SAMS 2024 Project Log

Submitted by Group 2

Course: SWEN-746
Group Members:
Londhe, Soudagar, srl1622
Sajjala, Swetha, ss5743
Skalicky, Stacy, sms7705
Vellaisamy Senthilkumar, Rohini, rv8542

1. Week 0

Week 0 Project Log	
Description	Understanding Problem Statement and Discussing Team Collaboration
Date Created:	21 September, 2023
Overall Project Activity	• The team read through the project (SAMS 2024) description and requirement document to understand the problem statement. • We decided to use a combination of LucidChart and draw.io to create UML diagrams for the projects (such as use case, sequence diagrams, etc.) • For team collaboration, we decided to primarily use discord for team meetings and problem solving sessions. Additionally, we also planned to leverage Google Docs to collaborate on group assignments. • We made use of When2meet to help us synchronize schedules and find time slots to collaborate on group assignments. • Lastly, we created a GitHub organization to store all our created documents and diagrams. This will also be the VCS used for development of code for the project.
Model Analysis	N/A, as there were no models/assets created in this week
Model Evolution	N/A, as there were no models/assets created previously

2. Week 1

Week 1 Project Log	
Description	Creating SAMS 2024 Initial Use Case Model
Date Created:	28 September, 2023
Overall Project Activity	• In addition to the quick catch-ups after class, the team met thrice virtually over Discord to discuss our approach to developing the use case models for SAM2024. • Each team member selected an actor, for whom we created use cases (Rohini: PCM, Stacy: Admin, Swetha: Submitter/Author, Soudagar: PCC) • We created a first version of the use case models for the project containing only diagrams for identified actors. After realizing that a template (for Use Case Modeling) was provided by the professor, we

	refactored our assets to follow the template. This version of the document was submitted before the deadline. ● In the following days, during a discussion in class over the submitted assignments, the team realized we had not submitted Actor Modeling as a part of the assignment. The team had another quick ad hoc meeting to rectify this and created the actor modeling for the project and included it in the same document. This version of the document was submitted after the deadline. ● For all collaboration, the team used Google Docs (for document tracking) and Discord for communication. ● The final version of the document was committed to our GitHub repository.
Model Analysis	● **Why and how did we use this modeling technique? What were the results created?** We used Use case modeling for SAMS 2024 using the template provided by the professor. We identified the Actors present in SAMS 2024 and identified the system boundary. With the help of preconditions and postconditions for every Use case, it became easy to find the Normal and alternative flow. In the end interactions between all the actors were performed well. ● **What were the activities and tasks performed? How was it performed?** We in a team divided the actors as Administrator, PCC, PCM and Submitter. We identified the use cases for each of the actors and at the end we found the interactions between each of the actors and finalized the Use diagrams. Use case was created using Lucid chart and draw.io ● **How do we define and measure expected results?** Results were measured using the interactions of the actors and checked whether all the flow is covered including the alternative flow.
Model Evolution	There is no comment to be added or questions to be answered on the model evolution as this is the first model that is created by the team.

3. Week 2

Week 2 Project Log	
Description	Creating SAMS 2024 System Sequence Diagrams
Date Created:	3 October, 2023
Overall Project Activity	● Similar to the previous week, the team had small catch-ups in person after class, as well as, scheduled virtual meetings over Discord to discuss and collaborate on developing the system sequence diagrams for SAM2024. ● Each team member agreed to develop the sequence diagrams for the actors they had taken responsibility over last week (Rohini: PCM, Stacy: Admin, Swetha: Submitter/Author, Soudagar: PCC) ● Although the team assignment required us to only create two-three major system sequence diagrams, the team decided to create diagrams for all the use cases listed during Week 1 to account for all the aspects of the system.

	● After completion of sequence diagrams by individual members, the team reviewed and provided feedback for the diagrams to ensure consistency and correctness. ● For all collaboration, the team used Google Docs (for document tracking) and Discord for communication. ● The final version of the document was committed to our GitHub repository.
Model Analysis	● **Why and how did we use this modeling technique? What were the results created?** We utilized system modeling techniques to analyze the interactions between actors and the SAMS 2024 system.We made sequence diagrams to show how each person works with the system. This gave us a picture of how to remember all the important steps in the system and how people interact in the system. The results of our modeling efforts were a set of system sequence diagrams, one for each use case that we came up with for each role in the SAMS 2024 system. These captured the normal and alternative flows of each use case and clearly depicted the interactions between actors and the system. ● **What were the activities and tasks performed? How was it performed?** Using the existing Use case diagram created by an individual member in the team we created the sequence diagram including alternative flow. ● **How do we define and measure expected results?** A team review of the System Sequence Diagram was used to determine if our sequence diagram represented the use case each team member worked on. We questioned the naming conventions, ordering and normalized the format for the most part for all our diagrams to make it more readable and predictable between them.
Model Evolution	● **What techniques were used to transform/evolve models at different levels of abstraction?** Right now all our initial sequence diagrams are at the highest (least specific) level of abstraction. Although some might be used as operation, attribute and object names, they may not be the final identifier naming conventions. ● **How do we maintain traceability between a family of co-evolving models?** By keeping the documents easily accessible to each team member, making copies and revising from the original and creating a work history, either with commit messages or embedded in the document itself. ● **How do we ensure that the models developed consistently describe the same software system from multiple consistent views?** Keep the same attribute, object and operation names same between diagrams, if they change in one place, they should change everywhere else. For example, if we make a class diagram with operation functionA and its used in lower level/less abstract sequence diagrams then if that functionA becomes functionB, it would need to be changed in all associated diagrams/models. This would include if design had moved to implementation, and apply to source code, and

	vice versa.

4. Week 3 (does not include Fall break)

Week 3 Project Log	
Description	Creating SAMS 2024 Domain Model Diagram
Date Created:	19 October, 2023
Overall Project Activity	● Keeping up with our regular meet-up, the team had multiple catch-ups scheduled as virtual meetings over Discord to discuss and collaborate on developing the domain model diagram for the system. ● The group identified key domains of the document with the help of the actor-system sequence diagrams and created the domain model using LucidCharts collaboratively. ● We also briefly discussed the implementation part in terms of software and created a trello board to begin research/considerations for unit testing, and database (haven't started yet). ● The team agreed upon using Java and IntelliJ software development environments for the implementation. ● For all collaboration, the team used LucidCharts, Google Docs (for document tracking) and Discord for communication. ● The final version of the document was committed to our GitHub repository.
Model Analysis	● **Why and how did we use this modeling technique? What were the results created?** UML diagramming techniques, including encapsulation principles for attributes and operations, multiplicity, associations. A high level domain model with objects needed in the design of SAMS2024 ● **What were the activities and tasks performed? How was it performed?** Initial draft made by team, then adjustment of associations, adjustment of desired object and color additions, a legend to define line colors to improve clarity and separation, and a work log with details on team changes. Utilized lucidchart, discord for collaboration and google docs to create and then export as pdf. ● **How do we define and measure expected results?** A finished chart that is iterable and accessible to the team. A deliverable provided to stakeholders.
Model Evolution	● **What techniques were used to transform/evolve models at different levels of abstraction?** From the Use case and sequence diagram made previously we identified the Objects and Systems for the domain model. From the objects we identified the relationships between the Objects and identified the attributes and methods for each object present. ● **How do we maintain traceability between a family of co-evolving models?** By keeping the documents easily accessible to each team member, making copies and revising from the original and creating a work history, either with commit messages or embedded in the document

	itself. ● **How do we ensure that the models developed consistently describe the same software system from multiple consistent views?** By keeping the same methods and attributes for each object the same as the previous diagram so that it remains consistent in all the flows.

5. Week 4

Week 4 Project Log	
Description	Creating SAMS 2024 Detailed Requirements Model using Use Cases and SAMS 2024 Architectural Model
Date Created:	26 October, 2023
Overall Project Activity	● The team requested the professor to review the initial use case model submitted and used the feedback to improve the first draft of use cases from the comments given by the stakeholder to develop the new version of the detailed requirements model using use cases. ● The team scheduled virtual meetings over Discord to go over the updated version of the document. The changes in the document includes changing the use case diagram arrows, bubbles and merging alternate flows that had been broken down into separate use cases. ● The team also scheduled separate calls to develop the system's architecture model by identifying the different views and creating a package diagram. We also included the allocation required for the system. ● For all collaboration, the team used LucidCharts (for diagrams), Google Docs (for document tracking) and Discord for communication. ● The final version of the document was committed to our GitHub repository. ● Next steps are to decide how to divide work and confirm the development environment. ● Used for Conference Paper Template: https://avidnote.com/conference-paper-format/#:~:text=A%20typical%20conference%20paper%20should,literature%20review%2C%20and%20funding%20footnote. ● Used for Review Template: https://globalconference.ca/how-to-review-a-conference-paper/#:~:text=Reviewers%20typically%20evaluate%20the%20conference,the%20significance%20of%20the%20findings.
Model Analysis	● **Why and how did we use this modeling technique? What were the results created?** We analyzed the different views and provided allocation, logical and process view details in our Architecture model. We determined our use case view was covered in the use case model/detailed requirements deliverable. For allocation view discussed the system requirements our PC's could support and chose to utilize them as baseline for requirements since that is where we are developing. For a logical view we created a package diagram that focused on high level

	the main objects within our class diagram that other objects depend on. For the process view we each created process flow diagrams for the primary use cases and included their alternative flows. ● **What were the activities and tasks performed? How was it performed?** We collaborated together to create the architectural model using Lucid Charts and collaborating over discord. ● **How do we define and measure expected results?** If the interaction between the actors happens in a correct flow it defines a good outcome. Also all the alternative flows were taken into consideration.
Model Evolution	● **What techniques were used to transform/evolve models at different levels of abstraction?** We used team discussions to limit and refine. We referenced material from some online material(for constraints for templates and credentials) and class to determine which views were valuable. ● **How do we maintain traceability between a family of co-evolving models?** By referencing the previously modeled use case diagrams and domain model diagrams. ● **How do we ensure that the models developed consistently describe the same software system from multiple consistent views?** Being careful not to make new identifier/attribute/component names, evaluating the ones we do have. Keeping consistent naming practices.

6. Week 5

Week 5 Project Log	
Description	Creating SAMS 2024 Detailed Design Model (Iteration 1)
Date Created:	6 November, 2023
Overall Project Activity	● The team had multiple catch-ups scheduled as virtual meetings over Discord to discuss and collaborate on developing the first iteration of the detailed design model. ● The group identified the main deliverables of the document (Detailed Class Diagram and Behavioral Diagrams like State/Process and Object-Sequence diagrams) and developed them accordingly. ● Each team member helped in providing feedback to individual Object-Sequence diagrams to ensure correctness and uniformity. ● For all collaboration, the team used LucidCharts, Google Docs (for document tracking) and Discord for communication. ● The final version of the document was committed to our GitHub repository. ● The team also delved into discussing an overview of the implementation details and also setting up mySQL and its user interface tool locally on our laptops.
Model Analysis	● **Why and how did we use this modeling technique? What were the results created?** Object-Sequence diagrams were developed by the team that revised

	the original diagrams (Actor/System- Sequence diagrams) submitted and now included its actual attributes, function calls, results and respective object instances on which the operations are invoked. ● **What were the activities and tasks performed? How was it performed?** The team ensured smooth collaboration by connecting over discord several times within the week. The task of revising the class diagram was done together whereas the tasks for the revised sequence diagrams were separated on the basis of actors (as done previously). The group as a team then refactored the deliverables based on internal feedback.. ● **How do we define and measure expected results?** The deliverables/results are considered to be as expected when there is good clarity and understanding on the implementation pieces of the application that can be directly inferred from the diagrams/models.
Model Evolution	● **What techniques were used to transform/evolve models at different levels of abstraction?** The process of refining and adding layers of details to the identified high-level designs without affecting the solution to be achieved ● **How do we maintain traceability between a family of co-evolving models?** By ensuring consistency in the relationships between entities are not changed (unless the need is identified). ● **How do we ensure that the models developed consistently describe the same software system from multiple consistent views?** By cross-verifying that the different attributes, relationships and operations defined at each layer detail produce similar patterns/outcomes.

7. Week 6

Week 6 Project Log	
Description	Planning Basic Implementation of SAMS 2024 Application
Date Created:	14 November, 2023
Overall Project Activity	● The team discussed the division of tasks (weekly basis) to tackle the implementation of the SAMS 2024 application. ● According to the plan drafted, the first week will involve setting up the DB tables (Stacy & Swetha) and designing the wireframes/mockups (Rohini & Soudagar). The week afterwards the team will be hands-on with the BE implementation of classes. The week after integration between classes and finally during the last week: its integration with FE (stretch goal). ● The team also identified that a fully-fledged frontend is a stretch goal and as a part of the MVP, the team will have a Invison frontend mock-up to showcase the potential of the FE. ● All collaboration of code is done over GitHub using separate repositories for frontend and backend. The tasks are tracked over Trello.

Model Analysis	**N/A**
Model Evolution	**N/A**

8. Week 7 & 8

Week 7 & 8 Project Log	
Description	Basic Implementation of SAMS 2024 Application
Date Created:	2 December, 2023
Overall Project Activity	● The team started implementing various pieces of the application, focussing mainly on the backend and database code. The team engaged in multiple debugging and pair programming sessions to collaboratively implement the application. ● The team pivoted towards using JavaFX to implement the UI/UX of the application due to its ease of not requiring a backend server. A basic happy path flow was identified for the role of Submitter and executed with simple components. ● For testing, basic JUnit tests were included. ● Once the application pieces were developed, the team worked on the final detailed design document, iterating over the previously created one and enhancing it based on our learnings from implementation. ● For all collaboration the team used Discord for communication, Google docs for documents, and GitHub for code version control.
Model Analysis	● **What were the activities and tasks performed? How was it performed?** The team prioritized developing a Minimum Viable Product (MVP) by focusing on key tasks such as identifying the app flow, backend development, JavaFX implementation, and thorough testing. Each member undertook specific responsibilities within these domains, ensuring a collaborative and efficient development process. The goal was to create a functional MVP suitable for effective in-class demonstrations. ● **How do we define and measure expected results?** Expected results are gauged by two primary criteria. Firstly, success is determined by user interaction with the MVP along the basic happy path, assessed through feedback and usability testing. Secondly, the alignment between code model-generated diagrams and the team's designs is measured for accuracy. These criteria succinctly capture the project's success in user experience and code model fidelity.
Model Evolution	● **How do we maintain traceability between a family of co-evolving models?** Maintaining traceability within a family of co-evolving models was achieved by designating the design model as the master reference. This model served as the guiding blueprint for implementing attributes and methods consistently across the evolving models. While certain features were excluded to prioritize the MVP, the fundamental skeleton of the design model was preserved, ensuring alignment between the conceptual framework and the implemented code. This approach ensured a cohesive and traceable evolution of the models.

	● **How do we ensure that the models developed consistently describe the same software system from multiple consistent views?** Ensuring consistent depiction of the software system across multiple views was achieved through systematic side-by-side comparisons. The team rigorously assessed auto-generated model diagrams with those crafted through MDD, validating and aligning them to guarantee uniformity in representing the software system from diverse perspectives.

9. Week 9

Week 9 Project Log	
Description	SAMS 2024 MVP, Presentation & Feedback
Date Created:	7 December, 2023
Overall Project Activity	● In addition to the Submitter flow, the team also implemented a basic Administrator flow to manage accounts with its respective user interface. ● Post-completion of the application, the team had two virtual catch-ups to polish the final deliverables (detailed design document and implementation) and also created the slides for the group presentation. ● The team had a mini dry-run of the presentation within ourselves, by segregating responsibilities of different parts of the slides. ● The feedback provided by the class and the professor post-presentation were noted down by the team ● For all collaboration the team used Discord for communication, Google workspaces for documents and slides, and GitHub for code version control.
Model Analysis	● **Why and how did we use this modeling technique? What were the results created?** Based on the administration flow sequence diagram, a domain diagram was prepared, aiding in the identification of objects within the system. Implementation was carried out accordingly, and ultimately, the administrator succeeded in creating and managing user accounts. ● **What were the activities and tasks performed? How was it performed?** The team connected on Discord and divided tasks based on frontend design for administration, which includes creating an account, deleting an account, and updating an account. In the backend, we created models and established relationships among them, facilitating interaction with frontend implementation. ● **How do we define and measure expected results?** We conducted test cases for the administrator part, which involves validating all the required fields when creating a new account for a specific actor or user. The details updated by the administration should be reflected in users' accounts, ensuring accurate results.
Model Evolution	● **What techniques were used to transform/evolve models at different levels of abstraction?**

	We initially focused on database design to lay the foundation for implementing classes. This early stage provided a basic understanding and structure for our system. As we progressed into the actual implementation phase, we encountered conflicts between the classes and the initially designed database structure. To resolve these issues and achieve a cohesive system, we employed a technique of refining and rearranging the database tables and the associated classes. This process involved critically evaluating and adjusting the database schema and class design to ensure they were aligned and functioned effectively together. This iterative approach of evaluation and modification was crucial in evolving our models from a basic abstract level to a more detailed and practical implementation level. ● **How do we maintain traceability between a family of co-evolving models? // How do we ensure that the models developed consistently describe the same software system from multiple consistent views?** We continued to prioritize our design model as the master reference, ensuring consistent alignment across all evolving models. We introduced a version control system, which played a pivotal role in tracking changes and maintaining the integrity of the models as they evolved. Each alteration made in the models was meticulously documented and linked back to the master design model. We conducted regular team reviews, collaboratively examining the changes, ensuring that every modification served the overall project objectives. This comprehensive approach, combining a robust version control system with collaborative reviews, facilitated a traceable and coherent progression of our models, even as they underwent complex transformations. Even still, lucid chart diagrams of behavior still quickly ran out of date. We did our best to retranslate. Pursuing other ways to model final implementation would be good.

Overall Reflections

- **What did we learn?**
  Building an application using model-driven development (MDD) techniques, such as Actor Modeling, Use Case Modeling, Domain Diagrams, and Architecture Modeling, imparts valuable insights: through actor modeling and use case modeling, a comprehensive understanding of system requirements is achieved, while domain diagrams provide a visual representation of key concepts and architecture modeling aids in structuring the system. MDD encourages a unified language, promoting effective communication and collaboration. The use of abstraction and code generation enhances reusability and adaptability, allowing for efficient change management. This approach improves overall development efficiency and facilitates seamless adaptation to evolving requirements.

- **What went well?**

    - *Communication:* Regular catch-ups with the team virtually allowed to facilitate easy problem solving and timely submissions of deliverables.
    - *Design De-obfuscation and Translation to Implementation*: Clarify and translate design elements to facilitate seamless implementation within the team.
    - *Timeline Decisions for Scope Limitation*: Strategic decisions on the timeline to restrict and manage project scope.
    - *Database Implementation*: Execute the database plan outlined.
    - *Quick Spike of JavaFX*: Swift adaptation to leverage JavaFX framework.

- **What could've been done well?**
    - *Express course learnings:* More focus on course learnings towards the application implementation during final team presentation.
    - *MySQL and Blob Enhancements for PDF Upload/Download:* Enhancements focused on improving PDF upload and download functionality using MySQL and Blob.
    - *Elevated Emphasis on Unit Testing:* Augmented efforts in unit testing for enhanced code quality and reliability.
    - *Code Comments Enhancement:* Increase the use of comments for better code understanding.
    - *Optimized Task Tracking:* Utilize Trello/Jira more effectively for improved task management.

- **Closing thoughts**
    - With the advent of new technologies, it would be good to consider looking into design to implement code generators. This would allow direct transformation of a detailed design model into something written.
    - As a team, we meshed well, considered each other time and worked on keeping each other up to date with meeting summaries and asks and to-dos.
    - We had unspoken roles, database lead, design document and testing lead, and as well as java lead and team lead. We mostly kept within a particular role on the team and tried to meet the need. Additionally, a lot of peer programming and in-team demonstrations were done to help the team understand. Every person on the team was able to execute the code, both front end and back end and participated in improving the final deliverable.
    - As for design, we did so much up front that once it came to implementation our java developers had little questions or concerns, just some adjustments to the database entities to meet some needs, however, because of the complexity it was initially difficult to follow the final implementation. This could be improved by streamlining and compacting some of our choices for class associations as well as better documentation.