Integral Solutions

University of Maine

COS 397 | Capstone I

System Requirements Specification

Team Members:
Vincent King | Peter Riehl | Mac Creamer
Liam Blair | Aaron Wilde

Professor:
Terry Yoo

October 2021

Integral Solutions

Table of Contents

1. Introduction

This capstone project serves as a step forward and a transition from the University of Maine's physical storage of Programs of Study (POS) documents, to the digital storage of these documents. This capstone project serves as a partial fulfilment of the Computer Science Bachelor of Science degree for the University of Maine. All parties involved with this capstone project benefit from its development and completion, those parties being: ***Integral Solutions***, a capstone group of five University of Maine undergraduate seniors who are the facilitators of this project, our client Professor Harlan Onsrud, the School of Computing and Information Science (SCIS) graduate faculty mentors, our instructor Doctor Terry Yoo, the Programs of Study department at the University of Maine, and the University of Maine as a whole.

1.1 Purpose of This Document

The purpose of this document is twofold. This document serves as a contract between ourselves, *Integral Solutions*, and our client, Doctor Harlan Onsrud. Additionally, this document outlines the requirement specifications of our product, whose temporary name at the moment is Project Graduate Modernization (PGM). Due to this project intending to be largely a "proof of concept" for the idea, the intended readership should be for both our client and for the high-ranking academic staff at the University of Maine, with the former being prioritized to ensure absolute clarity for agility's sake.

Within this document we discuss the need that this product fulfills and the plan to accomplish it, such as through: our requirements sections where the aspects of the system are carefully outlined in order for the system to function properly; specifications of the user interface where we demonstrate at a high level what the system is going to look like and requirements to be inferred from it; as well as the logistical overhead associated with the partnership between ourselves and our client, that being what will be delivered, expectations, and signatures recognizing that this document has been accepted by all involved parties.

1.2. References

- "UMaine Graduate Student Program of Study Creation and Approval System" Proposal
  - *Author: Doctor Harlan Onsrud*
  - *Date: September 2021*
- No further references used at this moment in time. Each future reference will use the following format.
  - Title
    - *Author*
    - *Publisher (if applicable)*
    - *Date*
    - *URL (if applicable)*

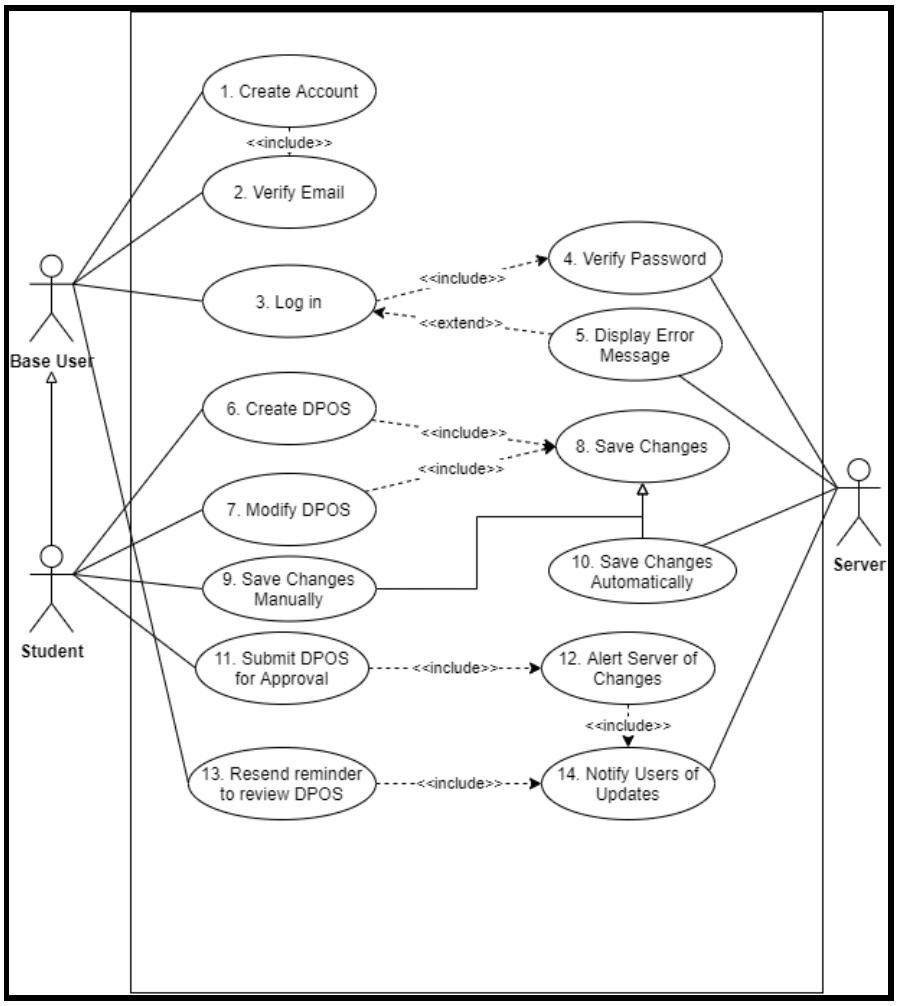
1.3. Purpose of the Product

The original proposal for this document outlined the need for a digital system to be created due to the large swathe of physical documents which are housed for the various graduate departments at the University of Maine. Specifically, the POS that graduate students use to dictate their graduate careers are cumbersome in both maintaining and updating. Professor Onsrud outlined the idea that POS's will sometimes get lost and that a number of signatures from advisors are required with every minute change that occurs with a given POS. While physical documents may be fine for a smaller number of students, at larger scales such as the one that the SCIS graduate programs are currently experiencing, it becomes unruly and a new solution is needed.

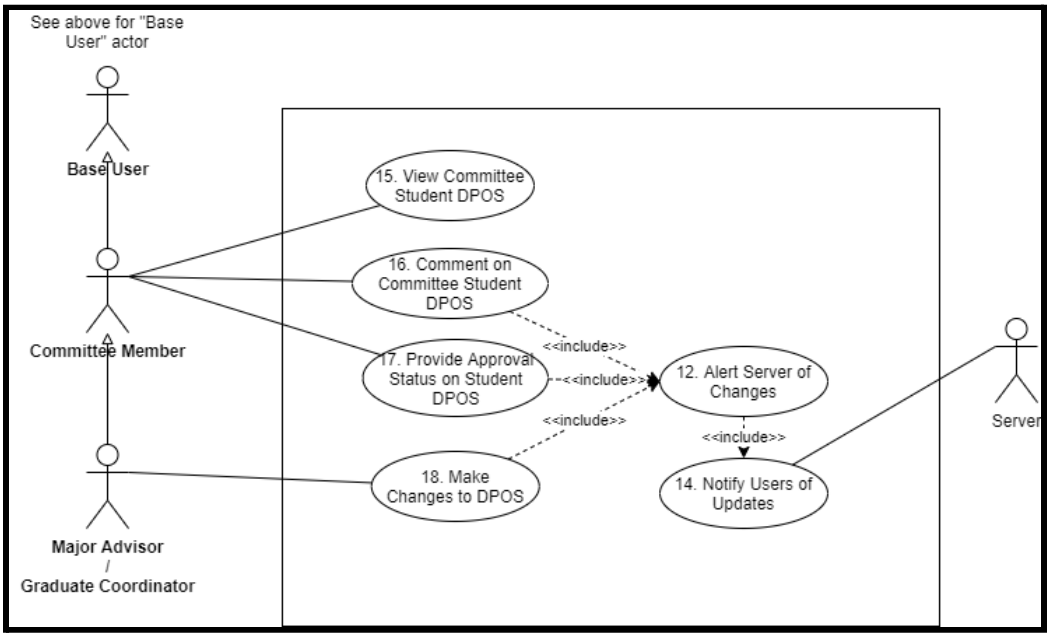
The goal of this project is to completely digitalize future POS's and remove the difficulty associated with the logistics of having physical copies, such as the ones outlined above. The system will house a given student's current POS, as well as provide avenues for students and advisors alike to make changes to a POS. Signatures needed will be replaced with electronic signatures and greatly reduce the labor and overhead associated with both significant and insignificant changes alike. Through housing a student's current POS and having a centralized digital hub for students to access their POS, pressure will be taken off of both students and professors alike. This should serve as a proof-of-concept for the University of Maine to take the proposal seriously and integrate our system into Mainestreet as a whole.

1.4. Product Scope

The next page (page 5) gives a visual representation of the project's scope. It details a limited number of actors and does not include moderators or administrators of the project. In another document, an updated variation of this Unified Modeling Language (UML) model will give information about such actors. The baseline of the project itself though will only include a high level view with actors that will experience this project on a "day-to-day" basis. Said "day-to-day" basis modeled below includes the process of creating an account, a student's POS experience, and a student's advisory committee experience interacting with their advisee's Draft POS (DPOS).



Account Creation and POS Management Diagram



Advisory Committee Actions Diagram

2. Functional Requirements

As outlined in section 1.4, there are a number of use cases relevant to the proposed system. In this section: Showcase the requirements that these use cases will satisfy and list the tests that will be performed to ensure that all functional requirements of the system have been met. In Appendix D, we will demonstrate all of the use cases and break them down with a template that has been adapted from Alistair Cockburn (as seen below). This is done to mitigate the "fluff" that becomes present with the addition of the use cases below.

Number	< use case number >	
Name	< use case name - a short active verb phrase >	
Summary	< a brief summary of the use case >	
Priority	< how critical this use case is to the customer (1 to 5, 5 being most critical >	
Preconditions	< conditions that must be true before the use case trigger >	
Postconditions	< conditions that will be true after the use case completes >	
Primary Actor	< a role name for the primary actor >	
Secondary Actors	< other systems that are relied upon to accomplish the use case >	
Trigger	< the action that starts the use case >	
Main Scenario	**Step**	**Action**
	1	< steps of the use case from trigger to goal delivery >
	2	< … >
	3	< … >
Extensions	**Step**	**Branching Action**
	1a	< condition causing branching > **:** < action or name of sub use case >
Open Issues	< list of issues awaiting decisions that affect the use case >	

2.1 Requirements

2.1.1 Website Navigation Requirements

1. The system shall have a navigation bar at the top of the screen.
2. The system shall contain several dropdowns in the navigation bar of areas a user would need to access for full site functionality.
3. The system shall have a "back-to-previous page" button in the navigation bar.
4. The system should contain a navigation panel.
5. The system should have a "?" help button in the navigation bar.

2.1.2 Account Management Requirements

1. The system shall allow users to log in with a specified email and password combination.
2. The system shall allow users to create accounts with a specified email and password combination.
3. The system should allow users to enter additional information.
4. The system should assist users based on the additional information they have entered.

5. The system shall allow the user to submit an account creation form upon filling out the required account creation information.
6. The system shall verify that the email used is in the “University of Maine System” (UMS) email domain.
7. The system shall notify the user that their email will not be accepted if it is not in the UMS email domain.
8. The system shall notify the user if the email they have entered does not follow a proper email format.
9. The system shall not accept the account creation submission if the user does not enter a valid UMS email address.
10. The system shall return a message if the account creation submission is not accepted.
11. The system shall clarify the reason why an account creation submission was not accepted on a case-by-case basis.
12. The system shall send the user an email to their specified email address if the account creation submission was accepted.
13. The system shall store an inactive account with the user’s information upon receiving a valid account creation submission.
14. The system shall set the status of an inactive account to active upon the associated user clicking on a link within a system-sent email to verify their account.
15. The system should remove accounts that have an inactive status after a period of 7 days.
16. The system should have an account management feature for administrators to remove/modify user accounts.
17. The system should have an account management feature for administrators to swap a user’s account from inactive to active.
18. The system should have a log of account management actions taken by administrators.
19. The system should have a button to allow an administrator to undo the most recent account manipulation action taken.
20. The system shall include an account recovery button.
21. The system’s account recovery form shall include an email field.
22. The system shall provide a submission button for the user to send an account recovery email.
23. The system shall only send an account recovery email if the email exists within the database.
24. The system shall inform the user that an email has been sent, regardless of whether the email is in the database.
25. The system shall not allow the user to sign up with an email that already exists in the database.
26. The system should inform the user that an email has been sent, regardless of whether the email exists in the database.
27. If a user attempts to sign up with an email that is already in use, the system should notify the email address owner of the attempt via email.

2.1.3 DPOS Form Submission Requirements

1. The system shall let students with accounts create DPOS's.
2. The system shall not let a student create multiple DPOS's for the same field of study.
3. The system shall allow the user to fill out a digital form for a DPOS.
4. The system shall have a dropdown that includes each type of DPOS.
5. The system shall adjust the input fields within a DPOS form based on the dropdown option selected.
6. The system should provide a link to the catalog of courses for students to reference.
7. The system shall allow a user to electronically sign their DPOS for review via typing their name.
8. The system shall allow a user to submit their DPOS form for review.
9. The system shall save the progress of the form every 30 seconds.
10. The system shall also include a button that manually lets a user save their form changes.
11. The system shall let students with accounts adjust their DPOS forms they have not yet submitted for approval.
12. The system shall send the user a confirmation email after submitting their DPOS for approval.
13. The system shall update the user's DPOS for the given Field of Study when their form is submitted.
14. The system shall immediately notify a student's advisory committee that the student's DPOS has been submitted for review.
15. If members of a student's advisory committee do not exist within the database, the system shall allow a user to have those advisors receive email notifications.
16. The system shall allow the user to indicate their "primary" advisor via a selection of registered SCIS graduate advisors.
17. The system shall allow advisory committee members to comment on a submitted DPOS.
18. The system should default to a committee member giving their approval if the respective committee member does not comment or post a status within 2 weeks.
19. The system shall allow advisory committee members of a given student to request revisions for that student's DPOS.
20. The system shall allow the graduate coordinator to request revisions for a student's DPOS.
21. The system shall allow an advisor to give their approval status (includes rejections and revision requests) of a DPOS that has a review requested.
22. The system shall allow for an advisor to electronically sign off on their approval status of a DPOS.
23. The system shall allow for the graduate coordinator to electronically sign off on their approval status of a DPOS.
24. The system shall notify the student user of their advisor's approval status.
25. The system shall allow a user to adjust their DPOS if it was not approved.
26. The system shall allow a user to re-request review of a DPOS that was not approved.

27. The system shall flag a DPOS as an Approved POS (APOS) when all necessary parties have signed off.
28. The system shall create a PDF document when a DPOS becomes an APOS.
29. The system shall email the PDF document to Harlan Onsrud when a DPOS becomes an APOS.
30. The system should allow for the graduate coordinator to download programs of study.
31. The system shall add the PDF of the user's APOS to that respective user's database entry when their DPOS becomes an APOS.
32. The system shall store relevant identifying information of an APOS, such as date of approval and version number, when entering the form data into the user's database entry.
33. The system shall allow a student to create a new DPOS to make changes to from a previous APOS.

2.1.4 Changes to APOS

1. The system shall highlight courses that a student has changed that have not yet received approval.
2. The system shall create a DPOS when a student has made a change to a given APOS that the student has on record.
3. The system shall remove highlights from courses when it receives advisory committee and graduate coordinator approval.
4. The system shall remove flags on a DPOS when it receives advisory committee and graduate coordinator approval.
5. The system shall include a button that manually lets a user save their POS form changes.
6. The system shall automatically save form changes every 30 seconds.

2.1.5 Communication Requirements

1. The system should give students a button to "remind" their advisors of necessary action relevant to the student's DPOS.
2. The system should automatically inform the users of important upcoming deadlines within 2 weeks of said deadlines.
3. The system shall keep emails to a 50 word limit.
4. The system shall include pertinent information only (Student name, subject of POS, relevant deadlines, accreditation level) when sending emails.
5. The system should allow users to opt-out of emails.

2.1.6 Database Management Requirements

1. The system shall always store the most up to date POS that a student has requested changes on.
2. The system shall always store the most recent DPOS's for each field of study the user has a POS for.
3. The system shall contain all APOS's associated with each student.
4. The system should contain an up-to-date list of the SCIS graduate advisors.

2.1.7 Other Functional Requirements

1. The system shall be accessible from Google Chrome.

2.2 Test Cases

Per a discussion with Professor Terry Yoo, the test cases outlined are **generalized**. It is the intent of integral solutions to create a functioning system that both works as a standalone product and as a proof of concept. To ensure we manage to create this system, and in line with the requirements of the class itself, we are following through with the generalization of test cases. This does mean that they will not be entirely exhaustive, however, they serve as a guide for which to ensure the system works sufficiently.

The test case section will be broken down into the use cases shown in section 1.4.

2.2.1 Create Account

1. Attempt to put in all "valid" student information. (Valid email, valid password)
2. Attempt to put in all "valid" employee information. (Valid email, password, ID)
3. Attempt to put in invalid email. (Non-UMS email)
4. Attempt to put in a valid email and password with invalid employee ID number. (valid email, valid password, invalid employee ID)
5. Attempt to put in a valid email with student information but an invalid password (valid email, invalid password)
6. Attempt to put in a valid email with employee information but invalid password (valid email, valid ID, invalid password)

2.2.2 Verify Email

1. Attempt to use a "valid" email address within 7 days. (Email address in database)
2. Attempt to use an invalid email address within 7 days. (Email address not in database)
3. Attempt to use a valid email address after 7 days.
4. Attempt to use an invalid email address after 7 days.

2.2.3 Log In

1. Attempt to log in with an invalid password and a valid email address.
2. Attempt to log in with a valid password and an invalid email address.
3. Attempt to log in with a valid password and valid email address.

2.2.4 Verify Password

1. *Test cases here are analog to Use Case "Log In"*

2.2.5 Display Error Message

1. Pass a -1 value to the "Display Error Message" function with an attempt to log in with an invalid password and a valid email address.

2. Pass a -1 value to the "Display Error Message" function with an attempt to log in with a valid password and an invalid email address.

2.2.6 Create DPOS

1. The user selects "Graduate Certificate" and fills out the form to completion.
2. The user selects "Master's Degree" and fills out the form to completion.
3. The user selects "PhD Degree" and fills out the form to completion.
4. The user selects "Graduate Certificate" and does not fill out the form to completion.
5. The user selects "Master's Degree" and does not fill out the form to completion.
6. The user selects "PhD Degree" and does not fill out the form to completion.
7. The user attempts to create a DPOS when they currently have no DPOS.
8. The user attempts to create a DPOS when they have between 1 and 4 DPOS's on file.
9. The user attempts to create a DPOS when they have 5 DPOS's on file.

2.2.7 Modify DPOS

1. The user does not make changes to the form.
2. The user does make changes to the form.
3. The user makes changes to the form but does not manually save them.
4. The user makes changes to the form and does manually save them.
5. The user attempts to select "modify DPOS" when they have no DPOS on file.
6. The user attempts to modify a DPOS when they have a DPOS on file.
7. The user attempts to modify a DPOS when they have multiple DPOS's on file.

2.2.8 Save Changes

1. Trigger the use cases that are children of this use case.
   This interacts with the server and the database, so the children of this use case include the test cases present here. Storage of a DPOS in the database, specifically, are included in the save changes stuff below.
2. The DPOS form being saved to the database has information stored within it, but the storage is unsuccessful.
3. The DPOS form being saved to the database has information stored within it, but the storage is successful.

2.2.9 Save Changes Manually

1. The user clicks on the "Save Changes" button on a form with an empty DPOS that has not been previously saved.
2. The user clicks on the "Save Changes" button on a form with a filled DPOS that has not been previously saved.
3. The user clicks on the "Save Changes" button on a form with a partially-filled DPOS that has not been previously saved.
4. The user clicks on the "Save Changes" button on a form with an empty DPOS that has been previously saved.
5. The user clicks on the "Save Changes" button on a form with a filled DPOS that has been previously saved.

6. The user clicks on the “Save Changes” button on a form with a partially-filled DPOS that has been previously saved.

2.2.10 Save Changes Automatically

1. A 30-second, looping timer triggers on a form with an empty DPOS that has not been previously saved.
2. A 30-second, looping timer triggers on a form with a filled DPOS that has not been previously saved.
3. A 30-second, looping timer triggers on a form with a partially-filled DPOS that has not been previously saved.
4. A 30-second, looping timer triggers on a form with an empty DPOS that has been previously saved.
5. A 30-second, looping timer triggers on a form with a filled DPOS that has been previously saved.
6. A 30-second, looping timer triggers on a form with a partially-filled DPOS that has been previously saved.

2.2.11 Submit DPOS for Approval

1. The user attempts to submit a DPOS that has not been completely filled out.
2. The user attempts to submit a DPOS that has been filled out completely.
3. The user clicks outside of the prompt area that asks if they are sure they want to submit their DPOS for approval.
4. The user clicks “Yes, submit” on a form that is completely filled out.
5. The user clicks on “No” on a form that is completely filled out.

2.2.12 Alert Server of Changes

1. Trigger the use cases that require this use case.
   This has no direct interaction with the user, so it should only trigger off the other use cases triggering. It sends a response to the server, so it should interact with the server in that regard. As long as the server receives notice of the alert, this use case is satisfied.

2.2.13 Resend Reminder to review DPOS

1. A user manually submits a reminder for members of their advisory committee to provide comments/feedback on their DPOS.
2. Advisory members have not reviewed the DPOS within a 2 week period.

2.2.14 Notify Users of Updates

1. The server sends out a reminder for all advisory members to review the DPOS.
2. The server sends out a reminder for specific advisory members to review the DPOS.

2.2.15 Join Student Committee

1. A committee member accepts an invitation to join a student’s committee.
2. A committee member declines an invitation to join a student’s committee.
3. An advisor accepts an invitation to join a student’s committee.
4. An advisor declines an invitation to join a student’s committee.

2.2.16 View Committee Student DPOS

1. An advisor attempts to view a committee student's DPOS.
2. A committee member attempts to view a committee student's DPOS.
3. A graduate coordinator attempts to view a committee student's DPOS.

2.2.17 Comment on Committee Student DPOS

1. An advisor attempts to provide a comment on a committee student's DPOS with a text field that is left blank.
2. An advisor attempts to provide a comment on a committee student's DPOS with a text field that is filled out.
3. A committee member attempts to comment on a committee student's DPOS with a text field that is left blank.
4. A committee member attempts to comment on a committee student's DPOS with a text field that is filled out.
5. A graduate coordinator attempts to comment on a committee student's DPOS with a text field that is left blank.
6. A graduate coordinator attempts to comment on a committee student's DPOS with a text field that is filled out.

2.2.18 Provide Approval Status on Student DPOS

1. A major advisor attempts to provide their approval status on a committee student's DPOS.
2. A committee member attempts to provide their approval status on a committee student's DPOS.
3. A graduate coordinator attempts to provide their approval status on a committee student's DPOS.

2.2.19 Make Changes to DPOS

1. A major advisor attempts to make changes on a change form that has no changes in it.
2. A major advisor attempts to make changes on a change form that has changes in it.
3. A graduate coordinator attempts to request changes on a change form that has no changes in it.
4. A graduate coordinator attempts to request changes on a change form that has changes in it.

3. Non-Functional Requirements

Listed below are the non-functional requirements (NFRs) associated with the project as a whole. Test cases associated with each requirement are listed below, along with the respective ID # that the test case addresses. Some requirements will have more than one test case, while others may be lumped together in a single, larger test case.

ID #	NFR Description	Priority
1	The system shall abide by all FERPA requirements.	3
2	The system shall abide by any addendums that the University has made to its specific FERPA requirements.	3
3	The system database shall be hosted from a University of Maine owned computer.	1
4	The system shall handle up to 1,000 asynchronous users.	4
5	The system shall handle up to 100 concurrent users.	4
6	System response times must not exceed 8 seconds (real time).	2
7	Modified data in the database shall be updated for all users accessing it within 1 minute (real time).	3
8	The system shall notify student users of approved proposals within 1 minute of approval.	3
9	The system shall verify user passwords within 8 seconds of the login attempt.	3
10	The server shall automatically save modified data within 1 minute of any changes made.	3
11	The system database shall be copied to a backup database every hour.	3
12	The system database shall be protected from power surges.	3
13	The system database shall be stored in a room secured from unauthorized access at the University of Maine.	3

3.1 Test Cases

Per a discussion with Professor Terry Yoo, the test cases outlined are **generalized**. It is the intent of integral solutions to create a functioning system that both works as a standalone product and as a proof of concept. To ensure we manage to create this system, and in line with the requirements of the class itself, we are following through with the generalization of test cases. This does mean that they will not be entirely exhaustive, however, they serve as a guide for which to ensure the system works sufficiently.

The test case section will be broken down into the use requirements shown earlier in this section.

3.1.1. NFR #1
   i. System review will take place to ensure this is followed through and unauthorized access is not permitted.

3.1.2. NFR #2
   i. Analog to NFR #1.

3.1.3. NFR #3
   i. System review will take place of our physical server location and shall be ensured that it is on a University of Maine owned device.

3.1.4. NFR #4
   i. A testing tool that creates and populates 1000 accounts shall be utilized to test that the server is capable of handling this number of stored users.
3.1.5. NFR #5
   i. A testing tool analogous to NFR #4 will be used to create batches of 99 accounts at a time in order to stress test the number of concurrent users the system can handle.
3.1.6. NFR #6
   i. Testing tools will be utilized during the testing of NFR #4 and NFR #5 to ensure that even under high stress environments, the system maintains a low page response time.
3.1.7. NFR #7
   i. Analog to NFR #6.
3.1.8. NFR #8
   i. Analog to NFR #6.
3.1.9. NFR #9
   i. Analog to NFR #6.
3.1.10. NFR #10
   i. Analog to NFR #7.
3.1.11. NFR #11
   i. Tools will be used to monitor and inspect a database backup that occurs every hour.
3.1.12. NFR #12
   i. Largely analogous to NFR #3, proper inspection of physical equipment will be necessary to ensure proper protection of the server.
3.1.13. NFR #13
   i. Analog to NFR #3.

4. User Interface

See the User Interface Design Document (UIDD) for Integral Solutions' website application after the required turn in date.

5. Deliverables

The following is a list of deliverable items which will be delivered to the customer by Integral Solutions. Included in the chart is the format of the document, expected completion date, and the means of delivery.

Item	Expected Completion Date	Format	Delivery means
SRS	October 10th 2021	Word Doc	Email / In Person
SDD	November 10 2021	Word Doc	Email / In Person
UIDD	November 29 2021	Word Doc	Email / In Person
User Manual	TBD	Word Doc	Email / In Person
Administrator Manual	TBD	Word Doc	Email / In Person
Final Software (FS)	TBD	TBD	Email / GitHub
- FS *Source Code*	TBD	TBD	Email / GitHub
- FS *Website*	TBD	TBD	Email / GitHub

6. Open Issues

The following is a list of all open issues which are being actively worked on by Integral Solutions. The chart below includes the issue name, expected completion date, and priority level (Low-Medium-High)

Issue	Expected Completion Date	Priority Level
Find University Owned Host Computer	November 29 2021	Medium
Pick Programming Language	November 22 2021	Medium
Pick Database System	November 22 2021	Medium
Aquire Email Account for Automated Messaging System	TBD	Medium
Get FERPA Approval	November 22 2021	Medium

Appendix A – Agreement Between Customer and Contractor

By signing this document, the customer and development team agree to the requirements listed above. Both parties will also agree to the defined function and scope of the project, as well as to all functional and non-functional requirements that the project must meet. Both parties will additionally agree to the contents of each promised deliverable stated above. The development team agrees to provide a software system that meets said requirements at a later date.

In the case of changes to the document, the customer will be informed of the changes via email. These changes would have to be approved by the customer before they are made. Meetings may be scheduled in order to discuss any proposed changes to the document. By signing this document, both parties agree to use said procedure in the event of changes to the document.

By signing below, the customer and development team agree to the above. Additionally, the customer may write any comments or concerns they may have in the space below.

Customer Comments:

Customer Signature: _____

Development Team Signatures:

_____ _____

_____ _____

_____

Appendix B – Team Review Sign-off

By signing below, both parties confirm that they have reviewed the contents of this document. Additionally, both parties will confirm that they have agreed on the document's content and format.

Team Member Comments:

1. _____
   _____
   _____
2. _____
   _____
   _____
3. _____
   _____
   _____
4. _____
   _____
   _____
5. _____
   _____
   _____

Customer Name: _____ Customer Signature: _____

Date of Signature: _____

Team Names:	Team Signatures:	Date of Signatures:
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Appendix C – Document Contributions

Liam Blair
- Reviewed and proofread elements of the document **(20% of total effort)**
- Provided both functional and non-functional requirements **(10% of total effort)**
- Provided the following sections:
    - Section 2 Introduction **(50% of total effort)**
    - Appendix D **(15% of total effort)**

Mac Creamer
- Reviewed and proofread elements of the document **(20% of total effort)**
- Provided both functional and non-functional requirements **(55% of total effort)**
- Provided the following sections:
    - Section 1.1 **(100% of total effort)**
    - Section 1.2 **(100% of total effort)**
    - Section 1.4 **(100% of total effort)**
    - Section 2 Introduction **(50% of total effort)**
    - Section 3 Formatting and Introduction **(100% of total effort)**
    - Appendix C **(80% of total effort)**
    - Appendix D **(30% of total effort)**
    - Test Cases **(100% of total effort)**

Vincent King
- Reviewed and proofread elements of the document **(20% of total effort)**
- Provided both functional and non-functional requirements **(10% of total effort)**
- Provided the following sections:
    - Section 1 Introduction **(100% of total effort)**
    - Section 1.3 **(100% of total effort)**
    - Appendix D (**15% of total effort)**

Peter Riehl
- Reviewed and proofread elements of the document **(20% of total effort)**
- Provided both functional and non-functional requirements **(15% of total effort)**
- Provided the following sections:
    - Section 5 **(100% of total effort)**
    - Section 6 **(100% of total effort)**
    - Appendix C **(20% of total effort)**
    - Appendix D **(25% of total effort)**

Aaron Wilde
- Reviewed and proofread elements of the document **(20% of total effort)**
- Provided both functional and non-functional requirements **(10% of total effort)**
- Provided the following sections:
    - Appendix A **(100% of total effort)**
    - Appendix B **(100% of total effort)**
    - Appendix D **(15% of total effort)**

Appendix D – Use Case Tables

Below are the use case tables referenced in Section 2. These are included here to keep the document itself from being flooded with these tables.

Number	1	
Name	Create Account	
Summary	Summarizes creating user account which give database access to student and staff	
Priority	4	
Preconditions	Valid @maine.edu email account,	
Postconditions	Valid login account, ability to create/destroy/modify/submit POS. If the account is owned by staff they may create/destroy/modify/approve/request changes to POS.	
Primary Actor	Student and Staff	
Secondary Actors	Server	
Trigger	Clicking on create account link. The link may be represented in many forms: button, link, ect.	
Main Scenario	**Step**	**Action**
	1	User navigates to website
	2	User clicks create account
	3	User enters @maine.edu email
	4	User enters desired passcode
	5	System displays creating account loading screen
	6	System directs the user to the account home page.
Extensions	**Step**	**Branching Action**
	3a	Verify email is invalid Return -1 / Loop back to step 3.
	3b	Email is valid Check to see if email is registered with staff member
	3b.1	Email is not registered with staff member Return 1/ Assign Student Key
	3b.2	Email is registered with staff member Ask user for employee ID number / Assign Staff Key - Can be Graduate Coordinator, Major Advisor, or Committee Member depending on email / Return 1
	4a	Passcode invalid (Does not contain: 1 capital letter, 1 special character, string length <8 characters.) Return -1
	4b	Passcode valid (Contains: 1 capital letter, 1 special character, string length >=8 characters) Return 1
Open Issues		

Number	2	
Name	Verify Email	
Summary	Server checks to make sure @maine.edu email is valid	
Priority	3	
Preconditions	Server must be set up	
Postconditions	User will be able to create or access their account and use all website functionalities	
Primary Actor	Server	
Secondary Actors	Student and Staff	
Trigger	Step 3 in log in, step 3 in create account	
Main Scenario	**Step**	**Action**
	1	User attempts to log in or create an account
	2	Email is sent to server as a query
	3	Server searches database
	4	Returns sentinel value to requesting interface
Extensions	**Step**	**Branching Action**
	3a	Server confirms email valid Returns 1
	3b	Server confirms email invalid Returns -1 / Goes to next step
Open Issues		

Number	3	
Name	Log in	
Summary	Server logs user into account and directs them to user homepage	
Priority	4	
Preconditions	User must own account	
Postconditions	User will be able to access website functionalities	
Primary Actor	Student and Staff	
Secondary Actors	Server	
Trigger	Students or Staff click login on the website UI.	
Main Scenario	**Step**	
	1	User enters username and password
	2	User clicks on login button
	3	Server searches through database to verify email
	4	Server searches through database to verify passcode
	5	User is directed to their profile homescreen
Extensions	**Step**	
	3a	Server confirms email valid Returns 1
	3b	Server confirms email invalid Returns -1

	4a	Server confirms passcode is valid Return 1
	4b	Server confirms passcode is invalid Returns -1
Open Issues		

Number	4	
Name	Verify Password	
Summary	The server verifies that the password submitted by the user is correct for the email provided.	
Priority	5	
Preconditions	The user must have pressed the login button and filled out the email and password fields.	
Postconditions	There will exist a variable that indicates whether the password/email combination was correct.	
Primary Actor	Server	
Secondary Actors	Base User	
Trigger	The server receives an email and password combination from the user.	
Main Scenario	**Step**	**Action**
	1	Server receives receives an email and password combination
	2	Server queries database for the provided email
	3	Server compares the hashed value provided by the user and the hashed value stored in the database
	4	Server returns the result of the compared values from the previous step
Extensions	**Step**	**Branching Action**
	2a	Server finds a database entry for the provided email **:** Server grabs the hashed value of the user's password
	2b	Server does not find a database entry for the provided email: Server skips step 3 and returns -1
	3a	Compared values are the same: Returns 1 in step 4
	3b	Compared values are not the same: Returns -1 in step 4
Open Issues		

Number	5	
Name	Display Error Message	
Summary	The system displays an unsuccessful login attempt message	
Priority	2	
Preconditions	Use case #4 returns -1	
Postconditions	The server displays an error message that the user sees to notify them of an unsuccessful login attempt.	
Primary Actor	Server	
Secondary Actors	Base User	
Trigger	Use case #4 returns -1	
Main Scenario	**Step**	**Action**
	1	The front-end of the system receives a -1 value from use case #4
	2	The system updates the front-end display to indicate the user's login attempt was unsuccessful.
	3	The user sees the login screen displaying an error message stating that the email and password combination was not valid.
Open Issues		

Number	6	
Name	Create DPOS	
Summary	The user creates a new Draft Program of Study that the system stores.	
Priority	5	
Preconditions	The user must have a registered account	
Postconditions	The server must store the completed form that the user creates.	
Primary Actor	Student User	
Secondary Actors	Server	
Trigger	The student user selects the "Create DPOS" button.	
Main Scenario	**Step**	**Action**
	1	The system makes sure the user doesn't already have an open DPOS for desired program credentials (Prompt asks what SCIS graduate degree or certificate is being sought)
	2	The user is brought to a screen that defaults to "Graduate Certificate", but has a dropdown option for the user to select "Master's Degree" and "PhD Degree"
	3	The user makes additions to fill the form
	4	The user or server attempts to save the work completed within the form so far
	5	The user completes the DPOS form they selected
	6	The user selects the "Submit" button
	7	Send notification to Graduate Coordinator, Major Advisor, and Committee members.
	8	Committee member timer starts.
Extensions	**Step**	**Branching Action**

	1a	The user has an open DPOS for desired program credential: Use case is halted and user is presented an error message
	3a	The user does not fill out the form: A variable called “formStarted” is set to 0
	3b	The user does fill out at least some of the form: A variable called “formStarted” is set to 1
	4a	If “formStarted” equals 0: The system will not save the DPOS form
	4b	If “formStarted” equals 1: The system will trigger use case 9
	5a	The user has completed the DPOS form: The system sets a variable called “formComplete” to 1
	5b	The user has not completed the DPOS form: The system sets a variable called “formComplete” to 0
	6a	The user does not select the submit button: Loop back to step 3
	6b	The user selects the submit button and “formComplete” equals 0: System displays an error message and loop back to step 3
	6c	The user selects the submit button and “formComplete” equals 1: Trigger use case #12
Open Issues		

Number	7	
Name	Modify DPOS	
Summary	The user attempts to modify a DPOS they have previously saved.	
Priority	3	
Preconditions	The user must have at least one saved DPOS.	
Postconditions	The DPOS will be updated per the student user’s changes	
Primary Actor	Student User	
Secondary Actors	Server	
Trigger	The user selected the “Modify DPOS” button.	
Main Scenario	**Step**	**Action**
	1	The server receives the request from the user to modify their DPOS.
	2	The server queries the database for the DPOS.
	3	The server returns the DPOS to the front-end system.
	4	The form that the user originally used to create or previously modify their DPOS is populated with their previously saved information. (See use case #6)
	5	The user makes changes as necessary to the form.
Extensions	**Step**	**Branching Action**
	3a	The server fails to locate and/or return the requested DPOS **:** The use case ceases and an error message is displayed to the user.
	5a	The user makes at least one change to the form: The system sets a variable called “formModified” is set to 1

	5b	The user does not make any changes to the form: The system sets a variable called “formModified” is set to 0
Open Issues		

Number	8	
Name	Save Changes	
Summary	The server stores the DPOS that the user is currently populating with information	
Priority	5	
Preconditions	The user must have created a DPOS and populated some of the fields with information.	
Postconditions	The server stores the DPOS or updates a previously stored iteration of the DPOS.	
Primary Actor	Student User	
Secondary Actors	Server	
Trigger	Either the student user presses a “save” button or the server automatically saves the form after 30 seconds. (See use cases 10 and 11)	
Main Scenario	**Step**	**Action**
	1	The server receives one of the two triggers mentioned above.
	2	The server creates a temporary form object if the form currently being worked on is not blank.
	3	The server adds the unique identifier that corresponds with the current DPOS.
	4	The server updates fields of the temporary form object with exact copies of the information present in the DPOS at the time of the trigger.
	5	The server updates the DPOS with the same unique identifier on record with the temporary form object that contains the same unique identifier.
	6	The server returns the status of whether the temporary form object was successfully saved.
	7	The system displays whether the attempt to save the current form was successful or not.
Extensions	**Step**	**Branching Action**
	2a	The DPOS form is blank **:** The use case ends and no further action is taken
	3a	No such unique identifier exists : The DPOS is assigned a unique random identifier
	5a	Update is unsuccessful: Return -1
	5b	Update is successful: Return 1
	7a	Value from step 5 was -1: Display to the user that the save was unsuccessful

	7b	Value from step 5 was 1: Display to the user that the save was successful
Open Issues		

Number	9	
Name	Save Changes Manually	
Summary	User manually request for the server to save changes they have made to a DPOS or POS.	
Priority	2	
Preconditions	User must have an account, have logged in, has an open POS or DPOS	
Postconditions		
Primary Actor	Server	
Secondary Actors	Student or staff	
Trigger	Student or staff clock on save changes button	
Main Scenario	**Step**	**Action**
	1	The server receives user initiated save prompt
	2	The server creates a temporary form object if the form currently being worked on is not blank.
	3	The server adds the unique identifier that corresponds with the current DPOS.
	4	The server updates fields of the temporary form object with exact copies of the information present in the DPOS at the time of the trigger.
	5	The server updates the DPOS with the same unique identifier on record with the temporary form object that contains the same unique identifier.
	6	The server returns the status of whether the temporary form object was successfully saved.
	7	The system displays whether the attempt to save the current form was successful or not.
Extensions	**Step**	**Branching Action**
	2a	The DPOS form is blank **:** The use case ends and no further action is taken
	3a	No such unique identifier exists : The DPOS is assigned a unique random identifier
	5a	Update is unsuccessful: Return -1
	5b	Update is successful: Return 1
	7a	Value from step 5 was -1: Display to the user that the save was unsuccessful
	7b	Value from step 5 was 1: Display to the user that the save was successful
Open Issues		

Number	10	
Name	Save Changes Automatically	
Summary	System automatically saves documents every 30 seconds.	
Priority	4	
Preconditions	Users must have an open POS or DPOS. Server timer enabled in real time.	
Postconditions		
Primary Actor	Server	
Secondary Actors	Students or staff	
Trigger	Server sentinel value timer hits 30 second mark	
Main Scenario	**Step**	**Action**
	1	The server receives user initiated save prompt
	2	The server creates a temporary form object if the form currently being worked on is not blank.
	3	The server adds the unique identifier that corresponds with the current DPOS.
	4	The server updates fields of the temporary form object with exact copies of the information present in the DPOS at the time of the trigger.
	5	The server updates the DPOS with the same unique identifier on record with the temporary form object that contains the same unique identifier.
	6	The server returns the status of whether the temporary form object was successfully saved.
	7	The system displays whether the attempt to save the current form was successful or not.
Extensions	**Step**	**Branching Action**
	2a	The DPOS form is blank **:** The use case ends and no further action is taken
	3a	No such unique identifier exists : The DPOS is assigned a unique random identifier
	5a	Update is unsuccessful: Return -1
	5b	Update is successful: Return 1
	7a	Value from step 5 was -1: Display to the user that the system auto saved unsuccessfully
	7b	Value from step 5 was 1: Display to the user that the system auto saved successfully
Open Issues		

Number	11
Name	Submit DPOS for Approval

Summary	User submits their DPOS for approval	
Priority	5	
Preconditions	User must have filled out the form completely	
Postconditions	Alert server of changes	
Primary Actor	Student	
Secondary Actors	Server	
Trigger	User clicks on Submit for Approval button	
Main Scenario	**Step**	**Action**
	1	The user receives a prompt asking if they are sure
	2	The user responds to the on screen prompt
	3	The server alerts the appropriate users to review the POS
Extensions	**Step**	**Branching Action**
	2a	User clicks on “Yes, Submit” **:** The server alerts the appropriate users to review the POS
	2b	User clicks on “No” : The system returns to the DPOS
Open Issues		

Number	12	
Name	Alert Server of Changes	
Summary	After the user makes changes to their submission, the server gets notified of it, and notifies the appropriate users to review.	
Priority	3	
Preconditions	User must have submitted a prior POS submission.	
Postconditions	Server and relevant users have been notified of the changes made.	
Primary Actor	User	
Secondary Actors	Server	
Trigger	All - Use Case #13 Student - Use Case #11 Advisor Committee/Grad Coordinator - Use Cases #16 and 17 Major Advisor/Grad Coordinator - Use Case #18	
Main Scenario	**Step**	**Action**
	1	A user triggers any of the above use cases
	2	The system recognizes that a noteworthy event has occurred and begins a notification process.
	3	The system triggers the server to start use case #14
Open Issues		

Number	13
Name	Resend reminder to review DPOS
Summary	A user can send out a reminder to any member to review the DPOS with any changes it may contain.

Priority	1	
Preconditions	A DPOS exists and has not been commented on or viewed by at least 1 associated user	
Postconditions	Notify Users of Updates (Use Case #14) is triggered	
Primary Actor	User	
Secondary Actors	Server	
Trigger	A user selects the “Resend Reminder to Review DPOS” button for a given DPOS.	
Main Scenario	**Step**	**Action**
	1	The system receives the request to remind users who have not yet commented on or viewed the DPOS.
	2	The system queries the database to find the DPOS to see who has not yet viewed, commented on, or approved the DPOS.
	3	The system takes note of the users who have not yet viewed the most recent changes to a DPOS, as well as making note of whether the graduate coordinator and Major advisor have provided their approval status yet.
	4	The system triggers Use Case #14 with the noted users
	5	The system emails displays a message that the server is in the process of sending the reminder.
Extensions	**Step**	**Branching Action**
	2a	The system fails to find the DPOS in question: Use case ends and displays an error message to the user
	3a	There is no one that the system is able to take note on, as all members have viewed the DPOS and the graduate coordinator and the Major advisor have both provided their approval statuses : Use case ends and displays a message to the user saying there is no need to resend the reminder.
Open Issues		

Number	14	
Name	Notify Users of Updates	
Summary	The server notifies the appropriate users of updates to a DPOS	
Priority	4	
Preconditions	The server must have received changes or comments to a DPOS	
Postconditions	The server will notify users to review the DPOS	
Primary Actor	Server	
Secondary Actors	User	
Trigger	Use Case #12 (Alert Server of Changes) and Use Case #13 (Resend Reminder to Review DPOS)	
Main Scenario	**Step**	**Action**
	1	Server receives a request to notify users of updates
	2	Server sends the notification email to the appropriate people
Open Issues		

Number	15	
Name	View Committee Student DPOS	
Summary	Display Committee Student DPOS	
Priority	5	
Preconditions	Committee Student DPOS must be submitted.	
Postconditions	Committee Student DPOS will be displayed	
Primary Actor	Student, Graduate Coordinator, Advisory Committee	
Secondary Actors	Server	
Trigger	Server receives request to display committee Student DPOS	
Main Scenario	**Step**	**Action**
	1	Server receives request to display committee student DPOS
	2	Server Locates committee student DPOS
	3	Server Displays committee student DPOS
Open Issues		

Number	16	
Name	Comment on Committee Student DPOS	
Summary	Users are able to leave comments on an Advisory Committee student's DPOS submission.	
Priority	3	
Preconditions	Committee Student DPOS must be submitted.	
Postconditions	A comment will be left on the DPOS submission.	
Primary Actor	Student, Graduate Coordinator, Advisory Committee	
Secondary Actors	Server	
Trigger	Server receives request to write a comment on the Student DPOS	
Main Scenario	**Step**	**Action**
	1	Server receives request to write a comment on the Student DPOS
	2	Server locates committee student DPOS
	3	Server opens a window for the user to write in.
	4	User submits their comment to the server.
	5	Server updates the DPOS with the user's comment
	6	Server notifies the DPOS' owner of the placed comment
Extensions	**Step**	**Branching Action**
	2a	The server fails to locate the DPOS in question**:** The use case ends and an error message is displayed to the user.
	3a	The window is left blank**:** The use case ends and no further action is taken
	5a	Update is unsuccessful**:** Return -1
	5b	Update is successful**:** Return 1

	6a	Value from 5 was -1**:** Display to the user that the save was unsuccessful
	6b	Value from 5 was 1**:** Display to the user that the save was successful
Open Issues		

Number	17	
Name	Provide Approval Status on Student DPOS	
Summary	An advisory member alerts the server of their approval status on a given student's DPOS.	
Priority	5	
Preconditions	Students' DPOS must be submitted for approval.	
Postconditions	Approval Status displayed.	
Primary Actor	Graduate Coordinator/Major Advisor	
Secondary Actors	Server	
Trigger	An advisory member or the graduate coordinator selects either the "Changes Requested - No Approval" button or the "Approve DPOS" on a given DPOS.	
Main Scenario	**Step**	**Action**
	1	The system receives the update to either approve a DPOS or request changes.
	2	Server locates committee student DPOS
	3	The system sends the update to the server to attach that higher level user's approval or disapproval to a DPOS (Use Case #12 triggered)
	4	The system displays to the user that their approval status has been sent to the server.
Extensions	**Step**	**Branching Action**
	2a	The server fails to locate committee student DPOS: Use case ends and system displays a message of failure to locate DPOS
Open Issues		

Number	18	
Name	Make Changes to DPOS	
Summary	User Makes Changes to DPOS	
Priority	5	
Preconditions	Students DPOS must be submitted.	
Postconditions	Requests for changes to DPOS will be made	
Primary Actor	Graduate Coordinator, Major Advisor	
Secondary Actors	Server	
Trigger	User selects "Makes changes to student DPOS"	
Main Scenario	**Step**	**Action**

	1	The system triggers a modified use case of use case #7
	2	The higher level user makes modifications to the DPOS form.
	3	The higher level user submits modifications of the DPOS form to the server.
	4	The server receives the modifications to the DPOS and triggers use case #12 (Alert Server of Changes)
	5	The advisor/graduate coordinator receives confirmation that their changes were successfully passed to the server.
Open Issues		