The TODO Desktop Application

An Introduction to an Application to Help Struggling Programmers

Ryan Szilvasi
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
rszilvasi@vt.edu

Tyler Streator
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
tys305@vt.edu

Junxiang Feng
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
junxiang@vt.edu

Xiaolong Xuan
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
xxiaolong@vt.edu

Kechen Yu
Computer Science
Virginia Tech
Blacksburg, Virginia,USA
kechen21@vt.edu

ABSTRACT

Every day, Computer Scientists are constantly worrying about their deadlines and work with changing requirements. It doesn't help that many industries forget to remind their workers of important deadlines for their programs. In an example provided by "Impact of time pressure on software quality: A laboratory experiment on a game-theoretical model," the authors explain that when delivering underdeveloped software solutions near a deadline, the developer must choose between a high-quality but late application or a low-quality application that meets the deadline requirements [1]. This TODO system that we are proposing will be able to help developers achieve their high-quality products without sacrificing their deadlines. This proposed solution will be able to notify individuals of the code they need to complete by the end of the day. The view of the application will have a high contrast so that important information can stick out to the developer. In addition, programmers can prioritize/rank their assignments in order of importance, or another order of their choice. Between Team Managers and Team Members, this application will be able to share and assign tasks to individuals on their team to ensure that everyone is meeting their deadlines.

CCS CONCEPTS

• Java • Windows • Linux • OS •

KEYWORDS

TODO Application, Deadlines, Programmers, Developers, Software, Computer Science

ACM Reference format:

Ryan Szilvasi, Tyler Streator, Junxiang Feng, Xiaolong Xuan and Kechen Yu. 2024. The TODO Desktop Application: An Introduction to an Application to Help Struggling Programmers

1 INTRODUCTION

As part of their work, programmers must regularly keep track of many different tasks at the same time. Any major programming project can be broken down into many subtasks that must be completed in order or in parallel, as well as meetings and other important events. Keeping track of these is complicated further by the possibility of changes in the requirements for the program partway through the project. The number of independent moving parts in the process can easily become too large to effectively keep track of, and developers could benefit from the ability to more easily see which tasks they need to accomplish within a given timeframe. Our proposed solution for this problem is an application to help programmers track tasks and events. The system would allow users to set daily or weekly goals for themselves or their teams, remind them of important deadlines, and sort their tasks based on importance or a variety of keywords. This would aid both groups and individuals in meeting their targets and help increase overall productivity.

From personal experience, we know how often our work piles up, especially with our plethora of Computer Science projects. We’ve noticed that trying to implement the entire project at once is not efficient, but rather splitting the entire project into smaller components is a much better use of our time. With these smaller components, having a tracking system, such as the one we are proposing, ensures that no subtask is forgotten, and all are completed by the proposed deadline.

2 RELATED WORK AND RESEARCH

2.1 Git: Version Control System

Git is an excellent and relevant software tool with powerful capabilities for managing source code changes, facilitating collaboration, and maintaining a history of project evolution. With its branch and merge capabilities, developers can develop distinct functions simultaneously without affecting the stability of the main codebase.

Branching capabilities enable developers to create separate lines of development for new features or bug fixes, which is critical to experimenting with and implementing new features without affecting stable versions of the application.

The merge capability enables changes in different branches to be consolidated back into the main branch, consolidating development efforts, and seamlessly integrating new functionality.[2]

2.2 Kanban: Enhancing Project Management

Task visualization: Kanban visualizes tasks at different stages of the development process. This visualization helps the team understand the workload and progress in detail, allowing for better planning and prioritization.

Workflow management: Kanban can organize tasks into columns and use cards to represent individual work items. The team can identify the process and measure the workload to ensure the sustainability and stability of the task in development.

Git integration: Kanban can be integrated with Git repositories to automatically update task status based on Git actions. For example, when the pull request is open, the task is moved to the Review bar, and when the pull request is merged, the task is moved to the Complete bar. This automation reduces manual updates and enables Kanban to accurately reflect project status. [3]

2.3 Maximizing Efficiency: A Comprehensive Review of Daily Productivity Growth with Todo Manager [4]

In the provided research paper, Gilberto Perez and Nidhaulla Sheikh proposed a To-do manager that their team created. Throughout their research, they noticed that an organizing system for a software developer's tasks can help, "prioritize tasks and work on the most important ones, provides reports to analyze daily task efficiency, and motivates users to increase progress and avoid distractions" [4]. Trying to achieve these results, they developed their own event-based system to help developers keep track of tasks. Knowing the importance of this, we hope to also provide developers with a similar solution to ensure that most software engineers have the opportunity to use a variety of platforms.

3 IMPLEMENTATION

3.1 Process

In this project, we chose the Prototyping Process. We're going to use a modified Prototyping Model for our software development process. The requirements for the software may evolve based on the problems and ideas we found in the project.

1. Analyze the project: We begin by collectively defining the initial project requirements. This stage includes discussion and agreement on the basic functionalities and goals of our software.
2. Building the Initial Prototype with Java and VS Code: For compilation, we use Visual Studio Code (VS Code), which is the best platform for our students who are new to software development.

3. Iterative Development with Git and Kanban: Our development process is an iterative process. We use Git to manage our code base. It allows us to update our progress in real-time and easily revert to a previous version when needed. In addition to Git, we also use Kanban for task management, giving us a clear visualization of project progress and facilitating seamless collaboration among team members.
4. Prototype Refinement through Team Feedback: Each prototype iteration is experienced and tested by our team. We give each other feedback and discuss and synthesize each member's point of view. This process helps us identify areas for improvement and improve the prototype accordingly [1].
5. Final Development and Integration: After a few iterations and improvements, our final version is here. This release passes all our proven features and functionality and gives users a great experience.

3.2 High-Level Design Decision

For our high-level design approach, we believe that the most effective architecture pattern for our project is a layered architecture. This pattern provides a hierarchy of layers that can communicate with each other and is well-suited for applications that must communicate with a server. We think that the Model-View-Controller pattern would be the best fit for our project. This pattern is very common for web applications and is generally suitable for developing user interfaces. Our application must store information about tasks on a server (model), retrieve and display this information to users (view), and allow users to add and modify tasks for themselves or their teams (controller).

3.3 Testing

As the goal of testing is to find errors within our systems, we are currently proposing 12 different black box test cases. As our system would rely heavily on input from the user and the user seeing the system's output, we believe this type of testing would be the best. In addition, since we have not fully implemented the system, having test cases that ignore most of the internal fragmentation of the code and focus heavily on the functionality is the best approach. These test cases include the following:

1. TestLogin – a test case to see if the user can log in successfully with a valid email and password
2. TestSignUp – a test case to see if the user can create an account
3. TestForgotPassword – a test case to see if the user can create a new password from a valid email
4. TestMoveEventFromStartToProgress – a test case to see if the user can move an event from the “Needs to be Started” column to the “In-Progress” column
5. TestMoveEventFromProgressToFinish – a test case to see if the user can move an event from the “In-Progress” column to the “Finished” column

6. TestMoveEventFromSharedToStart – a test case to see if the user can move an event from the “Shared With You” row to the “Needs to be Started” column
7. TestMoveEventFromSharedToProgress – a test case to see if the user can move an event from the “Shared With You” row to the “In-Progress” column
8. TestAddEvent – a test case to see if the user can add an event to their home page
9. TestShareEvent – a test case to see if the user can share an event with another user
10. TestDeleteEvent – a test case to see if the user can delete an event from their home page
11. TestChangeEventColor – a test case to see if the user can change the color of an event
12. TestEmailNotification – a test case to see if an email with receive an email for an approaching deadline

For more information on these test cases, reviewing our CS3704-BlackBoxTesting.pdf provides a further breakdown of each test. With these test cases, our testing approach would fall under Validation Testing, as we would want to ensure that our solution meets the requirements that we proposed. Once the system passes all of these tests, we could continue to advance our system to the deployment stage.

4 DEPLOYMENT AND MAINTENANCE PLAN

4.1 Deployment Plan

If we were to deploy this project, we would want to follow the Canary Deployment. As our system is still in the initial stages of development, we want to only give a limited number of users access to the system. Utilizing this type of development can ensure that our system is not overloaded by a surge of large numbers of users. In addition, the Canary Deployment complements our Prototyping process very well. After creating one iteration of our project, we can release it to a small subset of users. With this, we can obtain feedback from our users to see areas that can be improved upon. Utilizing the Canary Development also ensures that we complete our system in small phases at a time and fix issues that arise, rather than release an entire system at once with a lot of flaws.

4.2 Maintenance Plan

Our maintenance plan will tend to be Corrective Software Maintenance. We can utilize tools like Statuspage and Cloudflare which allows us to monitor our API services. Statuspage will allow us to track the API status over time when a certain API route is down and unfunctional. It will send notifications to developers to notify them as soon as possible. Also, Statuspage is a website that shows API status to the public so that our users can know which services are affected and if the services are being fixed. Other than the API route error, some users might experience local application bugs like some UIs not showing, buttons not working, etc. We will have a bug report website/email to let users report any bugs they encounter, and developers will start to fix the bug as soon as possible.

5 CONCLUSIONS

5.1 Work Completed

With our original problem statement, we wanted to provide a solution to ensure that software engineers are not falling behind in their projects, whether that be in an industrial or academic setting. At the current moment, we have not completed any software development for this system. However, we completed a requirement analysis of our system, in addition to creating a GUI that a user can interact with. The following figures are some of the screens that a user will see with our TODO Desktop Application:

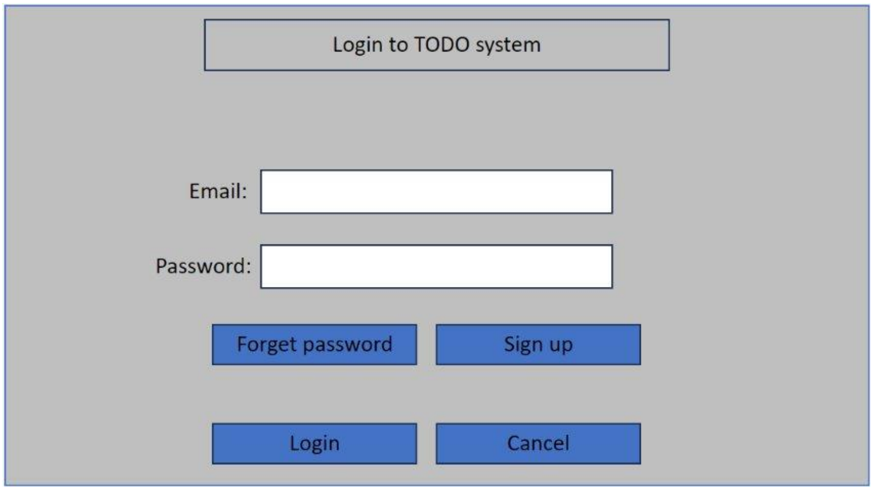


Figure 1. Login Screen

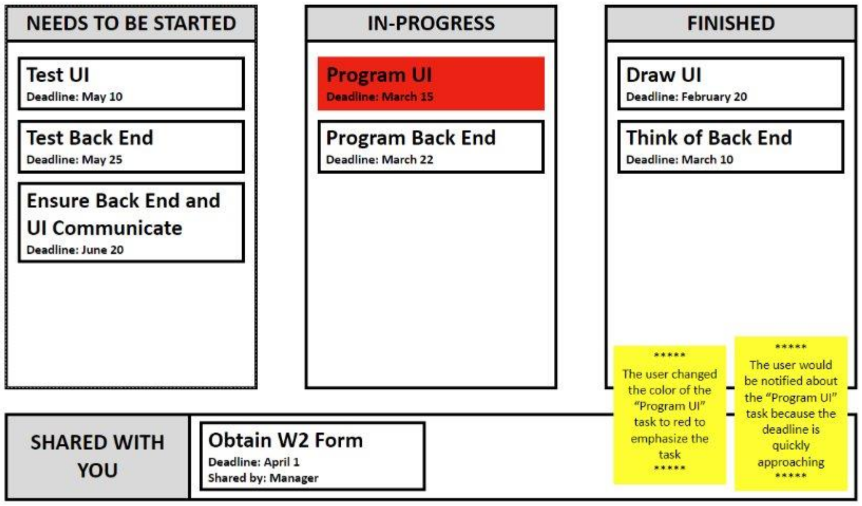


Figure 2. User home screen where they can see the events they need to complete

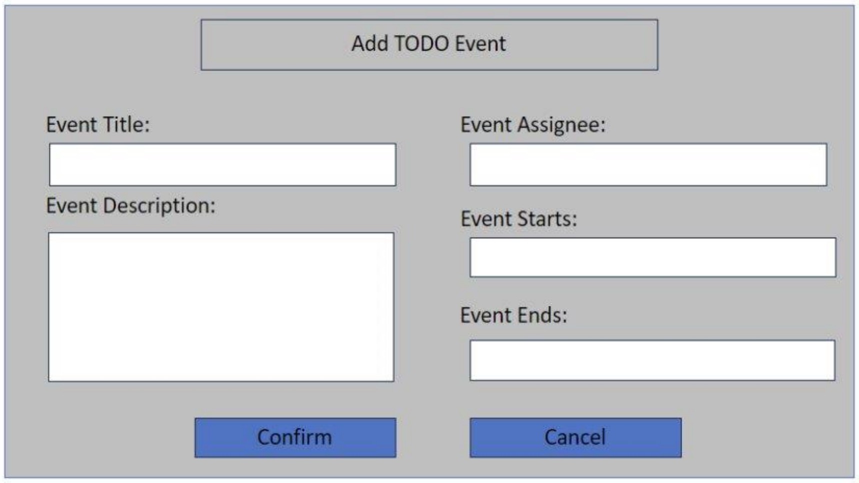
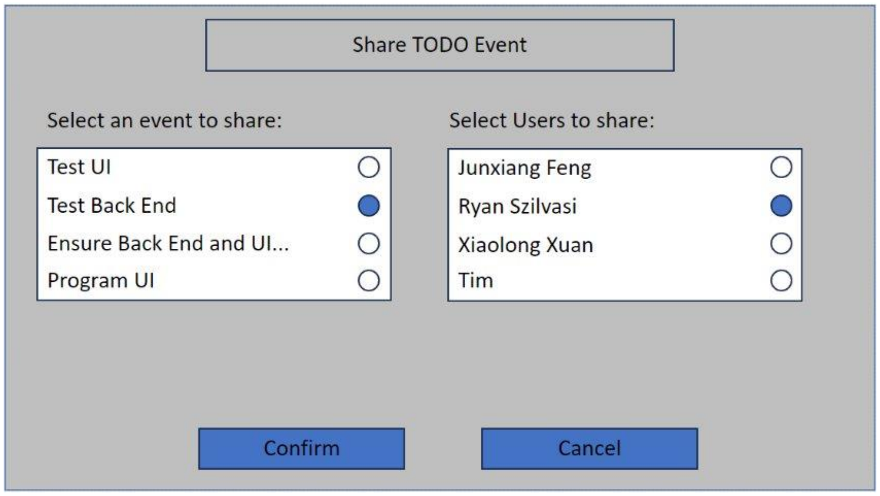


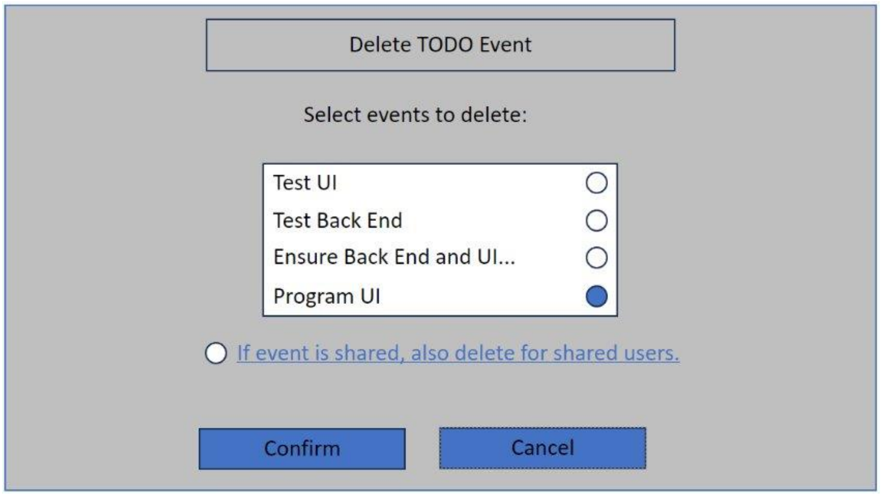
Figure 3. A screen for the user to create an event and add details to it.



Test UI
[✓ Test Back End]
Ensure Back End and UI...
Program UI

Junxiang Feng
[✓ Ryan Szilvasi]
Xiaolong Xuan
Tim

Figure 4. A screen for the user to share an event with another user



Test UI
Test Back End
Ensure Back End and UI...
[✓ Program UI]

Figure 5. A screen for the user to delete an event from their home screen.

5.2 Future Work and Limitations

TODO desktop applications have the potential for significant improvements in the future. By focusing on cross-platform compatibility, developers can create applications that run across various operating systems, such as Windows, Mac, and Linux. This eliminates the need for users to install multiple versions of the same app. Additionally, integrating cloud-based synchronization and storage services like iCloud enables real-time collaboration and access to tasks and notes from any device and from anywhere you want. Furthermore, incorporating advanced features like artificial intelligence can automate task scheduling, priority setting, and predictive task creation based on users' behaviors and preferences. For example, the application can analyze users' habits through machine learning and automatically create daily to-do lists when the user starts their workday. It can enhance productivity and user experience a lot.

However, TODO desktop applications also currently face several limitations that need to be addressed. Platform dependency is a significant issue since most applications are developed for specific operating systems. It will restrict their usability across different platforms without additional development work. This limitation will be inconvenient for users who work on multiple devices with different operating systems. Moreover, desktop applications often lack integrated real-time collaboration features, making it challenging for multiple users to work together synchronously on the same tasks. It will decrease the team's efficiency. Lastly, the requirement to install desktop applications on individual computers limits accessibility since users can only access the application and its data on the devices where it is installed. It will prevent them from accessing their content on other devices. Addressing these limitations will be crucial for the future success and widespread adoption of TODO desktop applications.

REFERENCES

[1] Basten, D., Müller, M., Ott, M., Pankratz, O., & Rosenkranz, C. (2021). Impact of time pressure on software quality: A laboratory experiment on a game-theoretical model. *PloS one*, *16*(1), e0245599. https://doi.org/10.1371/journal.pone.0245599

[2] Chacon, S., & Straub, B. (2014). Pro Git. Apress.

[3] Anderson, D. J. (2010). Kanban: Successful Evolutionary Change for Your Technology Business.

[4] Perez, G., & Sheikh, N. (2023, March). Maximizing Efficiency A Comprehensive Review of Daily Productivity Growth with Todo Manager. International Journal of Research Publication and Reviews.