The “A” List

Authors: Ahman Hale, Camden Johnson, Philip Clement, Tanay Khopey, and Jonathan Hsin

Abstract

This report outlines the process of designing a Todo list application. The primary motivation for this application was the stress and workload of being a college student. This project drew inspiration from other common Todo list applications to come up with a basic outline. The implementation part of this project was split into five different phases, where different parts of the application were developed. The first phase consisted of figuring out the main tasks and goals the project needed to complete. The next phase was where a low-level prototype was created to test the features of the program and ensure the logic made sense. The third phase focused on the user interface aspect of the application and making it user-friendly to use and look at. The fourth phase simply worked toward combining the last two phases and connecting the front and back ends. The final phase was testing to ensure there weren’t any other errors from the previous phases and to see what could be added or improved upon. After the implementation, creating a plan for how to deploy this project was made. It would continue to be tested and improved upon using an agile methodology, but it would also be ensured the core structure of the program would stay the same. Use cases and blackbox testing would be used to discover new features that could be added. Overall, all the parts of the project led to creating a Todo list application that is user-friendly, easy to maintain, and easily upgradable.

Introduction

Everyone has tasks they need to complete in a day whether it be for work, school, or just chores it can be stressful to remember everything you have to do. This is why todo lists are so popular, because it takes away the stress of having to remember. This is why this report will outline a user-friendly todo list application. This application was made by college students who understand the stress of having to juggle many different tasks and assignments and how having a simple todo list can make life much easier.

Our aim for this project was to create a to-do list application with students and software engineers in mind. We wanted to make sure it was user-friendly, but also easily maintained. Additionally, we wanted to ensure new features could be added without changing the overall functionality of the program. This is because we want to be able to add new features when the need arises, but also keep our app as simple as possible. In order to achieve these goals, we had to follow various software design processes to adequately create this application. Once the idea was formed and created, we then had to develop a plan to develop and deploy this application using iterative and agile processes.

Motivating Example

There are many different scenarios that could be applied to how our Todo List App helps organize a person's day. As students with a lot on our plate, it is only natural that we choose a scenario related to being a student. So for our motivating example, imagine a student at Virginia Tech studying Computer Science. We are choosing this persona because every member in our group will understand the following predicaments. The student has a project due for his Data Structures class, he has to give an oral presentation for his Public Speaking Class, he wants to exercise more to get his body in better shape, and on top of all of that he has to walk and feed his dog two to three times a day. As you can tell, the student has a lot on his plate and is having trouble keeping his day organized and remembering to keep up with deadlines.

That said, many college students have schedules that are busy just like the student we described. This motivated us to make a todo list app that not only helps students like him keep their days organized but also easy to use.

Another motivating example that is related to how our app is relevant to how software engineers work together is imagine a group of software engineers who were just tasked by their project manager to make a social media app from scratch. The use of software design processes will obviously be used by the group but there will also need to be a requirement for all group members to be organized and be able to complete their parts in a timely manner. In comes our Todo list app. The app can be used by each software engineer to organize what tasks they need to complete regarding the project they are working on. The app can also be used by each engineer in daily stand-up meetings. Software engineers are known to follow project management methodologies which usually include daily stand-up meetings. The daily stand-up meetings include asking each member three things: 1) What they have done, 2) What they are doing, and 3) What is blocking them? The todo app can allow each engineer to easily remember and articulate what they have done (1) and need to do next regarding the project (2). Therefore, this scenario goes to show how our Todo list app's role in streamlining project management processes and ensuring that each member can efficiently track and report on their progress is relevant to the role of being a software engineer.

Related Work

To-do apps have been created before by large companies, each app has had its own unique features. Our app will be no different, as it will not be a copy of any one app. Rather, we will draw from the already existing apps as well as our team’s creative ideas to make a to-do list app that 1) meets the client’s requirements, 2) includes some original, unique features, and 3) is completed in a timely manner using the Prototyping software engineering process. Examples of to-do apps that have been prominent in the industry include, Microsoft To Do, Google Tasks, and Monday.com. These apps have unique features that include the base features of a to-do list app and then more. Monday.com specifically has features for certain professions and different sections for the various methodologies of project management. This is something that we draw inspiration from since we are students in a software engineering and design course. Google Tasks also has some features we would want to include in our app including, but not limited to, being able to set due dates on certain tasks with a feature that lets you set times also. Microsoft To Do essentially has the same features. The common theme we see in all these existing apps is that they do not have many, if any, unique features. Essentially, they all copy each other’s features making a similar user experience but with a different user interface.

Implementation

Our implementation was done in five stages: planning, logic design, UI design, integration, and testing.

During the planning stage, we created some functional and non-functional requirements that acted as goals that our application should reach. Regarding functional requirements, we wanted our application to have modifiable font size and color for readability, autosaving functionality, allow adding and deleting items, change order or items, and be able to sort the list items automatically based on name or time added. The non-functional requirements, on the other hand, include the ability to add as many items as needed, respond to commands in less than 2 seconds, have an application that takes a small amount of storage space, readability, and use Python. Based on these requirements, we moved on to the logic design stage.

During the logic design stage, we implemented the to-do list application with only a text interface using Python because we wanted to reach all of the functional requirements in this stage and perfect the application's logic before working on how the application looks. As we worked on the application, we made the main flows and sub-flows of the application concrete for functions such as adding, removing, and swapping items on the list. This is also when we tackled edge cases such as an input of a list ID that is out of bounds, in which case we remind the user to put in the right ID. However, while the three aforementioned functions were implemented, we found that the other functional requirements like autosaving and automatic sorting would be too difficult to include in our application, so we opted to leave those out for possible future iterations. After completing the logic design stage with all of the functions implemented textually, we’re able to move on to the UI design stage.

During the UI design phase, we first explored the PySimpleGUI library to understand how to make simple windows with buttons and input boxes. Then we designed how our application window would look based on the functionalities we wanted to support. We added different buttons and views that would dictate how each user should interact with the application to achieve certain tasks. This is also the phase where we decided that to support simplicity, the button layout would be straightforward and avoid any submenus that may lead to difficult navigation. After the UI design was finalized, we were ready for the integration phase.

After the logic and UI design phase, the integration phase was relatively simple and we just plugged the code for the logic into the code of the UI to fully support the buttons as we envisioned. After the application logic was fully integrated, we found that more functions could be added to the code, so we included a new, but needed, “history” function in our application which allows users to view the tasks they have completed and removed from the list.

The last step of our implementation is testing the application. We opted for black box testing not only because it was required for the project milestone but also because it supports users better as they would not need to worry about how the works, just that they can do their desired tasks. For the black box testing, we tested all of the functions we support to ensure that they work as expected as well as edge case tests to ensure that users would not break the code if they made a mistake like typing out a word instead of a number or typing an ID that is out of bounds. After making sure our test plan is robust and the application returns the expected results for all test cases, the application is fully implemented and ready to be deployed.

Deployment plan

To deploy and maintain our project, our group would first adopt an agile approach which would allow us to iteratively make changes and keep our application running at a high level. We would either use Scrum or Kanban as our Agile frameworks to oversee the development process. This would make it so we can incorporate a continuous flow approach. We would also use the prototyping model to deploy and maintain our application. Utilizing the prototyping model allows us to create multiple iterations and test new features often. By deploying multiple prototypes we can gather feedback early in the development process and make changes to our application accordingly. In addition, prototyping would allow us to maintain a user centric approach meaning that we can really focus on how our application and our users interact.

We would also create multiple use case scenarios to further help us define the various potential interactions between the user and our application during the deployment phase. By outlining different user goals, such as adding, deleting, or organizing tasks on the to-do list, use case scenarios provide a structured framework for designing and then implementing features the user needs. Plus, creating multiple use case scenarios would help us prioritize development tasks.

When we are ready to test our application, we plan to use black box testing not only during the deployment phase but when maintaining our application as well. Using this testing method would allow us to validate our application’s functions from a user perspective.

To maintain our application, we plan to use the refactoring process. With this process, we would make changes to our code for improvement without changing the underlying functionality and behavior of our program. This is a project where we can improve and modify with more time, so using this process would help us maintain a level of structure that is needed to maintain a working application with changing requirements.

Finally, to further maintain our application, we would have code reviews often. By doing so, we can inspect our code frequently to improve our software quality and catch potential bugs or errors. Code reviews also serve as a way for us to ensure that our requirements are met.

Conclusion

Overall, the A list transcends mere task management and has become a cornerstone of productivity for individuals across various domains, from tech professionals immersed in the fast-paced world of technology to students navigating academic demands and beyond. Designed with a comprehensive feature set tailored to facilitate efficient workdays, the todo list empowers users with functionalities such as task addition, removal, and rearrangement, alongside a history feature offering insight into completed or removed tasks.

Our commitment to delivering a robust and user-centric solution was underscored by a rigorous testing regimen. Beginning with in-house unit tests, each component underwent meticulous testing to ensure its individual functionality met our exacting standards. Subsequently, our final testing phase employed black box testing, rigorously evaluating the app's performance in real-world scenarios to validate its reliability and seamless operation.