

# ECS 160, Assignment 1: Project Use Cases

Daniel Woodall, Matthew Leung, Ryan Loughlin, Crystal Tse, Zhaoping Yan  
April 10, 2011

**Goal:** User receives and acts upon person-to-person Place-it To-do.

**Initial state:** User arrives at a location with a Place-it To-do.

**Final state:** User completes To-do, and marks it as done, or declines it for this visit.

**Description:**

1. User moves towards a specific location with a To-do.
2. As user nears the location, system displays the header of To-do (or list of To-do headers).
3. User selects To-do.
4. System displays the To-do's details.
5. (If user declines To-do)
  - 5.1. User hits "Decline To-do" button.
  - 5.2. System notifies the creator of the To-do (if someone other than user) that To-do has been declined.
  - 5.3. System saves To-do, but it removes it from user's list, for this visit.
6. (If user accepts To-do)
  - 6.1. User hits "Accept To-do" button.
  - 6.2. System displays confirmation: "Are you sure you gonna do the To-do?"
  - 6.3. User confirms.
  - 6.4. System removes To-do from the user's list.
  - 6.5. System notifies the creator of the To-do (if someone other than user) that To-do has been done.
7. (If user ignores To-do)
  - 7.1. System alerts the user if the user moves away from the area without accepting or declining To-do.

**Goal:** User creates or edits a message for a particular person, at a particular location.

**Initial state:** User wants to create or edits a Place-it To-do.

**Final state:** User saves a Place-it To-do.

**Description:**

1. User wants to create or edit a Place-it To-do.
2. (If user wants to create a To-do)
  - 2.1. User selects "Create To-Do" button.
  - 2.2. User selects the To-do's location on Google Maps, and specifies the header, description, and person for the To-do.
3. (If user wants to edit a To-do)
  - 3.1. User selects a To-do.
  - 3.2. User edits the details of To-do.
4. User saves To-do.
5. System adds or updates To-do on the specified person's list.

**Goal:** User receives a geotagged picture.

**Initial state:** User arrives at a location with a geotagged picture.

**Final state:** User sees the geotagged picture, and marks it as seen.

**Description:**

1. User moves towards a specific location with a geotagged picture (or pictures).

2. As user nears the location, system displays the picture title (or list of titles).
3. User selects a picture.
4. System displays the picture.
5. User marks the picture as seen.
6. (If user ignores the alert)
  - 6.1. System alerts the user if the user moves away from the area without seeing the picture.

**Goal:** User creates or edits a picture for a particular person, at a particular location.

**Initial state:** User wants to create or edits a geotagged picture.

**Final state:** User saves a geotagged picture.

**Description:**

1. User chooses "Geotagged Pictures" from the main program.
2. System displays a list of geotagged pictures.
3. (If user wants to tag a picture)
  - 3.1. User selects "Pictures."
  - 3.2. System asks user to fill in fields for picture, contact name, location, and message.
    - 3.2.1. System displays the user's picture gallery on the phone.
    - 3.2.2. User selects a picture.
    - 3.2.3. System displays the user's Contacts list.
    - 3.2.4. User selects a Contact.
    - 3.2.5. System displays a list of saved locations on Google Maps.
    - 3.2.6. User selects a location.
    - 3.2.7. User types in a message about the picture.
4. (If user wants to edit a geotagged picture)
  - 4.1. User selects a picture.
  - 4.2. User edits the details of picture.
5. User saves the picture.

**Goal:** User creates a time and location sensitive configuration

**Initial state:** User chooses to create a new conditional configuration

**Final state:** User saves conditional configuration for use

**Description:**

1. User chooses "New Conditional Configuration Rule" option from main program.
2. User is presented with "Rule Information" screen
3. (Assume User cancels)
  - 3.1. App returns to main menu
4. (If User continues)
  - 4.1. User names the rule and chooses the trigger conditions (e.g. `_location_ within _100ft_ of _ECS160_ AND _time_ from _1pm_ to _2pm_ on _MWF_, _repeating_ from _TODAY_ to _6/3/11_`)
  - 4.2. User chooses "Settings for this Rule..."
  - 4.3. User is presented with the full Android Settings menu.
  - 4.4. User chooses settings for the rule and then exits the menu.
  - 4.5. User chooses to save the new rule.
  - 4.6. The rule is checked for conflicts with existing rules
  - 4.7. (Assume a conflict is found)
    - 4.7.1. User is notified of the conflicting settings
    - 4.7.2. User is returned to step 4.1 to make changes
  - 4.8. (If no conflict is found)
    - 4.8.1. The settings are saved in a database along with the given conditions.
    - 4.8.2. User is presented with confirmation of the new rule

#### 4.8.3. App returns to main menu

**Goal:** User edits a time and location sensitive configuration

**Initial state:** User chooses to edit an existing conditional configuration

**Final state:** User saves changes to conditional configuration

**Description:**

1. User chooses "Edit Existing Rule" option from main program.
2. User is presented with a list of existing rules.
3. (Assume user decides to delete a rule from the list)
  - 3.1. User presses delete button next to list item
  - 3.2. User is asked to confirm the delete
  - 3.3. (Assume user declines confirmation)
    - 3.3.1. User is returned to step 2.
  - 3.4. (If user accepts)
    - 3.4.1. The selected rule is removed from the database.
    - 3.4.2. User is returned to step 2 (with deleted rule no longer in the list).
4. (Assume user cancels)
  - 4.1. App returns to main menu.
5. (Assume User selects one of the existing rules)
  - 5.1. User is presented with "Rule Information" screen for the selected rule.
  - 5.2. (Assume User cancels)
    - 5.2.1. User is returned to step 2.
  - 5.3. (If User continues)
    - 5.3.1. User changes the rule's name or trigger conditions as desired.
    - 5.3.2. (Assume user clicks "Settings for this Rule...")
      - 5.3.2.1. User is presented with the full Android Settings menu.
      - 5.3.2.2. User chooses settings for the rule and then exits the menu.
    - 5.3.3. User chooses to save the new rule.
    - 5.3.4. The rule is checked for conflicts with other rules
    - 5.3.5. (Assume a conflict is found)
      - 5.3.5.1. User is notified of the conflicting settings
      - 5.3.5.2. User is returned to step 5.1. to make changes
    - 5.3.6. (If no conflict is found)
      - 5.3.6.1. The settings are updated in the database along with the conditions.
      - 5.3.6.2. User is presented with confirmation of the changes
      - 5.3.6.3. App returns to main menu

**Goal:** Conditional configuration settings are applied/removed when conditions are met/expire.

**Initial state:** Conditions for an existing configuration are met or expire.

**Final state:** Settings for the triggered configuration are applied or removed.

**Description:**

1. (Assume conditions for a saved rule are met)
  - 1.1. The program saves the current android settings in a database.
  - 1.2. The program loads the android settings for the given rule.
  - 1.3. The program marks the rule as active.
2. (Assume conditions for an active rule expire)
  - 2.1. The program loads the settings that were saved when the rule was triggered.
  - 2.2. The program marks the rule as inactive.

# Questions

## Place it notes

1. Do you want the user get an alert when their notes expire? (If Ryan had a note that told him to pick up milk from the Co-op on Tuesday, and he never picked up the milk, would he get an alert on Wednesday saying that that note expired?)
2. If the user is not the creator of the note, would the creator be notified when the note has been completed, or when the note is declined, or when the note has been ignored?
3. Would users be able to comment on the notes and leave messages for their creator? (Perhaps Ryan couldn't find regular milk and bought soy milk instead. Should we built in functionality to allow him to leave a comment?)

## Geotagged pictures

4. When the user sees a picture, should the creator of the picture be notified?
5. Would users be able to comment on the pictures and leave messages for the creator?
6. Can users see geotagged pictures that they have been tagged in, even if they're not at the location? (What if Matthew wants to share a picture of the Eiffel Tower with his mother, but knows that his mother will not go to Paris any time soon?)
7. Do the geotagged pictures have to have a specific location? (What if Matthew wants to tag his friend in a picture, but doesn't want to specify the location?) Does that defeat the purpose of the "geotagged" part?
8. Should the user be asked if a picture should be geotagged when they take it with the camera, or should the user launch our app and choose to geotag existing pictures?
9. Should geotagged pictures be usable in place it notes? (If Matthew wanted a friend to buy more of a specific brand of milk, could he take a picture and add it to the todo?)

## Location sensitive service customization

10. Do all fields for the customization (location, time, date, contacts) need to be filled in?
11. Should there be default values, and if so, what should the defaults be? (Location = Everywhere/Nowhere; Time = All day/Never; Date = Forever/Never; Contacts = Everyone/No one?)
12. If there is a conflict, should the system alert the user, or should the system attempt to resolve it by prioritizing the customizations? (For example, let's say that Daniel specified that he wants his ringtone to be disabled between 12am – 6am, for all locations, forever, for all contacts. He also specified that want his ringtone to be enabled if he were in the MU, forever, for all contacts. If, for some reason, he were in the MU at 5am, and someone calls him, what should happen? Should the "all locations" specification override the "MU" specification?)