using a centralized management system.

II. LITERATURE REVIEW

Table 2.1: Literature Review

Paper Title and Authors	Key Focus and Contributions	Significance
[1] Feature Toggles: Practitioner Practices and A Case Study (Rahman et al., 2016)	Empirical insights into industry practices related to feature toggles, challenges in real-world software development, impact on software evolution and deployment strategies	Practical implications for implementing toggles in industry settings
[2] The Modular and Feature Toggle Architectures of Google Chrome (Rahman et al., 2018)	Analysis of feature toggles in Google Chrome's architecture, insights into toggles managing complexity and variability in a large-scale system	Understanding toggles in a prominent software system like Google Chrome
[3] Software Development With Feature Toggles: Practices Used By Practitioners (Mahdavi-Hezaveh et al., 2019)	Exploration of practitioner practices in implementing feature toggles, addressing common strategies, challenges, and trade-offs	Insights into real-world usage and challenges in toggles' implementation
[4] Feature Development in BPMN-Based Process-Driven Applications (Schneid et al., 2020)	Investigating feature development in BPMN-based process-driven applications without specific insights provided in the query	Not specific information available for a detailed review
[5] Capture The Feature Flag: Detecting Feature Flags in Open-Source (Meinicke et al., 2020)	Introduction of methods to detect feature flags in open-source projects, implications on software evolution and maintenance	Understanding prevalence and implications of toggles in open-source projects
[6] How The Adoption of Feature Toggles Correlates with Branch Merges and Defects in Open-source Projects? (Prutchi et al., 2020)	Correlation analysis between feature toggle adoption, branch merges, and defects in open-source projects	Understanding the relationship between toggle adoption and project health in open-source
[7] How to Integrate with Real Cars - Minimizing Lead Time at Volkswagen (Kantert & Nolting, 2021)	Focus on integration strategies at Volkswagen without specific insights provided in the query	Specific insights about feature toggles may not be covered
[8] On The Interaction of Feature Toggles (Tërnava et al., 2022)	Investigation of feature toggles' interactions, possibly addressing how toggles interact within software systems	Understanding complexities in toggles' interactions within systems
[9] From Feature Models to Feature Toggles in Practice (Jézéquel et al., 2022)	Transition path from feature models to practical implementation of feature toggles, addressing challenges and opportunities	Insights into the process of implementing toggles from feature models
[10] An Extended Model of Software Configuration (Mahdavi-Hezaveh et al., 2022)	Extension of a software configuration model without specific insights provided in the query	Specific insights about feature toggles may not be covered

Potential gaps in the existing research: Practitioner Adoption: Long-term impact: While studies explore initial motivations and challenges, there's a lack of research on the long-term impact of feature toggles on team dynamics, code maintainability, and overall project success. Specific industry/domain analysis: Existing research often focuses on general software development. Studies on feature toggle adoption in specific industries or domains (e.g., healthcare, fintech) could reveal unique challenges and best practices. Quantitative data on benefits: More quantitative data is needed to measure the actual benefits of feature toggles in terms of reduced risks, faster releases, or improved user experience

Technical Aspects: Feature toggle testing and debugging: Limited research exists on effective testing and debugging strategies for code with feature toggles, especially in complex systems with multiple interacting toggles. Feature toggle security and authorization: Security implications of feature toggles, such as unauthorized access or manipulation, need further exploration and development of robust security mechanisms. Scalability and performance of feature management systems: As codebases and feature sets grow, research is needed on scalable and performant feature management systems that can handle large numbers of toggles efficiently

Development Process Impact: Feature toggle usage in Agile methodologies: More research is needed on integrating feature toggles seamlessly into Agile development processes, optimizing release cycles, and managing dependencies between toggles and user stories. Impact on team communication and collaboration: Studies are needed to understand how feature toggles affect team communication, collaboration, and knowledge sharing across development and product teams. Metrics for measuring toggle effectiveness: Establishing metrics to track the effectiveness of feature toggles in achieving desired outcomes (e.g., reducing rollbacks, improving user adoption) would be valuable for practitioners

Theoretical Implications: Formal models for feature toggle interactions: Developing formal models to represent and analyse the complex interactions between multiple feature toggles could provide valuable insights into potential risks and control strategies. Feature toggles and software evolution: Research is needed on how feature toggles impact software evolution, including version control, refactoring, and long-term maintenance of code with toggles. Relationship between feature toggles and architectural patterns: Exploring the relationship between feature toggles and different software architectural patterns could lead to new design principles and best practices for building maintainable and flexible systems

Building Feature Management Software from Scratch: Comparison of existing open-source solutions: A comprehensive comparison of existing open-source feature management tools, evaluating their strengths, weaknesses, and suitability for different use cases, could be helpful for practitioners. Trade-offs in custom vs. off-the-shelf solutions: Research on the trade-offs between building a custom feature management system versus using existing tools could guide organizations in making informed decisions based on their specific needs and resources. Emerging technologies and their impact on feature management: Exploring how emerging technologies like AI, cloud computing, and blockchain can be leveraged to develop next-generation feature management systems with advanced capabilities

These are just some potential gaps based on your areas of interest. By focusing on these or other underexplored areas, you can contribute valuable new insights to the field of feature toggle research and development.

III. Proposed Methodology

3.1 OptiToggle - Feature Toggle Management Tool:

OptiToggle a feature toggle/flags management tool: The tool offers a robust set of functionalities for seamless feature toggle management, accessible both as a microservice and through a user-friendly web UI. Key features include user registration, login capabilities, and role-based access control for administrators and developers. Optitoggle leverages JWT authentication for secure user access and authorization. Feature toggle or flags management tools play a pivotal role in enabling continuous delivery and experimentation within software development. This paper focuses on Optitoggle, highlighting its methodology, user roles, and authentication techniques.

Fig 1.0 : Basic Architecture

3.1.1 Optionaly you can use Toggle Manager UI to interact with system

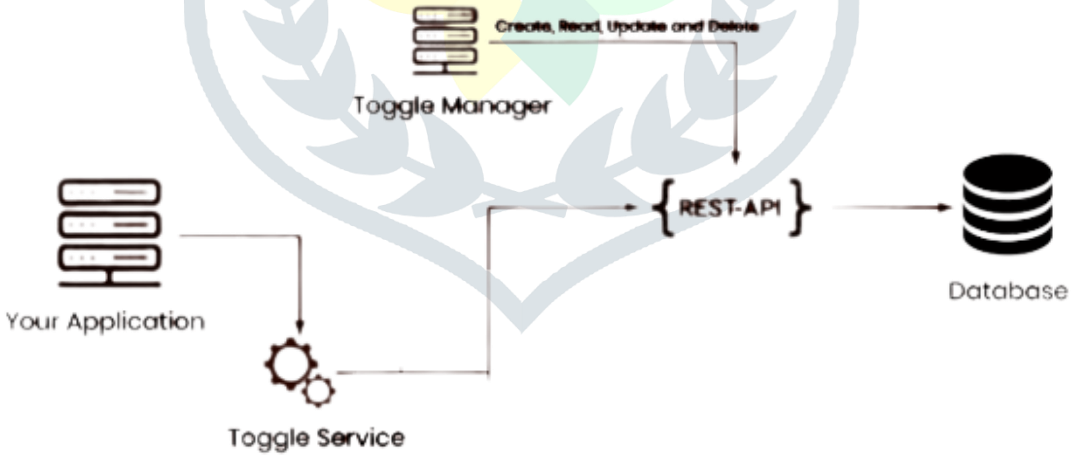

Fig 1.1: Basic Architecture

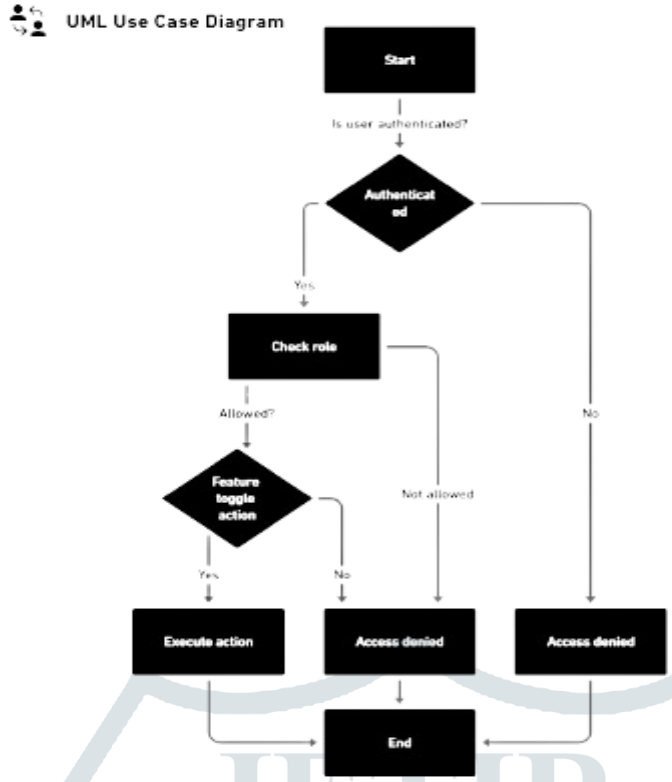

Fig 2: Use Case Diagram

3.2 User Roles: Optitoggle categorizes users into two distinct roles - administrators and developers. The roles define the permissions and functionalities available to each user category.
3.2.1. Administrator Role: Administrators hold the authority to create, update, retrieve, and delete feature flags. This role is crucial for overall system governance, ensuring the correct and secure management of feature toggles.
3.2.2. Developer Role: Developers are empowered with specific capabilities tailored to their responsibilities. They can update feature flags, specifically toggling them on or off, and retrieve information about all flags or a specific flag.

3.3 Authentication and Authorization: Optitoggle employs JSON Web Token (JWT) authentication, a widely adopted and secure technique in the industry. This ensures that only authenticated users with valid tokens can access the system.
3.3.1. JWT Authentication: JWT authentication involves the generation of tokens upon successful user login. These tokens contain information about the user and their assigned role. The tokens are then included in subsequent requests to the Optitoggle microservice, allowing the system to verify the user's identity and authorize access based on their role.
3.3.2. Role-Based Authorization: Optitoggle's role-based access control ensures that each user is granted access only to the functionalities relevant to their role. This minimizes the risk of unauthorized actions and enhances the overall security of the system.

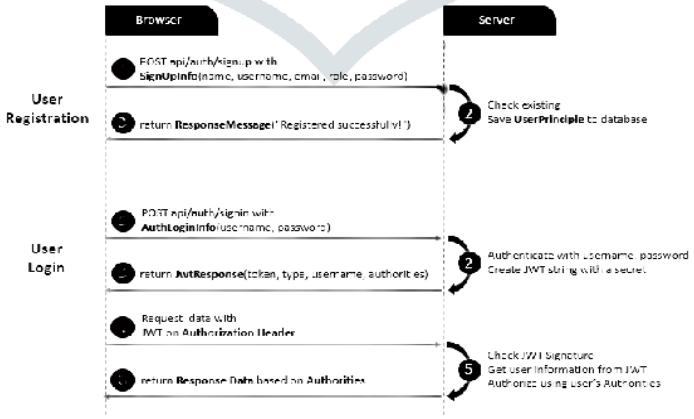

Fig 3: Authentication Flow

3.4 Feature Management Functionalities: Optitoggle provides a comprehensive set of feature management functionalities, allowing administrators and developers to perform tasks efficiently.

3.4.1. Creating and Updating Flags: Administrators have the privilege to create new feature flags and update their properties. Developers, on the other hand, can update flags but are limited to toggling them on or off.
3.4.2. Retrieving Flags: Both administrators and developers can retrieve information about all flags or specific flags. This feature facilitates transparency and aids in decision-making during the development lifecycle.
3.4.3. Deleting Flags: Administrators hold the authority to delete feature flags, ensuring a clean and organized feature management environment.

3.5 Implementation: OptiToggle provides a user-friendly interface for managing feature toggles. It allows developers to toggle features on or off, set rollout percentages, and track toggle usage.

3.6 Benefits: Centralized feature toggle management. Real-time updates without code changes or application restarts. Granular control over feature rollout.

3.7 Drawbacks: Dependency on OptiToggle tool. Potential cost associated with using the tool (if it's a paid service).

IV. PROPOSE TECHNIQUES

Feature toggles, also known as feature flags or feature switches, are a powerful technique used in software development to enable or disable features in an application at runtime. They provide flexibility and control over the release process, allowing developers to manage feature releases without changing code. Here are different ways feature toggles can be implemented, along with their benefits and drawbacks.

4.1 Conditional Statements:

1. Implementation: Developers can use conditional statements (if-else) to wrap the code related to a particular feature. The condition is based on the state of the feature toggle.
2. Benefits:
   - Simple and straightforward implementation.
   - No need for external dependencies.
3. Drawbacks:
   - Code can become cluttered with conditional statements.
   - Manual cleanup may be required when the feature is permanently enabled or disabled.
4. Examples:-

If-else Statements:-

```
if (OptiToggle.isFeatureEnabled("NewFeature")) {
// Code for the new feature
}
```

Feature Methods/Functions:

```
FeatureManager.executeFeature("NewFeature");
```

4.2 Configuration Files:

1. Implementation: Feature toggle states are stored in configuration files (e.g., JSON, YAML), and the application reads these files to determine the state of each feature.
2. Benefits:
   - Easy to manage toggles without changing the code.
   - Centralized control over feature states.
3. Drawbacks:
   - Requires application restart to apply changes.
   - Can become unwieldy with a large number of toggles.
4. Examples:

External Configuration Files:

```
{
    "features": {
    "NewFeature": true
}}
```

4.3 Database Flags:

1. Implementation: Feature toggle states are stored in a database, and the application queries the database to determine the state of each feature.
2. Benefits:
   - Dynamic control over feature states without requiring code changes.
   - Suitable for applications with frequent feature state changes.
3. Drawbacks:
   - Adds database queries overhead.
   - Potential latency if the database is not optimized.
4. Examples:

```
SELECT is Enabled FROM FeatureToggles WHERE featureName = 'NewFeature';
```

4.5 Environment Variables:

1. Implementation: Feature toggle states are stored as environment variables, and the application reads these variables to determine the state of each feature.
2. Benefits:
   - Quick and easy to update toggle states.
   - No need for additional storage mechanisms.
3. Drawbacks:
   - Limited control over dynamic changes.
   - May require application restart to apply changes.
4. Examples:

   export FEATURE_NEW=true

4.6 Feature Toggle Libraries/Frameworks:

1. Implementation: Leveraging specialized libraries or frameworks (like OptiToggle) that provide a dedicated interface for managing feature toggles.
2. Benefits:
   - Centralized management through a user interface.
   - Real-time updates without application restart.
3. Drawbacks:
   - Dependency on the third-party library or tool.
   - Learning curve for integrating and using the specific tool.

IV. Conclusion

In summary, feature flag management proves to be a valuable resource for developers, offering a dynamic and well-organized approach to handling various types of feature flags. This system enhances flexibility in deploying and testing features, thereby streamlining the development process for teams. Although the core concept is not entirely novel, our unique approach involves a thorough review of existing systems, leading to the implementation of distinct features. This innovative addition enables the storage of historical data for each flag, encompassing usage details, responsible developers, and duration information.

The primary objective of our project is to introduce a dynamic interface that simplifies flag management for organizations. The user-friendly GUI, equipped with toggle buttons, empowers developer teams to easily view and control the state of each feature. Going beyond traditional feature flag management, our tool offers accessibility through an API. This integration enables organizations to seamlessly incorporate flag management directly into their code, utilizing a generated API key for convenient and efficient toggle changes through API calls. In essence, our feature flag management solution not only addresses existing needs but also introduces innovative features, establishing it as a versatile and potent tool for organizations seeking efficient and flexible flag management in their development processes.

REFERENCES

[1] Md Tajmilur Rahman; Louis-Philippe Querel; Peter C. Rigby; Bram Adams; "Feature Toggles: Practitioner Practices and A Case Study", 2016 IEEE/ACM 13TH WORKING CONFERENCE ON MINING SOFTWARE ..., 2016.

[2] Md Tajmilur Rahman; Peter C. Rigby; Emad Shihab; "The Modular and Feature Toggle Architectures of Google Chrome", EMPIRICAL SOFTWARE ENGINEERING, 2018.

[3] Rezvan Mahdavi-Hezaveh; Jacob Dremann; Laurie Williams; "Software Development With Feature Toggles: Practices Used By Practitioners", ARXIV-CS.SE, 2019.

[4] Konrad Schneid; Sebastian Thöne; Herbert Kuchen; "Feature Development in BPMN-Based Process-Driven Applications", 2020.

[5] Jens Meinicke; Juan Hoyos; Bogdan Vasilescu; Christian Kästner; "Capture The Feature Flag: Detecting Feature Flags in Open-Source", PROCEEDINGS OF THE 17TH INTERNATIONAL CONFERENCE ON MINING ..., 2020.

[6] Eduardo Smil Prutchi; H. D. S. C. Junior; Leonardo Gresta Paulino Murta; "How The Adoption of Feature Toggles Correlates with Branch Merges and Defects in Open-source Projects?", SOFTWARE: PRACTICE AND EXPERIENCE, 2020.

[7] Jan Kantert; Michael Nolting; "How to Integrate with Real Cars - Minimizing Lead Time at Volkswagen", 2021 IEEE/ACM 43RD INTERNATIONAL CONFERENCE ON SOFTWARE ..., 2021.

[8] Xhevahire Tërnava; Luc Lesoil; Georges Aaron Randrianaina; D. Khelladi; M. Acher; "On The Interaction of Feature Toggles", PROCEEDINGS OF THE 16TH INTERNATIONAL WORKING CONFERENCE ON ..., 2022.

[9] J. Jézéquel; J. Kienzle; M. Acher; "From Feature Models to Feature Toggles in Practice", PROCEEDINGS OF THE 26TH ACM INTERNATIONAL SYSTEMS AND ..., 2022.

[10] Rezvan Mahdavi-Hezaveh; Sameeha Fatima; Laurie Williams; "An Extended Model of Software Configuration", ARXIV-CS.SE, 2022.