Key use cases are those that will be implemented this semester: “Create Report”, “Validate Input”, “Access Location”, and “Check Weather Conditions”.

Use case: *CreateReport*

Primary actor: User

Goal in context: To create a report of a homeless person for submission

Preconditions: User has access to internet and web application

Trigger: User logs on to web app and selects report module

Scenario:

1. User observes homeless person in cold conditions
2. User logs onto web application
3. User chooses report module
4. User enters relevant information
5. Report is stored in database
6. User receives confirmation after submission

Exceptions:

The form is planned to check for valid information dynamically during entering rather than after submission.

1. Weather conditions do not meet code blue condition: form is not accessible
2. User location is outside of the city of Philadelphia: User cannot submit form, receives message
3. User enters invalid information: form displays error message

Priority: Essential, application cannot exist without report submission

When available: After second increment (Assignment 3)

Frequency of use: Often, essential aspect of application

Channel to actor: Via web application

Secondary actors: Data verification use cases

Channels to secondary actors: Within the application through functional calls

Open issues:

1. How will reports be stored and parsed?
2. Should all information be entered on the same page, or should the form have multiple pages?
3. How should error messages be displayed?

Use case: *ValidateInput*

Primary actor: *CreateReport* use case

Goal in context: Validate user input for form submission

Preconditions: User has chosen to create a report

Trigger: User enters information in form

Scenario:

1. User enters information in a field of report form
2. Form invokes *ValidateInput*
3. Other use cases are invoked as needed for validation
4. Validity of input is determined
5. Validity is retuned to caller

Exceptions:

1. APIs are down: secondary validation is needed
2. Input is invalid: caller is notified of invalid input

Priority: High, robust input validation is necessary for a robust application

When available: After increment 3

Frequency of use: Every time a new field is populated

Channel to actor: Functional calls

Secondary actors: *CheckWeather*, *AccessLocation*

Channels to secondary actors: Functional calls

Open issues:

1. **How to validate address inputs? Use a NLP API? Mapping API?**
2. **What to do if a necessary API is down?**

Use case: *AccessLocation*

Primary actor:

Goal in context: Use geolocation API to facilitate address input

Preconditions: User has begun to enter form

Trigger: Location field of form is accessed

Scenario:

1. User starts report

2. User enters location of homeless person

Exceptions:

1. User tries to enter location out of Philadelphia: an exception is returned to caller
2. API is not working: unable to access location

Priority: High, location is needed for the reports

When available: At least third iteration

Frequency of use: Every time a report is made

Channel to actor: Function calls

Secondary actors: Location API

Channels to secondary actors: API calls

Open issues:

1. How will location be entered (address, map, etc.)?
2. Will IP geolocation be used to validate location input to avoid abuse from users out of Philadelphia?
3. How to mitigate API issues?

Use case: *CheckWeather*

Primary actor: *ValidateInput*

Goal in context: Check if conditions are met for code blue within the applications location boundary

Preconditions: User is creating a report

Trigger: User is creating a report and location is entered, then *ValidateInput* checks the weather in the location

Scenario:

1. User is already creating a report
2. User enters location of homeless person
3. *CreateReport* invokes *ValidateInput*
4. *ValidateInput* invokes *CheckWeather*
5. A weather API is called to check the weather status
6. Validity of weather conditions is returned

Exceptions:

1. Weather API is down: an alternative must be used
2. Weather conditions do not meet code blue standards: Invalid weather returned

Priority: Medium, the application can function without weather checking, but it helps prevent invalid reports

When available: At least iteration three

Frequency of use: Every time a form is created

Channel to actor: Via function calls

Secondary actors: Weather API

Channels to secondary actors: API calls

Open issues:

1. What to do if the weather API is down?