Automated Task Manager

Final Report

Ethan Werner
Computer Science
Virginia Tech
Blacksburg, VA, US
ethanw21@vt.edu

Jake Frohlich
Computer Science
Virginia Tech
Blacksburg, VA, US
jakefrohlich00@vt.edu

Neha Bangari
Computer Science
Virginia Tech
Blacksburg, VA, US
nehabangari@vt.edu

Vrishank Bangari
Computer Science
Virginia Tech
Blacksburg, VA, US
vrishank@vt.edu

ABSTRACT

In today's fast-paced software development environments, effective task management is essential for maintaining productivity and meeting project deadlines. Software engineers often deal with organizing, prioritizing, and distributing tasks manually, which can lead to inefficiencies and uneven workload distribution. Additionally, unexpected interruptions can disrupt workflow and can lead to delays within teams. Our solution addresses the problem of manual task allocation by introducing an automated task scheduler designed specifically for software engineering teams and individuals. The proposed solution uses algorithms to delegate tasks automatically throughout the week, considering factors such as task priority, individual skill sets, and current workloads. By automating the delegation process, the scheduler aims to optimize resource utilization and increase team productivity. Additionally, the product will reduce bias within teams in terms of deciding task importance. This system will be designed with the intent of fostering a more balanced and efficient work environment for software engineers. Ultimately, this will increase motivation, reduce burnout, and help software engineers prioritize their actual work.

INTRODUCTION

In the modern world of software engineering, managing many tasks and projects simultaneously is often subject to increased complexity for the engineer. Software engineers are often faced with the difficulty of manually prioritizing these tasks, allocating resources and balancing their own workload to finish deadlines on time and efficiently. This can lead to stress and decreased productivity. Additionally, these manual approaches not only consume valuable time for the engineer but can also result in inefficacious task distribution. With the dynamic nature of the general scope of software engineering–in which timelines, tasks and requirements can change at any given moment based on investor needs–there would naturally need to be a tool in place to maintain efficient task management without constant manual adjustments to the system.

The Task Scheduler we propose seeks to address this issue by automating the entire process of task delegation from the user's perspective. The solution we propose leverages AI to consider how long a given engineer should take to complete tasks, which tasks should come first and the deadlines associated with each task. With this system, the software engineer can adapt to shifting projects, demands or interruptions instantly through our tool, increasing the overall productivity and throughput of the engineer. In another perspective, with our product, a software engineer can allocate more focus to the development and problem-solving related work than spending time figuring out the delegation of the work at hand.

MOTIVATING EXAMPLE

Suppose a software engineer was working on a project that involves implementing a new feature while also fixing bugs in the production environment. In the current setup of their team, the developers use Jira for task allocation. Each week, the team spends a significant amount of time organizing tasks, updating them, and reprioritizing when issues escalate. As new tasks are added or deadlines shift, the software engineer has to continually balance priorities and experiences an interruption in their focus and productivity. With the proposed automated task scheduler, the software engineer's workflow becomes significantly more streamlined. The tool would automatically analyze their list of tasks, assesses deadlines, estimate durations, and reprioritized dynamically as new issues or changes arise. For example, if a critical production bug is reported, the system identifies it has a high priority and adjusts the software developer's schedule accordingly. The tool also tracks the developer's past performance and workload, ensuring tasks are distributed in a way that minimizes burnout.

By automating task delegation and prioritization, the task scheduler frees engineers to focus on solving more important problems rather than having to manage their own workloads in relation to the rest of the team. This not only boosts productivity but also reduces the cognitive load and stress that engineers experience in development environments.

BACKGROUND

There are multiple concepts related to our project. For example, the first one is the singleton database design decision. We decided in PM2 to implement the singleton database design pattern to lower the stress on the project and decrease lines of code, which would effectively increase our maintainability index. Additionally, we decided to use an event-based architecture design throughout the project. The event-based architecture design allows for asynchronous updates upon any triggering action associated with the code. This is the basis for our implementation, as the event-based architecture encourages lightening the impact load of the system at runtime and will directly increase uptime throughout the launch of the product should it ever be taken to market as a product in the future. To define "dynamic" in the name "dynamic task scheduler": dynamic is the ever-changing input by the user at any given time to the UI while the scheduler is up and actively running the AI algorithm during uptime.

RELATED WORK

There are some existing tools that software developers use to address similar concerns. Trello is a task management tool that uses boards, checklists, and cards to help teams organize their work [1]. Trello allows users to manually enter tasks and assign due dates and team members to them. There are multiple features to organize Trello boards, and the platform also offers templates so users don't have to create a board from scratch. Another similar task management tool is Monday.com, which offers the ability to create custom dashboards to view various features such as team progress and hours, budgets, and task assignments [2]. Jira is another common software application used for project management and issue tracking. It contains several features such as Kanban/Scrum boards and sprint planning. This tool also works well with other software like Confluence, Bitbucket, GitHub, and Slack, providing further development support. All of these tools allow users to manually create visual representations of task distribution for a given project and offer some support through templates and existing designs. However, these applications do not have the ability to intelligently and automatically distribute tasks across a team.

In a research study detailed in "A Study on Task Management System," authors Jyothi and Parkavi explain a proposed technique to define tasks using a matrix to aid in project management. The authors described the use of a task matrix based on an Eisenhower matrix to delegate tasks across a project management team [3]. In this study, the authors describe this concept as a 2x2 matrix that organizes tasks by importance and urgency and also lists features of project deliverables. This matrix model is useful because it clearly organizes tasks and descriptions, making it easier for teams to evaluate task importance. The authors also go over a few different prioritization models in order of complexity. The first one is a HI/LO model. This model evaluates tasks based on their impact and complexity. Another similar model is the CARVER model, which uses six factors to evaluate tasks: criticality, accessibility, return, vulnerability, effect, and recognizability. Each of these factors are assigned a score on a scale of 1 to 5 to prioritize tasks. Finally, the Carpenter model uses weighted comparisons between criteria in order to rank tasks. This model typically relies on automated calculations and visual results. The paper also lists some existing task management software, including Trello, Todoist, and Taskworld.

Our project is novel because it automates the process of scheduling these tasks. Many of the existing tools and concepts are helpful in structuring project management but require manual scheduling. Our tool leverages AI to automatically determine the optimal sequence of tasks based on deadlines, complexity, and the availability of resources. This eliminates the time-consuming process of task allocation. The task scheduler tool also adapts in real-time to changes in projects, requirements, and deadlines. Traditional tools require developers or project managers to make these updates on their own, which can potentially lead to inefficiency and errors. This tool intelligently assesses workload and productivity of developers within a team to assign tasks more effectively. This not only promotes accuracy, but also reduces the cognitive load associated with task management allowing developers to focus on their projects. Overall, the task manager increases productivity and throughput.

IMPLEMENTATION

The software engineering process we will be using is Agile, more specifically extreme programming. We chose this because the product we aim to create is heavily focused on user experience. The purpose of this software will be to make the lives of software engineers easier so we will want to receive constant feedback on each iteration of our plan and product so we can perfect it. This is the main reason why we chose to use Agile. Another reason is that Agile allows for high quality products at high speed and we have a relatively short time frame to complete this project. Agile will let us truly maximize our time. Additionally, it will allow us to gather feedback after each iteration in order to make changes and better the product. It will also let us be more open with communication, allowing us to convey our opinions and thoughts more frequently.

We chose to hone in on extreme programming because of its ability to support changing customer needs. This process will allow us to go through shorter development cycles, which can be useful in a group as we can get continuous feedback. Through this process, we will present every iteration to software engineers and get feedback for improvement on the next iteration. We will also hold regular scrum meetings to ensure all team members are in the loop on all work being done throughout the development process. As our product is user-centric, we hope to get feedback from our potential user group as well.

DEPLOYMENT PLAN

To successfully deploy the automated task scheduler, the first step involves designing a robust and scalable system architecture. The system will leverage a cloud-based infrastructure to ensure flexibility, scalability, and cost-efficiency. Cloud service providers such as AWS, Microsoft Azure, or Google Cloud Platform (GCP) will host the system. The system architecture will include several key components. The front-end will consist of a user-friendly web interface developed using frameworks like React.js or Angular.js, enabling software engineers and managers to interact seamlessly with the tool. The back-end will utilize RESTful APIs built with frameworks like Node.js or Flask to handle task management, prioritization logic, and communication with the database. A relational database, such as PostgreSQL or MySQL, will store task data, schedules, user information, and system logs. For enhanced performance, a non-relational database like MongoDB will be implemented to enable caching and quick retrieval of dynamic data. The AI engine, which is the core of the scheduler, will analyze workloads, prioritize tasks, and adjust schedules dynamically. This engine will rely on machine learning libraries such as TensorFlow or PyTorch and will be deployed as a containerized service. The system will use Docker for containerization and Kubernetes for orchestration, ensuring high availability, load balancing, and automatic scaling of resources. Finally, a Continuous Integration and Continuous Deployment (CI/CD) pipeline will automate testing, building, and deployment processes using tools like Jenkins or GitHub Actions to streamline updates and reduce manual intervention.

The deployment process will occur in distinct phases to minimize risks and gather valuable user feedback for refinement. The first phase, alpha testing, will involve deploying the initial version of the tool to a small group of software engineers and project managers within a controlled environment. The purpose of this phase is to test core functionalities and identify any bugs or issues. Metrics such as user feedback, task prioritization accuracy, and system performance will be collected using tools like Postman for API testing and Selenium for automated UI testing. Following successful alpha testing, the beta testing phase will roll out the tool to a larger audience within a pilot group. During this phase, beta testers will validate the tool's utility by integrating it with their existing workflows. Features such as task reassignment efficiency, workload balancing, real-time adaptability, and user satisfaction will be monitored closely. Any reported usability or performance issues will be resolved before the full-scale launch. The final phase, full deployment, will involve a broader rollout of the tool to all targeted teams. An incremental deployment strategy will be employed, starting with a smaller subset of users before expanding to others, ensuring a smooth transition and enabling continuous monitoring of adoption rates.

To facilitate seamless adoption, the automated task scheduler will integrate with existing tools commonly used by software teams. The system will integrate with Jira using Jira's REST API to pull tasks automatically, update priorities, and reflect these changes within Jira. Notifications will also be sent to team members via Slack when schedules are dynamically updated. The scheduler will align with GitHub to prioritize tasks based on code contributions, commits, and pull requests. Integration with calendar tools like Google Calendar and Outlook will allow users to visualize their schedules and deadlines effortlessly.

Ensuring reliability and performance will require the use of various deployment tools and monitoring systems. Cloud-based tools such as AWS Elastic Beanstalk, Azure DevOps, or Google Kubernetes Engine will be used to deploy the system. Real-time monitoring will be handled by tools like Prometheus and Grafana to track server uptime, API response times, and system load. Additionally, the ELK Stack—comprising Elasticsearch, Logstash, and Kibana—will be used to analyze system performance and logs, while tools like New Relic or Datadog will track application performance and identify bottlenecks. For error reporting, Sentry will log and notify the support team of runtime errors, ensuring quick resolution of issues.

To maintain the system's efficiency and reliability after deployment, a robust maintenance plan will be implemented. Regular updates, including feature enhancements, security patches, and bug fixes, will be released on a consistent schedule using a versioning strategy such as semantic versioning. A user feedback portal will allow users to report issues and request new features, which will guide future updates. Automated performance monitoring tools will continuously detect issues such as slow response times and high latency, with alerts notifying the support team of any anomalies. Scalability will be maintained through the cloud infrastructure, which can dynamically scale resources to accommodate increasing user demand. Daily database backups will ensure data is protected in case of system failures, and encryption protocols combined with authentication mechanisms like OAuth 2.0 and JWT will ensure the system remains secure. Regular security audits will also be conducted to address any vulnerabilities.

To promote user adoption, a comprehensive training and support strategy will be rolled out. Detailed user manuals and documentation will outline system usage, key features, and troubleshooting steps. Virtual and in-person training workshops will guide users on integrating the tool into their existing workflows. A dedicated help desk will provide technical support to address any queries or issues faced by the users during and after the rollout phase.

Finally, the success of the deployment will be evaluated based on several key performance indicators (KPIs). These will include reductions in manual time spent on task prioritization, improvements in developer productivity as measured by tasks completed within deadlines, and maintenance of system uptime at or above 99.9% for reliability. User satisfaction will also be measured through surveys and feedback mechanisms to ensure the tool meets its intended purpose. Regular post-deployment reviews will provide insights for continuous improvement, ensuring that the system evolves to address user needs effectively and remains scalable for future growth.

DISCUSSION

There are multiple outlets for future extension for our project. Firstly, the next course of action would be to set up more test cases associated with the code and perform stricter analysis of our current work. Then, we need to create a client-server interaction by centralizing the server that holds our project. For the dynamic scheduler, it would lighten the load with computation associated with each individual user's computer. In essence, this would reduce the stress of any single given computer. We also need to redesign our AI algorithm that decides which task has the highest importance. By certifying that the algorithm works as expected with extensive white box testing and black box testing, we can assure that the sorting of tasks in order of importance will maintain high accuracy. Furthermore, there are a multitude of limitations associated with the project. Firstly, we operated on the assumption that we would not be able to access a central server, as we did not have the time to implement this functionality in the timespan of the project. Then, another limitation associated with the project is the ability to refine our AI algorithm. As none of us students on the project have taken AI classes yet, the refinement of the algorithm to maintain high accuracy while also providing high security may be out of our expertise to implement in a short semester.

CONCLUSION

In today's dynamic software development environments, managing tasks effectively is critical to maintaining productivity and meeting deadlines. Software engineers often face challenges in organizing, prioritizing, and balancing workloads, leading to inefficiencies and increased cognitive load. Our project addresses these issues through the development of an automated task scheduler, designed to optimize task delegation and workflow management. By leveraging AI algorithms, the system dynamically prioritizes and distributes tasks based on factors such as deadlines, task complexity, and individual workloads.

Throughout the project, we have developed a robust framework that integrates cloud infrastructure, machine learning capabilities, and seamless user interfaces. The deployment plan ensures scalability, reliability, and adaptability, enabling the tool to integrate with widely used platforms such as Jira, Slack, and GitHub. We have outlined a structured deployment strategy, from alpha and beta testing to full-scale deployment, ensuring thorough validation of the system's performance and usability. Maintenance and support mechanisms have been incorporated to sustain the tool's efficiency and adaptability over time, addressing evolving user needs.

The automated task scheduler significantly reduces the manual effort involved in task allocation, freeing software engineers to focus on high-value activities. By improving task prioritization and minimizing disruptions, the system enhances productivity, optimizes resource utilization, and fosters a more balanced work environment. Ultimately, the project demonstrates a practical solution to the problem of manual task management, providing an innovative tool that transforms how software development teams manage and prioritize their workloads.

REFERENCES

[1] Trello. "Trello: Organize Anything, Together." Accessed September 27, 2024. https://trello.com/?campaign=18406634139&adgroup=143241824842&targetid=kwd-3609071522&matchtype=e&network=g&device=c&device_model=&creative=633330905204&keyword=trello&placement=&target=&ds_eid=700000001557344&ds_e1=GOOGLE&gad_source=1&gclid=Cj0KCQjwr9m3BhDHARIsANut04ZSVTouMpWjT1TBmoJ31O1RyRv7J1WUsBJfMTqgQUcF57qGTwh_FU4aAo6YEALw_wcB.

[2] Monday.com. "Dashboards-Monday.com Features." Accessed September 27, 2024. https://monday.com/features/dashboards.

[3] Jyothi, N.S., & Parkavi, A. 2016. A Study on Task Management System. *2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)*, 1-6. DOI: https://ieeexplore.ieee.org/document/7764421

[4] D. Stahl, “The dynamic versus the stable team: The unspoken question in large-scale agile development,” WILEY Online Library, Jun. 17, 2023. https://onlinelibrary.wiley.com/doi/full/10.1002/smr.2589#:~:text=The%20importance%20of%20the%20team,approach%20in%20any%20given%20situation. (accessed Sep. 27, 2024).