Todo Application

Ashish Bhat
Department of Computer Science
Virginia Tech
Blacksburg, Virginia, USA
ashishbhat@vt.edu

Disha Bhan
Department of Computer Science
Virginia Tech
Blacksburg, Virginia, USA
dishab2124@vt.edu

Sahana Basapathi
Department of Computer Science
Virginia Tech
Blacksburg, Virginia, USA
sahanabs@vt.edu

1 ABSTRACT

Time management is a virtue possessed by select few. Software engineers especially need to learn this because of the amount of work they are assigned on a normal day in the office. A lot of software engineers struggle with managing their tasks. They get overwhelmed and sometimes take hours to prioritize and properly complete their work. A plethora of tasks make these software engineers end up forgetting or delaying them because of their sheer amount. In addition to this, people with attention deficit hyperactivity disorder (ADHD) also find it difficult to focus on one task at a time. Our todo application helps people with this condition as well as software engineers in organizing their thoughts by prioritizing their work. It also helps categorize the type of work which gives the user a coherent idea of the tasks remaining. The todo application also helps the user in decluttering their thoughts and providing them with a sense of achievement, helping them improve their productivity. The application is built in order to create, modify, assign and delete todo lists.

2 INTRODUCTION

Since the pandemic, work from home has given software engineers the opportunity to work in an unsupervised environment at their own pace. However, this convenience also caused many inconveniences to software engineers with the most prominent being task management. Software engineers have many meetings and upcoming deadlines for their work, and since they are not in an office environment it gets challenging to keep track of all the things. Hence there is an inherent need for a todo app that will help software engineers organize their tasks in a systematic way so that it can maximize their productivity and prevent them from missing any of their tasks or deadlines. This problem is worth solving because efficiency is an important part of any software engineer's work and if they are not motivated and satisfied by their work, it can negatively impact their productivity which may in turn decrease their work performance. Employees, especially software engineers should be constantly motivated because they have to deal with a lot of rigorous and redundant tasks such as finding bugs, meetings, and receiving feedbacks from the client. Todo lists are important for anyone looking to accomplish various tasks within a set period. In a nutshell a todo app offers systematic management of tasks which helps in improving the productivity of people especially those who have ADHD. It helps in providing a sense of completion which in turn motivates people to complete their other tasks as well. And finally it helps in meeting deadlines, prioritizing and categorizing the work.

3 MOTIVATING EXAMPLE

The mode of operation for many software engineers has been changed to work from home (WFH) during COVID-19. However, this has created another problem for them. In addition to completing their office work on time, they also have to take care of other tasks at home. Often, these mundane tasks can distract software engineers from their office work which may result in wasted time and/ or missed deadlines. To counter this problem, a todo application is developed by us which not only helps software engineers keep check of their pending tasks but which also helps them categorize these tasks according to priority and category. This further allows software engineers to systematically manage their tasks and track their progress.

4 RELATED WORK

Lunatask.app is an all-in-one privacy-focused todo list that arranges tasks depending on their age, priority, and expected time to accomplish the task. [1] proposed a task-management system that predicts a user's future state on the basis of the user's lifelog and plans, using a simple linear regression model. Twelve users who used the system answered that they tried to increase time spent on tasks and decrease time on leisure. This shows the potential of task management using personal lifelog-based prediction. [2] proposes a Taskmaster system to narrow the gap between users and email tool features. The Taskmaster achieves this by recasting email as task management and embedding task-centric resources directly in the client. Their research shows that it is possible to significantly and positively affect email users' experience by embedding task management resources directly in the inbox, where they are most needed, as well as breaking down the barriers between the various components of contemporary email applications. However, there is much work left to do to perfect this vision. An online task management system is designed in [3] which works on effective notification management. The college staff using the application is able to notify every student about recent events or important activities taking place in college. [4] publishes a study to see what types of task management demands a task list manager system could support. They claim that the main issue with task management is not a lack of prioritization, but rather the work necessary. They detail the resources and procedures that people employ to guarantee that they are successful. They create TaskVista as a solution to lessen that effort based on this feedback and study. We used the

feedback from users in this article as a guide for developing our application. This was also the base upon which we augmented the idea of grouping the tasks according to categories.

5 IMPLEMENTATION

5.1 ARCHITECTURAL COMPONENTS AND THEIR INTERACTIONS

Model-View-Controller (MVC) architecture has been used to structure this system. This application has a user interface (UI) to interact with the users along with storage and retrieval of data. In MVC architecture there are three components: model, view, and controller and these are built to handle specific development aspects of an application. It helps with faster development as the programmers may work parallel to develop several components. For instance, for a todo application system, one programmer may work on the view while the other may work on the controller to create the business logic for the system. With this, scalable and extensible projects can be made. The architecture we used for this project is detailed in the below figure. The components are given below:

Model: Corresponds to all the data-related logic that the user works with. We have used the postgreSQL database since it is a free, open-source, highly customizable database. PostgreSQL also has great support for JSON data. There is also JSONB data type added, which improves the indexing ability. It takes the query as the inputs and returns the task lists accordingly.

View: The view component contains the frontend or the UI logic such as displaying a to-do list, having an editable view for each to-do task, and interface for creating a new task. Appropriate view is fetched according to the task requested by the user. We also have an activity feed to check other user's activities.

Controller: The controller is the main component that bridges the gap between the model and the view component by processing the input and formulating outputs. It controls the interaction between the model and the view, where the view calls the controller to update the model. A view can call multiple controllers if needed. Here we define the business logic. Eg, filter the task list according to the user, check if the user has appropriate permissions to change and update the tasks.

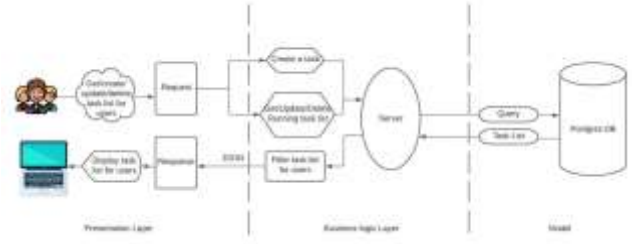

Figure 1: High Level Architectural Design

5.2 CONSTRAINTS OR GUIDELINES

Some of the constraints we encountered in building the software were in the usage of Google Calender API. This Calendar API had a limit of 1,000,000 queries per day. This was a constraint since this feature of the web application was used a lot. It also created a point of dependency on the working of the API. A failure of the API could make the feature display an error message. In addition to the API constraints, we used Django framework to build the web application. There were some aspects of Django that were hard to optimize. For example, we would always have to wait for Django when it was running requests through middleware, serializing and deserializing JSON strings, converting database queries into Python objects and running garbage collection, etc. This made the application a little slower.

5.3 ADDITIONAL DESIGN PATTERNS

Creational design patterns were useful for implementing this project. We could create only one instance of a class and provide only one global access point to that object. This could ensure that there was one point of access to the to-do list class, and reusability of code. Like google calendar, a todo list works the same independent of the devices though which it is accessed (global access to the object), thus ensuring consistency of data. In the creational design pattern, we used the singleton pattern which is a software design pattern that restricts the instantiation of a class to one "single" instance. This is useful when exactly one object is needed to coordinate actions across the system. Builder pattern was also considered for use by us. This was because a todo application would require the creation of many tasks and these tasks would have the same code i.e., creating a task for meeting preparation would not be different from creating a task for studying. This would increase flexibility as a lot of the code will be the same and only the semantics of the task would change. The APIs, task creation, editing, and deleting would be the same for all the todo tasks in the list.

5.4 DESIGN SKETCHES

With the screenshots below, we detail how to navigate and understand the flow of the application.

Figure 2 shows the snapshot of the dashboard of the application. It consists of the task, along with the category, respective priority, who it is assigned to and the due dates. There is also a list of completed tasks. This view can be seen by clicking on "Dashboard".

Figure 2: Overview of Dashboard

Figure 3 lists the task details. This can be accessed by clicking on the respective task from the dashboard.

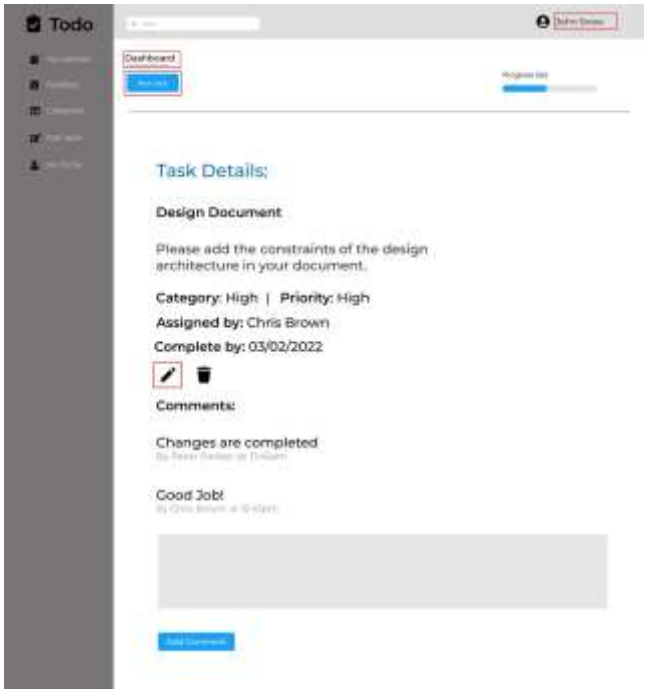

Figure 3: Details of a task

Figure 4 is the screenshot that is seen by a user to edit a task. Any task can be edited by using the edit button from the task's detail's view.

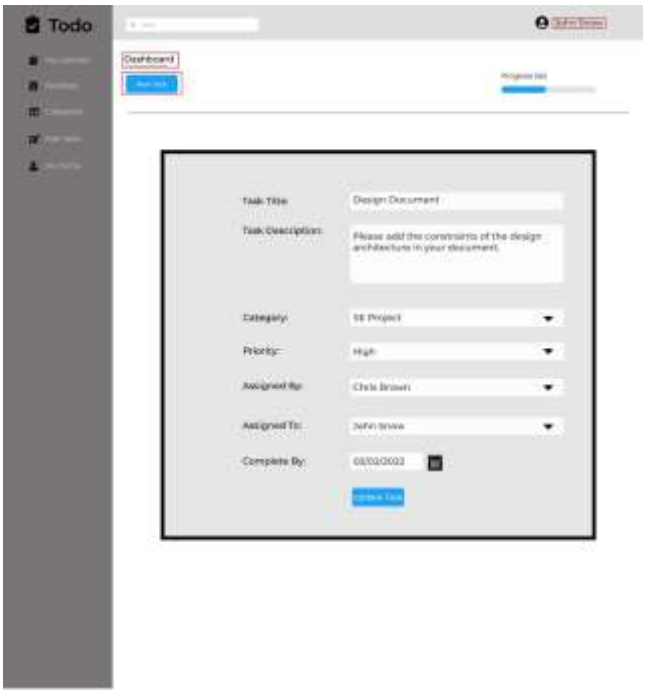

Figure 4: Updating the task

Figure 5 shows the profile information and the activity feed of the user. The activity feed shows tasks shared with the user, user's concurrent tasks to be completed, etc.

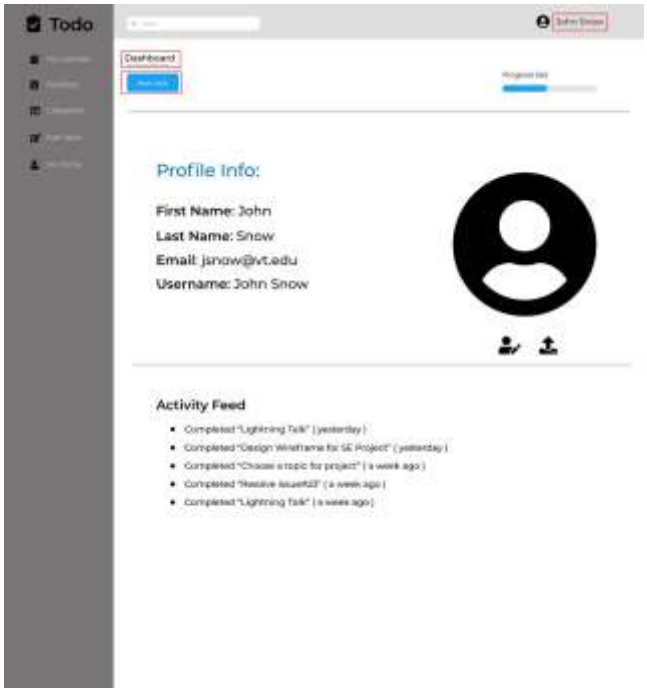

Figure 5: Overview of the profile of a user

Figure 6 shows the form that a user sees to create a new task.

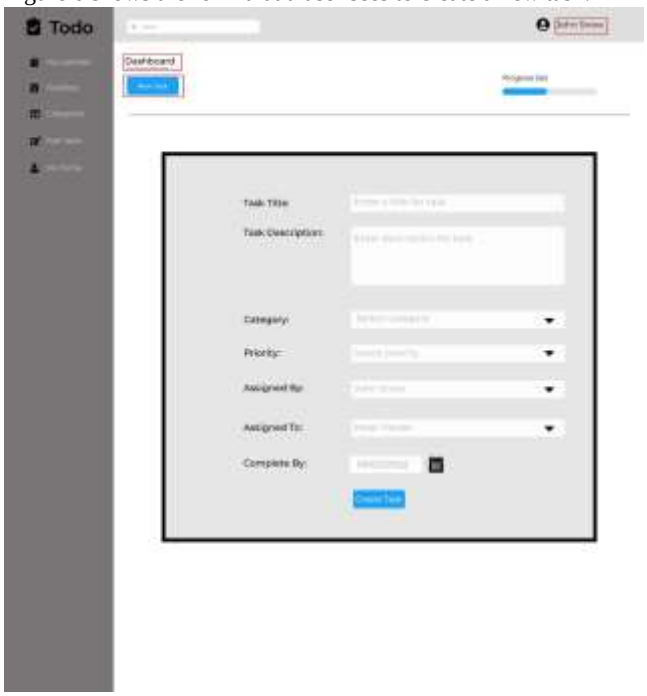

Figure 6: Creating a new task

5.5 DESIGN IMPLEMENTATION

The screenshots attached below show the implementation of the application in real time. The use cases for each image are also explained.

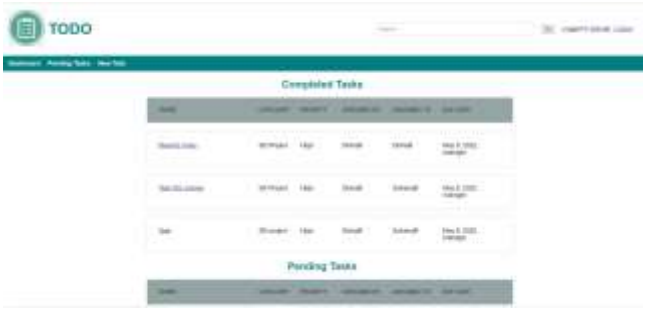

Figure 7: Completed tasks for a user

This page shows the tasks that have been completed by the user.

Figure 8: Pending tasks for a user

This page shows the tasks that are pending for the user.

Figure 9: Dashboard for the user

This page shows the dashboard containing the tasks for the user.

Figure 10: Task assigned to a user

This page shows the task that has been assigned to a user.

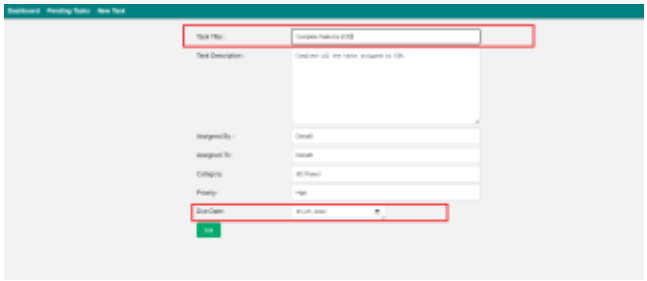

Figure 11: Task edition page

This page shows the form to edit the tasks.

Figure 12: Task assigned to the user along with comments

This page shows the task assigned to a user along with comments from the assigner.

Figure 13: Completed and pending tasks for a user

This page shows the completed and pending tasks for the user

Figure 14: User profile

This page shows the user profile for the user which can also be edited.

Figure 15: Adding a new task and assigning to a user

This page shows the form for adding a new task and assigning to a user.

5.6 SOFTWARE PROCESSES AND TESTING

Kanban and scrum were strategies used while developing our application. Adaptive software development process was used with an agile spirit. White box testing was performed to test our software. Unit tests and system/ integration tests were performed as part of white box testing. Alpha and beta testing was also performed as part of validation testing where our friends were asked to use the software with us present and without us present and to note down all the problems encountered at a regular interval.

6 DEPLOYMENT PLAN

The software would be deployed to the end users using basic deployment and the process would consist of delivery, establishing a support regime and garnering feedback from users. This deployment would be simple, fast and cheap. It would also be the perfect deployment strategy for us because our product is not critical in nature. Appropriate instruction materials would be provided along with the software. Our software would also need to be maintained such that user problems are addressed after operationalization. We would be providing corrective, preventative, perfective and adaptive maintenance for our software. To expedite this process, we would be equipping our software with proper documentation, unit tests and integration tests.

7 DISCUSSION

The server and database used for our project are not equipped to handle a huge load in case user traffic increases beyond expectations. In case we deploy the software on a larger scale, we would need to upgrade our server to an enterprise server (IBM) or something along the same lines instead of Heroku which is being used right now. PostgreSQL used for the database would work well with both small and large business loads. Though we have improved upon existing related works and incorporated categories and priorities for the tasks into our applications, it goes without saying that there is still a huge scope for improvement in the future. The user interface for the application can always be improved in the future according to the capabilities of the frameworks at that time. User feedback will always be used to incorporate changes into the application at regular intervals.

8 CONCLUSION

The problem we were trying to solve using this application was that of task management. Software engineers have many meetings and upcoming deadlines for their work, and since they are not in an office environment it gets challenging to keep track of all the tasks. Hence there was an inherent need for a todo application that would help software engineers organize their tasks in a systematic way so that it could maximize their productivity and prevent them from missing any of the tasks or deadlines. This problem was worth solving because efficiency is an important part of any software engineer's work and if they are not motivated and satisfied by their work, it can negatively impact their productivity which may in turn decrease their work performance. We were successful in creating such a todo application which not only helped software engineers manage their tasks, it also provided additional functionality to them including tracking their progress, assigning priorities and categories to their tasks in order to ensure proper scheduling of time and calculation of the estimated time left. Our application is important for anyone looking to accomplish various tasks within a stipulated period. Through its systematic management of tasks which helps people in improving their productivity, we can also make this application available to people who are not software engineers but would also benefit a lot from it, like people with ADHD.

9 ACKNOWLEDGMENTS

We would like to acknowledge Virginia Tech for providing us with the resources to complete our project on time. We would also like to thank Prof. Dwayne Brown for his time, guiding us throughout the semester.

10 REFERENCES

[1] Toshiki Takeuchi, Kyohei Suwa, Hiroto Tamura, Takuji Narumi, Tomohiro Tanikawa, and Michitaka Hirose. 2013. A task-management system using future prediction based on personal lifelogs and plans. In Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication (UbiComp '13 Adjunct). Association for Computing Machinery, New York, NY, USA, 235–238. https://doi.org/10.1145/2494091.2494166

[2] Victoria Bellotti, Nicolas Ducheneaut, Mark Howard, and Ian Smith. 2003. Taking email to task: the design and evaluation of a task management centered email tool. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03). Association for Computing Machinery, New York, NY, USA, 345–352. https://doi.org/10.1145/642611.642672

[3] Swapnil Kumbhalkar , Shubham Meshram , Grishma Hedaoo , Raksha Tabhane , Priyanka Thoke; Prof. Mukesh Barapatre "Online Task Management System" Iconic Research And Engineering Journals Volume 2 Issue 5 2018 Page 69-74

[4] Bellotti, Victoria & Dalal, Brinda & Good, Nathaniel & Flynn, Peter & Bobrow, Daniel & Ducheneaut, Nicolas. (2004). What a to-do: studies of task management towards the design of a personal task list manager. Proceedings of CHI, Vienna, Austria. 24–29. 735-742. 10.1145/985692.985785