

Assignment 4

Open Source Software Analysis and Design

1. Pathname to the Project on the VM

sysadmin@csc415-server28.hpc.tcnj.edu/servd

2. Pathname to the Project Repository on GitHub

TCNJSwEngg/servd

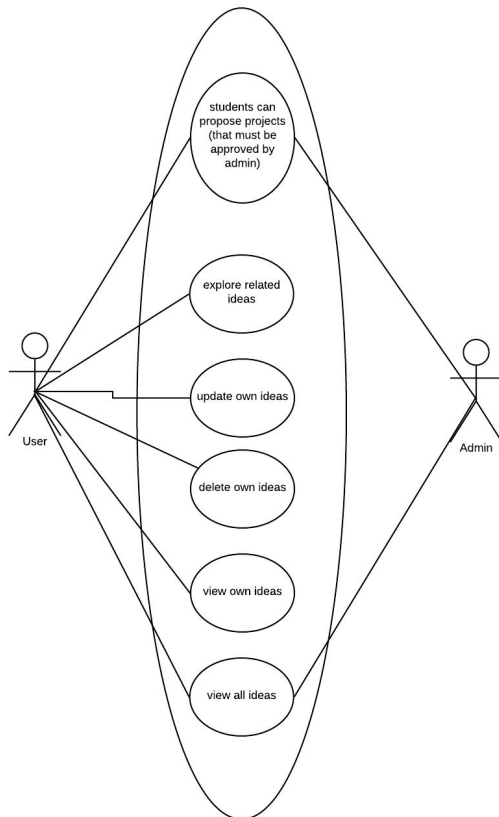
<https://github.com/TCNJSwEngg/servd>

Branch: altschuler_csc415

3. OSS License for Your Project

MIT License

4. Use Case Descriptions



Proposing Project Ideas	
Use Case:	Users can create ideas
Iteration:	First Iteration
Primary Actor:	User
Goal in Context:	To create and store the fields of an <i>idea</i>
Preconditions:	User must be login User must go to form to actually create the idea
Trigger:	The user has an idea to help the community and want feedback/support
Scenarios:	<ol style="list-style-type: none"> 1. The user logs in 2. The user clicks on "New Idea" 3. The user fills out the form 4. The user clicks "Submit"
Exceptions:	User is not login or doesn't have proper authentication
Priority:	Very High
When Available:	First increment
Frequency of Use:	Frequent
Channel to Actor:	Web Browser and Internet connection
Secondary Actors:	None
Channels to Secondary Actors:	None
Open Issues:	Admins have to actually verify the idea to be made public What are the relevant field of ideas (name/owner/description/labels/contact information/labels?) What fields must be present

Updating Project Ideas	
Use Case:	Users can update their own ideas
Iteration:	First Iteration

Primary Actor:	User
Goal in Context:	To modify the fields of an <i>idea</i>
Preconditions:	User must be login User must go to form to actually update the idea User must actually own the idea
Trigger:	The user wants to modify his/her idea
Scenarios:	<ol style="list-style-type: none"> 1. The user logs in 2. The user clicks on "Edit Idea" for their Idea 3. The user fills out the form 4. The user clicks "Update"
Exceptions:	User is not login or doesn't have proper authentication User doesn't own the idea
Priority:	Very High
When Available:	First increment
Frequency of Use:	Frequent
Channel to Actor:	Web Browser and Internet connection
Secondary Actors:	None
Channels to Secondary Actors:	None
Open Issues:	None currently

Deleting Project Ideas	
Use Case:	Users can delete their own ideas
Iteration:	First Iteration
Primary Actor:	User
Goal in Context:	To delete their own idea
Preconditions:	User must be login User must click "Delete" of idea they own
Trigger:	The user doesn't actually like their idea

Scenarios:	<ol style="list-style-type: none"> 1. The user logs in 2. The user clicks on “Delete” of an idea they own 3. The user confirms the action
Exceptions:	User is not login or doesn't have proper authentication
Priority:	Very High
When Available:	First increment
Frequency of Use:	Frequent
Channel to Actor:	Web Browser and Internet connection
Secondary Actors:	None
Channels to Secondary Actors:	None
Open Issues:	None currently

See All Public Project Ideas	
Use Case:	Users can see all ideas that are approved
Iteration:	First Iteration
Primary Actor:	User
Goal in Context:	To create and store the fields of an <i>idea</i>
Preconditions:	User must be login User must go to index page Idea must be approved by admin
Trigger:	The user wants to see all approved ideas
Scenarios:	<ol style="list-style-type: none"> 1. The user clicks All Ideas
Exceptions:	Idea is not approved by admin
Priority:	Very High
When Available:	First increment
Frequency of Use:	Frequent
Channel to Actor:	Web Browser and Internet connection

Secondary Actors:	None
Channels to Secondary Actors:	None
Open Issues:	What are relevant fields to show?

See All Own Ideas	
Use Case:	Users can see all ideas that they own
Iteration:	First Iteration
Primary Actor:	User
Goal in Context:	To see their own ideas
Preconditions:	User must be login User must go to dashboard
Trigger:	The user wants to see all approved ideas
Scenarios:	<ol style="list-style-type: none"> 1. The user logins 2. The user clicks Dashboard 3. The user can see their own ideas
Exceptions:	User is not login or doesn't have proper authentication
Priority:	Very High
When Available:	First increment
Frequency of Use:	Frequent
Channel to Actor:	Web Browser and Internet connection
Secondary Actors:	None
Channels to Secondary Actors:	None
Open Issues:	None

See Individual Idea	
Use Case:	Users can see all attributes of an idea

Iteration:	First Iteration
Primary Actor:	User
Goal in Context:	To see all the fields of an idea
Preconditions:	User must be login User must go to index page Idea must be approved by admin
Trigger:	The user wants to see all approved ideas
Scenarios:	1. The user logs in 2. The user clicks All Ideas
Exceptions:	Idea is not approved by admin and idea is not owned by user
Priority:	Very High
When Available:	First increment
Frequency of Use:	Frequent
Channel to Actor:	Web Browser and Internet connection
Secondary Actors:	None
Channels to Secondary Actors:	None
Open Issues:	None

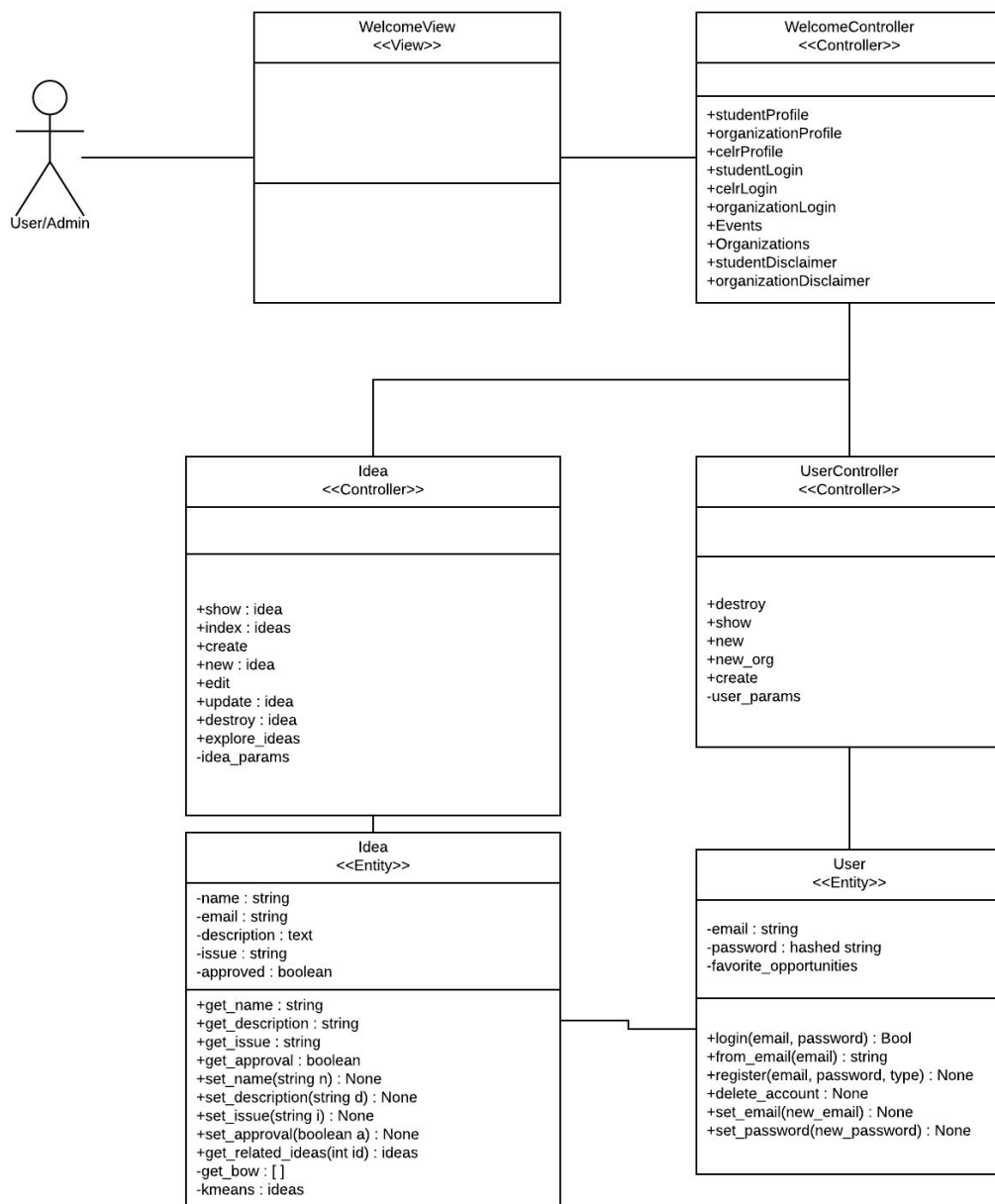
Admin Approves Idea	
Use Case:	Admin can approve an idea
Iteration:	Second Iteration
Primary Actor:	Admin
Goal in Context:	To approve an idea
Preconditions:	Admin must be login Idea must be not approved
Trigger:	The admin wants to approve ideas

Scenarios:	<ol style="list-style-type: none"> 1. The admin logs in 2. The user clicks All Ideas 3. The admin approves selected ideas
Exceptions:	User doesn't have admin permission Idea is already approved
Priority:	High
When Available:	Second increment
Frequency of Use:	Frequent
Channel to Actor:	Web Browser and Internet connection
Secondary Actors:	None
Channels to Secondary Actors:	None
Open Issues:	Should there be a way to notify admins of new ideas

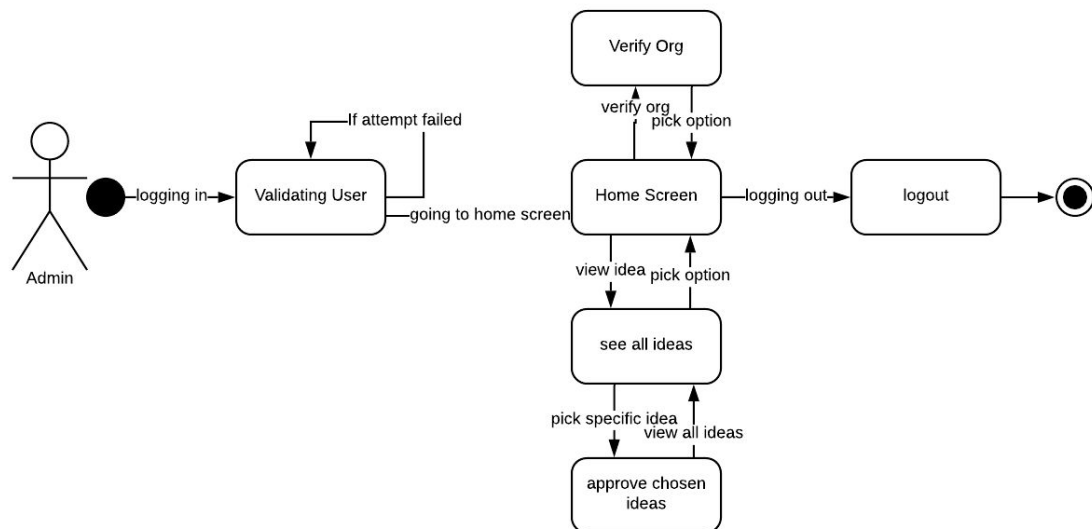
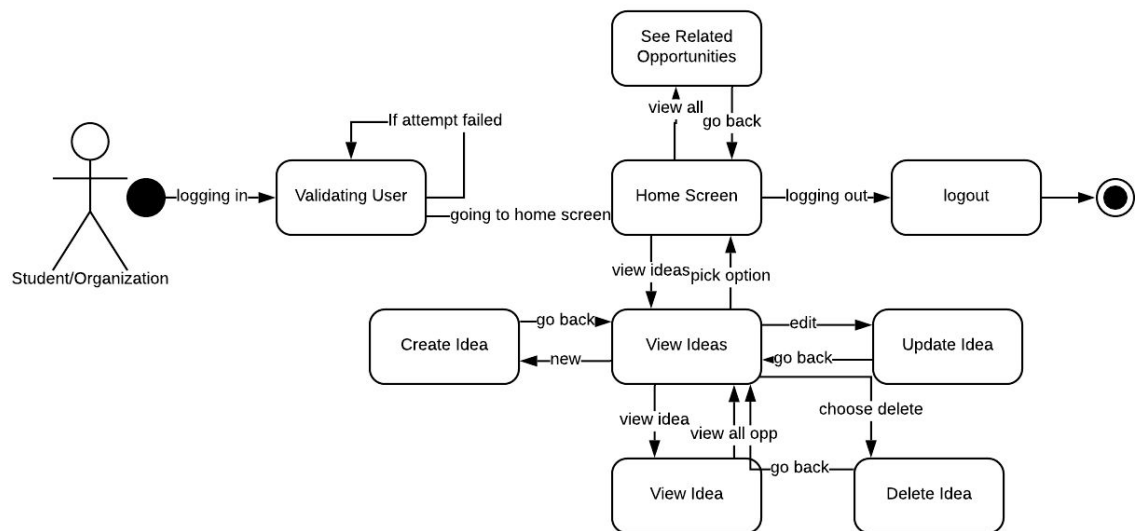
See Related Ideas	
Use Case:	Users can see all ideas related to their own ideas
Iteration:	Third Iteration
Primary Actor:	User
Goal in Context:	To see related ideas
Preconditions:	User must be login User must go to explore ideas ta
Trigger:	The user wants to see similar ideas
Scenarios:	<ol style="list-style-type: none"> 1. The user logs in 2. The user clicks Explore Ideas
Exceptions:	User is not login or doesn't have proper authentication Idea is not approved by admin
Priority:	Medium
When Available:	Third increment
Frequency of Use:	Frequent

Channel to Actor:	Web Browser and Internet connection
Secondary Actors:	None
Channels to Secondary Actors:	None
Open Issues:	What algorithm will be used (kmeans with sentence embeddings)?

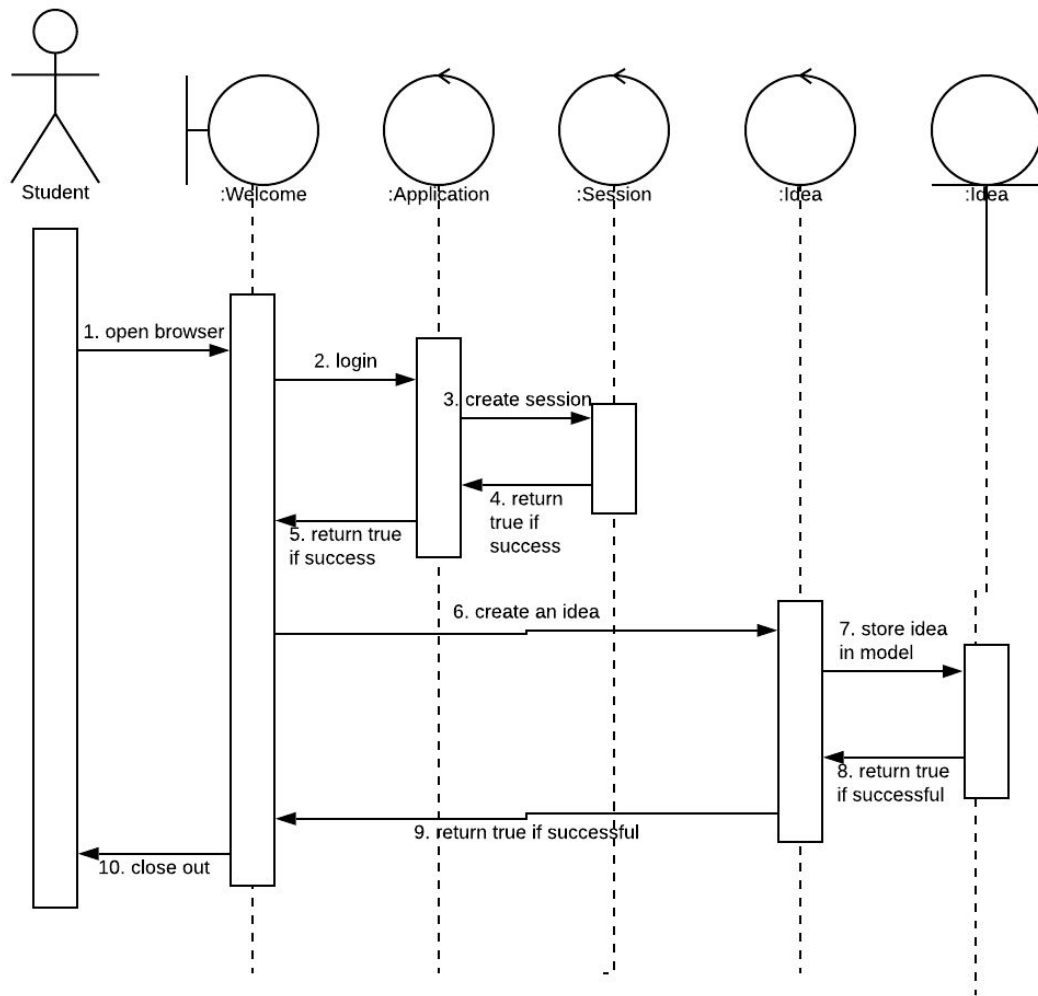
5. Detailed Design Class Diagram

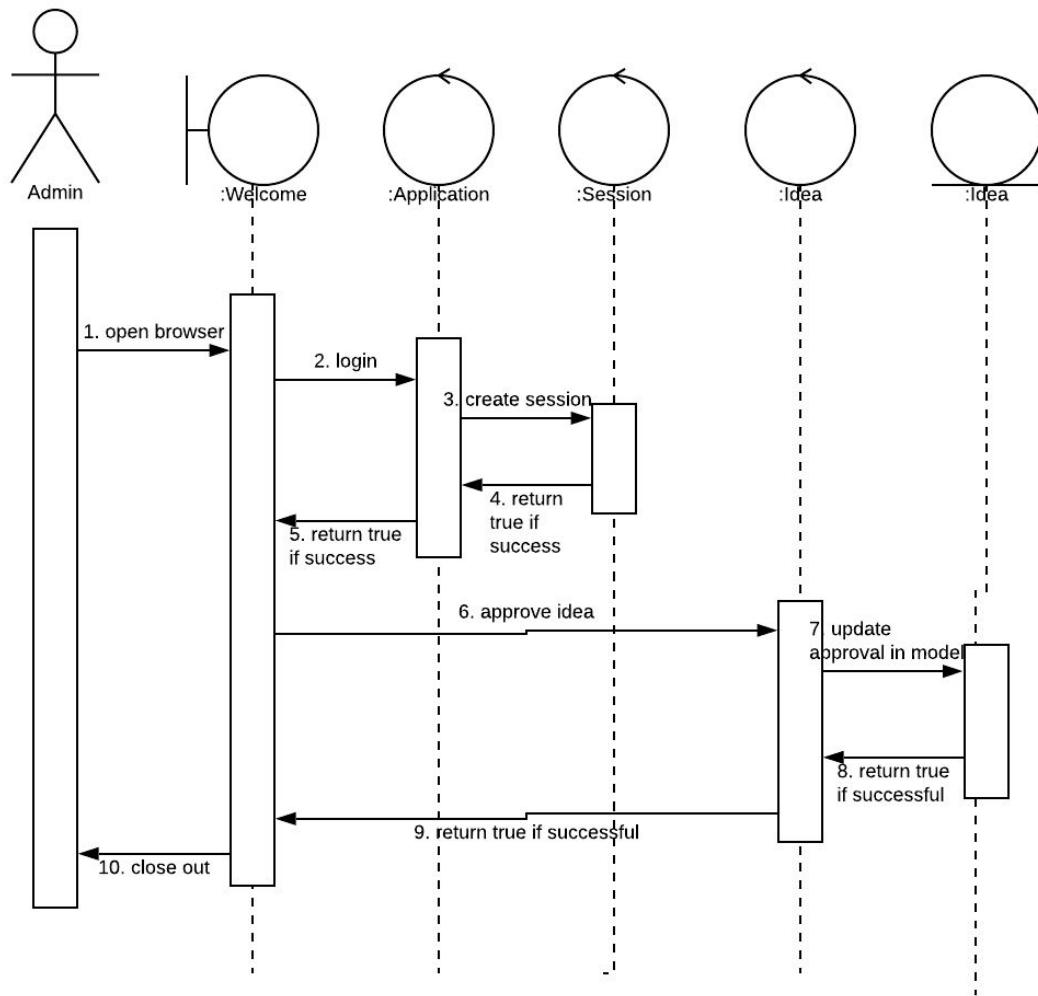


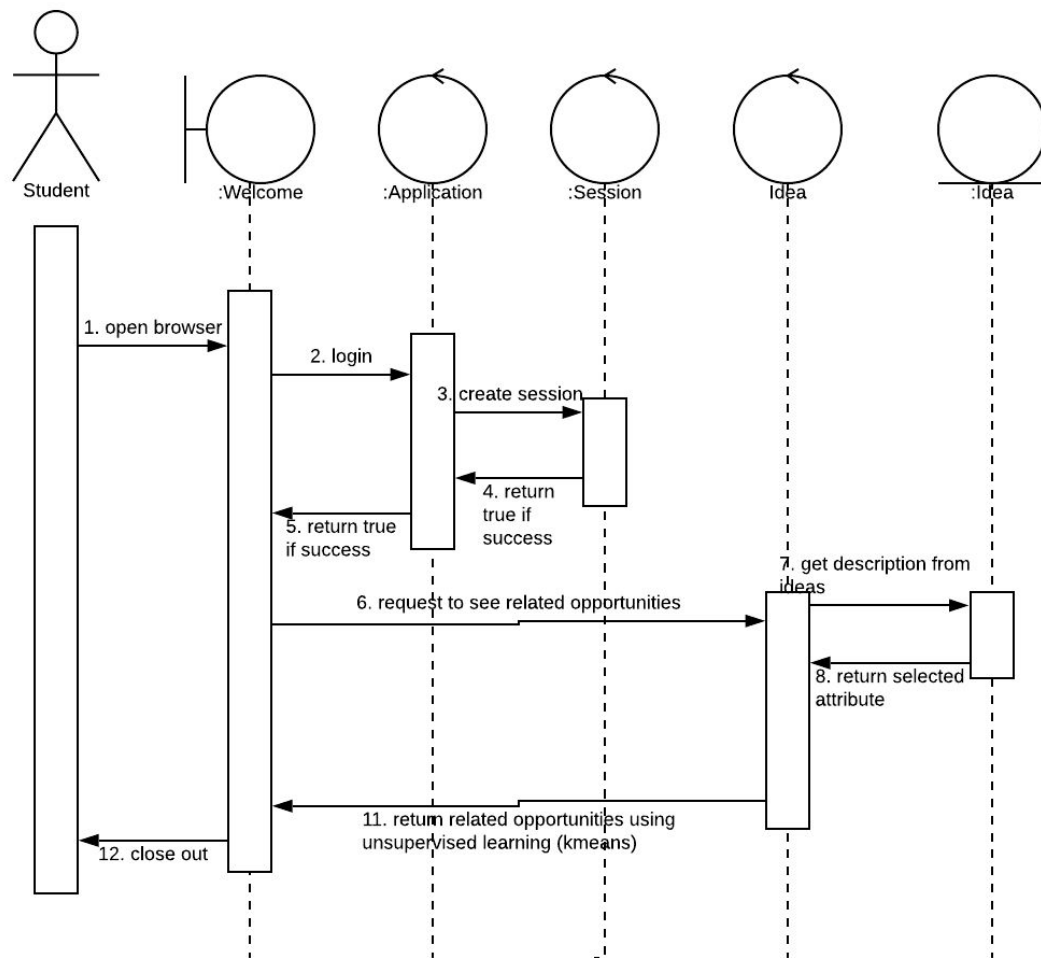
6. Statechart



7. System Sequence Diagrams








8. User Interface Mock-ups

← → ↺ ⬆

Title

http://lucidchart.com



Events Organizations Dashboard Explore Ideas altschm1@tcnj.edu

Related Ideas

Name	Issue	Approved
<div>View Edit Delete</div>		

Note: All expired events (with dates shown in red) will not be visible to the public, even if approved.

←


→

↻

🏠

http://lucidchart.com

Title



EventsOrganizationsDashboardYour Eventsaltschm1@tcnj.edu

New Ideas

Name *

On date

2019 ▾

April ▾

3 ▾

Start time

05 PM ▾

21 ▾

End time

05 PM ▾

21 ▾

Frequency

Address *

City *

State (00) *

Zip Code (00000) *

Transportation from TCNJ provided (Check if available)

☐

Description (Ex: Transportation details, recommended attire, contact information, etc.) *

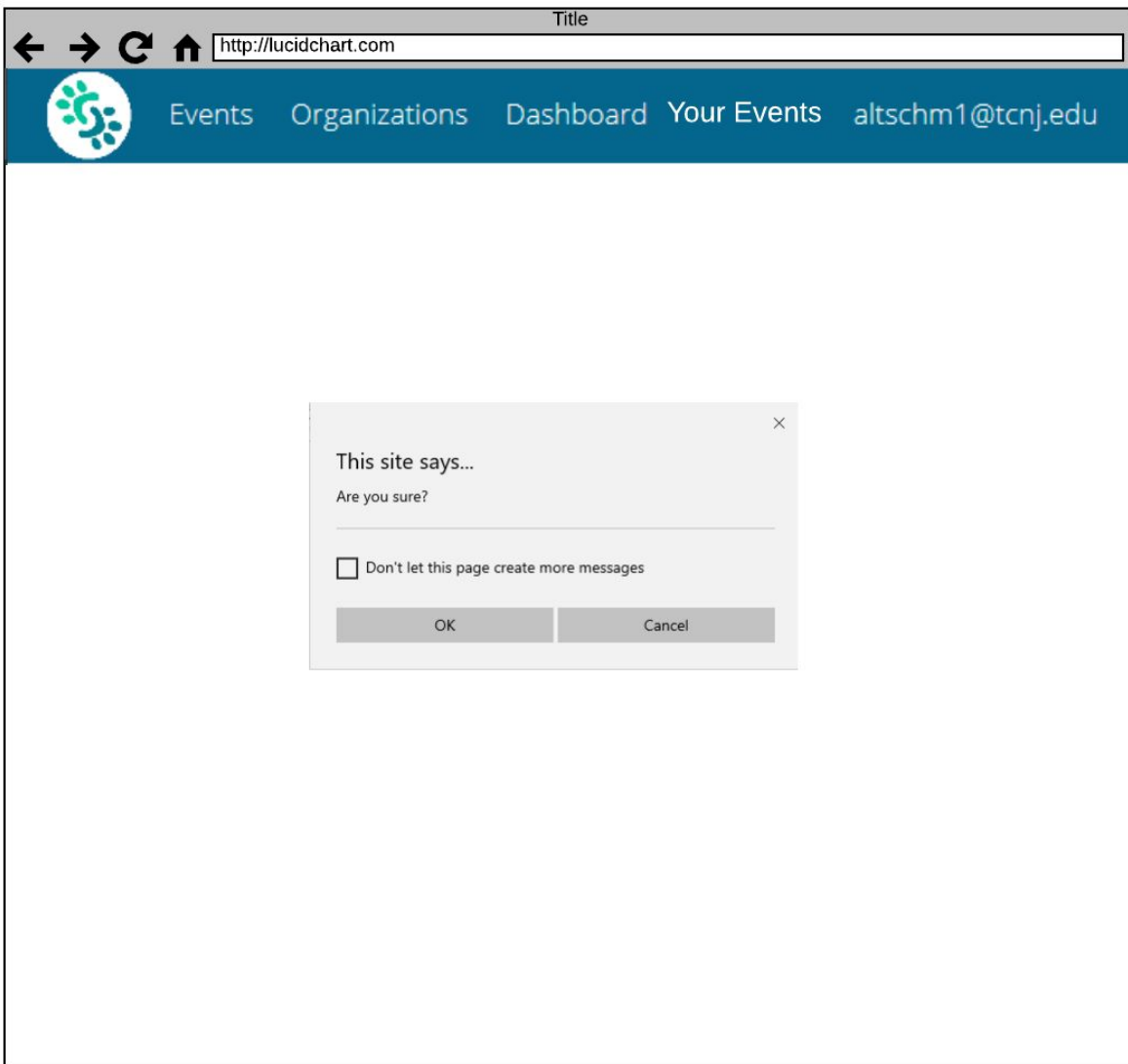
Issue area

Select One ▾

Note: Fields marked with * are required.

Create Opportunity


Back



← → ↺ ⬆

http://lucidchart.com

Title



Events Organizations Dashboard

altschm1@tcnj.edu

Your Ideas

NameIssueApproved

New


ViewEditDelete

Note: All expired events (with dates shown in red) will not be visible to the public, even if approved.

← → ↻ 🏠

http://lucidchart.com

Title



[Events](#)
[Organizations](#)
[Dashboard](#)
[Your Events](#)
altschm1@tcnj.edu

Idea Update

Name *

On date

2019 ▾

April ▾

3 ▾

Start time

05 PM ▾

21 ▾

End time

05 PM ▾

21 ▾

Frequency

Address *

City *

State (00) *

Zip Code (00000) *

Transportation from TCNJ provided (Check if available)

☐

Description (Ex: Transportation details, recommended attire, contact information, etc.) *

Issue area

Select One ▾

Note: Fields marked with * are required.

Create Opportunity

Back

The mockups fulfill requirements of visibility in the following ways. There are buttons and links that immediately lead to a new page that fulfill the required purposes. For example, the New button leads to a form that allows for the creation of a new page. The Edit button also leads to a form that allows for the field to be edited. The delete button sends an alert asking if you want the form to be deleted. The confirm button confirms all changes. There are no functionality that is not available to the user.

The mockups fulfill requirements of affordance in the following ways. The New button suggests that a new opportunity will be created. The show button suggests that the details of the opportunity will be shown. The edit button suggests that the details of the opportunity will be edited. The delete button suggests that the opportunity will be deleted.

Each mockup is also consistent with the already developed pages that have already been created.

There are no real shortcuts for frequent users, although frequent users could have their password saved or become more aware of where the links are.

Every button either sends an alert or sends to a link.

Every actions is sent with a confirmation.

There are error messages available with every form.

Every page comes with a back button.

The user's permissions are clearly defined, although the actual pages can't really be modified.

And most memory is retrieved from the database.

9. Requirements Report

Modularity and encapsulation is ensured as every entity is encapsulated as a model with a corresponding table and controller. For example, the user object is encapsulated as a user entity. Furthermore, there are many methods that encapsulate attributes such as `from_email()` or `set_email()` (along with many other). Even with modifications, encapsulation will be maintained. Any complex algorithms will be encapsulated in a controller class.

Most of the CRUD operations for models are already optimized by Ruby on Rails. With regards to the recommendation algorithm, it will be important to not keep excessive amount of memory stored (or else the webpage will significantly slow down). Using an implementation of k-means with bag-of-words can be time intensive, so it is important to only call those functions if the user selects on those modules.

Most of the data will be stored in a SQL table managed by Postgresql which is appropriate. Data will be retrieved from the SQL tables using ORM methods.

10. Test Case Design

For unit testing, I will be testing each individual method of every model and controller. For integration testing, I will be testing the set of methods of each class together (hence we will be testing components together). For system testing, I will be testing the entire system from start to finish and make sure each step is passed.

The tool that will be used for unit testing, is Test::Unit which is the default for Ruby on Rails. For system integration and integration testing, no tool will be used. Rather, the front end and back end will be tested manually to make sure everything works properly.

Functionality Tested	Inputs	Expected Output	Actual Output
Student logs in	Press 'Login'	Go to login page	
Student successfully logs in	Email Password Press 'login'	If successful, login Else, stay on same page	
Get to form to create idea	Press 'New' button	Go to page with blank form for opportunity	
Fill out form to actually create idea	Fill out all required fields (Name, Address, City, State, Zip Code, Description) with no conflicts of start time and end time Click 'Create Idea'	Idea is created Send user notification that it was successful	
Go to form a update idea	Click 'Edit' for Idea	Go to page with current form for idea	
Fill out form to update idea	Fill out all required fields (Name, Address, City, State, Zip Code, Description) with no conflicts of start time and end time Click 'Update Idea'	Idea is updated Send user notification that it was successful	
List all own idea	Click on 'Dashboard'	Display entire list of current ideas	
Delete own ideas	Click 'Delete' for	Your selected	

	idea	opportunity is deleted after receiving warning	
Show information for own idea	Click 'View' for Idea	Details for chosen Idea is shown	
See related ideas to idea that self created	Click 'Explore Ideas'	Show list of related ideas	

11. Analysis of OSS License

Per <https://opensource.org/licenses>, there are several types of open source license. There are the GNU General Public License, the BSD License, and the MIT License.

The GNU General Public License version 3

The purpose of this license is intended to allow users to share and change the code as desired. This code is meant to be free in terms of use, not price. Developers can charge people to use their code with this license. However, these developers must respect how other people use their code and respect their freedom. Furthermore, the code comes with no warranty and cannot be patented.

2-Clause BSD License

The purpose of the license is that any code (compiled or not) with this license must continue to use these license for the entire duration of the project. Also modifications of the code means that the original developer's name can't be used to endorse it.

MIT License

The MIT License is very short and simple and allows people to take the code and use it, modify it, publish, distribute, and sell without any restrictions. This guarantees complete freedom. Servd currently uses the MIT License, and this license feels best for its simplicity and how it guarantees complete freedom to anybody who uses this code.

12. OSS Maintenance

The code base is currently being maintained on GitHub. GitHub is able to manage issues (tasks that need to be performed) and milestone (a set of tasks that are related and mark a significant event). Communication is facilitated through the issues which designates tasks to be done. Determining which code is accepted is determined by the CELR staff and what they are looking for from the project as they are the major stakeholder in this project. Therefore, these are the people who ultimately determine if the code will be merged into master. Currently, there is no place where they discuss potential ideas for future iterations (such as a backlog) that doesn't navigate through the issues. Ultimately, there are many ways that maintenance to the code can be improved with better communication protocols. Therefore, these are the steps that

I will take to ensure proper communication and that my code gets accepted onto the master branch:

- Contact CELR staff with modifications that will be made to the project and their feedback on those updates
- Create a backlog file on the source code that states the following:
 - Thing to do
 - Type: (New Idea/Enhancement/Bug/Customer Feedback/Security/Other)
 - Priority (Low/Medium/High)
 - In Progress (Yes/No)
 - Expected Completion Date (MM/DD/YYYY)
- Organize backlogs through issues and milestones
 - Issues: tasks on the backlog
 - Milestones:
 - A set of related issues
 - Problem to be solved
 - User Story: When I ____, I want to be able to ____, so that I can ____.
 - Evaluation Metric
 - What is in scope and out of scope
 - Expected Done date
- Keep all code well documented
- Push the code