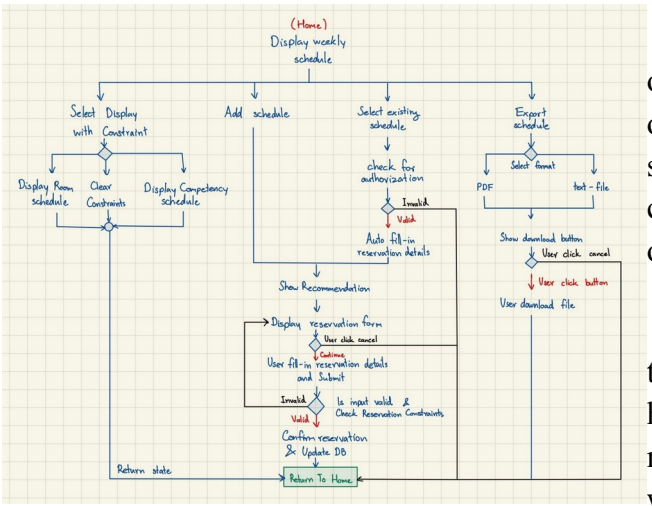
**SEN-201 – Software Engineering Process**
**00090 – Choose a software process paradigm**
**My Struggle with ScheDool's Development Process**

**Name: Thanawin Pattanaphol** **ID: 01324096**

ScheDool, a project consisting of four (hopefully) happy people, with their undoubted determination, to create a classroom reservation desktop application. This application, we said, aim to achieve two main goals: classroom management and reservation. Collectively, all team members decided on these two main central themes of our project. We had a goal of a smooth process of development. "This can't be that difficult, right?", said one of our team members. They would never expect what was about to come in the next couple of weeks.

We started our project on the basis of the aforementioned themes, essentially "Defining the Problem" that we would have to solve. We recognized that classroom reservation or scheduling can certainly be a huge hassle to be done manually on paper, along with the fact that this topic coincides with our URD-201 project theme of the "School Classroom Scheduling System", we thought that it would be best for us to work on a project topic that is similar to our URD-201 theme since we have done a certain amount of research on the chosen topic.

After defining our project definition, we continued on our development process by starting the design phase of our solution / implementation of this project for the following two weeks. We essentially started by drawing flowcharts of how our system will be like, as in, we laid out the plans of the inner workings of the program via various flowcharts: How users will navigate through the application? How does the program handle potential invalid inputs? Et cetera.




This process, however, did not come without its obstacles. We went through several iterations of our flowcharts, due to how we over-complicated some of the program's features or aspects which can lead to confusion among the users who use our program.

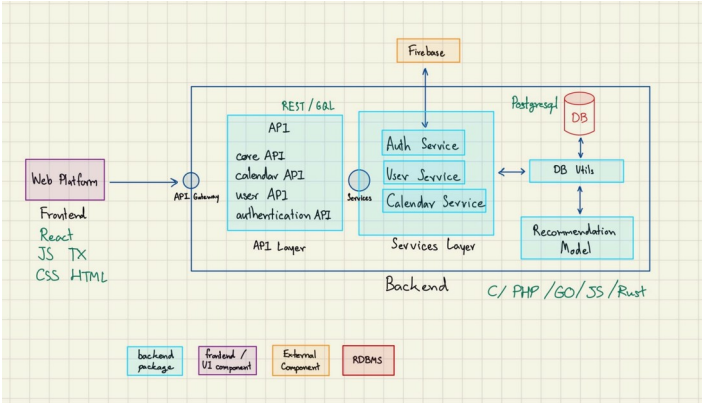
In the end, we went with the final flowchart on the left. The flowchart describes the flow of the how the program would function along the four main functionalities: selecting calendar display with constraints, scheduling a room, select existing reserved rooms, and exporting the schedule itself.

After designing our program's working flow, we then continued to design the program's architecture, essentially answering the question "How shall we implement this program if we were to actually do it?". This marks the beginning of a misunderstanding that most of the team members had while creating the architecture diagram. To understand this misunderstanding (ironic), we must remind ourselves of the requirement of this project. As labeled in the various slides provided by Professor Sally Goldin, the goal of this competency is to create a "**single-executable desktop**

**SEN-201 – Software Engineering Process**
**00090 – Choose a software process paradigm**
**My Struggle with ScheDool's Development Process**

**Name: Thanawin Pattanaphol** **ID: 01324096**

**application**" and not a "web-based application", however, the team submitted an architecture design based on a web application anyway.




As seen on the right hand side, we can observe that the architecture diagram clearly employs the elements of a web application. They are "Web Platform" (Frontend), REST API, an the Firebase connection.

This misunderstanding was carried throughout the week until the class on the 9th of September, 2025 A.D., in which, Professor Sally remarked "Why did you all send me architecture diagrams of web applications? You are supposed to send a single-executable application architecture diagram!". At that point in time, we realized, we have made a severe catastrophic mistake. For I, the writer of this very interesting report, realized that I have been right all along. My facial expression during the class was indistinguishably, completely, discombobulated as I, the writer, was the person who mentioned to the team during one of our meetings, "You know, we are supposed to write a single-executable desktop application, right guys?", then one of our members replied "Isn't a web-application also a single app?", I remarked "Well… I don't know.". In the end, we scrapped the web architectural diagram but still kept the overall theme of the concept.

After the architectural diagram fiasco, we continued on with our project. We continued on through our implementation of our solution. Our implementation process began on the the 22nd of September 2025 A.D. which embarked on a journey of "teaching Git to your fellow team members" activity. This activity was, to say the least, one of the most infuriating part of the journey. This is due to the unfortunate fact that some of our team members have a limited-to-no experience with the source control tool named "Git", developed by Linus Torvalds. This, however, is a totally understandable mishap; Git is one of the most complicated and confusing tools to use in managing source code versions i.e. source code merge conflicts occur when two different individuals update the same source file in two different directions. Therefore, I, the author, envisioned that this would be a tremendous opportunity to teach our team members how to use Git.

Consequently, I, the writer, along with my colleague, Mr. Poon, helped guide our other team members e.g. Mr. Beam and Ms. May to help understand the process of using Git. Eventually, we got our team members to be on the same page and catch-up to the intricate details of source control. After that, we (finally) started on the actual implementation process. We continued on by creating our initial draft in our GitHub Repository. In this rather "mundane" draft, we started with using the

**SEN-201 – Software Engineering Process**
**00090 – Choose a software process paradigm**
**My Struggle with ScheDool's Development Process**

**Name: Thanawin Pattanaphol** **ID: 01324096**

well known JavaScript front-end framework commonly known as "React.js", however, this decision will come back to haunt us in the final days of the project.

The main reason why we made a decision to choose React.js is our main framework is because of its popularity, easy-of-use and familiarity among the team members. With this decision, we specified the coding standards needed for this project, i.e., How our source code should look like, what it should contain, what format shall it be written in, and why is it written in that way. This however, began a long path of excruciating pain by our web developers to implement our program. It must also be noted that we included React Native for us to be able to compile our JavaScript / TypeScript based web application to a desktop application.

After a fortnight of our implementation process, we also started our validation process of our current program as labeled in Week 7 & 8 classes. The validation process employed the test cases of our program, that is, to test whether or not our program is able to handle unexpected inputs or prevent any errors from occurring. This process was mostly done by Mr. Beam, as he was put in charge of our team's documentation. Meanwhile, our web developer, Ms. May continued her rigorous journey of web development. The web development journey was filled with constant errors, warnings, difficulties and most importantly… miscommunication. There were several instances of misinterpretation of instructions and suggestions, Ms. May, however, with her uncharted determination, got through all of the hurdles with eloquence and ease.

After trenching through our implementation process, on the day before the final presentation, we finally started our packaging and deployment procedure. We have come to the realization (or reminder, of that fact) that this competency, as labeled by our most distinguished professor. Dr. Sally Goldin, would really much prefer that the executable file must be able to run on the Linux distribution, colloquially defined as "openSUSE". Nevertheless, we continued with our attempt at compiling our React Native code to a Linux executable file. But then, in an unexpected turns of event, it was apparent that React Native could not be compiled into a Linux build. The time was 21:00 GMT +7, the clock was ticking, the writer and Mr. Poon swiftly decided to discuss a potential plan B. After a couple of minutes of refining our plans and more obstacles of React Native compilation, we came to the conclusion of this nearly impossible task, to which to continue on this path within the designated deadline, that it is required for us to switch our entire codebase to Flutter, ultimately rendering the React Native code as redundant in the process, as our final solution.

During that scary night of nothing but coldness and stress, we did the impossible, implementing a codebase of an entire program in one. single. night. In the end, we successfully compiled the entirety of our newly written project into a single Linux executable as required in this competency. This allows the user to elegantly interact with our program with a single click of their fingers on their desktop mouse. In the end, we finished everything neatly and gave a splendid presentation in front of the class on the 27th of October 2025 A.D. thus, completing our long journey of SEN-201: Software Engineering Processes.

**SEN-201 – Software Engineering Process**
**00090 – Choose a software process paradigm**
**My Struggle with ScheDool's Development Process**

**Name: Thanawin Pattanaphol** **ID: 01324096**

Reflecting back on this expedition, we have reflected and recognized the potential improvements that can be made throughout this entire semester. Perhaps, it would be best, if starting this project from scratch, that we carefully considered the wide variety of options for a software framework along with making sure that we had understood the requirements for this competency to ensure a more smooth development process. Overall, we can deduce that we loosely followed the "waterfall" process paradigm as we had clear, laid out, steps of our project, however, the rapid and unexpected language switch, our process have morphed into closely resembling a more agile paradigm as we ended up iterating on the quality of our implementation of our improved selection of the language / framework.

In summary, our venturous process taught us several lessons that cannot be taught in a classroom, but to experience it firsthand. Therefore, we can conclude that, though the various obstacles we had faced, all the various erroneous instances of miscommunications, and a last-minute change, this project has overall been a huge success and a great learning experience for the four of us.