

Milestone 4 Report

The Diversity Research Symposium

Conducted by: Nathan Peterson, Cody LeFan-Weaver, Turner Daugherty, and Yibo Xiao



LEARN | INTERACT | NETWORK | SHARE IDEAS

Founded at Ball State University in 2009

Table of Contents

(Click a Title to go straight to content)

Overview:	7
Group Members:	7
The Client:	7
Links:	7
Summary:	7
1. Completed Works	9
a. Feasibility Report	9
i. Our Client:	9
ii. The Scope of Our Project:	9
iii. Risk Analysis:	10
iv. Software Development Process:	10
b. Requirements Analysis:	11
i. Functional Requirements	11
ii. Non-functional Requirements	11
iii. Technical Requirements	12
Fulfilled/Unfulfilled Requirements	12
Fulfilled Requirements:	12
Almost Completed Requirements:	12
Unfulfilled Requirements:	12
Dependency_Matrix	13
Business Considerations	13
System Design	13
2. DRS Use Cases:	14
Use Case 1: Register for Symposium	14
Use Case 2: Browse Presentations	14
Use Case 3: Create Account	15
Use Case 4: Submit Presentation	16
Use Case 5: Add Payment	17
Use Case 6: Admin Mass-Email	18
Use Case 7: Review Preference Selection	19
Use Case 8: Review Progress Tracking	20

Use Case Diagram:	21
3 .UI Design	22
i. Registration	22
ii. Home Page	24
Alternate to Home page Slide 2:	26
Alternate to Home Page Slide 3:	26
iii. About Page	27
V. Login Modal	29
VI. User Profile	29
VII. Reviewer Portal -Approved	30
VIII. Reviewer Portal - Pending	30
IX. Reviewer Portal - Rejected	31
X. Admin Portal - Overview	31
XI. Admin Portal - Reviewer Applications	32
XII. Proposal Application	32
XIII. Reviewer Application	33
4. Testing	33
i. UI Testing	33
ii. Verification Testing	34
Example of Karma Automated test for Angular2:	35
iii. Unit Testing	36
iv. Specific Unit Testing on Component	38
A.Home Component	38
B. Main Nav Bar	39
C. About Component	42
D. Event Component	42
E. Login Component	43
F. Register Component	43
E. Proposal Application Component	44
G. Reviewer Application	46
H. User Profile Component	47
I. Reviewer Portal - Pending	47
J. Reviewer Portal - Approved	49
K. Reviewer Portal Rejected	50
L. Admin Portal - Reviewer Applications	51
L. Admin Portal - Overview	52
5. Documentation	53
a. Flow Charts:	53

i. Main Model Flowchart:	54
ii. Registration Model Flowchart:	54
iii. Admin Model Flowchart:	55
iv. User Model Flowchart:	56
v. General User Model Flowchart:	56
vi. Viewer Model Flowchart:	57
vii. Presenter Model Flowchart:	57
b. Architecture:	58
Module Component:	58
Components:	59
Templates:	59
Metadata:	60
Data binding:	60
Directives:	60
Services:	61
Dependency Injection:	61
C. Database:	62
Database Design:	63
6. Dependencies	66
FireBase	66
Angular 2	66
AngularFire2:	66
FirebaseCLI:	66
AngularCLI:	66
7. Transfer Rights	67
8. Gantt Chart	67
9. Work Breakdown	68
10. File Structure	69
File Tree:	69
11. Source Code	75
Developer Manual	83
1.Install the appropriate developer environment	83
2. Getting the project on a new computer	85
First way: No command line	85
Second way: Command line Requires Git Bash if on Windows	86
3. Install the project. This will require git bash from now on.	89

4. Running the Project	89
Updating the project	90
Angular2 documentation:	90
Angularfire2 Documentation	91
Firebase Documentation:	91
Uploading the project to Firebase for live hosting:	91
Other useful developer programs we used:	91
Lorem Ipsum:	91
Trello:	91

Overview:

Group Members:

Nathan Peterson, napeterson@bsu.edu , nalexpeter@gmail.com

Cody LeFan-Weaver, crlefanweave@bsu.edu

Turner Daugherty, tjdaugherty@bsu.edu

Yibo Xiao, yxiao4@bsu.edu

The Client:

The Diversity Research Symposium, contacted through Dr. Linh Littleford and David Largent.

Dr. Linh Littleford

lnlittleford@bsu.edu

765-285-1707

Mr. David Largent

dllargent@bsu.edu

765-285-8641

Links:

Live Site: <https://drs-db.firebaseio.com/home>

Source Code: <https://github.com/NathanPeterson/DRS-Webapp>

Database: <https://console.firebase.google.com/project/drs-db/overview>

User Manual: Bottom of PDF

Summary:

The Diversity Research Symposium's representatives, Dr. Littleford and Professor Largent, have wanted to expand the symposium to reach more people. They desired a new website to draw attention and make signing up a simple process. The Diversity Research Symposium presents people with the opportunity to discuss important, diversity-related topics, with presenters invited to visit the campus selected for the year and talk to attendees who have paid to be admitted.

The goal of this project is to create a system that can be used by any of the hosts for the DRS, as the symposiums rotate amongst schools. User-friendliness in this case is equal parts being inviting for potential attendees and for the representatives of the DRS that would need to make use of the program. We've focused our energy on creating a system to allow users to make accounts and for proposals to be sent and then reviewed, as those two aspects were the most important based on our analysis.

1. Completed Works

In this section, we'll be taking a look at the completed documentation for our two semesters while working on this project for the DRS. It has been an interesting challenge for us, for a lot of different reasons, but I believe that we have all had the opportunity to learn and grow from it.

a. Feasibility Report

i. Our Client:

The Diversity Research Symposium's representatives, Dr. Littleford and Professor Largent, have wanted to expand the symposium to reach more people. They desired a new website to draw attention and make signing up a simple process. The Diversity Research Symposium presents people with the opportunity to discuss important, diversity-related topics, with presenters invited to visit the campus selected for the year and talk to attendees who have paid to be admitted.

ii. The Scope of Our Project:

In accordance with the requirements analysis done for this project, the primary goal of our project is to give the DRS representatives, attendees, presenters, and reviewers a portal through which to manage their DRS-related activity, including submitting proposals for presentations to be reviewed, being able to track review progress for the presentations, and the ability for users to have profiles that will assist the DRS representatives with preparing for upcoming symposiums based on demographic information of anticipated attendees.

The goal of this project is to create a system that can be used by any of the hosts for the DRS, as the symposiums rotate amongst schools. User-friendliness in this case is equal parts being inviting for potential attendees and for the representatives of the DRS that would need to make use of the program. We've focused our energy on creating a system to allow users to make accounts and for proposals to be sent and then reviewed, as those two aspects were the most important based on our analysis.

iii. Risk Analysis:

1. Changing/Incomplete Requirements: As the project develops, we may encounter changing requirements from our client based on new information they discover about what they will need for the project to be considered a success.
 - a. Fix: We'll be keeping regularly with the clients, at least every three weeks, to discuss the project and get feedback before we move forward.
2. Developer Skill: We're working in areas that not all of us are deeply familiar with (whether that's databases or web development), meaning that we may encounter a problem that is difficult to solve and interferes with our schedule.
 - a. Fix: By having a clear idea of what we need to do, we can delegate responsibilities to people with the appropriate experience and when there is no such person, we will allot extra time to account for the research that will be necessary.
3. Technical/Non-functional Requirements: Not all of these requirements are clear as of yet; the client has a general idea of what they want in some areas and a more concrete idea of what they want in others. This presents a challenge for us as developers, as we try to meet the requirements as we understand them.
 - a. Fix: When we meet with the client, it's important for us to describe the requirements as we understand them in order to get clarification. Since we will be delivering functional code with each iteration, that will help us to catch and resolve any misunderstandings that we may have.

iv. Software Development Process:

The project will be using practices from SCRUM, this is because the current requirements are not set in stone and will most likely change as we progress through the iterations. Therefore, using short time frame iterations would be the most beneficial. Using scrum allows us to output a new prototype after each iteration to show the client. This allows the client and team the following benefits.

1. Change in requirements – The client will be able to see a current build of the project thus far. This allows the client to know how the product is looking currently. And change any requirements that he/she wants.
2. Prototyping – Create modified versions of the current build and be able to show client. If the client likes/dislikes it, then it will be easy to merge or delete it.
3. Same schedule – The client and team members know the deadline of the current iteration and can prepare accordingly. If a user story is foreseen to not be finished than we can drop it to work on a feature that will be completed.
4. Interactive – Scrum allows us to have more of an interaction with the client. Therefore, it is like the client is part of our team; this will increase the overall happiness of the client.

Following will be our requirements analysis as created when we began the project.

b. Requirements Analysis:

1. Web Interface
 - a. Administrators will have the ability to modify other user accounts and send out messages to groups of users based on predefined settings.
 - b. Users will be able to mark their intent to attend symposiums, pay for their attendance, submit proposals for presentations, and apply to review presentations.
2. Database Management System
 - a. Will contain information about users, including a unique user ID.
 - b. Will be used to track demographic information and presentations.
3. Easily Modifiable by Administrators and Future Developers
 - a. The framework will account for changing leadership of the DRS on a regular basis.
 - b. We will make it simple for future developers to work with the database.

i. Functional Requirements

1. The administrators will be able to send e-mails to users based on predefined settings.
2. The administrators will be able to control what permissions other users have.
3. The administrators will be able to manage the content on the website.
4. Users will be able to control the privacy information for their user accounts.
5. Reviewers will be able to select presentation topic preferences that will inform reviewing assignments.
6. Reviewers will be able to manage the presentations they're tasked with reviewing, while users with presentations under review can track the progress.
7. Reviewers will be able to choose which topics presenters will be able to present.
8. Presenters will be able to track the reviewers reviewing progress.
9. Users will be able to browse previous presentations.
10. Users will be able to pre-register for symposiums.
11. Log in system.

Optional:

1. Integrate social media.
2. A system for users to submit feedback about presentations to the presenters.
3. Tracks number of users by type and the numbers of downloads of presentations hosted on the website.

ii. Non-functional Requirements

iii. Technical Requirements

1. Server
 - a. The project requires a stable server to host the website, accounting for anticipated traffic.
2. Database
 - a. A database of users and presentations is necessary, including personal information and payment information.
3. Web
 - a. The interface for users will be through a website. There needs to be a method to browse the information, update information, and also to host/download files from the website. Ideally we will have a method of payment set up through the website. There will be a forum set up for user-to-user communication.

Fulfilled/Unfulfilled Requirements

Fulfilled Requirements:

- The administrators will be able to send e-mails to users based on predefined settings.
- The administrators will be able to control what permissions other users have.
- Reviewers will be able to select presentation topic preferences that will inform reviewing assignments.
- Reviewers will be able to manage the presentations they're tasked with reviewing, while users with presentations under review can track the progress.
- Reviewers will be able to choose which topics presenters will be able to present
- Presenters will be able to track the reviewers reviewing progress.
- Users will be able to pre-register for symposiums.
- Log in system.
- Server (Firebase 3)
- Database(Firebase 3)
- Web

Almost Completed Requirements:

- The administrators will be able to manage the content on the website.
- Users will be able to browse previous presentations.

Unfulfilled Requirements:

- Users will be able to control the privacy information for their user accounts.

Dependency_Matrix

Functional Requirements Description:	Requirement #	Status	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	Total	UC = Use Cases
The administrators will be able to send emails to users based on predefined settings	F1	Completed						x			1	
The administrators will be able to control what permissions other users have.	F2	Completed			x						1	
The administrators will be able to manage the content on the website.	F3	Completed		x	x						2	
Users will be able to control the privacy information for their user accounts.	F4	Completed			x						1	
Reviewers will be able to select presentation topic preferences that will inform reviewing assignment	F5	In-Progress		x							1	
Reviewers will be able to manage the presentations they're tasked with reviewing, while users with presentations under review can track the progress.	F6	In-Progress		x		x					2	
Reviewers will be able to choose which topics presenters will be able to present.	F7	Completed		x							1	
Presenters will be able to track the reviewers reviewing progress.	F8	In-Progress				x				x	2	
Users will be able to browse previous presentations.	F9	Completed		x					x		2	
Users will be able to pre-register for symposiums.	F10	Completed	x		x						2	
Log in system.	F11	Completed	x		x		x				3	
Technical Requirements Description:												
Server: The project requires a stable server to host the website, accounting for anticipated traffic.	T1	Completed	x	x	x	x	x	x	x	x	all	
Database: A database of users and presentations is necessary, including personal information and	T2	Completed	x	x	x	x	x	x	x	x	all	
Web: The interface for users will be through a website.	T3	Completed	x	x	x	x	x	x	x	x	all	

Business Considerations

The DRS is an organization which charges an attendance fee for their symposiums and it has been our goal to make it simpler for them to manage payments of users. Unfortunately due to circumstances we were unable to implement a payment system in the current build of the project. This was due to current support for Angular2 and time restraints to design our own work around.

The user base we expect to encounter are academics interested in the idea and concept of “diversity” in the many forms it may take, from any number of different fields. The presenters will be knowledgeable persons with a passion for diversity, the reviewers likewise and perhaps even moreso as they would be volunteering their time to review possible proposals. The attendees will be drawn from the pools of presenters and reviewers as well as those who are simply interested in learning more or getting to see a particular presenter that has piqued their interest.

System Design

Our design is a two-way binding between a web application built in Angular 2 and a database using Firebase. Users of all types interact with the database through the website, where their user accounts are stored, proposals and review progress is stored, and admins are able to alter other user information such as permissions. Following are the use cases that we set up to work from for this project.

2. DRS Use Cases:

- Use Case 1: Register for Symposium

Primary Actor: Viewer

Other Actors:

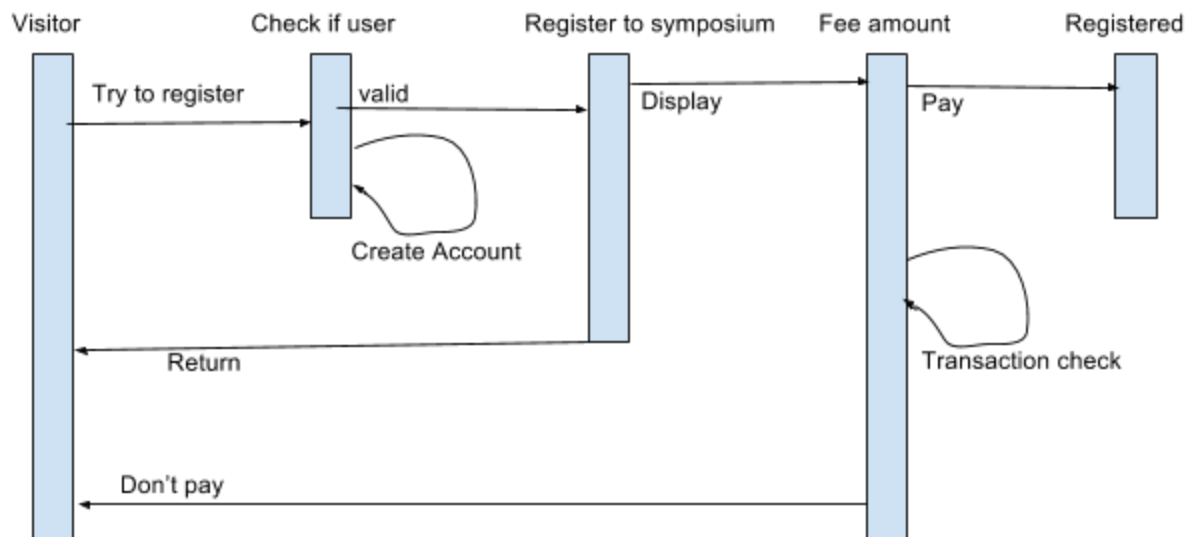
Preconditions: User has signed in the account.

Main Flow:

1. User with an account can choose to register for a symposium.
2. System displays fee for attendance.
3. User decides to pay and register.
4. System registers the user and delivers receipt to the user.

Alternate Flows:

- 1a. User chooses not to register for symposium.
- 3a. forego registration.
- 4a. User does not register therefore takes them to main page.



- Use Case 2: Browse Presentations

Primary Actor: Viewer

Other Actors:

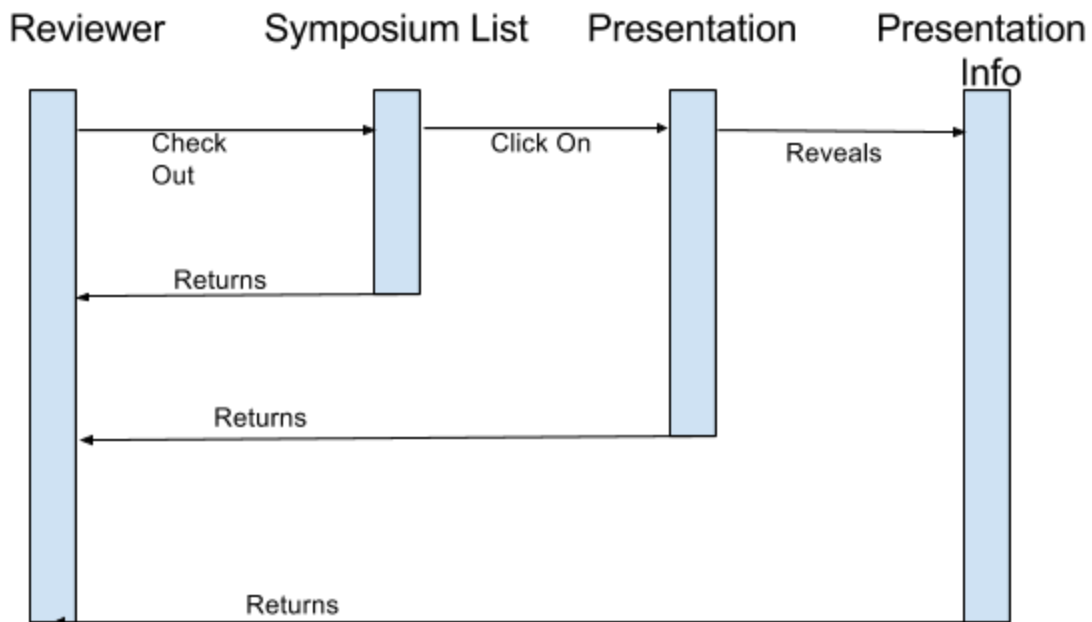
Preconditions: The website has been displayed.

Main Flow:

1. User can choose to look at presentations for upcoming and previous symposiums.
2. User can click on a listed presentation to get more information about it and the presenter.
3. User can navigate away from the presentation selection menu through the navigation bar.

Alternate Flows:

- 1a. User remains on current page.
- 2a. User misclicks and information is not displayed.
- 3a. User stays on page and continues to look at presentations.



● Use Case 3: Create Account

Primary Actor: Guest

Other Actors:

Preconditions:

Main Flow:

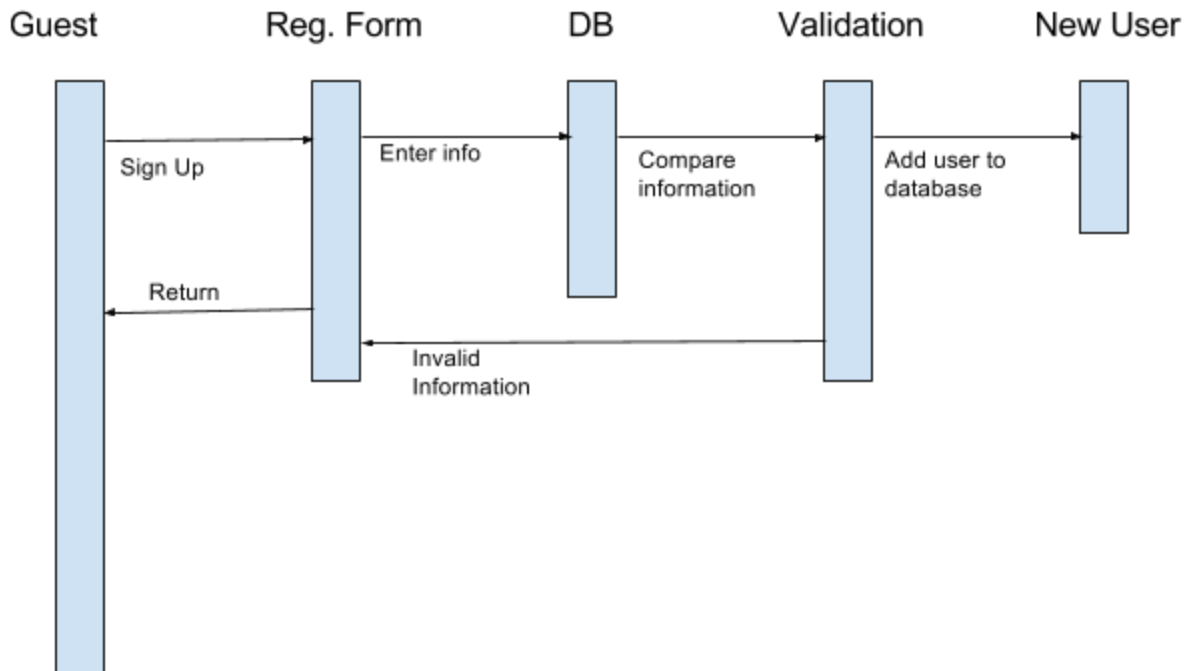
1. User can choose to make an account.
2. User can provide personal information.
3. User can decide what on their account is visible to other users.
4. User successfully makes account.

Alternate Flows:

- 1a. User chooses not to make account remains guest.
- 2a. User leaves personal information blank and makes anonymous account.

3a. User skips step and the default is set to private.

4a. User can exit account creation without creating an account.



● Use Case 4: Submit Presentation

Primary Actor: Presenter

Other Actors:

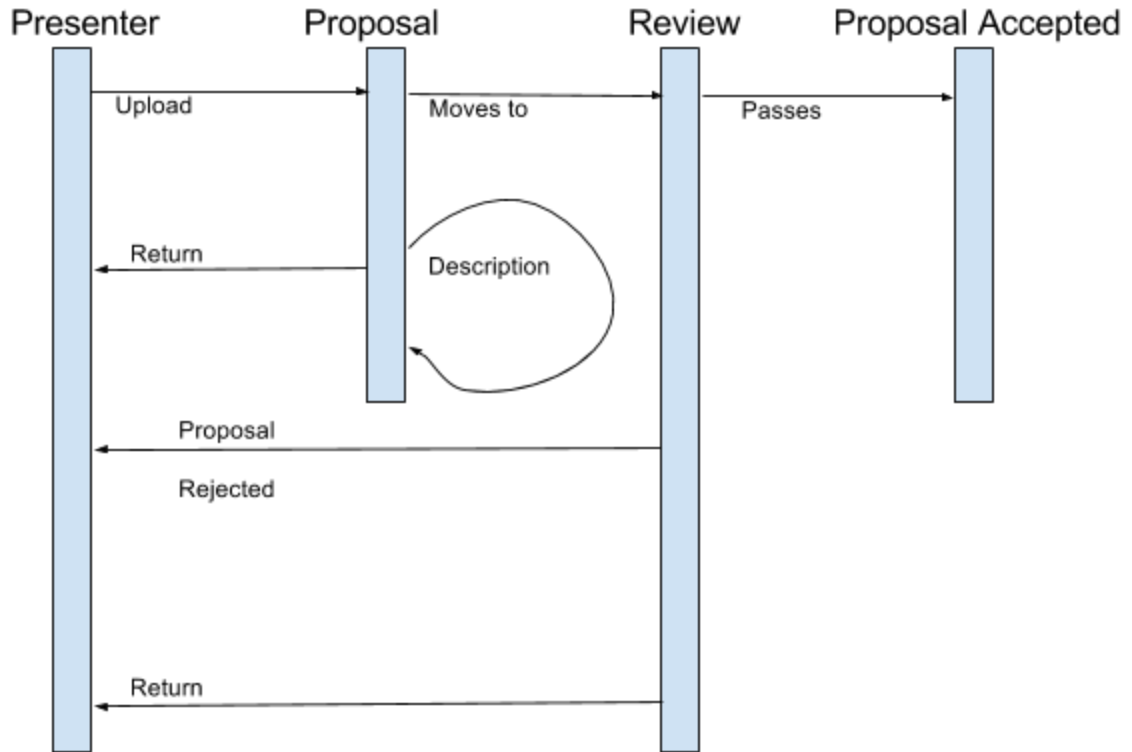
Preconditions: User has signed in the account.

Main Flow:

1. User with account can upload a proposal for DRS presentation.
2. User can submit a proposal for a DRS presentation with a description of the presentation to be reviewed.
3. User can choose to cancel the submission process.

Alternate Flows:

- 1a. User does not upload a proposal and remains a registered user.
- 2a. User does not add a description prior and continues to proceed through the process.
- 3a. Submission goes through a review process.



● Use Case 5: Add Payment

Primary Actor: Viewer

Other Actors:

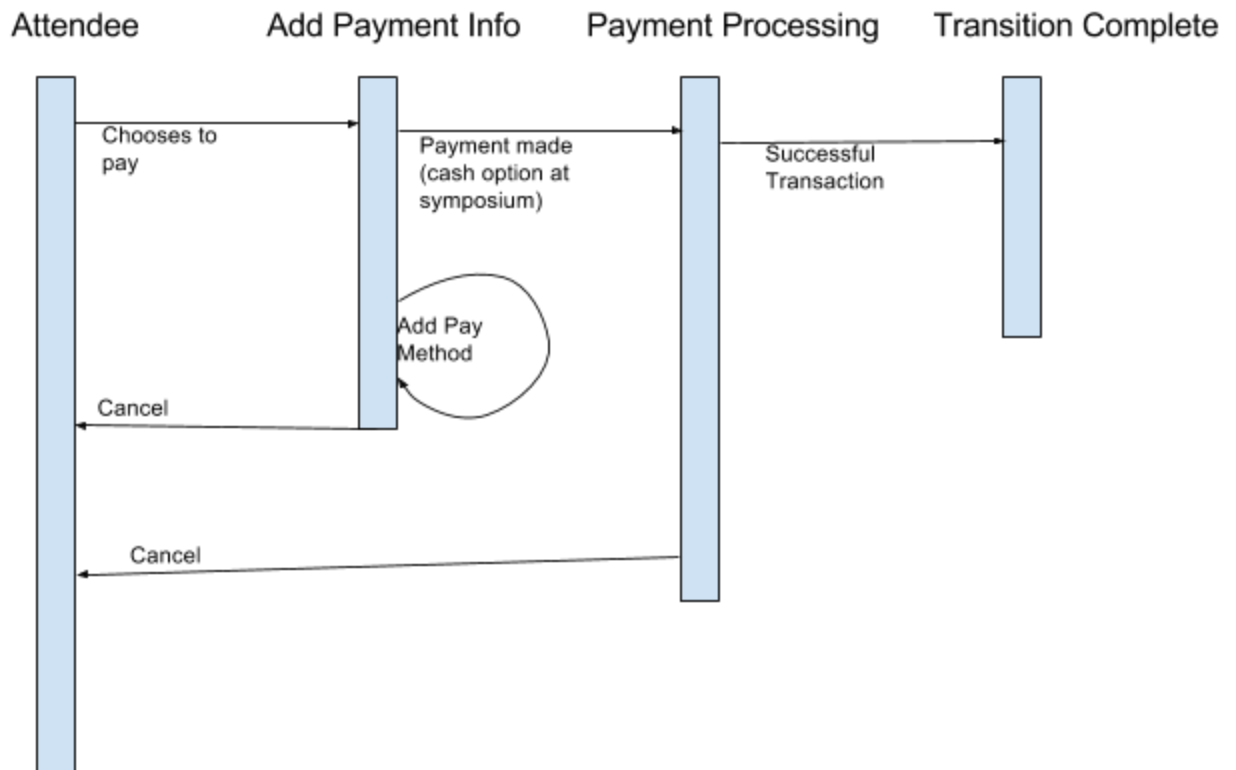
Preconditions: User has signed in the account.

Main Flow:

1. User with an account can add a paypal account or debit card information with adding necessary information.
2. User can add more than one payment.
3. User can cancel this process.

Alternate Flows:

- 1a. User does not have a paypal account/ debit card and pays with cash.
- 2a. User decides not to make payment and leaves page.
- 3a. User proceeds through transactions.



● Use Case 6: Admin Mass-Email

Primary Actor: Administrator

Other Actors:

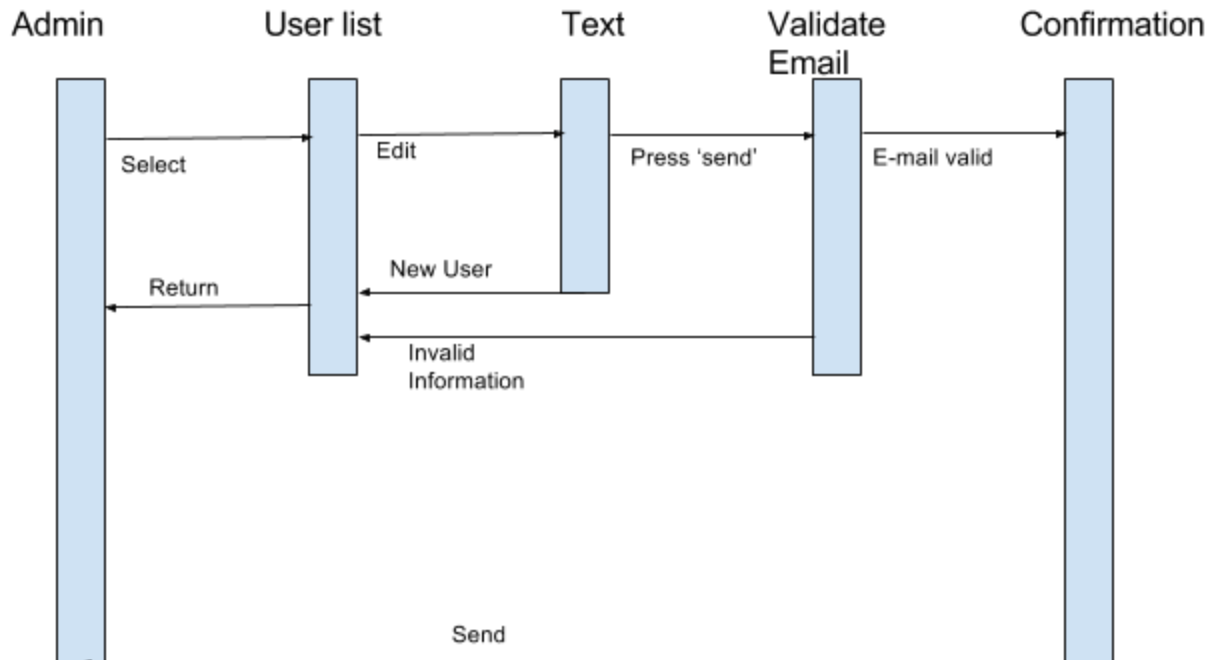
Preconditions: User has signed in the account

Main Flow:

1. User with admin account can select from a list of settings to define which users to send an e-mail to.
2. Admin can provide text body for e-mail to send to users with valid e-mail addresses.
3. Admin can edit text body from web page, use button to submit and have e-mails sent automatically.
4. Admin will receive confirmation that e-mails were sent correctly.

Alternate Flows:

- 1a. Admin has no users and waits for more users to register.
- 2a. A default generic body will be in place.
- 3a. Emails sent manually.
- 4a. Error report if emails failed to send.



● Use Case 7: Review Preference Selection

Primary Actor: Reviewer

Other Actors: Administrator, Viewer

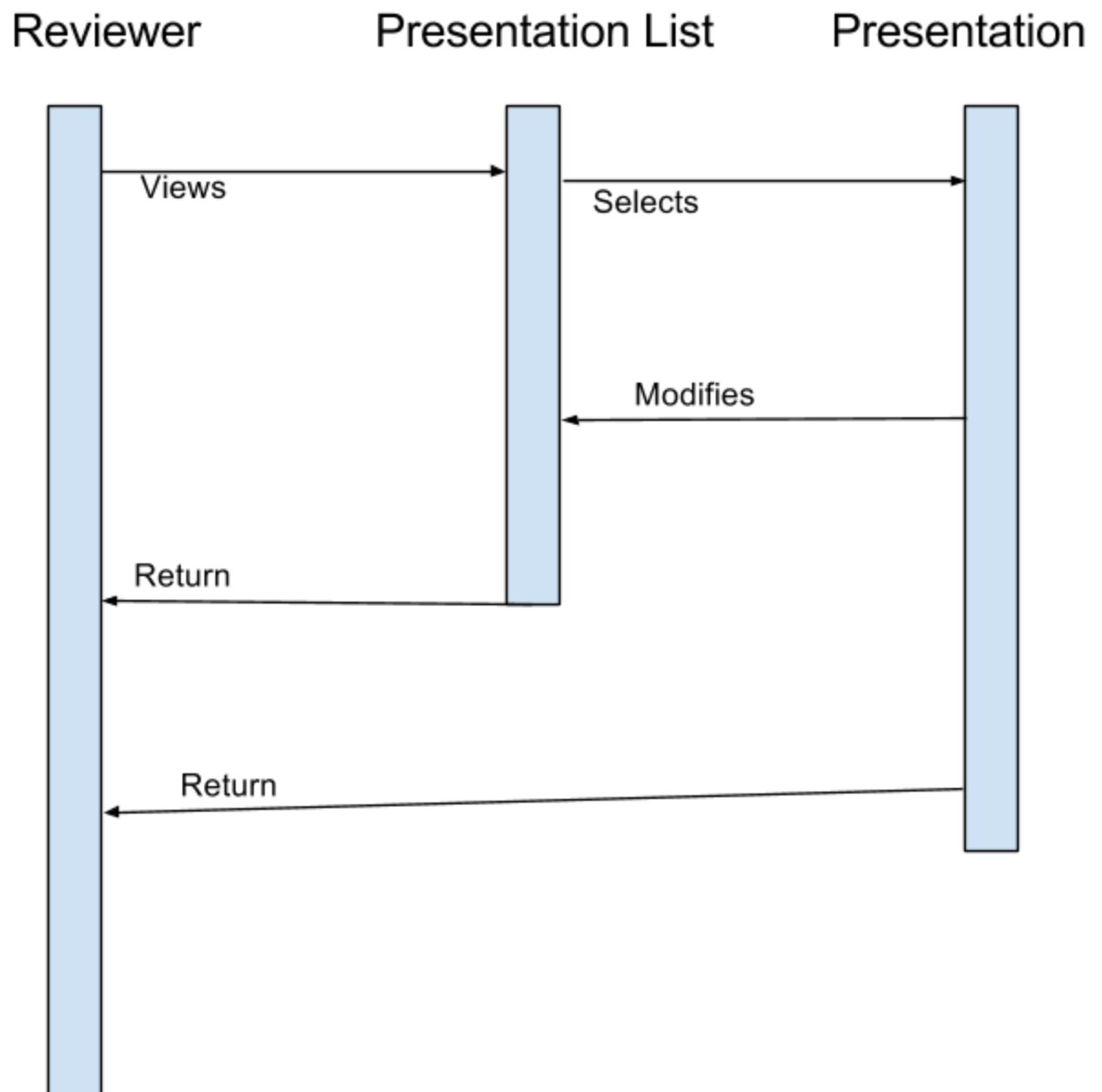
Preconditions: Viewer has been approved by an Administrator as Reviewer

Main Flow:

1. Reviewer sees brief summary of the list of presentations.
2. Reviewer selects presentations based upon interest.
3. Reviewer can alter selections at a later time.

Alternate Flows:

- 1a. Reviewer chooses to change their preferences.
- 2a. Reviewer can deselect current preferences.
- 3a. Reviewer can select new preferences.



- Use Case 8: Review Progress Tracking

Primary Actor: Presenter

Other Actors: Reviewer

Preconditions: Presenter has submitted a presentation.

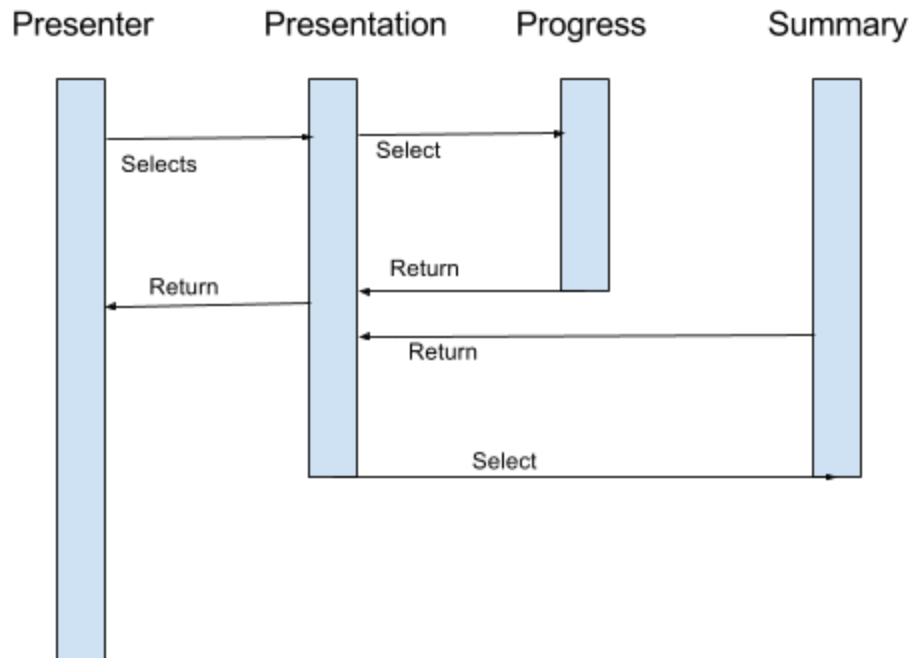
Main Flow:

1. Presenter chooses to check the progress of a submitted presentation.
2. Presenter can navigate to page of submitted presentations.
3. Presenter can see summary of presentation's review progress.

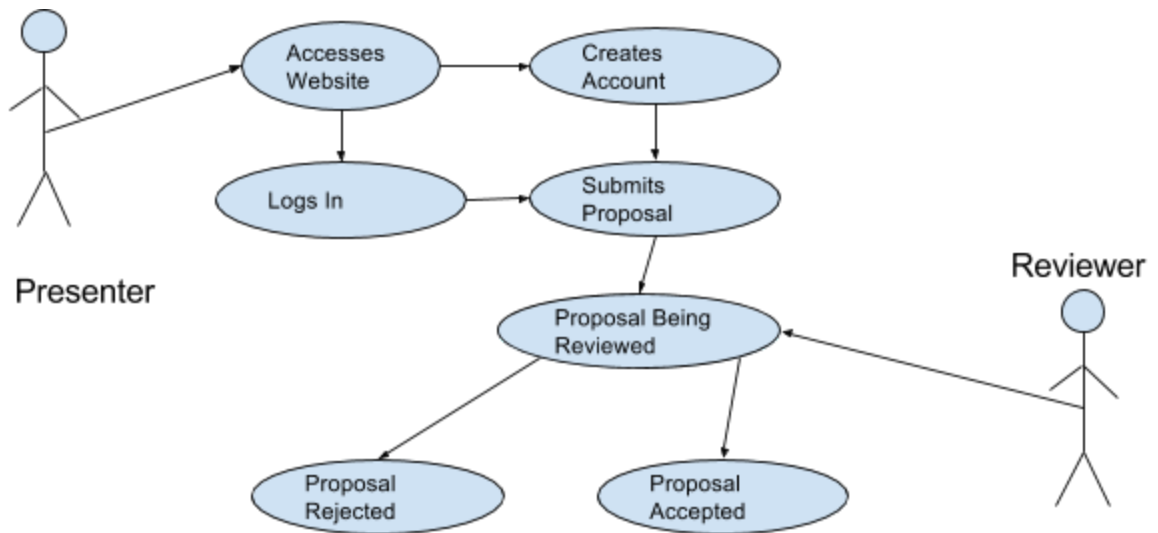
Alternate Flows:

- 1a. Progress has changed / progress has not changed

- 2a. Presenter chooses not to check submitted presentations
- 3a. Progress has changed / progress has not changed



Use Case Diagram:



DIVERSITY RESEARCH SYMPOSIUM

LEARN | INTERACT | NETWORK | SHARE IDEAS



[Home](#)

[About](#)

[Contact ▾](#)

[Event](#)

[Forums](#)

[Reviewer Portal](#)

[Admin Portal](#)

[criefanweave@bsu.edu \(Logout\)](#)

Register an Account

Required Info

Email

Email

Password

Password

Username

Username

Then there is personal information the user can provide to give the DRS the ability to better accommodate that user.

User Information

First Name

Middle Initial

Last Name

Address Information

Address

City

State

Zipcode

School Information

University

Department

Discipline

Subdiscipline

Additional Information

Phone Number

Do you need special considerations? Yes ☐ No ☐

Do you have any Dietary Restriction? Yes ☐ No ☐

li. Home Page

Our home page is a simple design, here's a look at the page from an active user where we can see options to submit a proposal or become a reviewer:

DIVERSITY RESEARCH SYMPOSIUM

LEARN | INTERACT | NETWORK | SHARE IDEAS



Home

About

Contact ▾

Event

Forums

Reviewer Portal

Admin Portal

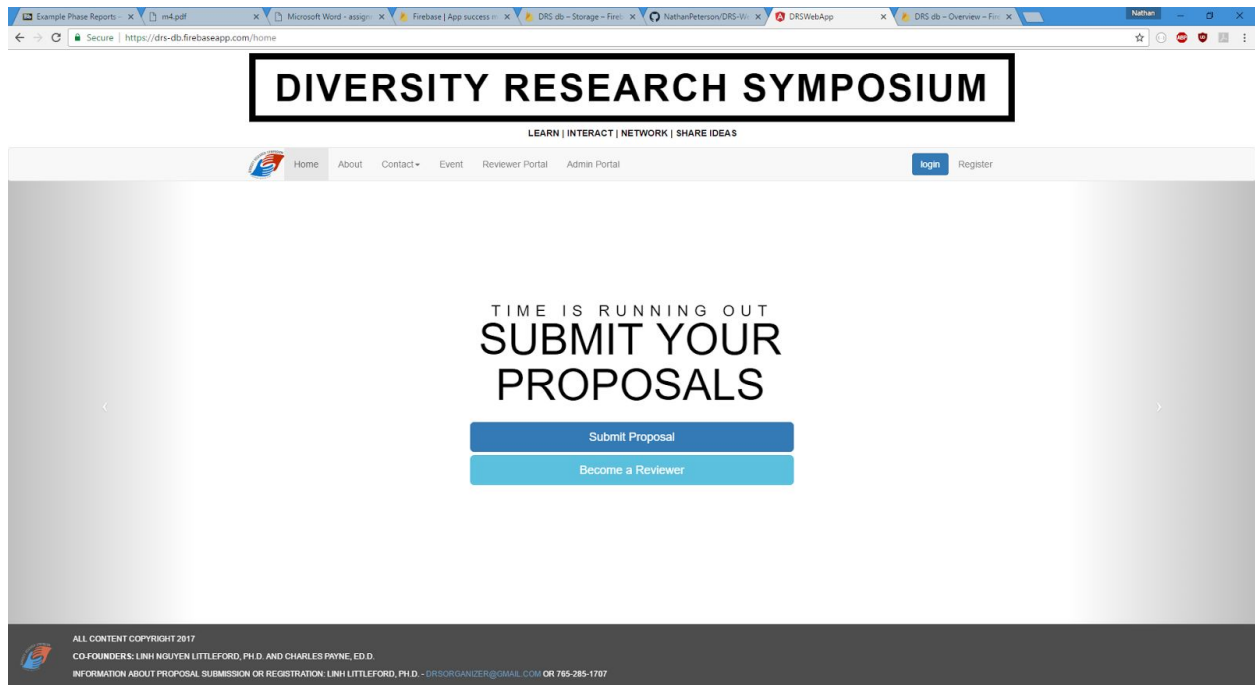
crlefanweave@bsu.edu (Logout)

TIME IS RUNNING OUT
SUBMIT YOUR
PROPOSALS

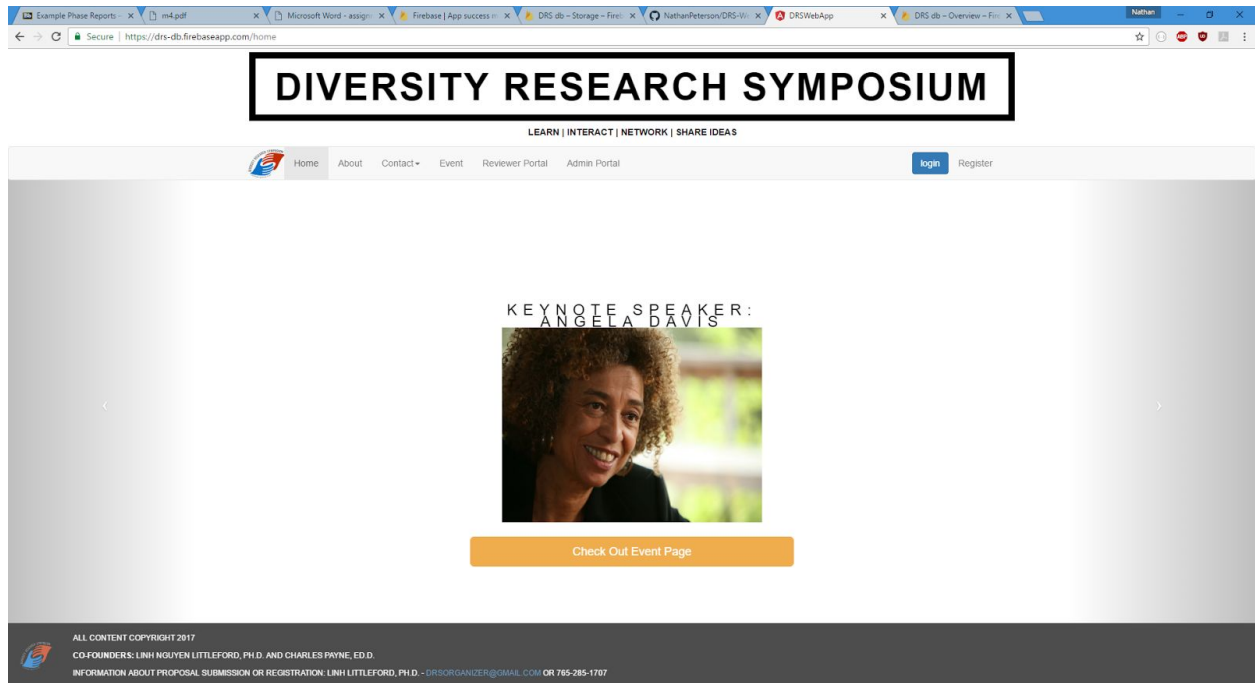
Submit Proposal

Become a Reviewer

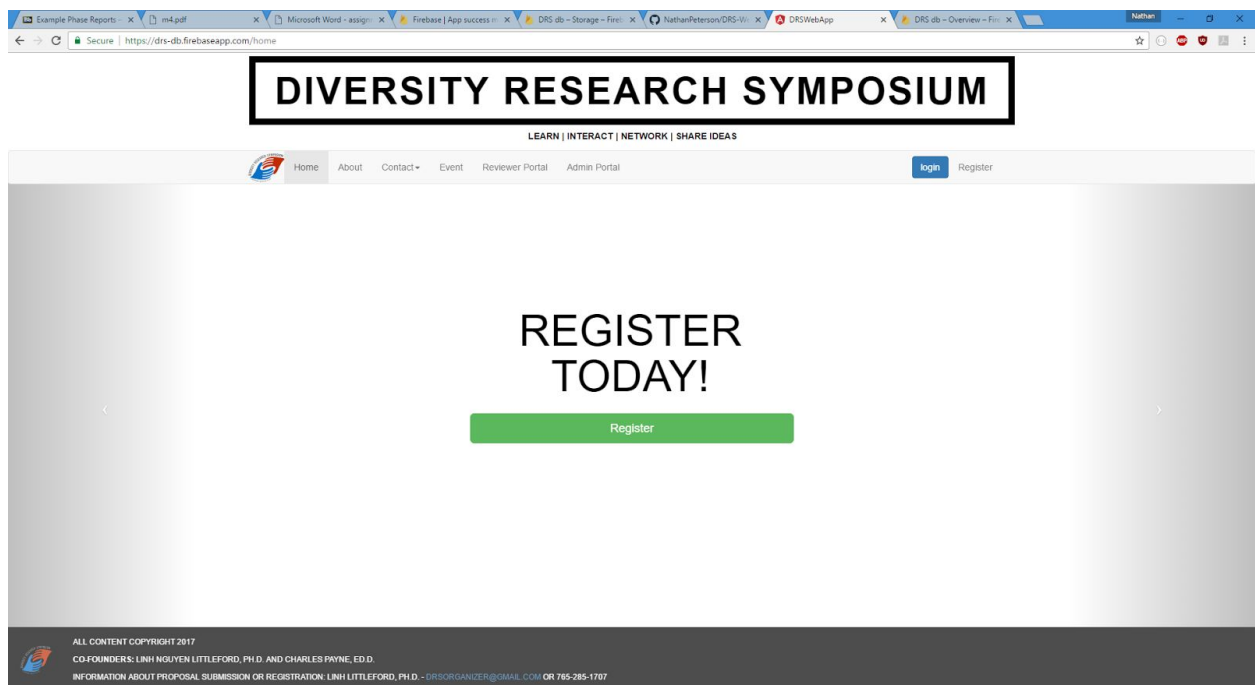
The same page for a user who is not logged in would redirect the user to registration/login if they tried to select proposal submission or becoming a reviewer and also features buttons for both features (login and register):



Alternate to Home page Slide 2:

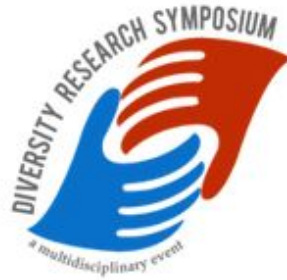


Alternate to Home Page Slide 3:



lii. About Page

On our about page, we prominently display information about the DRS:



LEARN | INTERACT | NETWORK | SHARE IDEAS

Founded at Ball State University in 2009

About Us

The Diversity Research Symposium (DRS) was co-founded in 2009 by Linh Nguyen Littleford (Associate Professor in the Department of Psychological Science) and Charles R. Payne (former Assistant Provost for Diversity, former Director of the Office of Institutional Diversity, and Professor of Secondary Education), both at Ball State University. The DRS aims to:

- provide an educational environment in which faculty, staff, community members, and students from all disciplines who are interested in cultural diversity issues can learn, interact, share ideas, and network with one another.
- encourage members of academic institutions to infuse cultural diversity issues into their research, curricula, and professional development.

Organizing and hosting responsibilities are rotated every year among three universities (Ball State University, Indiana State University, and Indiana University Southeast). In selecting the symposium's theme, keynote speakers, and activities, the organizers highlight the diversity-related values and objectives at their respective institutions while achieving the goals of the DRS.

The 2017 DRS will be our 9th annual conference and the fourth one hosted by Ball State University.

Ball State University's Office of the Vice President for Academic Affairs and the Office of Institutional Diversity provided financial support to host the 2009, 2011, and 2014 DRS. We are grateful to Dr. Terry King, Provost and Vice President for Academic Affairs, for his continued encouragement and support of the DRS.

Timeline

Iv. Event Page

Our event page has a large section for the key note speaker, with modifiable fields for other presenters at that symposium in a slideshow below:

Keynote Speaker: Angela Davis



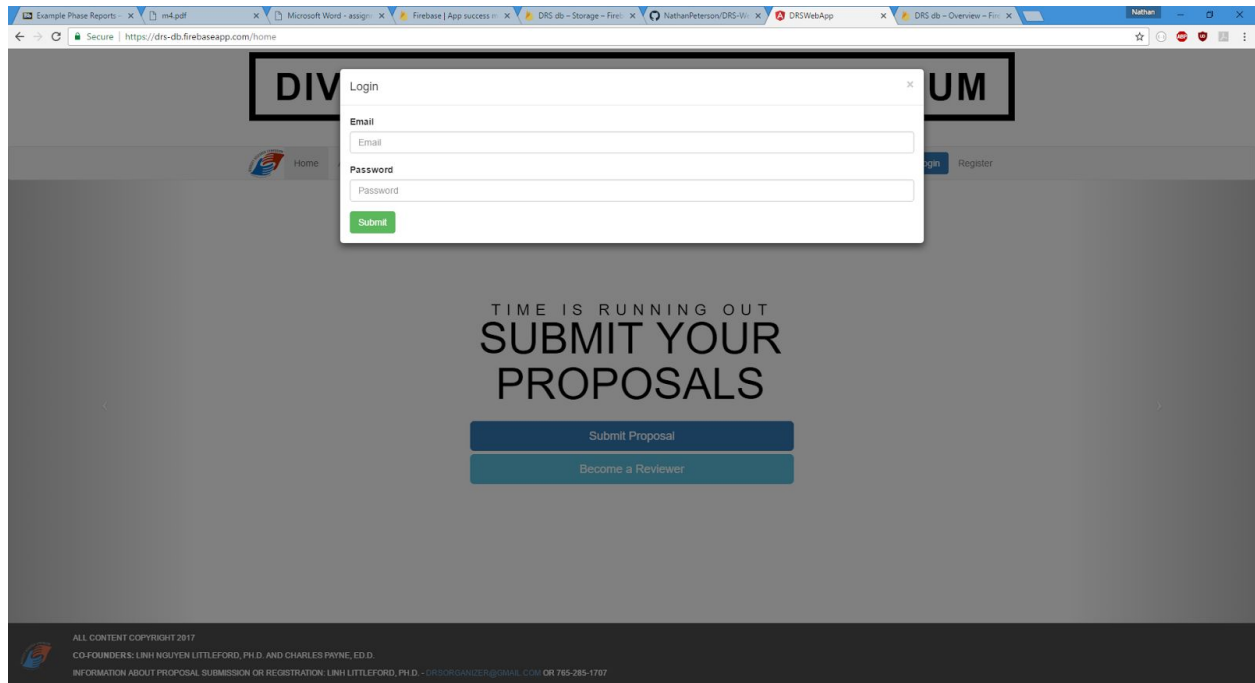
Third slide label

Praesent commodo cursus magna, vel scelerisque nisl consectetur.

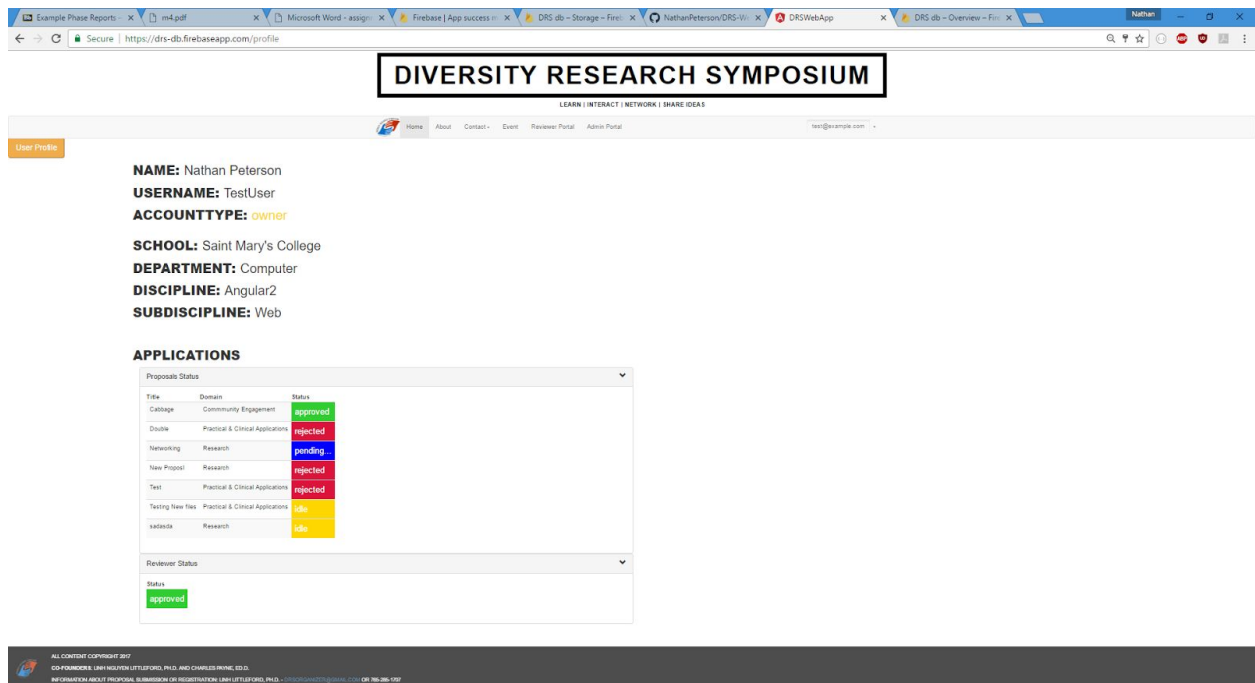


The menu is present for each of these, but in order to give a clearer image of the important parts of the page it has been cropped out. Navigation is, of course, accessible on each page.

V. Login Modal



VI. User Profile



VII. Reviewer Portal -Approved

The screenshot shows a web browser window with the URL <https://drs-db.firebaseio.com/reviewer-portal/proposal-approved>. The page features a large header "DIVERSITY RESEARCH SYMPOSIUM" with the tagline "LEARN | INTERACT | NETWORK | SHARE IDEAS". Below the header is a navigation bar with links: Home, About, Contact, Event, Reviewer Portal, and Admin Portal. The user is logged in as "test@example.com".

On the left, a sidebar menu shows "Pages" (Reviewer Portal Home) and "Proposals" (Approved 2, Pending 1, Rejected 1). The main content area displays two proposal cards:

- Cabbage** by Nathan A Peterson. It includes a link to "DRS2017 Proposal Submission Form.pdf" and buttons for "Open File", "Download Files", and "Reset Prop".
- My First Proposal** by Dasda A Asdas. It includes a link to "DRS2017 Proposal Submission Form.pdf" and buttons for "Open File", "Download Files", and "Reset Prop".

The footer contains copyright information: "ALL CONTENT COPYRIGHT 2017. CO-FOUNDERS: LINH NGUYEN LITTLEFORD, PH.D. AND CHARLES PAYNE, ED.D. INFORMATION ABOUT PROPOSAL SUBMISSION OR REGISTRATION: LINH LITTLEFORD, PH.D. - DRSGORGANDER@GMAIL.COM OR 765-285-1707".

VIII. Reviewer Portal - Pending

The screenshot shows a web browser window with the URL <https://drs-db.firebaseio.com/reviewer-portal/proposal-pending>. The page features a large header "DIVERSITY RESEARCH SYMPOSIUM" with the tagline "LEARN | INTERACT | NETWORK | SHARE IDEAS". Below the header is a navigation bar with links: Home, About, Contact, Event, Reviewer Portal, and Admin Portal. The user is logged in as "test@example.com".

On the left, a sidebar menu shows "Pages" (Reviewer Portal Home) and "Proposals" (Approved 2, Pending 1, Rejected 1). The main content area displays six proposal cards in a 2x3 grid:

- Demo Proposal** by Class Demo. It includes a link to "DRS2017 Proposal Submission Form.pdf" and buttons for "Open File", "Download Files", "Reject", "Reset Prop", and "Approve". The status is "0 / 0".
- Networking** by Nathan A Peterson. It includes a link to "project 2 networking.pdf" and buttons for "Open File", "Download Files", "Reject", "Reset Prop", and "Approve". The status is "0 / 0".
- Testing New files** by Nathan A Peterson. It includes a link to "DRS2017 Proposal Submission Form.pdf" and buttons for "Open File", "Download Files", "Reject", "Reset Prop", and "Approve". The status is "0 / 0".
- gqerqe** by Dslakjh A Asdkjh. It includes a link to "hwk2.pdf" and buttons for "Open File", "Download Files", "Reject", "Reset Prop", and "Approve". The status is "0 / 0".
- njerunfiw** by Asdas A Akjasd. It includes a link to "hwk2.pdf" and buttons for "Open File", "Download Files", "Reject", "Reset Prop", and "Approve". The status is "0 / 0".
- sadasda** by Nathan A Peterson. It includes a link to "hwk2.pdf" and buttons for "Open File", "Download Files", "Reject", "Reset Prop", and "Approve". The status is "0 / 0".

The footer contains copyright information: "ALL CONTENT COPYRIGHT 2017. CO-FOUNDERS: LINH NGUYEN LITTLEFORD, PH.D. AND CHARLES PAYNE, ED.D. INFORMATION ABOUT PROPOSAL SUBMISSION OR REGISTRATION: LINH LITTLEFORD, PH.D. - DRSGORGANDER@GMAIL.COM OR 765-285-1707".

IX. Reviewer Portal - Rejected

The screenshot shows a web browser window with the URL <https://drs-db.firebaseio.com/reviewer-portal/proposal-rejected>. The page features a header with the title "DIVERSITY RESEARCH SYMPOSIUM" and a navigation bar with links: "LEARN | INTERACT | NETWORK | SHARE IDEAS". Below the header, there is a sidebar on the left with a "Pages" section containing "Reviewer Portal Home" and a "Proposals" section with "Approved", "Pending", and "Rejected" links. The main content area displays three proposal cards, each titled "Double", "New Proposl", and "Test", all attributed to "Nathan A Peterson". Each card has buttons for "Open File", "Download Files", and "Reset Prop". The footer contains copyright information for 2017 and contact details for the co-founders.

X. Admin Portal - Overview

The screenshot shows a web browser window with the URL <https://drs-db.firebaseio.com/admin-portal/admin-overview>. The page features a header with the title "DIVERSITY RESEARCH SYMPOSIUM" and a navigation bar with links: "LEARN | INTERACT | NETWORK | SHARE IDEAS". Below the header, there is a sidebar on the left with a "Admin Portal Home" section containing "Overview" and "Reviewer Apps" links. The main content area displays a "Dashboard" with six statistics: "11 Users", "3 Reviewer Applications", "2/30 Reviewers", "11 Proposal Applications", "4 Admin", and "\$96 Earned". Below the dashboard, there is a table titled "Database Users - 11 in table" with columns: "#", "Username", "Email", "UID", "Proposal App", "Reviewer App", "Status", "Promote", "Demote", "Paid?", and "Delete?". The table contains 11 rows of user data.

#	Username	Email	UID	Proposal App	Reviewer App	Status	Promote	Demote	Paid?	Delete?
1	spence	test1@example.com	0htLkWXcNUYMFcS4weDmyg3JV1B2	true	false	admin	⬆	⬇	false	🗑
2	A	a@a.com	20x6wwjoAdZBRrXb4uAK1uxbYTF2	false	false	standard	⬆	⬇	false	🗑
3	asdaklajshd	a@d.com	9yH5KLc3XaS4udxPIUpYpAmQgc2	false	false	standard	⬆	⬇	false	🗑
4	ClassDemo	class@demo.com	B4alozhE41esh3Z23gQijelB7aZ2	false	false	standard	⬆	⬇	false	🗑
5	TestUser	test@example.com	C2YmLpFXvfiFveGHXQrevjrwG3	true	true	owner	⬆	⬇	false	
6	abc	a@c.com	OvfECWbhOKY3gHvUxnOO2IS92Em1	false	false	standard	⬆	⬇	false	🗑
7	FazeMaGee	testMe@me.com	TeRspysvVJa8Uild8zBthPyGh4H3	true	true	reviewer	⬆	⬇	false	🗑
8	Testing default	testing@testing.com	VX86czXa8cNxlOp9vaeT2r9pJDt1	false	false	reviewer	⬆	⬇	false	🗑
9	paradoos	paradoos233@gmail.com	YkOEew3ETbP8bzm9YvU2m1DawG1	false	false	admin	⬆	⬇	false	🗑

XI. Admin Portal - Reviewer Applications

The screenshot shows the 'Reviewer Applications Manager' interface. At the top, there's a header with the site name 'DIVERSITY RESEARCH SYMPOSIUM' and navigation links: 'LEARN | INTERACT | NETWORK | SHARE IDEAS'. Below this is a sidebar with 'Admin Portal Home', 'Overview', and 'Reviewer Apps'. The main content area displays three application cards. Each card shows the applicant's name, full name, and answers to three questions. At the bottom of each card are three buttons: 'Reject' (red), 'Pending...' (blue), and 'Approve' (green).

- CodyBobJim**
Full Name: Cody LeFan-Weaver
Answer To Question 1: ergerg
Answer To Question 2: erggege
Answer To Question 3: qrgqergvegrwe
- FazeMaGee**
Full Name: Faze MaGee
Answer To Question 1: Ut auctor aliquam justo nec ultrices. Quisque nec dolor quis leo aliquet dignissim ac et ligula. Fusce enim est, accumsan in semper quis, gravida eu ex. Aliquam erat volutpat. Ut rhoncus ante ligula, pharetra pellentesque mi faucibus et. Donec ornare nunc nibh, et scelerisque nulla ullamcorper nec. Donec tempus fermentum eros eu fringilla. Praesent est lectus, suscipit quis orci at, faucibus convallis sapien. Cras mattis leo a nisi hendrerit faucibus. Vivamus id nunc eu risus faucibus tristique non sed mi. Nunc nisi ex, tristique eget du at, porta euismod augue. Ut elementum neque et mauris commodo molestie. Cras gravida sollicitudin consequat.
- TestUser**
Full Name: Nathan Peterson
Answer To Question 1: tasd
Answer To Question 2: asdas
Answer To Question 3: asdasda

XII. Proposal Application

The screenshot shows the 'Proposal Application' form. At the top, there's a header with the site name 'DIVERSITY RESEARCH SYMPOSIUM' and navigation links: 'LEARN | INTERACT | NETWORK | SHARE IDEAS'. Below this is a sidebar with 'Computer', 'Discipline', 'Subdiscipline', and 'Web'. The main content area is titled 'Proposal Information' and contains a form with fields for 'Title', 'Domain' (with a dropdown menu), and 'Select files' (with a 'Drop your files here' area). At the bottom, there's an 'Upload Files' section with a button to 'Upload files to Firebase' and a 'Clear files' button. The footer contains copyright information and contact details.

Proposal Information

- Title
- Domain:
- Select files

Upload Files

ALL CONTENT COPYRIGHT 2017
CO-FOUNDERS: LINH NGUYEN LITTLEFORD, PH.D. AND CHARLES PAYNE, ED.D.
INFORMATION ABOUT PROPOSAL SUBMISSION OR REGISTRATION: LINH.LITTLEFORD, PH.D. - DRSORGANIZER@GMAIL.COM OR 765-285-1707

XIII. Reviewer Application

The screenshot shows a web browser window with the URL <https://drs-db.firebaseio.com/review-app>. The page title is "DIVERSITY RESEARCH SYMPOSIUM" with the tagline "LEARN | INTERACT | NETWORK | SHARE IDEAS". Below the title is a navigation bar with links: Home, About, Contact, Event, Reviewer Portal, Admin Portal, and a user email "test@example.com".

The main content area is titled "Register to be a Reviewer" and includes a "Cancel Application" button. The form is divided into three colored sections:

- User Information (Pink background):** Fields for First Name (filled with "Nathan"), Middle Initial (filled with "A"), and Last Name (filled with "Peterson").
- School Information (Orange background):** Fields for University (dropdown menu showing "Saint Mary's College"), Department, Computer, Telephone, and Email (all empty).
- Additional Information (Green background):** Field for Phone Number (filled with "5550279").

Below the form is the "Reviewer Application" section with the following questions and options:

- What is the highest degree you have obtained? (Degree type)
- Why would you like to be a reviewer? (Explain why you would want)
- How will your background help an individual presenters that match our theme? (Explain how your background)
- Select which topics you would be interested in reviewing:
 - ☐ Education (Pedagogy & Curriculum)
 - ☐ Research
 - ☐ Community Engagement
 - ☐ Student Engagement
 - ☐ Professional Development & Mentoring
 - ☐ Cooperative Medical, Mental Health
 - ☐ Practical & Clinical Applications
 - ☐ Business
 - ☐ Arts & Entertainment

At the bottom, there is a "Register as Reviewer" button and a footer with copyright information for 2017 and contact details for the organizers.

4. Testing

i. UI Testing

For our testing of the user interface, we took a look at how users could interact with the website through a variety of browsers and devices. This was done to help us figure out what improvements could be made to the styling and programming of the website, with some additional help from impartial users to help determine the friendliness of the website's design.

Black Box Testing initially provided some inconsistent results with resizing of screens, where on smaller devices the formatting would stop functioning as intended, but on medium and larger devices everything functioned as intended. Contact is a broken link.

Test										
File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive										
Comments Share										
fx										
	A	B	C	D	E	F	G	H	I	J
1	Win 10 (2560 * 1440)	home	about	contact	event	registration	reviewer portal	admin portal	Reviewer Application	Proposal Application
2	chrome			x						
3	edge			x						
4	firefox			x						
5	IE11			x						
6										
7	Win 10 (1600 * 900)	home	about	contact	event		reviewer portal	admin portal		
8	chrome			x						
9	edge			x						
10	firefox			x						
11	IE11			x						
12										
13	macOS Sierra (1440 * 900)									
14	chrome			x						
15	safari			x						
16										

ii. Verification Testing

One of the biggest parts of the project was making sure that the database and website were interacting correctly. As we discussed in our last presentation, there is 2-way data-binding between the website and database which enables some live updates for the users.

FireBase is a cloud service which can introduce difficulties in end-to-end testing, but unit testing is as simple as mocking the FireBase client library. Karma is where we did a lot of our testing, although it was apparent that there were some issues with the injections done in the code. The end-to-end testing was done using the full, actual system to ensure that all parts were working together after unit testing to make sure that the methods should be functioning as intended.



With a live site, we were able to step through the user account creation process, tracking that the user was successfully built in the database. Then we moved on to making sure that files submitted for proposals would a) be of the correct file type and b) appear in the database properly.

Up next was making sure that the application process for becoming a reviewer worked smoothly. The user would first submit their application, an admin would have to review the application, and then their status as a reviewer could be updated.

Another very important part of the project for the client was that presenters be able to track the review progress for their proposals. This can be done in real time from the user's profile, where they will see the proposal submissions and the status of those proposals.

Example of Karma Automated test for Angular2:

Checking if Home component is working correctly. Most components have a very similar format to the test, as they are generated automatically when we generate components on the command line.

```
/* tslint:disable:no-unused-variable */
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { By } from '@angular/platform-browser';
import { DebugElement } from '@angular/core';

import { HomeComponent } from './home.component';

describe('HomeComponent', () => {
  let component: HomeComponent;
  let fixture: ComponentFixture<HomeComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ HomeComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(HomeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

iii. Unit Testing

Testing involving the database proved a little more difficult than we anticipated, just due to a lack of documentation for Angular2 and Firebase working in concert. Our primary concern, of course, has been making sure that we accurately retrieve data from the database which does not use the SQL most of us have become accustomed to. This information consists of user accounts and proposals for presentations, two of the most important pieces of our project. We have successful testing of the two-way data binding between the database and web application, including real time account creation/verification and proposal submissions via in-site uploads. Our redirection process also works smoothly, whether through the regular nav bar or the hamburger menu. Redirecting is also working as intended when a user tries to use a URL to visit a page they don't have permission for (redirecting to login, for instance, when someone tries to submit an application despite not being logged in).

Test Case	Requirement	Results
1.1	Establish connection to Firebase from localhost web application.	Test is successful, we are able to update entries in the database as well as query the database.
1.2	Establish connection to Firebase from live web application.	Test is successful, we are able to update entries in the database as well as query the database.
1.3	Modify existing data from web application.	Test of administrator portal is successful, user accounts and applications can be altered by an admin in-site.
2.1	Navigate using standard navigation bar.	Test of the navigation bar is successful, redirects work as intended.
2.2	Navigate using hamburger menu.	Menu overlaps images, but test of navigation is successful.
2.3	Redirect when user does not have correct permissions to view page.	Test is successful when trying to directly navigate to the admin portal via URL despite not being logged in and not being a member, unable to submit applications without first logging in.

Test Case	Requirement	Results
3.1	User submission of proposal.	Proposals are successfully uploaded to the database.
3.2	Reviewer may access proposal.	Reviewers can access proposals from their profiles and approve or deny them, download them, or view them in-browser.
3.3	Admin can access proposals.	Admins can see proposals through their portal, can reset their review progress successfully.
3.4	User tracking of proposal progress.	Users can see where their proposals are at in real time from their profiles (tested on multiple screens) thanks to the two-way binding.
4.1	User can apply to be a reviewer.	The applications are successfully submitted to the database as pdf files.
4.2	Admin can review applications for reviewers.	The admin portal is successfully tracking the number of reviewers and the number of active reviewer applications, users can be made into reviewers through the portal.

iv. Specific Unit Testing on Component

Statistics: 12 failed, 148 passed, 92.5% success rate

A.Home Component

Test ID	Test Case	Condition	Expected Results	P / F
1.a	Slide Transition	Wait 15 seconds	Slide Changes to next slide	P
1.b	Manual Slide Transition Right	Click on Right Arrow	Slide moves forward 1 slide	P
1.c	Manual Slide Transition Left	Click on left arrow	Slide moves back 1 slide	P
1.d	Slideshow reaches end of slides to right	Wait 15 seconds or click right arrow	Goes to first slide	P
1.e	Slideshow reaches end of slides to left	Wait 15 seconds or click left arrow	Goes to last slide	P
2.a	Slide 1: click Submit proposal Button	Click button	Redirects to Proposal Page	P
2.b	Slide 1: click Become a Reviewer Button	Click button	Redirects to Reviewer Application Page	P
2.c	Slide 2: Click Check out Event Page Button	Click Button	Redirects to Event Page	P
2.d	Slide 3: Click Register Button	Click Button	Redirects to Register Page	P
3	Email DRS	Click email link in footer	Open up Mail service to email DRS team	F

B. Main Nav Bar

Test ID	Test Case	Condition	Expected Results	P / F
4	Go Home	Click Home link on nav bar	Redirect to home Page	P
4.a	Go Home as User	Click Home link on nav bar as a User	Redirects to Home Page	P
5	Go About	Click on About link on nav bar	Redirects to About Page	P
5.a	Go About as User	Click on About link on nav bar as a User	Redirects to About Page	P
6	Go to contact Page	Click on Contact link on nav bar	Redirects to Contact Page	F
6.a	Go to contact Page as a User	Click on Contact link on nav bar as a User	Redirects to Contact Page	F
7	Go to Event Page	Click on Event link on nav bar	Redirect to Event Page	P
7.a	Go to Event Page as User	Click on Event link on nav bar as User	Redirect to Event Page	P
8.	Default Reviewer Portal Link	Not logged in	Not Showing	P

8.a	Reviewer Portal Link as Standard User	Logged in as Standard User	Not Showing	P
8.b	Reviewer Portal Link as Reviewer	Logged in as Reviewer	Showing Link	P
8.c	Reviewer Portal Link as Admin/Owner	Logged in as Admin/Owner	Showing Link	P
8.d	Click Reviewer Portal	If link exist Click Reviewer Portal Button	Redirect To Reviewer Portal	P
9.	Default Admin Portal Link	Not logged in	Not Showing	P
9.a	Admin Portal Link as Standard User	Logged in as Standard User	Not Showing	P
9.b	Admin Portal Link as Reviewer	Logged in as Reviewer	Not Showing	P
9.c	AdminPortal Link as Admin/Owner	Logged in as Admin/Owner	Showing Link	P
9.d	Click Admin Portal	If link exist Click Admin Portal Button	Redirect To Admin Portal	P
10.	Not logged in	Not logged in	Displays login and register button in nav bar	P

10.a	Logged in	Logged in	Login button and Register button removed and replaced with user name button	P
11	Login	Click login button	Open up a login modal	P
12.	Go to register	Click on Register button	Redirect to Register Page	P
13	User button exist	User logs in	Login button and Register button removed and replaced with user name button	P
13.a	User button has user email on it	Current user email on button as text	If logged in successfully, display user email on button	P
13.b	Click user button	Click user button	Redirects to user's profile page	P
13.c	Click user arrow button	Click the down arrow beside user button	Display 2 links, Profile and Logout	P
13.d	Click Profile link	Click profile link	Redirects user to their profile Page	P
14	Log out	Click logout link in down arrow drop down	Logs current user out, and redirects them to home page	P
14.a	User button turns back to login	User logs out	User button changes back to login and register buttons	P

14.b	Reviewer logs out and removes special links	Reviewer logs out	Nav bar removes Reviewer portal from nav bar	F
15.c	Admin logs out and removes special links	Admin logs out	Nav bar removes Reviewer Portal and Admin Portal Links	F
15.d	Owner logs out and removes special links	Owner logs out	Nav bar removes Reviewer Portal and Admin Portal Links	F
15.e	Anyone logs out and refreshes page	User logs out and refreshes page	Nav bar resets to default, removing special links and starts on slide 1.	P

C. About Component

Test ID	Test Case	Condition	Expected Results	P / F
16.	Display Big logo	On redirect to about	Display big DRS logo	P
17	Display About us blurb	On redirect to about	Display About us blurb	P
18.	Display Time line	On redirect to about	Display Timeline	P
18.a	Display Event information for year	On redirect to about	Display event information for year	P

D. Event Component

Test ID	Test Case	Condition	Expected Results	P / F
19	Display Keynote speaker	On redirect to event	Display Keynote speaker name	P
20	Display Keynote picture	On redirect to event	Display Keynote picture	P
21.	Slide Transition	Wait 15 seconds	Slide Changes to next slide	P

22	Manual Slide Transition Right	Click on Right Arrow	Slide moves forward 1 slide	P
23	Manual Slide Transition Left	Click on left arrow	Slide moves back 1 slide	P
24	Slideshow reaches end of slides to right	Wait 15 seconds or click right arrow	Goes to first slide	P
25	Slideshow reaches end of slides to left	Wait 15 seconds or click left arrow	Goes to last slide	P

E. Login Component

Test ID	Test Case	Condition	Expected Results	P / F
26	Correct information	Enter correct existing information	User logs in, and gets redirected to home page	P
27.	Blank	Enter nothing and hit submit	Error alert saying form needs to be filled in	P
28.	Email doesn't exist	Enter a wrong email	Error alert saying email is not in system	P
29	Wrong password	Enter wrong password	Error alert saying the password does not match password for that user	P
30	Close with 'x'	User clicks 'x' in top right	Closes login modal	P
31.	Close by clicking outside	User clicks outside modal	Closes login modal	P

F. Register Component

Test ID	Test Case	Condition	Expected Results	P / F
32	Successful user creation	User enters in correct information in all fields and hits register	User gets created and redirects to home page	P
32.a	Just required info	User enters correct required info and hits register	User gets created and redirects to home page	P
32.b	Required info plus	User enters in correct	User gets created and redirects to	P

	a few extra	info in required and a few other fields	home page	
33.	User leaves required info blank	User leaves required info blank	User does not get created and an alert pops up telling them to fill out required info	P
34	All 50 states	Check if all 50 states are in drop down	All 50 states in abbreviated form are there	P
35	University drop down has universities	Check if university has information	Should display all universities in Indiana and one for out of state	P
36.	Email is in correct format	User enters in email	Checks to see if its in the correct email format	P
36.a	Email is in incorrect format	User enters in incorrect email format	Alert saying this field needs to have the correct format	P
37.	Correct password length	User enters in correct password length	Nothing	P
37.a	Incorrect password length	User enters to short of required password length	Alert asking them to make their password longer	P

E. Proposal Application Component

Test ID	Test Case	Condition	Expected Results	P / F
38.	Load Component	On redirect to proposal application	Load component into view and display instructions	P
39.	Download Form	Click download proposal form	Downloads pdf to desktop	P
40.a	Make proposal not logged in	Click make proposal Today button	Redirects to login page	P
40.b	Make proposal logged in as registered user	Click make proposal Today button	Changes html to display proposal form	P
41	Cancel proposal form	Click cancel	Changes html back to instructions	P
42.a	Load user info	loading form	Information user provided in	P

			registration is automatically displayed in proposal form	
42.b	Load user info partial	Loading form	Information user provided in registration is automatically displayed in proposal form	P
43	Successful application	User fills out all information	Files get pushed to storage and successfully added to database	P
44	Failed Application	User leaves field blank	Error alert telling user to fill in specified field	P
45	Disable upload button	User didnt provide file	Upload button is disabled	P
46.a	User drops pdf	User drops pdf file in box	File gets added to list below upload button	P
46.b	User drops text file	User drops .txt file in box	File gets added to list below upload button	P
46.c	User drops image file	User drops .png/.img/.jpg/etc file	File gets added to list below upload button	P
46.d	User drops word doc	User drops .doc / .docx file	Nothing happens	P
46.e	User drops movie file	User drops movie file	Nothing happens	P
46.f	USer drops audio file	User drops .mp4/.mp3/etc	Nothing happens	P
46.g	User drops multiple approved files	User drops a mixture of image and pdf files or txt file	Files are added to list below upload file in order of processed	P
46.h	User drops multiple not allowed files	User drops audio/movie/word documents	Nothing happens	P
46.i	User drops mixture of approved and not allowed files	User drops a comination of approved and not allowed files	Files that are allowed get placed in list below button Files not allowed do nothing	P
47	Active upload button	User dropped approved files	Upload button becomes active	P

48.	Upload files	User hits upload	Each file with display a progress bar that will fill up as it gets pushed to storage, afterwards nothing	P
49.a	Clear files with no files	User hits clear button	Nothing happens	P
49.b	Clear files with files listed below	User hits clear button	All files listed are removed from list and will not be uploaded	P

G. Reviewer Application

Test ID	Test Case	Condition	Expected Results	P / F
50.	Load Component	On redirect to reviewer application	Load component into view and display instructions	P
51.a	Become reviewer not logged in	Click become reviewer button	Redirects to login page	P
51.b	Become reviewer logged in as registered user	Click Become reviewer button	Changes html to display reviewer application form	P
52	Cancel proposal form	Click cancel	Changes html back to instructions	P
52.a	Load user info	loading form	Information user provided in registration is automatically displayed in reviewer application form	P
52.b	Load user info partial	Loading form	Information user provided in registration is automatically displayed in reviewer application form	P
53	Successful application	User fills out all information	Application successfully added to database	P
54	Failed Application	User leaves field blank	Error alert telling user to fill in specified field	P
55	Multiple checkmarks	User clicks multiple interest	All checked interested are noted as true in database, all none checked interested are denoted as false	P

H. User Profile Component

Test ID	Test Case	Condition	Expected Results	P / F
56	Load user profile	Click User profile	Loads information about user to page	P
57	View Proposal applications	Clicks on Proposal Status drop down	Loads a table containing information about all of the current users proposal applications	P
58.a	Check status of Proposal	New proposal	Status should say idle, with yellow background	P
58.b	Check status of Proposal	Proposal is being reviewed	Status should say pending... and have a blue background	P
58.c	Check status of Proposal	Proposal was accepted	Status should say Approved and background should be green	P
58.d	Check status of Proposal	Proposal was denied	Status should say Rejected and background should be red	P
59	View Reviewer application status	Click on Reviewer status drop down	Displays the status of the current users reviewer application	P
60.a	Check status of Reviewer application	New application	Status should say pending... and have a blue background	P
60.b	Check status of Reviewer application	Reviewer application was accepted	Status should say Approved and background should be green	P
60.c	Check status of Reviewer application	Reviewer application was denied	Status should say Rejected and background should be red	P

I. Reviewer Portal - Pending

Test ID	Test Case	Condition	Expected Results	P / F
61	Load Pending Proposals	On redirect to Reviewer portal Pending	Display all proposals Pending	P
62	PDF Button	If proposal has a pdf file create a button for it	A button for each pdf should be displayed in the middle of the proposal	P

63	PDF button pressed	Click PDF button	Open modal that displays pdf on screen	P
64.a	Close PDF modal	Click x in top right corner	Close pdf modal	P
64.b	Close PDF modal	Click outside the modal	Close PDF modal	P
65	Counter starts at 0 / 0	Counter is 0 / 0	Displays red 0 / green 0 in proposal	P
66.a	Action buttons Reviewer	Current User is Reviewer	Display 2 action buttons, reject and Approve	P
66.b	Action buttons Admin/Owner	Current user is Admin/owner	Display 3 action buttons reject, reset, approve	P
67.a	Action buttons enabled	Current user is not the same as proposal uid	Action buttons are enabled	P
67.b	Action buttons disabled	Current user is the same as the proposals uid	Action buttons are disabled	P
68.a	Click Reject - enabled	Click reject button	Increase red counter by 1	P
68.b	Click Reject - disabled	Click reject button	Nothing	P
68.c	Click Reject Twice - enabled	Click reject button	Nothing	F
69.a	Click Approve - enabled	Click approve button	Increase red counter by 1	P
69.b	Click Approve - disabled	Click approve button	Nothing	P
69.c	Click Approve Twice - enabled	Click approve button	Nothing	F
70	Click Reset	Click reset button as admin	Set counter back to 0 /0 And inscrease reset counter by 1	P
71	Click Reset Multiple times	Click reset button as admin	Set counter back to 0 /0 And inscrease reset counter by 1 each time button was clicked	P
72	Click open file	Click open file button	Display all files submitted with application	P

73	Click on drop down file from open file	Click on a file	Open file in new tab	P
74	Click download file	Click download file button	Display all files submitted with application	P
75	Click on drop down file from download file	Click on a file	Downloads selected file to computer	P
76	Blue border	Check Proposal style	Proposal style is blue	P
77	Red reaches 3	Red counter = 3	Remove proposal and place it into rejected list	P
78	Green Reaches 3	Green counter = 3	Remove proposal and place into Approved list	P

J. Reviewer Portal - Approved

Test ID	Test Case	Condition	Expected Results	P / F
79	Green border	Check Proposal style	Proposal style is green	P
80	Load Approved Proposals	On redirect to Reviewer portal Approved	Display all proposals Approved	P
71	PDF Button	If proposal has a pdf file create a button for it	A button for each pdf should be displayed in the middle of the proposal	P
82	PDF button pressed	Click PDF button	Open modal that displays pdf on screen	P
83.a	Close PDF modal	Click x in top right corner	Close pdf modal	P
83.b	Close PDF modal	Click outside the modal	Close PDF modal	P
84.a	Action buttons Reviewer	Current User is Reviewer	Display nothing	P
84.b	Action buttons Admin/Owner	Current user is Admin/owner	Display 1 action buttons, reset	P
85.a	Action buttons enabled	Current user is not the same as proposal uid	Action buttons are enabled	P
85.b	Action buttons disabled	Current user is the same as the proposals uid	Action buttons are disabled	P

86	Click Reset	Click reset button as admin	Set counter back to 0 /0 And increase reset counter by 1 Remove proposal from approved list and back to pending	P
87	Click open file	Click open file button	Display all files submitted with application	P
88	Click on drop down file from open file	Click on a file	Open file in new tab	P
89	Click download file	Click download file button	Display all files submitted with application	P
90	Click on drop down file from download file	Click on a file	Downloads selected file to computer	P

K. Reviewer Portal Rejected

Test ID	Test Case	Condition	Expected Results	P / F
91	Red border	Check Proposal style	Proposal style is red	P
92	Load rejected Proposals	On redirect to Reviewer portal rejected	Display all proposals rejected	P
93	PDF Button	If proposal has a pdf file create a button for it	A button for each pdf should be displayed in the middle of the proposal	P
94	PDF button pressed	Click PDF button	Open modal that displays pdf on screen	P
95.a	Close PDF modal	Click x in top right corner	Close pdf modal	P
95.b	Close PDF modal	Click outside the modal	Close PDF modal	P
96.a	Action buttons Reviewer	Current User is Reviewer	Display nothing	P
96.b	Action buttons Admin/Owner	Current user is Admin/owner	Display 1 action buttons, reset	P
97.a	Action buttons enabled	Current user is not the same as proposal uid	Action buttons are enabled	P
97.b	Action buttons	Current user is the same	Action buttons are disabled	P

	disabled	as the proposals uid		
98	Click Reset	Click reset button as admin	Set counter back to 0 /0 And increase reset counter by 1 Remove proposal from rejected list and back to pending	P
99	Click open file	Click open file button	Display all files submitted with application	P
100	Click on drop down file from open file	Click on a file	Open file in new tab	P
101	Click download file	Click download file button	Display all files submitted with application	P
102	Click on drop down file from download file	Click on a file	Downloads selected file to computer	P

L. Admin Portal - Reviewer Applications

Test ID	Test Case	Condition	Expected Results	P / F
103	Load Reviewer Applications	On redirect to Admin portal Reviewer Apps	Display all applications	P
104	Action buttons	Current user is Admin/owner	Display 3 action buttons reject, pending, approve	P
105.a	Action buttons enabled	Current user is not the same as proposal uid	Action buttons are enabled	P
105.b	Action buttons disabled	Current user is not the same as proposal uid	Action buttons are disabled	F
106	Click Reject - enabled	Click reject button	Reject reviewer application, Change style to red	P
107	Click Reject Twice - enabled	Click reject button	Nothing	P
108	Click Approve - enabled	Click approve button	Approve reviewer application, Change style to green	P
109	Click Approve Twice - enabled	Click approve button	Nothing	P

110	Blue border	Check Proposal style	Application is pending	P
111	Green border	Check Proposal style	Application is Approved	P
112	Red border	Check Proposal style	Application is Rejected	P

L. Admin Portal - Overview

Test ID	Test Case	Condition	Expected Results	P / F
113	Load overview of website	Redirect to overview	Load in requested information from database	P
114	Get total users created	Query the list of users	Expect to see the users created stat to match up with the default number of users in table	P
115	Get total reviewer application	Query all reviewer applications	Number of reviewer applications should be the same in reviewer applications link	P
116	Reviewer Approved	Query all approved reviewers	Should be same number of reviewers in the table when you query for just reviewers	P
117	Proposal Applications	Query all proposal Applications	Should be same number as the total number of proposals in reviewer portal	P
118	Get total Admins	Query all users that are Admins	Should be same number of admins as in table when you search only for admins	P
119	Get Money Earned	Query everyone that has paid and multiply by fixed amount	Should a number multiple of true statements on table that are under paid?	F
120	Email User	Click email link on table	Opens up email client to email user	P
121.a	Promote User that's standard	Hit promote on table for user	User should move up to reviewer	P
121.b	Promote User that's reviewer	Hit promote on table for user	User should move up to admin, promote button should switch to disabled	P
121.c	Promote User	Promote button should	Nothing	P

	that's admin	be disabled		
122.a	Demote User that's standard	Demote button should be disabled	Nothing	P
122.b	Demote User that's reviewer	Hit demote on table for user	User should move down to standard, demote button should switch to disabled	P
122.c	Demote User that's admin	Hit Demote button on table for user	User should move down to reviewer	P
123.a	Promote Owner	Promote button should be disabled	Nothing	P
123.b	Demote Owner	Demote button should be disabled	Nothing	P
124	Delete user	Hit the trash can on user	Ask for user to be delete	F
124	Delete User Confirm	Hit yes on confirmation box	Delete User from Database	F
125.a	Query just admins	Hit status dropdown and select admins	Only Admins should be displayed in table	P
125.b	Query just Reviewers	Hit status dropdown and select Reviewers	Only Reviewers should Show up	P
125.c	Query all standard accounts	Hit status dropdown and select Standard	Only Standard accounts should show up	P
125.c	Restore database	Hit status dropdown and hit restore database	Should restore table to default.	P

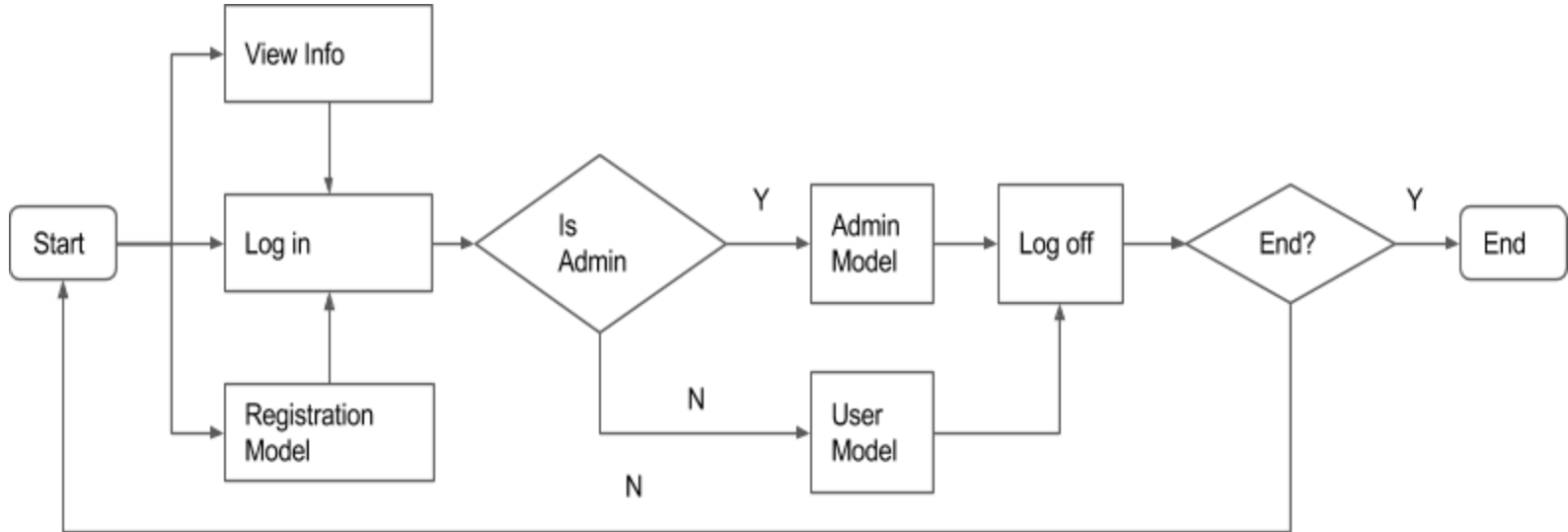
5. Documentation

a. Flow Charts:

Following are some of the flow charts that we developed early in our process, to help us shape the construction of the website.

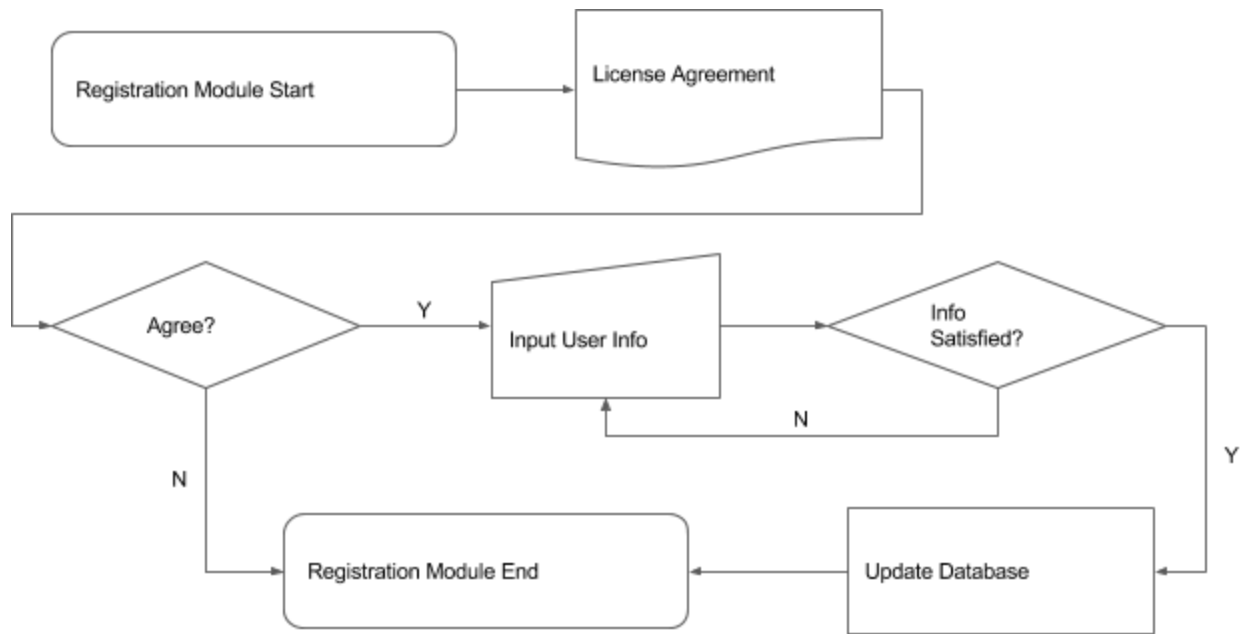
i. Main Model Flowchart:

User (Admin, Visitor, Attendee, Presenter, Reviewer): This element represents someone who accesses to website for various reasons.

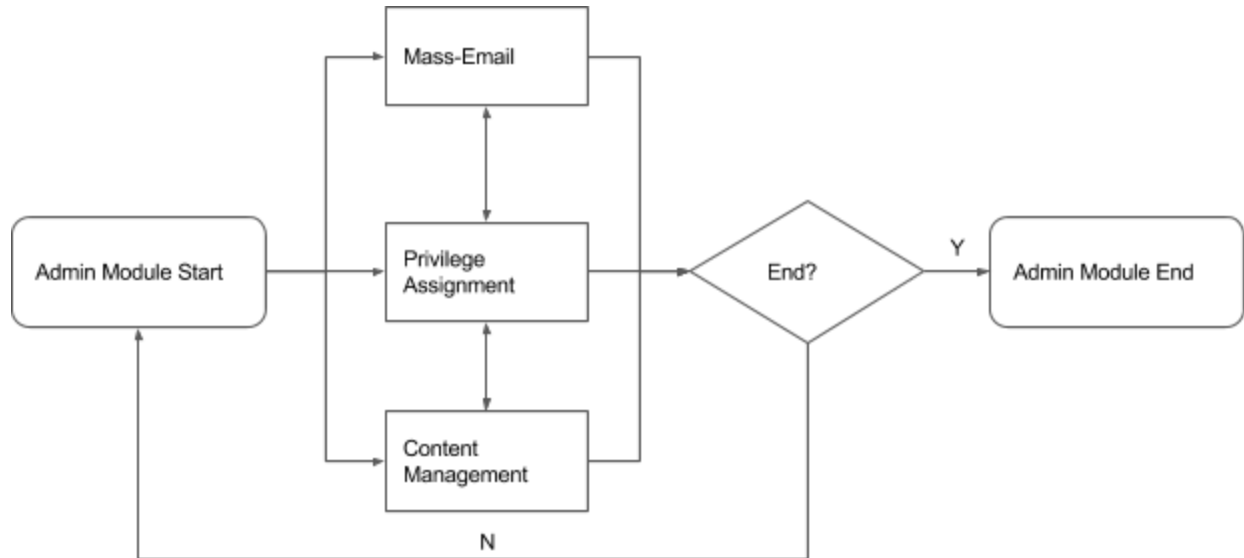


ii. Registration Model Flowchart:

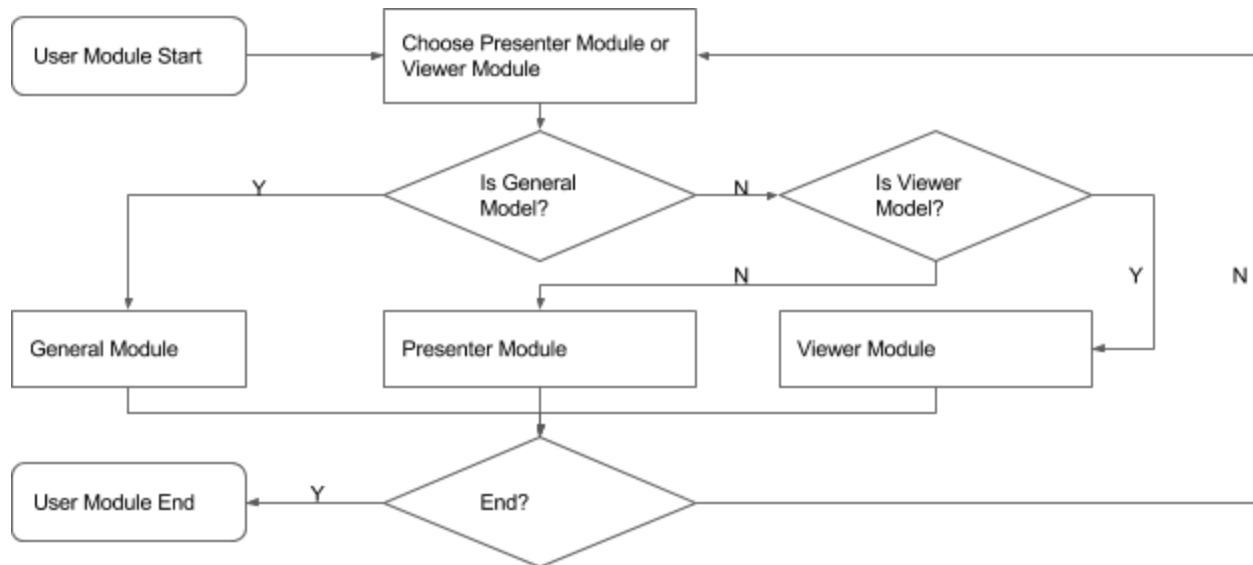
This element enables visiting people to create an account, which grants them further access to the features of the website such as attending symposiums, applying to be a reviewer, and applying to be a presenter. When a user attempts to create a new account, required information such as the e-mail will help to see if an account already exists for that user and they will be given the opportunity to log in/recover their password.



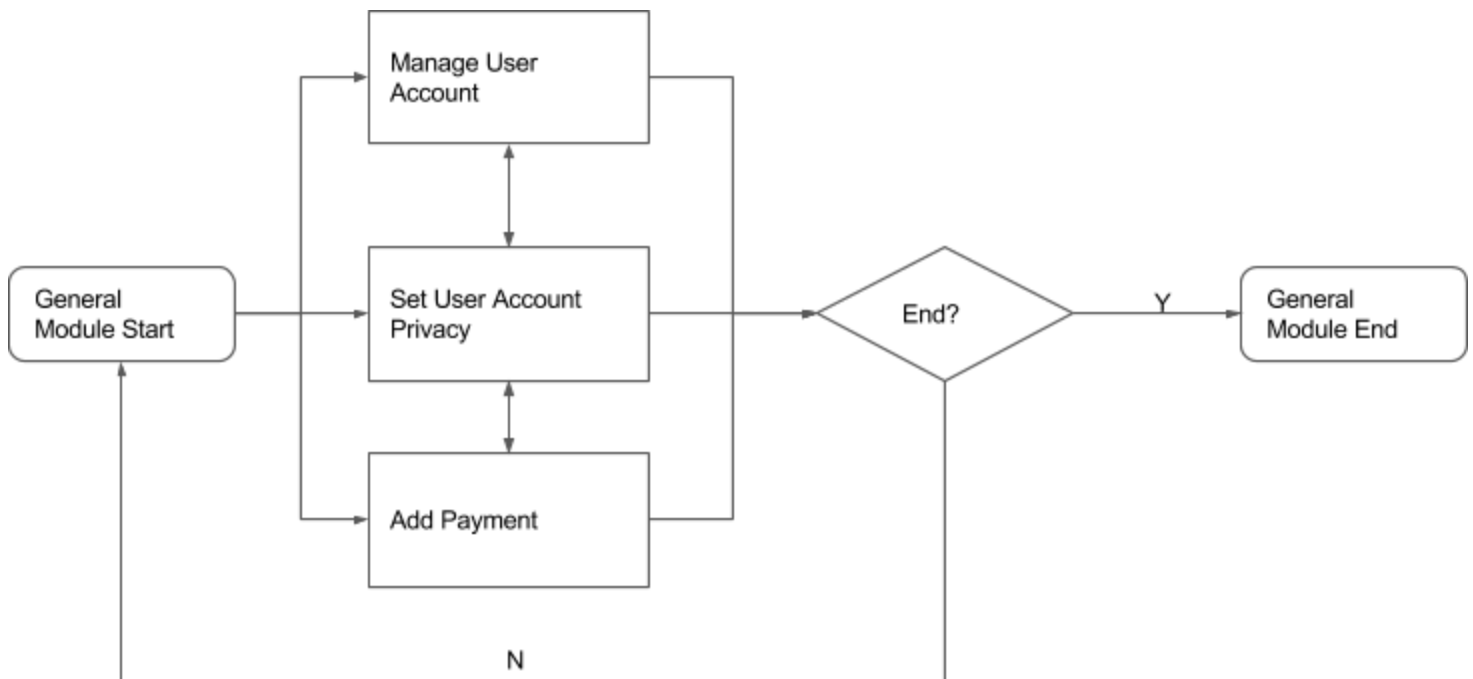
iii. Admin Model Flowchart:



iv. User Model Flowchart:

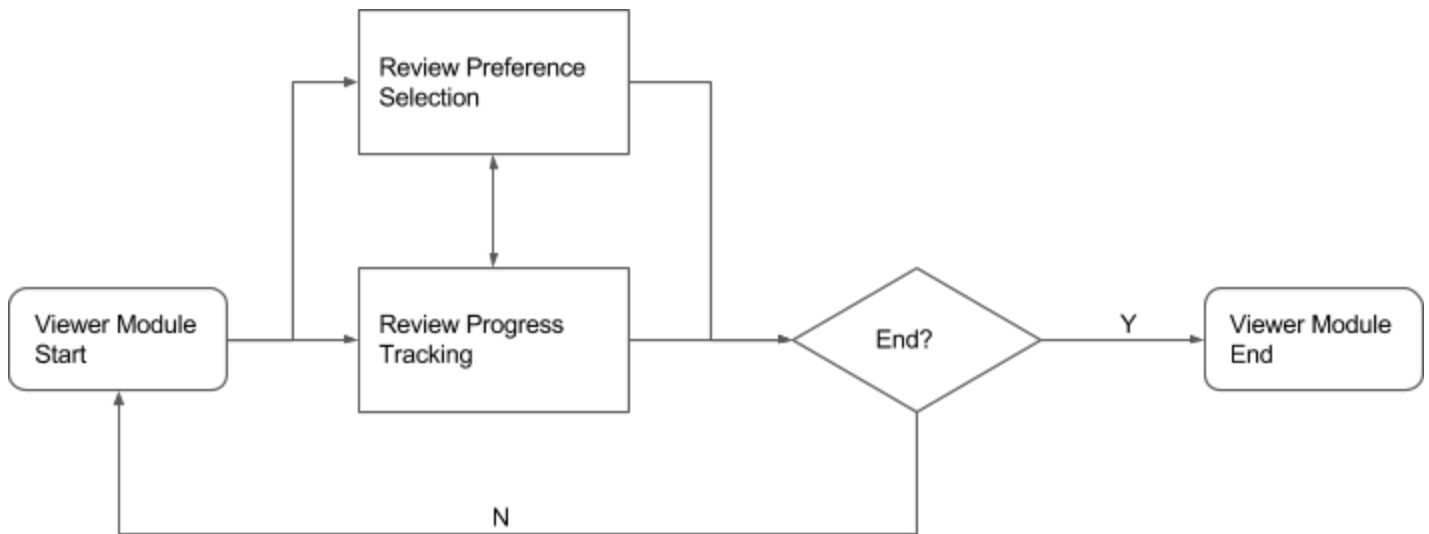


v. General User Model Flowchart:



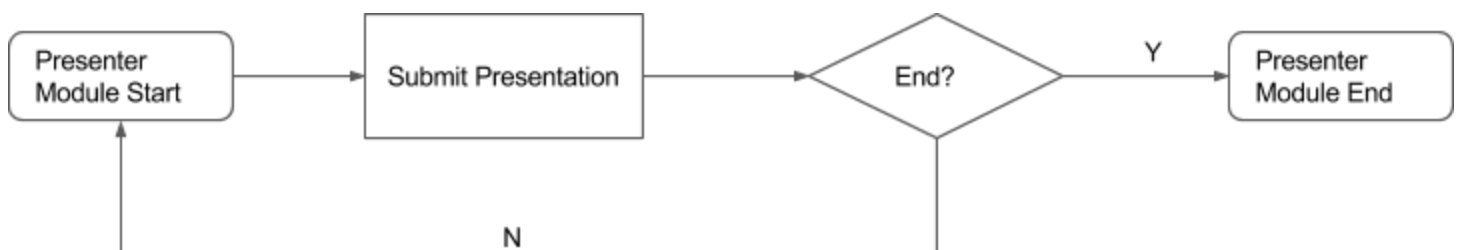
vi. Viewer Model Flowchart:

This element allows a reviewer user to update their progress for a proposed presentation and gives the presenter an easy way to see what stage their proposal(s) currently sits at.



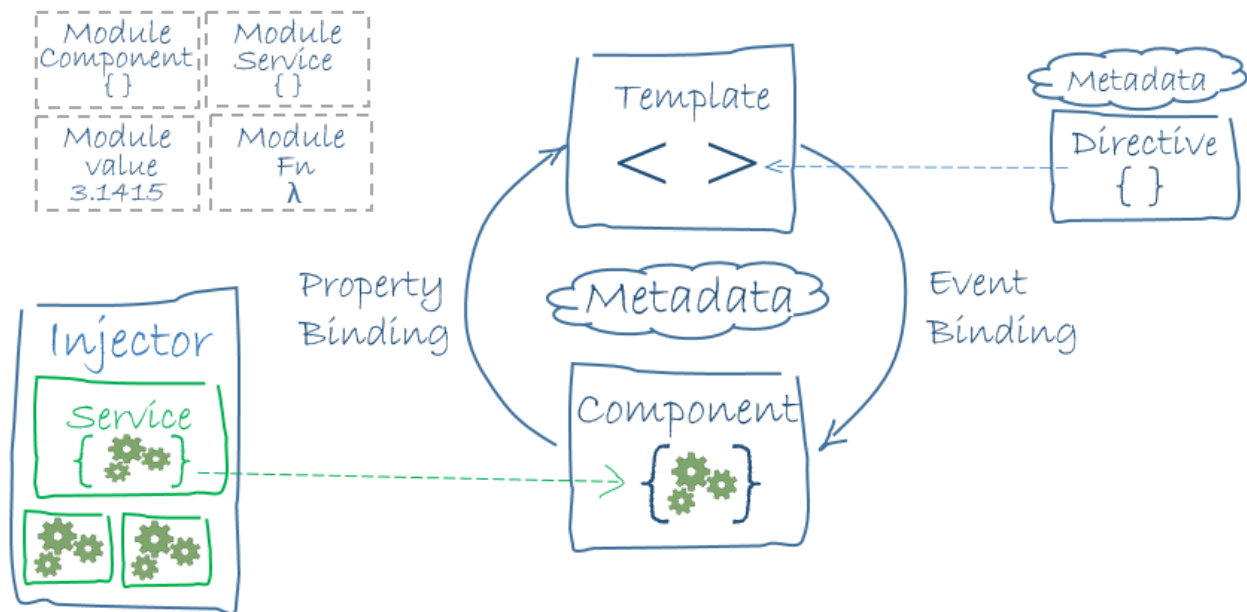
vii. Presenter Model Flowchart:

This element contains the user information for the creator of the presentation as well as the proposal itself. This will likely be a pdf or doc file type (restricted to one of those types or either of those types) that is then stored for access by admins/reviewers/the potential presenter.



b. Architecture:

Angular 2 Architecture: (images provided by Angular IO)



We have multiple modules that contain information for components, services, values, and functions. Beneath that are services that get injected into the component. The component has two way data binding between the template and itself. There are also directives being pushed into the template.

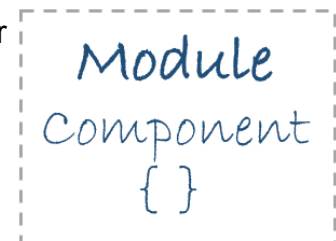
Module Component:

AppModule - every angular app has at least one module class or the "root module"

Feature Modules - can also contain feature modules dedicated to applications set of capabilities

NgModule decorator - the decorator attaches metadata to classes that know how the classes should work

Declarations - these are the view classes, which consists of three, components, directives and pipes

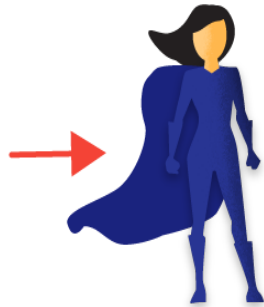
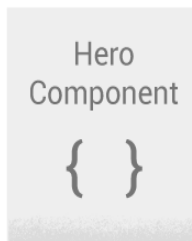


Exports - subset declarations that are seen and used in component templates of other modules

Imports - other modules whose exported classes are needed are declared in this

Components:

- Major Concept of Angular 2
- Everything is a component
- Requires three things



-First it needs a selector. The selector is a way for the root module to know that it exist and therefore can import the component into the app.

-Second, it needs a template which will be covered next. But essentially it is the markup (HTML)

-Lastly, it needs a style sheet, such as CSS for the template.

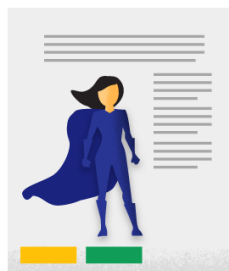
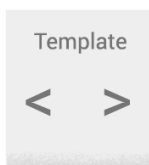
Component - controls patch of screen called view

application logic - what it does to support view - inside a class and the class interacts with the view through an API of properties and methods

As the user goes through the application they can create, update and destroy components

A few examples of components in our project are our different pages (home page, about page, event page), our forms are components, and so are the header and footer of the application.

Templates:

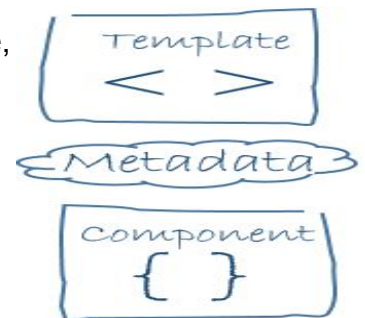


A template is used to define a components view. The template is an HTML form that tells Angular how to render the component. While many parts of the template strongly resemble normal HTML, there are parts of template-specific code used within the file. We'll find that the custom components mingle seamlessly with HTML, since the syntax is so similar.

Metadata:

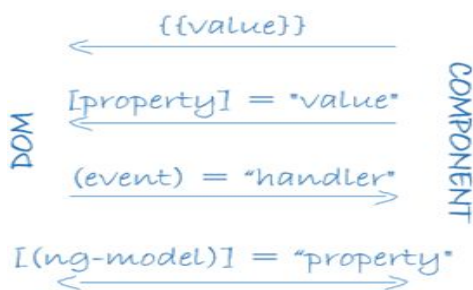
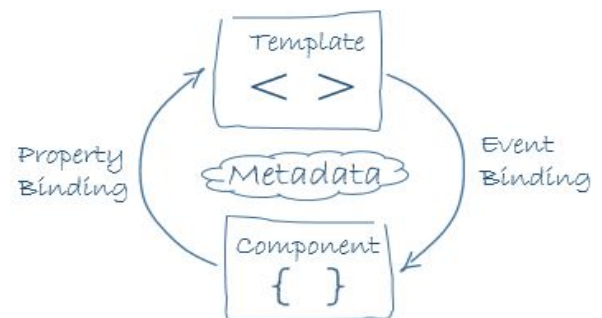
Tells Angular how to process a class. Within the Angular architecture, classes are simply classes until you tell Angular about them.

@Component is one decorator used to indicate that the following class is actually a component class. The metadata inside of the component tells Angular where we get the building blocks that we specify for that component. The template, metadata, and component together describe a view.



Data binding:

Coordinates parts of a template with parts of components. Including between template and its components and between parent and child components. Multiple ways to data bind that include:



- To the DOM
- From the DOM
- In both directions

Directives:

Two types:



Structural - alter layout by adding, removing and replacing elements in DOM
Attribute - alters appearance or behavior of existing element

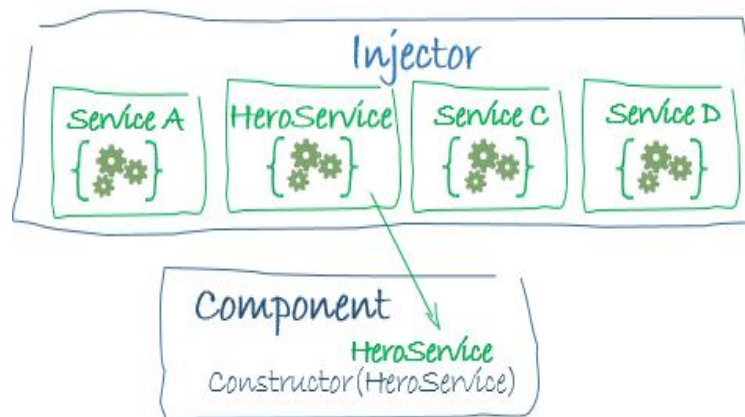
Services:

Services is a broad category and almost anything can be a service, from values to program logic.

In our application we currently have two services that contain the values of states and universities.



Dependency Injection:



Supply new class instances with its dependencies needed
Wired into the Angular framework and is used throughout

Injector is main mechanism and has its own container of services, can create a new instance from provider

Provider - creating a service

C. Database:



We choose firebase because of its compatibility with angular and because it had a designated API that easily allows communication between the two services. Firebase also works on security and authentication behind the scenes, so we don't necessarily have to worry about it. And it has a storage bucket connected to google drive that is managed by firebase. All we need to do is upload and download from it through api calls. This will be very useful when we are dealing with the client's proposals. Last but not least it offers free live hosting of our angular project.

Some Additional Pros

Pros

- Can be updated by "a lot" of users
- Other Authentication abilities
- Stored data is in the cloud and readily available
- It allows for notifications to be sent out
- They host the data for you
- And they have crash reporting

To The left is an example of a database entry for the user. Which will be discussed later in the documentation.



Database Design:

User Entry:

drs-db

users

-KdxSa-hBj23PbduiQ3N

address: [REDACTED]

city: [REDACTED]

department: [REDACTED]

diet: [REDACTED]

discipline: [REDACTED]

email: [REDACTED]

firstName: "Nathan"

lastName: "Peterson"

mi: "A"

phoneNumber: [REDACTED]

special: [REDACTED]

state: "IN"

subdiscipline: [REDACTED]

uid: [REDACTED]

university: [REDACTED]

username: [REDACTED]

zipcode: [REDACTED]

Reviewer Application:

reviewerApplications

Cody LeFan-Weaver

approved: true

uid: "v47JPQDGkaZe1SxWYHqJZOhVnBv"

Daniel Lux

approved: true

uid: "jVhKp2Cz3cP1nL55aJ0JVvRPnD7"

Faze MaGee

approved: false

uid: "TeRspysvVJa8U1Ld8zBthPyGh4f"

Nathan Peterson

approved: true

uid: "C2Ym1LpFXvfiFveGHXQrevjrwmC"

Proposal Application:



User Explanation: the string of characters directly beneath “users” is randomized to prevent overlap of user entities. Many of the fields contain string values, with exceptions including diet and special (boolean values). Diet is a marker for special dietary considerations that the user has when attending the symposiums, special is a flag for whether the user is more than “just a user” in some way.

Department is an indicator for things such as Computer Science that the user represents for their listed university, where discipline and sub-discipline will indicate the specialty of a visitor such as software engineering.

In the university field, we populated the list with a json file created by Quora user “hakimelek.” There is some optimization needed in order to best utilize the list, as it currently slows the load time for the page from near instantaneous to about a second.

The database interacts with Angular2 to create n-way databinding, allowing for changes to occur in a user-friendly way as the elements are being interacted with.

6. Dependencies

Our application depends on Angular 2 and its libraries as well as Firebase, a Google-created database system. The combination of these two is often named “AngularFire 2” and they are designed in such a way that they work together very smoothly.

a. FireBase

<https://firebase.google.com/docs/web/setup>

b. Angular 2

<https://angular.io/docs/js/latest/>

c. AngularFire2:

<https://github.com/angular/angularfire2>

Used to connect Angular2 to Firebase3

d. FirebaseCLI:

<https://firebase.google.com/docs/cli/>

Used for hosting database on live server.

e. AngularCLI:

<https://github.com/angular/angular-cli>

Used for basis of angular2 building, auto generates basic files needed.

7. Transfer Rights

As we discussed in our initial report, our intention is to sign over rights to the website and database to the client while retaining the ability to reference our work for demonstration of our technical skills. The DRS will be responsible for any fees associated with the website's use going forward, should they choose to upgrade to paid features for FireBase, as well as the cost of hosting the website which is a cost they have already been paying for the current/prior websites.

8. Gantt Chart

Gantt Chart Link:

https://docs.google.com/spreadsheets/d/1eYhCK8qO9A5Ck3SdnJ6X0C-K_5cf4fY3in_Cg3eDGDl/edit#gid=0



9. Work Breakdown

Our GitHub commit breakdown:



-Nathan was the lead developer for this project, testing, created developer manual, architecture, design of website, Demo presenter, and put fair share into documentation.

-Yibo acted as an assistant tester and helped to compile reports for the project.

-Turner helped with compiling the reports for the project.

-Cody was primarily responsible for the reports developed for each of the milestones and led presentations.

10. File Structure

The code is structured in multiple files inside the src package. Which when compiled to the host site will be merged into a simplified version in dist package so that the host server can understand our code.

Inside the src package we have many folders and files. The main components of the application are inside the app folder. Any picture/video/other forms of media are contained in the assets folder. Where we could also place the JS folder into but for the time being we left it outside.

Environment contains information that will not be compiled or sent out for public consumption. Such as passwords and API keys.

A closer look at what is inside of the app folder. The app folder contains many files and folders. The files on the outside of the folder are global modules that import the many components for Angular 2, these components are what the folders are in this directory. Each component has 4 files inside it. These files are the component class, the html and styles for the particular component and lastly each folder contains its own test file, for easily testing components by themselves.

File Tree:

```
Folder PATH listing
C:.\
| .editorconfig
| .firebaserc
| .gitignore
| angular-cli.json
| database.rules.json
| firebase.json
| karma.conf.js
| output.txt
| package.json
| protractor.conf.js
| README.md
| tslint.json
|
+---dist
| | favicon.ico
```

```

| | index.html
| | inline.d41d8cd98f00b204e980.bundle.js
| | inline.d41d8cd98f00b204e980.bundle.map
| | main.7a69c519bbdffaf1a66d.bundle.js
| | main.7a69c519bbdffaf1a66d.bundle.js.gz
| | main.7a69c519bbdffaf1a66d.bundle.map
| | styles.b2328beb0372c051d06d.bundle.js
| | styles.b2328beb0372c051d06d.bundle.map
| | styles.b3984c4ab88a1853ff1cbf0cf96d3d9a.bundle.css
| |
| \---assets
| |   big_logo.jpg
| |   diversity-hands.jpg
| |   diversity-print.jpg
| |   drop-images.png
| |   logo.jpg
| |
| +---eventPage
| |   1.jpg
| |   2.jpg
| |   3.jpg
| |   keynoteSpeaker.jpg
| |
| \---previousLogo
|     2009.jpg
|     2010.jpg
|     2011.jpg
|     2012.jpg
|     2013.jpg
|     2014.jpg
|     2015.jpg
|     2016.jpg
|
+---e2e
|   app.e2e-spec.ts
|   app.po.ts
|   tsconfig.json
|
+---node_modules
| +---.bin
+---src
|   favicon.ico
|   index.html

```

```

| main.ts
| polyfills.ts
| styles.css
| test.ts
| tsconfig.json
| typings.d.ts
|
+---app
| | app-routing.module.ts
| | app.component.css
| | app.component.html
| | app.component.spec.ts
| | app.component.ts
| | app.module.ts
| | index.ts
| |
| +---about
| |   about.component.css
| |   about.component.html
| |   about.component.spec.ts
| |   about.component.ts
| |   timeline.js
| |
| +---admin-portal
| | | admin-portal.component.css
| | | admin-portal.component.html
| | | admin-portal.component.spec.ts
| | | admin-portal.component.ts
| | |
| | +---overview
| | |   overview.component.css
| | |   overview.component.html
| | |   overview.component.ts
| | |
| | \---reviewer-apps
| |   application.component.css
| |   application.component.html
| |   application.component.ts
| |   reviewer-apps.component.css
| |   reviewer-apps.component.html
| |   reviewer-apps.component.ts
| |
| +---directives

```



```

| |   file-item.ts
| |   ng-drop-files.directive.ts
| |
| +---footer
| |   footer.component.css
| |   footer.component.html
| |   footer.component.spec.ts
| |   footer.component.ts
| |
| +---header
| |   header.component.css
| |   header.component.html
| |   header.component.spec.ts
| |   header.component.ts
| |
| +---header-parallax
| |   header-parallax.component.css
| |   header-parallax.component.html
| |   header-parallax.component.spec.ts
| |   header-parallax.component.ts
| |
| +---home
| |   home.component.css
| |   home.component.html
| |   home.component.spec.ts
| |   home.component.ts
| |
| +---login
| |   login.component.css
| |   login.component.html
| |   login.component.spec.ts
| |   login.component.ts
| |
| +---my-event-page
| |   my-event-page.component.css
| |   my-event-page.component.html
| |   my-event-page.component.spec.ts
| |   my-event-page.component.ts
| |
| +---navbar
| |   navbar.component.css
| |   navbar.component.html
| |   navbar.component.spec.ts

```

```

| | navbar.component.ts
| |
| +---profile
| | profile.component.css
| | profile.component.html
| | profile.component.spec.ts
| | profile.component.ts
| |
| +---proposal-app
| | proposal-app.component.css
| | proposal-app.component.html
| | proposal-app.component.spec.ts
| | proposal-app.component.ts
| |
| +---register
| | register.component.css
| | register.component.html
| | register.component.spec.ts
| | register.component.ts
| |
| +---review-app
| | review-app.component.css
| | review-app.component.html
| | review-app.component.spec.ts
| | review-app.component.ts
| |
| +---reviewer-portal
| | | reviewer-portal.component.css
| | | reviewer-portal.component.html
| | | reviewer-portal.component.spec.ts
| | | reviewer-portal.component.ts
| | |
| | +---proposal-approved
| | | proposal-approved.component.css
| | | proposal-approved.component.html
| | | proposal-approved.component.ts
| | |
| | +---proposal-pending
| | | proposal-pending.component.css
| | | proposal-pending.component.html
| | | proposal-pending.component.ts
| | |
| | +---proposal-rejected

```

```

| | | proposal-rejected.component.css
| | | proposal-rejected.component.html
| | | proposal-rejected.component.ts
| | |
| | \---review-proposal-object
| |     review-proposal-object.component.css
| |     review-proposal-object.component.html
| |     review-proposal-object.component.ts
| |
| | \---services
| |     institution.service.ts
| |     proposal-domain.service.ts
| |     states.service.ts
| |     total-proposals.service.ts
| |     upload-files.service.ts
| |
+---assets
| | .gitkeep
| | big_logo.jpg
| | diversity-hands.jpg
| | diversity-print.jpg
| | drop-images.png
| | logo.jpg
| |
| | +---eventPage
| |     1.jpg
| |     2.jpg
| |     3.jpg
| |     keynoteSpeaker.jpg
| |
| | \---previousLogo
| |     2009.jpg
| |     2010.jpg
| |     2011.jpg
| |     2012.jpg
| |     2013.jpg
| |     2014.jpg
| |     2015.jpg
| |     2016.jpg
| |
+---environments
|     environment.prod.ts
|     environment.ts

```

```

|   firebase.config.ts
|
\---js
    jquery-2.1.3.min.js
    jquery.countdown.min.js
    jquery.plugin.min.js

```

11. Source Code

Here is a link to our GitHub repository:

<https://github.com/NathanPeterson/DRS-Webapp/tree/master>

Angular 2 breaks the code up into more files than one might expect from a normal web application. This is a sample of the TypeScript code for the Overview Page:

```

import { Component, OnInit } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { AngularFire, FirebaseListObservable,
FirebaseObjectObservable } from 'angularfire2';
import { Subject } from 'rxjs/Subject';

@Component({
  selector: 'app-admin-overview',
  templateUrl: './overview.component.html',
  styleUrls: ['./overview.component.css']
})
export class AdminOverviewComponent implements OnInit {
  authState;
  currentUser;
  accountArray=[];
  totalUsers;
  currentTotal;
  MAX_REVIEWERS;
  totalReviewers;
  totalAdmins;
  totalProposals;
  totalReviewerApps;
  flag;

  constructor(private af: AngularFire,private router: Router) {
    this.af.auth.subscribe((auth) => {

```

```

        this.authState = auth;
    });
    this.flag = {
        key: 'uid',
        val: '',
        triggered: false,
    }
}

ngOnInit() {
    this.loadData();
}

loadData() {
    let listOfUsers = this.af.database.list('/users/', {
        query: {
            orderByChild: this.flag.key,
        }
    });

    listOfUsers.subscribe((users) =>{
        this.accountArray = [];
        this.totalReviewers = 0;
        this.MAX_REVIEWERS = 30;
        this.totalAdmins = 0;
        this.totalProposals = 0;
        this.totalReviewerApps = 0;

        users.forEach(info =>{
            let reviewApp = false;
            let proposalApp = false;
            if(info.reviewerApplication){
                reviewApp = true;
                this.totalReviewerApps++;
            };
            if(info.proposals){
                proposalApp = true;
                for(let proposal in info.proposals){
                    this.totalProposals++;
                }
            }
            if(this.isReviewer(info)){
                this.totalReviewers++;
            }else if(this.isAdmin(info)){

```

```

        this.totalAdmins++;
    }
    this.accountArray.push({
        username: info.username,
        email: info.email,
        uid: info.uid,
        pApp: proposalApp,
        rApp: reviewApp,
        status: info.accountType,
        paid: false,
    });
    });
    this.currentTotal = this.totalUsers =
this.accountArray.length;
    });
}

isReviewer(data){
    return data.accountType === 'reviewer';
}
isAdmin(data){
    return data.accountType === 'admin' || data.accountType ===
'owner';
}

promote(data){
    let selectedUser = this.af.database.object('/users/' +
data.uid);
    let currentType;
    selectedUser.subscribe(info =>{
        currentType = info.accountType;
    });
    if(!currentType){
        selectedUser.update({accountType: 'standard'});
    }else if(currentType === 'standard'){
        selectedUser.update({accountType: 'reviewer'});
    }else if(currentType === 'reviewer'){
        selectedUser.update({accountType: 'admin'});
    }else if(currentType === 'admin'){
        return;
    }
}
}

```

```

demote(data) {
    let selectedUser = this.af.database.object('/users/' +
data.uid);
    let currentType;
    selectedUser.subscribe(info =>{
        currentType = info.accountType;
    });
    if(currentType === 'admin'){
        selectedUser.update({accountType: 'reviewer'});
    }else if(currentType === 'reviewer'){
        selectedUser.update({accountType: 'standard'});
    }else if(currentType === 'standard'){
        return;
    }
}

loadSortedData() {
    let listOfWorkers;
    if(this.flag.val === ''){
        listOfWorkers = this.af.database.list('/workers',{
            query: {
                orderByChild: this.flag.key,
            }
        });
    }else{
        listOfWorkers = this.af.database.list('/workers',{
            query: {
                orderByChild: this.flag.key,
                equalTo: this.flag.val
            }
        });
    }
    listOfWorkers.subscribe((workers) =>{
        this.workerArray = [];
        workers.forEach(info =>{
            let reviewApp = false;
            let proposalApp = false;
            if(info.workerApplication){
                reviewApp = true;
            };
            if(info.proposals){
                proposalApp = true;
            }
        }
    }

```

```

        this.accountArray.push({
            username: info.username,
            email: info.email,
            uid: info.uid,
            pApp: proposalApp,
            rApp: reviewApp,
            status: info.accountType,
            paid: false,
        });
    })
    this.currentTotal = this.accountArray.length;
});
}

canPromote(data) {
    if(data.status === 'admin' || data.status === 'owner'){
        return false;
    }else{
        return true;
    }
}

canDemote(data) {
    if(data.status === 'standard' || data.status === 'owner'){
        return false;
    }else{
        return true;
    }
}

sortByStatus(input) {
    this.flag.key = 'accountType'
    if(!input){
        if(this.flag.triggered === false){
            this.flag.triggered = true;
            this.loadSortedData();
        }else{
            this.accountArray.reverse();
        }
    }else if(input === 'admin' || input === 'standard' || input ===
'reviewer'){
        this.flag.val = input;
        this.loadSortedData();
    }else if(input === 'default'){

```



```

        this.flag.val = '';
        this.loadSortedData();
    }
}

sortByReviewApp(input, type){
    if(type === 'r'){
        this.flag.key = 'reviewerApplication'
    }else{
        this.flag.key = 'proposals'
    }
    if(!input){
        if(this.flag.triggered === false){
            this.flag.triggered = true;
            this.loadSortedData();
        }else{
            this.accountArray.reverse();
        }
    }else if(input === 'submitted'){
        this.loadSortedData();
    }else if(input === 'notSubmitted'){
        if(type === 'r'){
            this.flag.key = "!reviewApp";
        }else{
            this.flag.key = "!proposals"
            this.loadSortedData();
        }
    }else if(input === 'default'){
        this.flag.val = '';
        this.loadSortedData();
    }
}

delete(data) {
    this.af.database.list('/users/').remove(data.uid).then(()=>{
        alert("Are You sure you want to delete this user?")
    }).then((success)=>{}).catch(err=>{})
}

canDelete(data){
    if(data.status === 'owner'){
        return false;
    }else{
        return true;
    }
}

```

}

}

Developer Manual

This manual is intended to get the next developer up to speed on how to develop for this project and get their feet wet with Angular 2 and Firebase 3.

1. Install the appropriate developer environment

-Node.js <https://nodejs.org/en/>

Install the current version of node.js. The image below has a red rectangle on the button you should click for the current version.



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Download for Windows (x64)



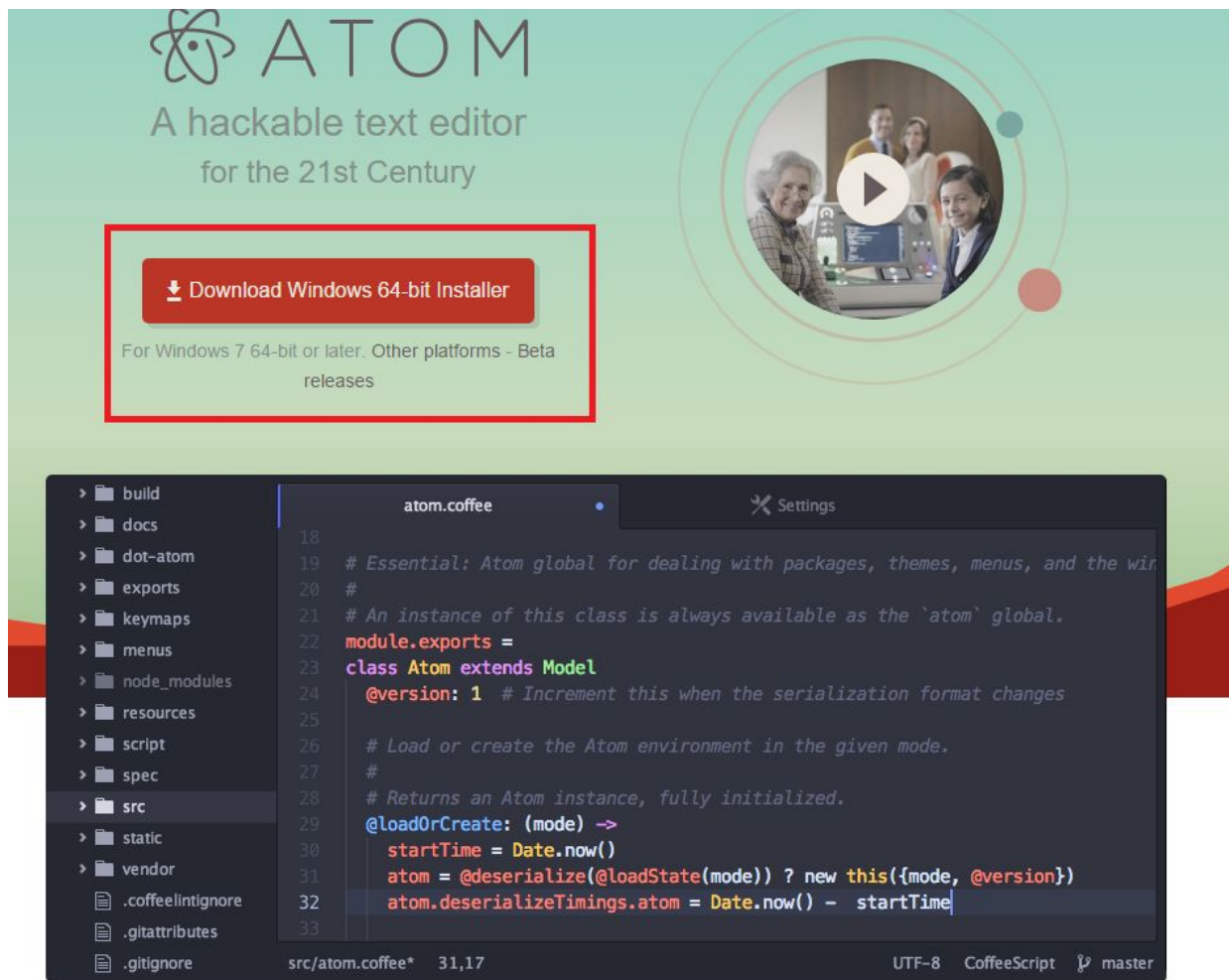
-Git <https://git-scm.com/downloads>

Download and install your OS's version of git and ensure you get **Git Bash**



-Atom <https://atom.io/>

This is the text editor I used for the project, you'll need to use this along side Git Bash to get the code up and running.



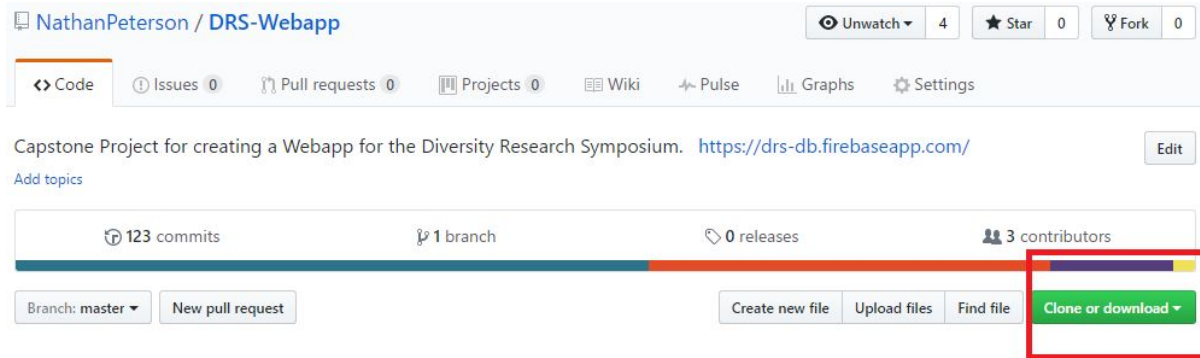
2. Getting the project on a new computer

There are 2 ways to get the project onto your computer. The first way is a straight download, whereas the second way requires the use of the command line.

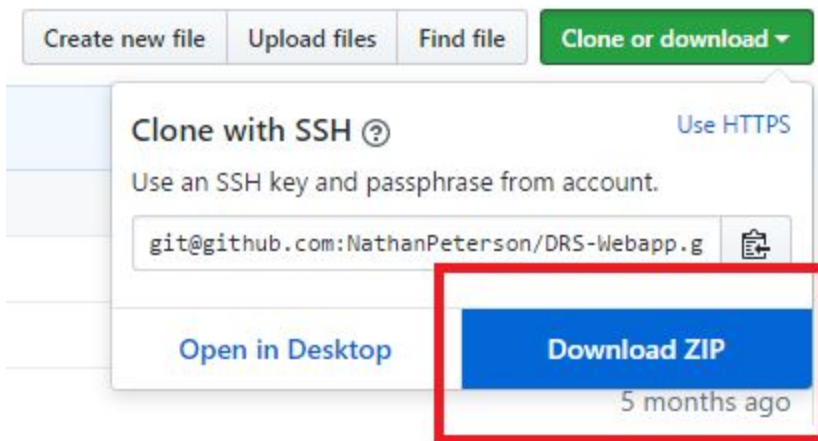
First way: No command line

Step 1: Go to this url: <https://github.com/NathanPeterson/DRS-Webapp>

Step 2: Click on the green 'Clone or Download' Button



Step 3: Click Download Zip button:



Step 4: Unzip the downloaded file to your desired location.

Step 5: You now have the project on your machine

Second way: Command line Requires Git Bash if on Windows

Step 1: Setup Git on command line

Run these commands on cmd:

```
$git config --global user.name "Your Full Name"
$git config --global user.email "youremailaddress@example.com"
```

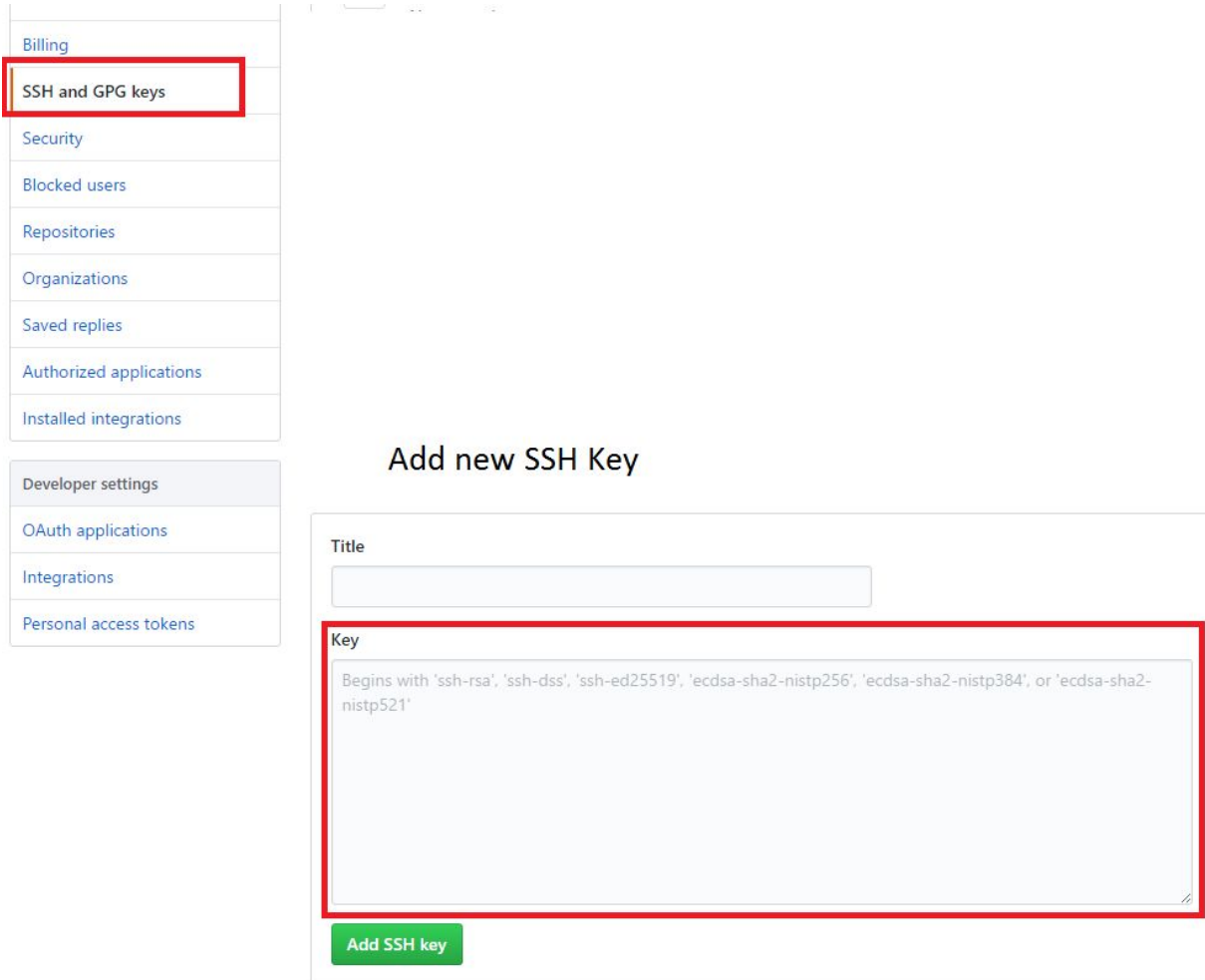
```
//Generate ssh
$ssh-keygen -t rsa -C "youremail@example.com"
```

```
//Change the file permission
$chmod -R 700 .ssh/
$chmod 600 .ssh/id_rsa
```

//Copy SSH Key

```
$cat .ssh/id_rsa.pub | pbcopy
```

Step 2: Add key to your github account by copy and pasting the SSH key into your githubs SSH settings



The screenshot shows the GitHub 'SSH and GPG keys' settings page. On the left, a sidebar menu lists various settings, with 'SSH and GPG keys' highlighted by a red box. The main content area is titled 'Add new SSH Key'. It contains a 'Title' text input field and a 'Key' text area, which is also highlighted by a red box. The 'Key' area contains placeholder text: 'Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521''. Below the key area is a green 'Add SSH key' button.

Step 3: Click add SSH key after giving it a title and pasting in key

Step 4: Check if your SSH Key is working

Run this command on command line

```
$ssh -T git@github.com
```

Step 4.A: Successful key

Should receive this message: You've successfully authenticated, but GitHub does not provide shell access.

Step 4.B: Failed Key

Should receive this message: This private key will be ignored.
Load key "/home/whoever/.ssh/id_rsa": bad permissions
Permission denied (publickey).

Just rerun these two commands:

```
$chmod -R 700 ~/.ssh/  
$chmod 600 ~/.ssh/id_rsa
```

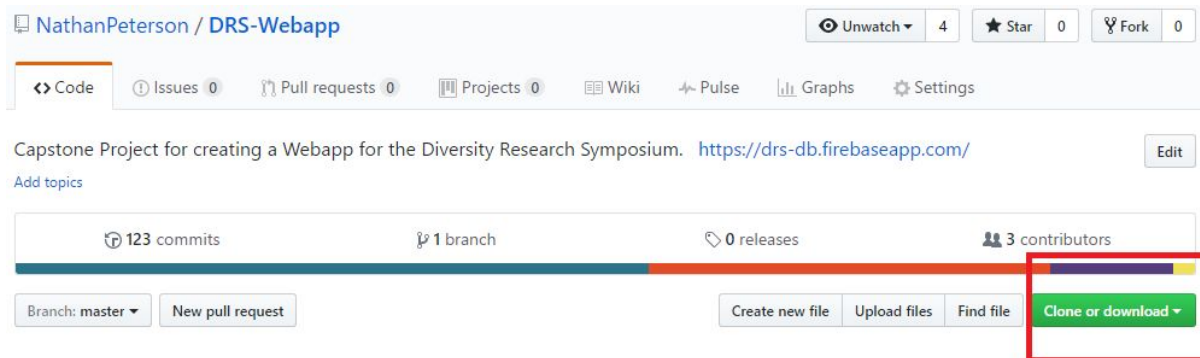
That should fix the error. Therefore rerun this command:

```
$ssh -T git@github.com
```

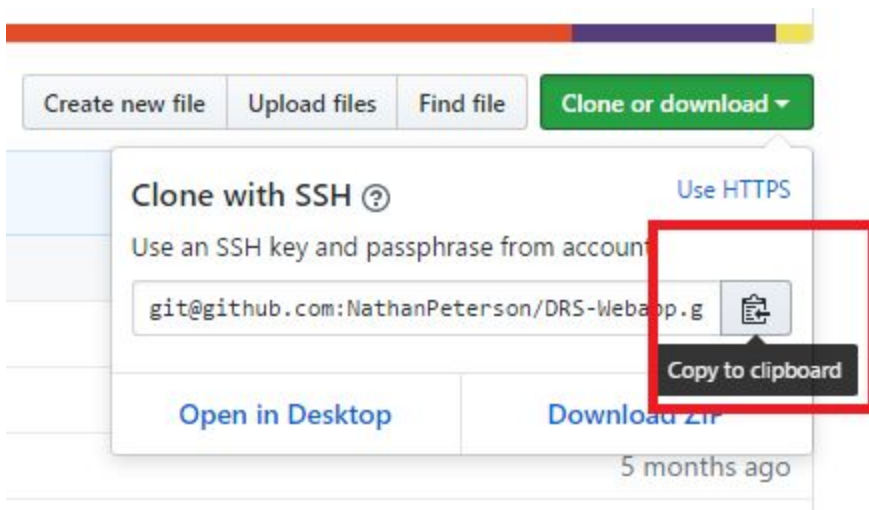
Step 5: Now you are successfully connected to github.

Step 6: Go to this url: <https://github.com/NathanPeterson/DRS-Webapp>

Step 7: Click on the green 'Clone or Download' Button



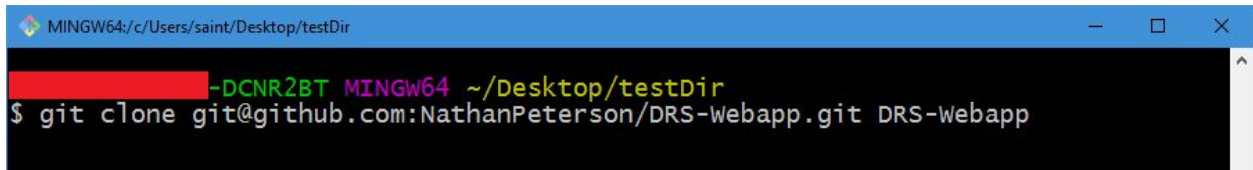
Step 8: Click the button next to the url



Step 9: Clone the repository

Run this command on the command line:

\$ git clone [url you copied] [name of folder you want to clone to]

A screenshot of a Windows command prompt window. The title bar shows 'MINGW64: c:/Users/saint/Desktop/testDir'. The command prompt shows a red prompt character followed by the text '-DCNR2BT MINGW64 ~/Desktop/testDir'. The command entered is '\$ git clone git@github.com:NathanPeterson/DRS-Webapp.git DRS-Webapp'.

Step 10: Now you have the project on your computer.

3. Install the project. This will require git bash from now on.

Step 1: Access the folder you made with \$cd [folder name]

Step 2: Run these command

```
$npm install
```

```
$npm install -g
```

These two commands will install all the dependencies that the project is currently needing. The '-g' command will install it to your global scope and the one without it will install it to your local scope.

This may take awhile, as Angular 2 requires a lot of programs to run.

Step 3: You'll should see a File tree after everything is installed.

Step 4: The project is now installed.

4. Running the Project

Step 1: Run this command:

```
$ng serve
```

This may take a bit but afterwards the web app should be running on localhost:4200,

Step 2: go to the url: <http://localhost:4200/home> and the app will be displayed.

Updating the project

Angular-cli documentation: <https://github.com/angular/angular-cli>

Follow the readme to learn the commands of how to manipulate the angular2 project.
Most important commands are:

\$ng generate component [name of component]

This is the most basic building block to angular2, this command will automatically build and implement a blank component into the project. It generates a component folder, which contains 4 files. An Html, CSS, Test file, and the Component file.

This is how they get all connected in component file.

```
@Component({
  selector: 'app-admin-overview',
  templateUrl: './overview.component.html',
  styleUrls: ['./overview.component.css']
})
```

\$ng test

This command will run the automated Karma test that angular2 is built on.

\$ng build --prod

This command will build the project for production. **You need to run this each time you want a new build of the production version of the code!!**

\$ng serve

This command as mentioned above is what will run the webapp on your local host.

Angular2 documentation:

Official Documentation: <https://angular.io/docs/ts/latest/>

This will be your guide on getting on your feet with angular 2.

Check out the architecture portion of the report for information on angular2 parts:
page 59

Angularfire2 Documentation

Official documentation:

<https://github.com/angular/angularfire2/blob/master/README.md>

Angularfire2 will be the bread and butter to connecting Angular2 to our database Firebase.

Firebase Documentation:

Official Documentation: <https://firebase.google.com/docs/web/setup>

This documentation will describe how Firebase works and how you can manipulate the database.

Uploading the project to Firebase for live hosting:

Documentation: <https://firebase.google.com/docs/hosting/>

Other useful developer programs we used:

Lorem Ipsum:

Link: <http://www.lipsum.com/>

Description: Lorem Ipsum is a random text generator. Useful when needing filler text for prototyping different parts of a web page.

Trello:

Link: <https://trello.com/>

Description: Trello is a collaboration tool that organizes your projects into boards. In one glance, Trello tells you what's being worked on, who's working on what, and where something is in a process.