Table of Contents

Software Engineering Documentation

Evolving Software
CJ Conti, Kevin Defendre, Marcos Rodriguez, Daniela Meneses
Software Engineering II, Hubert Johnson

Specification Analysis

Evolving Software

CJ Conti, Kevin Defendre, Marcos Rodriguez, Daniela Meneses

Software Engineering II, Hubert Johnson

Summary by CJ Conti

Name of the System

An aegis is defined as the protection, backing or support of a particular person or organization. Evolving Software will provide a software system that will support Spectrum Works' participants, the company itself, and communication between parties interested in building an inclusive workforce for young adults with autism. Because of this, the system is named Aegis.

What System Does

Aegis will have a wide variety of key functions to handle in order to aid Spectrum Works. First Aegis will let participants in Spectrum Works' programs access new employee training and workforce skill building courses that Spectrum Works created and Aegis will allow Spectrum Works to upload courses as well. Second, Aegis will allow participants to enter data for their goals and for their job coaches to enter surveys about their supervisees' progress. These surveys will be able to be used to evaluate participants and to determine whether to move them into higher level programs. Administrators will be able to set what program participants are in. The data that is entered will also be used to show how participants are learning from Spectrum Works' programs. Third, a convenient to access scheduling system will be implemented so that participants have fewer issues with their schedules. Also, participants, job coaches and other interested parties will be able to create accounts and users will only be able to access their own data, job coaches will be able to access the data of all participants and Administrators will be able to access all users' data. Before these accounts can access anything, administrators will verify the accounts. These users will be able to communicate on a forum where they will be able to create topics and message each other. Although Evolving Software hopes that all users will have positive behavior while using the forum, the team is aware that this is not always the case. Admins will have the ability to remove topics, remove posts and ban users if they find their behavior to be egregious by the standards of Spectrum Works. Spectrum Works' Partners will also be able to evaluate participants and Spectrum Works will be able to use this data to show improvement. Finally, Aegis will be exportable to other companies so that Spectrum Works can sell the software. Here is a list of the features for easier readability.

All Features

1. Add Courses
2. View All Courses
3. View Specific Course
4. Set Participant's Program
5. Create Account
6. Verify Account and set type of Account
7. Send Message

8. View Messages
9. Forums
10. Create Topic, View Topic, Edit Topic
11. Create Post, View Post, Edit Post
12. Ban user
13. Remove topic
14. Remove post
15. Surveys and viewing their results
16. Exportability

Who will use the system, what needs will your system satisfy and how it will help users

Evolving Software has determined the primary users of Aegis Software. Participants in Spectrum Works' Programs, Job Coaches, Employees of Spectrum Works and Workers at other companies visiting forums shall utilize Aegis for their benefit Spectrum Works will be able to access the data of all of its participants in one place. Job coaches will also easily be able to enter data about the people they are supervising too and see the data of all participants. Participants will be able to both see their progress and access resources that will help them. The forum will also allow people to communicate, which will benefit partner companies, participants and Spectrum Works itself. Lastly, Aegis will benefit Spectrum Works since the software will ideally show that participants are improving and it will be able sell the exportable system.

Most Important Features of the System

Aegis will a few key functions. Spectrum Works must be able to add its material to the system. Participants need to be capable of accessing this material. Accounts and forums shall be a critical feature since Spectrum Works heavily emphasized communication between companies and users. That communication can be controlled by Admins if a user posts something inappropriate. Aegis will also let the system's administrators designate the type of account a given user will have. Surveys are crucial since participants need to see that they are improving and Spectrum Works needs to display results. Participants need to be able to store events so that they can more easily keep track of their schedule. The system must be exportable so that other companies are able to help young adults with autism. Without exportability, Spectrum Works will not be able to maximize the systems benefits.

Performance Goals

Spectrum Works has provided a performance goals for the software to ultimately fulfill. Access to data needs to be quick and simple queries need to be underneath one second. In order to protect the confidentiality of participants, they should only be able to view their own data and outside companies should not be able to access their data without permission. Job coaches will be able to view and edit the data of all participants. Only Spectrum works employees will be able to access the data of all accounts. Finally, the system needs to be exportable and expandable as Spectrum Works grows and reaches more people and locations.

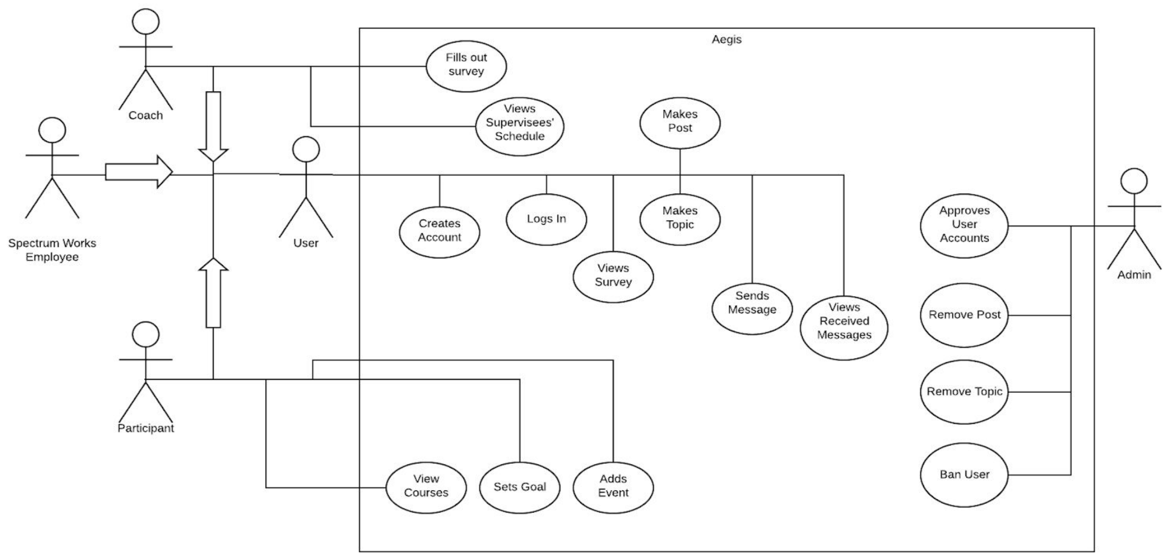
Physical Environment

Aegis will be used on the mobile devices and computers of participants and job coaches, so the final system must work effectively on phones. However, it will not be optimized for phones. In addition, other companies must be able to access the forums.

Details concerning System User Interactions by CJ Conti

Aegis will provide a variety of functions. The system will let participants access training and courses. Employees at Spectrum Works will be able to add courses to the system. Users will be able to enter their goals. Job coaches will be able to enter feedback. Participants will possibly be shifted into different programs based on their progress. Workers at Spectrum Works will be able to create a schedule made up of events for participants. A participant will be able to view their own schedule and add events to their own schedule. Partner Company Employees and job coaches will fill out surveys and Spectrum Works employees will be capable of accessing those surveys. Participants, workers at partner companies and Spectrum Works will be able create and manage account. Account users will be able to create topics, see messages in a topic, reply to topic, send messages, and view received messages.

UML Diagram



Components that will take inputs

The components that will take inputs are the course uploading system, the goal system, the event system, the account setup system, the topic system, the survey system, the message system, the post and topic removal system and the banning system.

How inputs will be handled by Aegis

Evolving Software has not only decided on the systems key functions, but it also determined how the system will handle various inputs.

Course-The course system will take the author of a course, the name of a course, a description of a course and text for the course.

Goal- The goal system will accept the name of a person/user who has a goal as text, the text describing that goal and the date the person wants to achieve the goal by.

Event-The event system will accept will accept the name of a person as text, events as text, a start time, an end time, a start date, an end date, a boolean for whether the event is repeated and text for how often the event is repeated. Events will be stored in a person’s schedule.

Survey-The survey will take a participant’s name as input, the date and answers to a group of questions.

Create Account-Creating an account will take a username, a password, an address, a name, an email address and a phone number as inputs for setting up an account. The address will include a state, a city, a street, an apartment, and a zip code. The name will include a first name, middle name and last name.

Log Into Account-Logging into an account will take a username and a password as input.

Topic- A topic will take a topic name as text, the name of the user who posted it, and the date and time it was posted.

Post- A post will take the text of the post, the name of the user who posted it, and the date and time it was posted.

Message-A message will take the username of the sender and the username of the receiver along with the text of the message itself as input.

Post Removal-Post removal will take the username of the user who made the post, the name of the topic it was in and the number of the post.

Topic Removal-Topic removal will take the username of the user who made the topic and the name of the topic

User Ban-User removal will take the username as input.

Handling both expected and unexpected inputs

Here is how the Aegis will handle both expected and unexpected inputs.

Course-If the input is expected, the goal component will log the course into a database or somewhere else the goal can be saved and accessed. If one of the inputs is left blank, the software will remind the person using to fill in all inputs. If the person attempting to create the course does not have a Spectrum Works account, they will be prevented from doing so. If another unexpected error occurs, the goal will not be entered and the user will be alerted of this.

Goal-If the input is expected, the goal component will log the goal into a database or somewhere else the goal can be saved and accessed. If one of the inputs is left blank, the software will remind the person using to fill in all inputs. If another unexpected error occurs, the goal will not be entered and the user will be alerted of this.

Event-If the input is expected, the event component will log the goal into a database or somewhere else the goal can be saved and accessed. If one of the inputs is left blank, the software will remind the person using to fill in all inputs. If another unexpected error occurs, the event will not be entered and the user will be alerted of this.

Survey-If the input is expected, the survey component will log the goal into a database or somewhere else the goal can be saved and accessed. If one of the inputs is left blank, the software will remind the person using to fill in all inputs. If another unexpected error occurs, the survey data will not be entered and the user will be alerted of this.

Create Account-If the input is expected the account component will create an account. If one of the inputs is left blank, the software will remind the person using to fill in all inputs. If an account has the same username that the person currently trying to create an account wants to use, the person will be prevented from creating an account with that username. If an account has too weak of a password, the person will be prevented from creating an account with that password.

Topic- If the input is expected the topic component will create a topic. If one of the inputs is left blank, the software will remind the person using to fill in all inputs. If a topic with the same name exists, the topic will be unable to be created unless the name is changed.

Post-If the input is expected the post component will create a post. If the post is empty, the user will be informed that they need to add text to their post.

Message-If the message is expected, the message component will send a message from the sender to a user with the message sender’s username listed as the sender. If the receiver does not exist, the user is informed that the receiver does not exist.

Survey- If the input is expected the survey will be submitted. If a question is empty, the user will be informed that they need to fill out that question.

Post Removal-If the given user has made post number inputnumber in the given topic, that post will be removed. If that combination does not exist, then the software will report that the given combination does not exist.

Topic Removal-If the given user has made a topic with this name, the topic will be removed. If that combination does not exist, then the software will report that the given combination does not exist.

User Ban-If a user with that username exists, then the software will ban that user. Otherwise, the software will report that a user with that username does not exist.

Input Ranges, Precision and Accuracy

The ranges for inputs have been determined as well.

Course-The length of a course name will be up to 64 characters.

Description-The length of the description will be up to 8192 characters

Author-The range of author names will be less than or equal to 64 characters that contains a letter

Username-The range of usernames will be less than or equal to 32 characters that contains a letter

Password-The range of passwords will be 8 to 32 characters.

Event- The range of the event will be any date in 2000s and the accuracy will be to the minute.

Goal- The range of a goal will be 2048 characters and the accuracy will be the exact input the user entered.

Topic-The range of a topic name will be 64 characters.

Post-The range of a post will be 2048 characters

Message- The range of the sender field will be 32 characters, the range of the receiver field will be 32 characters and the range of the message will be 2048 characters.

Survey-Answers to each question will be a maximum of 2048 characters.

Topic Removal-the range for the username will be 32 characters, the range for the topic will be 64 characters.

Post removal-the range for the username will be 32 characters, the range for the topic will be 64 characters and the range for the post number will be 4 characters.

User Ban-The range will be less than or equal to 32 characters

Sample transcripts show a dialog between a user and the system. This will help show how the users will interact with the system.

Sample Transcripts

Create Course

1. Tries to create a course while logged into a Spectrum Works Account

Aegis: “Enter course information.”

User:Enters course name, description and text

Aegis:”Your course”+coursename+” has been created.”

2. Tries to create a course while not logged into a Spectrum Works Account

Aegis: “You are not permitted to create a course unless you are a Spectrum Works employee.”

Create new account

Log In: Username and Password

1. Enters existing username and the corresponding password

Aegis: “Welcome to Aegis. Enter your username and password.”

User: Enters correct username and password.

Aegis: “Welcome”+ username”+”You are now logged in to Aegis.”

2. Enters existing but not matching password.

Aegis: “Welcome to Aegis. Enter your username and password.”

User: Enters an existing username, but the wrong password.

Aegis: “The username and password entered do not match. Enter your username and password again.”

3. Enters username that does not exist

Aegis: “Welcome to Aegis. Enter your username and password.”

User: Enters a username that does not exist, and a password.

Aegis: “A user with that username does not exist. Enter your username and password again”

Participant adding an event to their own schedule

Aegis:”Add an event to your schedule”

Participant: Enters “‘Meet with job coach’,’2:00 PM’,’September 2’,’Weekly’”

Aegis: Adds that event to the participant’s schedule.

Aegis:”Meet with job coach at 2:00 PM weekly starting on September 2 has been added to your schedule.”

Job Coach adding an event to a participant’s schedule

Job Coach: Enters Enters “‘Abram’,‘Meet with job coach’,’2:00 PM’,’September 2’,’Weekly’”

Aegis:”Meet with job coach at 2:00 PM weekly starting on September 2 has been added to Abram’s schedule.”

Goal

Aegis:”Add a goal”

Participant: Enters “Become better at communicating’,’September’,’8’,’2018’”

Aegis:”You added become better at communicating by September 8, 2018 to your goal list.

Topic

Aegis: “Enter the name of your topic and your initial post”

User: Sets the topic to be “Thank You Spectrum Works”

User: In the post section, writes, “I have learned so much at Spectrum Works and I appreciate the support of everyone there. I would especially like to thank my job coaches. Because of all of you, I have been offered a permanent position at Green Distribution. I really appreciate your help.”

User: Submits post.

Aegis: “Your topic Thank You Spectrum Works has been posted.”

Post

Aegis: “Enter your post in the topic Thank You Spectrum Works.”

User: Enters “I am so glad to hear that Spectrum Works has helped you this much. Stories like this motivate my work. I hope that you do well at Green Distribution.”

User: Submits post

Aegis: “Your post has been submitted.”

Message

Aegis:”Enter your message”

User: Enters receiver and message.

User: Sends message

Aegis: “Your message has been sent to”+receiver

Survey

Aegis:”Answer questions about this participant.”

User:Enters answers

Aegis:”Your survey has been completed.”

Topic Removal

Aegis:”Enter the name of the topic you would like to remove”

User:Enters topicname

Aegis:”Enter the name of the user who posted the topic”

User:Enters username

Aegis: Removes topic with name topicname by user username

Aegis: “Topic ”+topicname+” by ”+username+” has been removed”

Post Removal

User: Clicks on a button to delete a post. The button holds the username,topicname and postnumber.

Aegis: “Are you sure you would like to delete this post by “+username.

User:Selects Yes

Aegis:Removes post by user username in topic topicname with number postnumber

Aegis:”The post has been removed successfully”

User Ban

Aegis:”Enter the name of the user you would like to ban. Consider before banning them.”

User:Enters username

Aegis:”WARNING: Banning CANNOT BE UNDONE. Enter YES to ban the user.”

User:Enters Yes

Aegis: Bans user with username

Aegis:”User with “+username+” has been banned.”

Feasibility - by Daniela Meneses

Chance of project being completed in a semester

The project is likely to be completed in a semester

Major Classes

Participant - participants are the core of Aegis. The participants and their coaches should be able to easily access the system, communicate effectively with each other, and see results clearly through our analysis system. Participants include and are not limited to, students, coaches, administrators, and license holders.

Coach- A coach supervises one or more participants and plays a key role in their success. They offer feedback when it is needed and provide assistance to participants.

Course-Employees at Spectrum Works should be able to design courses and participants should be able to view courses.

Goal - Goals are a major cornerstone of Aegis. Goals should be set reasonably and once achieved, give the user and their coaches a sense of fulfillment that keeps them wanting to learn and using our system. If the user feels good about the progress they are making then it is very likely they will share the same sentiment about our software as a whole. Alongside the feedback section these functions will empower the user to keep going through the modules and accomplish more and more goals.

Feedback - The feedback section will be designed to accomplish two main goals. The first one is to give the user and their coaches accurate and easy to read feedback about how they are doing. This data should give grades about specific subjects and areas where they need improvement. The second goal of the feedback function is to give the user a good feeling about the system. A user who is doing well should be rewarded as such by goal accomplishments and a good feedback section. Like goals, this function will keep the users happy and coming back to the system. This also makes it more likely coaches and users will recommend the software to friends and colleagues which will expand growth. It is important to understand feedback should be designed to not be crushing either. If a user is not doing well in their program the software should give them tips and clearly show areas where they can improve and how they can improve. This lets the user know that they are not doing well now, but can improve by doing what the software tells it to.

Schedule - Scheduling is another major function of the system. Schedules should be set up so that the user is given specific due dates by the system for assignments and can accept custom input by the users themselves or their coaches. This all should be set up in an easy to read interface.

Account- Accounts will be set up by the users and coaches at the time of registration. During coach account set up the coach should have to provide all information deemed necessary by Spectrum Works and it will be saved into the system. The system will then provide the coach with an ID number so that the coach can distribute it to the their students so that they can link their accounts together. Student accounts will be set up in the same way however at the end of their registration they will have to go through a preliminary examination to determine their education level. This test does not need to be completed immediately but is required before the student can begin modules. After registration students can link their accounts with coaches by using the coaches ID number.

Topic - Users will be able to post topics in a general form that all other users, coaches, and system moderators will be able to see and respond to.

Message-Users can send messages to other users.

Surveys-Partner Companies can fill out surveys about Spectrum Works' participants.

Functionality

At its core Aegis is a learning platform designed for people with developmental issues, specifically autism. This software should enable the student to learn and grow while their coaches have access to monitor their growth. Students will be able to view courses. Coaches can have multiple students they are monitoring but the student should only have one coach. Students are able to explore different modules based on their education level, the system and their coaches will be able to set goals for them to complete which will appear on their schedule. Goal completion will be rewarding so that the student is empowered to keep going through the modules and further their education. Students should be able to access courses so they are able to review important work skills. Employees at Spectrum Works should be able to design those courses. Coaches and students will be able to view a feedback section which will be designed to be rewarding, but at the same time not be diminishing to the students moral. While the feedback system will report factual analysis about the students progress it should also offer a solution should a student not be doing well and highlight specific areas where they need to work on and improve. The software will also graduate a student to a higher learning level after they have shown that they are ready to move on. Coaches will be able to raise or lower a students education level based on their assessment of the student. Students are also able to message their coaches and other students that coach is also monitoring so that they can communicate and ask each other questions if they choose. Partner companies will be able to fill out surveys about employees, which will allow Spectrum Works to see if its participants are progressing.

Relationship Between Them

Participant-A participant is coached by a coach, sets goals, receives feedback from coach, owns an account, writes a schedule, posts topics, and sends messages.

Coach-Coaches participants, reviews goals of supervised participants, provides feedback to supervised participants, owns an account, posts topics and sends messages.

Course-Employees design courses and participants view courses

Goal- A goal is set by a participant and reviewed by a coach.

Feedback- A participant reviews feedback and a coach provides feedback.

Account- Both participants and coaches set up accounts.

Schedule-A participant creates a schedule.

Topic-Both users and coaches post topics.

Message-Both users and coaches send messages to either another user or another coach.

Surveys-Surveys are filled out by partner companies.

Functions that will not be included

- Optimized mobile access. Evolving Software will not develop a mobile app for the system.
- Online Coaching
- Online tutoring
- The educational content itself
- Non functional reporting system (eg. bad behavior)

Packages

Tier 1: Bronze Package

- Student and coach access to online modules

Tier 2: Silver Package

- Everything listed in Tier 1
- Goals and Feedback about how the student is doing

Tier 3: Gold Package

- Includes everything in Tier 1 and Tier 2

The Order that components are added

1. The first thing that will need to be added is the basic user accounts, coach account, and admin accounts. This should include all necessary contact information and information required for the education modules to function, eg. education level, gradebooks, and other related variables.
2. Next the education modules should be added and the users should be able to access the appropriate lessons based on their selected level.
3. Following that grade and performance analysis should be added as well as the functionality that allows a student to advance to the next level of education
4. Finally all quality of life functions should be added individually. This includes calendar, messaging notifications, and reporting system.

Summary- By Kevin Defendre

Aegis is a system that let users access educational content that is used by the administrator of that content. This software will include several things in the system such as online tutoring, online coaching, mobile access, the educational content itself, and non functioning report system. The system can handle unexpected responses that the user input by alerting the user that what they have put to be wrong. The functionality of the program is that the program is designed for people that has autism. The coach can have multiple students but the students can only have one coach.

Optional Section- By Marcos Rodriguez

System runtime should ideally be under one second when data is being accessed. Anything longer than one second would mean that Aegis is not functioning at its intended speed.

Aegis will not use up a large amount of memory when downloaded.

It is extremely vital for the data of users to be well-secured in order to prevent third-parties from using data without permission.

The software should be compatible for desktop/laptop computers. Software updates would be delivered to fix any performance issues and improve overall user experience. The installation agreement will be for Aegis to receive these updates to improve its stability.

Evolving Software is a team of beginning software developers, meaning that Aegis is the first major project that the team will produce.

The project is expected to be delivered in its completed form to Spectrum Works by May 2019.

Computer time for developing Aegis will take several hours.

Aegis should be fully functional for everyone in Spectrum Works to use, including employees and participants in Spectrum Works' programs.

Software Engineering Test Plan

Evolving Software
CJ Conti, Kevin Defendre, Marcos Rodriguez, Daniela Meneses
Software Engineering II, Hubert Johnson

Introduction and Modules

Evolving Software will be designing software for Spectrum Works. Evolving Software seeks to create a transferable system for Spectrum Works that will help monitor and evaluate participants in Spectrum Works' programs and provide a way for connected people and companies to communicate. The final product will make Spectrum Works' pre-existing training program available on the internet and store data about participants that can be used for participants themselves to monitor their progress and for Spectrum Works to use to display that they helped participants learn and improve on essential work skills. The team has designed a plan for testing the components of the final product so that its quality can be guaranteed.

Evolving Software currently has ten main testing modules in mind. These are:

1. A training system that will help with new employee training for participants and will provide access to Spectrum Works' lessons.
2. A survey system that will allow partner company employees to offer feedback on employees.
3. A goal entry system that will include goal progress and that will allow goals to be considered successful or failed, or just to be removed.
4. A database that will provide access to scheduling for participants. Schedules will include assignment due dates, appointments, and dates for goal completion.
5. A productivity rate tracking and an analytics system that will help organize productivity data, ideally to show how participants are improving. If participants

are improving they should eventually be placed at a higher education level based on performance.

6. An account system that would allow participants and people outside Spectrum Works to create accounts and would secure individuals’ data by only permitting to access their own data and the data of people they are supervising.
7. A forum that would allow for communication and for users to create and reply to various topics and message each other directly.
8. The ability for admins to remove topics and posts that violate rules
9. The ability for admins to ban users that repeatedly break rules.
10. A way of ensuring that the system would easily be transferable and exportable so that other companies can use this software.

Evolving Software plans to start the first testing phase by working with all components isolated from each other except for user accounts and forums due to how closely connected they are. During the second testing phase, Evolving Software will combine the training system with the survey system and accounts so that users can enter data into their accounts and view educational resources. In that same phase, the scheduling database will be combined with the productivity tracking system and user accounts, so that users can have productivity associated with them, which will hopefully show signs that participants in the program are improving. The user accounts and forums will still be kept separate from other components besides the exporting module during this phase. All of the separate groups of modules for the second testing phase will implement the exportability module. During the third testing phase, the user account system will be combined with training courses, productivity and the survey system so that the system based around participants' work can be designed. The secondary group of modules will focus on having the scheduling and account data set up while user accounts can be made and users can access forums. Finally, all modules will be combined for the final testing phase. the The team's module-to-test techniques are shown below and have been designed so that features may be combined easily.

Testing Phase 1

(1),(2),(3),(4),(5),(6,7,8,9),(10)

Testing Phase 2

(1,2,3,6,8),(4,5,6,8),(6,7,8,9,10)

Testing Phase 3

(1,2,3,5,6,8),(4,5,6,7,8,9,10)

Final Testing Phase

(1,2,3,4,5,6,7,8,9,10)

Summary of the module to test technique

First Testing Technique: Inspection

1. Training System

The system for creating a course is set up.

Courses are listed as prescribed and are easy to locate for a potential user.

2. Survey System

All of the inputs are shown on the screen along with a way of submitting the inputs.

3. Goal Entry System

4. Schedule Database

A daily calendar with tasks is displayed.

5. Productivity rate and Analytics System

Graphs and charts are created based on sample data. Data cannot be inputted in this stage.

6. Account System

An area to set up an account is created and a user can enter their username, password and personal info, but nothing will occur.

7. Forum

Various subforums, topics are listed and accessible, and the inputs for making a new post are set up, but new posts cannot yet be created.

8. Removing Posts and Topics

The button for removing posts and topics can be accessed and the function's output can be printed out.

9. Banning Users

The button for banning a user can be accessed.

10. Exportability

The system appears to be exportable.

Second Testing Technique: Demonstration

1. Training System

An admin can fill out a form for creating the course and then submit it.

A user can click on the link to a course and enter the course.

2. Survey System

A user can enter inputs and the inputs are correctly stored in a database. The inputs can be accessed later.

3. Goal Entry System

A user or their supervisor can enter goals and the goals are correctly stored in a database. The goals can be marked as completed or not completed.

4. Schedule Database

A daily calendar with tasks is displayed and tasks can be added, removed, or completed.

5. Productivity rate and Analytics System

Graphs and charts are created based on real data. The view of the data can be changed by altering settings.

6. Account System

An account can be set up and the account information can be stored in a database. In addition, the user can log into their account.

7. Forum

New topics and posts can be created by users. Users are able to reply to other user's posts.

8. Removing Posts and Topics

A post can be removed and a topic can be removed.

9. Banning Users

A user can successfully be banned.

10. Exportability

The system can be exported to a different computer and still work correctly.

Third Testing Technique: Testing

1. Training System

A course named testing can be created, that course can be selected and its text can be viewed. A user can select a course name testing and view its content.

2. Survey System

A user with username testing can enter a 500 character answer to each question and those answers can be accessed.

3. Goal Entry System

A user named testuser can add a goal named “Finish testing the goal entry system” and set its date to be 5/4/2018. They can then check off that goal as being completed.

4. Schedule Database

A user named testuser can add a task named “Check if the schedule system works” and set its date to be 5/4/2018 and have it repeat weekly. They can then modify the task to repeat daily. Finally, they can delete the task.

5. Productivity rate and Analytics System

Graphs and charts are created based off of data for a few people.

6. Account System

An account with the username testuser can be created with password password. Their name, address, age and account information can be stored in a database. Testuser is able to log into their account.

7. Forum

A topic named testtopic by testuser can be created with post “I am testing to see if the posting system works”. Another user named testuser2 is able to reply with “Reading you loud and clear.”

8. Removing Posts and Topics

A post saying “Helping people is a waste of time” can be removed and a topic titled “Freaks should be ignored” can be removed (While these are very tame, the team does not wish to consider more explicit and vulgar examples).

9. Banning Users

A user named ihatedifferentpeople can be banned.

10. Exportability

The system can work correctly on a PC at Montclair State.

Final Testing Technique: Analysis

Tests that are designed to stretch the system and determine where it will break are still being determined.

1. Training System

A user can try to go to page -1 for the courses. A user can try to go to page 100000 for the courses. A user can try to access a course that doesn’t exist and fail to do so.

An unauthorized user can try to access this system and be unable to do so.

2. Survey System

An unauthorized user can try to access this system and be unable to do so.

3. Goal Entry System

A user can try entering a goal with an end time before the start time and the system prevents the user from doing so.

An unauthorized user can try to access this system and be unable to do so.

4. Schedule Database

A user can try entering an event that repeats -1 times and the system prevents the user from doing so.

An unauthorized user can try to access this system and be unable to do so.

5. Productivity rate and Analytics System

Test data for 1000 people can be viewed, graphed and analyzed in a reasonable amount of time.

An unauthorized user can try to access this system and be unable to do so.

6. Account System

A user can attempt to use the same name as another user and the system will notify the user that the username is taken.

An unauthorized user can try to access this system and be unable to do so.

7. Forum

A topic named testtopic by testuser can be created with post “I am testing to see if the posting system works”. Another user named testuser2 is able to reply with “Reading you loud and clear.” Another topic named testtopic by testuser3 can be created with post “I am testing to see if the posting system works”. Another user named testuser is able to reply with “Reading you loud and clear.”

8. Removing Posts and Topics

An unauthorized user can try to access this system and be unable to do so.

9. Banning Users

25 users can be banned in a reasonable amount of time.

10. Exportability

The system can work correctly on an older PC at Montclair State and on a phone.

Summary of the monitoring, reporting and correcting procedures

Monitoring will primarily consist of using sample inputs and checking for expected outputs after major updates are made to the software. Reporting will consist of messages being sent to the team when a substantial unexpected behavior or error appears. Minor errors that do not require changing a large amount of the software will be corrected by the team member that detects them. Correcting will be done by using a version of the software on an individual team member's computer and followed by that member sharing the improved results before implementing it to the shared version of the software.

Proposed dates for submission of individual test reports

The first set of tests will be done by March 7, which will provide the team with enough time to design the system. The second set of tests will be done by March 21 since combining the modules will require a solid amount of time. The third set of tests will be done by March 28, since only a few modules need to implemented to different groups. The final set of tests will be done on April 11 since ensuring the entire software works as intended is critical so extra time needs to be devoted to it.

Defense of the Integration Plan

This testing plan is quite reliable since the majority of the components are isolated first and are easier to test by themselves. Bugs and other issues can be eliminated while the amount of code is smaller, and once multiple components are confirmed to be working as pre-defined, they can be combined. Another benefit of testing each component by itself is that determining the optimal design and platform for the system can occur early in the programming of the system and changes can be made without a great loss of time.

The second phase of testing focuses on combining only similar portions of the program so that way testing still remains relatively simple. Exportability and accounts are implemented to all subsystems so that implementing them into the final product will prove simple. The parts are also divided so that similar structures are combined first.

Due to this being the team’s first project of this scale, the third testing phase will combine a small number of features so that fewer issues arise. Combining data entry, data analysis, and forums along with scheduling would prove quite difficult, which is why data entry and data analysis are being combined first. From there, adding the forum and scheduling should be much easier. Overall, Evolving Software believes that this integration plan would be quite efficient.

Phase Testing - Daniela Meneses

During the development of this system we will implement multiple testing techniques throughout 4 different phases to ensure the product is working as intended and minimal patching is needed after product delivery. First we will start with functional testing, this is where we take each function of the system and test it individually to make sure each part of the system works before testing the entire system. After we ensure each part of the system is functional on its own we will transition into performance testing. This includes everything the functional test is not, for example we will do stress testing to make sure the system can handle large quantities of data, regression testing to make sure the software is an improvement on the system we are replacing, etc, etc. This will ensure that the system as a whole is still working when exposed to different scenarios. Once this is complete we will move into acceptance testing which is when the client comes and either accepts or denies the product as presented. If the product is accepted we then move into installation testing where the product is installed at the clients site.

Phase 1: Functional Testing

The functions of this system that we will be testing include:

- New User Creation
- Education Modules
- Scheduling
- Goal creation
- Education and goal progression
- Tracking of education and goals
- Analysis of performance
- Advancement in placement

New User Creation:

What are we testing?

1. User should be able to create an account using an email and password they would like to associate with their account.
2. User should be asked to complete account registration by verifying email, entering full name, DOB, address, and phone number.
3. Information should be kept private from everyone excluding system admins.

How will we test?

1. Create multiple test user accounts and one single admin account.
2. Make sure all required data is able to be entered and is saved properly.
3. Ensure a basic user is unable to see private information of other users.

4. System admin should be able to see all users information and modify that information if need be.

Education Modules:

What are we testing?

1. Once a user is created they will need to be placed into an education level either by a system admin or a placement test.
2. Based on that placement level the user will have access to the appropriate education modules to help them improve their skills.
3. Users should be able to flawlessly navigate through the education modules, and module questions should be able to be answered appropriately.
4. Users should be able to see what answers they got wrong, why they got them wrong, and the solution to the correct answer.
5. At the end of a module a grade is presented and is reported to a tracking system.

How will we test?

1. Create multiple users at varying placement levels.
2. Ensure they are seeing the education modules appropriate to their education level.
3. Make sure they are able to navigate through the module and questions are able to be answered appropriately .
4. Questions will be answered wrong intentionally to ensure the proper dialog appears and the student is shown the proper solution.
5. The whole module is completed and should be reported to the tracking system.

Scheduling

What are we testing?

1. Students will have access to a calendar and should see all due assignments from the modules they are currently working on.
2. Users should be able to make appointments as well as invited and accept invitations from other users. These appointments should also appear on the schedule separate from the other assignments.
3. Appointments and events should be able to be changed after creation and if that should occur all invited and attending users should be notified.

How will we test it?

1. Create multiple test user accounts and place them at different skill levels.
2. Make sure assignments are able to be assigned and appear on the schedule.
3. Invite multiple users to different appointments and ensure they appear on the schedule.
4. Once appointments are on the schedule multiple appointments will be changes and all users attending should get a notification stating that the appointment has been changed.

Goal Creation:

What are we testing?

1. Goals should be automatically created by the system based on the module and skill level of the user.
2. Goals can also be manually created by a system admin for a specific or group of users.
3. Goals can have a complete by date but they do not need to.

4. Goals should also appear on the calendar if they should have a completion date.

How will we test it?

1. Using test user accounts with varying skill levels they will traverse through the modules and and goals should appear in their profile.
2. System admin accounts will add goals with and without competition dates. Those with completion dates should appear on the schedule and the ones without should not.

Education and goal progression

What are we testing?

1. Goals are being created according to the plan set out in the education module.
2. Students are able to complete those goals as intended
3. Coaches and students can create goals and set completion conditions to the goals

How will we test it?

1. Check the multiple user accounts we created for testing and see if they have goals set up for them and that they are able to complete the goals when going through the education modules
2. Coach accounts should set goals up for the users as well and the user accounts should attempt to complete them by satisfying the completion condition.

Tracking of education and goals

What are we testing?

1. How much the student has completed and how well they have done with the education
2. Grades should be reported to the analysis system
3. Goals that have been completed and their completion condition

How will we test it?

1. After a student test account completes a module it should be reported and displayed on the analysis section of the user page
2. Number of goals and their completion condition should be displayed as well on the user test account

Analysis of performance

What are we testing?

1. Analysis section should take in grades from the education modules and be notified when goals are complete and their conditions
2. A GPA should be calculated and GPA's for specific sections should be displayed as well.
3. This informations should be reported back to the education system so that adjustments can be made and students can work on areas where they need improvements.

How will we test it?

1. New user test accounts should be created and have them run through the education modules but purposely fail in specific subjects but excel in others
2. The analysis section should show that the user is doing well in the subjects which were purposely excelled in and show that the student needs to show improvement in the subjects where we purposely did poorly.
3. The test accounts should then check the education section and see if the education modules are adjusted to focus on the sections where the student did poorly

Advancement in placement

What are we testing?

1. Once the student has done sufficiently in the fields required for advancement the system should upgrade, or “level up”, the student and start them on a new track of education modules.
2. The student and coaches should be notified that the student has advanced

How will we test it?

1. Take student test account and have them complete the requirements for advancement and see if the system knows to advance that student
2. Ensure the student and coach receive notification of the advancement.

Phase 2 - Performance Testing

Stress Testing: This system should be able to handle multiple users at the same time. Because all data will be stored in a database that database needs to be strong enough to handle multiple queries at the same time and over an extended period of time. In order to conduct this test we need to attempt to break this system and find the breaking point. Based on the number of units sold and user accounts created we should have a general idea of how many users we can expect to be using our system at once. Using that data we have a benchmark of where we want our system’s breaking point to be at. We should be overloading the database and improving the database until that breaking point is where we would like it to be.

Volume Testing - Unlike stress testing where we are finding out how many users can access the data at one time we are finding how much data our system can hold. Using the same method that we used for stress testing we should be able to find out the maximum number of users we would have to accommodate based on created accounts and units sold. Using that information we can load our database with data and keep improving the database until it is able to handle the amount of data we need it to.

Configuration Testing - Multiple configurations will need to be tested in this system. User accounts with and without coaches should be tested for errors. User accounts of both types should be tested through all education levels to ensure complete functionality. Since coaches should always have a student only one type of coach account should need testing. Admin accounts should also be tested to ensure adjustments can be made after the system is implemented.

Compatibility Testing - Since this system uses its own databases and its own infrastructure minimal compatibility testing will be needed.

Regression Testing - Developers should acquire or be able to analyze the former system that was in use and make sure the new system as a whole is better than the previous system and meets the complaints that were made about the old system.

Security Testing - Testers should make every effort to get into secure user accounts and databases, without admin privileges, by any means possible to ensure no holes in the security of our system.

Timing Testing - This system should be able to operate with minimal lag and delay in the response of the system. Timing testing is also similar to stress and volume testing where we are ensuring that the system is able to hand the requests of the users and respond appropriately especially during peak times such as school hours.

Environmental Testing - Because this system is going to be used mainly from home and in school settings this system should be tested on multiple types of home computers and operating systems to ensure it can be used flawlessly from home. Support and admin systems should be tested at Spectrum Works locations so that they are able to use the system from their offices and make changes as needed with minimal interruption

Quality Testing - The testers should test and make sure everything is up to the quality that is expected upon delivery to the client. This will again be checked during the acceptance testing by the clients themselves but the initial test should be done by the testers.

Recovery Testing - Should anything happen to the databases and / or the system as a whole there should be a crisis plan in place that prevents a total loss of data which could prove fatal to the system. This crisis plan should be enacted so that plan can be improved and ready to go once the system has been enacted.

Maintenance testing - We should test how we are going to perform general maintenance after the acceptance of this system. This system should be tested prior to delivery of the system so that there are no flaws that could cause the system to break during testing.

Phase 3: Acceptance Testing

Acceptance testing is done by the client prior to the final delivery of the product. They should follow our test plan and run through each system, to the best of their ability, as we have to double check for any flaws or functions that are not up to the standards they expected. A document should be made and submitted to the developers for final corrections. After the changes have been made the client should run a second acceptance test making sure everything is to their liking. If all is well then we are ready for the final delivery of the system.

Phase 4: Installation Testing

During the final delivery of the system a installation test should be performed at the site of delivery to ensure the process goes well and then client is left in a good position when accepting the system.

Details of monitoring, reporting and testing procedures

Monitoring will be done by having weekly meetings and by using a document for keeping track of when tests are done. All of the tests will be inserted into the document and sorted by who is responsible for them before testing begins so that each team member knows what tests there are and which ones they will do. Once a team member finishes a test, they will check it off so that other members can see that it has been completed. Once all of the tests are checked off, that round of testing has been completed.

Team members will report the test results by writing them into a document for the test results. The results will be underneath the name of the test. The document will be looked over regularly so that way the team will stay up to date on the status of tests. If an urgent issue arises during testing that needs to be handled as soon as possible, each team member will be notified by the team member responsible for the section where the issue occurred. This form of documentation will be used since it is both simple to implement and allows for fast response to major issues.

Testing will be done by using the system itself and verifying if the given output matches with the expected output. The expected output will be determined before the test begins in order to aid verification. If it is done afterwards or during the testing, the correct result could easily be missed by a team member, which is why the correct result will be determined before a test commences. A test will be considered completed once the output matches and other parts of the tested functions, such as data being saved, work correctly. When modules are combined for testing, testing will involve checking if connected functions all work together. For example, a test may involve checking if an account can be created and then that a team member can successfully log into the account they just created. This testing methodology is effective since it is quite simple and relies on checking if results match with what is expected.

Details of individual team member assignments

Determine Individual Team Member Assignments

The exact assignments will be determined at a later date, but each team member will handle at least one set of tests and handle implementing at least one part of the system.

Appendix for Negotiated Changes

Design Document

Evolving Software
CJ Conti, Kevin Defendre, Marcos Rodriguez, Daniela Meneses
Software Engineering II, Hubert Johnson

Courses

Accesscourse

Abstract

This function is used to allow a user to access a certain course based on the course name that is used as input. Participants benefit from these courses and need to access them. Only participants and admins can view courses.

Design

This function is called after a user selects a course from the function ViewAllCourses() and then uses the name of the selected course to acquire the content corresponding to that name. The content is then displayed on screen. A user is only able to view the content if they are logged in as a participant, a job coach or an admin. If they are not, they are redirected to the login function.

Error handling

There are no inputs outside of the coursename, so there will be a minimum number of errors to handle. The software prevents unpermitted access by requiring a user to be logged in as a participant, a job coach, or an admin.

Overall importance

This module is important since viewing content was highly emphasized by Spectrum Works.

Pseudocode

```
Function accesscourse(coursename)
If logged in as a participant or logged in as admin or logged in as a coach
        coursecontent=search for course named coursename
        Display coursecontent on the screen
Else
        print("You need to be logged in as a participant, a coach, or an admin in order to view
this")
        Request login as a participant
        Goto login function
```

ViewAllCourses

Abstract

This module is used to view all courses in a list. This moudle will be used whenever a participant wants to view course content.

Design

This list will work like a book with multiple pages and the user will be to access links to 10 courses at a time on one "page". This will make viewing the courses more manageable than viewing all of the courses at once. When the user clicks on a link to a course the function accesscourse will be triggered.

A user is only able to view the content if they are logged in as a participant, a job coach or an admin. If they are not, they are redirected to the login function.

Error handling
The software prevents errors by checking if the pageon is positive and there are still courses left.

Overall importance
This module is quite important since it allows users to view and access courses. Without this, they cannot access the courses they signed up for and will be unable to learn from Spectrum Works' content.

Pseudocode
```
Function viewallcourses(pageon)
courseon=pageon*10
print("You must be logged in as a participant, job coach or admin to access courses, but all
users can see what courses Spectrum Works has to offer.")
if(pageon>0 and pageon*10<=coursecount)
    while(courseon<=10*pageon+9)
        coursename=get course name of course number courseon
        coursedescription=get course description of course number courseon
        print(coursename)
        print(coursedescription)
        Display button that when clicked triggers accesscourse(courseon)
```

Add Course
Abstract
This function is used by admins to add courses.

Design
This module will allow the user to access courses, if logged in as either a participant, job coach, or as an admin. They will be allowed to view and add courses. If the user is not logged in, then they will only be able to view courses.

Error handling
The software handles errors by only letting admins access this function.

Overall importance
The importance of this module is that it is needed for users to add the courses that Spectrum Works offer.

Pseudocode

```
Function addcourse()
If logged in as admin
    print("Enter the course name")
    coursename=getNextString()
    print("upload the course file")
    coursefile=getNextFile()
    Creates course with coursname and content coursefile
    print("Course "+coursename+" has been created.")
Else
    print("You need to be logged in as an admin in order to add a course)
    Goto login function
```

Goal Entry

Entergoal

Abstract

This module is used to allow participants to enter goals, which will motivate them to try and work toward those goals. Being able to look back at those goals is quite motivational, so saving them all in one place will help them. The goals users enter into this module are saved for the future so participants can view them.

Design

This module will check if a user is logged in as a participant. Once the user is authorized,the software will then take the text for the goal and the date for the goal as input.

Error handling

In order to prevent errors with the date, the module will check whether the text entered for goaldate is a date. The module also checks that all of the inputs are entered so inputs are not left empty. This module also requires users to be logged in as a participant or an admin in order to enter a goal.

Overall importance

Keeping track of goals and achieving those goals is very crucial to Spectrum Works' users.

Pseudocode

```
Function entergoal()
If logged in as a participant or logged in as admin
    user=GetCurrentUser()
    goaltext=getNextString()
    goaldate=getNextDate()
    if(goaltext is filled and goaldate is filled)
```

```
                Store new goal set by user with goaltext for text and goaldate for the goal date
        else
                if(goaltext is not filled)
                        print("Fill in goaltext")
                if(goaldate is not filled)
                        print("Fill in goaldate")
Else
        print("You need to be logged in as a participant in order to view this")
        Goto login function
```

ViewGoals

Abstract

This module is used to allow participants to view goals they previously entered using the entergoal module. Being able to check back on goals provides motivation and also provides something to work for each day. Participants have something specific to work towards since they can see where they wanted to improve. Job coaches also benefit from this module since they are able to see what the participants they are supervising want to achieve. They can then provide feedback and advice that will push participants towards reaching the goals they entered. This module is designed to aid participants.

Design

The user will be able view the goals that hey have to accomplish throughout the training course and will increase base on their completion. The admin will also have the right to view the accomplishments that the user has. In order for the admin to view this, they would have to login into the app.

Error handling

In order to prevent random people from entering into the program, a login system will be provided in order to protect the data in the program. If the admin fails to properly put their password then the program will tell them that they need to login in order for them to see the user's progress.

Overall importance

The overall importance of this is for the admin to keep track of user and overlook their progress to see the user's accomplishments.

Pseudocode

```
Function viewgoals()
If logged in as a job coach or logged in as admin
        print("Print the name of the user whose goals you wish to view")
        selecteduser=getNextString()
        For each goal set by user selecteduser
                print(goaltext)
                print(goaldate)
                print(“\n”) //Helps for legibility
Else If logged in as a participant or logged in as admin
        For each goal set by user currentuser
                print(goaltext)
                print(goaldate)
                print(“\n”) //Helps for legibility
Else
        print(“You need to be logged in as a job coach or a participant in order to view this”)
        Goto login function
```

Survey

FillOutSurvey

Abstract

Spectrum Works wishes to gain more partner companies and in order to do so, they need to show that participants in their programs are improving. By having employees at partner companies enter surveys, they can provide feedback on how the participants are doing. If later surveys show better ratings than earlier ones, this can be used to display that Spectrum Works is helping its participants improve. Job coaches can also fill out these surveys in order to provide feedback on how participants are doing.

Design

The survey is designed so that the user answers a series of questions. After all of the questions are answered, the user then is able to submit the survey. The system then saves the answers to the questions, the user who filled out the survey and the date the survey was submitted.

Sub-modules

Error handling

This module handles errors by requiring users to enter an answer for every question and to be logged in as a job coach or an admin.

Overall importance

This module is important since it allows coaches and partner company employees to assess how the participants are doing. This saved feedback can be used to show improvement.

Pseudocode

```
Function filloutsurvey()
If logged in as a job coach or logged in as admin
        Aegis shows a series of questions for job coaches
        provides area for inputs to be entered.
        User attempts to submit survey
        if(all questions are filled in)
                Save username, their answers to all questions in the survey and the date the
                survey was entered
```

```
    Else
        For each unanswered question n
            print(“You need to answer question”+n)
If logged in as a partner company employee or logged in as admin
    Aegis shows a series of questions for partner company employee
    provides area for inputs to be entered
    User attempts to submit survey
    if(all questions are filled in)
        Save username, their answers to all questions in the survey and the date the
        survey was entered
    Else
        For each unanswered question n
            print(“You need to answer question”+n)
Else
    print(“You need to be logged in as a job coach, a partner company employee or an
    admin in order to view this”)
    Goto login function
```

ViewSurveys

Abstract

In order to protect the privacy of participants, users that are partner company employees or job coaches will be to access the name of participants in surveys. Other users will only be able to access the answers to surveys and will see a number corresponding to a participant rather than their name.

Design

Here, the user will be be checked if they are a job coach, partner company employee, admin, or just a regular user. Regular users will only be able to see numbers corresponding to survey participants, while all other users will have access to the names of participants.

Error handling

If the person logging in is not a user, then a message will be printed out stating that they have to be a registered user in order to access the surveys.

Overall importance

Viewing surveys is important since this will allow companies to see

Pseudocode

```
Function viewsurveys()
If logged in as job coach or logged in as partner company employee or logged in as admin
    For each survey
        print(“Name: ”+name_of_person);
        For each question
```

```
                print(question)
                print(answer)
Else if logged in as user
For each survey
          print(“Participant Number: ”+number corresponding to participant);
          For each question
                print(question)
                print(answer)
Else
     print(“You need to be logged in as a job coach, a partner company employee or an
     admin in order to view this”)
     Goto login function
```

Scheduling

AddEvent

Abstract

Users will benefit from adding events since they will be able to keep track of what they need to do. This module outlines a form that users can fill out in order to add an event. It is designed to be relatively simple so that way users are able to benefit from a clear, versatile system. This module is useful since it allows users to set the name of an event, what time it occurs, the date it occurs, how often it repeats and how many times it repeats.

Design

The module is designed to allow the user, if logged in as a job coach, participant, or admin, to create and schedule events. The user will be allowed to create the name of the event as well as the time and date of said event.

Error handling

This handles errors by preventing events from being added too many times and makes sure that all values are set.

Overall importance

This module is important for adding events that the users will keep track of.

Pseudocode

```
Function addevent()
If logged in as job coach or logged in as participant or logged in as admin
     eventname=getNextString()
     eventtime=getNextTime()
     eventdate=getNextDate()
     repetitionrate=getNextString()
     repetitioncount=getNextInt()
```

```
        If ((eventname is set and eventtime is set and eventdate is set) and (repetitionrate is
        onetimeonly, daily, weekly, or monthly) and repetitioncount<=365)
                Add event with eventname at eventtime for currentuser on eventdate and repeat
        that event repetitioncount times for the dates corresponding to the repetitionrate
        Else if repititioncount>365
                print("The event has been repeated too many times.")
        Else If repetition rate is not onetimeonly, not daily, not weekly, or not monthly
                print("Set the repetition rate to be one time only, daily, weekly or monthly")
        Else
                print("Make sure the name time, and repetition rate are all set")
Else
        print("You need to be logged in as a job coach, a partner company employee or an
        admin in order to view this")
        Goto login function
```

EditEvent

Abstract

Events can change over time, so it is important that the system allow events to be edited. This module is useful since it allows users to change the name of an event, what time it occurs, the date it occurs, how often it repeats and how many times it repeats.

Design

Edit event has the name of the event a user wishes to edit as the input. The user then sets the name,time,date, repetition rate, and repetition count again for a new event and the old event is modified.

Error handling

The module catches when inputs are not filled in or are filled in incorrectly.

Overall importance

This module is quite important since users should be able to change their events. If they make typos, they should be able to fix them.

Pseudocode

```
Function editevent(currenteventname)
If logged in as job coach or logged in as participant or logged in as admin
        currentuser
        neweventname=getNextString()
        eventtime=getNextTime()
        eventdate=getNextDate()
        repetitionrate=getNextString()
        repetitioncount=getNextInt()
        If ((eventname is set and eventtime is set and eventdate is set) and (repetitionrate is
        onetimeonly, daily, weekly, or monthly) and repetitioncount<=365)
                Change event with currenteventname to be an event with neweventname at
        eventtime for currentuser on eventdate and repeat that event repetitioncount times for
        the dates corresponding to the repetitionrate
        Else if repititioncount>365
                print("The event has been repeated too many times.")
        Else if repititioncount<1
                print("The event has been repeated too few times.")
        Else If repetition rate is not onetimeonly, not daily, not weekly, or not monthly
                print("Set the repetition rate to be one time only, daily, weekly or monthly")
        Else
                print("Make sure the name time, and repetition rate are all set")
Else
        print("You need to be logged in as a job coach or a participant or an admin in order to
        view this")
        Goto login function
```

RemoveEvent

Abstract

Users may want to get rid of their events due to them no longer being valid or plans changing, so this function provides a way for users to get rid of their events.

Design

This module checks if a user is logged in and takes the event's date and name as input. The module then deletes an event by the currently logged in user named eventname on eventdate at eventtime.

Error handling

This module handles errors by checking whether the user is logged in as a job coach, participant or admin.

Overall importance

This module is crucial since it allows users to adapt to changing plans by getting rid of events that are no longer valid.

Pseudocode

```
Function removeevent(eventname,eventdate,eventtime)
If logged in as job coach or participant or logged in as admin
        current=getUser()
        Delete event with eventname on eventdate at eventtime by currentuser
Else
```

```
        print("You need to be logged in as a job coach, a participant or an admin in order to view
        this")
        Goto login function

Function viewevent(eventname,eventdate)
If logged in as job coach or participant or logged in as admin
        currentevent=get event by current user with name eventname at date eventdate
        print(currentevent's name)
        print(currentevent's date and time)
Else
        print("You need to be logged in as a job coach or a participant in order to view this")
        Goto login function
```

Create Account

CreateAccount

Abstract

This module is used to allow users to create an account. Users first enter their personal information followed by their username and password.

Design

Users first enter their personal information followed by their username and password. The module then checks if the inputs are all filled and then checks if the password is strong enough. If the username has not been taken, the module then creates an account with the entered username and password.

Error handling

The error that would be implemented in the created account subset is that the user would be told that the username and password do not match the required amount in order to have a account.

Overall importance

Accounts allow users to verify who they are and therefore access certain other features of the software. Without accounts, participants are unable to identify themselves.

Pseudocode

```
Function createaccount()
firstname=getNextString()
middlename=getNextString()
lastname=getNextString()
street=getNextString()
city=getNextString()
state=getNextString()
```

```
username=getNextString()
password=getNextString()
If all inputs are set
    strongenough=checkpasswordstrength(password)
    If strongenough is true
        If username is not taken
            Create account with this information
            print("Your account has been created")
        Else
            print("A user already has that username")
    Else
    print("Change your password to be stronger")
Else
    print("Enter all inputs")

Function checkpasswordstrength(password)
    If the password is shorter than 8 characters
    Return false
    If the password has no numbers or special characters
    Return false
```

Account Approval

VerifyAccount
Abstract
This module is used to verify new accounts that users create. Admins are given access to approve accounts and select the type of user an account will be for.

Design
This module is designed by having an admin select an account and set its role as user, participant or job coach.

Error handling
This module handles errors by preventing a user from selecting an option that does not exist and requires the user to be logged in as an admin.

Overall importance
This module is essential since it allows users who are not verified to begin accessing other functions.

Pseudocode

```
Function verifyaccount()
if(logged in as admin)
```

```
    print(“Select an account to approve”)
    accountselected=getInput()
    print(“Select what type of user you want the selected account to be”)
    optionselected=getInput()
    //Only options are user,participant and job coach
    Set account accountselected to be a optionselected type account
Else
    print(“You must be logged in as an admin to use this function”)
```

Login to Account

Login

Abstract

A user needs a way to authenticate themselves and gain access to authorized functions. Logging in will allow the user to access other modules in the system.

Design

Login checks whether a user exist with a username and password that a person entered. That person is then logged in as a participant if they are a participant, a partner company employee if they are a partner company employee, a job coach if they are a job coach or an admin if they are an admin. They are logged in as a regular user if they are not part of one of these groups.

Error handling

The module checks if the user has a registered account. If the user does not already have an account, then they will receive a message saying, “A user with that username does not exist”. Also, if the user types in the wrong password, then they will also get an error for that.

Overall importance

Since Spectrum Works will be using a system for participants, employees, and job coaches, it is really necessary to have accounts for each user.

Pseudocode

```
Function login()
selectedusername=getNextString()
selectedpassword=getNextString()
If selectedusername matches up with one username
    If the selectedusername equals the corresponding password
        If user with selectedusername is a participant
            Log in user as a participant
        If user with selectedusername is a partner company employee
            Log in user as a partner company employee
        If user with selectedusername is a job coach
            Log in user as a job coach
        If user with selectedusername is an admin
            Log in user as an admin
    Else
    print("Username and password do not match")
Else
    print("A user with that username does not exist")
```

Topic

CreateTopic
Abstract
Topics provide a way for users have a place to group posts. This allows like minded users to start discussions.

Design
A user that is logged in creates a topic by entering its name and a initial post to begin the topic. From there, the topic is saved and the initial post is added to the topic.

Error handling
This module handles errors by making sure that the topicname is not empty.
Exports

Overall importance
This module is important since it allows users to create a place to discuss a particular topic.

Pseudocode

```
Function createtopic()
If logged in
    print("Enter the topic name")
    topicname=getNextString()
    if(topicname is not empty)
        print("Post:")
        firstpost=getNextString()
        Create new topic named topicname
        Create post in topicname with text firstpost
        print("Your topic has been made")
    Else
        print("Your topic must have a name")
Else
    print("You need to be logged in to view this")
    Goto login function

Function createpost(topicname)
```

```
If logged in
print("Post:")
currentpost=getNextString()
post(user,currentpost)
print("Your post has been made")
Function post(user,post)
Create post with text post by user user in topic topicname
Else
        print("You need to be logged in to view this")
        Goto login function
```

ViewTopic
Abstract
Users need to be able to view topics and see what users other have to say.
Design
This module prints out all of the posts in the given topic. It then provides a button for a user to click on in order for them to reply to that topic.
Sub-modules
CreatePost is a submodule that is used to allow users to make posts in topics. A post is written by replying to a topic.
Error handling
No major errors are handled by this module.

Overall importance
This module is important since it allows users to view what other users have said about topics and gives an easy way for users to move to replying to topics.
Pseudocode

```
Function viewtopic(topicname)
For each post in topicname
        print(postauthor)
        print(posttext)
button=new Button()
button.text=Reply
button.onclick=createpost(topicname)
```

Message

ViewMessages
Abstract
Users need to be able to view messages that were sent to them. This module allows a user to access all messages that have been sent to them.

Design

This module prints out links to all of the messages with receiver user and provides links to access each message. Each message has a messageid so they are easy to access and delete.

Error handling
This module handles errors by checking if a user is logged in before sending messages.

Overall importance
This module is important since viewers benefit from being able to receive messages.

Pseudocode
```
Function viewmessages()
If logged in
    user=GetCurrentUser()
    For each message with receiver user
        viewbutton=new Button();
        viewbutton.text=”View Message”
        deletebutton=new Button();
        deletebutton.text=”Delete Message”
        viewbutton.onclck=viewspecificmessage(message.messageid)
        deletebutton.onclick=deletespecificmessage(message.messageid)
Else
    print(“You need to be logged in to view this”)
    Goto login function

Function viewsentmessages()
If logged in
    user=GetCurrentUser()
    For each message with sender user
        viewbutton=new Button();
        viewbutton.text=”View Message”
        deletebutton=new Button();
        deletebutton.text=”Delete Message”
        viewbutton.onclck=viewspecificmessage(message.messageid)
        deletebutton.onclick=deletespecificmessage(message.messageid)
Else
    print(“You need to be logged in to view this”)
    Goto login function
```

ViewSpecificMessage
Abstract
This module is used to a specific message. This allows users to read the details of a message.

Design

This module works by getting the id of a message and then showing the message with that id.

Error handling
This module prevents errors from occurring by checking if a user is logged in.

Overall importance
This module is quite important since it allows users to actually read a specific message. This will allow for more communication to occur.

Pseudocode
```
Function viewspecifcmessage(messageid)
If logged in
        currentmessage=get message with messageid
        print(currentmessage.sender)
        print(currentmessage.receiver)
        print(currentmessage.subject)
        print(currentmessage.text)
        button=new Button()
        Button.text=”reply”
        button.onclick=sendmessage(currentmessage.sender)
Else
        print(“You need to be logged in to view this”)
        Goto login function
```

SendMessage
Abstract
This module is used to allow one user to send a message to another user.

Design
This module checks if a user is logged in and then lets them start sending a message. A user can input a receiver, a subject and the text of the message. The user then submits the message and it is sent to the user designated as the receiver.

Error handling
The error that will show up is that the user will be told that they need to logged in to view the messages.

Overall importance
This module is used to send messages, which is important since it promotes communication. That communication can allow for opportunities to occur, which is why this module is emphasized.

Pseudocode

```
Function sendmessage()
If logged in
    receiver=getNextString()
    subject=getNextString()
    text=getNextString()
    Send message to receiver with subject subject, text text and sender currentuser
    print(“Message sent”)
Else
    print(“You need to be logged in to view this”)
    Goto login function
```

```
Function sendmessage(lastsender)
If logged in
    receiver=lastsender()
    subject=getNextString()
    text=getNextString()
curmessageid=getmaxmessageid()+1
    Send message to receiver with subject subject, text text and sender currentuser and
messageid curmessageid
    print(“Message sent”)
Else
    print(“You need to be logged in to view this”)
    Goto login function
```

DeleteMessage

Abstract

This module lets users delete messages. This allows manage how many messages they have while also benefiting the system by lessening the number of messages that need to be stored as they are deleted.

Design

This module deletes a message with a specific id and then informs the user that the message has been deleted.

Error handling

The module checks if there are messages to delete.

Overall importance

In order for users to be able to keep track of what messages are important, they need to be able to delete their messages.

Pseudocode

```
Function deletemessage(messageid)
```

```
If logged in
    Delete message with messageid
    print(“The message has been deleted”)
Else
    print(“You need to be logged in to view this function”)
    Goto login function
```

Ban User

Abstract
Banning users is necessary when they do not follow rules set by the company.

Design
The admin will be prompted to add the name of the user they wish to ban. They will then be asked if they are completely sure of banning the user.

Error handling
A valid username has to be entered in order for the ban to proceed. Otherwise, the ban will be cancelled.

Overall importance
There will always be users that violate Spectrum Works’ terms of service, so it is important to revoke access from those users.

Pseudocode
```
Function banuser()
If logged in as admin
    print(“Enter the name of the user to ban”)
    selecteduser=getNextString()
    print(“Warning: Banning CANNOT BE UNDONE. Enter YES if you want to ban
”+selecteduser)
    optionselected=getNexString()
    if(optionselected==”YES”)
        Set selecteduser account as banned
        print(selecteduser+” has been banned”)
    Else
        print(“Ban canceled”)
Else
    print(“This is an admin only function.”)
    Goto login function
```

Remove Topic

Abstract

Sometimes topics may be posted that are entirely inappropriate, so they will need to be removed. This function allows admins to remove those topics.
Design
This function removes a specific topic by removing a topic with a specified username and a specified topicname. This will allow for removing only one topic.

Error handling
The error handling system checks if the topic exists before attempting to delete it.

Overall importance
This module is important since preventing inappropriate content from being in the system helps users and protects Spectrum Works' reputation. This will keep the forum more orderly.

Pseudocode
```
Function removetopic(username,topicname)
If logged in as admin
    if(topic with topicname by username exists)
        Remove topic with topicname by username
        print("Topic "+topicname+" by user "+username+" has been removed")
Else
    print("This is an admin only function.")
    Goto login function
```

Remove Post

Abstract
Sometimes a user may make a post that is inappropriate, so admins need to have the power to remove posts. This will prevent the system from being overrun by low quality or inappropriate posts.

Design
This module works by having an admin click on an x button nearby a post in order to choose to remove it. The button passes the poster's username, the topic name and the post number to the function. From there, a post by user username, in topic topicname and post number postnumber will be removed from the topic.

Error handling
The module checks if the post exists.

Overall importance
This module is important since it helps control issues with malicious and vulgar comments. This module gives admins some more control over communications.

Pseudocode

```
Function removepost(username,topicname,postnumber)
If logged in as admin
    if(there is a post number postnumber in topicname by username)
        Remove post number postnumber in topicname by username
        print("The offending post in topic "+topicname+" by user "+username+" has been removed")
Else
    print("This is an admin only function.")
    Goto login function
```

Index

getNext()-requires the user to input a variable of the given type
getNextString()-requires the user to input a String
getNextTime()-requires the user to input a time
getNextDate()-requires the user to input a date