CPS406 Introduction to Software Engineering project iterations 2/3

This assignment is a team effort. To facilitate collaboration, this time teams will have four members each. As you should know by now, privacy concerns preclude D2L showing, or me sending, the names and emails of other team members to you. But you can send them email - you won't see their emails as they will be in the bcc: field - or try to initiate chat.

You can choose between the two projects described below.

This project counts as iterations 2 and 3, and it is therefore worth 20 marks.

What to submit

Your submission should include a report similar to the one in iteration 1, with the following documents:

- - a title page in PDF format listing the team members who contributed, and only the team members who contributed, to the team effort;
- - a ZIP archive containing the requirements document (in particular, any changes or omitted functionality from the specification below), design documentation, and source and executable code of the project; and
- - a video showing your system in operation (see below for details).

Submissions should be clearly labeled as team_xx_report.pdf, team_xx_code.zip, and team_xx_proof_of_life.mp4, where xx denotes your team number. Improperly labeled submissions will cost you one mark.

The deadline to submit the assignment is April 10, at 11:59pm. The usual late-submission policy applies, with one mark deducted for submissions made by 11:59pm on April 11; two marks deducted for submission made by 11:59pm on April 12; and three marks deducted for submissions made by 11:59pm on April 13.

Cypress

Inspired by the fearless government leaders, Toronto City Council has unanimously decided to automate the system that provides its citizens the ability to report problems they notice on the streets, and allows them (i.e., citizens) to track the resolution of those problems.

The current process is manual and tedious. Upon noticing a problem, a citizen has to find the right phone number to call, as different service organizations deal with different types of problems. (There may be a web site that lists all the numbers, but one still has to remember the exact location of the problem situation.) Furthermore, upon reporting a problem, there is virtually no way that a citizen can track the process of resolving the problem.

The Cypress system will change all that. It will allow the citizen to

- pinpoint the location of the problem on a user-friendly map with arbitrary accuracy,
- accurately specify the type of problem,
- report it to the City of Toronto with all necessary detail, and
- subscribe to subsequent notifications about the resolution of the problem.

Also,

- Citizens should be able to consult the map in order to learn about the problems in a given area.
- Also, duplicate notifications should be identified and flagged as such. You may define a certain perimeter to help the system determine which problem has already been reported.
- Measures have to be taken to prevent false reports.

Your task is to develop as much of the Cypress system as possible, based on the attached requirements document.

As before, there are no restrictions on the tools that you'll use or the programming language. Since the project assumes there is a database somewhere, do not be shy in using stubs to mimic the behavior of the database. You may consult the document explaining the other side (i.e., traditional or agile) of the same problem, but please note that your work will be evaluated against your starting document.

For the record, the problem formulation is due to an older colleague of yours in a course I taught … and a younger colleague sold it as his own later – after he got it as a project in another course I taught.

CYPRESS
City of Toronto Problem Reporting and Solution System
Requirements Document

1. Introduction

1.1 Purpose:

The Cypress system is designed to allow citizens to report problems that concern them about their city that they notice on the streets. Along with that they will be able to follow up on the status of their problem from when it is first reported and until a solution is found.

This document outlines the features of the CYPRESS system which will illustrate the guidelines of the developers and provide the client with the necessary software validation document.

1.2 Scope:

The following describes what features are in the scope of the software and what are not in the scope of the software to be developed. *In Scope:*

a. Managing reported problems from citizens, which includes maintaining information about the citizens as well as maintaining their profiles, their complaints, their suggestions and maintaining their privacy and security.
b. Computing the amount of complains per a specific problem and elevating that problem on the list of problems that are known on a priority list.
c. Providing the unique user who first reported about a specific problem with an email outlining what the user of the Cypress system (City of Toronto Road Safety Officials) plans to do to resolve the issue and also provide a minimized version of this solution to all others who have reported a similar or the same problem.
d. Downloading the most current map of the City of Toronto from the web in order to stay precise and up-to-date for the citizens who wish to report problems they have noticed. e. User authentication. *Out of Scope:*

a. Features for allowing citizens to get involved beyond the jurisdiction of this system to solve a particular problem.
b. Any other city related issues like parking tickets, building construction issues or any other city related problems.

1.3 Definitions, Acronyms, and Abbreviations:

Acronyms and Abbreviations:

a. CYPRESS: City of Toronto Problem Reporting and Solution System
b. SRS: Software Requirements Specification.
c. WWW: World Wide Web.
d. GUI: Graphical User Interface.
e. CYTRSO: City of Toronto Road Safety Officials (Users)

Definitions:

a. Problem: A real event that involves property damage to the City of Toronto. In the context of potholes, utility failures, tree collapse, flooded streets, property vandalism, mould and spore growth, eroded streets and garbage/ road obstructions.

b. Security: A set of characteristics which must match when an entity logs in with a specified user name in order to be given access to the system.

c. Portfolio: A set of data for a particular user who is active in the system.

1.4 References:

Appendix A: User Screens.

1.5 Overview:

The following topics of the SRS are organized as follows: Section 2 gives an overall description of the software. It gives what level of proficiency is expected of the user, some general constraints while making the software and some assumptions and dependencies that are assumed. Section 3 gives specific requirements which the software is expected to deliver. Functional requirements are given by various use cases. Some performance requirements and design constraints are also given. Section 4 gives some possible future extensions of the system. Finally the appendices in Section 5 describe respectively the user screen.

2. Overall Description:

2.1 Product Perspective:

CYPRESS is tailored towards people who value their environment and want to make it a better and safer place for others. CYPRESS should be a simple piece of software that allows a citizen to report a city road or property problem to CYTRSO, the users at the Toronto City Council. CYPRESS should provide a user friendly map that allows the user to pinpoint the exact location of the problem that they are reporting, hence making it easier for the city workers to fix the problem as soon as possible. CYPRESS will also make sure to notify the user when the city has received their report and as well if steps have been taken to solve the problem. In order to ensure that no false report are being posted, CYPRESS will check with a list of citizens that reside in Toronto, thus making the whole CYPRESS program a faster way to make the city better and safer.

2.2 Product Functions:

CYPRESS should support the following use cases:

Class of use cases	**Use cases**	**Description of use cases**
Use case related to System authorization	Login	*Login in to CYPRESS*
	Cancel	*Moving away from login page*
Use case related to the select language	English	*Move User to English page*
	French	*Move User to French page*
Use case related to Registering	Enter Information	*User enter their information to register*
	Create Username	*User chooses a username*
	Create Password	*User chooses a secure password*
Use case related to information change	Change Information	*User changes their information*
Use case related to creating report	Create Report	*User chooses to create a report about a specific location*
Use case related to editing report	Edit Report	*User chooses to edit the report that they have created*
Use case related to deleting report	Delete Report	*Use chooses to delete the report*
Use case related to rankings of report	Rankings	*Rank each report based on the location and how many complaints of the same report has been received*
Use case related to resolution	Report resolution	*System notifies city council about the report*
Use case related to notification	Notification	*System notifies the user if their report has been taken in to consideration*
Use case related to FAQ	FAQ Questioning	*System show a list of common question and answer for the user*
Use case related to Contacting	Contacting	*System provides a list of contact information for the user*
Use case related to logout	Logout	*System logs the user out and saves the last saved input*

2.3 User Characteristics:

a. The User should be reliable.
b. The User should know the details and location of the problem.
c. The User must have a fair bit of knowledge about the city of Toronto (current mayor, streets, laws, etc.).
d. The User must have sufficient vocabulary skills and adequate grammar.
e. The User must be competent when using a computer and the internet, and know that nothing is completely confidential but privacy from the city's side is guaranteed.

2.4 Principal Actors:

The two principal Actors in CYPRESS are "user" and "system".

2.5 General Constraints:

a. For full working CYPRESS requires Internet connection.
b. CYPRESS is single-user software that takes user input.

2.6 Assumptions and Dependencies:

a. Full working of CYPRESS is dependent on the availability of Internet connection.
b. Access to CYPRESS is found in www.toronto.ca/cypress .CYPRESS would not work on any other website.

3 Specific Requirements:

3.1 Functional Requirements:

We describe the functional requirements by giving various use cases.

Use cases related to initial visitation to CYPRESS website:

Use Case 1: Language Selection

Primary Actor: User Precondition:
None Main Scenario:

1. User arrives at website; default language is set to English Alternative flow:

1. (a) User changes language

1. (a)1. User clicks on language option button to change language to French

Use Case 2: Registration

Primary Actor: User
Secondary Actor: System
Precondition: None Main Scenario:

1. User goes to website, and clicks register option button and is redirected.
2. User must agree to terms and conditions of this site
3. User provides personal information and login name and password 4. System checks that password is secure enough and login name is free Alternative flow:

4. (a). Login name/ password is not secure
   4. (a)1. User is re-prompted to enter new login name / password
   5. User is taken back to main page of website

Use Case 3: Login
Primary Actor: User
Secondary Actor: System
Precondition: User must be registered.

Main Scenario:

1. Go to website and click member's area tab
2. User gives login info
3. System checks user info
4. Members area is displayed Alternate Scenario:

4. (A) Login fails

4. (A) 1. Re-prompt for login info

4. (A) 2. User is allowed to enter info 3 times before being banned for an hour

4. (B) User forgot password

4. (B)1. User is prompted for answer to secret question

4. (B)2. System checks to see if answer is right and if answer is right, sends password to email account

Use cases related to User profiles:

Use Case 4: Change Information

Primary Actor: User Precondition:
logged in Main Scenario:

1. User clicks on profile info tab
2. All the profile info is displayed (password, address, number, etc.)

3. User clicks save and exit when done changing profile Alternate Scenario:

3. (b) If not all required fields are filled out

3. (b) 1. User has to fill in the required fields before exiting

Use Case 5: Delete Profile

Primary Actor: User
Precondition: User is logged in Main Scenario:

1. User clicks delete profile tab
2. User is prompted for answer to secret question (to make sure it is the correct person)
3. User is prompted if he or she is sure they want to delete profile
4. User is prompted for reason for leaving (to better improve our customer service)
5. User information is erased from system along with reports

Use cases related to reporting problems:

Use Case 6: Create Report/ Report a Problem

Primary Actor: User
Precondition: User is logged in Main Scenario:

1. User clicks on create a report tab
2. User is prompted for a location on the city map
3. User is prompted for complaint about the selected area 4. User then saves report and exits Alternative flow:
4. (a) a required field is missing and user cannot save and exit
   4. (a)1. User is prompted for information the required field

Use Case 7: Edit Report
Primary Actor: User
Secondary Actor: System Precondition: User is
logged in Main Scenario:

1. User clicks edit report tab
2. System displays a list of all the Users reports are displayed
3. User clicks on a report to change
4. User is prompted for city area via a map
5. User is prompter for a problem 6. User then saves report and exits Alternative flow:
6. (a) A required field is missing and the User cannot save and exit 6. (a)1. User is prompted for information the required field

Use Case 8: Delete Report

Primary Actor: User

Secondary Actor: System Precondition: User is
logged in Main Scenario:

1. User clicks delete report tab
2. System displays a list of all the Users reports are shown
3. User clicks on a desired report to delete
4. System prompts the User if he/she is sure before deletion of report

Use cases related to Report Resolution and User Notifications:

Use Case 9: Ranking
Primary Actor: System Precondition:
None Main Scenario:

1. System sorts all the reports and checks for patterns
2. System checks where the most problems are coming from
3. System checks which problem is most frequent
4. System then ranks the type of problem and part of town from good to bad

Use Case 10: Report Resolution

Primary Actor: System
Secondary Actor: City Officials Precondition:
None Main Scenario:

1. System notifies city officials about the problem
2. City Officials try their best to resolve conflict and the gets back to the system tech with the course of action being taken
3. System notifies the User that the problem has been resolved and thanks them for their contribution to society.

Use Case 11: Notify
Primary Actor: User Precondition:
None Main Scenario:

1. User is prompted to part-take in a survey about the city
2. User accepts and answers all the questions in the survey and submits Alternative flow:

2. (a) User declines request and continues to site
3. Results for survey are stored and tallied up

Use cases related to other features of the CYPRESS website:

Use Case 12: Suggest
Primary Actor: User
Secondary Actor: System
Precondition: None Main Scenario:

1. User clicks suggest tab

2. System displays a list of reports to the User
3. User clicks on the report and options are shown Alternative flow

3. (a) User clicks like button

3. (b) User suggests a possible solution for the problem and submits it

Use Case 13: Contacting
Primary Actor: User
Secondary Actor: System
Precondition: User needs to speak to someone from the CYTRSO.

Main Scenario:

1. User clicks on contact us tab
2. System displays a list of city officials to choose from
3. User picks city official and contact information and office hours are displayed.

Use Case 14: FAQ Questioning

Primary Actor: User
Precondition: User is unsure about a specific matter.

Main Scenario:

1. User clicks FAQ tab
2. A list possible/ most common answers is shown to the User

Alternative flow

2. (a) User scrolls down and reads each problem

2. (b) User clicks on question category and is brought to the question and answer immediately

Use Case 15: Redirecting
Primary Actor: User Precondition:
None Main Scenario:

1. Go button is to finalize the decision after clicking on a tab, takes User to selected page

Alternative flow

1. (a) User has not clicked a tab. Nothing happens in this case and the User screen remains active.

Use Case 16: Logout
Primary Actor: User
Precondition: User is logged in.

Main Scenario:

1. User clicks log out button
2. User is redirected to the main page of the site

Use Case 17: Home
Primary Actor: User Precondition:
None Main Scenario:

1. User clicks on home tab
2. User is redirected to main section of the members area

Use Case 18: Tell a friend Primary Actor: User

Precondition: User wishes to spread the word about the Cypress website.

Main Scenario:

1. User clicks on tell a friend tab
2. User is prompted for friends email address
3. User is prompted to add a personal message

Alternative flow

3. (a) User can decline to add a personal message

3. (a)1. A default message is sent instead

Use Case 19: Vote
Primary Actor: User
Secondary Actor: System
Precondition: None Main Scenario:

1. User clicks on the vote tab
2. Systems displays a list of options to vote from
3. System tracks the number of votes and displays the results thus far to User via an email message to Cypress account.

3.2 Performance Requirements:

(a) Should run on 500 MHz, 64 bit MB machines.
(b) Majority of responses which include buttons and page transitioning should be quick and respond within 2 seconds, except when loading the map function which will take more time.
(c)

3.3 Design Constraints:

1. *Accessibility:* Anyone should be able to understand the layout of the website and to navigate it without issue.
2. *Reliabili*ty*:* People should be able to easily login and check or submit any problems without issue and be notified as soon as possible.

3.4 External Interface Requirement

(a) The front page has the name of the site and the city of Toronto logo under it and two buttons under the logo. The two buttons are for either English or French which the user may choose.

(b) The main page is split into two panes; the right contains images of the city and the site motto under it, the left contains options the user may choose from such as Register, Login, Report a Problem, Suggestion, Vote, FAQ, Contact Us.

(c) In the register page, multiple field and text boxes are placed asking for personal information such as name, address, phone number. Under theses boxes are two buttons, register and cancel and at the bottom left corner of the page is the Frequently Asked Questions button.

(d) In the report a problem page, a field box is placed asking for the address of where the problem is and under the box are multiple check-boxes allowing the user to choose the option(s) that correspond to the problem.

(e) In the login page, a short message is placed describing what the site does. Under the message are two text boxes and field asking for the username and password and under the field are two buttons labelled login and cancel.

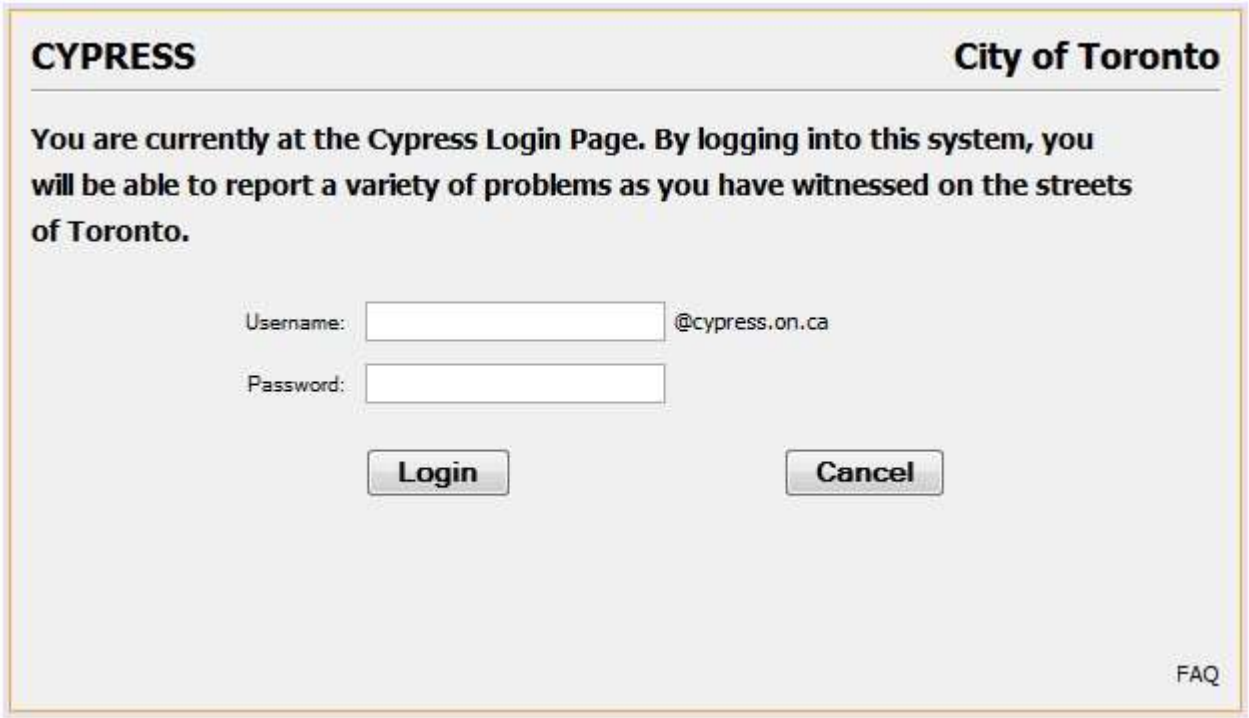
4. Future Extensions:

a. CYPRESS is intended to be single user software. A possible future extension would be to allow multiple users. Multiple city officials could possibly work on resolving a single problem together as opposed to a single user working a single conflict.

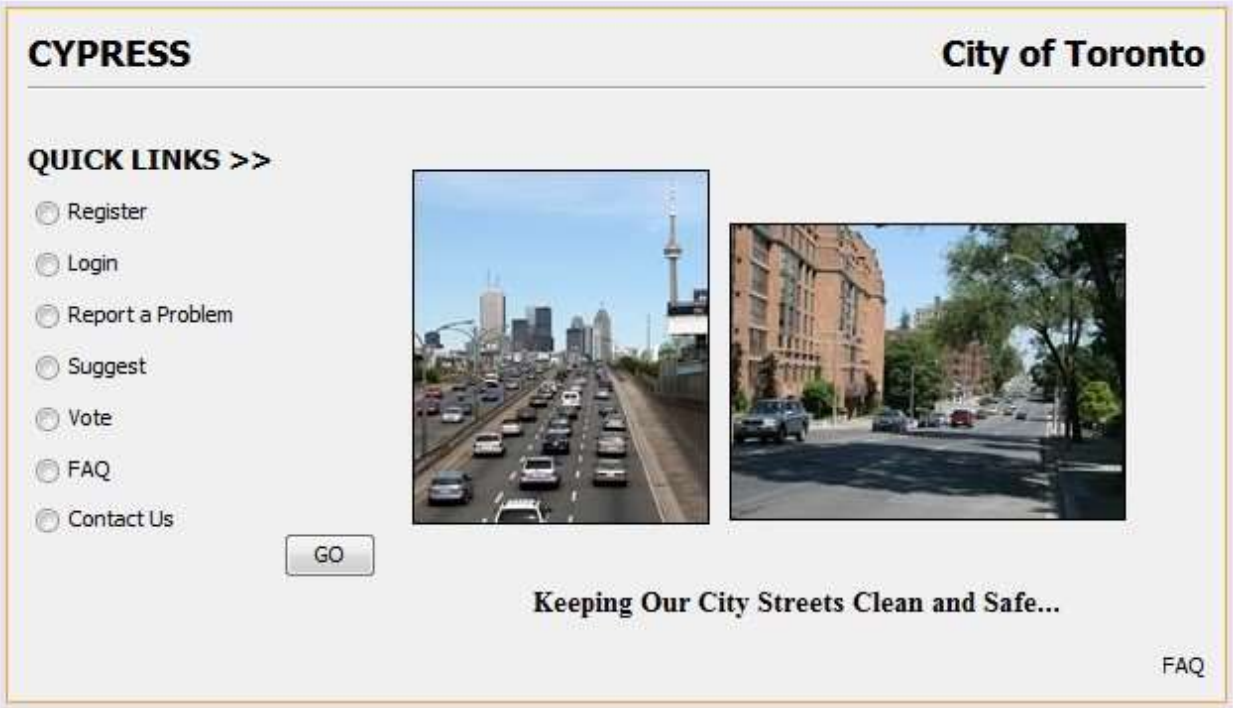
5. Appendix

5.1 Appendix A: User Screens

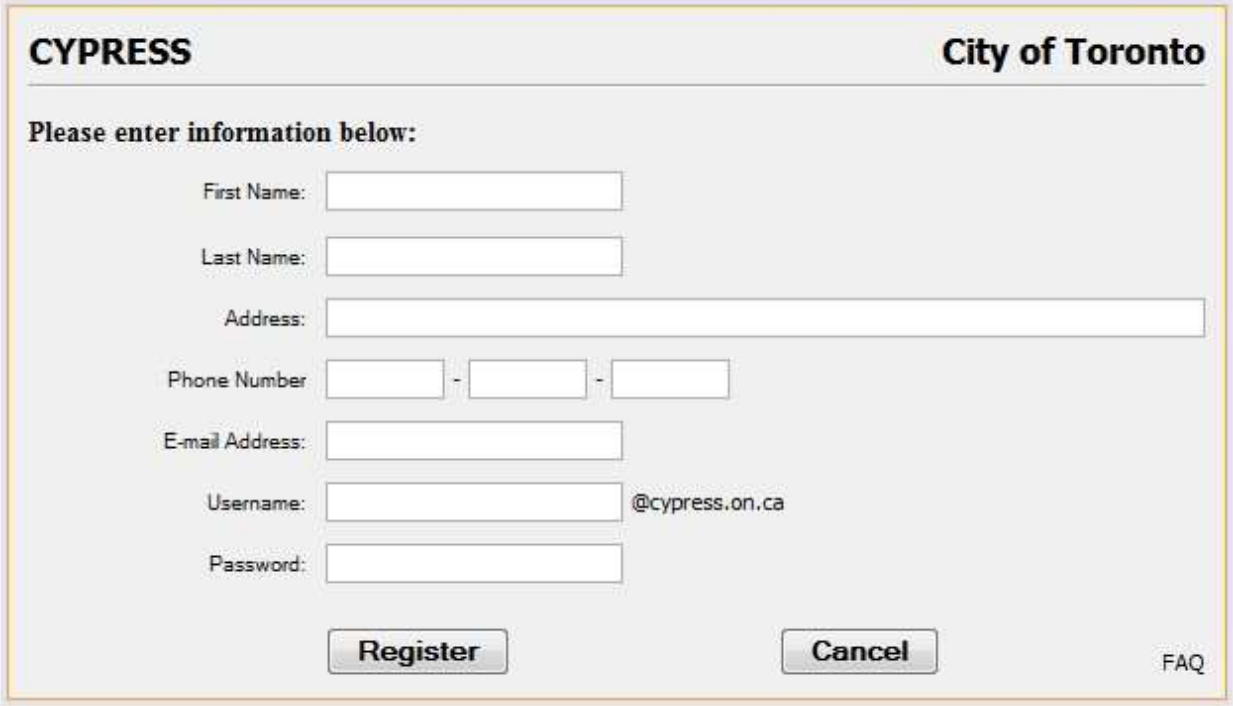
Main Screen (Language Selection)



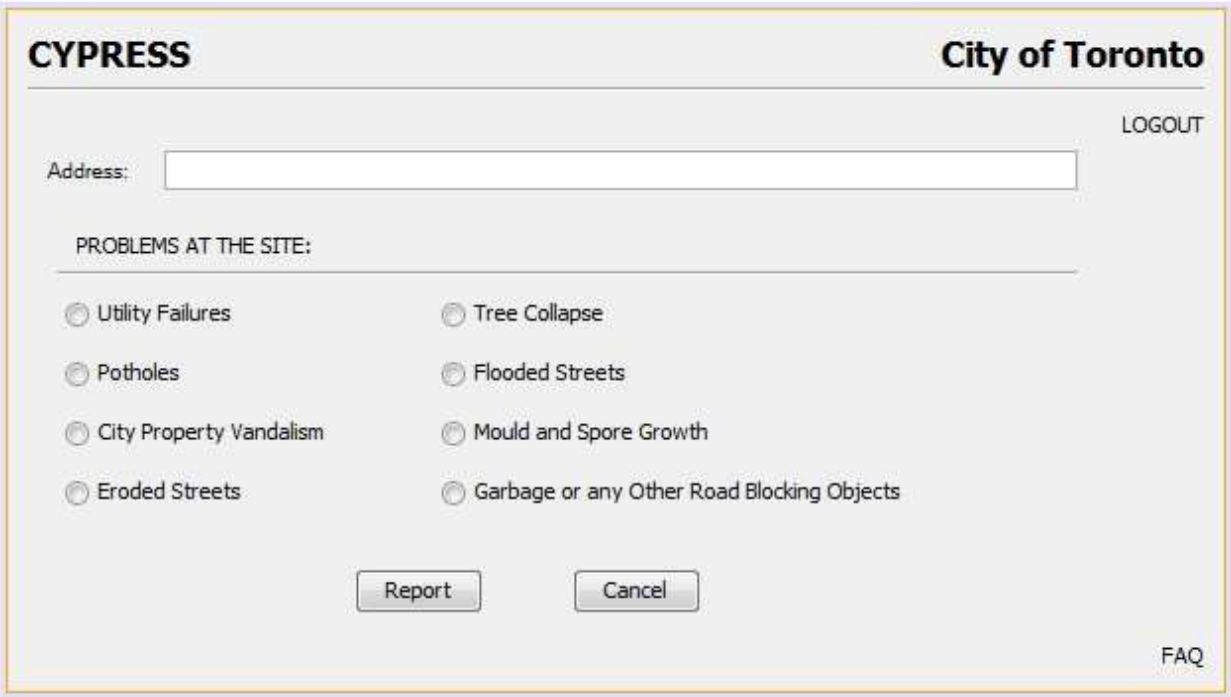
Login Screen



Portal Screen



Registration Screen

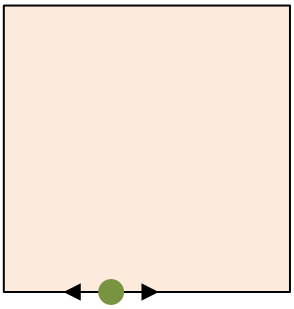


Reporting Screen

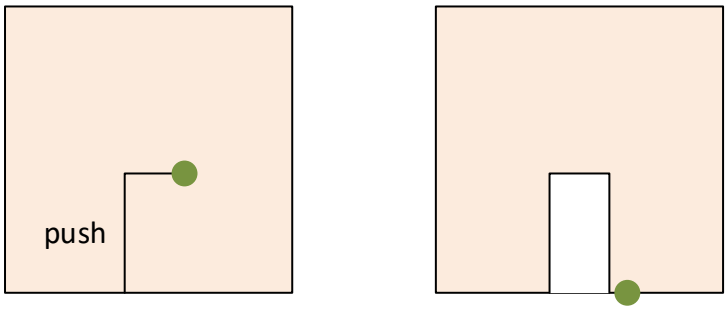
The Qix game

Your desire to qualify for the Ryerson Computer Gamers Hall of Fame, you need to qualify by implementing as much of the Qix game as possible. The basic outlines of the game follow.

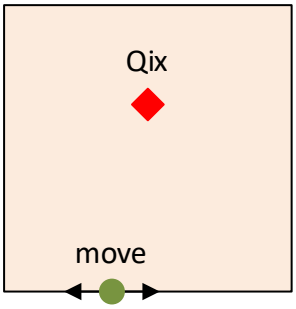
The game is played on a square field. The player (that would be you) moves a marker, say, of green colour along the edges of the field – the commands would be to Move left, right, u, and down. For simplicity, you may use just two of those, and change direction as you pass a corner of the field.

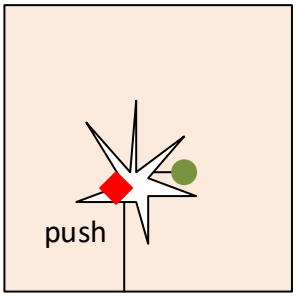


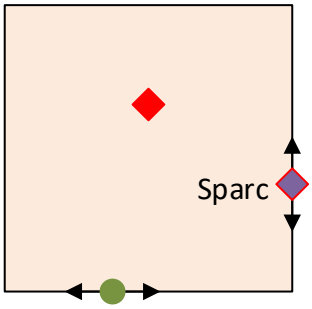
Your goal is to claim as much of an area inside the field as possible. To this end, you need another command, say, Push which causes the marker to enter the field. By simultaneously pressing one of the Move commands, you create an outline of an incursion into the field. Once your marker reaches the edge of the original field, the incursion changes colour and the field is effectively reduced. Your goal is to claim a predefined portion of the area of the field, say, 50% or so. (Ideally, this should be adjusted as a parameter before the game.) Did I say that the edge includes the original edge plus the inner side of all successful incursions?



The game as described so far would be extremely easy, were it not for the fact that there are enemies which try to get to you. First, there is the Qix, a terrifying creature that wanders randomly through the field itself. As long as you're on the edge, you're safe from the Qix. To be fair, the Qix does not chase yu – it just wanders around, but it is still deadly. Namely, if you begin a push and Qix manages to reach you before you actually finish the incursion, you will be set back (say, lose a life or a predefined amount of life force) and the incursion is effectively canceled – you are sent back to the location where the push has begun. But if you successfully end a push, you're safe again, and the free area in which the Qix can wander is reduced.

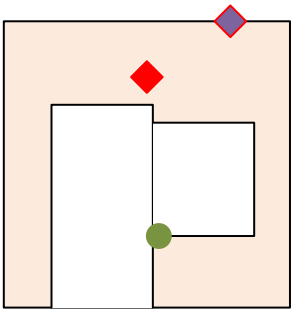






The other enemy (or enemies) are the Sparx. These are markers not unlike yours, and they move randomly along the edges of the field just like you do. You can begin with a single Sparc, and maybe progress to two of them – more than two would make for a rather annoying game. Also, it is wise to make them move at a slower pace than your marker, otherwise you're toast. The reason is, the Sparx are almost as lethal as the Qix – if they catch up with you as they move along, they will also eat up some of your life force. The same happens if a Sparc reaches the beginning of an unfinished push – the push is cancelled, your marker is back at the edge, and the Sparc is happily moving away from you.

You can assume that a Sparc always moves in the same direction, but that direction may change if they catch up with you, just to make your game more interesting.



So, the game basically unfolds as follows. You begin with a certain amount of life force, which is reduced at every encounter with the Qix or a Sparc. The goal is to claim as large a portion of the area of the field as possible whilst remaining alive. Once you do that, you can progress to the next level, with fresh supply of the life force and a faster Qix, or maybe two Sparx instead of a single one. And so on. Allowed variations also include a different shape of the Qix – a rotating stick, or a snake-like movement. But that's only if you want bonus points.

Your task is to develop as much as possible of the functionality of the game described above., and to submit the deliverables specified. You may proceed in steps: first the user and a single Sparc, then two Sparx, then two Sparx and the Qix. You check youtube.com for samples of Qix game (which are, admittedly, far too complex, but may still give you an idea

Please note that you're free to choose the implementation platform – any operating system, any language, any graphics – and the list below shows some useful resources for Python/Pygame and Unity platforms.

Some resources

Python game programming: https://wiki.python.org/moin/GameProgramming

Making Games with Python and Pygame: http://inventwithpython.com/pygame

Pygame tutorial (one of a few): https://realpython.com/pygame-a-primer/

Unity: https://unity3d.com/get-unity/download

(with tons of links for downloading and for learning Unity)