Reengineering Lockheed Martin's Customer Portal

Alexander J. Zevin
University of Connecticut, Storrs, CT, USA
Email: alexander.zevin@uconn.edu
Sponsored by Sikorsky Aircraft, Advised by Dr. Seung-Hyun Hong

***Abstract*—Sikorsky Aircraft is a Lockheed Martin Company which principally manufacturers helicopters for civilian and military use. Sikorsky customers interact with Lockheed Martin in a self-service capacity using a Customer Hub Web Portal. Our Computer Science Senior Design team rebuilt Lockheed Martin's Customer Hub Web Portal from an architecture and functionality perspective. Using Amazon Web Services (AWS), Java, Spring Boot, and Angular, our Senior Design team developed a feature rich customer portal prototype with microservices which allow customizability and scale, so the application meets future user needs. Our team also researched software development principles, including Scrum, an Agile technique which allowed flexibility in responding to stakeholder requirements and development of our prototype with speed.**

Keywords—customer portal, web application, Agile, Sikorsky

I. Overview

Lockheed Martin is a global security and aerospace company engaged in the research, design, development, manufacture, integration and sustainment of technology systems, products, and services, which acquired Sikorsky Aircraft from United Technologies Corporation in 2015. We partnered with Sikorsky in August 2020 to develop a redesigned Customer Hub Web Portal (customer portal) as part of a year-long Computer Science Senior Design Project at the University of Connecticut.

Prior to partnership, Sikorsky maintained a legacy customer portal, Sikorsky360, which used outdated architecture built on AngularJS. Sikorsky noted issues with performance and difficulty managing years of code development which interacted with many external databases and systems. Sikorsky provided our team flexibility to redesign their customer portal with technologies and languages of our choosing.

Our Senior Design team was provided guidance of features the customer portal should provide users. Among those features: a secure sign-in and authentication process, a dashboard that can be customized depending on the type of user logged into the customer portal, and the ability to order parts.

We met on a weekly basis with the Sikorsky team, consisting of a software developer, a product manager, and a business analyst. Meetings with a diverse team of Sikorsky stakeholders allowed us to consider not only technical requirements, but also business implications.

For instance, we were attentive to an additional adaptability requirement of our new customer portal, since it would be used by a variety of Sikorsky customers. Sikorsky provides products and services through the customer portal to military and civilian customers with varying interests, such as local governments like the Los Angeles Police department, Princes, and individual contractors who manage helicopter fleets. Users within these organizations interact with the customer portal in different capacities. While an accountant may be interested in retrieving cost summaries and tax documents from the customer portal, a helicopter owner is interested in a high-level view of their available helicopters, order status, and service bulletins. To meet this need, we built a modular dashboard consisting of distinct views depending on the user type, called widgets. This design philosophy cascaded to other considerations, such as how user information is stored in the database, and dashboard modularity to show or hide features relevant to the user.

In addition to weekly meetings with Sikorsky stakeholders, our team met with Dr. Seung-Hyun Hong, a Research Professor at the University of Connecticut School of Engineering, to discuss the timeline for our objectives and to present progress. In coordination with Dr. Hong and Sikorsky, we agreed on a timeline for the first semester of project development with key objectives to be achieved between August and December 2020, including: development of a login page, the ability to store user information in a database, a main dashboard screen presented to the user following login, a parts ordering checkout process consisting of product search and selection, and customer support live chat messaging. In the second semester of project development, which lasted between January and April 2021, our objectives included expanding the live chat function to include automated replies, a knowledge base, an event calendar, a vehicle status page, a recent orders widget, login tokens, and overall UI enhancements.

Following the agreed timeline, our team designated leaders for each milestone, including identifying sub-requirements and assigning tasks to ensure equal contribution and leadership opportunities. We adopted Agile methodology through sprints, iterative development with feedback from Sikorsky and Dr. Hong, and flexibility to adapt to emerging needs, such as implementing a database system to meet overlapping objectives effectively.

A. System Description

The prototype for our customer portal is written with an Angular frontend, Java backend, and the Amazon Web Services (AWS) SDK.

In Gartner's Programming Language Index Rating, Java is ranked as the most popular programming language overall [1]. Although other languages such as Python are quickly changing this state, we chose Java due to its reputation as a favored language worldwide, and familiarity of the language from our own team. The Java backend was constructed using Spring Boot, an open-source framework used to create microservices. Through Spring Boot, our team was able to quickly deploy the backend framework for the customer portal prototype without the need for additional configuration. The use of Spring Boot reduced the time required for development.

Sikorsky's legacy customer portal was built using AngularJS, an open-source JavaScript framework. Although Sikorsky was flexible in our choice of languages, for the front-end Angular was determined as a key specification by Sikorsky for the new customer portal. Particularly, the AngularJS version in the legacy customer portal is the first release of Angular, which has since been rewritten by Google. Angular is frequently used by developers and was the second most used language in the Frameworks, Libraries, and Tools category in Stack Overflow's 2018 Developer Survey [2].

The use of Amazon Web Services (AWS) was another specification communicated by Sikorsky. AWS is a foremost cloud platform service recognized as the leader in Gartner's 2020 Infrastructure and Platform Services category [3]. *Fig. 1* is Gartner's Magic Quadrant which demonstrates AWS's significant lead in the category among competing products from Microsoft, Google, and Oracle.

Fig. 1. Gartner Magic Quadrant for Infrastructure and Platform Services [3].

AWS complimented our Java backend well using the AWS SDK for Java.

The AWS ecosystem provides several database options, including Amazon Aurora, Amazon Redshift, and Amazon DynamoDB. Aurora and Redshift are Relational database services used primarily for traditional applications and e-commerce. DynamoDB is a NoSQL key-value database used for high-traffic web apps. Although all three services support Java, we determined DynamoDB would be the ideal database method at this time for the customer portal because of its scalability, which will be essential in handling Sikorsky's thousands of customers. Unlike SQL databases, NoSQL databases are designed for horizontal scalability, which is preferred for large data sets.

DynamoDB provides a local version (DynamoDB Local) allowing the database to be self-contained, which is optimal for developing and testing applications. Although we considered using DynamoDB Local, we determined it would be advantageous to access the DynamoDB web service directly, which would mimic a real-world use case. For our limited prototype use case, the cost was less than a dollar per month.

Amazon Cognito is an AWS service allowing sign-up and log-in for web applications through user and identity pools. In addition to user defined credentials, Cognito supports sign-in through services such as Google and Amazon, making it a flexible client for users with existing company services in these ecosystems. Cognito adds an additional layer of security to the customer portal. In addition to Cognito, we proposed the implementation of a Two-factor authentication (2FA) service in the login process. 2FA is an additional security measure which requires the user to submit a code generated through a mobile app or text to login to a web application. We first considered Duo, a Cisco company, to satisfy this requirement. However, Duo's per user pricing model was determined to be cost-prohibitive. We are also explored other options, including LastPass, Twilio Authy, Google Authenticator, and Microsoft Authenticator.

Throughout the project, versions of the customer portal protype were executed locally, but eventually moved to the AWS cloud using Amazon S3 and EC2. Amazon S3 is a reliable storage service known for its scalability and data availability. Amazon EC2 enables access of cloud-based servers.

One strategy our team used when designing the system is the use of mockups for how features should look and behave prior to implementation. This ensured all team members had a similar vision of the final product. For instance, we used a mockup for

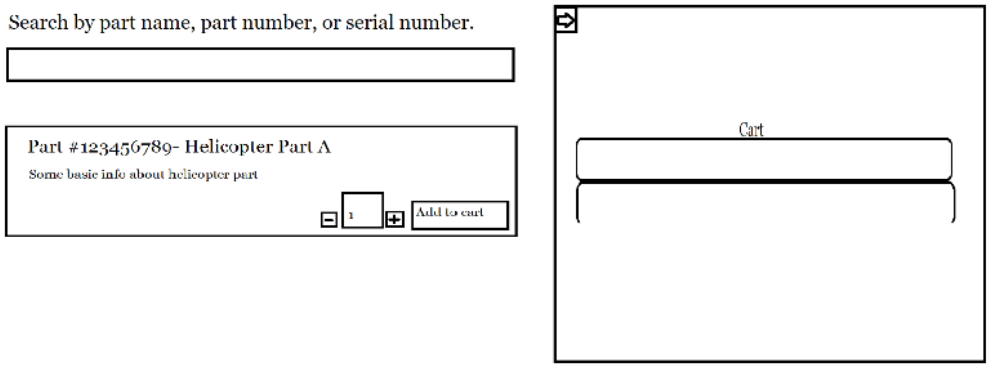

Fig. 2. Checkout process mockup (*Demonstrates our vision for part search, part summary, and cart*). The result looked nearly identical.

the parts ordering checkout process, which was a key milestone objective. *Fig. 2* exhibits the mockup which includes a search field, part summary field, and a cart used to store selected parts prior to checkout.

The final checkout process, shown in the User Manual section, looked like our mockup. Just like the mockup, there is an option to search for a part by name, part number, or serial number, the item is shown with the item name and the option to configure a quantity, and there is an add to cart button. Items added to the cart are populated on the right side of the user's webpage. In the future, the checkout process will also determine, and display part compatibility based on information about the aircraft owned by the user to ensure users do not accidently order incompatible parts. This will eliminate unnecessary costs due to expenses such as restocking fees.

Our team researched a variety of options for the customer support live chat messaging feature, which we called Sikorsky Support. The goal was to implement an easy-to-use feature that allowed users of the portal to connect and message directly with a Sikorsky agent in a manageable portal. Rather than developing such a feature ourselves, we decided to use a service called Tawk to achieve this goal, a service currently used by Information Technology Services (ITS) at the University of Connecticut. Tawk offered additional features to the live chat, including multiple-agent support, user tracking and real-time monitoring, and customizable widgets. Most importantly, Tawk was completely free for our use case. Although there may be other alternatives to Tawk on the market, for the sake of our prototype Tawk delivered all that we required. Deploying the Tawk service required configuration, which was completed mostly through the Tawk website.

In addition to the programming language technologies mentioned, the team has also used GitHub and Slack to communicate development progress. GitHub is a well-known hosting company for software development which allows developers to upload versions of code changes to the cloud, which can then be accessed by other users. The advantage of GitHub is it allows team members to work on multiple software files at the same time, and each file can be individually uploaded to the cloud without overwriting new code. Additionally, GitHub provides additional safety nets which can rollback changes if necessary. GitHub provides paid and free services, but for our use case the free plan was sufficient.

Since our team worked on the customer portal project at various times throughout the week, it was necessary to communicate changes to the code. This was done through documentation on GitHub, as well as messages through Slack. Slack is a modern messaging platform that allows teams to communicate among themselves. Slack is particularly advertised towards developers and teams within companies. Our team hosted a Slack chat room (channel) to communicate with each other and members of the Sikorsky team. This capability ensured we could communicate among each other quickly, without the need for email communication. Slack is a freemium service, meaning it provides both free and paid plans. For our use case, the free plan was sufficient.

Prior to deciding on Slack, our team had researched the possibility of using Microsoft Teams as our primary communication platform. Microsoft Teams has several advantages over Slack, including enhanced integration with other Microsoft 365 services. We considered the real-time shared document editing feature from Microsoft Teams an advantage over Slack. Ultimately, we settled on Slack because of its simplicity and familiarity from our own team from previous software development projects.

B. User Specifications

The customer portal will be primarily used by three categories of users: (1) domestic, international, and private militaries, (2) government entities such as the previously discussed Los Angeles Police Department, and (3) commercial customers.

Military and government users are similar in their use-case. Both require a standardized set of the previously discussed core features including the ability to view service bulletins, order parts, and view maintenance information and history.

Commercial customers are a more diverse category of users. An example of a commercial customer is helicopter rental companies. These companies own helicopters directly but lease their property to their own clients. In this use case, a helicopter rental company primarily would be interested in the ability to keep track of their helicopter fleet by quickly determining when their helicopters leave and return. This demonstrates the importance of adaptability of the customer portal. The previously discussed widget system could satisfy this type of

user by providing an appropriate view, such as a scheduling calendar, directly in the dashboard.

User profiles will determine which features a customer has access to. In the current customer portal, we store information of the user type in the database which will be referenced as we build new features next semester which may be appropriate for only a certain subset of all customers. In addition to the three broad categories of users, various subset of users exists, including engineers, helicopter owners, and accountants..

Users will be provisioned a username and password by Sikorsky system administrators prior to accessing the dashboard. Alternatively, there will also be an option for users to submit their own information, which would then be approved by system administrators. Key information about the user, including information related to their user profile will be stored in the database. Information includes their customer category, user profile, and additional privileges applicable to the user.

Users will interact with the customer portal in a self-service capacity. This means that in addition to providing key features to the user, it is equally necessary to provide the user with an intuitive interface. Today, users primarily interact with the legacy customer portal through a mobile-centric interface. Therefore, a key consideration throughout development of the customer portal is how the portal will appear to users from a mobile devices.

The current design of the customer portal copies design philosophies employed by other companies. Navigation appears at the left part of the screen through a series of links to features that direct the user to corresponding feature pages. Users can minimize the navigation to center their attention on the current page. This design philosophy is intuitive, since all links to different customer portal features can be found in one recognizable column. Google is one company that implements an equivalent design philosophy in their web and mobile applications using a "hamburger menu." Equivalently, all application features can be found in one column that can be minimized using a three-line menu image that resembles a hamburger. We expect users to have experience with this sort of user-interface, since it is employed by companies like Google. The user's familiarity to existing interfaces will lessen the learning curve associated with using the customer portal, which will reduce impact on Sikorsky Customer Support, who presently need to guide customers in accessing unintuitive features in the legacy customer portal.

C. Software Development

During the development of our project, our team also researched software development methodologies. According to IBM Research, "Software development refers to a set of computer science activities dedicated to the process of creating, designing, deploying, and supporting software" [4]. In other words, software development is a systematic method used to design, launch, and manage software projects, such as our customer portal. Software development typically follows a life cycle, known as the software development life cycle (SDLC), which defines deliverables and the execution strategy from the start of a project to the end to ensure successful completion. There are various approaches used by teams and companies in their development process methodology, including: Agile development, continuous integration, incremental development, rapid application development, spiral development, and Waterfall development.

Popularized by the 2001 publication, *Manifesto for Agile Software Development* [5], Agile has become one of the most recognizable software development methodologies in modern times. According to the *Harvard Business Review*, 44 percent of companies report Agile practices well integrated throughout their development teams, and while eight in 10 organizations have committed to transitioning to Agile, more than half are in the process of doing so [6]. Agile follows an iterative approach which largely arose as a contrast to the Waterfall methodology employed by many organizations prior to the adoption of Agile. While the Waterfall methodology follows a linear model where objectives are executed sequentially, Agile uses a flexible continuous model where objectives are incrementally developed through feedback from stakeholders such as customers and managers. The primary advantage of Agile development is it allows faster execution with greater ability to respond to changes. These advantages demonstrate why many organizations are eager to adopt Agile.

There are several Agile practices, such as Scrum and Kanban, which can benefit teams with different needs. To this report, we will explore Scrum in length. Before doing so, we will first define and examine common misconceptions of the Agile philosophy.

1) The Philosophy of Agile: The term Agile is used frequently not only by software developers, but other individuals in business including marketers and consultants. For this reason, it has become difficult to distinguish what exactly Agile is. To be clear, Agile is not necessarily a methodology in its traditional definition, nor a technique of developing software, or even a distinguished process. Rather, in its original definition, Agile is a set of values and principles that guide organizations.

This distinction is unclear to many, and has led Agile to become a form of "cargo cult" in many organizations. The term cargo cult traces back to World War II, when planes would drop off cargo and war supplies on islands in the South Pacific. When warplanes no longer visited the islands, inhabitants would build fake airstrips and radio towers believing they would summon planes. In the same way, when organization leaders learn about Agile, they believe it will lead to greater efficiency and cost-savings, so they adopt what they believe to be Agile techniques, but fail to recognize that it is in fact Agile values and principles which lead to organizational success.

For example, a daily-standup is one technique used by Agile organizations. In a daily-standup, team members will meet each day, typically at the same time, to inform everyone of information vital to project coordination. Team members discuss their individual completed tasks, as well as obstacles they have encountered. The expected benefit of the daily-standup is to encourage information sharing, and to address

individual obstacles before they hinder the larger team. A daily standup appears fine in theory. One would expect daily-standups to lead to greater organizational efficiency and communication. However, this is not always the case. Often, daily-standups become simply a form of "status reports," where team members report information to a team manager, rather than exchange information among team members. Additionally, daily-standups can become a "no problem" meeting where no members even raise any challenges or obstacles, even if they exist. This could be the result of fear of admitting mistake, or lack of confidence. While team members may find that even in this case daily meetings are helpful, they are not living up to Agile values and principles. Rather, the organization is simply repeating a set of rituals which provide little realized benefit.

What are Agile values and principles? The *Manifesto for Agile Software Development (Agile Manifesto)* provides a clear explanation in only 68 words. The *Agile Manifesto* makes a series of comparisons of which ideas organizations should value in order to improve software development. The *Agile Manifesto* calls to value: (1) individuals and interactions of processes and tools, (2) working software over comprehensive documentations, (3) customer collaboration over contract negotiation, (4) responding to change over following a plan [5]. While the *Agile Manifesto* does not entirely discount the value of the latter portion of each point, the *Agile Manifesto* does call for greater value to be placed on the former part.

Principles Behind the Agile Manifesto provide additional clarity of how Agile can be accomplished. The principles state: (1) focus on customer satisfaction, (2) welcome requirement changes, (3) deliver working software frequently, (4) business people and developers should work together, (5) encourage a motivated team that receives the support they need, (6) face-to-face conversation is the most effective method of conveying information, (7) measure progress by working software, (8) promote development that is sustainable, meaning all stakeholders can stay at the same pace, (9) pay attention to technical excellence and good design, (10) encourage simplicity, (11) teams should be self-organized, and (12) the team should reflect on how to become more effective, and make adjustments as necessary [7].

Combined, the values and principles of the *Agile Manifesto* and *Principles Behind the Agile Manifesto* provide direction for making decisions within an organization. For example, rather than establishing set requirements from a customer through a contract prior to project development, a customer ought to be involved in the software development process, by *Agile Manifesto* (3). Additionally, *Principles Behind the Agile Manifesto* (4) encourages business people and developers to work together. This means not only developers within the customers organization, but rather members at all levels of the business should work together. Likewise, within one's own team, business leaders should work together with developers to accomplish projects.

Within our own customer portal project, we appealed to the Agile values and principles when developing the DynamoDB database. As we were working on the customer portal, it became clear that we would need a database in order to accomplish certain tasks, including storing user and part information. Rather than immediately changing goals to focus on building out the database, we completed relevant parts of the portal including the sign-up and log-in page which were set to be delivered and shown to Sikorsky. Rather than changing objectives, we remained flexible and completed the task at hand, understanding we would need to go back and build the database. We valued delivering software frequently by building what was necessary of individual features, rather than delaying the development of those features. Eventually, we were able to work back and build up the DynamoDB database integration without cost to the tasks which were previously at hand.

Agile teams adopt practices which support Agile values and principles. When selecting practices, functional Agile teams will consider why they should choose practices, rather than what practices they necessarily choose. A team should ask, "How will this practice promote Agile within our organization?", and "Will this practice fit within the scope of the organization, and will it provide benefits to efficiency?" Though teams should not look for practices believing they alone will transform the team into an Agile one, evaluating existing practices can be beneficial in finding what works for a particular team. One of such practices is Scrum.

2) The Philosophy of Scrum: Understanding that practices alone do not make an effective Agile organization, they can be helpful in determining the team direction. Scrum is an Agile practice used by approximately 90% of all Agile teams according to Forrester Research [8]. Fig. 3 shows the growth and dominance of Scrum compared to other Agile and non-Agile practices within organizations. While Scrum has become the standard in organizations, both Kanban and Lean, two other Agile practices, have been growing in organizational adoption rate.

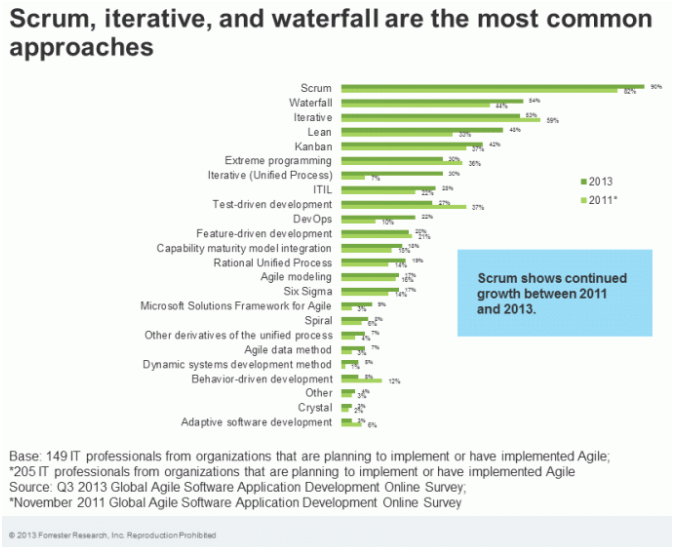

Fig. 3. Forrester Most Common Software Development Practices [8].

Scrum was largely developed through a collaborative effort between Ken Schwaber and Jeff Sutherland, who both contributed to the previously mentioned *Manifesto for Agile*

Software Development, which assisted in the spread of Scrum worldwide. *The Scrum Guide* [9], written by Schwaber and Sutherland in 2009, established the Scrum philosophy in a freely available publication.

According to *The Scrum Guide*, "Scrum is a framework for developing, delivering, and sustaining complex products" [9]. Simply put, Scrum is a method of solving problems in a way that would provide "the highest possible value." Time is a central focus in Scrum, with projects being divided into manageable fixed-length "Sprints," which are a set of tasks achieved by the team in a period of time, typically up to thirty days. The goal of Scrum is to ensure that a product is incrementally being improved upon, and can be ready for shipment or evaluation by the product owner at any time. Like Agile, team inclusiveness is important in achieving tasks. Both business and technical stakeholders work together to ensure all members are working to achieve a common goal. Success is measured by functional software that showcase features planned upon in an earlier Sprint.

Scrum is effective when dealing with unexpected obstacles, or changes in goals. Since work is divided into manageable Sprints and tasks for each Sprint are re-evaluated often, the team can adapt to customer requests or changes. At the same time, the customer always has an idea of the status of a project, and what features have been implemented. Customers can actively voice their input to drive product changes, ensuring that the product more accurately fits their vision. This feedback ensures that time is not being wasted on designing features which do not meet customer expectations, leading to greater efficiency and cost reduction in billable hours.

An assumption of Scrum is that a problem cannot be entirely understood from the start, but instead requires specification refinement during the duration of a project. Rather than focusing on completely understanding the problem at once, the team works to break the project into manageable parts, and work on them incrementally with the goal of working towards solving the larger problem. This problem-solving method is similar how we solve many day-to-day problems. Complicated problems can be broken into individual goals which can be met to solve a larger challenge. Therefore, Scrum requires an adaptable team capable of envisioning the end-goal, while not necessarily always focusing on it.

Scrum also emphasizes a self-motivated autonomous team. Since not all team members will always work on every task, individual team members should be able to oversee their particular tasks in a way that can contribute to the larger shared goal. Team members should work well with and trust one another since day-to-day goals may not always overlap. Weak teams will encounter additional obstacles if team members do not work well with one another.

3) Overview of the Scrum Process: The Scrum process starts with a product owner who defines as accurately as possible what features they would like to see in the final product. This wish-list of features is called the Product Backlog. The features in a Product Backlog are often, but not always, prioritized for the team to know which features are most valuable to the customer. The product owner does their best to represent the customer, and is considered a key project stakeholder themselves. Once these features are defined, the team will meet for Sprint planning, in which they select, with direction from the product owner, which tasks to complete in the upcoming Sprint. These tasks are moved to a different backlog known as the Sprint Backlog with the intent to complete all those tasks by the end of the Sprint.

During the meeting, the team will also discuss how to complete the tasks by dividing work, selecting technologies, and ensuring all team members are on the same page. The timeframe to complete the tasks, the Sprint, is also decided upon during the meeting. A Sprint should typically take no longer than 30 days. If the time-horizon is too long, risks such as product delay may occur. Sprints with long time-horizons may also overlook additional complexities which may arise during the Sprint which require change in direction. On the other hand, Sprints which are too short could also set unreasonable expectations. When features are rushed, there may be insufficient time to build them in a deliverable fashion. As a rule of thumb, the shorter the Sprint, the more Agile the team becomes.

After determining the length of the upcoming Sprint, team members will continue to meet for a Daily Scrum. The purpose of the mandatory Daily Scrum meeting is for team members to have an idea of the tasks they must complete during that day. Members discuss their progress towards completing goals in the Sprint, and all obstacles are addressed by the team. The Daily Scrum is essential in understanding whether the overall Sprint goals will be accomplished in-time. The Scrum Master has an important role in the Daily Scrum. The Scrum Master leads to the meeting in a way that ensures all voices are heard in a productive discussion. The Scrum Master must be careful to avoid the challenges of Agile, such as the potential for a meeting to become simply a status-update meeting, which is not necessarily in the spirit of Agile.

At the end of the Daily Scrum, team members should be aware of the progress accomplished by other members, what effect the accomplishments of other members have on their own tasks, and what tasks they should complete for the day. Any obstacles team members are facing should be addressed immediately, since obstacles can have a direct impact on the progress other members of the team. It is essential these sorts of challenges are not simply swept under the rug.

Scrum Masters should also gauge the confidence of team members in achieving their Sprint objectives. If there are too many obstacles, team members may feel less confidence about accomplishing their goals. Knowing how team members feel can give a good idea of whether the Sprint will be successful or not. This is important when evaluating the progress of a particular Sprint.

There are a variety of methods to gauge team member confidence levels. One method is a percentage system. For example, a member could report, "I feel 90% confident we'll achieve the Sprint goals." Members can also use a rating system, such as, "There's a 3/5 chance we'll achieve the Sprint goals." Finally, members could also use the OKR traffic light system, which categorizes goals in just three categories: red, yellow, and green. This allows members to focus on different solutions depending on the goal. A red OKR, which is assigned to goals

team members are not confident with, would seek to overcome obstacles and understand what is getting in the way. A yellow OKR, assigned to goals members are unsure about, would focus on understanding what has been learned from previously overcome challenges in order to determine how far off-track the team is. Lastly, a green OKR, which is assigned to goals team members reasonably believe will be achieved, will focus on the successes made, and ensure the team is learning from what is working.

During the Daily Scrum, the Scrum Master may use a Scrum Board to help visualize Sprint Backlog items. The board, which can be physical or virtual, categorizes tasks based on stories, tasks to-do, tasks in-progress, and tasks done. Stories ensure the team looks beyond simply accomplishing tasks, but rather are addressing the needs of the customer. A story could read, for example, "As an accountant, I want to be able to quickly download company tax information from a customer portal, so that I can complete my role as an accountant." Tasks should be designed and evaluated around these sorts of user stories. Items to-do are those stories for which work has not begun. In-progress items are tasks the team is working on and are discussed during the Daily Scrum, and items done are completed action items, which will be discarded after the Sprint is complete.

Daily Scrums are essential in ensuring the work at the end of a Sprint can be shipped to the user. At the end of the Sprint, the team will meet during a Sprint review to see if this is the case. Team members will demonstrate new features either formally or informally. Sometimes, teams will avoid the use of PowerPoint slides, for example, to focus on the demonstration instead of a set of bullet points. The Sprint review will take longer than Daily Scrums, and can last as long as four-hours. All stakeholders should be invited, including the product owner. The goal of the meeting is to gain feedback and promote collaboration.

A Sprint retrospective is the last part of the Scrum cycle. The Sprint retrospective is a meeting used to reflect on the strengths and weaknesses of the team, and whether the Scrum methodology is working properly. Members should be vocal about the changes they wish to see in the processes based on the successes or challenges during the previous Sprint. It is important the team can exit the retrospective having learned from the Sprint and understand which corrective actions can improve the next Sprint.

After the Sprint retrospective, the cycle will repeat once more until all items from the Product Backlog are complete. The product owner may add additional tasks to the backlog, and the priority of those tasks may change. The backlog can also be improved in a process known as backlog refinement. In this process, the team will re-evaluate items in the backlog and ensure they are clearly defined.

The Scrum Master should ensure that the methodology is adequately working for the team, and adjustments are made if needed. No team is the same, and some teams may use different methodologies in combinations with Scrum which are discussed in the next section. The *Scrum Alliance*, which was founded by Scrum creators Jeff Sutherland and Ken Schwaber, offers a Certified Scrum Master course which trains Scrum Masters in effectively managing the Scrum process. Although not all team have certified Scrum Masters, this trained role has become more common recently. Many teams will have a single member whose primary responsibility is planning the Scrum process.

Fig. 4 shows a flow diagram for the Scrum process discussed in the section, including the Product Backlog, Sprint Backlog, Sprint cycles lasting 2-4 weeks, daily progress over 24 hours shown in the Daily Scrum, and finally the potentially shippable product towards the end of the Sprint.

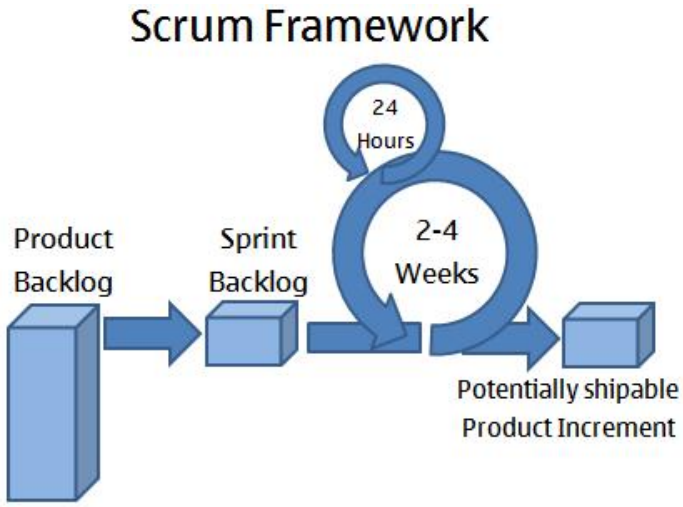

Fig. 4. Scrum Framework Flow Diagram [9].

4) Scrum Compared to Other Methodologies: Although Agile and Scrum have become more popular development methodologies in recent times, they are not necessarily always the most effective or best ones to use for a team. Agile and Scrum require commitment from a self-motivated team to successfully achieve goals, and can suffer when team members are not adequately motivated. The methodologies also require experiences members who can carry out their duties independently when necessary. Finally, Agile and Scrum can cause frustration among team members because of the extra steps they require. Not all members may see the need to meet on a daily-basis for a Scrum daily, for example. Rather, members may see these sorts of activities as a chore.

Waterfall is a methodology which is perhaps most dissimilar to Agile and Scrum. Waterfall is a method which approaches projects in a linear series of steps. Traditionally, those steps are (1) requirements, (2) analysis, (3) design, (4) coding, (5) testing, (6) operations. Most Waterfall methods begin with defining requirements. Both system and software requirements are recorded in a product requirements document. The document tells all stakeholders all pertinent information about a product, allowing some leeway for developers to decide how the task should be accomplished. Next, requirement analysis will be conducted on the product which determines more clearly what the needs of the product are. Often, analysis will result in models that organize information to be used further on in the project. Once requirements have been set and analysis has been conducted, the team will begin the design phase of the project. The software architecture will be determined, and all decisions critical to the coding will be

made. Coding is the next part of the Waterfall methodology. In this phase, most of the software development will be executed to achieve the project objectives. Finally, code is tested to ensure there are no bugs in a verification stage, and the end-result is delivered to the customer. From there, during the operations phase, maintenance of a system may occur, but typically no additional features are added. *Fig. 5* depicts a flow diagram for the Waterfall steps discussed.

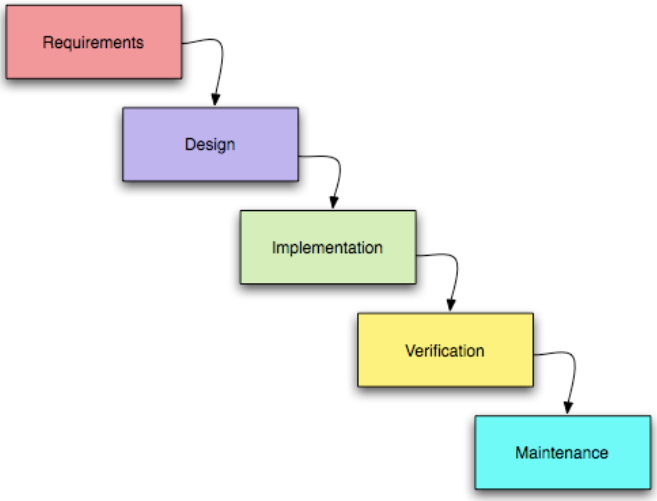

Fig. 5. Waterfall Framework Flow Diagram [10].

The primary difference between Agile, Scrum, and Waterfall is the lifecycle of a project. While Agile and Scrum are iterative in nature, where projects are completed in cycles of planning, analyzing, designing, deploying, and improving, in Waterfall there is one linear order in which tasks are completed. All project requirements are stated up-front in the Waterfall model, whereas in Agile and Scrum, objectives are prone to change. Consequentially, using the Waterfall method may cause code to be different from what the customer had expected. Since customers are not always kept informed, and cannot change requirements as necessary, developers must do their best to ensure the customer's vision is executed.

The benefit of Waterfall is often cost and predictability. The customer has a good idea of when a product will be delivered and over what period of time. Since there is no iterative process, the customer can expect to have a finished product, although it may not always be to their vision, at the end of the project timeline. The customer also has better control of the price for the project, since expectations are defined and often solidified at the start of a project. For the customer, balancing cost and predictability will have a direct impact on the end-product. A rushed product produced at a cheaper price may not have the features the customer expected. Also, if a customer expects a product to be predictably delivered on a certain date, developers may cut corners to deliver on-time.

For Waterfall to work effectively, expectations must be made clear to the team from the start. This comes through extensive documentation and analysis which can lead to good design and implementation. When executed properly, Waterfall can be an easier method to employ, and is particularly useful for less-experienced teams. Teams are able to perform their duties without getting bogged down on the additional obligations of Agile and Scrum.

Rapid application development (RAD) is a software development methodology similar to Agile. Like Agile, RAD emphasizes an iterative development process which focuses on prototyping and development, rather than on planning. While in the Waterfall method, careful planning ensures the final product meets the customer's expectations, RAD uses prototypes to demonstrate requirement completion to the customer. When requirements are set, they are often loose, allowing the developers to get an idea of the customer's vision, but not all the specifics. Next, developers will build a prototype that captures their interpretation of the customer's requirements. Prototypes will not always satisfy all the customer's expectations, and often most features will not be fully-functional in order to save time. After the prototype is created, developers will show the prototype to the customer and receive feedback. Just like in Agile, RAD assumes customers may change their minds when they review progress of the project. Using the feedback gained, developers will return to a cycle of prototyping and collecting new feedback. When the customer is satisfied with a prototype, developers will finalize it. During the finalization stage, developers may re-engineer the product, but the core features and design will remain unchanged.

5) Scrum in the Industry: According to the U.S. Bureau of Labor Statistics, in 2018 the demand for certified Scrum Masters grew by 24 percent, and is considered one of LinkedIn's "Most promising jobs of 2017" [11]. These statistics demonstrate the demand and growth of Scrum in the workplace. Technology firms such as IBM, Google, and Spotify, have used Scrum practices within their teams to develop products, and are some of the employers of Scrum Masters. Scrum is used not only by technology firms, however. Scrum has also been used in other industry sectors including education, marketing, and the military.

Many universities and local-school districts have adopted the use of Scrum to organize and collaborate. In this setting, Scrum can be especially beneficial in enabling a variety of teams to collaborate with one another. One of those local-school districts in the one in Chesterfield County, VA. Chesterfield uses the Agile and Scrum framework to manage a school system of over 60,000 students [12]. Throughout the week, the central office team holds 15-minute Daily Scrums to check in with one another and define objectives to be met for the day. According to the district's Chief Academic Officer, employees are able to achieve more throughout the day as a result of employing Scrum methodologies. The school district has also encouraged teachers to assign projects to be completed collaboratively using Agile and Scrum, which enables students to work more creatively.

Scrum has also become popular among marketing firms. Using Scrum, teams within the firm assign marketing goals in the form of Sprints, meet daily to ensure goals are being achieved, and respond to feedback from other stakeholders to ensure the success of marketing campaigns.

Finally, in recent times, some militaries have also adopted Scrum practices. According to a study published by IEEE, military operators reported a satisfaction level of higher than 90% when using Sprint deliveries through Scrum [13]. Just like in software development, some military operators use Scrum boards to keep track of projects and maintain a backlog of objectives needed to be accomplished. This can assist commanders in achieving their goals and provides a better experience for soldiers in training.

6) Scrum for Development of the Customer Portal: After evaluating the several software development methodologies, our team concluded that Scrum was a good contender for our purposes, with some modification. Scrum is an ideal methodology because it focuses on iterative development through constant feedback and collaboration. In our own experience, we received feedback from Sikorsky as we built features, and used the feedback to make changes as needed. Rather than providing us with a set of requirements, Sikorsky enabled us to make decisions on our own, which allowed us to shape the direction of the project. We also found a number of Scrum tools, such as the Scrum board and daily standups to be useful in coordinating tasks.

Our team was unable to fully utilize Scrum during the first semester, but made improvement in the second semester by using an agile management tool called Jira. Jira is a product tracking tool which allows assignment of tasks to team members. We also met with one of the Sikorsky team members, who is a certified Scrum Master. Using more Scrum tools, we were able to work more efficiently on our project.

We modified the Scrum method to work better for our team. In a traditional Scrum framework, teams work on a daily basis to review team status. In our use case, it was not be feasible to meet every day to work on the project. Over the course of the project, we spent certain days and times to work on the project individually, and met throughout the week at different times. Therefore, we would fit the meetings around our individual school schedules.

Our workstyle somewhat reflected the Rapid Application Development framework. Our team was approached with a broad set of requirements, and it was up to our team to conduct research, envision a solution, and build a prototype. Throughout our meetings with Sikorsky, we demonstrated features in the prototype and made changes as necessary. Although this method did work for our team, we understand the value of trying new tools such as Scrum.

7) Scrum for Development of the Customer Portal: As mentioned previously, Scrum is not always ideal for every team. At the core of Scrum is the Agile principles defined in the Agile Manifesto. Without working in the spirit of Agile, Scrum is unlikely to be an effective development methodology for a given team. Scrum also requires additional time and resources in order to implement. Typically, a team will designate a member as a Scrum Master who oversees the Scrum process. This can create additional workload for that member. If we were to use Scrum next semester for our own project, designating a member as a Scrum Master may cause a conflict in workloads, which may be unfavorable. Scrum also takes time to plan. Not all teams may have the luxury of spending up to four hours to review Sprint progress in addition to daily meetings. Particularly in the use case of our team, other school obligations throughout the semester may prohibit our team from dedicating extra time in effectively using the Scrum method.

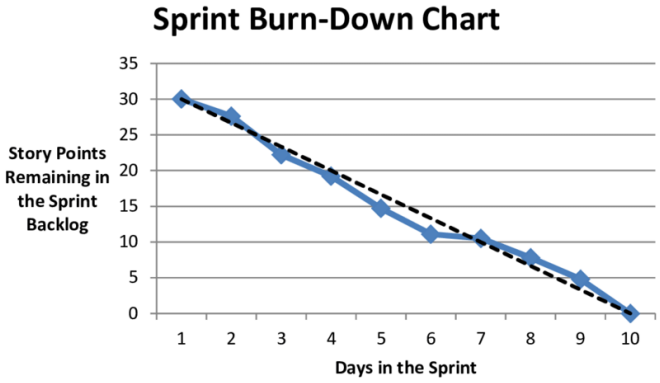

Fig. 6. Sprint Burn-Down Chart [14].

Several extensions of Scrum can be used to provide data-based feedback to the team on progress and effectiveness of the Scrum framework itself. One of those extensions is the Sprint Burndown chart. The Sprint Burndown chart displays the work remaining in a Sprint Backlog. The chart is updated daily and provides a simple view of progress that is being made. *Fig. 6* shows a sample Sprint Burndown chart, with days left in a Sprint in the horizonal axis, and work remaining each day in the vertical axis. This extension of Scrum is favorable because it can encourage team members by visualizing the work that has already been completed.

Another Scrum extension is velocity. Velocity is a numerical value calculated when a Sprint has ended that measures the amount of work done. The value is computed by totaling points earned for each user story that has been completed. A user story is an informal description of a software feature from the perspective of an individual stakeholder. The velocity metric is useful in determining the workload to be scheduled in future Sprints.

Despite its widespread adoption, Scrum has encountered many outspoken critics who critique it as wasteful. Scrum has hidden complexities that are not clear at first glance. For instance, when tasks change, a user must spend time updating tools such as Scrum boards. It is also not always possible to predict ahead every time when planning Sprints to determine whether they will be achievable. A simple task that may appear to take half an hour could end up taking several days. Therefore, while it may be advantageous to most teams, the use of Scrum should be carefully analyzed by a team and adopted gradually.

II. Changes since start

From the beginning of the project, our team was given freedom to come up with new features for the customer portal.

Few objectives were set in stone, except to develop a prototype that maintained core Sikorsky Customer Portal usability. This was largely completed during the first semester of the project, in which we worked on development of a login page, the ability to store user information in a database, authentication, a main dashboard screen presented to the user following login, a parts ordering checkout process consisting of product search, selection, checkout fields including authorizations and paperwork, order confirmation, and live chat support messaging.

In the second semester of project development, which lasted between January and April 2021, our objectives included expanding the live chat function to include automated replies, a knowledge base, an event calendar, a vehicle status page, a recent orders widget, login tokens, and overall UI enhancements.

Our team received positive feedback from our implementation of live chat, and Sikorsky was interested to see how this feature could be further improved. To that end, our team implemented an automated reply chat bot that could serve users with the information they requested without the need of a support agent. A user can, for instance, request information about availability of parts and place orders from within the chat system without needing to speak to an agent. This was implemented through a variety of triggers available within the Tawk software. When certain keywords were mentioned by the user, the software would serve relevant replies. This system was effective for our prototype but would be unfeasible in real world implementation due to the number of accidental triggers of these replies. We explored other options, but there were none available within the existing Tawk interface. Building our own live chat with automated replies would have been unfeasible for the scope of this project. The automated replies feature received positive feedback from Sikorsky and became one of the highlights of our portal. Since Sikorsky's existing customer portal does not have live chat, let alone automated replies, Sikorsky stakeholders found this to be an interesting feature that could be used in future implementations of Sikorsky's customer portal redesign.

We also developed a knowledge base using Tawk for users to access information in a self-service capacity. The knowledge base includes articles about using various features in our customer portal prototype, including placing orders. This feature is useful in combination with the live chat and ticketing system, since these articles could also resolve the need of a customer speaking to an agent, resulting in reduced load for live chat agents. Sikorsky had no existing knowledge base prior to our implementation. This feature proved useful, since it can be used for a wide variety of applications, since all information available to customers is stored in one centralized location.

Another feature we came up with is a vehicle status page. The vehicle status page provides information about a user's aircraft, including maintenance information. Sikorsky had told us that their helicopters already have sensors that can communicate the status of certain parts, including when they are due for maintenance. However, this information was not previously leveraged in any way. The vehicle status page displays this information to the user and provides them information about these sensors. A user can, for instance, see when their aircraft engine might require replacement, and act accordingly. This feature supports a preemptive maintenance approach, which provides data necessary to react to issues before they cause prolonged downtime for an aircraft. This feature was used to demonstrate our idea, since we were not permitted to retrieve actual sensor data from Sikorsky's databases. Instead, we created demonstration variables in our own database, which were then retrieved and displayed in the front end.

A calendar was built to further display this information, as well as communicate other events to the user. A user can schedule aircraft for repair and view these events in one location. Some users rent their aircraft to other parties, and this information can also be stored in the calendar, so users can see where their aircraft are, and predict aircraft downtime.

We had originally planned to add widgets and useful information that can be accessed briefly from the dashboard. We successfully implemented a recent orders widget to this end, which provides users with information about orders they had placed. Since widgets are part of the user's dashboard, this important data is communicated to the user immediately when they log into the portal. We had planned to add other useful widgets like this one, but time limitations did not allow it.

Another feature we had planned, but did not get to, is an admin interface that allows Sikorsky to add and edit users, set quotas, and approve requests. Because our project is a prototype, we deemed this feature to be unessential, since all these actions could be done in the backend through DynamoDB. Additionally, a feature we would have liked to implement, but that could be beneficial in the future, is multilingual support. This would have been a great feature since a large portion of Sikorsky customers do not speak English natively. However, we found our team lacked multilingual skills to effectively translate the pages, and other methods for translation were imperfect. Since most users of our prototype are native-English speakers, this also was an objective we decided was unessential.

Lastly, out team received an informative presentation from a UI expert at Sikorsky, who provided recommendations of how we could improve the user experience. Suggestions included more natural methods of requesting information in order forms, alternative phrases to communicate options to users, and general aesthetic recommendations to make the portal feel more refined. After implementing these changes, our portal received positive feedback from an aesthetic point of view.

All the features we added in the second phase of the project further enhanced the portal with new ideas our team came up with after gaining feedback from the prototype developed during the first phase of our implementation, and through conversation with Sikorsky stakeholders. They polished our prototype to effectively demonstrate what could be accomplished with some creativity and new ideas. To this end, our team believes we were successful.

III. Accomplishments

The result of our project is a refined customer portal prototype which will be used to redevelop Sikorsky's outdated customer portal. Due to the scope of such a change, particularly due to security issues, our prototype will likely not be used

directly by Sikorsky. However, it will be used as a starting point. Particularly, the new features we developed will demonstrate capabilities that could be implemented by Sikorsky in the future. Almost all the new features we developed were not previously implemented and were only ideas. Being able to showcase these features was a valuable contribution from our team and an accomplishment. We successfully designed a prototype using these new features that a user can interact with directly from their browser. Any user can create an account, log in, and immediately see the with new features. Although we did not get to all objectives, we achieved most of them, and they contribute to a comprehensive user experience.

As a team member, I came up with new ideas and lead the development of some of the new features, including live chat and the knowledgebase. Continuously building upon feedback, I achieved and exceeded expectations from Sikorsky and the rest of my team members. I also contributed to all other parts of the project to ensure they were completed.

We concluded our project with an hour-long presentation to Sikorsky stakeholders, including business executives. The leaders were impressed with our ideas and implementation, and specifically our ability to come up with new features with limited direction from Sikorsky. Although our project is only a prototype, it will be used as a starting ground for future redesign of Sikorsky's customer portal and will be used to demonstrate the value of new ideas, such as the live chat, knowledgebase, calendar, widgets, and vehicle status information. The positive feedback we received reaffirmed that we completed the objectives of our project.

IV. WHAT I LEARNED

As a team, we agreed this project was beneficial for us prior to entering the workforce. We gained a good understanding of how software development is conducted within an organization, including the previously designed software design methodologies. Speaking with Business Analysts on the Sikorsky team also provided us insight into the various business decisions that go along with software development. When developing features, our team needed to be conscientious of how features would be used by end-users, and what their overall needs were. For example, when designing the live chat service, we wanted to make the experience seamless not only for customers of the portal, but also for support agents, who would be interacting with customers. Support agents needed tools to provide users with effective support in a text-based form of communication. The knowledgebase was one of the design decisions we employed to resolve this.

Most importantly, our team agreed that our good teamwork was essential in the success of our project. Our team members brought different strengths and weaknesses, different skills, and different interests. Dividing the work amongst ourselves, we ensured everyone participated equally and worked on features they were knowledgeable about. All team members worked on all parts of the project, but we also specialized so some team members worked on the front end, and some on the backend. Overall, we were satisfied with each other's progress. Therefore, learning to communicate with one another was a valuable outcome of this project, and we did so effectively.

As a team member, I came into the project never having worked on a Java backend system, and this was one of the first software development projects I have worked on. This made the project an extremely beneficial learning experience, as I gained exposure to different development techniques. All our team members benefitted from support from Sikorsky software developers, who spent time reviewing code with us, and teaching us new techniques. Having a completed this project, I feel more confident in my coding skills, and am excited to work on other software development projects in the future.

V. REDESIGNING THE SYSTEM

The first semester of our project provided our team with a good foundation to build new features on. We had already implemented login authentication, and core features including a dashboard. Thankfully, most of these systems did not require changes, which allowed us to easily implement new features. Our database, DynamoDB, was flexible and we successfully implemented new features using it, the backend allowed for the new features we desired, and the frontend was easy to manipulate and required no major changes. We were satisfied with the core user interface, and changes here were mostly cosmetic. When adding new features to the portal, such as the calendar, we could easily create new pages and deploy them in the navigation menu. This meant we could roll out these features in a modular fashion, and they did not break existing ones.

Login implementation was one of the challenges our team encountered during project development. Our goal was to implement 2-factor authentication and JSON web tokens for login security. 2-factor authentication was difficult to implement, since there are few free options available. Due to the budgetary constraints of our project, we decided this feature would be infeasible. If the system were worked on in the future, 2-factor authentication would be useful to add as an additional layer of login protection for users. JSON web tokens was also an extremely difficult task to implement because when we originally created the backend, we used two different frameworks for user creation and management. To set up the token exchange process, we had to determine which framework to use, and ensure the relevant methods and objects were used correctly. Although we successfully completed implementation of JSON web tokens, it took a long time and delayed development of other features which were contingent on using the tokens. Nonetheless, implementation of JSON web tokens was essential in the development of the portal and was a valuable learning experience for our team.

Deciding on a software methodology was also a challenge for our team. We originally wanted to use Agile/SCRUM, but due to the touch and go nature of development, it was difficult to follow the methodology as intended. In the second semester, our team committed to Agile and began using Jira, an Agile tracking board where tasks were posted. This helped somewhat but was ultimately an incomplete solution. Although our team agreed that Agile provided the most flexibility, if we were to redo the project, our team would pick any methodology, even if it were not Agile, if the methodology was flexible enough for our use case. Under other circumstances, we still think Agile is the best methodology for this sort of project.

VI. THE IMPACT OF THE SYSTEM TO THE COMMUNITY

Sikorsky's previous customer portal was outdated. It was built on aging technologies, lacked useful features, and needed to be revamped. Our project will have a lasting impression on how the customer portal is rebuilt, and we successfully tested new ideas. Although the larger project will require additional effort from Sikorsky, mostly due to its scale and security concerns, we were successful in deploying a valuable prototype. During our final presentation to Sikorsky stakeholders, we were able to present our new ideas, and they were well received. Some features, like the live chat and vehicle status page, were previously considered, but no action had yet been taken by Sikorsky. We demonstrated that they fit well within the customer portal and improved the user experience.

We also demonstrated how Amazon Web Services (AWS) could be used as the engine for many of the portal's components. For example, Sikorsky's previous customer portal is powered by a spread of many different databases, making development difficult. Our team was able to power all our features with a single database.

Ultimately, stakeholders were impressed with what they saw, and we believe it will lead to further investment in Sikorsky's customer portal. The result of this investment is a better experience for Sikorsky customers, which will benefit the company overall. When users can interact with their aircraft seamlessly from within the customer portal, they will be more likely to remain within the Sikorsky ecosystem of products.

VII. PRODUCT USER MANUAL

The customer portal prototype can be hosted from the external link (1) when an instance of EC2 is launched in Amazon AWS. To minimize cost, the instance is disabled when not in use. When accessing the link, users will first be presented with the login and signup webpage.

newsikorsky720.s3-website.us-east2.amazonaws.com (1)

The application can also be run locally. The files for the local version of the application are located on our private GitHub page (2). The GitHub page has been marked as private since it contains AWS access keys which should not be made public.

https://github.com/jackaaron25/SeniorDesignProject (2)

To run the project in the Linux/Ubuntu/Mac Terminal, the user will need to have Java version 8 (3) and Maven (4) installed. Installation instructions can be found at:

https://www.oracle.com/java (3)

https://www.maven.apache.org/install (4)

Once these are installed and working, in the Application2/application folder, execute commands (5) and (6):

mvn clean install (5)

java -jar target/application-0.0.1-SNAPSHOT.jar (6)

Inside the Sikorsky360-Frontend folder, execute commands (7) and (8) to launch the application:

npm install (7)

ng serve --open (8)

These commands will take up to several minutes to complete. After execution, the application will launch. Once launched, an administrator will need to provide an AWS access key and secret key to connect with AWS. When a user enters the application, they are presented with the login screen shown in *Fig 7.*

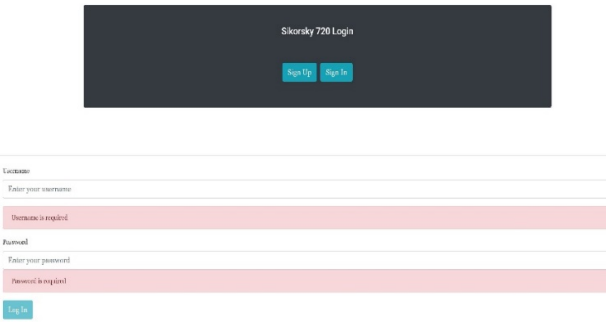

Fig. 7. Sikorsky 720 Login Prototype

The user is prompted for their username and password. If correct credentials are entered, the user will advance to their application dashboard. No action will occur if the user enters incorrect credentials. The login screen also provides users the option to sign up. The sign-up screen prompts the user for their email address in addition to a username and password.

Fig. 8. Sikorsky 720 Dashboard Prototype

Advancing to the dashboard, users are presented with their widgets in the center of the page. For now, there is one placeholder widget that prompts users to navigate to view their recent orders. On the left side of the screen, the user's name and email address are displayed. A navigation menu allows users to switch pages and access features including Part Ordering and Sikorsky Support, the live-chat messaging feature. There is also an option to navigate to Sikorsky Support through a convenient button at the bottom right of the screen. Finally, the top of the dashboard screen allows users to minimize the sidebar and log out. The top of the dashboard screen, the sidebar, and the live chat button in the bottom right of the screen are persistent in all other views of the portal. *Fig. 8* depicts the dashboard from a user's point of view.

When navigating to the Part Ordering page, users are prompted to search for parts by their part number. The search field automatically populates with suggestions based on user input. On the right side of the screen, the user can preview their cart, which stores items selected by the user. Finally, the user can advance to the next step of the ordering process, which will be completed during the second semester, by interacting with the checkout button. This interface is previewed in *Fig. 9*.

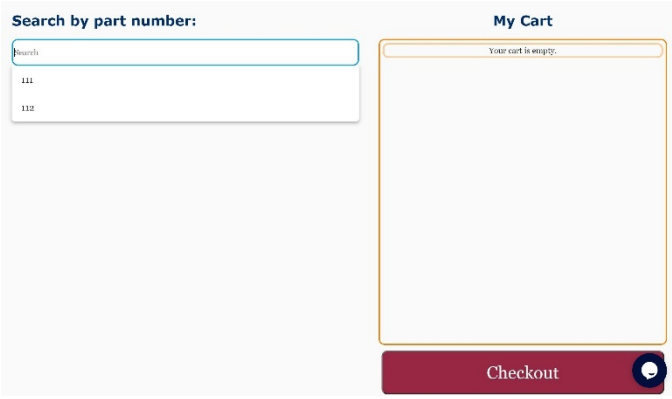

Fig. 9. Sikorsky 720 Part Ordering Prototype

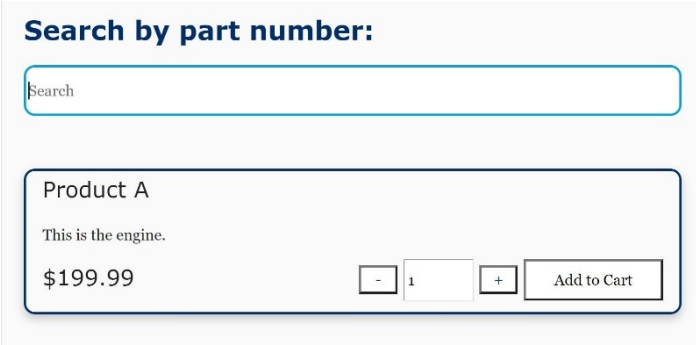

Fig. 10. Sikorsky 720 Part Ordering Information

Fig. 11. Sikorsky 720 Part Ordering Cart

Fig. 10 provides an example of the order preview after the user has searched for an item. The preview includes a product title, a short description, a price, the ability to control the quantity of the ordered part, and an add to cart button. *Fig. 11* demonstrates the user's cart after an item has been added.

All product information is stored in the DynamoDB database and can be manually updated by Sikorsky stakeholders. This allows parts to be removed or added quickly without needing to change the customer's experience or switch the system offline for maintenance. The product information is fetched from the database by the user when they search for an item. Items can be searched for by part number or other fields, such as the part name or even cost.

Users are told if parts are in-stock in the part preview window, and whether the part is compatible with a selected aircraft. Upon filling their cart, users will be provided the option to fill out a purchase order for their parts, including forms the user can digitally fill and submit. As the user advances through the checkout process, a preview of the parts they are ordering will remain in-sight.

Sikorsky Support is the live chat messaging feature which allows users to interact with Sikorsky agents. Users can access the service through the navigation menu or through a convenient button located at the bottom right of their screen. Depending on availability of agents, the support window will dynamically change to inform the user whether agents are available to support them. If a support agent is unavailable, the user will be prompted to leave a message which can later be viewed by a Sikorsky agent. Otherwise, users can immediately begin chatting with a Sikorsky agent. *Fig. 12* demonstrates an active chat session between an agent and a user.

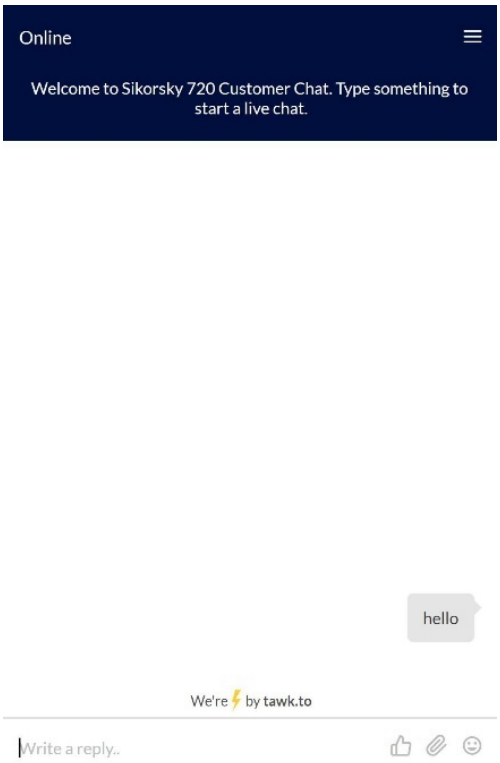

Fig. 12. Sikorsky 720 Support Live Chat Prototype

Sikorsky Support agents interact with customers through a live chat portal provided by Tawk. The portal allows agents to track users, including which pages they have navigated to. The portal also shows users other information about their user, such as their approximate location and IP address. Agents are prompted with a notification when a customer attempts to contact them, and agents are also able to contact one another.

System admins can add and edit users and part information directly through the DynamoDB database interface in AWS. Admins are provisioned an AWS account which allows them to access DynamoDB tables. Within the client table, admins can view all users by selecting items. Admins can add new users by clicking the create item option and can edit an existing table row by selecting it and choosing Actions. These features are depicted in *Fig 13.*

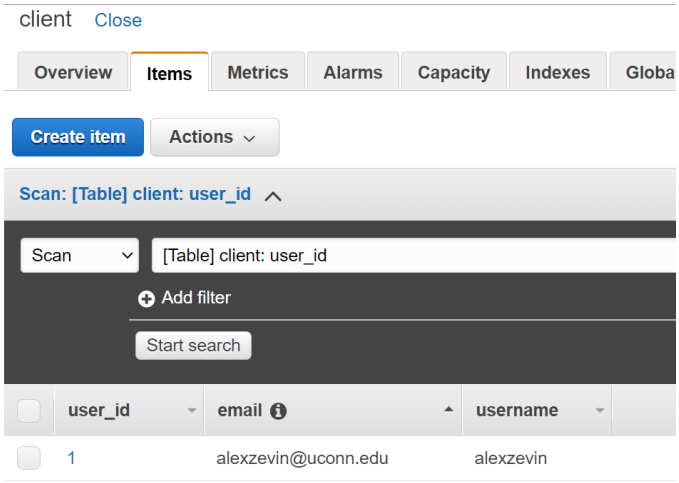

Fig. 13. Sikorsky Admin DynamoDB Interface

When users communicate in the live chat interface with certain keywords, they will receive automated replies based on triggers. *Fig 14* shows a demonstration of this system. In this example, a user sends, “Hello I have an engine that requires replacement.” The trigger replies, “Hi Alex, I see you own 3 aircraft with engines: Model A, Model B, and Model C. Which one are you having problems with?” The user replies, “Model A.” The trigger replies, “There are 4 engines in stock for Model A. Would you like me to place an order, and have it shipped to the address on file?” The user replies, “Yes! Thank you.” This triggers the system to place an order, and the chat communicates that an order has been placed. Not all queries are able to be resolved in this automated capacity. The user may ask, “I placed an order for an engine a few days ago, but it never arrived.” The bot will detect that the user has an issue which requires elevated support and connect the user to an agent.

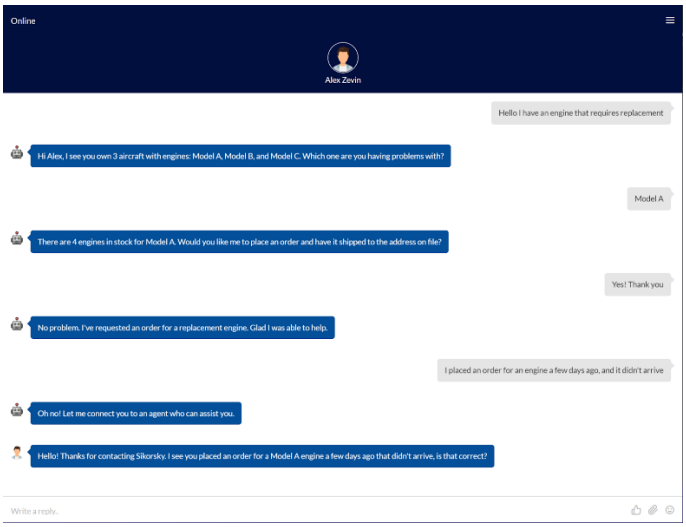

Fig. 14. Demonstration of automated replies in live chat

The user can also access the knowledgebase from within the navigation menu. The knowledgebase includes articles a user can interact with. *Fig 15* shows the knowledgebase dashboard with categories of articles a user can select from, including: orders, calendar, widgets, and vehicle status. A user can also search from within this window to find relevant articles.

Vehicle status is a feature implemented in the second part of the project and can be accessed through the navigation menu. An example of the interface is shown in *Fig 16*. Vehicle status provides the user with information about their current vehicles, and when maintenance is necessary. Part information is divided into three categories: satisfactory, pending maintenance, and unusable, among three example vehicles. The categories are color-coded to allow for quick readability. In our prototype, part information is included for rotor, engine and tail. In a future implementation, there could be more parts, and a more elegant way of selecting a user’s vehicle could be developed.

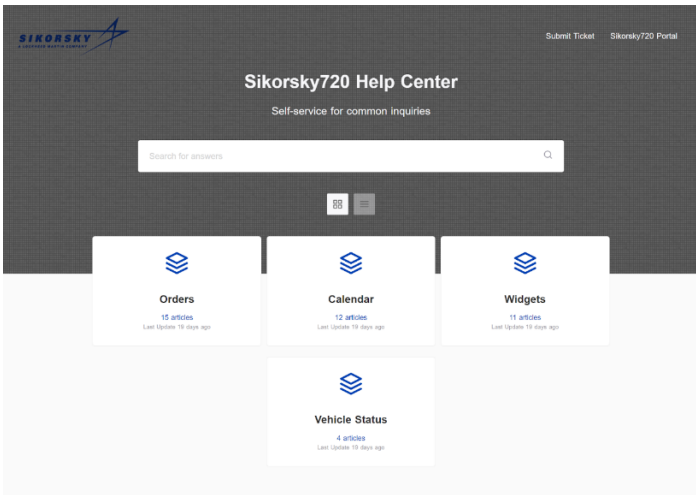

Fig. 15. Knowledgebase dashboard with articles

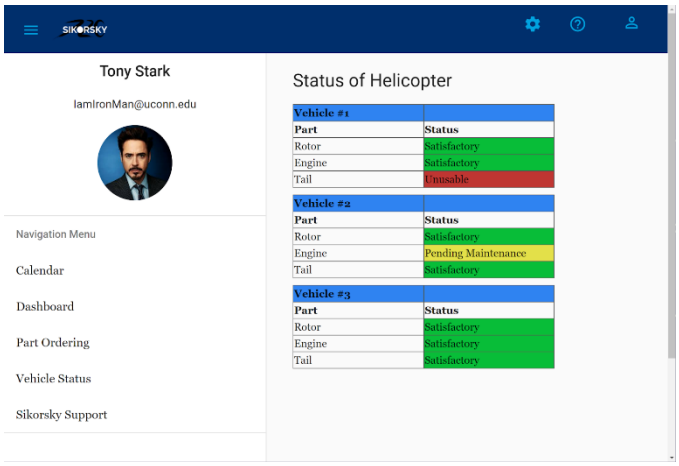

Fig. 16. Vehicle information pane shows status of a user's helicopters

Vehicle status also shows up in the calendar page, which is also accessible through the navigation menu. Additionally, users can see other scheduled events, such as when a helicopter is rented out. *Fig 17* provides a sample view of a calendar configured to a user. There are buttons for users to view the calendar by month, week, day, and in a list. Users can also toggle weekends and toggle the calendar itself. Arrows at the top of the calendar allow a user to navigate between different months. If a user would like to return to the current month, they should select the today button. The user can also add new events by clicking directly on a cell, and events can be deleted by clicking on the event and confirming deletion.

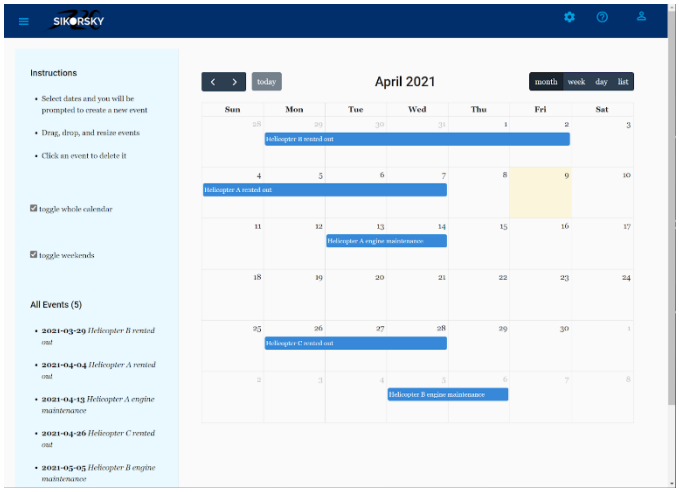

Fig. 17. Calendar view with maintenance and other scheduled events

VIII. Summary

Our year-long Computer Science and Engineering project culminated in a new feature-rich customer portal for users to interact with the products and services provided at Sikorsky. We developed exciting new features, along with basic features, that will enable the purchasing of products and services. We added features such as a live chat feature for customer support, a vehicle status widget, and a calendar widget, on top of the usual login, sign up, dashboard, and checkout features users would expect in a customer portal. All these features are built on an efficient architecture using Java, Spring Boot, Angular, Amazon Web Services, and Tawk. During the project, our team learned software methodologies, new languages, and had the freedom to come up with new, creative solutions. Our customer portal prototype will have a lasting impact as a starting point for future redesign of Sikorsky's legacy portal, and as a demonstration of new features Sikorsky can bring to the end-user, ultimately enhancing the user experience.

IX. References

[1] M. Revang, J. Duggan, M. Sobejana, and M. Driver, “IT Market Clock for Programming Languages, 2014,” Gartner Research, Oct. 2014.

[2] “Stack Overflow Developer Survey 2018,” Stack Overflow. [Online]. Available: https://insights.stackoverflow.com/survey/2018. [Accessed: 14-Nov-2020].

[3] D. Poccia, “AWS Named as a Cloud Leader for the 10th Consecutive Year in Gartner’s Infrastructure & Platform Services Magic Quadrant,” Amazon Web Services, 04-Sep-2020. . R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[4] “What is software development?,” IBM. [Online]. Available: https://www.ibm.com/topics/software-development. [Accessed: 14-Nov-2020].

[5] Fowler, Martin, and Jim Highsmith. “The Agile manifesto.” Software Development 9.8 (2001): 28-35.

[6] D. K. Rigby, J. Sutherland, and H. Takeuchi, “Embracing Agile,” Harvard Business Review, May 2016.

[7] “Principles behind the Agile Manifesto.” Principles behind the Agile Manifesto. Web. 10 Nov. 2014.

[8] D. L. Giudice, H. Kisker, and N. Angel, “How Can You Scale Your Agile Adoption?,” Forrester Research, Feb. 2014.

[9] Schwaber, K. and Sutherland, J., 2013. The Scrum Guide. 1st ed. [ebook] Available at: http://www.Scrumguides.org/docs/Scrumguide/v1/Scrum-Guide-US.pdf#zoom=100. [Accessed: 14-Nov-2020].

[10] V. Sauter and D. Hughey, “Comparing Traditional Systems Analysis and Design with Agile Methodologies,” University of Missouri-St. Louis, 2009. [Online]. Available: http://www.umsl.edu/~hugheyd/is6840/introduction.html. [Accessed: 10-Dec-2020].

[11] M. Alexander, “What is a Scrum master? A key role for project success,” CIO, 19-Jul-2019. [Online]. Available: https://www.cio.com/article/3223139/what-is-a-Scrum-master-a-key-role-for-project-success.html. [Accessed: 10-Dec-2020].

[12] L. Loewus, “Schools Take a Page From Silicon Valley With 'Scrum' Approach,” Education Week, 08-Dec-2020. [Online]. Available: https://www.edweek.org/leadership/schools-take-a-page-from-silicon-valley-with-Scrum-approach/2017/11. [Accessed: 10-Dec-2020].

[13] L. Benedicenti et al., "Applying Scrum to the Army - A Case Study," 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), Austin, TX, 2016, pp. 725-727.

[14] Hayes. William, Miller. Suzanne, Lapham. Mary Ann, Wrubel. Eileen, and Chick. Timothy, "Agile Metrics: Progress Monitoring of Agile Contractors ," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Note CMU/SEI-2013-TN-029, 2014. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=77747