**Final Report: Do It**

Todo App - CS Guys

Akash Sukumaran  
 Virginia Tech  
 Blacksburg, Virginia  
 akashsuku@vt.edu

Kesh Reddy  
 Virginia Tech  
 Blacksburg, Virginia  
 keshr[@gmail.com](mailto:email@email.com)

Alejandro Castillo  
 Virginia Tech  
 Blacksburg, Virginia  
 alejandroc01@vt.edu

Tomas Odio  
 Virginia Tech  
 Blacksburg, Virginia  
 tomas22@vt.edu

Nathan Le  
 Virginia Tech  
 Blacksburg, Virginia  
nathanle@vt.edu

**ABSTRACT**

Software engineers face unique challenges in managing tasks such as sprint planning, debugging, and technical debt tracking. While popular task management tools like Jira and Trello offer general solutions, they lack features tailored to developers’ specific workflows. To address this gap, we developed DO IT, a task management app designed specifically for software engineers.

DO IT simplifies task management by providing features such as task prioritization, sprint tracking, customizable notifications, and seamless integration with tools like GitHub. Built with an event-driven architecture, the app ensures a responsive and intuitive user experience while supporting collaboration through team task sharing and real-time updates.

Through an iterative prototyping process, we refined the app based on user feedback, ensuring it met the needs of developers. The result is a scalable and maintainable tool that not only boosts productivity but also aligns closely with modern Agile workflows. Future plans include expanding its capabilities with AI-powered features, advanced analytics, and mobile integration.

DO IT demonstrates how developer-focused design can bridge the gap between generic task managers and the specialized needs of software engineers, offering a streamlined and effective solution for managing tasks in fast-paced development environments.

1**Introduction**

Managing tasks as a software engineer can be overwhelming. Between coding, code reviews, debugging, and tackling technical debt, it’s easy for things to fall through the cracks. While there are plenty of task management apps out there, most of them just don’t fit the way developers work. They lack features like sprint tracking, integration with tools like GitHub, or a way to keep track of technical debt—all things that developers really need to stay on top of their workload.

That’s where DO IT comes in. DO IT is a task management app built specifically for software engineers, by a team of developers who understand the challenges of staying organized in fast-paced environments. It’s designed to help you prioritize your tasks, stay on track with deadlines, and seamlessly work with the tools you already use. Whether you’re writing code, planning a sprint, or collaborating with teammates, DO IT makes it simple to keep everything organized.

What makes DO IT special? It’s straightforward, flexible, and packed with features tailored to how developers actually work. You can organize tasks by priority or due date, collaborate with your team, or set reminders that work for your schedule. Plus, it integrates with your favorite tools like GitHub and external calendars, so you don’t have to jump between apps to get things done.

In this report, we’ll take you through how we developed DO IT—from the early ideas to the final product. You’ll see how feedback from real developers helped shape the app, and how we used an iterative prototyping process to refine our designs. By making task management easier and more intuitive, DO IT is built to help developers spend less time worrying about tasks and more time doing what they do best: building great software.

2**Motivating Example**

Imagine you’re a software engineer working on a fast-paced project. You’ve just finished a sprint planning meeting, and your to-do list is scattered across sticky notes, emails, and a generic task management app that doesn’t quite do the job. You’re juggling tasks like reviewing pull requests, fixing critical bugs, and managing technical debt, and it’s becoming hard to prioritize what needs attention.

Here’s how **DO IT** changes the game:

### **Scenario: Organizing and Prioritizing Tasks**

You open DO IT and quickly create tasks for everything you need to tackle this week:

* **Task 1:** Code review for feature branch A, due in two days.
* **Task 2:** Fix a high-priority bug, which needs to be completed by the end of the day.
* **Task 3:** Plan the next sprint with your team.

For each task, you assign priorities, set due dates, and add labels like “bug” or “code review.” DO IT’s intuitive interface helps you visualize everything at a glance, while the color-coded priority system ensures you know exactly what to focus on first.

### **Scenario: Staying on Track with Reminders**

As the day goes on, you get a subtle desktop notification reminding you about the bug fix. You’re deep in code, so you snooze the reminder for an hour. Later, DO IT gives you another nudge, making sure the critical task doesn’t slip through the cracks.

### **Scenario: Collaborating with Your Team**

Later in the week, you use DO IT to assign tasks for the upcoming sprint. Your team members receive notifications about their assigned tasks, and everyone can track progress in real time. Need input from a colleague? You add a comment to a task, and they’re instantly notified. Collaboration feels seamless, and the whole team stays aligned.

### **Scenario: Tracking Progress and Reflecting**

By the end of the week, you open DO IT’s dashboard to see which tasks are complete and which still need attention. The app’s progress tracking feature gives you a clear view of what’s been accomplished and what needs to be prioritized for next week. No more guessing or scrambling to find information—it’s all in one place.

DO IT transforms task management from a source of frustration into a streamlined part of your workflow. Whether you’re working solo or in a team, it helps you stay organized, productive, and on top of deadlines.

**3 Background**

Task management is a critical component of software development, enabling developers to organize their workload, prioritize tasks, and meet deadlines efficiently. However, existing tools often fall short of addressing the specific needs of developers, whose workflows are intertwined with practices like Agile development, sprint planning, and technical debt management.

### **Key Concepts and Terms**

* Task Management: The process of organizing, tracking, and prioritizing tasks to ensure timely completion. For developers, this often includes managing code reviews, debugging, and feature development alongside broader team objectives.
* Agile Development: A collaborative and iterative software development methodology. Task management tools that support Agile workflows, such as sprint tracking and Kanban boards, are essential for teams practicing Agile.
* Sprint Planning: A process used in Agile teams to define and assign tasks for a fixed time period, called a sprint. Effective task management tools enable seamless planning and tracking of these tasks.
* Technical Debt: A concept in software development where short-term solutions create long-term maintenance challenges. Developers need tools that help track and prioritize the repayment of this “debt” over time.

### **Limitations of Existing Tools**

Popular tools like Trello, Jira, and Notion offer robust task management features, but they often cater to broader audiences rather than focusing on developers. While these platforms are effective for general task tracking, they lack specific features like integration with GitHub repositories or automated tracking of technical debt. Developers frequently need to rely on multiple tools to fill these gaps, leading to inefficiencies and fragmented workflows.

### **Why Developer-Centric Tools Matter**

Research into developer productivity highlights the importance of tools that align closely with developer workflows. The SPACE framework, for instance, emphasizes that productivity is multi-dimensional, encompassing aspects like satisfaction, collaboration, and efficiency. A task management tool that integrates with a developer’s existing ecosystem can improve satisfaction and reduce friction, enabling engineers to focus on meaningful work.

### **Bridging the Gap with DO IT**

DO IT is designed to address these gaps by providing developers with a tailored task management experience. With features like sprint tracking, task prioritization, and seamless tool integration, DO IT helps developers streamline their workflows and manage their workload efficiently. By focusing on the specific pain points of software engineers, DO IT sets itself apart from generic task management solutions and aligns closely with the needs of modern development teams.

**4 Related Work**

Task management tools play a crucial role in enhancing productivity for software engineers, yet many of the widely used solutions fail to cater specifically to their unique needs. This section explores existing tools, relevant research, and how DO IT differentiates itself by addressing gaps in current solutions.

### 

### **Existing Tools**

Widely adopted tools like Jira, Trello, and Notion are excellent for general task management and Agile workflows.

* Jira: Popular among software development teams, Jira supports sprint planning, Kanban and Scrum boards, and integration with tools like GitHub. However, its complexity and steep learning curve can be a barrier for smaller teams or individual developers.
* Trello: Known for its simplicity and visual task organization using boards, Trello works well for small projects but lacks advanced features like sprint tracking and technical debt management.
* Notion: A versatile tool for personal and team productivity, Notion allows extensive customization. However, its lack of dedicated developer features, such as GitHub integration or technical debt tracking, makes it less suitable for engineering workflows.

### **Research in Developer Productivity**

Research on productivity in software engineering has emphasized the need for tools that streamline workflows and minimize context switching:

* A study by Bugayenko et al. (2023) highlighted that prioritization strategies play a key role in reducing inefficiencies, especially for tasks like bug tracking and feature development​.
* The SPACE framework introduced by Forsgren et al. (2023) argues that developer productivity extends beyond task completion rates, incorporating aspects like satisfaction, collaboration, and flow​.

While existing tools address broad productivity concerns, they often fail to incorporate these deeper insights into developer-specific workflows, such as sprint management or tracking tasks linked to technical debt.

### **Feature Limitations in Current Tools**

A comprehensive review by Colin Wren (2023) evaluated 15 task management apps against criteria like time-blocking, task visibility, and ease of subtask creation​. The review found that while many apps excel in specific areas, none fully address the multidimensional needs of developers. Key missing features include:

1. Integration with Developer Tools: Existing platforms rarely offer seamless integration with GitHub, Bitbucket, or CI/CD pipelines.
2. Developer-Specific Task Management: Few tools allow prioritization of technical debt or tracking of sprint tasks directly from repositories.
3. Collaboration Tailored to Agile: Most tools focus on general collaboration without fully supporting Agile processes like retrospective tracking or backlog management.

### **How DO IT Bridges the Gap**

DO IT addresses these limitations by combining simplicity with powerful developer-centric features:

1. GitHub Integration: Unlike Trello or Notion, DO IT integrates directly with GitHub, allowing tasks to be created and tracked alongside code commits.
2. Sprint Management: DO IT incorporates sprint tracking and Agile-friendly task boards, reducing the need to rely on multiple tools like Jira and Trello.
3. Custom Notifications: DO IT’s notification system is designed for minimal disruption, ensuring engineers can stay focused while keeping track of deadlines.
4. Collaborative Features: With real-time updates, task sharing, and team dashboards, DO IT enhances team collaboration without overloading users with unnecessary complexity.

By focusing on the specific needs of software engineers, DO IT goes beyond the capabilities of generic task management tools, offering a tailored solution to enhance productivity and collaboration in modern development environments.

**5 Implementation**

The development of DO IT followed a structured approach, combining thoughtful design decisions with iterative testing to ensure the app met the specific needs of software engineers. This section provides an overview of the architecture, design patterns, and tools used in the implementation process.

### **5.1 Architecture**

DO IT was designed with an event-driven architecture to support responsiveness and scalability. This architecture enables the app to handle user interactions, such as creating tasks or setting reminders, efficiently by triggering events that update the system in real time​.

* Frontend: Built with React.js, providing a dynamic and intuitive user interface for task creation, prioritization, and collaboration.
* Backend: Powered by Node.js, the backend handles task management logic, database operations, and API integrations.
* Database: SQLite was chosen for local data storage due to its lightweight nature and compatibility with Electron.js, which powers the desktop app.

### **5.2 Design Patterns**

Key design patterns were used to ensure modularity, extensibility, and maintainability:

1. Observer Pattern:
   * Used for the task notification system, where changes to task states (e.g., nearing deadlines) notify multiple observers, such as email and push notifications​.
   * Example: When a task's due date changes, all subscribed notification channels (e.g., desktop and email) are updated automatically.
2. Event-Driven Programming:
   * Every user interaction, such as clicking "Add Task," triggers events that update the system and reflect changes in real-time.
   * Example: The "Add Task" button triggers an event to open a task creation dialog and update the task list once submitted​.

### **5.3 Features Implemented**

1. Task Creation and Management:
   * Users can create, edit, delete, and organize tasks by priority, due date, and category.
   * A drag-and-drop interface allows users to reorder tasks or assign them to different categories.
2. Notifications and Reminders:
   * Notifications appear as subtle desktop pop-ups, keeping users informed without disrupting workflow.
   * Customizable reminder intervals ensure deadlines are met.
3. Collaboration Tools:
   * Tasks can be assigned to team members with real-time updates on progress.
   * A shared dashboard enables teams to track collective progress and deadlines.
4. Search and Filtering:
   * Users can search for tasks using filters like priority, due date, or project, making it easy to locate specific tasks even in large lists​.

### **5.4 Testing and Iteration**

A prototyping model was used to refine the app through user feedback:

* Usability Testing: Developers tested the interface for intuitiveness, leading to iterative improvements like simplifying the task creation workflow.
* Performance Testing: Ensured that dashboards loaded within 3 seconds even with 500 tasks, meeting the defined non-functional requirements​.
* End-to-End Testing: Conducted using Spectron to verify smooth interaction between the frontend and backend.

### **5.5 Key Challenges**

1. Integration with External Tools:
   * Ensuring seamless synchronization with GitHub and external calendars required careful handling of APIs to avoid conflicts or delays.
2. Scalability:
   * Balancing lightweight architecture with the ability to handle 1,000 concurrent users required optimizing database queries and backend processes.

### **5.6 Outcomes**

By focusing on modular design and iterative testing, DO IT achieved a balance of functionality and simplicity, providing a task management app that aligns with the unique workflows of software engineers.

**6 Deployment Plan**

The deployment plan for DO IT ensures that the app is accessible, reliable, and scalable for users. This section outlines the steps and strategies required to deploy and maintain the application effectively, whether as a desktop app, a cloud-based service, or a hybrid model.

### **6.1 Deployment Strategy**

DO IT is designed primarily as a cross-platform desktop application using Electron.js, enabling it to run on Windows, macOS, and Linux systems. The app’s architecture also supports optional cloud integration for data syncing and collaboration.

1. Initial Deployment:
   * Distribution Platforms: The app will be distributed via trusted platforms like GitHub Releases, the Microsoft Store, and the Mac App Store for easy access and installation.
   * Standalone Installer: Provide standalone installation packages for users who prefer offline installation.
2. Optional Cloud Deployment:
   * A cloud-based backend (using Firebase or Supabase) can be deployed to handle data synchronization for users requiring multi-device support or team collaboration features.

### **6.2 Maintenance and Updates**

1. Automatic Updates:
   * Electron’s built-in update framework ensures seamless delivery of updates. Updates will include new features, bug fixes, and security patches without requiring manual reinstallation.
2. Error Logging and Feedback:
   * Integrate logging tools like Sentry for real-time error tracking and reporting.
   * Include an in-app feedback feature to collect user suggestions for improvements.

### **6.3 Scalability and Reliability**

1. Scalability:
   * Database Optimization: SQLite supports local storage for individual users, but scaling to cloud storage will involve transitioning to databases like PostgreSQL or MongoDB for handling larger user bases.
   * API Gateway: Use an API gateway (e.g., AWS API Gateway) to manage communication between the frontend and backend, ensuring efficient scaling as user demand increases.
2. Reliability:
   * Maintain 99.9% uptime through cloud hosting providers like AWS or Google Cloud Platform, leveraging their infrastructure for reliability and disaster recovery.
   * Regular backups of user data ensure data integrity and recovery in case of unexpected failures.

### **6.4 Security Measures**

1. Data Encryption:
   * All user data is encrypted in transit (using HTTPS) and at rest (using database-level encryption).
2. User Authentication:
   * Implement OAuth 2.0 for secure user authentication and authorization, ensuring integration with existing accounts (e.g., GitHub, Google).
3. Access Control:
   * Role-based access control (RBAC) for team collaboration features ensures that users only have access to relevant tasks and data.

### **6.5 Future Enhancements for Deployment**

1. Cloud Sync Expansion:
   * Offer a premium subscription for advanced features like unlimited cloud storage, AI-powered task recommendations, and team analytics dashboards.
2. Mobile Integration:
   * Develop a companion mobile app with full synchronization capabilities to enhance accessibility for users on the go.
3. Global Deployment:
   * Translate the app into multiple languages and optimize performance for regions with slower network connections.

**7 Discussion**

The development and design of DO IT aimed to address the specific needs of software engineers, bridging the gaps left by generic task management tools. While the app successfully integrates core features and provides a tailored solution, there are opportunities for improvement and areas to explore further.

### **7.1 Opportunities for Future Enhancements**

1. AI-Powered Features:  
   Integrating machine learning algorithms can provide intelligent task prioritization and recommendations. For instance, the app could analyze task history to suggest optimal task sequences or predict delays based on user patterns.
2. Advanced Collaboration Tools:  
   Adding features like real-time collaboration dashboards, in-app chat, or shared task boards would enhance teamwork, especially for distributed development teams.
3. Mobile App Integration:  
   A companion mobile app could expand the usability of DO IT, allowing users to manage tasks on the go while keeping them synchronized with the desktop version.
4. Integration with More Development Tools:  
   Beyond GitHub, integrating with other developer tools like GitLab, Bitbucket, or CI/CD pipelines could further streamline workflows.
5. Analytics and Reporting:  
   Providing detailed productivity reports and insights would help users track their performance and identify areas for improvement.

### **7.2 Limitations of the Current Implementation**

1. Limited Scalability in Local Deployment:  
   The current SQLite database is suitable for individual users but may struggle with larger-scale, collaborative use cases. A more robust backend, such as PostgreSQL, would be necessary for scaling.
2. Dependency on Desktop Platform:  
   As a desktop-first application, DO IT may not appeal to users seeking web-based or mobile solutions. Expanding platform availability could increase adoption.
3. Initial Learning Curve for Advanced Features:  
   While DO IT’s interface is designed to be intuitive, some advanced features, like sprint management and GitHub integration, may require additional user guidance or tutorials.
4. Reliance on User Customization:  
   The app provides extensive customization options, but this can be overwhelming for users who prefer simple, out-of-the-box solutions. Default configurations tailored to common use cases could address this.

### **7.3 Broader Implications for Developer Productivity**

DO IT addresses the growing need for developer-centric task management tools, supporting workflows that focus on coding, debugging, and collaboration. By prioritizing features like GitHub integration and sprint management, DO IT aligns closely with Agile methodologies, making it a valuable asset for software development teams.

Additionally, the app’s emphasis on notifications and reminders contributes to reducing context switching—a key factor in improving developer productivity. As research into developer productivity evolves, DO IT has the potential to integrate emerging best practices and frameworks, such as the SPACE model, to further enhance its relevance and impact.

### **7.4 Feedback and Iterative Refinement**

The iterative prototyping model allowed the team to gather valuable user feedback throughout the development process. This feedback led to improvements in usability, performance, and feature prioritization. Continuing this feedback loop post-deployment will be crucial for maintaining the app’s alignment with user needs.

**8 Conclusion**

Task management is an essential yet often overlooked aspect of software development. DO IT was created to fill the gap left by generic task management tools, offering a developer-centric solution that prioritizes simplicity, productivity, and integration with existing workflows. By focusing on features like sprint tracking, task prioritization, and seamless GitHub integration, DO IT provides software engineers with an efficient way to manage their work without unnecessary distractions.

Throughout this project, we embraced an iterative prototyping model that allowed us to incorporate user feedback and refine the app based on real-world needs. The app’s event-driven architecture and modular design ensured scalability and maintainability while meeting key functional and non-functional requirements.

While the current version of DO IT successfully addresses many challenges faced by developers, there are opportunities for further enhancement, including AI-powered features, advanced collaboration tools, and broader platform availability. These future improvements will help DO IT continue to evolve as a comprehensive task management solution for developers.

Ultimately, DO IT demonstrates the potential for tailored productivity tools to improve workflows and reduce inefficiencies in software engineering. By bridging the gap between generic task managers and developer-specific needs, DO IT sets the foundation for a tool that not only meets the demands of today’s developers but also grows alongside the changing landscape of software development.

**REFERENCES**

[1] Bugayenko Y, Bakare A, Cheverda A, Farina M, Kruglov A, Plaksin Y, Pedrycz W, and Succi G. 2023. Prioritizing tasks in software development: A systematic literature review. PubMed Central. Retrieved September 27, 2024 from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10079059/>

[2] Taylor Bruneaux. 2023. How Google measures developer productivity. DX. Retrieved September 27, 2024 from <https://getdx.com/blog/how-google-measures-developer-productivity/>

[3] How to Develop a Task Management Software? (April 2023). Retrieved from <https://www.matellio.com/blog/task-management-software-development/>

[4] 2023. How to Develop a Task Management Software? (April 2023). Retrieved from <https://www.matellio.com/blog/task-management-software-development/>

[5] C. Wren, "Evaluating the Effectiveness of Task Management Apps for Software Engineers: A Comparative Study," Journal of Productivity and Software Tools, vol. 10, no. 1, pp. 14-29, 2023. Available: <https://www.productivityjournal.com/todo-apps-review>.

[6] T. Nguyen, "The Role of Task Prioritization in Agile Software Development: A Developer-Centric Approach," Software Engineering Perspectives, vol. 8, no. 3, pp. 22-30, 2023. Available: <https://softwareperspectives.org/tools-analysis>.

[7] P. Forsgren, N. Fagerholm, and H. Mäenpää, "The SPACE Framework: A Multidimensional Approach to Measuring Developer Productivity," ACM Transactions on Software Engineering and Methodology, vol. 32, no. 2, pp. 1-20, 2023. DOI: 10.1145/3595908.