Development Practices Attempted

- Separation of tasks:
    - Every team member is primarily responsible for one page of the app, ensuring a clear delineation of tasks. Other tasks are split evenly with minimal components that require cooperation. Sherlock is responsible for developing the Calendar page, adding the daily calendar to the Home page, and recording meeting minutes. Amy is responsible for developing the Summary page, implementing the remainder of the Home page, and writing release notes. Sean is responsible for developing the Todo page, constructing databases, and packaging installers. Jude is responsible for the development of the Notes page. The separation of tasks provides each team member with well-defined scope of work, fostering a clearer understanding of their individual contribution to the project.
- Followed sprint iteration.
    - On the kickoff day, the team meets for 30 minutes to establish new issues for the sprint. Each team member will brainstorm tickets for their designated section. Subsequently, all team members will review the tickets together to evaluate the urgency, assign priority rankings, and ensure an equitable distribution of workload among all members.
- Used Gitlab flow to organize commit history and collaborate
    - Every team member has their feature branch to work on their part of the project. As a member commits changes on a feature branch, tests run automatically on that branch. This facilitates a swift turnaround, allowing for timely recognition and resolution of errors.
- Practiced pair programming:
    - Sprint 1 & 2: driver and navigator pair programming between Amy and Sherlock, Sean and Jude. For these two sprints, the team primarily focused on UI design and displaying data from a hypothetical database class. Each team member has designed a layout for the page they are responsible for. During the pair programming, the navigator will ensure that every component uses the most suitable container (ex. Box inside LazyColumn), and the driver will implement the component and adjust details such as size and color.

    - Sprint 3 & 4: ping pong coding between Amy and Sherlock, Sean and Jude. These two sprints primarily focus on transitioning to SQLite and web service. Every team member has to fetch data from the database and potentially save data to the database. These two sprints involved many edge cases and team members' class schedules became busy. There is less time for team members to gather and code together, so the team decides to use ping-pong coding where one person designs several test cases for their coding partner's section.
- Code Reviews
    - An informal code review is conducted between Amy and Sherlock, Sean and Jude. The team used the merge request functionality on Gitlab to ensure that the app was functioning as expected before being merged into the main branch.
    - A formal code review is conducted before creating the installer. Team members will add unit tests to each others' code to ensure that all components have edge cases covered. After a component is fully tested, team members will also close issues on Gitlab together and create the release note.

Future Areas for Improvement

- In the future, our team should reserve more time for unfamiliar tools and tasks. For example, our team underestimated the time to build a web service. Consequently, we did not have enough time to have 100% unit test coverage for sprint 3. We wrote our unit tests in sprint 4 and discovered several bugs that needed to be fixed. This reduced the time we reserved to complete the bonus.
- At the start of the project, we should refer to the sample project and set up a multi project structure. This will ensure that our code follows the multi project structure from the start, and reduce the time to adjust to the new structure later on. It not only fosters code consistency but also decreases debugging time.
- Another component that the team should focus on is gathering reflections from users regularly. For example, during our 4th demo, our TA identified a valuable enhancement for user experience, which is that our recurring events should have an end date. This insightful suggestion is logical from a user's point of view, but had not been initially considered by our team. Since it is the last demo, our team had to relocate our time to add this functionality.

- Through this team project, we also recognized the importance of communication between team members at the planning stage. More communication between teammates can help us understand everyone's strengths, and increase development efficiency. In our team, Sean and Jude had more experience with databases, and Amy and Sherlock had more experience with front-end UI. However, since Sean and Jude pair programmed together, they spent significantly more time in the first two sprints. Conversely, Amy and Sherlock had minimal experience with databases and web service, and sprints 3 and 4 became very challenging for them. If team members are familiar with each other's strengths, perhaps they can reorganize the pair programming so that there is always one person who is experienced with the sprint's focus.
- Finally, during our interaction with Git, we encountered some merge issues. So team members force pushed their feature branch to the main branch. Our team is not fully aware of its consequences until the end of sprint 4. We realize that force push will overwrite the commit history in the main branch, which could potentially introduce a problem if we wish to revert to one of the previous commits in the main branch.