**PROJECT IMPROVEMENT OF SIMPLE CALCULATOR USING C# IN ASP.NET MVC APPLICATIONS**

Computer Science Elective 3

Full Stack Development

BY:

**Gagarin, Maurice Gabriel P.**

(Gagarin-Maurice-BSCS32E1)

2024

**INTRODUCTION**

In today’s technology, the demand for efficient and user-friendly applications continues to evolve. One such fundamental tool is the calculator, which has been a staple in both personal and professional spheres for decades. However, with the advancements in web development frameworks like ASP.NET MVC and programming languages like C#, there lies an opportunity to enhance the capabilities of traditional calculators and offer users a more intuitive and feature-rich experience.

The Improvement of Simple Calculator using C# in ASP.NET MVC Applications project aims to enhance the standard calculator functionality by the use of C# within the ASP.NET MVC framework. This project seeks to enhance the user experience by implementing additional features such as design, improving usability, and optimizing performance.

Moreover, performance optimization techniques are employed to ensure smooth operation even under heavy usage. Customization options cater to diverse user preferences, while rigorous testing methodologies guarantee high standards of quality and reliability. Ultimately, this project seeks to deliver a modern, feature-rich calculator application that meets the evolving needs of users in today's digital landscape.

**Current Process or Workflows**

The current process for software development of the Simple Calculator involves several stages:

1. **Requirements Gathering:**

* Initially, the requirements for each application are gathered from stakeholders, including functionality, features, and design preferences.
* This stage involves discussions, meetings, and documentation to ensure a clear understanding of the project scope.

1. **Design Phase:**

* Once requirements are finalized, the design phase begins, where wireframes and mockups are created to visualize the user interface and overall layout.
* Design decisions are made regarding color schemes, typography, navigation, and other visual elements.

1. **Development:**

* The development team starts coding the applications based on the approved designs and requirements.
* This phase involves writing code, implementing features, and ensuring functionality according to specifications.

1. **Testing:**

* After development, the applications undergo testing to identify and rectify any bugs, errors, or inconsistencies.
* Various testing methods, such as unit testing, integration testing, and user acceptance testing, are employed to ensure the quality and reliability of the software.

1. **Deployment:**

* Once testing is completed and issues are resolved, the applications are deployed to production environments for public use.
* Deployment involves ensuring compatibility across different platforms and devices.

1. **Maintenance and Updates:**

* After deployment, the applications require ongoing maintenance and updates to address user feedback, fix bugs, and implement new features.
* Regular updates are essential to keep the software relevant, secure, and optimized for performance.

**Bottlenecks, Inefficiencies, and Areas for Improvement**

1. **Communication Breakdowns:**

One bottleneck is communication breakdowns between developers and designers during the requirements gathering and design phases. Clear communication channels and regular updates can mitigate this issue.

1. **Lengthy Development Cycles:**

Development cycles may be prolonged due to inefficient coding practices, unclear requirements, or scope creeps. Streamlining development processes and conducting regular progress reviews can help expedite development.

1. **Testing Delays:**

Testing may experience delays due to inadequate testing resources, lack of automation, or incomplete test coverage. Implementing automated testing frameworks and involving testers early in the development process can accelerate testing.

1. **Deployment Complexity:**

Deployment processes may be complex and error-prone, leading to delays and downtime. Implementing continuous integration/continuous deployment (CI/CD) pipelines and standardized deployment procedures can streamline deployment and reduce risks.

1. **Limited User Feedback Loop:**

The feedback loop from users may be limited, hindering the identification of usability issues or desired features. Implementing feedback mechanisms such as user surveys, analytics tools, and beta testing programs can facilitate continuous improvement based on user input.

1. **Maintenance Challenges:**

Maintenance tasks may become overwhelming due to a lack of prioritization or resource allocation. Prioritizing maintenance tasks based on impact and urgency, and allocating dedicated resources for ongoing support and updates can alleviate maintenance challenges.

**PROPOSED IMPROVEMENT**

1. **Improved Communication Channels**:

* Implement a centralized project management platform such as Github to facilitate transparent communication and collaboration among developers and designers.
* Conduct regular meetings, status updates, and reviews to ensure alignment on project goals, requirements, and progress.
* Establish clear communication channels for feedback and clarification, such as email, instant messaging platforms, or dedicated discussion forums.

1. **Agile Development Methodology**:

* Adopt an Agile development approach, such as Kanban, to break down development cycles into smaller, manageable iterations (sprints).
* Prioritize requirements and features based on business value and stakeholder feedback, allowing for incremental delivery and flexibility in adapting to changing needs.
* Conduct regular sprint planning, daily stand-up meetings, and sprint reviews to track progress, address issues, and adjust priorities as needed.

1. **User Feedback Mechanisms**:

* Introduce feedback mechanisms within the applications, such as in-app feedback forms, rating prompts, or user surveys, to gather insights directly from users.
* Establish a dedicated feedback channel (e.g., a support email address or community forum) for users to submit feature requests, bug reports, and suggestions.

1. **Prioritized Maintenance Tasks:**

* Implement a prioritization framework (such as the MoSCoW method or the Eisenhower Matrix) to categorize maintenance tasks based on urgency, impact, and effort required.
* Allocate dedicated resources, such as a maintenance team or support staff, to address critical issues promptly and proactively.
* Establish service level agreements (SLAs) for response times and resolution times to ensure timely resolution of maintenance requests.

How the Solution Improves the Process and Achieves Desired Outcomes:

1. **Enhanced Collaboration and Alignment**: Improved communication channels foster collaboration and alignment among stakeholders, leading to clearer requirements, faster decision-making, and reduced misunderstandings.
2. **Increased Agility and Adaptability**: Adopting Agile methodologies enables the team to respond quickly to changing requirements, deliver value iteratively, and maintain flexibility throughout the development process.
3. **Informed Decision-Making**: User feedback mechanisms provide valuable insights into user preferences, pain points, and needs, enabling informed decision-making and prioritization of features and improvements.
4. **Efficient Maintenance and Support**: Prioritizing maintenance tasks and allocating dedicated resources improve the responsiveness and efficiency of maintenance and support activities, enhancing user satisfaction and system reliability.

Considered Approaches and Selection Rationale:

* While there are various approaches to address the identified issues, the proposed solution leverages best practices in software development, project management, and quality assurance to comprehensively improve the entire software development lifecycle.
* The selected solution prioritizes collaboration, agility, automation, and user-centricity, aligning with modern industry standards and proven methodologies for delivering high-quality software efficiently and effectively.
* Each component of the proposed solution addresses specific pain points and inefficiencies identified in the current process, ensuring a holistic approach to process improvement and organizational success.

**BENEFITS AND IMPACT**

1. **Enhanced Product Quality and Reliability**:

Continuous testing and automated code reviews ensure consistent code quality, reducing the likelihood of defects and enhancing overall product reliability.

1. **Increased User Satisfaction and Retention**:

Implementation of user feedback mechanisms and iterative delivery of features based on user input result in a more user-centric product experience.

1. **Aligned with Organizational Goals and Priorities**:

The proposed improvement aligns with the organization's goals of delivering high-quality software efficiently, maximizing customer satisfaction, and staying ahead of competitors.

**PROJECT PLAN AND TIMELINE**

Phase 1: Planning and Preparation

* Milestone 1: Project Kickoff Meeting
* Date: March 8, 2024
* Objective: Align stakeholders, define project scope, and establish goals.
* Milestone 2: Requirements Gathering
* Date: March 9-11, 2024
* Objective: Collect detailed requirements for each application from stakeholders.
* Milestone 3: Resource Allocation
* Date: March 12-13, 2024
* Objective: Assign roles and responsibilities to team members, ensuring availability of necessary resources.

Phase 2: Design and Development

* Milestone 4: Design Phase Completion
* Date: March 14-23. 2024
* Objective: Complete the design phase, including wireframes and mockups for each application.
* Milestone 5: Development Commencement
* Date: March 18-23, 2024
* Objective: Begin development of the Simple Calculator
* Milestone 6: Initial Development Progress Review
* Date: March 23, 2024
* Objective: Review initial development progress, ensuring alignment with requirements and design specifications.

Phase 3: Testing and Quality Assurance

* Milestone 7: Test Automation Implementation
* Date: March 25, 2024
* Objective: Implement automated testing frameworks and establish test cases for each application.
* Milestone 8: Integration Testing
* Date: March 26-30, 2024
* Objective: Conduct integration testing to ensure seamless functionality across different components.

Phase 4: Deployment and Launch

* Milestone 10: Deployment Preparation
* Date: April 1 – 20, 2024
* Objective: Prepare deployment environments and finalize deployment configurations.
* Milestone 11: Deployment and Launch
* Date: April 22 – May 4, 2024
* Objective: Deploy the Simple Calculator, Simple Rock-Paper-Scissors Game, and Blog Engine to production environments for public use.
* Milestone 12: Post-launch Review
* Date: May 6-8, 2024
* Objective: Conduct a post-launch review to evaluate the success of the deployment and address any issues or feedback.

Phase 5: Maintenance and Support

* Milestone 13: Maintenance Plan Implementation
* Date: May 10 – 25, 2024
* Objective: Implement a maintenance plan, including procedures for addressing bugs, updates, and user support.
* Milestone 14: Ongoing Support
* Date: May 25-31, 2024
* Objective: Provide ongoing support and maintenance for the applications, addressing user feedback and implementing incremental updates.
* Resource Allocation:
* Development Team: Software developers, designers, and QA engineers.
* Project Manager: Responsible for overall project coordination, scheduling, and communication.

**EVALUATION AND RISK ASSESSMENT**

**Evaluation Assessment**

**User Satisfaction**:

1. **Metric**: User satisfaction surveys and feedback ratings collected from users of the Simple Calculator.
2. **Method**: Implement in-app feedback mechanisms and post-launch surveys to gather qualitative and quantitative feedback from users.
3. **Analysis**: Analyze survey responses and feedback ratings to assess overall user satisfaction levels and identify areas for improvement.

**Feedback and Iterative Improvement**:

1. **Metric**: Number of feature requests implemented and response time for addressing user feedback.
2. **Method**: Maintain a feedback log and track the implementation of user-requested features and improvements.
3. **Analysis**: Evaluate the frequency and effectiveness of incorporating user feedback into product updates and enhancements.

**Risk Assessment**

**Technical Risks**:

1. **Risk:** Integration challenges between different components of the applications.
2. **Mitigation:** Conduct thorough integration testing throughout the development process to identify and address compatibility issues early on.

**User Acceptance**:

1. **Risk**: Low user adoption or dissatisfaction with the applications.
2. **Mitigation**: Involve stakeholders and end-users throughout the development process, gather feedback iteratively, and prioritize user-centric design and features.

**CONCLUSION**

In summary, the proposed project aims to enhance the Simple Calculator by integrating captivating designs, streamlining development processes, and prioritizing user feedback. By implementing modern design principles, Agile methodologies, and automated testing frameworks, we aim to improve user satisfaction, increase efficiency, and drive innovation within the organization. The anticipated benefits include improved user engagement to ensure the success of the project, we have outlined specific metrics for measuring success and identified potential risks along with mitigation strategies. We believe that by investing in this project, we can elevate our software offerings and achieve our organizational goals. We hereby request approval to proceed with the proposed improvement project and look forward to collaborating with team developers to bring this vision to success.