University of California San Diego
CSE 110 - Fall 2020

Design Use Cases

Philanthropy Connect

Team: **QuaranTeam**

Unduwap KandageDon - Project Manager
Branson Beihl - Database Specialist
Vivian Chiong - Senior System Analyst
John Ge - Software Development Lead
Patrick Jorgensen - Quality Assurance Lead
Daniel Kubeck - Business Analyst
Andrew Liang - Software Architect
Tri Truong - Algorithm Specialist
Danny Vo - Software Architect
Angela Wang - User Interface Specialist
Dominick Lee - Software Development Lead

Table of Contents

Legend

Glossary

Design Use Cases

DUC 6 - Individuals to Organization Connect

DUC-6.1: Toggle Follow Organization	1	**C**
DUC-6.2: View Organization's Information	1	**C**
DUC-6.3: View Event's Information	1	**C**

Legend

Priority

1	Must Have: Main features that make the application a minimum viable product
2	Should Have: Feature that would add useful functionality to the application
3	Nice to Have: Features that are not essential to the application's goal

Status

D: Dropped
NS: Not Started
IP: In Progress
C: Complete

Test Phase

NS: Not Started
IP: In Progress
C: Complete

Glossary

Users	The users of the application, including both organizations and individuals
Organizations	Users representing nonprofits, companies, or philanthropic groups
Individuals	Individual users of the app aiming to support a philanthropic cause
Cause	The philanthropic category that aligns with the organizations and their events. Causes could be general (health), specific (lung cancer), or religious (Christian).
Skills	These are skills that the individuals would have. Skills include knowing how to cook, knowing how to sew, knowing how to draw, knowing how to give speeches, and many more. These also include having resources such as time, money, or a car.

Technical Term Highlights

Color & Example	Description
React	A library, module, algorithm, or important concept.
GET	REST API call method (GET/POST/PUT/DELETE)
`api/events/create`	REST API call endpoint.
users	Database table.

DUC-1.1: Register

Description:
The user registers for their account by linking their Google or Facebook account.
Actors:
The user, an individual or organization, of the application.
User Goals:
The user wants to register for an account to start using the application.
Desired Outcome:
The user logged in via their 3rd party account and is ready to continue to profile creation.
Dependent Design Use Cases:
None
Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-1.1
Pre-Conditions:
The user has gone to the website.
Post-Conditions:
The user is redirected to create their profile and has been differentiated into an individual user or an organization user.
Trigger:
The user wants to use the application.

Workflow for Design Use Case:

1. The frontend shall render the login screen.
2. The user shall click on the "Continue with Google" or "Continue with Facebook" button.
3. The frontend shall render the interface with the 3rd party login platform through **Firebase SDK**.
4. The user shall log in to their 3rd party account, and the interface closes.
5. The frontend shall retrieve the user token, which is a JSON Web Token, or **JWT** in short, and keep it throughout the entire user session for authentication purposes.
6. The frontend shall send a **GET** request to the **`/api/profile/:id`** endpoint where id is the user id decoded from the **JWT**.
7. The backend shall query the *individuals* and *organizations* table through **Firebase Cloud Firestore**.
8. The backend shall return http status **404** because the user has not created a profile.
9. The frontend shall ask the user if they would like to be an individual or an organization.
10. The user shall select one of the two buttons corresponding to the options.
11. The frontend shall redirect the user to the profile creation page corresponding to their choice.

Alternate Workflow:
The user logs in to an account with a profile already created.

1. Main workflow steps 1 - 7.
2. The backend shall return a **JSON** (http status **200**) because the user has already created a profile.
3. The frontend shall redirect the user to the home page.

DUC-1.2: Login

Description:
The user logs into their account using their registered account.
Actors:
The user, an individual or organization, of the application.
User Goals:
The user wants to login to their account to start using the application's features.
Desired Outcome:
The user is logged into their account.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-2.1: Create User Profile

Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-1.2
Pre-Conditions:
The user has created a profile.
Post-Conditions:
The user is redirected to the home page.
Trigger:
The user wants to use the application.

Workflow for Design Use Case:

1. The user shall go to the website.
2. The frontend shall render the login screen.
3. The user shall click on the "Continue with Google" or "Continue with Facebook" button.
4. The frontend shall render an interface with the 3rd party login platform through **Firebase SDK**.
5. The user shall log in to their third party account, and the interface closes.
6. The frontend shall retrieve the user token, which is a JSON Web Token, or **JWT** in short, and keep it throughout the entire user session for authentication purposes.
7. The frontend shall send a **GET** request to the **`/api/profile/:id`** endpoint where id is the user id decoded from the **JWT**.
8. The backend shall query the *individuals* and *organizations* table through **Firebase Cloud Firestore**.
9. The backend shall return the result containing their profile as a **JSON**.
10. The frontend shall redirect the user to the home page.

Alternate Workflow:
The user logs in to an account without a profile created.

1. Main workflow steps 1 - 8.
2. The backend shall return http status **404** because the user has not created a profile.
3. The frontend shall ask the user if they would like to be an individual or an organization.
4. The user shall select one of the two buttons corresponding to the options.
5. The frontend shall redirect the user to the profile creation page corresponding to their choice.

DUC-1.3: Logout

Description:
The user logs out of their account.
Actors:
The user, an individual or organization, of the application.
User Goals:
The user wants to log out of their account.
Desired Outcome:
The user is logged out of their account.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile
Priority:
1 - Must Have

Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-1.3
Pre-Conditions:
The user is logged in to their account and is on the home page.
Post-Conditions:
The user is logged out and is redirected to the login screen.
Trigger:
The user wants to log out of their account.

Workflow for Design Use Case:

1. The user shall click on the "Log out" button.
2. The frontend shall log out the user through **Firebase SDK**.
3. The frontend shall redirect the user to the login screen.

Alternate Workflow: N/A

DUC-1.4: Continue as Guest

Description:
The user wants to continue as a guest to use the application.
Actors:
The individual guest user of the application.
User Goals:
The user wants to continue as a guest to use parts of the application's functionality.
Desired Outcome:
The user is able to use the application to search for organizations.
Dependent Design Use Cases:
None
Priority:
3 - Nice to have
Progress Status:
Dropped
Test Phase Status
Dropped
Requirements:
SR-1.4
Pre-Conditions:
None
Post-Conditions:
The user is redirected to the guest home page.
Trigger:
The user wants to use the application.

Workflow for Design Use Case:

1. The user shall go to the website.
2. The frontend shall render the login screen.
3. The user shall click on the "Continue as Guest" button.
4. The frontend shall render the guest home page.

Alternate Workflow: N/A

DUC-2.1: Create User Profile

Description:
The user creates their profile in the application.
Actors:
The user, an individual or organization, of the application.
User Goals:
The user wants to create their profile.
Desired Outcome:
The user fills in their information in their account profile.
Dependent Design Use Cases:
DUC-1.1: Register
Priority:
1 - Must Have

Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-2.1
Pre-Conditions:
The user has successfully registered for an account and is on the profile creation page.
Post-Conditions:
The user's profile information is stored.
Trigger:
The user wants to use the application and its personalized features.

Workflow for Design Use Case:

1. The user shall fill in the form fields they choose based on if they are **individual** or **organization** user: for **individual** users, these include the mandatory fields "First Name", "Last Name", "Zip", "Age", and optional fields "Causes", "Skills"; for **organization** users, these include the mandatory fields "Title", "Mission", "Zip", "Contact", "URL" and optional field "Causes".
2. The user shall click the "Create" button.
3. The frontend shall validate the user's form.
4. The frontend shall send a **POST** request to the **`/api/profile/create`** endpoint with the user form populated.
5. The backend shall validate the user's **JWT** and the form in the payload.
6. The backend shall insert the user profile object into the *individuals* or *organizations* table through **Firebase Cloud Firestore**.
7. The backend shall return http status **200**.
8. The frontend shall redirect the user to the home page.
9. The frontend shall display a message acknowledging the profile creation.

Alternate Workflow:
The user enters one or more invalid fields including leaving mandatory fields blank.

1. The user shall incorrectly fill in the form fields they choose based on if they are **individual** or **organization** user: for **individual** users, these include the mandatory fields "First Name", "Last Name", "Zip", "Age", and optional fields "Causes", "Skills"; for **organization** users, these include the mandatory fields "Title", "Mission", "Zip", "Contact", "URL" and optional field "Causes".
2. The frontend shall detect one or more invalid fields, highlight them, and display a message telling the user to rectify the invalid highlighted fields.
3. The user shall rectify the invalid fields.
4. Main workflow steps 1 - 9.

DUC-2.2: Edit User Profile

Description:
The user edits their profile.
Actors:
The user, an individual or organization, of the application.
User Goals:
The user wants to edit the information in their profile.
Desired Outcome:
The user updates their information in their profile.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile
Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-2.1
Pre-Conditions:
The user is on the home page.
Post-Conditions:
The user's profile information is updated.
Trigger:
The user wants to edit the information in their profile.

Workflow for Design Use Case:

1. The user shall click on the "Profile" button.
2. The frontend shall display the Edit User Profile page.
3. The user shall edit their information in the fields they choose based on if they are **individual** or **organization** user: for **individual** users, these include the mandatory fields "First Name", "Last Name", "Zip", "Age", and optional fields "Causes", "Skills"; for **organization** users, these include the mandatory fields "Title", "Mission", "Zip", "Contact", "URL" and optional field "Causes".
4. The user shall click "Save" at the bottom of the page.
5. The frontend shall validate the form.
6. The frontend shall send a **POST** request to **`/api/profile/:id`** endpoint with the form, where id is the user id decoded from the **JWT**.
7. The backend shall validate the user's **JWT** and the form in the payload.
8. The backend shall update the user profile in the *individuals* or *organizations* table through **Firebase Cloud Firestore**.
9. The backend shall return http status **200**.
10. The frontend shall redirect the user to the home page and display a pop-up message: "Success", "Profile updated".

Alternate Workflow:
The user provides invalid input for one or more of the fields.

1. Main workflow steps 1 - 2.
2. The user shall enter invalid input in one or more editable fields.
3. The frontend shall detect one or more invalid fields and display a message telling the user to rectify the invalid fields.
4. Main workflow steps 3 - 10.

The user is on the home page and wants to edit the profile without clicking on the Profile tab.

1. The user shall click the “Edit” button on the bottom of the profile card located at the left side of the home page.
2. Main workflow steps 2 - 10.

The user leaves the page after editing without saving preferences.

1. Main workflow steps 1 - 3.
2. The user shall click any button that takes the user off of the Edit User Profile page.
3. The frontend shall redirect the user to the relevant page.

DUC-3.1: Filter Organizations by Cause and Distance

Description:
The individual filters their exploration for organizations by causes and distance.
Actors:
The individual user of the application.
User Goals:
The individual wants to filter their exploration for the organizations based on their preferred causes and distance.
Desired Outcome:
The individual specifies explore filters and the exploration is narrowed down.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile

Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-3.1
Pre-Conditions:
The individual is on the home page.
Post-Conditions:
The system displays an updated list of organizations according to the filtered query.
Trigger:
The individual wants to find organizations.

Workflow for Design Use Case:

1. The user shall click on the "Find Org" button.
2. The user shall click on the “Causes” or “Distance” button at the right side of the navbar.
3. The frontend shall display a pop-up with filter options to choose from; in the Distance case, the user shall have a slider to specify the search radius which defaults to 100km.
4. The user shall select “Causes” and pick ones that align with them; they shall have the option to select nothing at all which will filter based on the causes listed on the user’s profile page.
5. The user shall move the slider for distance in the case that they are filtering distance.
6. The user shall click “Confirm”.
7. The frontend shall send a **POST** request to the **`/api/organization/filter`** with a **url-extended** form containing the filters and user’s **JWT**.
8. The backend shall validate the user’s **JWT** and the filters in the payload.
9. The backend shall query the *organizations* table through **Firebase Cloud Firestore**.
10. The backend shall return the result containing a list of profiles as a **JSON**.
11. The frontend shall render the organization cards based on the selected filter queries.

Alternate Workflow: N/A

DUC-3.2: Filter Events by Skills and Distance

Description:
The individual filters their exploration for events based on the distance from their location and skills they have chosen.
Actors:
The individual user of the application.
User Goals:
The individual wants to filter their exploration for events based on the distances between their location and the organizations' and skills.
Desired Outcome:
The individual specifies explore filters and the exploration is narrowed down.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile

Priority:
2 - Should Have
Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-3.2
Pre-Conditions:
The individual user is on the home page.
Post-Conditions:
The system displays an updated list of events according to the filtered queries.
Trigger:
The individual user wants to find events.

Workflow for Design Use Case:

1. The user shall click on the "Find Event" button.
2. The user shall click on the "Skills" button.
3. The frontend shall display a pop-up with skills options to choose from. If the user wishes not to specify skills, they may just click "Confirm" and the search will take the user profile skills as the parameter.
4. The user shall select from a list of skills rendered by the frontend, including the option to select "My Skills" which will filter based on the skills listed on the user's profile page.
5. The user shall click "Confirm".
6. Main workflow steps 10-14 will run, updating the search.
7. The user shall click "Distance".
8. The user shall move the slider to specify the search radius, which defaults to 100km.
9. The user shall click "Confirm".
10. The frontend shall send a **GET** request to the **`/api/organization/filter`** with a **url-extended** form containing the filters and user's **JWT**.
11. The backend shall validate the user's **JWT** and the filters in the payload.
12. The backend shall query the *organizations* table through **Firebase Cloud Firestore** based on the distance between the user and organization and the skills.
13. The backend shall return the result containing a list of profiles as a **JSON**.
14. The frontend shall render the event cards based on the selected filter queries.

Alternate Workflow: N/A

DUC-4.1: Manage Events (Organization)

Description:
The organization user views their news feed.
Actors:
The organization user of the application.
User Goals:
The organization user wants to see events they have posted.
Desired Outcome:
The organization user sees their events on their feed.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile
Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status:
Complete
Requirements:
SR-4.1
Pre-Conditions:
The organization user has logged in.
Post-Conditions:
The frontend displays the user's news feed.
Trigger:
The organization user wants to see their events.

Workflow for Design Use Case:

1. The organization user shall click on the "Home" page button from the navbar.
2. The frontend shall send a **GET** request to the **`/api/organization/feed?type=organization`** endpoint with the user's **JWT** in the payload.
3. The backend shall validate the user's **JWT** in the payload.
4. The backend shall query the *events* table through **Firebase Cloud Firestore** to find all events posted by the user.
5. The backend shall return the result containing a list of events as a **JSON**.
6. The frontend shall display a list of the organization user's events on the news feed.

Alternate Workflow:
The organization user has not created any event posts yet.

1. Main workflow step 1 - 5.
2. The frontend shall display a message to the organization user on the news feed: "Create some events to see them here!".

DUC-4.2: Discover Following Organization's Events (Individual)

Description:
The individual user will view events of the organizations they follow on their news feed.
Actors:
The individual user of the application.
User Goals:
The individual user wants to see events posted by organizations they are following.
Desired Outcome:
The individual user sees the events posted on their news feed.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile, DUC-6.1: Toggle Follow Organization
Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status:
Complete
Requirements:
SR-4.2
Pre-Conditions:
The individual user has logged in.
Post-Conditions:
The frontend displays the individual's news feed.
Trigger:
The individual user wants to see events of the organizations they follow on their news feed.

Workflow for Design Use Case:

1. The individual shall click on the "Home" page button from the navbar.
2. The frontend shall send a **GET** request to the **`/api/organization/feed?type=individual`** with the user's **JWT** in the payload.
3. The backend shall validate the user's **JWT** in the payload.
4. The backend shall query the *events* table through **Firebase Cloud Firestore** to find all events posted by the organizations followed by the user.
5. The backend shall return the result containing a list of events as a **JSON**.
6. The frontend shall display a list of events posted by the organizations the user has followed.

Alternate Workflow:
The individual has not followed any organizations yet.

1. Main workflow step 1 - 5.
2. The frontend shall display a message to the individual user on the news feed: "Follow some orgs to see their events here!".

DUC-5.1: Create Event

Description:
The organization user makes a post about a specific event, where volunteers are needed for specific tasks.
Actors:
The organization user of the application.
User Goals:
The organization user wants to make their events public and visible to students.
Desired Outcome:
The organization user has made a post about an event.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile, DUC-4.1: Manage Events (Organization)

Priority:
1 - Must Have
Requirements:
SR-5.1
Progress Status:
Complete
Test Phase Status
Complete
Pre-Conditions:
The organization user is on the home page.
Post-Conditions:
The system displays the organization's event post on an individual's news feed.
Trigger:
The organization user wishes to share information about their event.

Workflow for Design Use Case:

1. The user shall click on the "Post" tab.
2. The frontend shall display the create a new event page.
3. The organization user shall enter required fields on information about their event, including the fields "Title", "Details", "Zip", "Date", and the optional field "Skills".
4. The user shall then click "Create".
5. The frontend shall send a **POST** request to **`/api/events/create`** endpoint with the form and user's **JWT**.
6. The backend shall validate the user's **JWT** and the form in the payload.
7. The backend shall insert the event in the *events* table through **Firebase Cloud Firestore**.
8. The backend shall return HTTP response status 200 to signify the creation was successful.

Alternative Workflow:
The organization user does not complete the required fields of the event post.

1. Main workflow steps 1 - 2.
2. The organization user shall not fill in all required fields.
3. The frontend shall display a warning pop-up message to the organization user that they have not filled out all the required fields.
4. The organization user shall rectify the invalid fields.
5. The frontend shall clear the pop-up.
6. Mainwork flow 3 - 8.

The organization user discards the event post by going to the home page.

1. Main workflow steps 1 - 3.
2. The organization user shall click on the "Home" page button.
3. The frontend shall redirect the user to the home page.

The organization user discards the event post by going to the user's profile page.

1. Main workflow steps 1 - 3.
2. The organization user shall click on the "Profile" page button.
3. The frontend shall redirect the user to the user's profile page.

DUC-5.2: Edit Event

Description:
The organization user edits a post that they have posted.
Actors:
The organization user of the application.
User Goals:
The organization user wants to make changes to an existing event post.
Desired Outcome:
The organization user has made the changes they wish to the event post.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile, DUC-5.1: Create Event, DUC-4.1: Manage Events (Organization)
Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-5.2
Pre-Conditions:
The organization user is on the home page.
Post-Conditions:
The event's information is updated and is correctly displayed on the organization user's news feed.
Trigger:
The organization user wants to change something in an existing event post they have made.

Workflow for Design Use Case:

1. The organization user shall click the “Edit / Delete” button on the event they want to edit.
2. The frontend shall display the edit/delete event popup.
3. The organization user shall edit required fields on information about their event, including the fields "Title", "Details", "Zip", "Date", and the optional field "Skills".
4. The organization user shall then click “Save”.
5. The frontend shall send a **PUT** request to **`/api/events/:id`** endpoint with the form, where id is the event id retrieved from the selected event object in memory.
6. The backend shall validate the user’s **JWT** and the form in the payload.
7. The backend shall update the user’s event in the *events* table through **Firebase Cloud Firestore**.
8. The backend shall return http status **200**.
9. The frontend shall display the updated post on the user’s news feed.

Alternate Workflow:
The organization user does not complete the required fields of the event post.

1. Main workflow steps 1 - 2.
2. The organization user shall not fill in all required fields.
3. The frontend shall display a warning pop-up message to the organization user that they have not filled out all the required fields.
4. The organization user shall rectify the invalid fields.
5. The front end shall hide the pop-up.
6. Mainwork flow 3 - 9.

The organization user leaves the edit event page and discards the event post.

1. Main workflow steps 1 - 3.
2. The organization user shall click on the "X" button.
3. The frontend shall redirect the user to the news feed.

DUC-5.3: Delete Event

Description:
The organization user deletes an event post that they have posted.
Actors:
The organization user of the application.
User Goals:
The organization user wants to delete an existing event post.
Desired Outcome:
The organization user has deleted the event post.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile, DUC-5.1: Create Event, DUC-4.1: Manage Events (Organization)

Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status:
Complete
Requirements:
SR-5.3
Pre-Conditions:
The organization user is on the home page.
Post-Conditions:
The system deletes the event.
Trigger:
The organization user wants to delete a post they have made.

Workflow for Design Use Case:

1. The organization user shall click the “Edit / Delete” button on the event they want to delete.
2. The frontend shall display the edit/delete event popup.
3. The organization user shall click “Delete” at the bottom of the popup.
4. The frontend shall send a **DELETE** request to **`/api/events/:id`** endpoint with the form, where id is the event id retrieved from the selected event object in memory.
5. The backend shall validate the user’s **JWT** in the payload.
6. The backend shall delete the user’s event in the *events* table through **Firebase Cloud Firestore**.
7. The backend shall return http status **200**.
8. The frontend shall display the updated event feed.

Alternate Workflow: N/A

DUC-6.1: Toggle Follow Organization

Description:
The individual user follows or unfollows an organization.
Actors:
The individual user of the application.
User Goals:
The individual user gets more exposure to opportunities from a desired organization when they explore, and is able to revert the exposure if wanted.
Desired Outcome:
The individual user follows the organization.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile, DUC-3.1: Filter Organizations by Cause and Distance
Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status:
Complete
Requirements:
SR-6.1i, SR-6.1ii
Pre-Conditions:
The individual user is on the Find Org page and has explored for organizations.
Post-Conditions:
The system stores the organization in the following field in the individual user's account information.
Trigger:
The individual user wants to follow and see the organization's events on the news feed, or the user who already follows an organization wants to unfollow to remove the events from their news feed.

Workflow for Design Use Case:

1. The individual user shall press the follow organization "+" button on the organization card they want to follow.
2. The frontend shall send a **POST** request to **`/api/organization/:id?follow=true`** endpoint where id is the organization id retrieved from the selected organization object in memory.
3. The backend shall validate the user's **JWT** and the form in the payload.
4. The backend shall update the user's following organization id's in the *individuals* table through **Firebase Cloud Firestore**.
5. The backend shall return http status **200**.
6. The frontend shall change the button image from the "+" to the "check mark" and display a pop-up message: "Success", "Organization followed".

Alternate Workflow:
The individual already followed the organization and wants to unfollow the organization.

1. The individual user shall press the unfollow organization "check mark" button on the organization card they want to unfollow.
2. The frontend shall send a **POST** request to **`/api/organization/:id?follow=false`** endpoint where id is the organization id retrieved from the selected organization object in memory.
3. Main workflow step 3 - 5.
4. The frontend shall change the button image from the "check mark" to the "+" and display a pop-up message: "Success", "Organization unfollowed".

DUC-6.2: View Organization's Information

Description:
The individual user views the organization's information.
Actors:
The individual user of the application.
User Goals:
The individual user wants to view an organization's information and learn how to connect with the organization.
Desired Outcome:
The individual user sees an organization's information.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile, DUC-3.1: Filter Organizations by Cause and Distance
Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-6.2
Pre-Conditions:
The individual user is on the Find Org page and has explored for organizations.
Post-Conditions:
The system displays the organization's information.
Trigger:
The individual user wants to learn more about the organization.

Workflow for Design Use Case:

1. The user shall click on an organization card.
2. The frontend shall send a **GET** request to the **`/api/profile/:id?type=organization`** where id is the organization profile's id, extracted from the organization object from the existing list.
3. The backend shall query *organizations* table through **Firebase Cloud Firestore**.
4. The backend shall return the result containing their profile as a **JSON**.
5. The frontend shall render the organization profile page as a modal pop-up.
6. The user shall click the X at the top left of the pop-up to return to the organization search results.

Alternate Workflow: N/A

DUC-6.3: View Event's Information

Description:
The individual user views the event's information.
Actors:
The individual user of the application.
User Goals:
The individual user wants to view the event's information.
Desired Outcome:
The individual user can view an event's information.
Dependent Design Use Cases:
DUC-1.1: Register, DUC-1.2: Login, DUC-2.1: Create User Profile, DUC-3.2: Filter Events by Skills and Distance, DUC-6.1: Toggle Follow Organization
Priority:
1 - Must Have
Progress Status:
Complete
Test Phase Status
Complete
Requirements:
SR-6.3
Pre-Conditions:
The individual user is on the Find Event page and has explored for events, or is on the home page and has followed organizations with events.
Post-Conditions:
The system displays the event's information.
Trigger:
The individual user wants to learn more about the event.

Workflow for Design Use Case:

1. The user shall click anywhere on an event card.
2. The frontend shall send a **GET** request to the **`/api/events/:id`** where id is the event's id, extracted from the event object from the existing list.
3. The backend shall query the *events* table through **Firebase Cloud Firestore**.
4. The backend shall return the result containing the event information as a **JSON**.
5. The frontend shall render the event information page as a modal pop-up.
6. The user shall click the X at the top left of the pop-up to return to the event search results.

Alternate Workflow: N/A