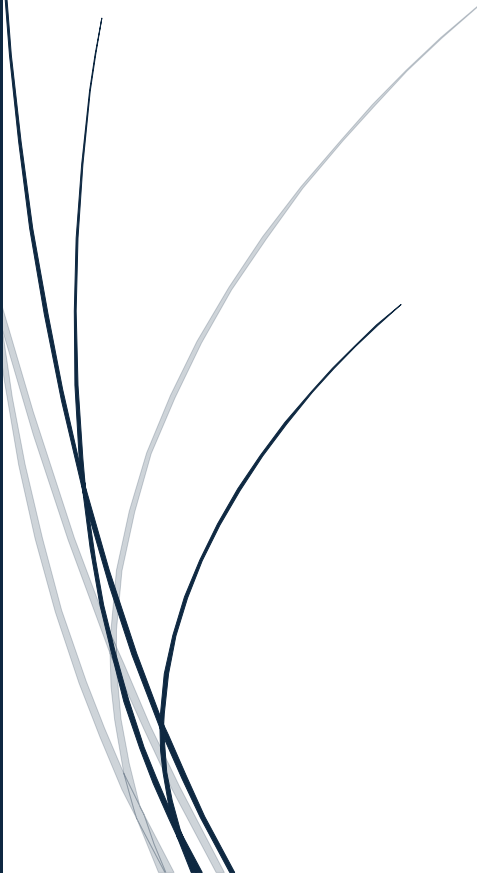


4/25/2025

Scrum Gang

Scrum Insurance Claim
Management System



Scrum Master: Josh Mangham
Brady Werling, Jack Sibbitt, Keegan Jackson
SE 361: INTRODUCTION TO SOFTWARE ENGINEERING

Table of Contents

Chapter 1: Project and Team Introduction ----- 2

- Brief Overview
- Team Structure
- Agile Methodology

Chapter 2: Sprints ----- 4

- Sprint planning
- Product and Spring Backlogs
- Daily Standups
- Project Timeline

Chapter 3: Individual Reflections ----- 9

- Brady
- Jack
- Josh
- Keegan

Chapter 4: Conclusions ----- 15

- Addressing Goals
- Scrum Process Thoughts
- Project Evaluation
- Future Improvements

Chapter 1: Introduction

Brief Overview

The primary goal of SE 361 was to practice and implement the agile methodology. A semester-long project was chosen to implement the agile methodology, Scrum. Our team, Scrum gang, set out to create the application for our chosen project, an insurance claim management system. The insurance claim management system (ICMS) as outlined by the product owner, is an end-to-end system for clients, administrators, claim managers, and finance managers for an insurance company. The ICMS is designed to accommodate the diverse functional requirements of its four user groups: clients, administrators, claim managers, and finance managers.

Clients should be able to login into the ICMS, view and edit their profile, apply for claims, upload documents, view the messages sent by the finance- and claim managers, view the status of their current claim(s), and download the claim reports. Clients must be able to navigate all these requirements through the ICMS graphical user interface (GUI). Administrators should be able to login into the ICMS, view all user details and view said details by date, assign roles to other users, and create folders for clients to store their information. Finance managers should be able to login into the ICMS, view and edit their profile, view the profile of their assigned client, communicate with other ICMS users, calculate client estimates, and validate and transfer claim amount details. Lastly, claim managers should be able to login into the ICMS, view and edit their profile, view the profile of their assigned client, communicate with other ICMS users, download client-uploaded documents, and transfer files to the finance manager.

Team Structure

Our Scrum team, Scrum gang, self-organized based on individual interests and backgrounds, worked collaboratively to address all functional requirements outlined above. Josh Mangham was the Scrum Master and worked primarily on the front end (GUI) of this project. Brady Werling worked on the front end and back end (database) of this project. Jack Sibbitt focused primarily on front-end development, while also contributing to back-end tasks for the project. Keegan Jackson worked predominantly on the back end and as the information coordinator while contributing to some front-end tasks for the project. However, as a note, our team was incredibly cross functional. All members worked on multiple facets of this project.

Agile Methodology

Our team followed the standard framework outlined by the agile methodology, Scrum, to approach this project. Goals and user stories were created and prioritized for every two-week Sprint. The agile methodology will be described in greater detail below.

Chapter 2: Sprints

Sprint Planning

Sprints are the iterative framework of the agile methodology. Our Sprints consisted of several key parts: Sprint planning, the Sprint itself, daily standups, the Sprint presentation, and the Sprint Retrospective. Additionally, the Product Backlog – a list of every functional requirement for the project – and the Sprint Backlog – a temporally prioritized subset of the Product Backlog – were the driving frameworks for the agile methodology.

Prior to each Sprint period, planning occurred to outline and prioritize the goals for that Sprint. Sprint planning was done following a Sprint Retrospective. After a Sprint period, a Sprint presentation occurs which updates the product owner and our peers on our deliverable artifacts and overall progress, we received immediate feedback from our peers in the form of Google Forms. With the feedback, we began our Sprint retrospectives. During these Sprint Retrospectives, as a team we discussed what worked well, what did not work well, what changes we would like, and evaluated our individual and team performance. These retrospectives serve to correct any budding issues before they cause significant issues later down in the development process. Following these retrospectives, we would also plan for our goal for upcoming Sprint period.

Product and Spring Backlogs

Goals were outlined based on the feedback we received from a retrospective, the product backlog, and what we decided as a team was most important. Deciding these goals was a collaborative process. Our Scrum master allowed all members to pick what

they wanted to work on for that Sprint. This led to the high level of cross-functionality that characterizes this team. Members worked together, suggesting ideas for code and designs to complete items from Sprint backlog. These chosen goals and user stories were taken from the Product Backlog and placed into the Sprint Backlog, populated to a Trello board. Furthermore, the status of these goals was updated throughout the Sprint to openly display how things were progressing with the other team members. The Product Backlog was based off the functional requirements created by the Product Owner. We displayed the list of items to be completed on a PowerPoint, separated by user type. At the end of each Sprint, the items on the Sprint Backlog (and thus Product Backlog) were completed and marked as such.

Daily Standups

Moreover, during the two-week Sprint period, daily standup meetings were conducted every class meeting. These meetings were brief and consisted of three questions. Team members discussed what they did yesterday to help with the project, what they will do today/next, and what obstacles they are encountering. These questions ensured that our communication saturation was relatively high. Everyone knew what everyone was doing.

Project Timeline

In our first Sprint, Sprint #0, we achieved several basic goals. First, we decided on and chose our semester-long project – ICMS. We introduced ourselves to each other, discussed our strengths and weaknesses and what our interests were for this project. With that information, we assigned teams roles. Additionally, we determined our team's name,

chose Trello as our documentation platform, agreed upon a software and database setup, and established communication protocols outside of class through a group chat. To kickstart the project, we created a prototype login form with limited functionality and outlined a couple broad user stories.

Building on the foundation of this initial Sprint, we gradually started addressing the project requirements. In Sprint #1, we set out to add functionality to the login page, learn how to switch to other pages, brainstorm our GUI design, and configure the database. In this Sprint, we created a functional login page with linked create account and forgot password pages. All forms displayed some level functionality, however some of this functionality was hardcoded into our program. We also established our database connection using DBeaver.

In Sprint #2, we planned to develop our team's branding, store data in the database, and separate pages based on user type. In this Sprint, we successfully updated our GUI, were able to add accounts to the database, created insert and select queries, created separate admin and user landing pages, created the session class, created the database and subsequent tables, and fully outlined all the user stories for the ICMS roles.

For Sprint #3 we aimed to accomplish the following: update GUI further, fix the select query, add a claim page, add a message page, add an admin report page, add manager (Finance and Claim) pages, and refine the database. During this sprint, we were able to update the GUI, update the client landing page, create both manager landing pages, create the message page, add edit/view profile functionality, create the view claim page,

create a dashboard flow of controls, and update the database with the remaining tables needed for the project.

By this stage of the semester, our velocity had significantly improved compared to Sprints #0 and #1. Consequently, during Sprint #4, we effectively tackled and completed a considerable portion of the remaining functional requirements from our Product Backlog. In this sprint on the front end, we added functionality to the apply claim page, added upload document functionality, added messaging functionality, updated password requirements for accounts, updated account creation ctrl, updated forgot password ctrl, added view claim and claim list functionality, add claim status functionality, and continued to update our GUI with our developing branding.

Regarding the back end, we added database security measures using a config.json file, had a slight database reorganization with the creation of a user table. Organized our files in Visual Studio, added an abstract query class, updated the insert and select queries, and updated objects with the ability to function as constructors. Row objects were instrumental to the previous functions as an intermediate for database data and our custom database objects.

In the final stretch of Sprint #5, our team maintained peak velocity in the implementation of final functional requirements. This sprint brought many refinements to the communication system: message composition, manager claim transfers, and landing page recent message and claim info, all pieces of the backbone in claim applications.

Additionally, admins could now modify user login info and role through selection in their landing page's data grid.

Another focus this sprint was on rebuilding front-end experience with dynamic form resizing and panels. The message viewer, claim viewer, and document upload in the claim application page now all dynamically create panels for displaying each's contents. Responsiveness through display of database errors and successful actions also contributed to end user improvements.

Back-end work this sprint introduced the use of the Last Inserted ID function for follow-up inserts in connected tables, nullable return types for communication of database errors, and the return of custom object lists for dynamic data. While not visible by the user, these functions empower our application's response time and optimize the number of calls to the database.

Chapter 3: Individual Reflections

Brady

My goal for this project was mainly to work on the middleware layer, but I bounced around everywhere when needed. My first main contributions were designing and many times reworking the Session object, Scrum User Control extension class, and the dashboard control. Our form uses these three for smooth transitions in between controls in the main form, which is entirely covered by our main panel, a table layout panel. The Scrum User Control class contains many methods for loading, deleting, and swapping controls as well as the flow of data between controls. Keegan and I built the code to upload documents, but I designed the code to download documents. I implemented the usage of the database config.json file for secure storage of credentials. I enforced the usage of nullable Boolean and empty object returns for tracking the success state of database queries for conveying info to front-end.

The login, create account, and forgot password controls are largely of my rewriting and styling. I designed background art for the login control in Photoshop and took the scrum logo, password, and trash icons from online. I brought the use of error providers for validation to multiple controls. I reworked the styling on controls: admin landing, claims viewer, edit profile, and claim apply. I implemented flow layout panels for loading in multiple controls in the inbox, claim application, and claim viewer controls along with Jack and Josh.

I created the query classes and execution methods we use to facilitate almost all database queries, including the get last inserted id for follow-up inserts. I refactored the Data Set functions for getting and updating bulk data for the admin landing control. I did a complete rewrite of the database when our program needed a different structure and enforced strict naming conventions on it. I first implemented the use of Row objects as constructors for our other database objects like Message, Row, and Claim, storing each column as a property with the proper data type.

I was the demo guy, recording and reviewing the demonstrations during our sprint presentations. I initiated group meetings over text and offered other group members help whenever necessary.

Jack

My initial role for this project was to work mostly in the front-end, however I also did a little bit of work with the back end. Towards the second half of the project, many of the front-end pieces required functionality from our database, which led to me working more with the middle-ware functions so that the back end could be connected to the front-end. My earliest contributions to the project were creating necessary forms and designing GUI standards for our forms. I created our original welcome form, as well as the early create account and forgot password forms. I was the one who created our GUI standards and focused on trying to keep pages consistent with our styling. During sprint #2, I worked with our session class to ensure that a user's role would be utilized in the session in order to give access to the correct pages for that user. I also made an early version of our dashboard

as a concept that later would be used by Brady to be fully implemented as our main user control.

For our fourth sprint (sprint #3), I created our claim viewer page, our profile edit page, the list pages for messages and claims, also assisting with the landing pages for the client, finance manager, and claim manager roles alongside Josh. I also assisted with the dashboard and assured that it would intake the correct information needed when loading different controls so that any information needed when loading the form could be used. This included making sure that when the home button was pressed, it would take the user to the user control for their user type (client home page, finance manager home page, etc.).

I continued working with many of these forms in sprint #4 and began connecting them to the database and backend. I created the panel object to load in information for each claim / message and use that information to load either a new control with the claim information or load in the selected message. To do this, I had to create new functions in the middleware using our database controller in order to write queries that would obtain all of the information I needed and pass it back to the front end. I later ended up working with Brady to further refine these methods once the implementation for our Query C# class was finished.

Lastly, during sprint #5, I continued work with my framework for the claim list and message list pages in order to create a compose message page. This used the same idea of utilizing a panel to load information on users that one can select to be a recipient of a message. I also had to use an insert function in order to insert a message into our database

so that it could be accessed by the message viewer page. I also worked on the functionality of allowing claims to be transferred between the required roles by utilizing user info in our session class to set the status type of each claim and assign it to the appropriate manager in our claim viewer.

Josh

Throughout the project, I served as the scrum master, making sure the team sprint review surveys were completed, and worked on various aspects of the front end and the back end. For sprint zero, I wrote a few basic user stories to give a user role a unique purpose. For sprint one, I added the account class and static session class, so users could create an account, login and reset their password, with the information being stored in the Session instead of being passed around. For sprint two, I decided to change the color scheme to blue and light blue boxes since a major concern from the previous sprint was the hideous grey and white GUI and designed the forgot password panel. I also modified our select query to actually return the account information, so the password could be reset using a security question stored in the database. For the third sprint I designed a completely new landing page and migrated other pages to use its top bar, sidebar and bottom border. Either on this sprint or the next one, I also added the update and delete queries so users could edit and delete their profile.

For sprint four I started experimenting with background images for pages where the user isn't logged in. I also made the claim creation page function by having it put claims into the database and being attached to the account that submitted them. I also designed

a rough claim status system, so claims submitted by clients could be approved and evaluated by the claim and finance manager respectively on the existing claim viewer page. To do this I also had to modify the database so all table IDs related to and including the claim table would be integers instead of strings to match the user table. I also created a row class that would store a dictionary of columns, so the select query could return multiple of these rows for If a user had multiple claims. For the final sprint, I finally gave functionality to the admin role, by allowing the admin to use a data grid view to view and modify the database's user table using a data set that is passed back and forth and reads and writes to the database using a MySQL adapter.

Keegan

The majority of my contributions focused on backend database design and development, as well as creating user stories, presentations, and co-authoring this document alongside Brady. During Sprint #0, I familiarize myself with Visual Studio, Git, and C#, taking additional time outside of class to become more skilled with these items. During Sprints #1 & #2, I designed, created, and began implementing the database. To do this, I wrote out all the user-stories in more tangible language and created their subsequent ERDs to better visualize the database and the relationship between the roles and functions, which was shared on the Trello board. After several iterations and refinements, I finalized the database design based on the user stories. Using DBeaver, I created all the tables, primary keys, and foreign keys for our shared Scrum Gang database. Brady then helped to refine and reorganize some key aspects of the database, using his knowledge of the queries he made to guide the reorganization.

I did contribute to some front end tasks as well. I did help to create and design the edit user profile ctrl. However, much of the personal details that I added ended getting scrapped due to time limits. Further, I implemented the upload documents feature and laid the groundwork for the documentation upload/download functionality. Again, Brady helped me and managed to figure out how to use the Long Blob data type to store documents in the database. I also refined the create account and forgot password pages by introducing stricter password requirements, mandating the inclusion of a symbol, a number, an uppercase letter, and a lowercase letter. Brady supported this effort by refactoring the code. I throughout all the Sprints would add comments for code implementation. I played a significant role in non-technical tasks, including creating presentations, drafting papers, and handling documentation. I also maintained consistent communication through the Trello board and by leaving detailed comments.

Chapter 4: Conclusions

Addressing Goals

Addressing the functional requirements, we implemented most features listed for admin, client, claim manager, and finance manager roles and reworked some. We scrapped the usage of emails and storing personal information like names and phone numbers, instead opting for usernames. Admins do not create folders for clients, the clients upload documents along with their claim application. Clients do not download their claim reports; they view them in a claim viewer control. We added that all roles can message each other, even sending messages to multiple people. That is how the financial manager communicates the claim amount to the client.

Scrum Process Thoughts

Brady

I thought the Scrum process, when used properly, facilitated communication of both current and completed work. This I believe to be the most important part of the Scrum process, no one can build off another's work or make use of it without knowing about it or how it functions. Daily standups especially helped fill this gap in understanding of other's work. Our group faced some communication issues at the start, but the Scrum meetings helped us learn each other's strengths, becoming stronger as a team. Tasks on a Trello board do feel great to move over and check off; they really keep you focused on the full implementation of one feature before moving on to the next. Sprint retrospectives were the only hitch in the road for me, it felt like we repeated much that was already discussed in other meetings and a waste of time.

Jack

I think that the Scrum process made the development process much cleaner overall by getting our group to be very organized and efficient with getting tasks done. While we did have some initial issues with communication, as we got further into the semester, we really started utilizing things like the backlog and the meetings to figure out what the highest priority items were and who the best fit for each would be. This ended up making our velocity increase dramatically and it was evident in our deliverables at the end of each sprint. I think that Scrum overall did a great job at keeping us focused on one task at a time, which ended up making it much easier to develop. We were usually able to get at least a base level implementation of a deliverable by the end of the sprint, and this allowed us to take a step back and decide if we were going in the right direction and what would be best to develop next. I absolutely can see myself using parts of the Scrum methodology outside of just my career, as in any sort of group process it does a great job of checking off tasks and ensuring that a team is interconnected through daily stand-ups and utilizing a backlog for priority on tasks.

Josh

Using scrum provided a great structure for completing this application. Reading through the book, it is no surprise it revolutionized software development. Having a constantly depleting list of everything that needs to be done is a great display of progress and velocity. I've never worked in a team creating something like this before and in the beginning, I was concerned about merge conflicts, but by segregating tasks, it was very rare for someone to edit the same file as someone else at the same time. I can't imagine spending weeks planning this entire project using the waterfall method, since many functional requirements I completed, I had no

idea how to do, or how long it would take to learn how to accomplish and then complete. Planning like that would put everything behind schedule and demoralize us as developers seeing project completion become more and more unlikely, not to mention all the time wanted planning. Overall scrums solution to this made the development process much easier, providing a great structure for who would complete tasks and the planning every two weeks made it so we would never feel we were running behind schedule, only increasing velocity as we became more accustomed to using the new tools provided to us, but also working as a team. I will be happy to continue to work according to the scrum process in my future career.

Keegan

Overall, I really enjoyed the Scrum process. I thoroughly enjoyed reading the Sutherland text alongside my own integration of the agile methodology. I have seen firsthand how Scrum lends itself to ever-increasing productivity. I believe that the daily stand-ups and the idea of communication saturation are absolutely imperative to work. As Sutherland highlights, without a consistent flow of information and communication, productivity suffers. Teams are unable to be efficient nor effective due to gaps in knowledge. Furthermore, I thought that the idea of failing early but always delivering a working piece of *something* to be important. These ideas combined put a comfortable amount of pressure on a group. It's not the end of the world if you fail, but it keeps a team focused to have at least something. I believe that the agile methodology can and should be implemented elsewhere. Coding and software engineering may not be my life work, but Scrum is something that I will use in the future. Even in research, which can be comparatively stagnant, Scrum could absolutely improve output. I've already started to do

so in my research group through the creation of a documentation guide. It's a small step, but as I've gleaned throughout this course, communication is king.

Project Evaluation

The overall evaluation of the project is positive. The team gained significant knowledge and experience, especially considering many members had little to no prior exposure to Git, C#, Visual Studio (VS), or the agile methodology. The project provided valuable insight into a semi-realistic software engineering working environment, fostering improvement in both individual coding skills and collaborative teamwork abilities. Many of us have never worked on full stack development at this scale before and consequently learned a lot throughout this semester. This project has allowed us to develop several invaluable skills relating to the software engineering process as a whole. Importantly, it allowed us to connect all the individual silos of computer science and software engineering knowledge and skills into one meaningful experience.

A substantial portion of user stories were successfully accomplished, resulting in a functional product. We are proud of the final product we created. Even though the generate and download reports functionality is missing, we believe our project fulfills the most important parts of the product description. We believe that this project is critical to developing computer science and software engineering students.

Future Improvements

There isn't much we would do differently. The project went well, and we worked together effectively. However, we do wish we had established a more robust

communication system from the outset. We had a group chat, but it was used infrequently. While we didn't know each other prior to this course, we could have made a better effort to build team rapport in the beginning.

Spending more time together while working on code, either individually or collaboratively, would have been beneficial. Working amongst or with each other would us ask each other questions, bounce ideas off one another, and be better aware of what everyone was doing. This would have greatly improved our communication saturation. On several occasions, the lack of clarity led to questions and delays as we explained in the newly written code. This could have been avoided with better initial coordination.

We also had the same issues as other groups with using GitHub. There were several merge conflict issues and general push/pull issues. While these issues were frustrating, they were part of the growing pains of learning to use Git, the agile methodology, VS, and C# all at once, together.