Digital Kanban Boards: From Inception to Deployment

Team Octo

Rodrigo Pardini Azzolin
Computer Science
Virginia Tech
Blacksburg, VA, USA
rodbot@vt.edu

Aaron Ye
Computer Science
Virginia Tech
Blacksburg, VA, USA
aarony@vt.edu

Wills McGraw
Computer Science
Virginia Tech
Blacksburg, VA, USA
willsmcgraw21@vt.edu

ABSTRACT

The success of a software development project is dependent on a variety of factors: among the most important is long-term productivity. Unfortunately, this important metric is seldom met naturally: poor attempts at establishing inter-developer communication frequently lead to a loss of the sustained health of a project. Practices like Kanban were devised to combat such sources of project instability, and we took it upon ourselves to translate that into a modern, digital context. We designed and developed a digital equivalent of Kanban boards, following the software development lifecycle closely while using Agile as our main methodology. The result is a lightweight application that makes the sharing of ideas and tasks exponentially easier for development teams of all sizes.

INTRODUCTION

One of the greatest weaknesses in software projects and product creation is the lack of clear communication or understanding between team members [3]. Even if an avenue of communication is established like a group chat, it's very easy for progress messages to be buried among the many text messages sent afterward. Remote work is becoming more and more common with software engineering especially, without being in a physical place at the same time ideas and plans can be misinterpreted and the format of having to send messages may affect a team member's willingness to ask for clarification or logistical questions without coming off as ignorant or a bad teammate. This leads to a lack of productivity both due to the slower rates at which members will be actually working on the project and also how certain members may accidentally do similar or the exact same tasks or do something that is only tangentially related to the project. Long-term productivity is one of the greatest metrics of a project's success, and due to these weaknesses, it is very hard to achieve this in a normal scenario without the perfect and perfect product.

But what if we didn't need the perfect team or perfect project to achieve longevity in productivity. One of the common agile practices utilized is Kanban boards, physical boards that use post-it notes like cards to write down tasks under 3 possible sections to help both list and keep track of the status of different tasks whether level of completion, type, or member assigned to. However, as stated above in the realm of software development physical meet-ups are rare, and remote work is more common as members are more likely to stick to their own schedule in the comfort of their own environments. A physical board would not only be extremely ineffective but would also greatly tax one member to maintain.

It is through this limitation we propose the solution of an online web application for the Kanban board. A board that retains all of the base features of a Kanban Board able to be accessed from anywhere and by anyone. Including additional features that a physical board can not have like the ability to change its view mode and built-in chatroom.

REAL-WORLD SCENARIO

If our design were to be fully implemented and incorporated into the real world, it would become a great tool not just for any group work or team projects but especially for software engineers. Although our application is designed to be useful for teams of any scale, the most pertinent example to our use case involves teams on the smaller side: corporate teams have access to managerial roles that alleviate some of the communication pressure, albeit not to an extent that completely solves the issue. The motivating scenario we picture are startups, group projects, and other assorted efforts led mostly by individuals: these groups simply do not have the knowhow of how to organize a project in the long-run, which is where our app will shine.

While group chats and other trivial methods of communication establish a baseline source of coordination in a team, it's hard to keep track of everything that is happening: it's highly unusual for people to log everything

they do within such a space. Crucial information, such as the state of the current tasks, what should be prioritized, and what should be done next are especially difficult to represent in this format of communication. It's no wonder teams lose productivity and create more unnecessary stress and uncertainty. in this state

The Kanban Board application is designed to have multilevel customization, designation, and state control. At any time, a team member will be able to see the level of priority of each task, which task has already been assigned to who, when tasks may be due or the window to do them, and what tasks are completed, in progress, and still needs to be completed. Team members are now not only able to have a clear understanding of the current situation but also can easily plan and communicate the next steps they want to take, as all members can see their edits to the board, and the built-in chatroom will allow easy communication without the need of establishing a separate contact method.

A multi-platform web application would mean that team members can both access and make changes to their Kanban boards anywhere on any device, promoting convenience and ease of use. The Kanban's board ability to also change to a list and timetable allows better viewing of the cards when the numbers become larger amounts, allowing a much easier ability to identify cards by the various sorting possibilities including Due Date, Priority, User, etc. Moreover, this product will have a massive impact because the Kanban board and all of its features will be completely free for all users so that anyone who is working on a software-related project, whether in high school or professional occupation, can utilize and master it.

BACKGROUND

Kanban is a framework of Agile software development that originates from Japan in the late 1940s as people tried to visualize work tasks. Now, it refers to the use of a Kanban board, which is a physical or digital display where all project tasks are listed off on cards that are further subdivided into different categories such as priority, level of difficulty/complication, state of completion to name a few. One of the things that sets Kanban apart from other development processes and Agile frameworks is that Kanban focuses on managing and optimizing work task flow over managing the individual programmers and developers themselves.

RELATED WORK

We are aware that there already exist many online tools and services that help with team projects and product management and development. And among these services include Kanban board services on a web base. However, we do not see this as a limitation but rather a chance to improve upon existing examples. We looked into multiple examples of existing Kanban Board services, including Jira, Notion, Monday, and Trello. We found that some of these services are mainly another project productivity service that incorporated a Kanban feature, which inherently limits the number of features in the Kanban board, like the lack of extra view modes and communication support [1].

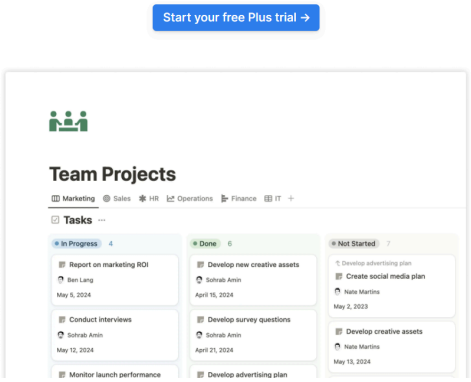

Figure 1 : **Notion Kanban Board Example**

Alternatively, with services purely dedicated to Kanban boards, many of its useful features and functions are locked behind a paywall of monthly subscriptions, which may not be in the budget plan, especially for large groups and projects [2].

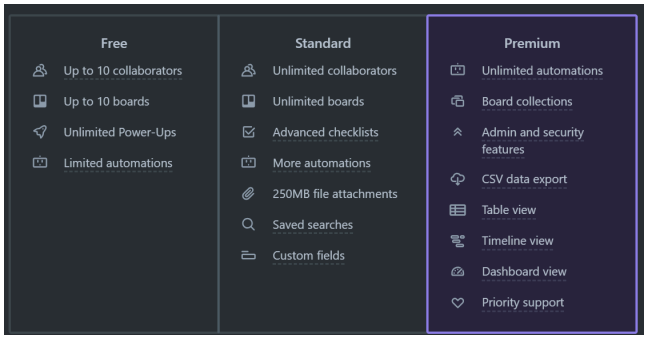

Figure 2: **Trello's "Plans" Page with Different Levels of Pricing**

It is thanks to drawbacks like these from existing products that give us an advantage over other products in the Kanban Board market.

IMPLEMENTATION

Our implementation process started with defining our high-level design. While there were many different design patterns that we weighed the pros and cons of, ultimately

we chose event-based architecture. The reason we chose this design pattern is that event-based is common in microservices and web applications. Even if not technically always the case, our intention is to have our Kanban board able to see and handle multiple actions and inputs at once, which is the core of event-driven architecture. Furthermore, the frames that utilize this design like React are extremely helpful and fast and are definitely a platform we should utilize along with associated tools like MERN and D3. It is also important to note that we will most certainly also make use of client-server architecture, this is because our implementation will require both a backend implementation and front-end implementation, where the backend is in charge of storing and sending Kanban board data and the front end displays the board and handles the actions taken on it and send saved data to the backend.

The design we chose for implementation is behavior-driven development. This is because while we initially wanted to utilize test-driven development, the multi-layered nature of our product utilizing multiple tools for both its frontend and backend would require us to write multiple types of unit tests for different programming languages and software, sometimes having to figure out even how to write one as certain tools simply do not have the capacity to incorporate unit tests into them and will have to have them built around it. To account for this instead of specific unit tests we will instead define a certain behavior we wish our product to achieve with specific requirements for factors like load time and/or process integrity.

When it comes to the specificity of implementation, we will most likely use a tool like React for our front-end implementation, as its event-driven model aligns with our own high-level design and it can utilize both tools like D3 and MUI/MUI X to dynamically create beautiful Kanban boards that immediately respond to live updates. In addition, the good practice of “MVC” in React is extremely beneficial to our application. MVC or “Model-View-Controller” is the design format that separates the model/data from the viewing objects with a controller in the middle, essentially making it so that neither Model nor View have direct access nor control of the other, and instead refer to the controller to both handle events/inputs and return values for either viewing or updating the database. Since there will be many actions that take place on our web application, having a controlled design like MVC will be very beneficial and will minimize the possibility of fundamental errors and keep a clean line between front-end and backend implementation without having the interjections become too plentiful and messy. If we were to go through with React as our front end then our backend must be supported by tools like MERN stacks, which incorporated React with other middleware and back-end software like Express and MongoDB. These backend tools will be able to efficiently store user data and also retrieve and store them between the frontend and backend.

For testing, we would utilize a combination of multiple testing strategies. First, we want to make sure our product at least meets base requirements by utilizing a combination and modified version of unit testing and integration testing. The idea is to have developers test the functionality and integration of separate parts utilizing white box tests to make sure everything is at least functional. However, we understand that as developers we will have invisible biases on how to utilize our product, so the next step is to perform validation testing from independent testers utilizing black box tests with the main goal of attempting to break out the product either through edge cases, overloading actions, or a combination of both. Only through these two parts of testing can we ensure a successful product without overtaxing either testing party.

DEPLOYMENT & MAINTENANCE

In terms of deployment, a plan is already in place. The idea is to utilize servers as back-end data storage for all Kanban boards, and a web application front-end that will take that data and convert it into the associated Kanban board. This format is picked both for its straightforwardness and the technical portability to the mobile platform, as all smart devices are able to access the internet and our application will be able to understand the kind of device utilized and attempt to match its configuration.

In terms of maintenance, we used the four concepts taught in class: the four types of software maintenance: Corrective, Preventative, Perfective, and Adaptive. When it comes to corrective maintenance, the first step is to perform heavy testing on our Kanban boards before release to catch as many bugs as possible. However, we know that no matter how hard we try to find bugs before launch there will always be new ones found by our customers. To accommodate for that, a forum will be set up so that users can send their questions there to be answered by a developer or at least given a temporary solution by a fellow user. There will also be an email address set up for those who want their problems fixed in a more private manner. Our maintenance team will periodically post updated logs on our web page that will not only cover any bugs that have been fixed but also any bugs they are aware of and are in the process of fixing. The goal is to establish some form of “communication” between users and our maintenance team so that in addition to our own bug testing, new bugs can be found and properly addressed.

Unlike corrective, preventative is about bug prevention instead of correction. To that end, it will be part of our original implementation that is continued into maintenance as new features, security updates, and other forms of upgrades will always be required. The idea is to have the code implemented with high integrity, containing mane safety nets even if they might be viewed as “unnecessary,” the idea is that even if we are not able to account for every single error possible to least have nets those are able to course correct in some fashion so that a bug will not lead to a shutdown of the back or front end. As mentioned above this is also applied over to any new updates in the future, whether it is a minor bug patch or a major security update, our update integrity will be held to a high level so that hard crashes are prevented as much as possible.

One of the last things we want to do is unnecessarily change the UI and structure of our Kanban boards and call it “software improvement/relevancy.” To that end, Perfective maintenance is covered via our feedback form and usage analysis. In addition to bug reports, users can send developers non-bug-related feedback, whether it’s about their level of satisfaction with the product or a certain desired feature that does not exist in the product as of yet. We also understand that the phrase “usage analysis” may make some users uneasy as if we would be tracking the exact activity of our users. Rest assured it will be the basic ability that all web hosts have to identify the level of activity going into the website, which will help us identify trends of usage and give us an early hint when it’s time to implement new features. With these two tools, we hope to be able to identify good features to add and slowly but securely integrate them into our Kanban boards for users to enjoy.

Lastly, we aim to implement adaptive maintenance by keeping an open mind about the amount of usage of certain resources. As Computer Scientists it is our duty to keep our ears on the ground and in the “clouds” to make sure we’re getting the latest information regarding technological breakthroughs and availability. When new technologies come out, our teams will immediately jump to create a version of our product that utilizes that technology, making sure to follow our previously established maintenance rules to build both a stable and thoroughly tested model before uploading it to the live version.

FUTURE EXTENSIONS

Even with our extensive planning in both implementation and maintenance, there are still limitations our product adheres to. One of our biggest features that also makes us different from existing Kanban Board tools is we intend to make our product completely free of use with no secret monetized bonus level of premium service so that it can be utilized by anyone, however, such a model means that we will have no method to cover the cost of server maintenance especially if with great amounts of accounts and data thanks to our free status.

However, we don’t ever intend to change our level of monetization so instead as a form of future extensions, we are entertaining the idea of giving server maintenance and hosting responsibilities to the user. Essentially, a win-win for many individuals. As now users will be able to physically own the data of their own kanban boards without reliance on a remote server outside their control, and this will give us the resource breathing room to focus on other maintenance-related tasks like bug fixing and new feature implementations.

CONCLUSION

Our initial goal was to address the issue of lacking clear communication in software design and development teams, especially so in teams that work remotely and do not have the advantages of physically seeing each other and communicating in the office. Ideally, addressing the issue of inefficient communication would lead to project productivity, longevity, and quality.

Throughout the course of the past few months, our team has worked to design a digital Kanban Board application that would allow for developer teams to optimize workflow and limit inefficiencies. Our process led us through requirements elicitation, requirements specification and analysis, high and low level design, and testing. We are currently at a point where we can look back happily knowing

Moving forward with the Kanban application development, we would start creating functional models and iteratively redesigning until we reach a point where we are happy with the final product that fulfills the project requirements and functionalities. We would then start deployment and maintenance, with occasional updates in response to feedback and bug reports.

Overall, we were able to effectively work through the software design and development process with few issues, allowing us to create an application with a foreseeable future and next step. We have created a product that allows for easier communication of tasks and issues amongst developer teams. With this improved communication network, project productivity and product quality. In all, we worked to assist others with the software design and development process by working to better the shortcomings and flaws that come with communication, especially so in remote environments, while reinforcing the strengths of productivity.

REFERENCES

[1] Anon. The best tool for project management. Retrieved December 17, 2024 from https://www.notion.so/lp/pm/project-management?utm_source=google&utm_medium=cpc&utm_campaign=Search_US_NB_ProjectManagementPhraseBroad&utm_term=free+project+management+software&utm_content=Software&gad_source=1&gclid=Cj0KCQiAvP-6BhDyARIsAJ3uv7YrgEFA-2uXcHr1k_jLtn7mS11XvgJIPkFBCwzq8CLWEVLTJTfTnz8aAnMzEALw_wcB

[2] 2024. Trello. New York City, NY, USA, 2 pages. https://trello.com/

[3] Swaminathan Iyer. 2024. Why effective communication is a must-have skill in software development: | interview kickstart. (September 2024). Retrieved December 17, 2024 from https://interviewkickstart.com/blogs/articles/why-effective-communication-is-must-have-skill-in-software-development