

***Guttershipe (0.3)***

***Software Requirement  
Specifications***

## Revision History

Date	Name	Description	Author
September 14, 2016	0.3	Draft	Mitchell Verter

# Table of Contents

- 1. Introduction
  - 1.1. Proposal
  - 1.2. Technical Introduction
    - 1.2.1. Purpose
    - 1.2.2. Scope
    - 1.2.3. Acronyms and Abbreviations
    - 1.2.4. Summary
- 2. Overview
  - 2.1. Use Case Diagram
    - 2.1.1.
  - 2.2. Collaboration Diagrams
    - 2.2.1. SHAREABLE: READ
      - 2.2.1.1. ViewShareable
      - 2.2.1.2. SearchShareables
    - 2.2.2. ACCOUNT CONTROL: EXTERNAL
      - 2.2.2.1. Login
      - 2.2.2.2. Register
      - 2.2.2.3. RecoverPassword
    - 2.2.3. ACCOUNT CONTROL: INTERNAL
      - 2.2.3.1. EditProfile
      - 2.2.3.2. EditSchedule
      - 2.2.3.3. RenewMembership
      - 2.2.3.4. ChangePassword
      - 2.2.3.5. Logout
      - 2.2.3.6. DeleteAccount
    - 2.2.4. SHAREABLE: CREATE, UPDATE, DELETE
      - 2.2.4.1. AddShareable
      - 2.2.4.2. EditShareable
      - 2.2.4.3. DeleteShareable
    - 2.2.5. SHAREABLE: ANNOTATE
      - 2.2.5.1. RateShareable
      - 2.2.5.2. CommentOnShareable
      - 2.2.5.3. DeleteComment
    - 2.2.6. COMMUNICATIONS
      - 2.2.6.1. SendMessage
      - 2.2.6.2. ReadMessage
      - 2.2.6.3. DeleteMessage

- 2.2.6.4. BlockUser
- 3. Class Definitions and Diagrams
  - 3.1. User and Subclasses
    - 3.1.1. User and Subclasses Diagram
    - 3.1.2. User Definition
    - 3.1.3. Guttersnipe Definition
    - 3.1.4. Caretaker Definition
  - 3.2. Shareable and Parts
    - 3.2.1. Shareable and Parts
    - 3.2.2. Shareable Definition
    - 3.2.3. Time Definition,
    - 3.2.4. Thing Defintition
  - 3.3. 3.3.ER Class Diagram
- 4. Screenshots from 0.2 release and Future Wireframes
  - 4.1. Front Page
  - 4.2. Create Shareable Wizard
    - 4.2.1. CreateShareable: Start
    - 4.2.2. CreateShareable: Instructions
    - 4.2.3. CreateShareable: Describe
    - 4.2.4. CreateShareable: Classify (I)
    - 4.2.5. CreateShareable: Classify(2)
    - 4.2.6. CreateShareable: Map
    - 4.2.7. CreateShareable: Schedule (1)
    - 4.2.8. CreateShareable: Schedule (2)
    - 4.2.9. CreateShareable: Schedule (3)
  - 4.3. SearchShareable
    - 4.3.1. SearchShareable: ResultList
    - 4.3.2. SearchShareable: ResultCalendar
    - 4.3.3. SearchShareable: ResultMap
    - 4.3.4. SearchShareable: SearchByCategory
    - 4.3.5. SearchShareable: SearchByTag
    - 4.3.6. SearchSharable: SearchByLocation
    - 4.3.7. SearchShareable: SearchByTime
  - 4.4. Authentication
    - 4.4.1. SignIn
    - 4.4.2. SignUp
  - 4.5. Documentation
    - 4.5.1. Mission Page
    - 4.5.2. FAQ
    - 4.5.3. Presentation (2013)
      - 4.5.3.1. Start
      - 4.5.3.2. Objective
      - 4.5.3.3. Audience

# **Part 1: PROPOSAL**

**PROPOSAL:**                **GUTTERSNIPE**

**1. What is the site/app?**

Guttersnipe is a web portal and mobile app that caters to anarcho-communist street youth (and adults) who desire to subvert capitalism by sharing resources.

It will enable people to broadcast to each other locations of shareable resources, distributed among four main categories:

1. Housing: squats, abandoned buildings, punk houses, etc.
2. Food: dumpsters, Food Not Bombs, free meals, etc.
3. Healthcare: clinics, needles, condoms, etc.
4. Movement: rideshares, train maps

Eventually, other types of resource sharing will be integrated into the application.

Each Shareable will be characterized as a

1. Thing: Categorization and Tags
2. Space: Geolocation
3. Time: Schedule

Further development will require a close study of the writings of Kropotkin and Fourier.

**2. What need does this meet? or problem does it solve?**

This application serves the urgent need to overthrow capitalism by helping people to self-organize outside and beyond the market of commerce.

The ultimate intention is to facilitate the creation of alternate avenues of exchange, freely organized by free individuals.

**3. Who is going to go/use to this site/app?**

In the current incarnation, it is mostly aimed towards the freegans gutterpunks, who live off of dumpstered food, live in squatted housing, and travel by hopping trains.

As we get a better sense on the needs of the anticapitalist community and possibilities for alternative organizing, we will expand the possibilities for anti-market resource sharing.

**4. Why will they go to this site/app?**

To find food, clothing, shelter, etc.

**5. Why will they keep coming back to your site/app?**

See above.

**6. How is it different from other similar sites?**

There are similar sites of various types, but many of them have certain faults.

There is a site called rideshare.com; there is a site named couchsurfer.com; there is freecycle.com, which allows the sharing of goods.

These are all laudable efforts. Some of these are marred by an underlying desire for profit. But some of them are motivated out of genuine desire to promote Mutual Aid.

The very mission of Guttersnipe.net will be to promote the organization of the lumpenproletariat and to create alternative exchanges outside of capitalism. This mission will enable Guttersnipe.net to be singularly focused on this goal.

It will thus be able to bring together whatever resources necessary for the undermining of capitalism: the various services— such as squatting, dumpster diving, hitchhiking, train hopping, resource sharing, etc — will be coordinated on a singular web portal.

In addition, there are several web portals that are dedicated towards the promotion of anarcho-communist goals..

Such sites are

- [Freegan.info](#)
- [Picture the Homeless](#)
- [Squat.net](#)
- [Foodsharing \(Germany\)](#)

Many of our initial design specifications will be taken from the freegan group and Picture the Homeless.

In addition, we intend Guttersnipe to be cross platform, available both via the web and as a mobile app.

To my knowledge, there are not yet any apps dedicated with such a task.

## **7. What steps will a person go through interacting with the site/app?**

Most of the various interactions will be handled using forms.

The various services offered by Guttersnipe all boil essentially boil down to two types of transaction:

1. information submission;
2. information retrieval.

One person posts about an abandoned building or a good dumpster; another person searches for such information.

**\*\* All Users can view a Shareable or search all Shareables.**

Shareables can be searched and results will be shown with the following data:

1. Thing: Description, Categorization, Tags
2. Space: Map
1. Time: Calendar

**\*\* Registered Users (known as “Guttersnipes”) can add, edit, and delete Shareables. Guttersnipes may also rate Shareables, comment on Shareables, and erase these comments.**

Guttersnipes can manage their profiles, which contains their availability schedules, names, account expiration date, optional email, optional password, optional location, and optional contact info.

All Guttersnipe user accounts expire after a certain date, but this date may be extended at any time.

Guttersnipes can communicate to each other messages that contain Schedules and Text in order to coordinate a meeting time.

**\*\* Caretakers**

Caretakers are Guttersnipes with administrative capacities.

They can delete any Guttersnipe account, any Shareable, and any Comment.

#### Practical Constraints

#### ***Security***

We will have to build in a security infrastructure in the project in order to guarantee anonymity of transactions.

Tor will be used to anonymize transactions.

Whisper will be used to encrypt communications and interactions.

The host server will have to be able to run Python/Flask.

The client will have to have an accessible webview for the deployment of Javascript.

We will have to design a User Interface that supports a 1.8 inch QQVGA (128x160) Display to support the phones provided by government assistance <http://newsroom.assurancewireless.com/custom-page/product-information>



### 1.1. Purpose

Guttersnipe promises to be a platform for individuals and groups to freely share resources such as food, shelter, and medicine.

This Document will detail the features of Guttersnipe, and will serve as a guide to developers, and as a legal document and users manual for prospective clients.

### 1.2. Scope.

#### 1.2.1. Users

##### 1.2.1.1. User

All System users can search the Shareables view a single Shareable, and read the rating and the comments ascribed to it.

##### 1.2.1.2. Vagrant: This class represents all visitors to the site who has not yet signed in as a member.

The Vagrant class inherits all the properties and functionalities of the base class User.

All Vagrants may register to become a user, may login as a user, and may retrieve a lost username or password.

##### 1.2.1.3. Guttersnipe: This class represents a user who has registered for an account in the system.

This class inherits all the properties and functionalities of the base class User.

In addition, the Guttersnipe can exercise control over its own account.

The Guttersnipe may edit its own profile, its own location and edit its own availability schedule.

It may renew its membership and change its password.

It may logout of its account.

The Guttersnipe may also create a new Shareable, and edit or

delete a Shareable that it has created.

The Guttersnipe may rate or comment on a Shareable and may delete a previous comment.

The Guttersnipe may send messages and send the schedules of other Guttersnipes. It may read messages and schedules as well. It may block any other Guttersnipe except for a Caretaker.

1.2.1.4. Caretaker: The Caretaker class represents the administrative users of the System.

Caretakers have all the same properties and functionality as the Guttersnipe, but their functionalities are unlimited in scope.

The Caretaker may edit or delete any Shareable, may delete any User, and may delete any comment.

## 1.2.2. Business Objects

1.2.2.1. Shareable: Each shareable is classified according to its categorization and description, its schedule, and its location.

1.2.2.1.1. Time: Each Shareable is available at a certain period of time (e.g. every Monday from 2PM-4PM). Time information provides these schedules as well as a note for further clarification.

1.2.2.1.2. Space: Each Shareable is located in a certain place (e.g.. Times Square). Space information provides longitude and latitude information, as well as textual information detailing the canonical and alternate information, and any additional notes

1.2.2.1.3. Thing: Each Shareable is characterized as a certain Type (food, shelter, medical, travel) and can have certain subtypes. Shareables can also be given “Tags” for further categorization. Textual information about how to acquire the shareable and other notes will also be included.

1.2.2.2. Shareable Annotations

1.2.2.2.1. Ratings: Guttersnipes will be able to rate shareables

1.2.2.2.2. Comments: Guttersnipes can add comments to shareables.

## 1.2.3. Communications

- 1.2.3.1.
- 1.3. Acronyms and Abbreviations
  - 1.3.1. V: Vagrant
  - 1.3.2. G: Guttersnipe [User]
  - 1.3.3. B: Caretaker
  - 1.3.4. SRS: Software Requirements Specification
  - 1.3.5. GUI: Graphical User Interface.
  - 1.3.6. FSM: Finite State Machine.
  - 1.3.7. 1 DB: Database.
  - 1.3.8. 1 ERCD: Entity-Relation Class Diagram.
- 1.4. Summary

The rest of this SRS is organized as follows:

**Section 2:** Gives the overall description of the Guttersnipe application. It contains the Use-Case diagram and descriptions for Guttersnipe. Section 2 also contains the assumptions and dependencies of the system.

**Section 3:** Gives specific software requirements and functionalities in the form of Mini Use- Case diagrams along with accompanying Collaboration diagrams, Finite State Machine of the system, and ER Class diagram of the system. This section also contains supplementary software requirements of the systems.

**Section 4: GUI Components:** The appendix contains user interface prototypes for the system, including many screenshots from the 0.2 release of the application.

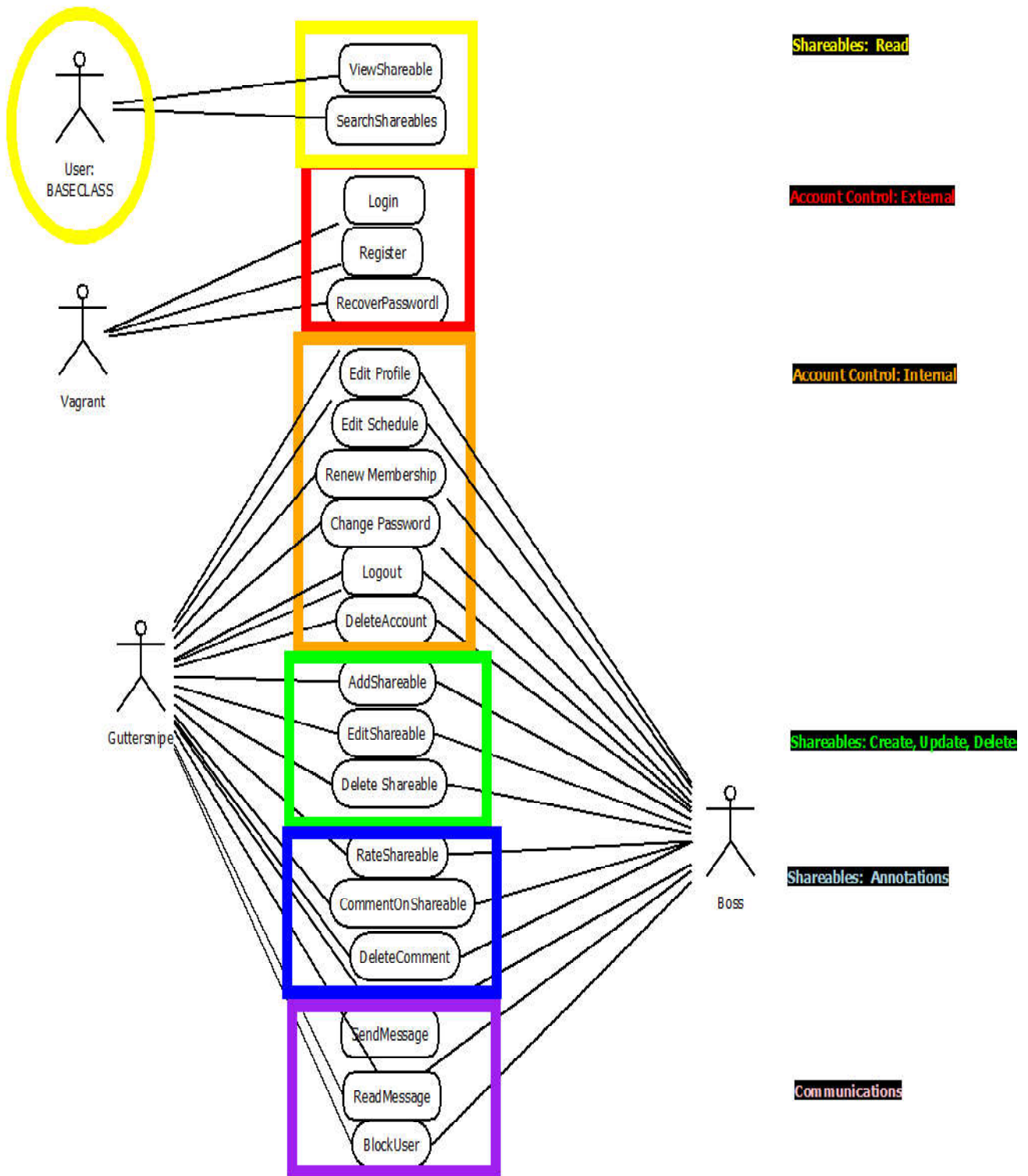
## 02: Overview

02.01: Use Case Diagrams and Descriptions

02.02: Server/Client Assumptions and Dependencies

## 2. Overview

### 2.1 User Case Diagram and Descriptions



## A. 02.01.01

## B. Usecases: USERS

### A. User (U)

#### **i. Shareables (Read)**

**a. View: Any U may view a search result.**

**b. SearchShareables: Any U may search for Shareable by Thing (Categorization, Tags), Space (Geolocation), and Time (Schedule)**

### B. Vagrant (V)

#### **i. Account Management (External)**

**a. Login: Any V can login to the system, which transforms V into a G or B .**

**b. Register: Any V may register to become a G.**

**c. RetrieveUsername: Any V may request a username reminder.**

**d. RecoverPassword: : Any V may request a password reminder.**

## **C. Guttersnipe (G)**

### **i. Account Management (Internal)**

- a. EditProfile:** Can edit information on own account.
- b. EditSchedule:** Can edit availability schedule.
- c. RenewMembership:** Can renew the terminal date of membership
- d. Logout:** Can log out of own account.
- e. ChangePassword:** Can reset own password.

### **ii. DeleteAccount: Can delete own account. Shareables (Create, Update, Delete)**

- a. CreateShareable:** Can add a Shareable resource.
- b. UpdateShareable:** Can update Shareable created by self.

### **iii. DeleteShareable: Can delete Shareable created by self.Shareables (Annotate)**

- a. RateShareable:** Can rate any shareable
- b. CommentOnShareable:** Can comment on any shareable.
- c. DeleteComment** Can delete own comment.

### **iv. Communications**

- a. SendMeetup:** Can communicate message (schedule + text) from other Guttersnipe.
- b. ReadMessage:** Can read message inbox.
- c. SetSchedule:** Can set availability calendar.
- d. BlockUser:** Can block any other user except Caretaker.

**D. Caretaker (C)**

**Caretaker has same capabilities as Guttersnipe, but they are unrestricted to apply to all system users. The relevant overrides are as follows:**

**i. UpdateShareable: Can update Shareable created by any user.**

**ii. DeleteShareable: Can delete Shareable created by any user.**

**\* Deleting a Shareable account triggers DeleteAccount, DeleteComments and DeleteShareables for that User.**

**iii. DeleteComment Can delete Comment written by any user.**

**\* Deleting a Comment triggers DeleteAccount, DeleteComments and DeleteShareables for that User.**

**iv. DeleteAccount: Can delete the account of any user, except for another Caretaker.**

**\* Deleting a user account triggers DeleteComments and DeleteShareables for that User.**

**v. BlockUser: Triggers a DeleteAccount option for the blocked user.**

**\* Blocking a user account triggers DeleteAccount, DeleteComments and DeleteShareables for that User.**



## 02.02.01

# Client / Server Assumptions and Dependencies

### Previous release

The previous release (0.2.1.5) of Guttersnipe was a MEAN stack application, utilizing Mongo/Mongoose, Node, Express, and Angular 1.x.

The deployment of a node application requires a server or PAAS that supports a node engine. We find this constraint too restrictive because many servers do not have a node engine, including the servers where we hope to do our initial deployments.

Although Mongo/NoSql is a fine technology, we believe that rapid, optimized database queries can best be done with an SQL database, which will obviate the need for a lot of the “middle-tier” post-processing of results from a database query.

### Current release

Client. The client requires a contemporary browser that supports standard HTML 5, CSS 3, and Javascript. It will use Angular 2.x as a front-end framework.

The application will be ported to mobile devices initially by taking advantage of libraries like phonegap which allow one to use the device webview to deploy web interfaces. Native Android/ iOS ports may be attempted as well.

For backend technologies, we sought a technology that could easily be ported to a variety of servers where no superuser access is required. We dislike bulky frameworks, preferring to add components as we need them. To this end, we have chosen the Flask / Python framework.

The most important platform to develop upon are the free phones given by government assistance (<http://newsroom.assurancewireless.com/custom-page/product-information>). Some of these have 1.8 inch screens. We are currently unsure of what toolkit we will need to use for the front end.

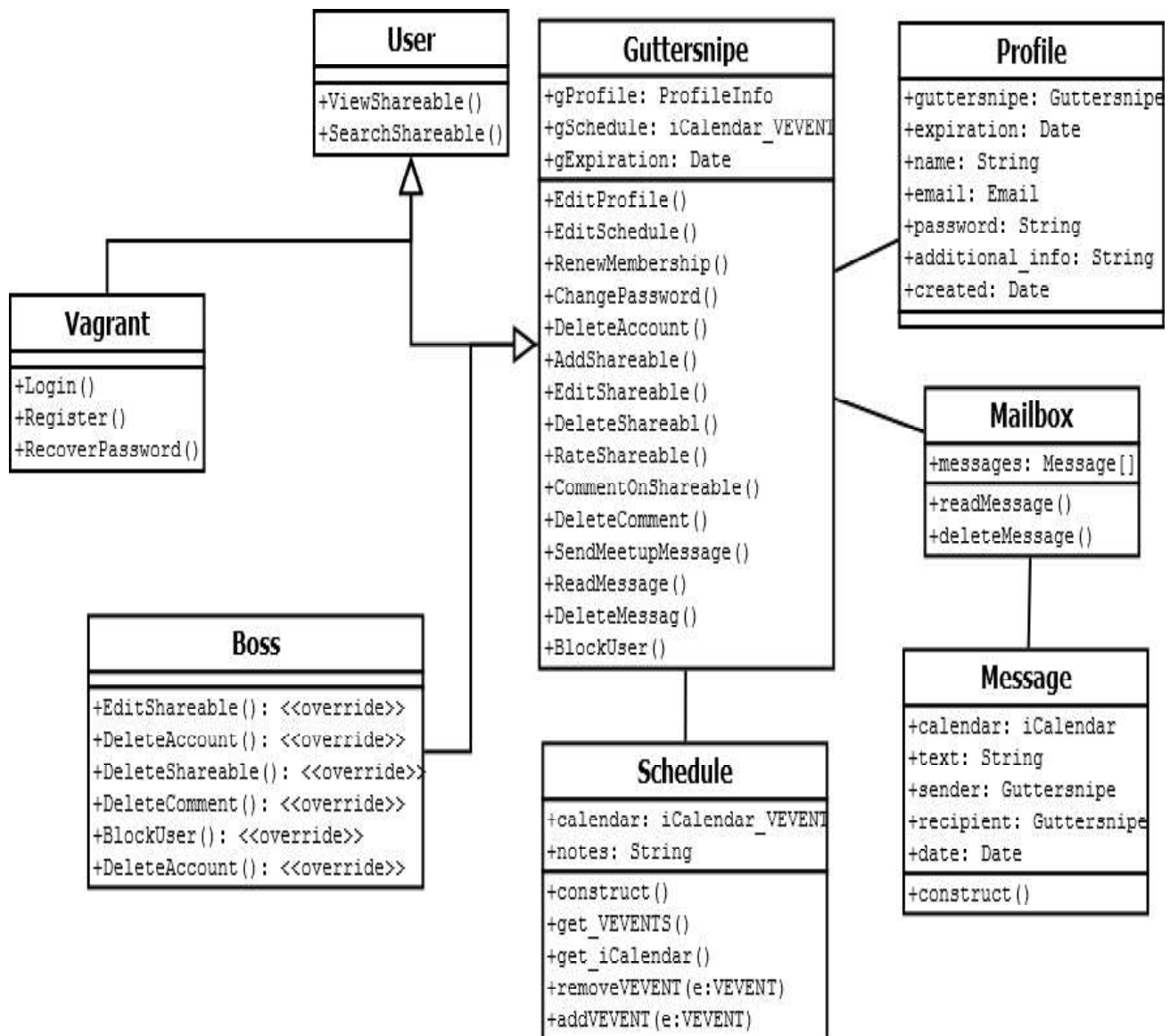
# SPECIFIC REQUIREMENTS

# Class D&D

## (diagrams && definitions)

# User Class

## (A) Diagram



### 03.02.01 User Class Definitions

#### User Class (abstract)

A User represents anyone using the application. It is an abstract class, so any actual user will be classified as a Vagrant, Guttersnipe, or Caretaker.

##### \* Methods:

- ++ViewShareable(): Any User may view a shareable
- ++SearchShareable(): Any User may search through shareables by selecting filters for Thing, Space, and Time

---

#### Vagrant (extends the User Class)

A Vagrant represents someone who is browsing the site and has not yet logged in

##### \* Methods:

- ++Login(): A Vagrant may log in to its own account
- ++Register() : A Vagrant may register to become a user
- ++RecoverUsername: A Vagrant may request to recover its Guttersnipe username
- ++RecoverPassword: :A Vagrant may request to recover its Guttersnipe username

---

#### Guttersnipe (extends the User Class)

A Guttersnipe represents a registered User who has logged into its own Account.

A

##### \*Attributes:

- ++gProfile: ProfileInfo : Personal Information about the Guttersnipe.
- ++gSchedule: iCalendar\_VEVENT: Availability information for the Guttersnipe
- ++gLocation: Space. The Guttersnipe's registered location
- ++gExpiration: Date. A Guttersnipe's account is a Temporary Autonomous Identification, which has an expiration date. This is done for security reasons. The expiration date may be extended at any time.
- ++ isAdmin: Identifies whether user is Guttersnipe or Caretaker.

##### \* Methods:

- ++EditProfile(): A guttersnipe may edit its own profile
- ++EditSchedule(): A guttersnipe may edit its own schedule
- ++EditLocation: A Guttersnipe may edit its own location.
- ++RenewMembership(): A Guttersnipe may renew the termination date of its membership.
- ++ChangePassword(). A Guttersnipe may change its own password.
- ++DeleteAccount(): A Guttersnipe may delete its own account.
- ++AddShareable(): A Guttersnipe may add a Shareable.
- ++EditShareable(): A Guttersnipe may edit a Shareable that it has added.
- ++DeleteShareable(): A Guttersnipe may delete a Shareable that it has added.
- ++RateShareable(): A Guttersnipe may rate any Shareable
- ++CommentOnShareable(): A Guttersnipe may comment on any Shareable
- ++DeleteComment (): A Guttersnipe may delete any comment that it has written.
- ++SendMeetupMessage(): A Guttersnipe may send a message to another Guttersnipe. This message contains schedule and text. It will enable
- ++ReadMessage(): A Guttersnipe may read a message.
- ++DeleteMessage(): A Guttersnipe may delete a message.
- ++BlockUser(): A Guttersnipe may block another

---

## Caretaker

A Caretaker is a registered user with administrative capacities. It has the same functionality as a Guttersnipe, but extends several of its methods.

### \* Methods:

- ++EditShareable() : A Caretaker may edit any shareable.
- ++DeleteAccount(): A Caretaker may delete any Guttersnipe account
- ++DeleteShareable () : A Caretaker may delete any shareable
- ++DeleteComment () : A Caretaker may delete any comment
- ++BlockUser () : When a Caretaker blocks a Guttersnipe, it may also delete that Guttersnipe's account.

# 03.02.01 User-Associated Object Definitions

## 1. Profile

The Profile contains personal information about the Guttersnipe or Caretaker. Only username is a required attribute. All others are optional.

- a. \*Attributes:
  - i. username: The name user uses to log in
  - ii. email (optional)
  - iii. full\_name (optional)
  - iv. password (optional)
  - v. additional\_info (optional): =Any additional details user wishes to add.

## 2. AvailabilitySchedule (optional)

A Guttersnipe may fill out a schedule to indicate it is most available for activities.

- a. \*Attributes:
  - i. availabilityCalendar: The user's availability
  - ii. notes; Any additional details

## 3. UserLocation (optional)

A Guttersnipe may indicate its location for participation in activities.

- a. \*Attributes:
  - b. Longitude
  - c. Latitude:
  - d. notes; Any additional details

## 4. Mailbox

A Guttersnipe has a mailbox where it can send and read messages

- a. \*Attributes:
  - i. messages: collection of messages.

## 5. Message

A message is intended to communicate between Guttersnipes their availability for activities.

- a. Attributes:
  - i. calendar: iCalendar: The user's availability for a certain activity.
  - ii. text = Text of the message
  - iii. sender Guttersnipe who sent message
  - iv. recipient = Guttersnipe who receives message
  - v. sendDate = Time Message was sent



## 03.02.01.05

# Guttersnipe Component Classes

### 1. Message

- a. Attributes:
  - i. calendar: iCalendar
  - ii. text: String
  - iii. sender: Guttersnipe
  - iv. recipient: Guttersnipe []
  - v. date: Date
- b. Functions
  - i. \_\_construct()

### 2. Profile

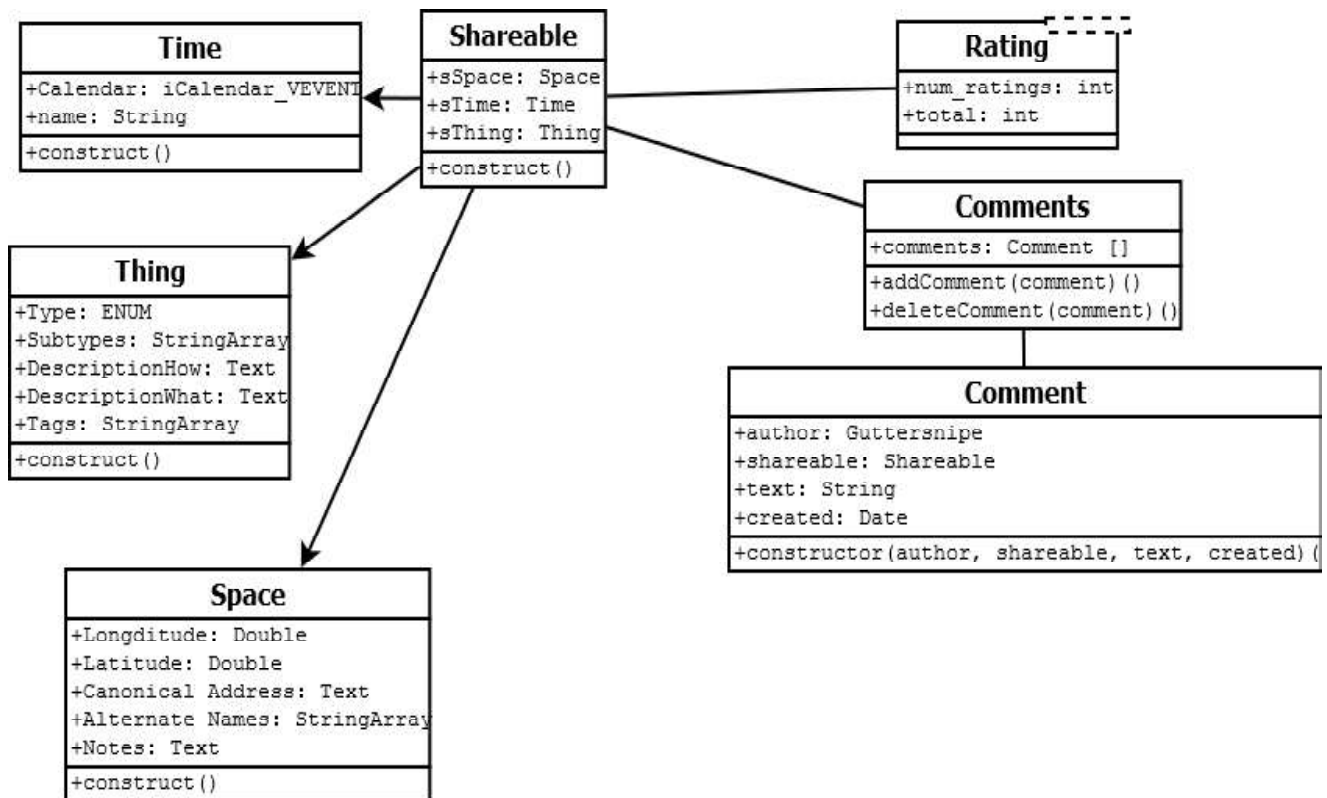
- a. Attributes:
  - i. guttersnipe: Guttersnipe
  - ii. expiration: Date
  - iii. name: String
  - iv. email: Email
  - v. password: String
  - vi. additional\_info: String
  - vii. created: Date
- b. Functions
  - i. \_\_construct()

### 3. Schedule

- a. Attributes:
  - i. calendar: sCalendar\_VEVENT
  - ii. notes: String
- b. Functions
  - i. \_\_construct()
  - ii. get\_VEVENTs() // Returns array of VEVENTs
  - iii. get\_iCalendar() // Returns iCalendar
  - iv. removeFromSchedule(VEVENT) // Removes VEVENT

# Shareable Class

## (A) Diagram



# A. 03.02.02 Shareable Class and Parts

## 1. Shareable

### a. \*Attributes:

- i. thing: Thing
- ii. space: Space
- iii. time: Time

### b. Functions

- i. `__construct(thing, space, time)` // Constructor for the Shareable class

## 2. Thing

### a. Attributes:

- i. type: ENUM
- ii. subtypes: String []
- iii. descriptionHow: String
- iv. descriptionWhat: String-
- v. tags: StringArray []

### b. Functions

- i. \_\_construct()

## 3. Space

### a. \*Attributes:

- i. longitude: Double
- ii. latitude: Double
- iii. canonical Address: String
- iv. alternate Names: String[]
- v. notes: String

### b. Functions

- i. \_\_construct()

## 4. Time

a. \*Attributes:

i. calendar: sCalendar\_VEVENT

ii. notes: String

b. Functions

i. \_\_construct()'''

## 5. Rating

a. Attributes:

i. num\_rating: int

ii. total: int

b. Functions

i. \_\_construct()

ii. getRating() : float'''

## 6. Comments

a. Attributes:

i. comments: Comment []

b. Functions

i. \_\_construct()

ii. addComment()

iii. deleteComment()

iv. deleteByGuttersnipeID(int gsnipe\_id)

## 7. Comment

### a. Attributes:

- i. author: Guttersnipe
- ii. shareable: Shareable
- iii. text: String
- iv. created: Date

### b. Functions

- i. function \_\_construct(\$commentID, \$commentText, \$userName, \$timestamp) // Constructor for the Comment class

## B. Collaboration Diagrams, Database

### 8. Collaboration Diagram Objects:

#### a. Managers

- i. (SearchManager, BrowseManager, MovieViewManager, LoginManager, RegisterManager, MovieRatingsInterface, MovieProcessor, CommentManager, CartManager, CheckoutManager, PasswordControl, LogoutControl, CommentControl, FlaggedCommentsManager, WarningSystem, DeleteManager)

#### b. GUI es

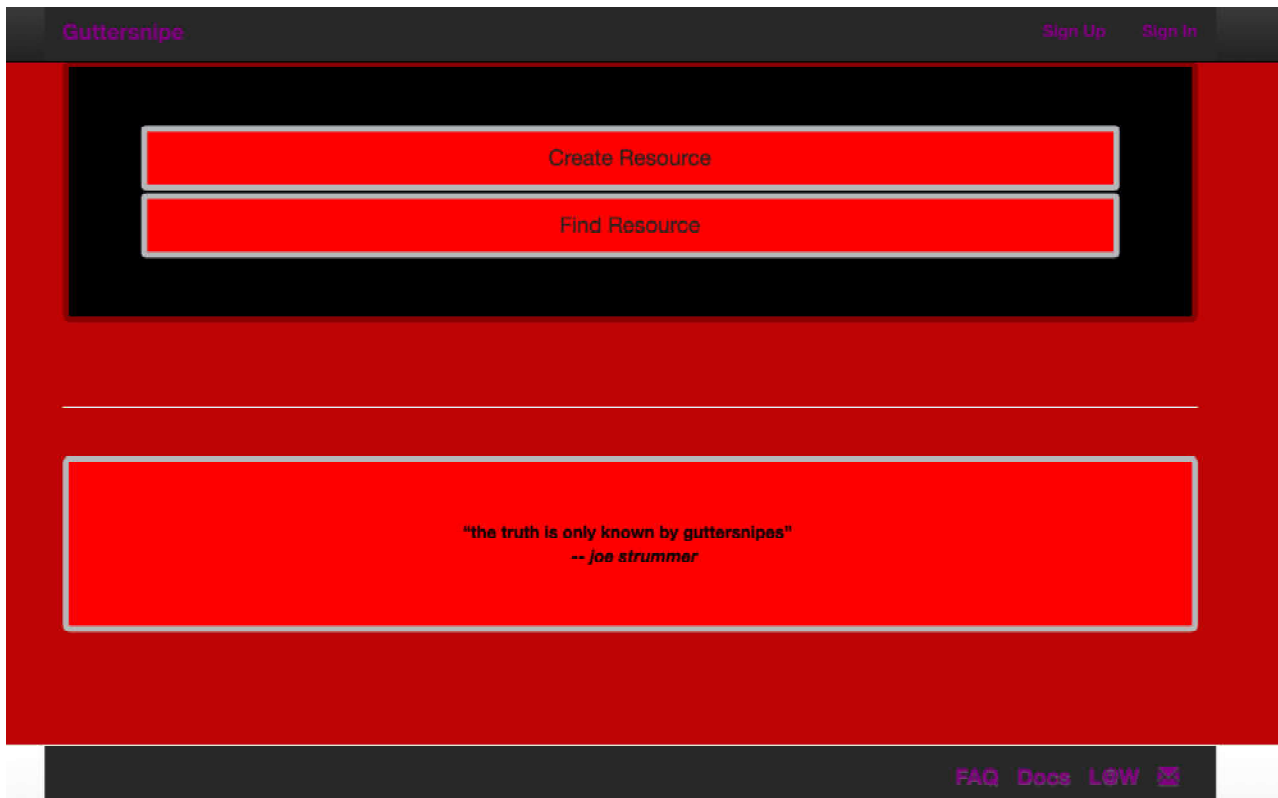
- i. (SearchGUI, BrowseGUI, ViewMovieGUI, LoginGUI, MainCustomerPage, MainAdminPage, RegisterGUI, RegistrationSuccessGUI, MoviesBought, MoviePlayerInterface, CommentGUI, ViewCartGUI, CartGUI, CheckoutGUI, ConfirmationGUI, ResetPasswordGUI,

# GUI COMPONENTS

**Screenshots from 0.2 release and Future Wireframes**

# 04.01.01

## Front Page:



Front Page shows

- Top Menu Links
  - Home (here)
  - Sign Up
  - Sign In
- Body Links
  - Create Resource
  - Find Resource
  - “the truth is only known by guttersnipes” – joe strummer
- Bottom Menu Links
  - FAQ
  - Documentation
  - Legal
  - Contact



# 04.02.01

## CreateShareable

### Start

In order to add a Shareable to the System, the User can enter data in the following Wizard.

This represents the first step of that wizard.

It shows buttons for consent and negation.

-----

Create New Resource

Through your usage of Guttersnipe, you agree to not put yourself or any other person in legal jeopardy.

Negate   Consent

You are free to use Guttersnipe as you wish.

# 04.02.02

## CreateShareable

### Instructions

### Create New Resource

Resource = Thing + Place + Time

Example: Free meal in Prospect Park every Wednesday from 4PM to 9PM

To report a Resource on Guttersnipe, please do the following:

1. Describe Resource
2. Map Place
3. Schedule Time
4. Confirm Resource Report

Create Resource

#### Instructions on how to create Shareable

---

##### Create New Resource

Resource = Thing + Place + Time

Example: Free meal in Prospect Park every Wednesday from 4PM to 9PM

To report a Resource on Guttersnipe, please do the following:

Describe Resource

Map Place

Schedule Time

Confirm Resource Report

# 04.02.03

## CreateShareable

### Describe

**Create New Resource**

**Describe**

**Headline**  
Short explanation  
You must provide a headline.

**Summary**  
Resource Description  
You must provide a summary.

**Method of Access**  
How do I acquire resource?  
You must provide a method of access.

**Notes**  
Additional Notes

Confirm Description

Form for entering textual data about the Shareable

- Headline
- Summary
- Method of Access
- Additional Notes

## 04.01.04(a) CreateShareable Classify (1)



The screenshot shows a dark-themed interface with a red border. At the top left, the word "Classify" is written in red. Below it, the text "Choose a resource type:" is also in red. There are three white buttons with black text: "food", "medical", and "housing".

Allows Guttersnipe to categorize Shareable as System-defined type

- Food
- Medical
- Housing
- Transport (new)

## 04.02.04(b) CreateShareable Classify (2)

The screenshot shows a web form titled "Classify" with a red header. The form is divided into three main sections: "Type", "Subtypes", and "Details".

- Type:** A text input field containing the word "food" in red. Below it is a red "Edit" button.
- Subtypes:** This section is divided into two columns.
  - Available Subtypes:** A list of three blue buttons: "Free Communal", "Food Donation", and "Dumpster".
  - Your Selections:** A single blue button labeled "Food Not Bombs".
- Details:** This section is also divided into two columns.
  - Add a Detail:** A text input field with a red border and a red plus icon below it.
  - Your Details:** A single blue button labeled "vegan".

At the bottom of the form is a grey button labeled "Confirm Classification".

Allows Guttersnipe to add system-defined **Subtypes** and Guttersnipe-defined **Tags** to Shareable.

## 04.02.05(b) CreateShareable Map Shareable Location



The screenshot shows a web form titled "Map Shareable Location". At the top left is a "Map" label. Below it is a large rectangular area for a map, which is currently blank. To the right of the map area is a label "Address of Resource :" followed by a text input field. Below the input field is a blue button labeled "Find Address". Below the map area is a "Notes" label followed by a text input field. At the bottom left is a button labeled "Confirm Map".

Shows Map centered on User's current location or Times Square if location is unavailable. Allows user to specify address of the Shareable.

# 04.02.06(a) CreateShareable Schedule (1)

**Schedule**

month week day Sep 11 - 17 2016 today < >

	Sun 9/11	Mon 9/12	Tue 9/13	Wed 9/14	Thu 9/15	Fri 9/16	Sat 9/17
12pm							
1pm							
2pm							
3pm							
4pm							
5pm							
6pm							

Schedule:

None

Shows blank calendar. When Guttersnipe clicks on a date, it will be shown popup in next Figure.

## 04.02.06(a) CreateShareable: Schedule (1)

**Choose Schedule**

☒ Repeating Event

Event occurs every Monday

Start Time

▲	▲	
02	:	00 PM
▼	▼	

Duration

▲	▲	
03	:	30 PM
▼	▼	

Cancel OK

Shows blank calendar. When Guttersnipe clicks on a date, it will be shown popup in next Figure.



## 04.02.06(c) CreateShareable Schedule (3)

The screenshot displays a web interface for creating a shareable schedule. At the top, the word "Schedule" is written in red. Below it, there are three buttons: "month", "week", and "day". To the right of these buttons, the date "Sep 11 - 17 2016" is shown in red. Further right, there is a "today" button and two navigation arrows. Below this header is a calendar grid with columns for each day from Sunday 9/11 to Saturday 9/17. The rows represent time slots from 12pm to 6pm. A blue event box labeled "2:00 - 3:30 Event" is placed in the Monday 9/12 column, spanning the 2pm and 3pm slots. Below the calendar is a section titled "Schedule:" in a light gray box. Underneath this title is a table with the following columns: "Date", "Repeats", "Start Time", "End Time", and "Delete". The table contains one row with the following data: "Every Monday", "02:00 PM", "03:30 PM", and a delete icon (an 'X' in a square). Below the table is a "Notes" section with a text input field containing the text "GET THERE EARLY!!!".

	Sun 9/11	Mon 9/12	Tue 9/13	Wed 9/14	Thu 9/15	Fri 9/16	Sat 9/17
12pm							
1pm							
2pm		2:00 - 3:30 Event					
3pm							
4pm							
5pm							
6pm							

Date	Repeats	Start Time	End Time	Delete
	Every Monday	02:00 PM	03:30 PM	X

Notes

GET THERE EARLY!!!

A calendar is shown once the user has selected a new schedule for the event.  
A list of schedules is shown under the calendar.  
The user can delete any schedules which are incorrect.

## **04.03. SearchShareable**

# 04.03.01

## SearchShareable

### Results List

The screenshot displays the 'Resources' section of the SearchShareable application. It features a list of three resource cards, each with a 'Full Record' link. The cards are organized into columns: 'Headline', 'Thing' (with 'Type' and 'Subtypes'), 'Place', and 'Time'. The first card is for 'La Bagel Delight', the second for 'Trader Joe's', and the third for 'Perlestrand'. Each card also includes a 'Created by' timestamp and a 'by' field.

**Resources**

[List](#) [Map](#) [Compare](#)

Headline	Thing	Place	Time	Full Record
La Bagel Delight	Type • food Subtypes • dumpster	104 Riving St Brooklyn, NY 11201	• Every Monday From Monday, 12:00 AM To Monday, 2:59 AM • Every Tuesday From Tuesday, 12:00 AM To Tuesday, 2:59 AM	<a href="#">Full Record</a>
Created by Apr 16, 2015 1:21:24 PM by				
Trader Joe's	Type • food Subtypes • dumpster	130 Court St Brooklyn, NY 11201		<a href="#">Full Record</a>
Created by Apr 16, 2015 1:21:24 PM by				
Perlestrand	Type • food Subtypes • dumpster	175 Remsen St Brooklyn, NY 11201	• Every Monday From Monday, 12:45 AM To Monday, 2:59 AM	<a href="#">Full Record</a>
Created by Apr 16, 2015 1:21:24 PM by				

Filter Resources by

[Thing](#) [Place](#) [Time](#)

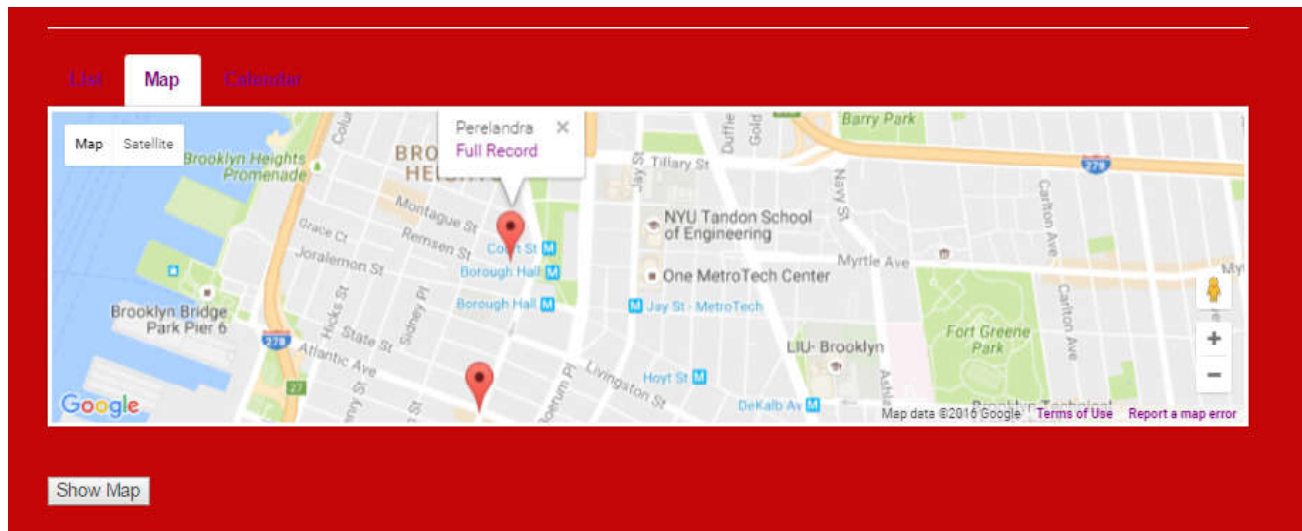
Shows the results of the Shareable Search as a list.  
Columns for

- Headline
- Thing (Type && Subtype)
- Place
- Times
- Link to Full Record

# 04.03.02

## SearchShareable

### Results Map



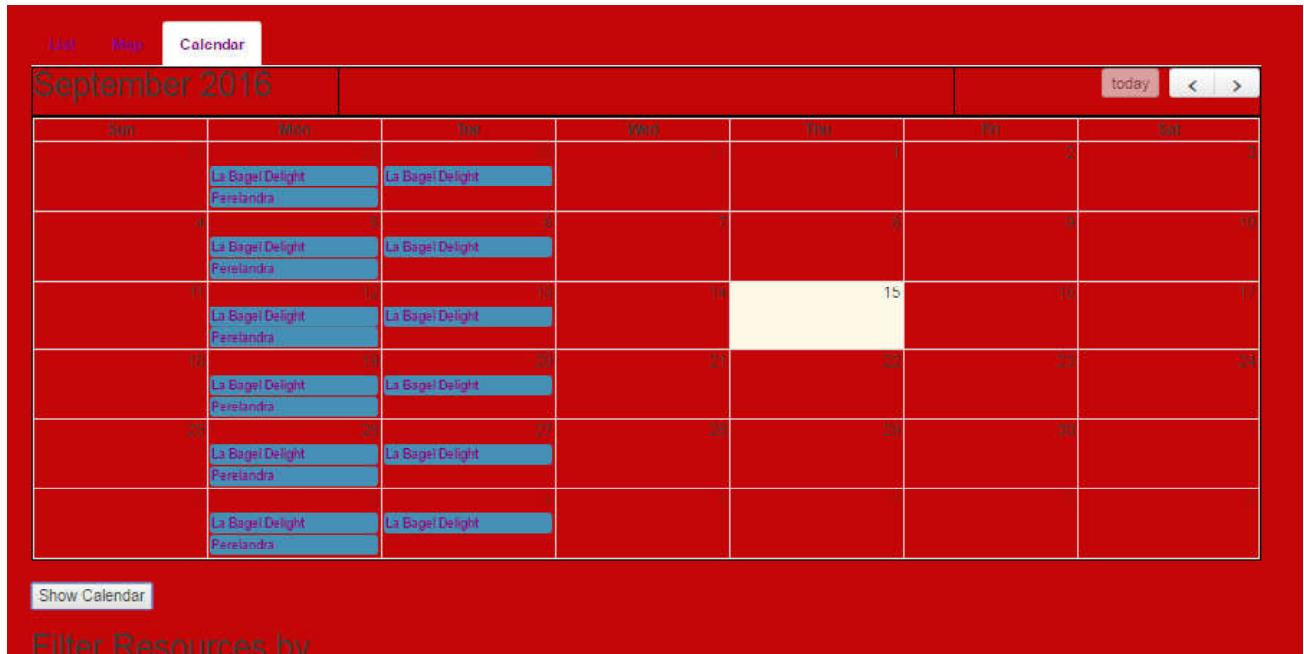
Shows Shareable Search Results as a Map.

Each point on the map has a tooltip that opens up a window with a Summary and a link to the Shareable.

# 04.03.03

## SearchShareable

### Results Calendar



Shows Shareable Search results as a Calendar.  
Each entry on the calendar a link that can be clicked for the full record.

# 04.03.03

## SearchShareable

### Search Filters



Search Interface will have 3 filters. These have not been designed yet

- Thing
  - By Type and Subtype
  - By Tags
- Time
  - By Date/Time
- Space
  - Choose Map Location

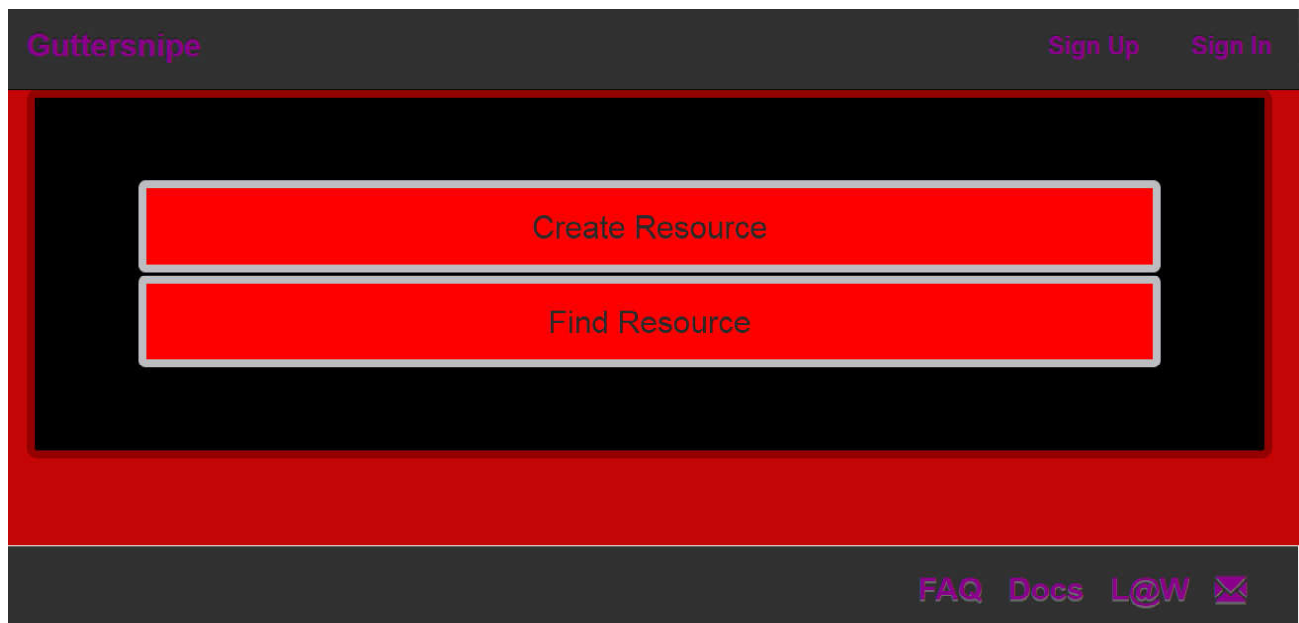
User will be able to refine these 3 search filters and submit a search query when done.

## **04.03. Account Management**

# 04.04.01

## Account Management

### Sign In and Sign Up Buttons

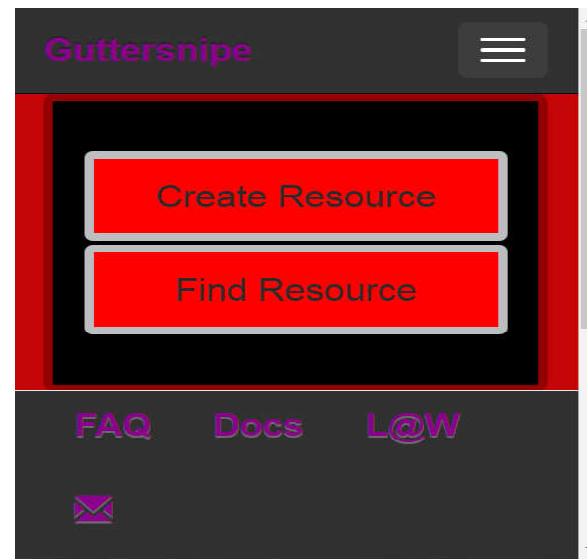


In the top right corner of the menu bar are buttons for “Sign Up” and “Sign In”



## 04.04.02

# Account Management Smaller Screens

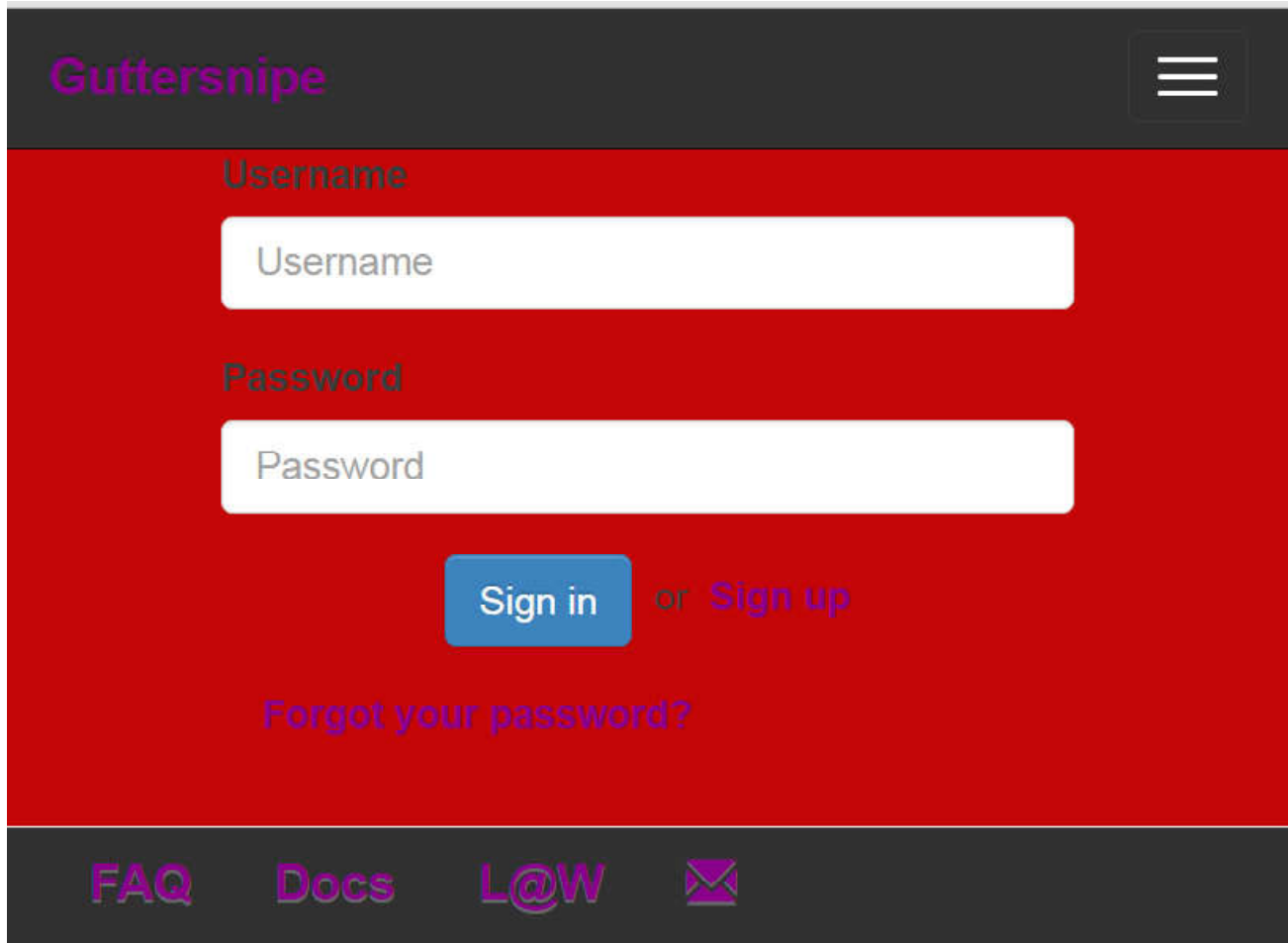


On Smaller Screens, users can click on the top right burger-shaped menu button to display the Sign Up and Sign In buttons

# 04.04.05

## Account Management

### Sign In



The image shows a web form for signing in to 'Guttersnipe'. The form is set against a red background. At the top, there is a dark grey header with the 'Guttersnipe' logo in purple on the left and a hamburger menu icon on the right. The form itself contains two white input fields: one for 'Username' and one for 'Password'. Below these fields is a blue 'Sign in' button, followed by the text 'or' and a purple 'Sign up' link. A purple link for 'Forgot your password?' is positioned below the sign-in options. At the bottom of the form, there is a dark grey footer containing four purple links: 'FAQ', 'Docs', 'L@W', and an email icon.

Guttersnipe

Username

Username

Password

Password

Sign in or Sign up

Forgot your password?

FAQ Docs L@W

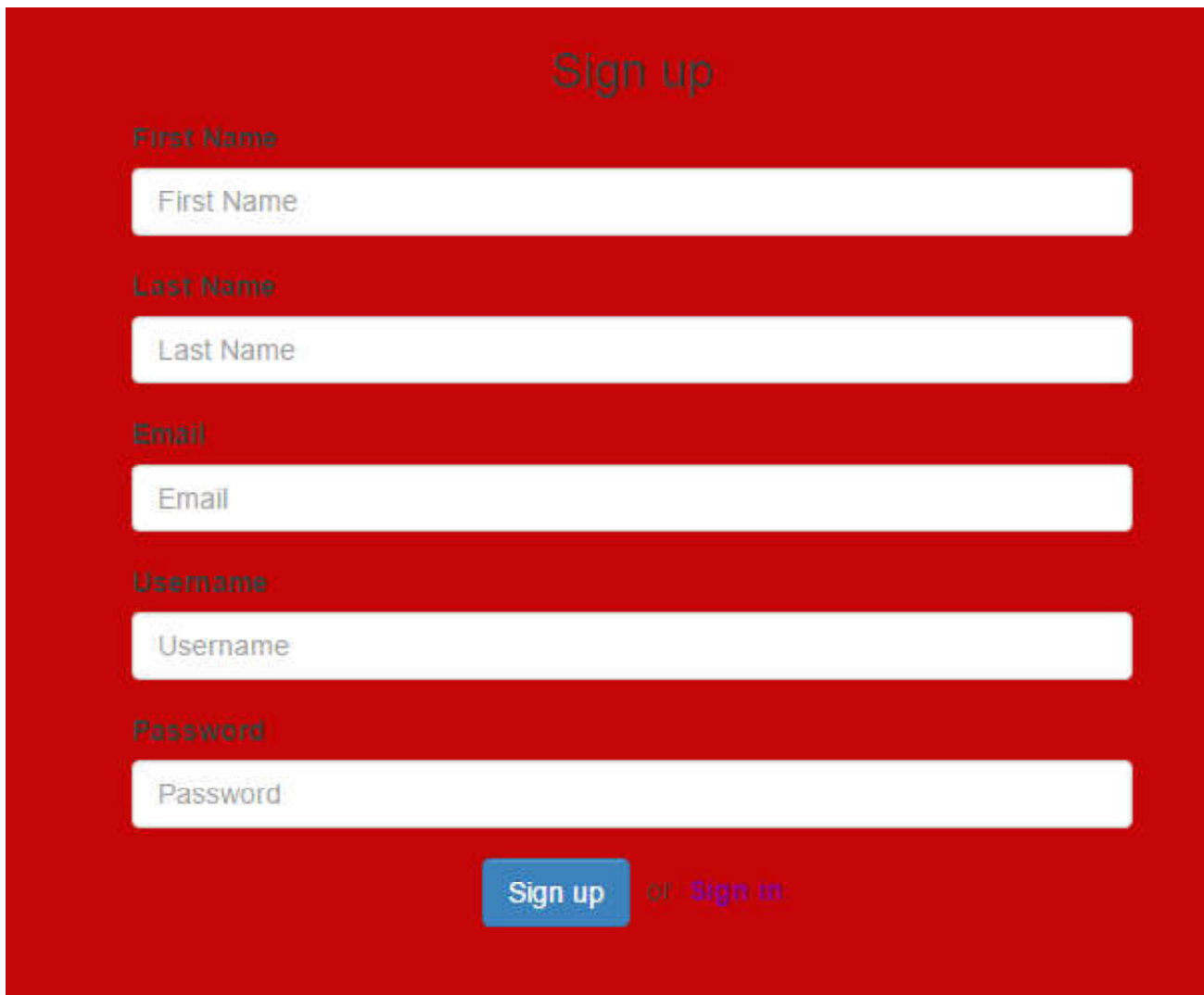
Shows form

- User
- Password
- Sign In Button

# 04.04.04

## Account Management

### Sign Up



The image shows a 'Sign up' form on a solid red background. The title 'Sign up' is centered at the top in a light gray font. Below it, there are five input fields, each with a label above it: 'First Name', 'Last Name', 'Email', 'Username', and 'Password'. The labels are in a light gray font. The input fields are white with rounded corners and contain placeholder text in a light gray font: 'First Name', 'Last Name', 'Email', 'Username', and 'Password'. At the bottom of the form, there is a blue button with the text 'Sign up' in white, followed by the text 'or Sign in:' in a light gray font.

Sign up

First Name

Last Name

Email

Username

Password

Sign up or Sign in:

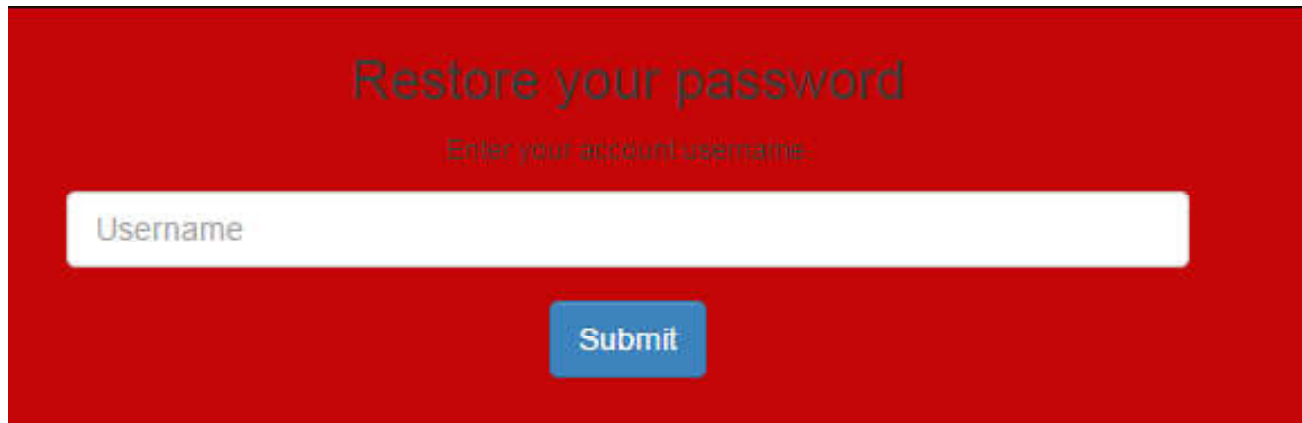
Signup Page asks user to fill in form:

- Username:
- Expiration Date
- Email (optional)
- Password (optional)

## 04.04.05

# Account Management

## Restore Password



The image shows a web form for restoring a password. It has a solid red background. At the top, the text "Restore your password" is displayed in a large, white, sans-serif font. Below this, in a smaller white font, is the instruction "Enter your account username:". Underneath the instruction is a long, white rectangular input field with the placeholder text "Username" in a light gray font. At the bottom center of the form is a blue rectangular button with the word "Submit" in white text.

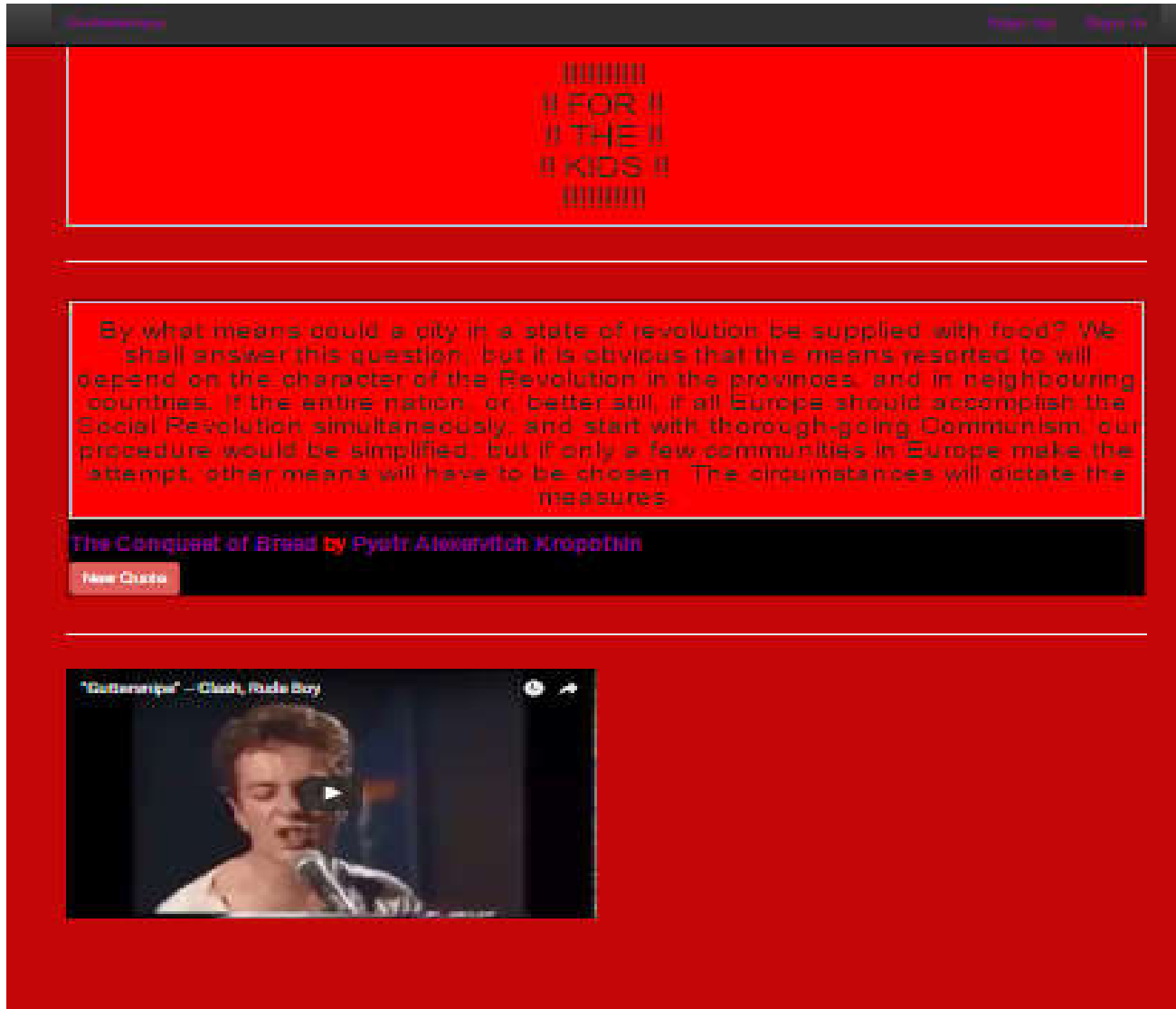
Provides a button for asking system to send user a password reset form.

# Documentation

# **DOCUMENTATION: GENERAL**

# 04.05.01

## Documentation: General Mission Page



Mission page shows

- “FOR THE KIDS!!!
- Kropotkin Quotes Widget
- Joe Strummer “guttersnipe” video.

04.05.02

# Documentation: General FAQ Page



Displays a FAQ about the project.

This is same content as section “01\_01: Proposal” of the current document.



# **DOCUMENTATION:**

## **2013 Presentation**

~~04.05.03.01~~

# ~~Documentation: 2013 Presentation Start Page~~

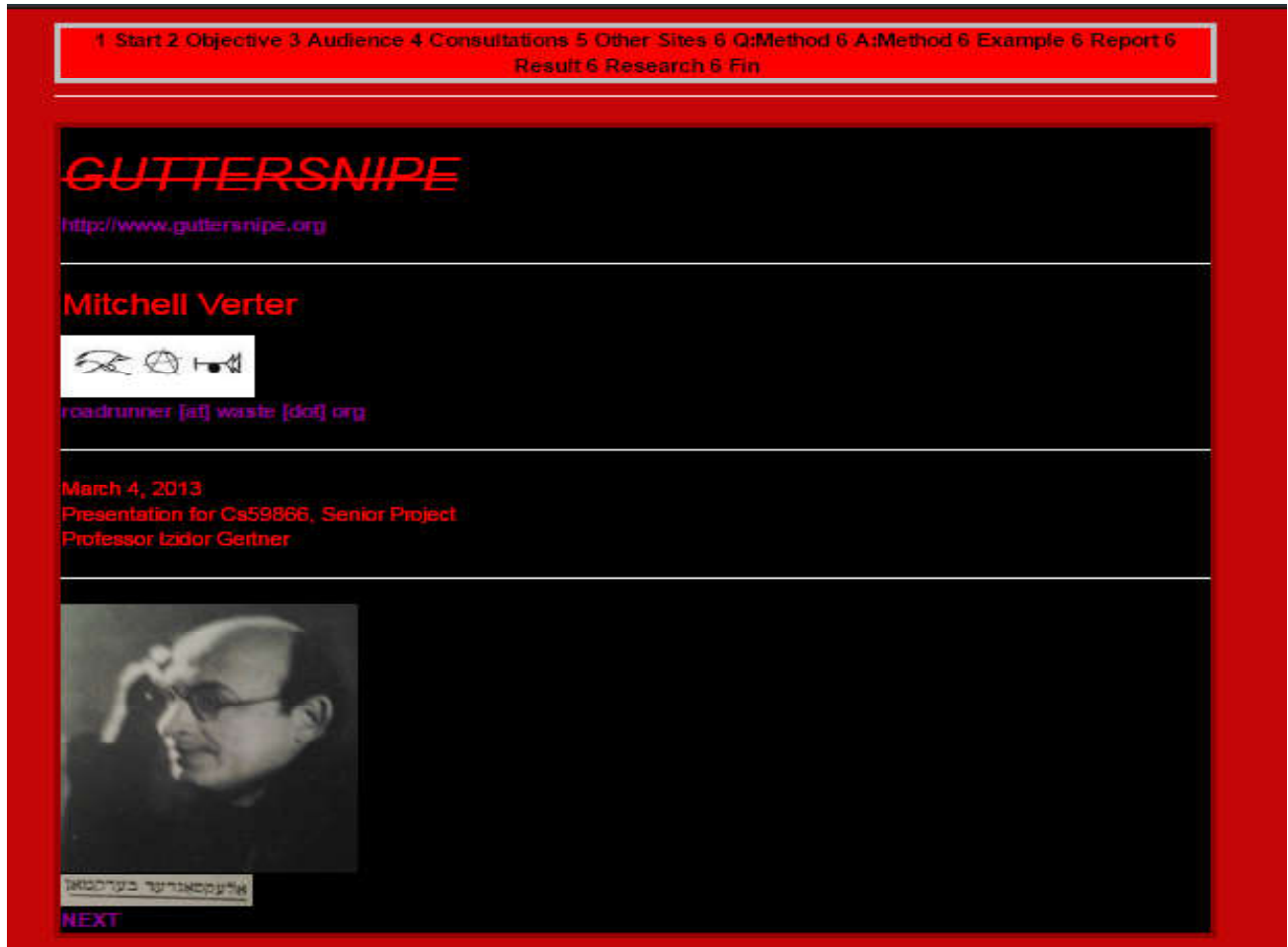


~~From 2013 Presentation:  
Section 6:  
START PAGE~~

~~The following slides are taken from my 2013 presentation prepared for what I intended to be my senior project at City College of New York~~

04.05.03.02

## Documentation: 2013 Presentation Front Page



From 2013 Presentation:

Section 6:

FRONT PAGE

=====

GUTTERSNIPE

<http://www.guttersnipe.org>

Mitchell Verter

roadrunner [at] waste [dot] org

March 4, 2013

Presentation for Cs59866, Senior Project

Professor Izidor Gertner

<Picture of Alexander Berkman>

04.05.03.03

# Documentation: 2013 Presentation Objective



From 2013 Presentation:

Section 6:

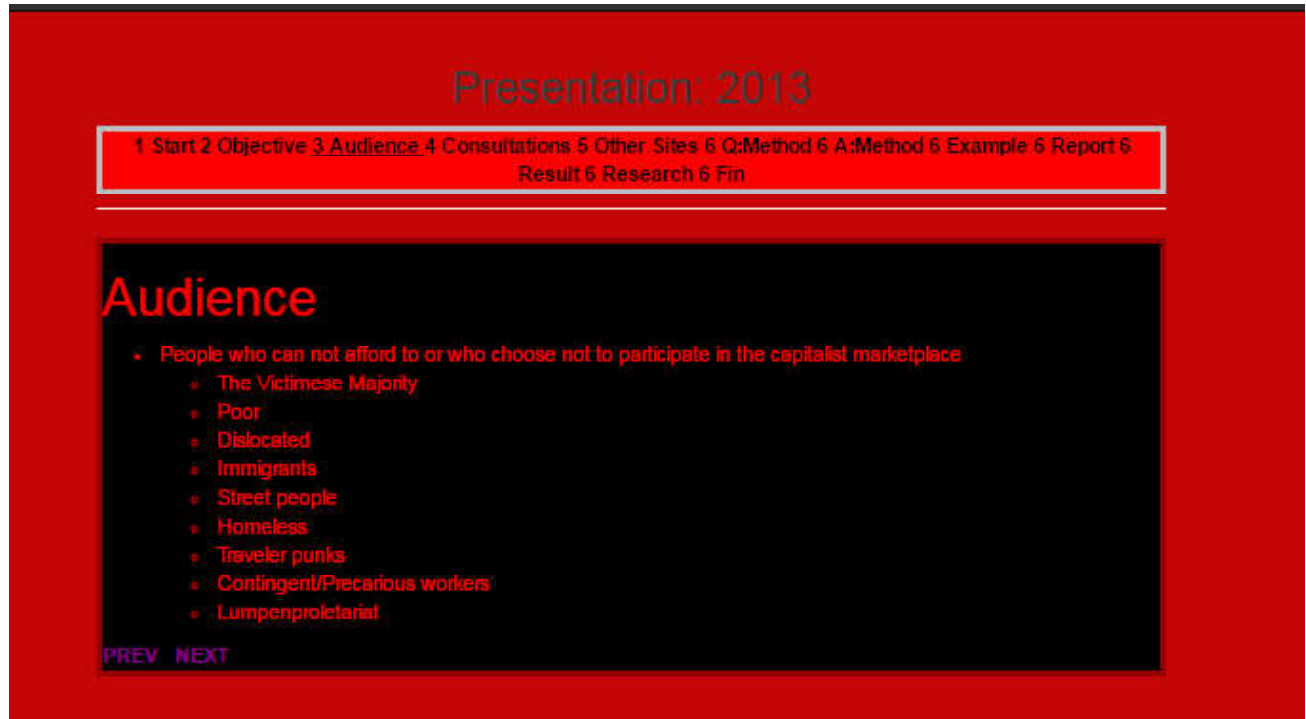
OBJECTIVE

=====

- To overthrow capitalism by helping to establish mediums of exchange outside of the capitalist marketplace.
- “Over a billion human beings live in absolute poverty, suffering from chronic malnutrition and other ills, while we have much more than an adequate material basis for a good life for all.” – John Clark, *The Impossible Community Realizing Communitarian Anarchism*

04.05.03.04

# Documentation: 2013 Presentation Audience



From 2013 Presentation:  
Section 6:

## AUDIENCE

=====

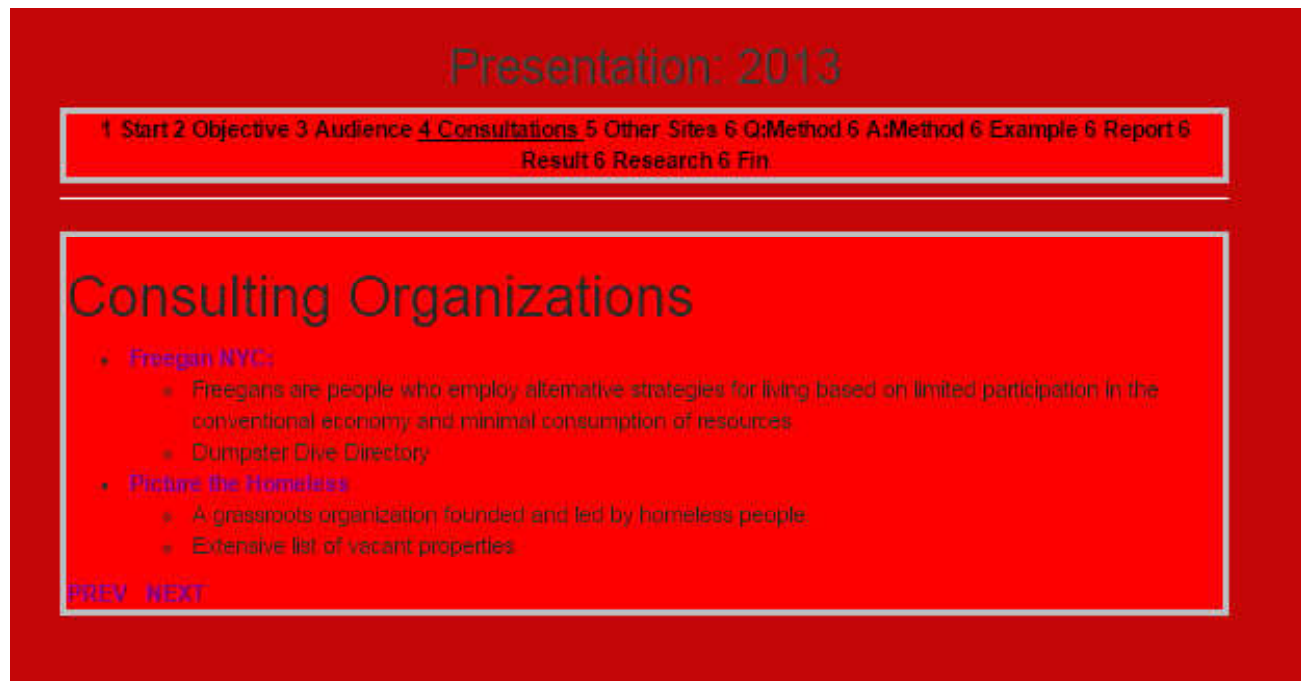
People who can not afford to or who choose not to participate in the capitalist marketplace

- The Victimese Majority
- Poor
- Dislocated
- Immigrants
- Street people
- Homeless
- Traveler punks
- Contingent/Precarious workers
- Lumpenproletariat

**04.05.03.05**

## **Documentation: 2013 Presentation**

### **Consulting Organizations**



From 2013 Presentation:

Section 6:

CONSULTING ORGANIZATIONS

=====

### Consulting Organizations

- **Freegan NYC:**
  - Freegans are people who employ alternative strategies for living based on limited participation in the conventional economy and minimal consumption of resources.
  - Dumpster Dive Directory
- **Picture the Homeless**
  - A grassroots organization founded and led by homeless people.
  - Extensive list of vacant properties

04.05.03.06

# Documentation: 2013 Presentation Other Sites



From 2013 Presentation:

Section 6:

OTHER SITES

=====

## Other Web Sites

- Other Websites
  - Squat.net
  - Foodsharing (Germany)

04.05.03.07

## Documentation: 2013 Presentation Question of Method



From 2013 Presentation:  
Section 6:  
QUESTION OF METHOD  
=====

Question of Method  
. How do you get to Carnegie Hall?



04.05.03.08

# Documentation: 2013 Presentation Answer of Method

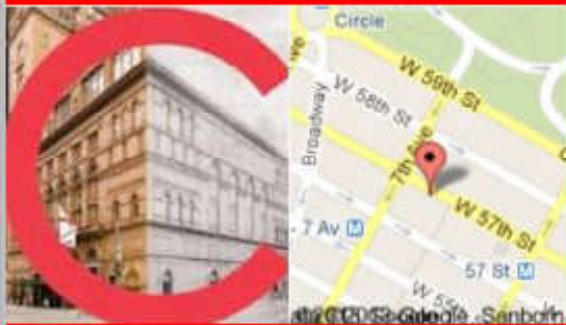
Presentation: 2013

1 Start 2 Objective 3 Audience 4 Consultations 5 Other Sites 6 Q:Method 6 A:Method 6 Example 6 Report 6 Result 6 Research 6 Fin

Answer of Method

Answer: Geolocation

- 881 7th Ave  
New York, NY 10019



PREV NEXT

From 2013 Presentation:  
Section 6:  
ANSWER OF METHOD  
=====

Answer: Geolocation  
881 7th Ave  
New York, NY 10019

04.05.03.09

## Documentation: 2013 Presentation Example (2002)



The following pages are taken from a previous incarnation of this project.  
Designed in 2002, these pages are still available at:  
<https://www.waste.org/~roadrunner/squat/>

04.05.03.12

## Documentation: 2013 Presentation Submit Property Report GUI (2002)

Presentation: 2013

1 Start 2 Objective 3 Audience 4 Consultations 5 Other Sites 6 Q:Method 7 A:Method 8 Example 9 Report 10 Result 11 Research 12 Pin

### Submit a property report

Use the following form to send information about a vacant property.

For example, to report on 1318 Park Street in San Francisco, California, 94109, type in "1318" under Number, type in "Park" under Name, choose "Street" under Suffix, type in "94109" under Zip Code.

You must fill out all fields. If you are having difficulty determining the zip code, you should try looking at some [zillow maps](#).

Then describe what you have found under Report.

**Street Address:**

Number	Name	Suffix
<input type="text"/>	<input type="text"/>	<input type="text" value="Alley"/>

**Zip Code:**

Zip Code
<input type="text"/>

**Report:**

Please describe what you have found out about this property.

**Disclaimer:**

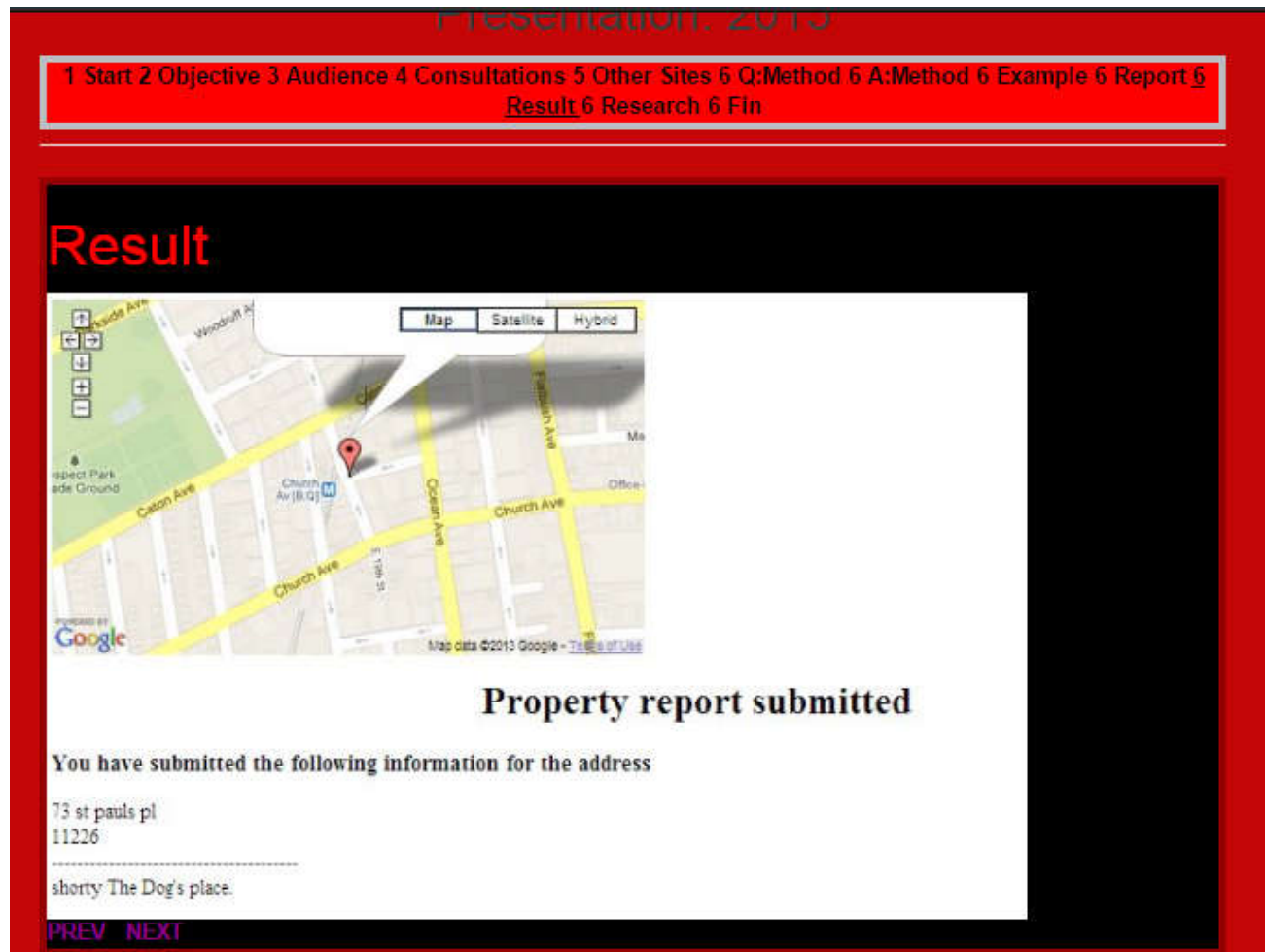
One does not know who might be using this service, so the I would recommend that users avoid giving out personal information or detailing one's plans on how to use any given space.

[W.A.S.T.E. - want want want main page](#)  
[Find vacant properties](#)  
[Look at maps](#)  
[Read PropertyBot's insights on this project](#)  
[Privacy](#) [About](#)

Shows the Submit Property Report interface from the 2002 deployment.

04.05.03.11

# Documentation: 2013 Presentation Property Submission Response (2002)



Shows a Resulting page for a Property Submission from the 2002 deployment.

04.05.03.12

## Documentation: 2013 Presentation Property Search (2002)

### Find vacant properties

Use the following form to research a vacant property and its neighborhood.

For example, to research 1315 Polk Street in San Francisco, California, 94109, type in "1315" under Number, type in "Polk" under Name, choose "Street" under Suffix, type in "94109" under Zip Code.

You must fill out all fields. If you are having difficulty determining the zip code, you should try looking at some [swell maps](#).

Then describe what you have found under Report.

#### Street Address:

Number	Name	Suffix
<input type="text"/>	<input type="text"/>	<input type="text" value="Alley"/>

#### Zip Code:

Zip Code
<input type="text"/>

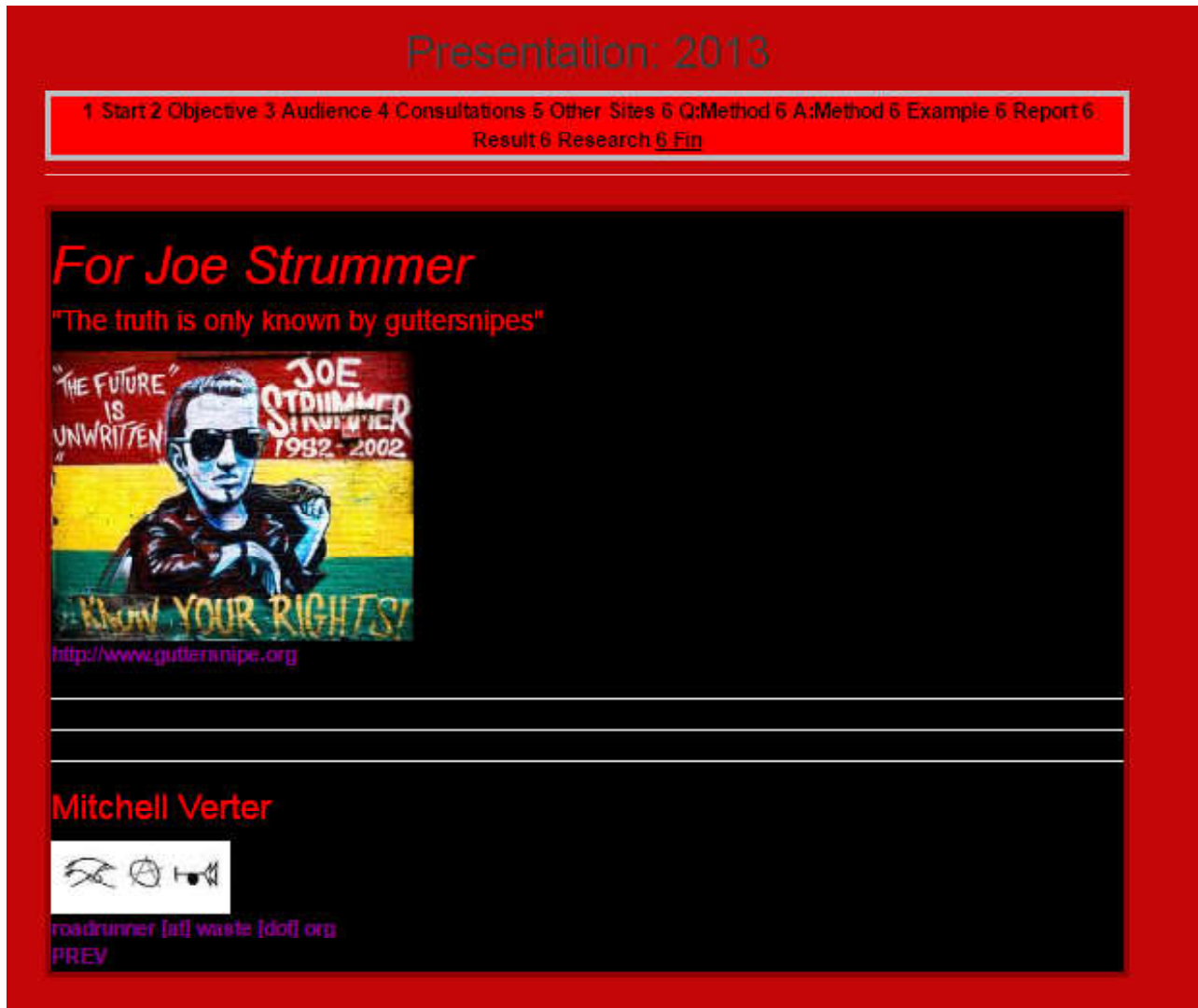
Are you having problems determining the address? Maybe you should try looking at some [Swell Maps](#).

[W.A.S.T.E. squat, want squat main page](#)  
[Report a vacant property](#)  
[Look at maps](#)  
[Read Kropotkin's thoughts on this project](#)  
[PREV](#) [NEXT](#)

Shows the Property Search GUI from the 2002 deployment.

04.05.03.13

# Documentation: 2013 Presentation FIN



Final Page from 2013 Presentation:

For Joe Strummer  
“The truth is only known by guttersnipes”

<http://www.guttersnipe.org>

Mitchell Verter

[roadrunner \[at\] waste \[dot\] org](mailto:roadrunner@waste.org)

# **DOCUMENTATION: ADMINISTRATIVE INFORMATION PAGES**

# 04.05.01

## Documentation: Administrative

### L@W Page



Shows Legal Information

=====

You are free to use GutterSnipe as you wish.

All Wrongs Righted  
All Rites Reversed

GNU General Public License



# Credits Page



Shows creator and inspirations for the project.

# 04.05.04.03

## Documentation: Administrative Contact Information

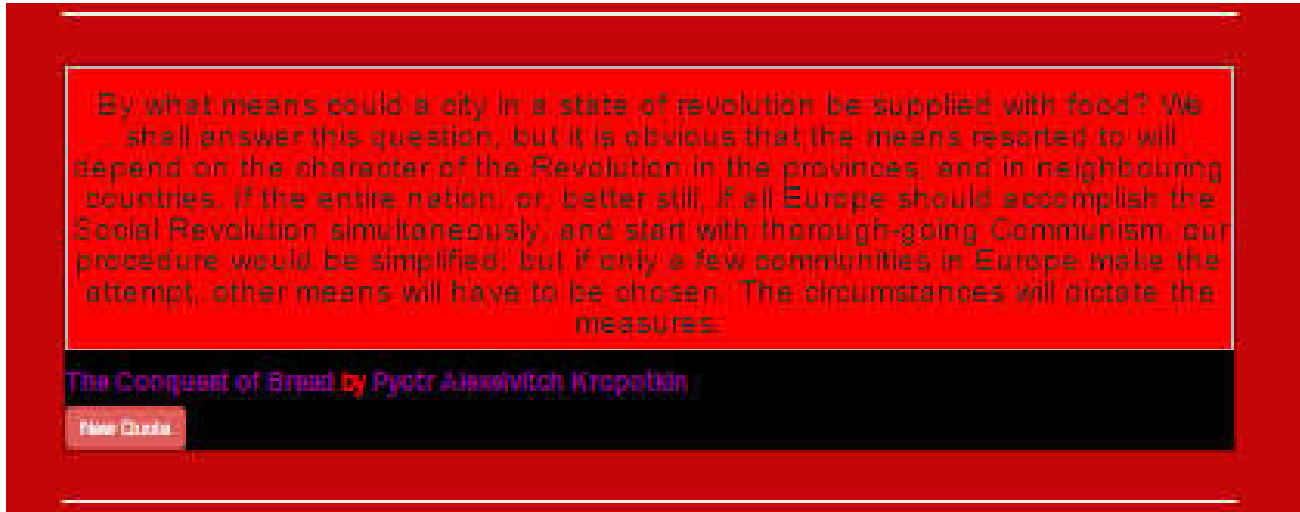


Displays email address and a clickable picture to send mail.

# Kropotkin Quote Widget

## 04.06.01

### Kropotkin Quote

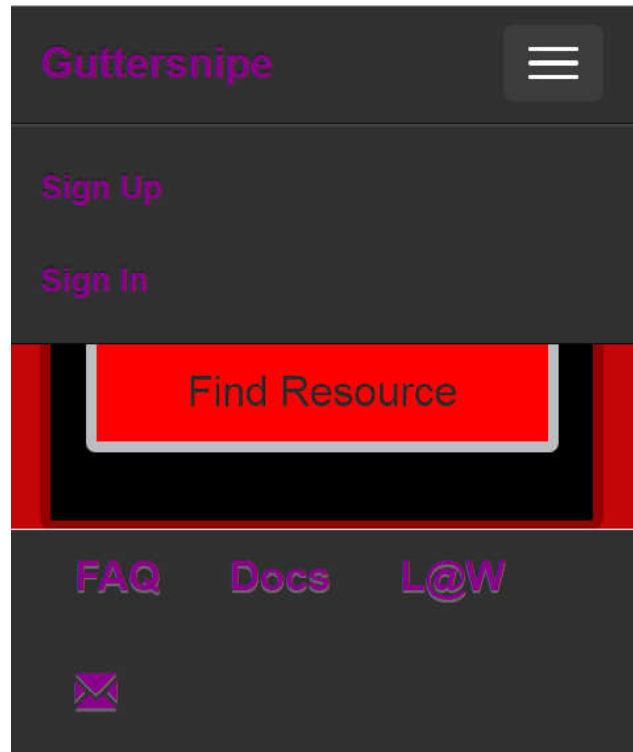
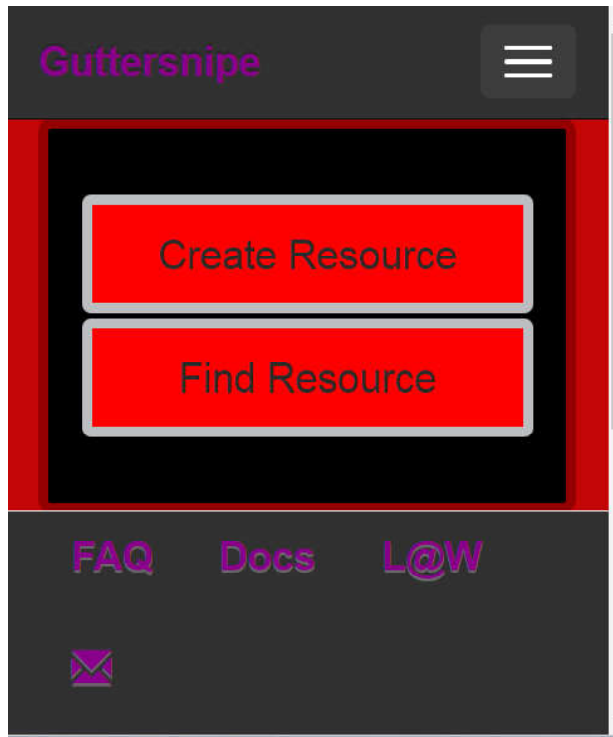


The Kropotkin widget displays quotes from “Conquest of Bread”  
The “New Quote” button produces another randomly-chosen quote.  
We intend to add another button that cycles through quotes every minute.

# **RESPONSIVE WEB DESIGN**

## 04.04.07

### Menu Dropdown



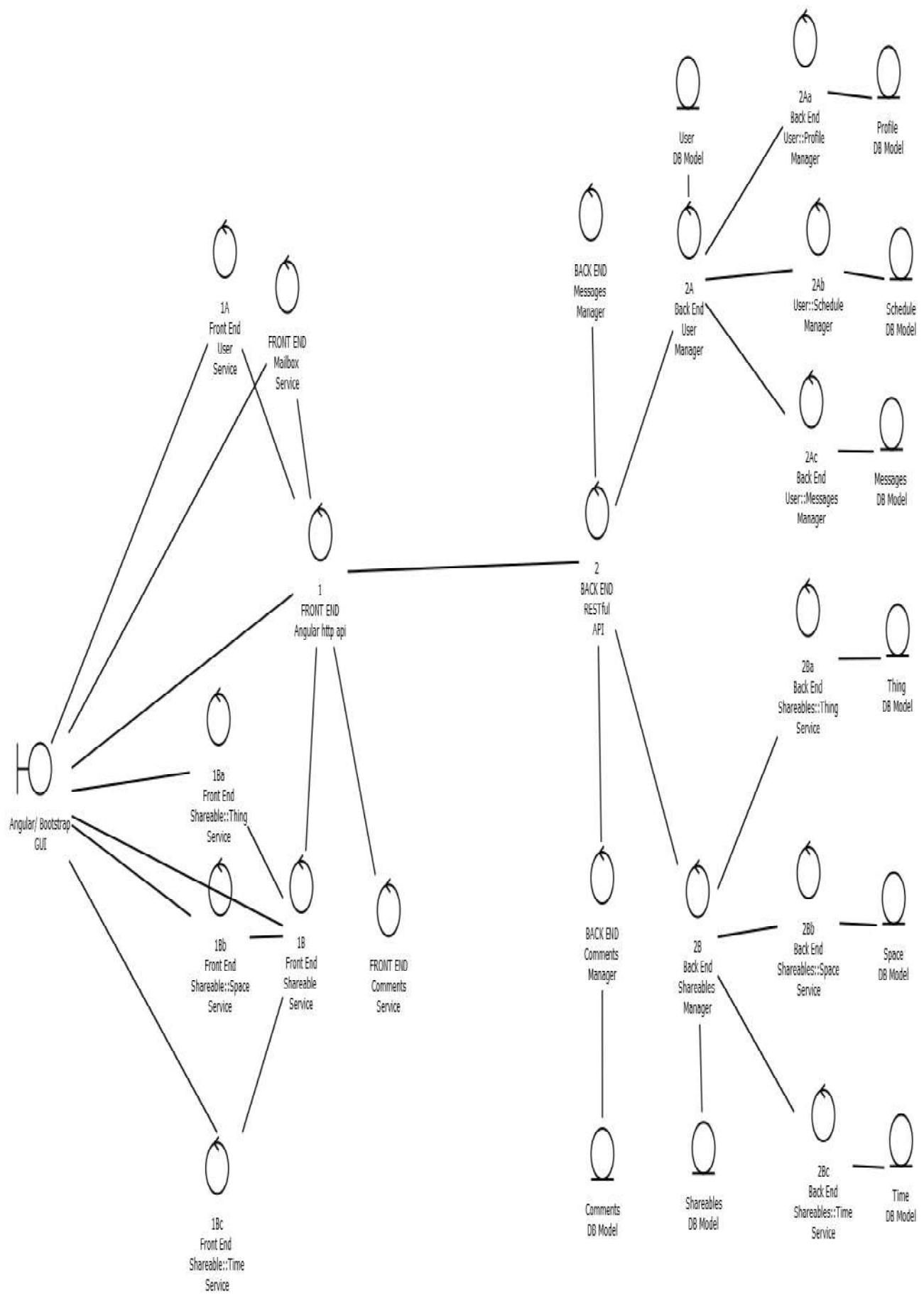
On smaller screens, the top toolbar is replaced with a button which displays a dropdown menu which reveals, when pressed, buttons for Sign In and Sign Up

# **Collaboration Diagrams**

# **Collaboration Diagram Stereotypes**

**Part 1: Control Objects**  
**Part 2: GUI Bound-**





# Control Objects

## 1. **FRONT END:** Angular HTTP API

An API Service wraps the native Angular \$http Service

### **a. F/E User Service**

A User Service maintains the User's account.

The User can login and logout through this

It can be used together with the Messages and the Comments Services

### **b. F/E Shareables Service**

A Shareables Service maintains the array of Shareables and functions for each Shareable.

Shareables Service maintains CRUD functionality for the list of Shareables and for each individual Shareable.

#### **i. F/E Shareables::Thing Service**

The Thing Service maintains Tags, Type/Subtype, and Description data

#### **ii. F/E Shareables::Space Service**

The Space Service maintains geolocation data.

#### **iii. F/E Shareables::Time Service**

The Time Service maintains schedule data

## **c. Comments Service**

The Comments Service maintains the comments that Guttersnipes put on Shareables.

CRUD functionality is available for the array and each member.

### **i. Messages Service**

A Shareables Service maintains the array of Messages that Guttersnipes send to each other as well as each individual Message.

CRUD functionality is available for the array and each member

## **2. Back END: RESTful API**

API written with Flask/SQLAlchemy

### **a. B/E User Model**

A User Model maintains the User's account.

The User can login and logout through this

It can be used together with the Messages and the Comments Models

### **b. B/E Shareables Model**

A Shareables Model maintains the array of Shareables and functions for each Shareable.

Shareables Model maintains CRUD functionality for the list of Shareables and for each individual Shareable.

#### **i. B/E Shareables::Thing Model**

The Thing Model maintains Tags, Type/Subtype, and Description data

#### **ii. B/E Shareables::Space Model**

The Space Model maintains geolocation data.

#### **iii. B/E Shareables::Time Model**

The Time Model maintains schedule data

### **c. Comments Model**

The Comments Model maintains the comments that Guttersnipes put on Shareables.

CRUD functionality is available for the array and each member.

### **d. Messages Model**

A Shareables Model maintains the array of Messages that Guttersnipes send to each other as well as each individual Message.

CRUD functionality is available for the array and each member

# DB Model Objects

## DB Models (SQLAlchemy/PostGIS)

API written with Flask/SQLAlchemy

Database uses postgresql with PostGIS

### a. B/E User DB Model

Stores

- i.id
- ii. profile
- iii. schedule
- iv. is\_admin
- v. created\_on

### b. B/E Profile DB Model

Stores

- i.id
- ii. username
- iii. email
- iv. full\_name
- v. password
- vi. additional\_info

### c. B/E Schedule DB Model

Stores

- i.id
- ii. calendar = sCalendar\_VEVENT
- iii. notes

### d. Messages DB Model

Stores

- i. calendar = sCalendar\_VEVENT
- ii. text
- iii. sender
- iv. recipient
- v. sent = db.Column(db.DateTime

## **e. Block User Join DB Table**

Stores

- i. blocker
- ii. blocked

## **f. B/E Shareables DB Model**

- i. id
- ii. thing\_id
- iii. space\_id
- iv. time\_id
- v. comments
- vi. number\_ratings
- vii. total\_ratings

## **g. B/E Shareables::Thing DB Model**

The Thing DB Model maintains Tags

- i. subtypes = String [] *# Not a good choice.*  
*Rethink this*
- ii. descriptionHow
- iii. descriptionWhat

## **h. B/E Tag DB Model**

- i. id
- ii. tag

## **i. Tag Thing DB Join**

- i. tag\_id
- ii. shareable\_id

## **j. B/E Shareables::Space DB Model**

- i. The Space DB Model maintains geolocation data.
- ii. id
- iii. longitude
- iv. latitude
- v. canonical\_address
- vi. alternate\_names
- vii. notes

## **k. B/E Shareables::Time DB Model**

- i. The Time DB Model maintains schedule data
- ii. id
- iii. calendar = sCalendar\_VEVENT  
*#sCalendar\_VEVENT will be defined soon ...*
- iv. notes

## **I. Comments DB Model**

The Comments DB Model maintains the comments that Guttersnipes put on Shareables.

CRUD functionality is available for the array and each member.

- i. id
- ii. author
- iii. shareable
- iv. text
- v. created

# Boundary Objects

## 1. Angular GUI (Angular/Ionic/Bootstrap)

GUIs written with Angular/Ionic JS Frameworks.  
CSS with Bootstrap

### a. B/E User Entity

A User Entity maintains the User's account.

The User can login and logout through this

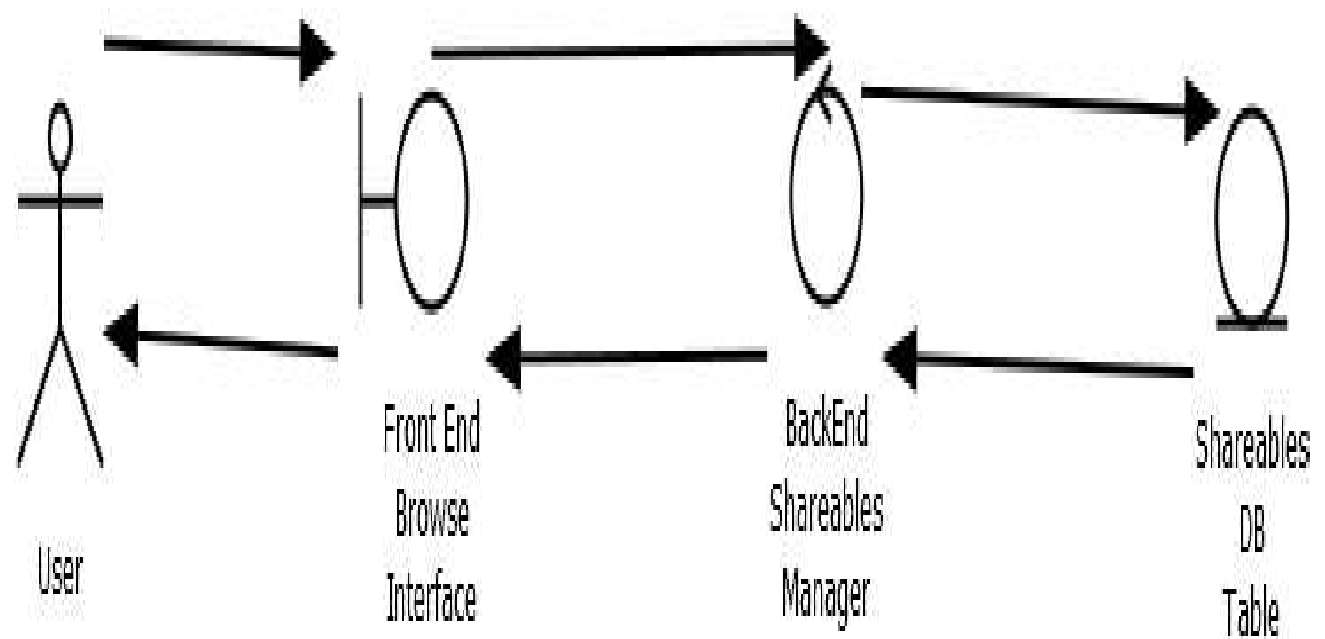
It can be used together with the Messages and the Comments  
Entitys



03.01.04.xx

View Shareable

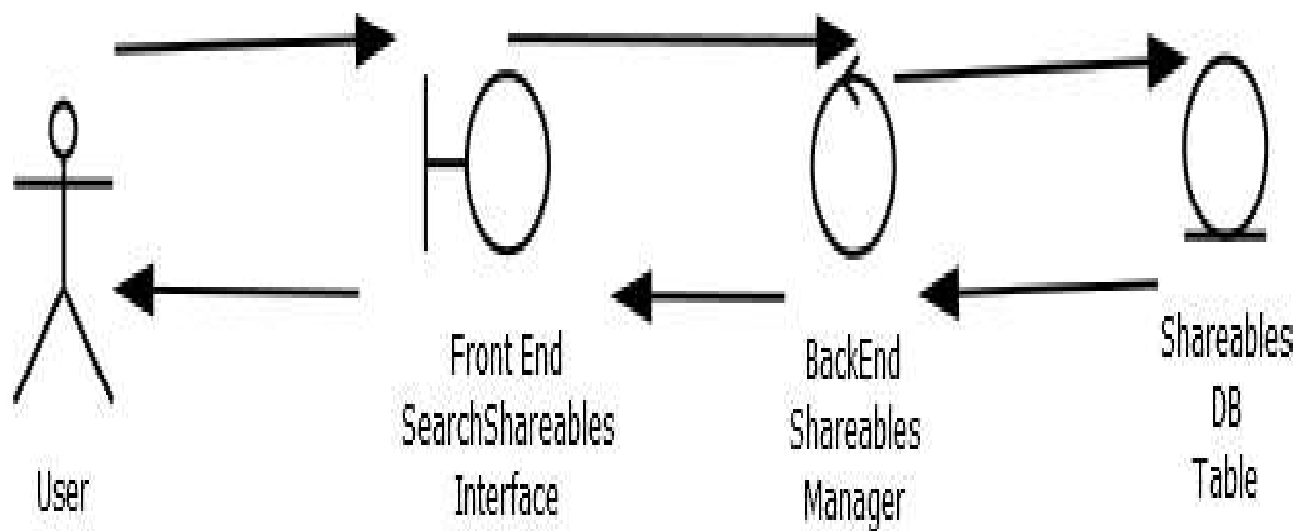
Collaboration Diagram



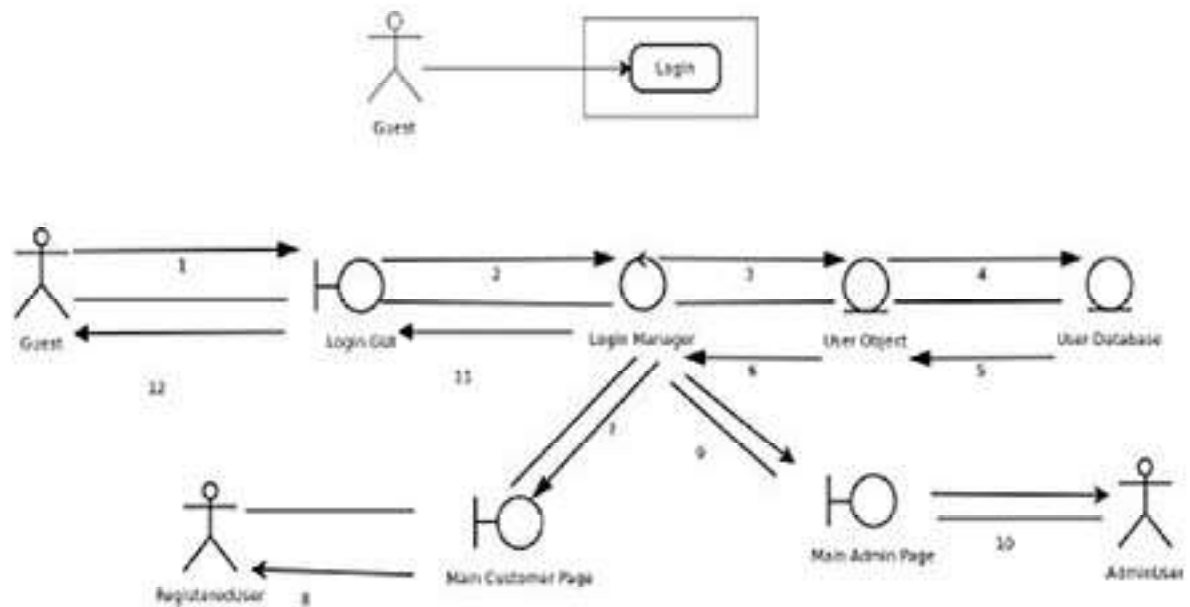
03.01.04.XX

Search Shareables

Collaboration Diagram



### 3. 2. 2. ACCOUNT CONTROL: EXTERNAL



1. Guest User fills in account information on the login page
2. User info passed to login manager
3. Pass login info to user object
4. Login object queries user database with login info
5. User database passes back query results to user object.
6. Login manager receives results
7. Login manager shows main customer page
8. System recognizes end user as Customer
9. Login manager shows main admin page
10. System recognizes end user as Admin

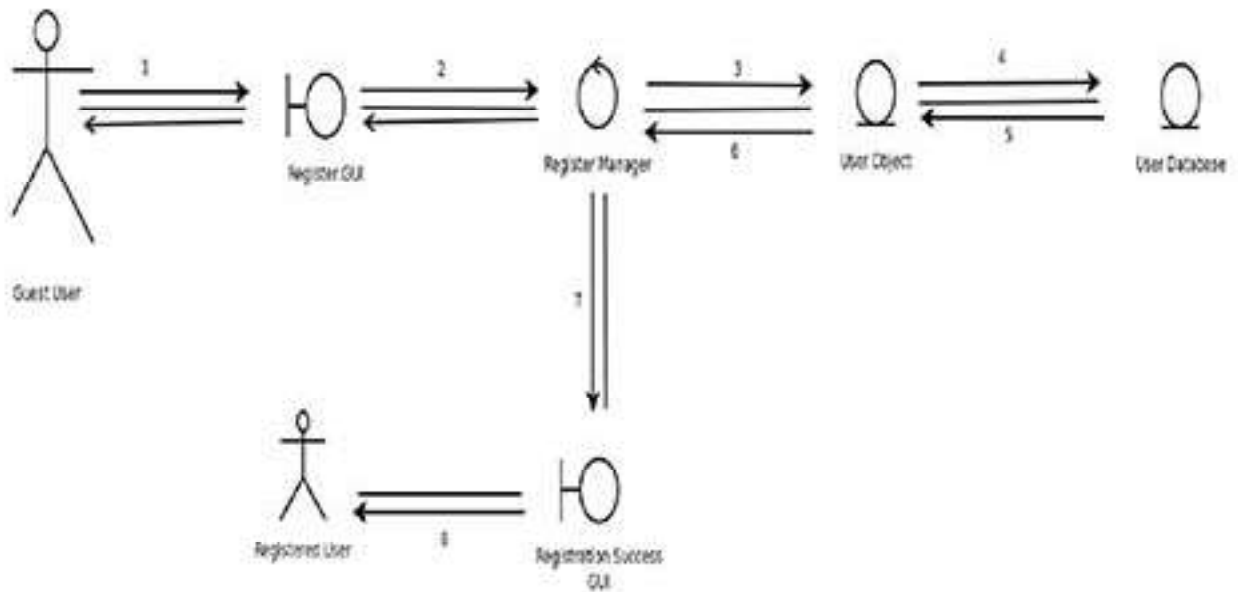
#### Exception 1

5. DB error sent to Login Object
6. Failure sent to Login Manager
11. Login manager presents Failure message on GUI
12. Guest receives error message

#### Exception 2

7. First time login user present with choose interest page

**Figure B2: Login**



1. Guest User clicks on "Register GUI" to be registered as a user.
2. The information is then passed to the register object.
3. This information is passed to the user object and generates password.
4. The information is stored in the user database.
5. User database send the information to user object.
6. The correct information passed from the user object is then passed to register object.
7. This opens a new GUI and welcomes the new user.
8. This turns a Guest User to be a Registered User.

#### Exceptional Case:

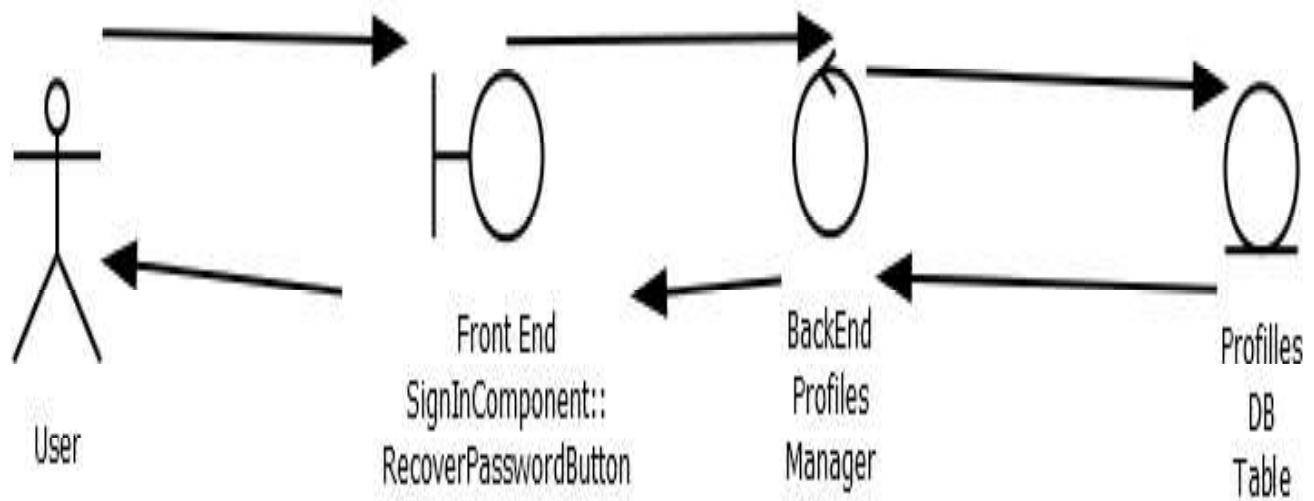
1. The visitor enter the information and clicks "Submit".
2. The information is passed to Register Object.
3. The processed information is wrong so is returned to Register GUI.
- 3a. Guest User is specified the information provided is wrong.

## B1: Register

03.01.04.XX

Recover Password

Collaboration Diagram

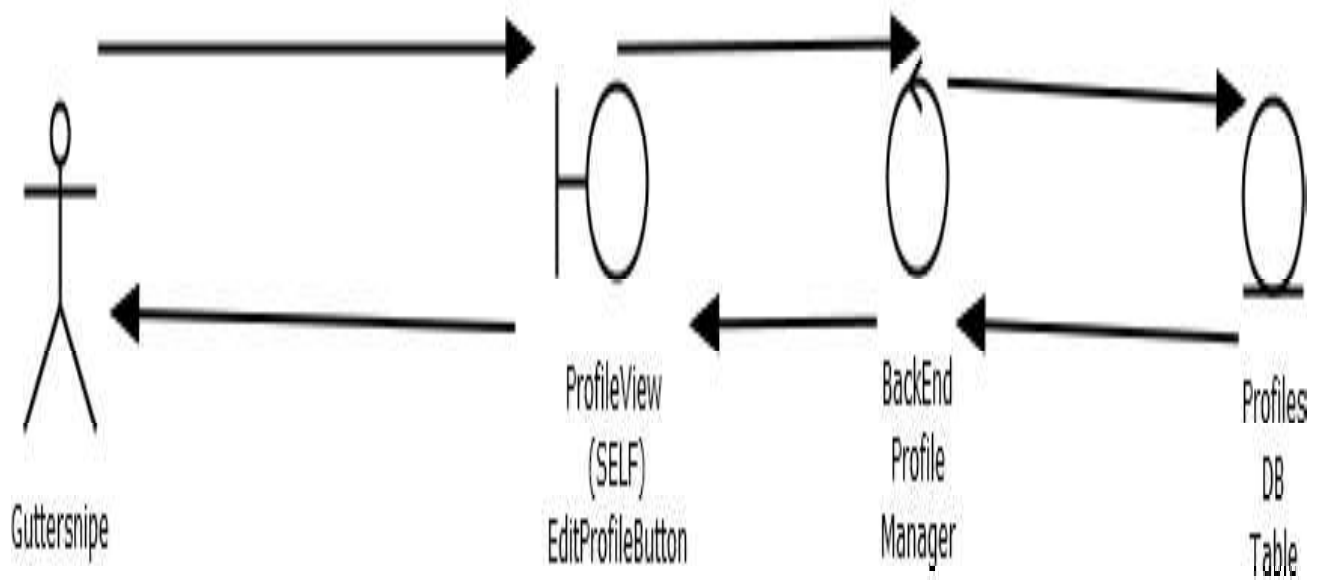


### 3. 2. 3. ACCOUNT CONTROL: INTERNAL

03.01.04.xx

Edit Profile

Collaboration Diagram

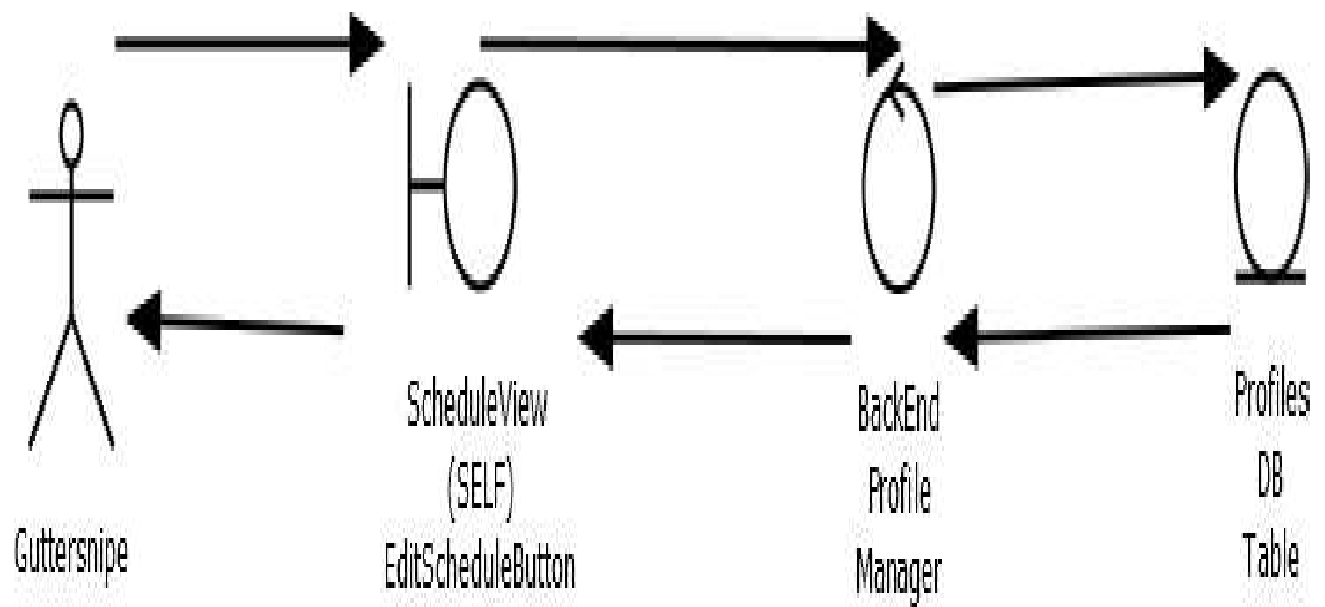




03.01.04.xx

EditSchedule

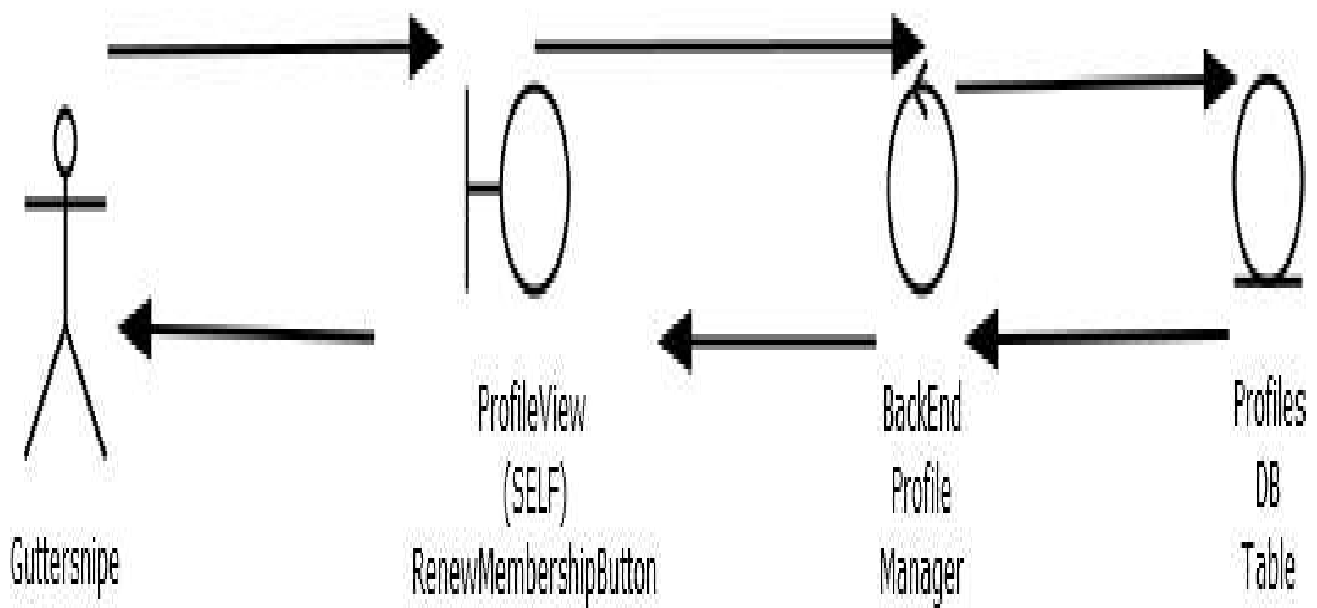
Collaboration Diagram



03.01.04.01

Delete Shareable

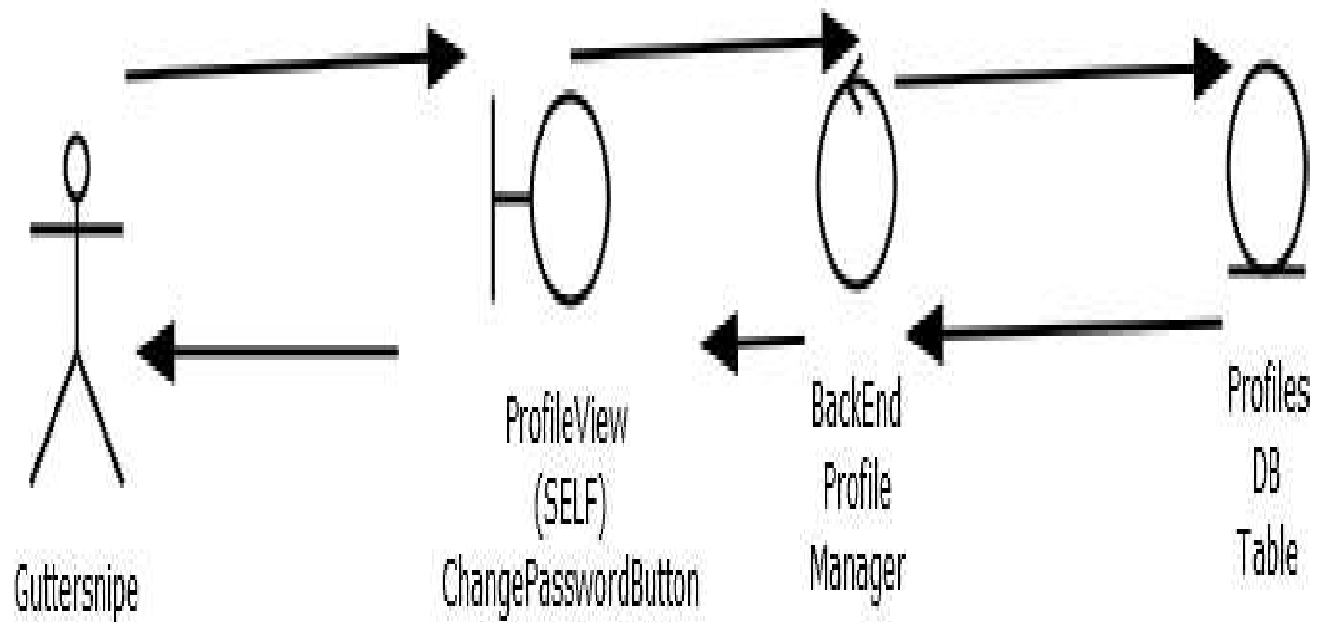
Collaboration Diagram

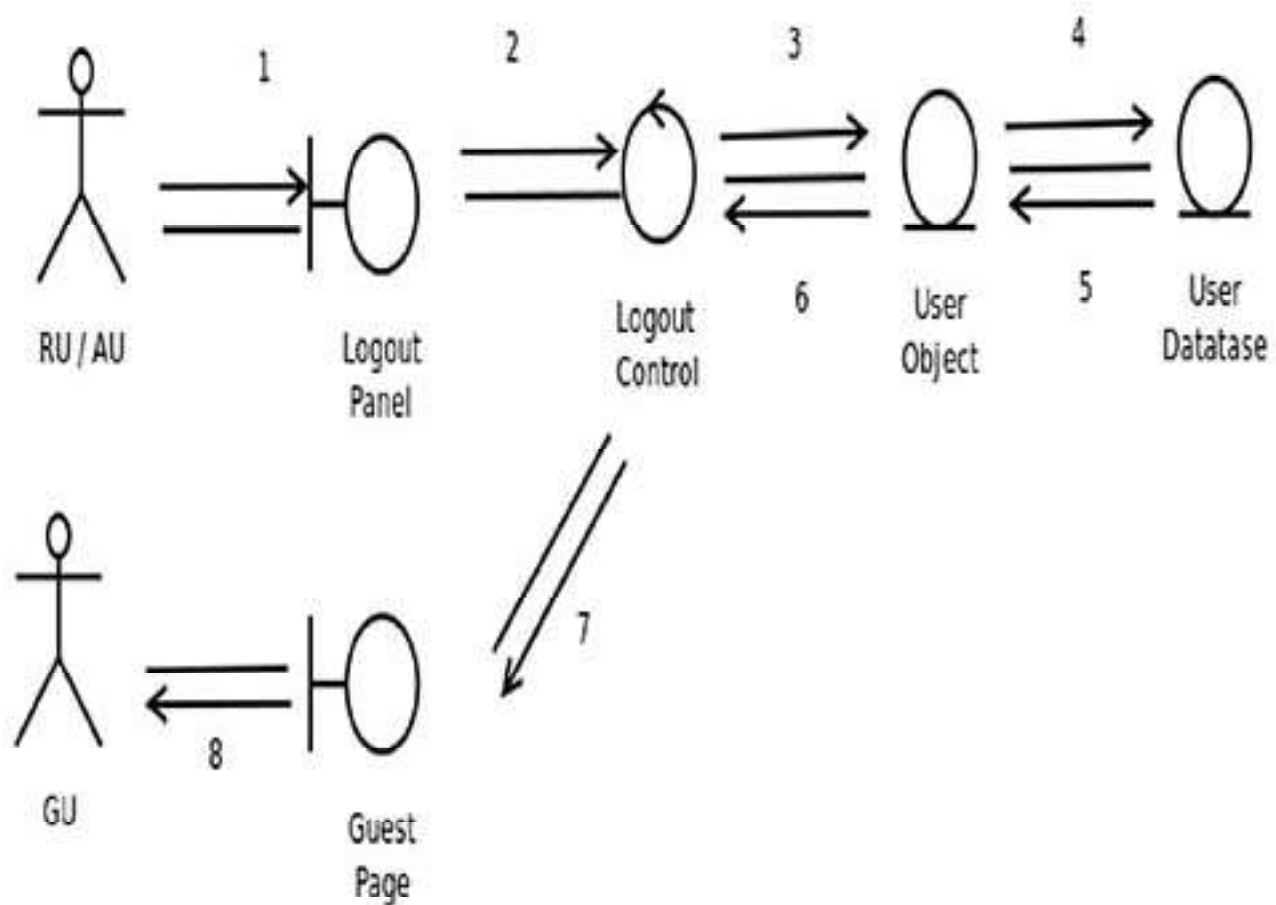


03.01.04.XX

## Change Password

### Collaboration Diagram





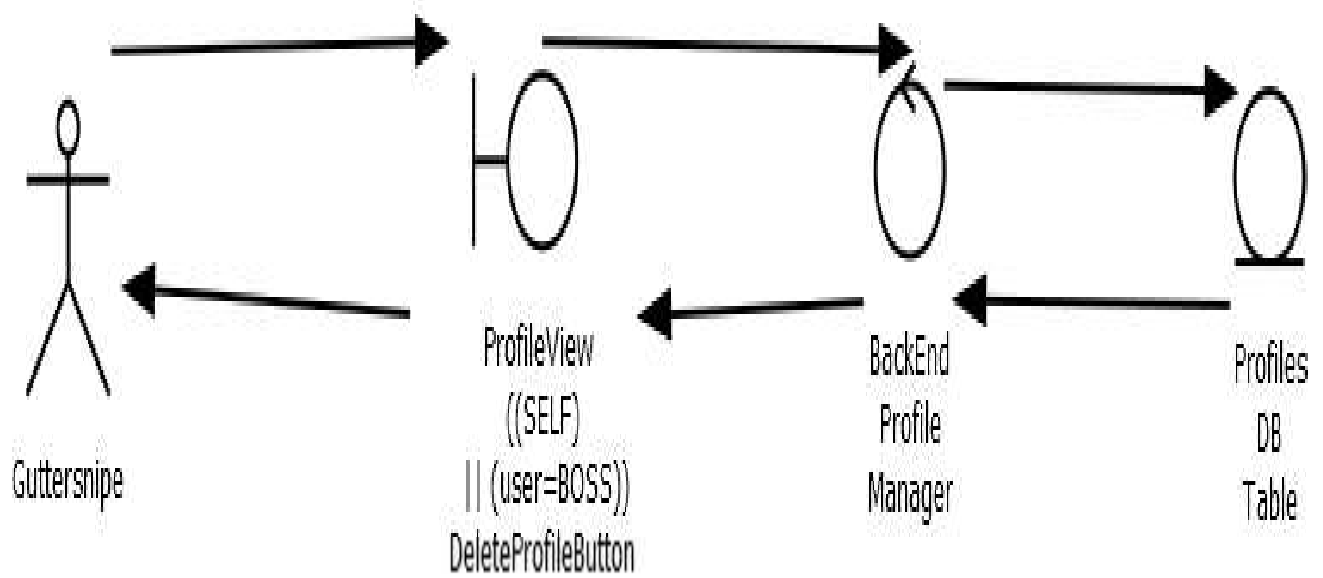
1. Registered User/Admin User clicks on the Logout button
2. Logout Control processes request, verifies user's desire to logout
3. Controller passes logout to user database, ends user's session, completes any pending transactions
4. Database returns results to Logout Control
5. Logout Control produces Guest page stating "You have now been logged out"
6. End User is now considered to be a Guest

**Figure C8: Logout**

03.01.04.xx

Delete Profile

Collaboration Diagram

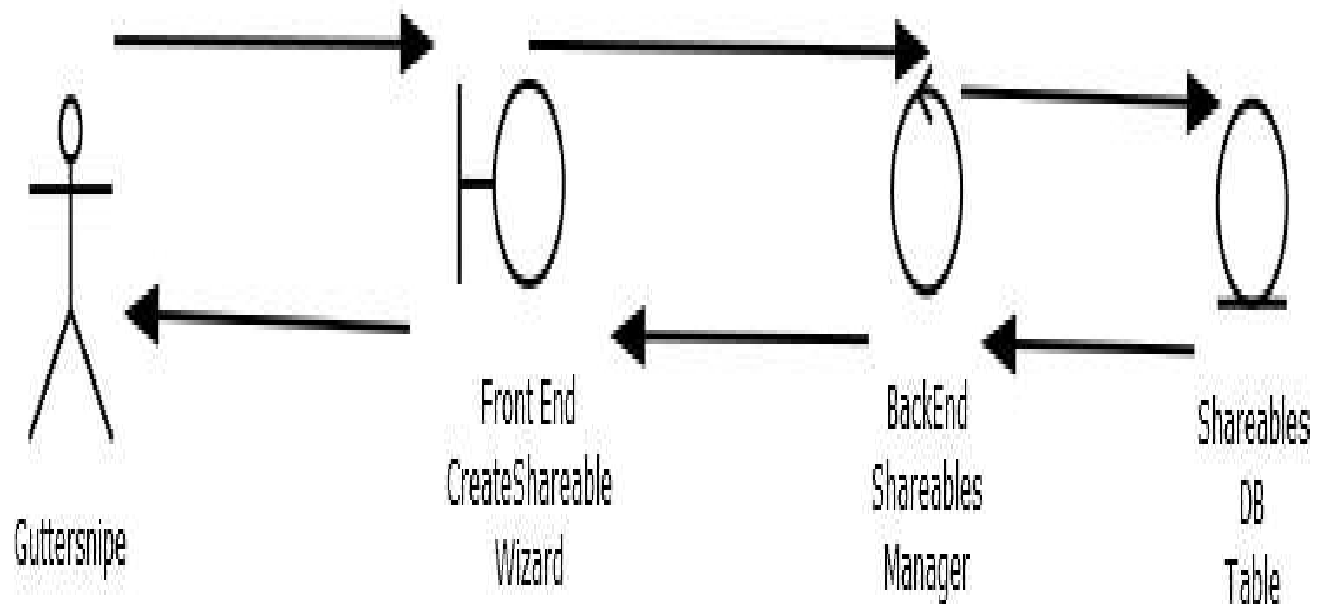


### 3. 2. 4. SHAREABLE: CREATE, UPDATE, DELETE

03.01.04.01

Delete Shareable

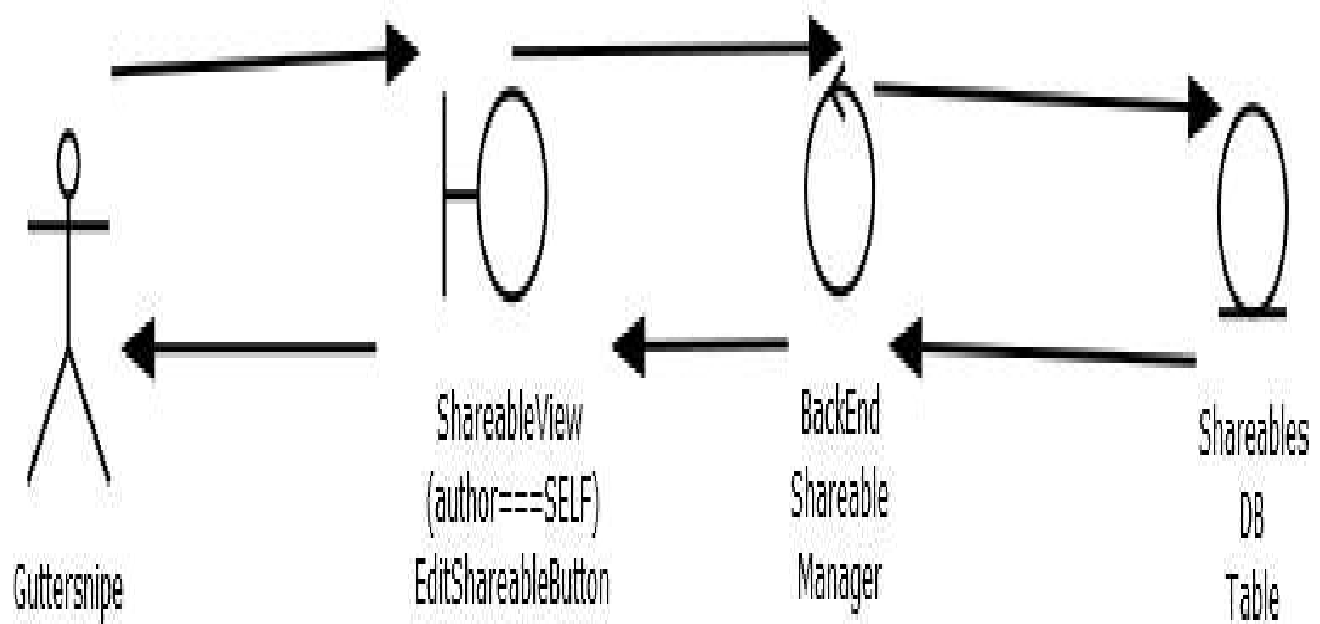
Collaboration Diagram



03.01.04.xx

Edit Shareable

Collaboration Diagram

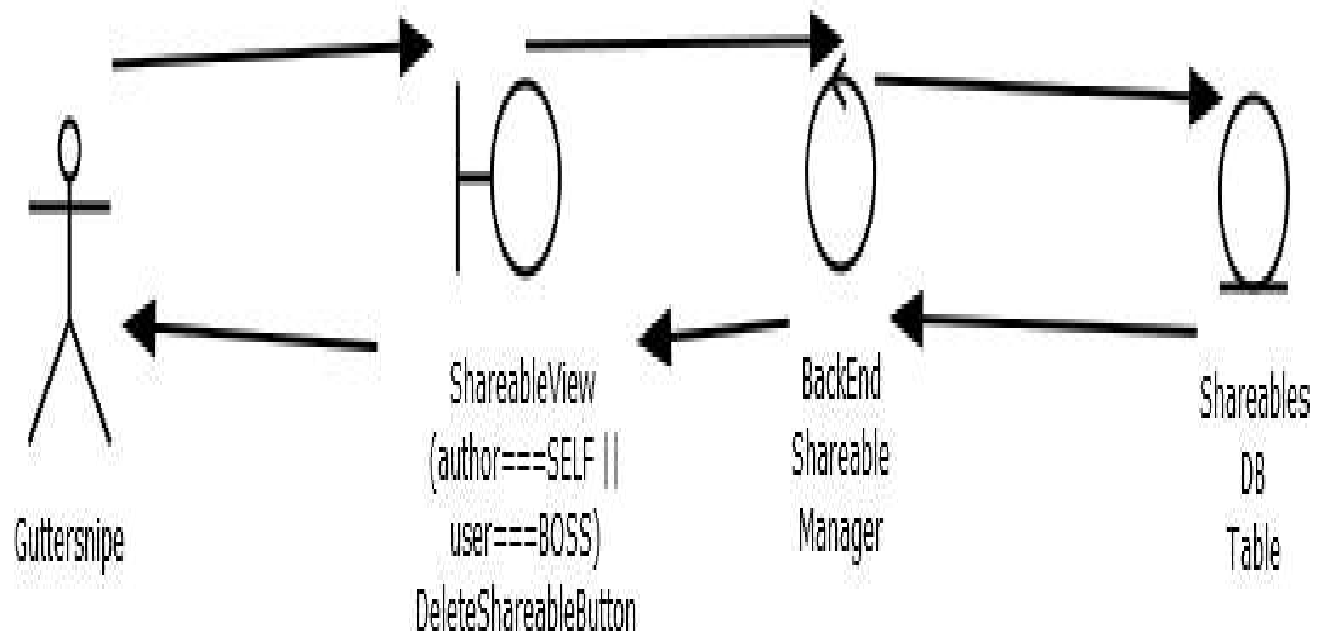




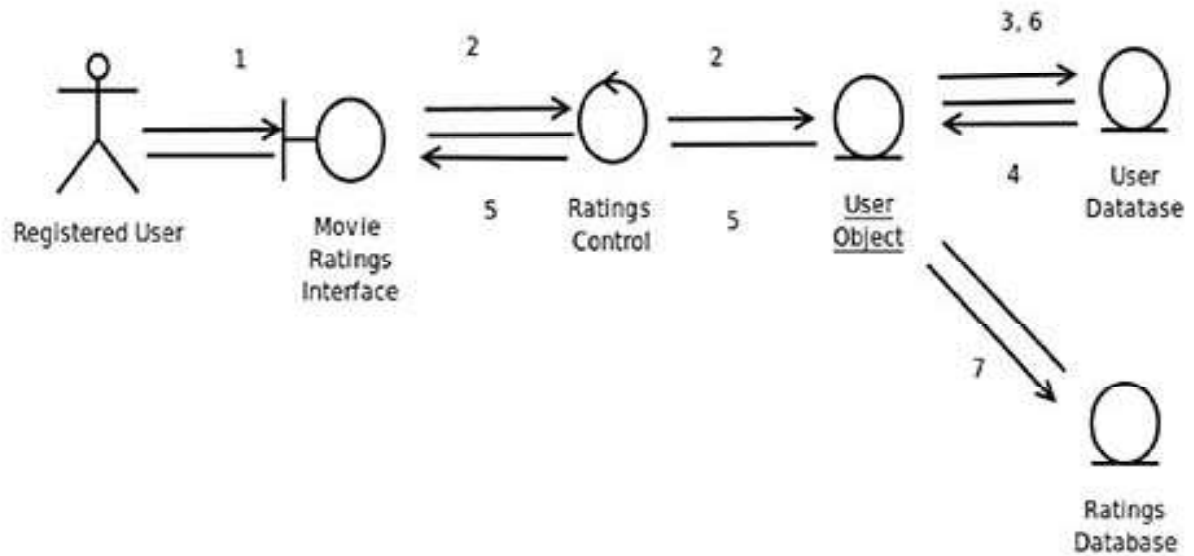
03.01.04.xx

Delete Shareable

Collaboration Diagram



3. 2. 5. SHAREABLE: ANNOTATE



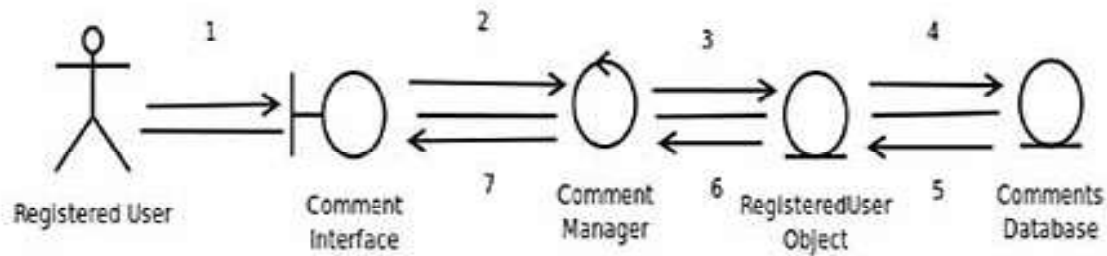
Figure

1. Registered User clicks on movie rating interface
2. Rating Interface submits UserID, MovieID, and Rating to Rating Control
3. Rating Control checks User DB to check User's ratings permissions
4. User DB returns RU's permissions
5. Ratings Control sends success or failure message back to Ratings GUI
- 6 Ratings Control updates Registered User Database to record Rating Behavior
7. Ratings Control updates Movie Database with new rating

Exceptions:

1. RU presses submit button without selecting a rating
2. RU does not have permissions to rate movie

**Figure C2: Rate**



1. RU visits the comment interface of a movie page
2. RU submits comment to be processed by the comment manager
3. The Comment Manager accesses RegisterUser object to processes the comment with a timestamp
4. The data is inserted to Comments Database
5. The comment information is sent back to the RegisteredUser object
6. The comment information is passed back to the comment manager
7. The Comment Manager updates the Comment interface with new comment(s)

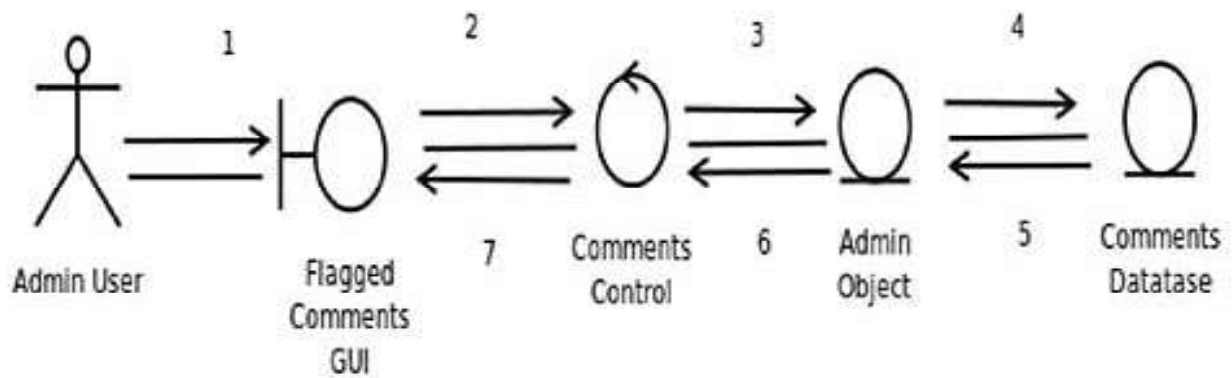
Exception 1:

2. Blank comment is submitted
7. The comment manger displays an error message to the GUI

Exception 2:

2. A comment that exceeds the character limit is submitted
7. The comment manager displays an error message to the GUI

**Figure C3: Comment**



1. Admin User selects comments for deletion from Flagged Comments Panel
2. GUI sends deletion request to Comments Control
3. Comments to be deleted are sent to Database
4. Comments are sent to comments database
5. Database returns remaining Flagged Comments from Database
6. The remaining flagged comments are sent to comments control
7. The remaining Flagged Comments are displayed on the GUI

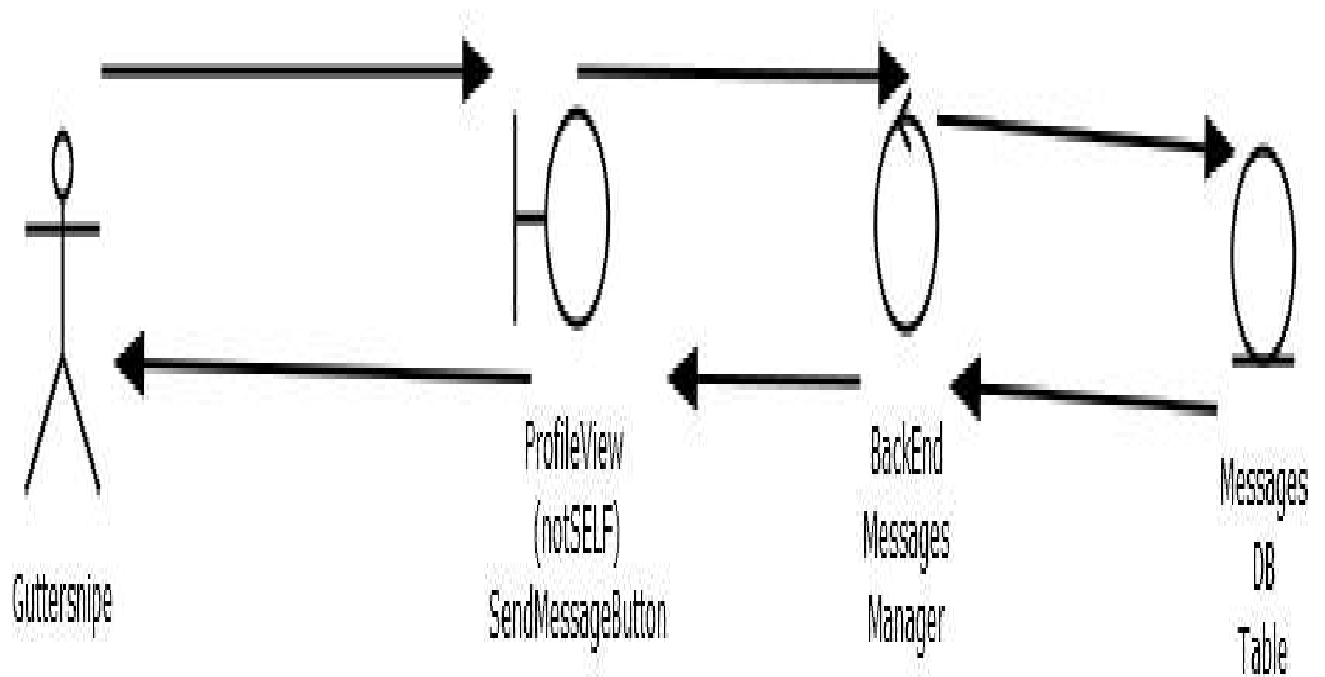
## **D1: Erase Comment**

### 3.2.6. COMMUNICATIONS

03.01.06.xx

Send Message

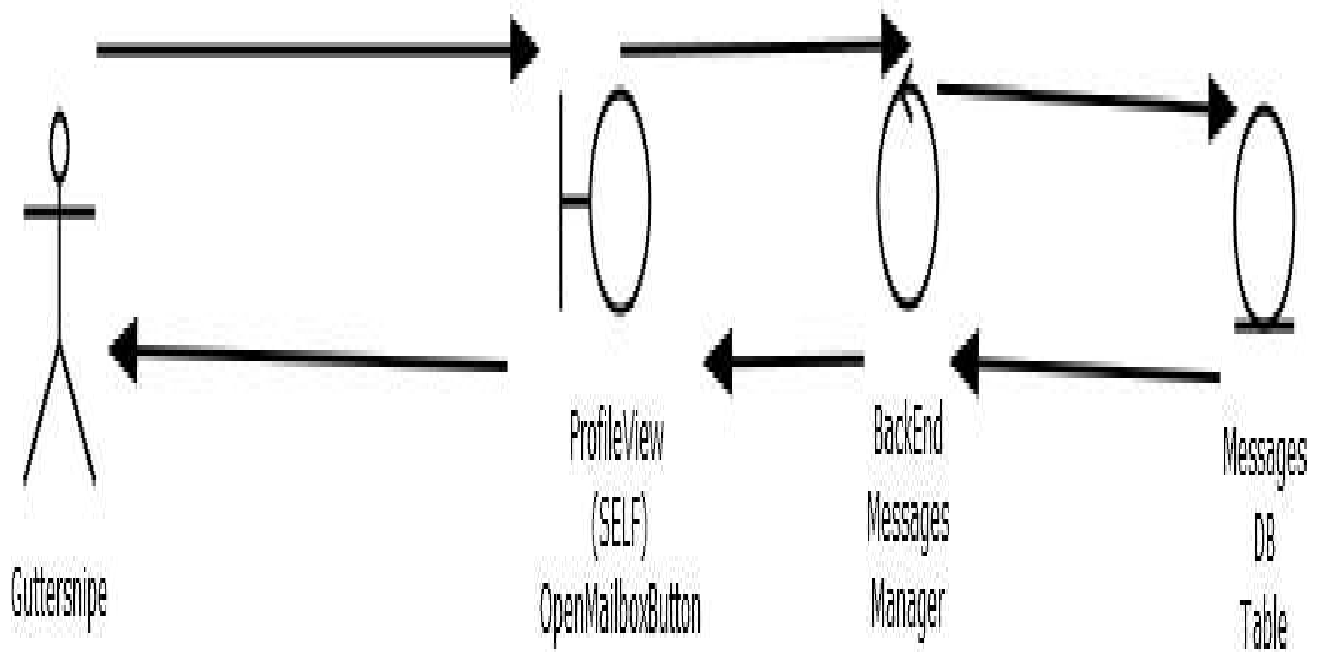
Collaboration Diagram



03.01.04.xx

OpenMailbox

Collaboration Diagram

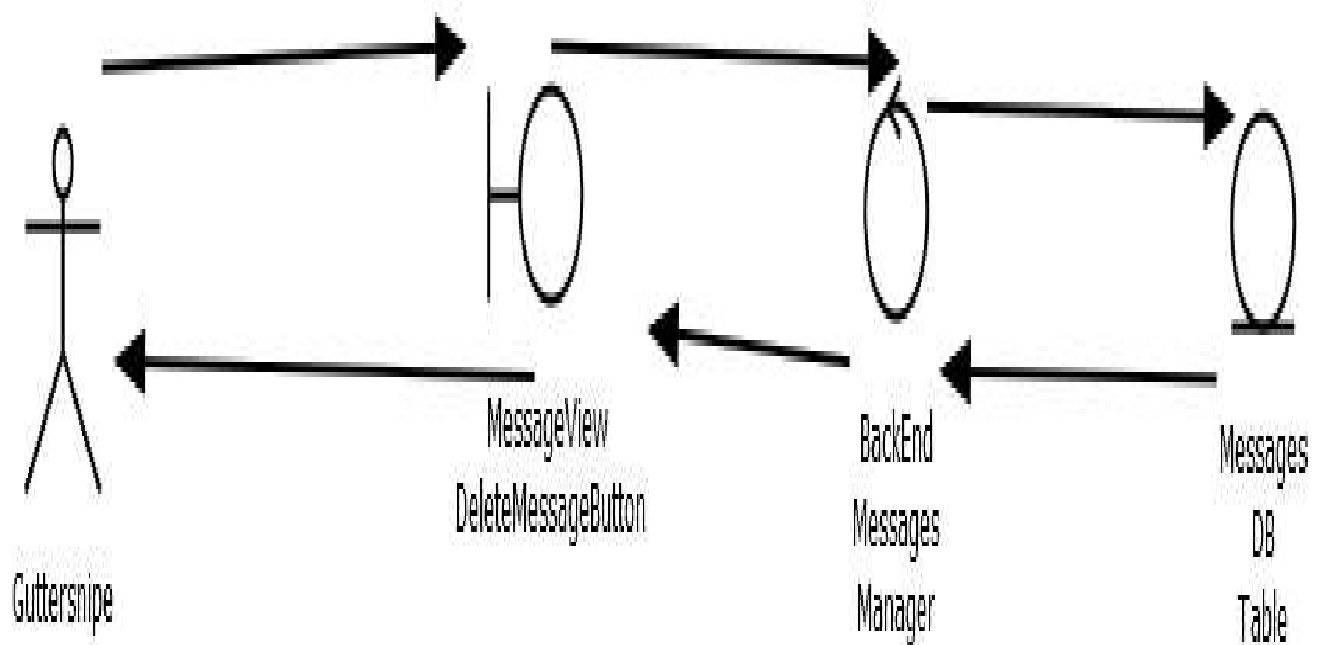




03.01.04.xx

## Delete Message

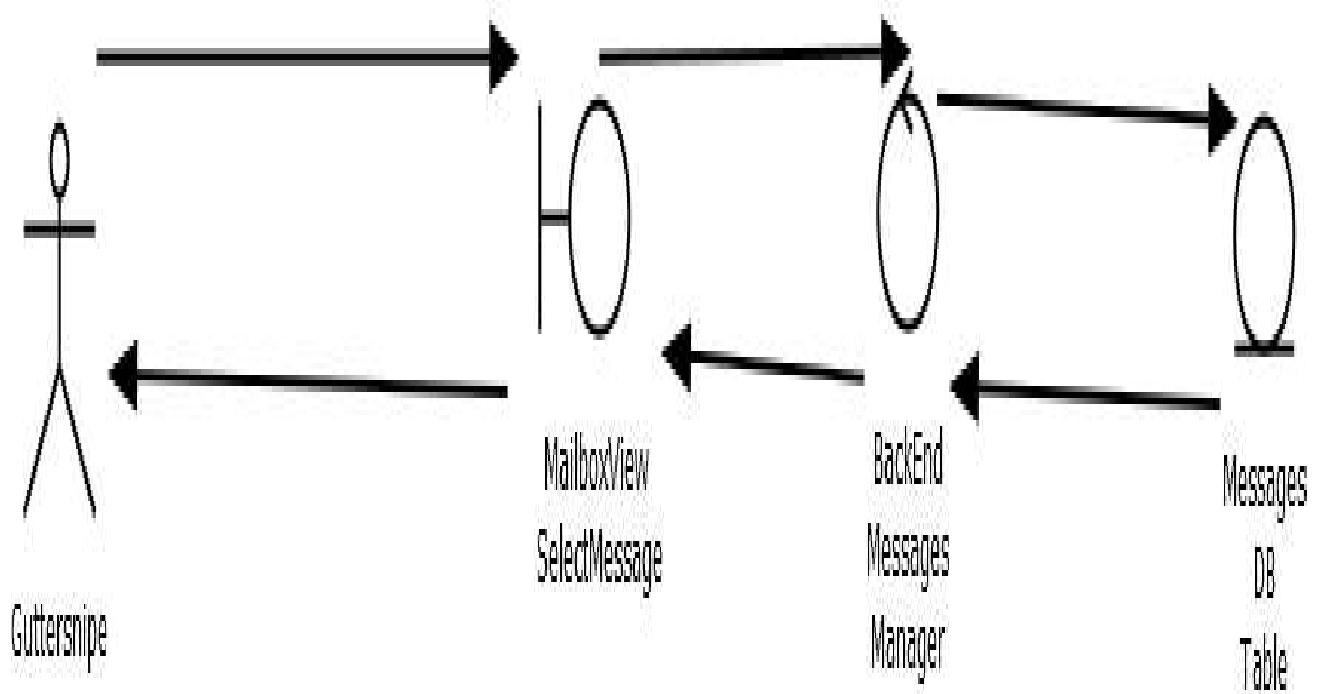
### Collaboration Diagram



03.01.04.XX

Read Message

Collaboration Diagram



03.01.04.xx

Block User

Collaboration Diagram

