Dasha Asienga, Lynca Kaminka, and Priya Bakshi

Functionality: 18/20
Our program is very functional in terms of the keyboard inputs and storing the user's data in files to re-read the files back into the calendar. The files are updated during runtime depending on what users delete and add on the calendar so that it's constantly updated and the app responds immediately to any input. Something that could have been improved on are the buttons that are shown on the DayView. Because of the package we used, the pane holding the timetable view sometimes comes forward and hides the buttons because of the way the calendar is added to the content pane. These buttons are still functional: one can either go back to the main page (top left button) or the month view (top right button), but we were unable to find enough documentation about the pane structure of the calendar. Given more time and research, we would be able to figure out how to change this so that the buttons always stay at the front and always show up.

Design: 18/20
Our calendar is designed well. Each of the tabs are easy to follow and allow the user to change between four theme colors for their calendar. Additionally, the to-do list has optimal space for the user to input their tasks. One design flaw is that the timetable for DayView on the calendar can sometimes be difficult to read. Additionally, the buttons that are printed on each page could be more aesthetically pleasing and the theme that is updated on the calendar could also be updated to each tab of our app rather than just being shown on the calendar. Essentially, we could've made more use of our coding time to better the graphic design of our app but, the flow is really good and a user is able to access many different tabs by option.

Creativity: 16/20
The creativity of our calendar is pretty good. We created a more functional project and coming up with the different tabs and the robust use of file input and output shows the creativity that went into creating this program. Although adding more aspects was creative on our part, a calendar project is a generally common app that has been previously created.

Sophistication: 20/20
The sophistication of our project comes mostly from the file creation and reading it back into the calendar based on different users – file input and output. We allow the users to create multiple accounts on one device as well as keep their information constantly synced even when they close the program. We use file input and output to store user data, create new files for each user, and write to and read exclusively from those files. Another area of sophistication in our program is using a package that isn't well-documented and understanding how to use it in our program to get our final product. It took hours to find a useful package, and then afterwards took days to understand it. In addition to the internal structure of our app as well as the numerous frames that are all seamlessly synced to each other, we believe our functional app is very sophisticated and

something that we're proud of. Given more time, we could implement even more features but as it stands, it works very well as a calendar.

Broadness: 20/20

We ticked boxes 1, 2, 3, 5, 6, 7, and 8. We used an external Java library to create the grid of our calendar and used some in-build functionalities to perform certain functionalities. We also used Java awt libraries and event libraries, to name a few but there are many imported classes that were vital to the success of our program. We used subclassing by creating a parent Calendar class that both the month view and the day view extend. We used the interface 'frame' which was implemented in many classes to set the frame visible based on which page the user chooses to view. We used LinkedLists and HashMaps as our main built in data structures to store all of the information inputted on the calendar into a file that was read when the program was rerun. We used file input and file output to save the users information and re-enter their previous information every time they open the program after exiting. We verify that the log-in information matches before allowing the user access to the app. This makes it so the user is always able to have their calendar information updated and avoid having them re-enter it every time they run the program. Lastly, all the in-build data structures we used were generic. We randomize the welcome message each time a user logs in.

Code Quality: 18/20

Our code is pretty well documented for this project. Our code is well commented so that whoever is reading the program code is able to follow what each aspect of the program means. We implemented many different classes and frames to make the code easy to follow and avoid having a lengthy amount of code in Main.java. For that reason, our Main class is pretty empty and simply launches the log-in page. There was a lot of trial and error done in order to achieve the final project but the final code is what was left in the program. We could have done a better job keeping detailed track of what each class and method does.