Project Title: ClockWork

Team Members:

- Anthony Menendez amenendezmen2022@my.fit.edu
- Christian Ott cott2020@my.fit.edu
- Pierson Hendricks phendricks2023@my.fit.edu
- Peter Stelzer pstelzer2023@my.fit.edu

Faculty Advisor: Dr. David Luginbuhl dluginbuhl@fit.edu

Client: Dr. David Luginbuhl CSE Professor & Faculty Advisor

Progress of Current Milestone:

Task	Completion %	Anthony	Christian	Peter	Pierson	To do
1. Cohesion between and functionality of session, timer, and completion pages	70%	0%	0%	0%	100%	Await further developme nt
2. Implement Session Data Serialization and Persistence in Kotlin	100%	0%	0%	100%	0%	Refactor and integrate with app functions
3. Implement New Task Session UI in Kotlin	70%	0%	100%	0%	0%	Time estimate interface
4. Implement user progression evaluation in Kotlin	0%	0%	0%	0%	0%	
5. Implement task time estimation in Kotlin	30%	100%	0%	0%	0%	

Discussion of Current Milestone Tasks:

Task 1: Added some basic page-to-page navigation as well as handling proper states for a timer and task completion page. The app is currently lacking cohesion between the task list page and the pages for the tasks themselves, and the newly introduced new task page has not yet been integrated into the navigation network. These will be addressed next milestone.

Task 2: After a review of the workable methods of app data storage in android, Jetpack Compose's Room database seemed to be the best with its tight integration with Kotlin and the Android development ecosystem. The logical representation of task objects needed to be revised for serialization into the database as well as expanded to support more of the app's intended features. The initial attempt at this had to be scrapped because the extent of Room's native integration was overestimated, and the first task design could not be serialized elegantly. In its current form, the logic exists to fetch, save, and delete task data to persistent storage, but the app cannot use it dynamically.

Task 3: The first implementation of the calendar in the New Task Session UI was a DatePicker, but since there was no Dialog box associated with it, the DatePicker would not close after a date was chosen. The second implementation uses the DatePickerDialog that opens a DatePicker within a DialogBox. As for the user's initial time estimate, the data structure behind–an InfinitelyCircularList–has not been implemented into the UI, but the logic of the data structure itself is complete.

Task 4: No progress has been made on this task this milestone. It was not a high priority task because a lot of the complementary systems are still missing, so it would be implemented without context. Once the timer writes its progress to persistent storage using segment format, we can begin to develop the app interactions with historical data. It is at this point that the progress evaluation implementation would be most auspicious.

Task 5: The prototype script designed to implement the formula is now written in Kotlin within the project repository. It remains yet to be implemented to the application UI, but that is the next priority for this task. Following its implementation, modifications will be made to the logic to ultimately work to a more accurate and reliable estimation model.

Discussion of Contribution:

Anthony: I spent this milestone working on transferring the prototype function for estimating time over to Kotlin so that it can be implemented into the application. Some things I am considering to improve the estimation function are adding secondary functions that consider time completion categorized by difficulty. So, for example, when estimating the time for an assignment of 4/5 difficulty, the main function would only consider the previous times for that assignment that the user rated a 4/5.

Christian: I coded for task 3 alone, implementing the calendar functionality for the due date button, creating the task title button, and a difficulty slider to allow the user to select an appropriate difficulty level.

Peter: I was the sole contributor to task 2: I worked on creating the app's persistent storage for tasks and improving the logical representation of tasks to be more developed and compatible with serialization to the database. See the task 2 discussion for more details. Much of my effort was wasted due to a misinterpretation of the Room documentation, and probably poor design choices regardless. I initially believed that Room could implicitly serialize Kotlin types and attempted to make the segment design polymorphic. To get the subtypes to serialize, however, I would have to add a discriminator which defeated the point of that design. Fetching from the database also proved to be a major hurdle because Kotlin enforces that database tasks be executed on a peripheral thread. As such, getting the fetched data back to the main thread where it was needed was not straightforward and required the use of unfamiliar Kotlin and JetpackCompose components, namely ViewModels, Flows, and delegates.

Pierson: I simply put some time towards app navigation and passing states about the different page components, along with some general organization regarding the work of others.

Task Matrix for Next Milestone:

Task	Anthony	Christian	Peter	Pierson
1. Cohesion between and functionality of session, timer, and completion pages	0%	0%	0%	100%
2. Integrate data storage with task creation, editing, and timer execution	0%	0%	100%	0%
3. Finish New Task Page and Create Edit Task Properties Page	0%	100%	0%	0%
4. Timer execution segments integration	0%	0%	100%	0%
5. Integrate task time estimation into task creation and the corresponding UI	100%	0%	0%	0%

Discussion of Next Milestone Tasks:

Task 1: As stated in the discussion for the current milestone's task 1, the app is currently lacking cohesion between the task list page and the pages for the tasks themselves, and the newly introduced new task page has not yet been integrated into the navigation network. Connecting

the edit task page, which is planned for the next milestone, to the navigation network will be included in this task.

Task 2: Application data storage has been implemented, but it is not being used. Multiple functions in the app still need to be updated fetch from the storage (as opposed to baked values). Additionally, some new app features have been added or will be added next milestone which will create or change data, and that will need to interface with the data storage as well.

Task 3: With the addition of persistent tasks and the (soon to be realized) ability to create tasks during runtime, the next evolution is to apply changes to tasks during runtime through an edit task page. At the moment, tasks have two components: their properties (e.g. name, difficulty, due date, etc…) and their execution segments. The purpose of this task is to implement a UI form to alter the former. This will ideally be a near total reuse of the new task page developed this milestone, however since that component is not final yet it might not turn out to be viable.

Task 4: Task data objects now represent the execution of a task as a sequence of durations called segments. Segments can represent a period where the timer is running, where the timer was paused for a break, and where the timer was suspended. The timer logic itself, however, does not use this yet, and nowhere else is this feature used so segments are nominal only at the moment (the functionality is also missing on the segment end). This task for the next milestone is to address those two issues: extend the segments to support use by the timer and extend the timer to record its execution with segments.

Task 5: The formula logic behind the time estimation will be merged with a UI element so that it becomes visible to the user as they go to create a new task. This will be refined over time as more data is collected to make more accurate estimations depending on the type of task and its assigned difficulty.

Meeting Dates with Client During Current Milestone:

- Meeting 1: Apr 7, 2025
- Meeting 2: Apr 18, 2025

Client Feedback on Each Task of Current Milestone:

- see Faculty Advisor Feedback below

Meeting Dates with Faculty Advisor During Current Milestone:

- Meeting 1: Apr 7, 2025
- Meeting 2: Apr 18, 2025

Faculty Advisor Feedback on Each Task of Current Milestone:

- Task 1: No feedback, other than to suggest the term “integration” instead of “cohesion,” which often has a different connotation in software development. I look forward to seeing this integration in the next milestone
- Task 2: I apologize for not pulsing you a bit more on the problems with the Room database. I’m not concerned about the progress and the problems you encountered here, but I am a bit unclear on the way forward. Is this the new Task 2?
- Task 3: No feedback here. Calendar function looks fine.
- Task 4: No feedback
- Task 5: Current time estimation approach is fine; my guess is there will be a lot of tweaking on this as the app matures to the point of being able to be user tested. As we discussed, we may need to do a deeper dive (research?) into this whole area (I think we’ll need more than the paper I got from Dr. Chew, which doesn’t provide a whole lot of detailed information).

Faculty Advisor Signature: _____ Date: _____