

Guttershipe (0.3)

***Software Requirement
Specifications***

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. Boss 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
 - 4.5.3.4. Consultations
 - 4.5.3.5. Other Sites
 - 4.5.3.6. Q: Method
 - 4.5.3.7. A: Method
 - 4.5.3.8. Example
 - 4.5.3.9. REprot
 - 4.5.3.10. Result
 - 4.5.3.11. Research
 - 4.5.3.12. Fin
 - 4.5.4. Administrative
 - 4.5.4.1. Legal
 - 4.5.4.2. About
 - 4.5.4.3. Contact
 - 4.5.5. Kropotkins
 - 4.5.5.1. Kropotkin Quotes
 - 4.5.6. Screensize
 - 4.5.6.1. Dropdown top menu (1)
 - 4.5.6.2. Droptown to menu (2)

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 three 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

**** Guttersnipe, registered users, 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, optional contact info.

All guttersnipes 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.

**** Bosses**

Bosses are Guttersnipes with administrative capacities.

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

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.

1. Technical Introduction

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 baseclass 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 baseclass User.

In addition, the Guttersnipe can exercise control over its own account. The Guttersnipe may edit its own profile, 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 Boss.

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

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

The Boss 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.

1.3. Acronyms and Abbreviations

1.3.1. V: Vagrant

1.3.2. G: Guttersnipe [User]

1.3.3. B: Boss

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.

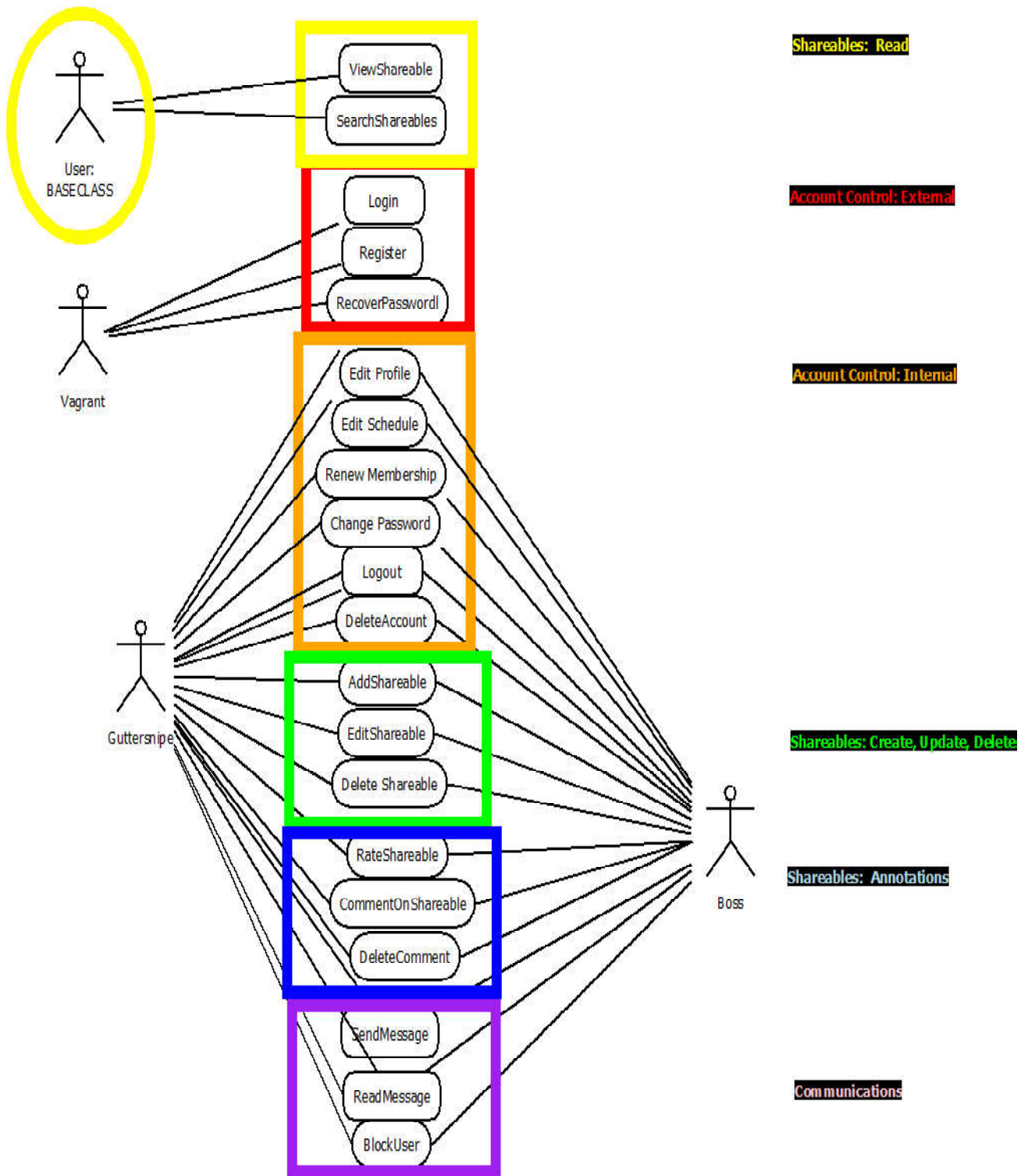
Part 1.2: Technical Introduction

Part 1.3: Acronyms and Abbreviations

Part 1.4: Summary

2. Overview

2.1 User Case Diagram and Descriptions



02.01.01. Usecases: USERS

A.

B.

i. User (U)

a. 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)*

ii. Vagrant (V)

a. 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.

i. Guttersnipe (G)

a. 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..

f) DeleteAccount: Can delete own account.

b. Shareables (Create, Update, Delete)

a) CreateShareable: Can add a Shareable resource.

b) UpdateShareable: Can update Shareable created by self.

c) DeleteShareable: Can delete Shareable created by self.

c. Shareables (Annotate)

a) RateShareable: Can rate any shareable

b) CommentOnShareable: Can comment on any shareable.

c) DeleteComment Can delete own comment.

d. 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 Boss.

ii. Boss (B)

Boss has same capabilities as Guttersnipe, but they are

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.

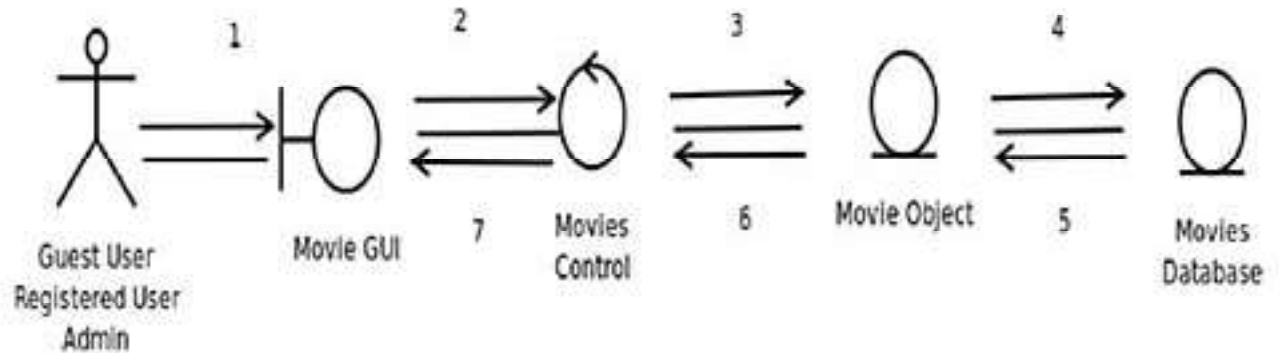
For backend technologies, we sought a technology that could easily be ported to a variety of servers where no superuser access is required. We dislike bulky frameworks, preferring to add components as we need them. To this end, we have chosen the Flask / Python framework.

SPECIFIC REQUIREMENTS

Collaboration Diagrams

3. 2. 1. SHAREABLE: READ

3.2.1.1. ViewShareable



1. GU/RU/AU clicks on the Movies GUI
2. Request is sent to the Movies control module for movie=movieID
3. A movie object is generated
4. Request is sent to the Movies database containing movieID
5. MovieObject is initialized with
(Name, Year, Genre, Director, Stars, RunningTime, Summary)
4. Information about the Movie is returned to Control
5. Abstract of Movie
(Name, Year, Genre, Director, Stars, RunningTime, Summary)
displayed on GUI

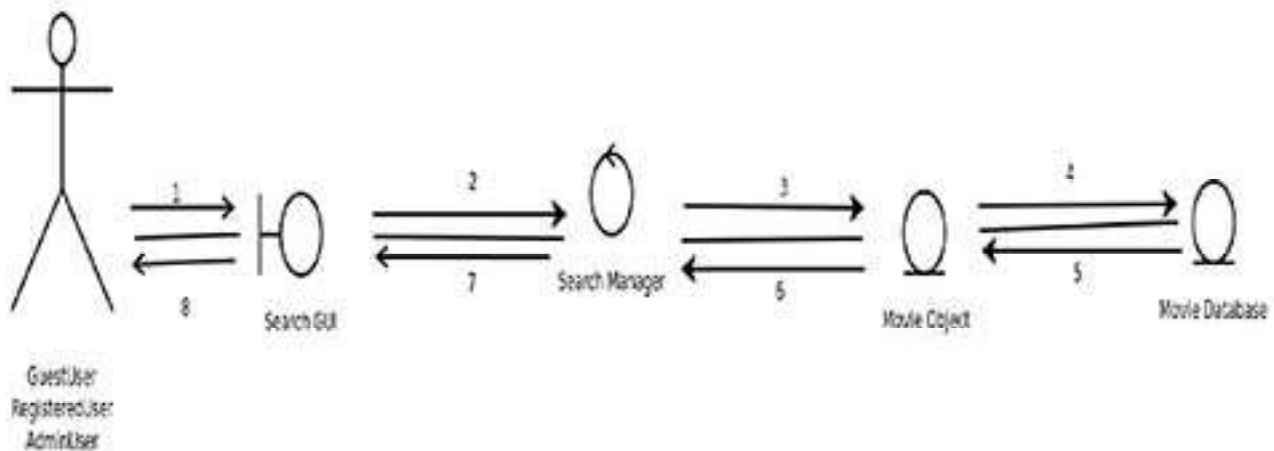
Figure A3: View Movie Info

3.2.1.2. SearchShareables

3. Specific Requirements

3.1 Collaboration Diagrams

3.1.1 Shared Collaboration Diagrams



1. GuestUser, RegisteredUser, or AdminUser click on Search GUI.
2. Search GUI transfers the information to search manager
- 3 Search manager passes the information to movie object.
4. Movie object verifies the search by looking into the database.
- 5 Movie Database passes the verified information to user object.
6. User object processes the information to search manager.
7. The information is published in Search GUI.
8. GuestUser/RegisteredUser/Visitor gets the result for searched information.

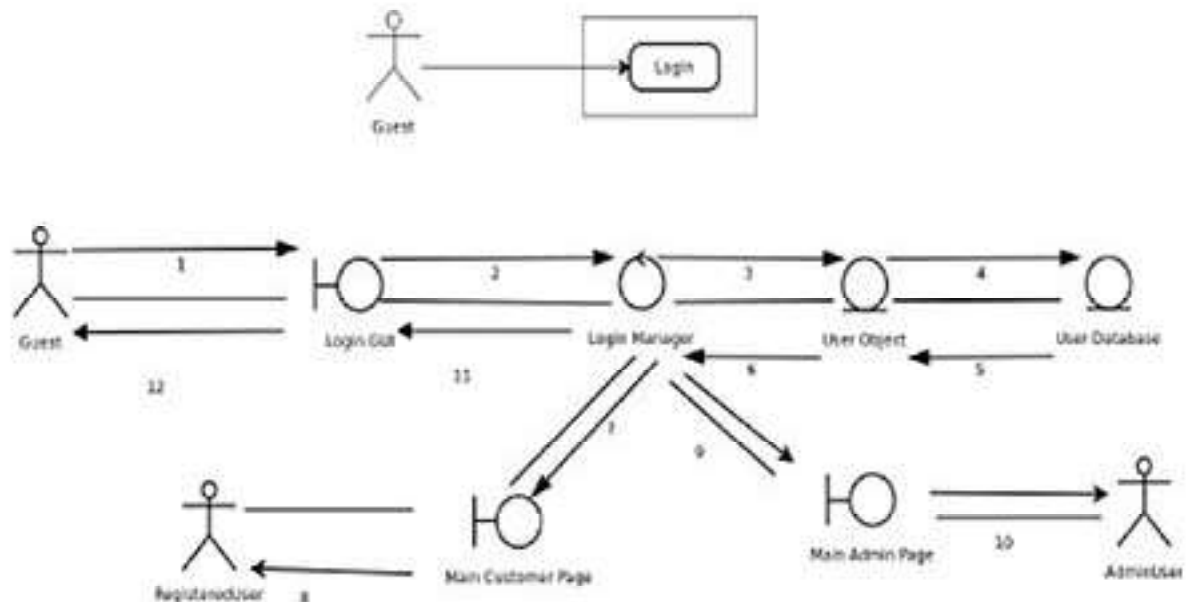
Exception

5. Search result not found
6. Empty database information sent to search manager
8. GUI displays "No Results Found"

Figure A1: Search

3.2.2. ACCOUNT CONTROL: EXTERNAL

3.2.2.1. Login



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

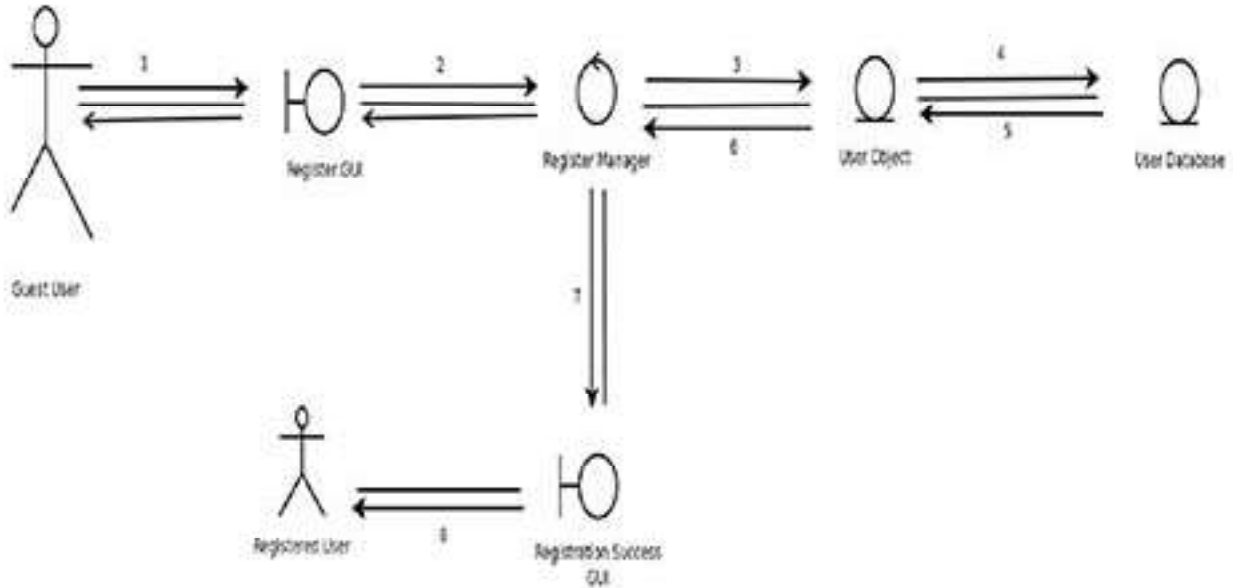
5. DB error sent to Login Object
6. Failure sent to Login Manager
11. Login manager presents Failure message on GUI
12. Guest receives error message

Exception 2

7. First time login user present with choose interest page

Figure B2: Login

3.2.2.2. Register



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 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

3. 2. 3. ACCOUNT CONTROL: INTERNAL

3.2.3.1. EditProfile

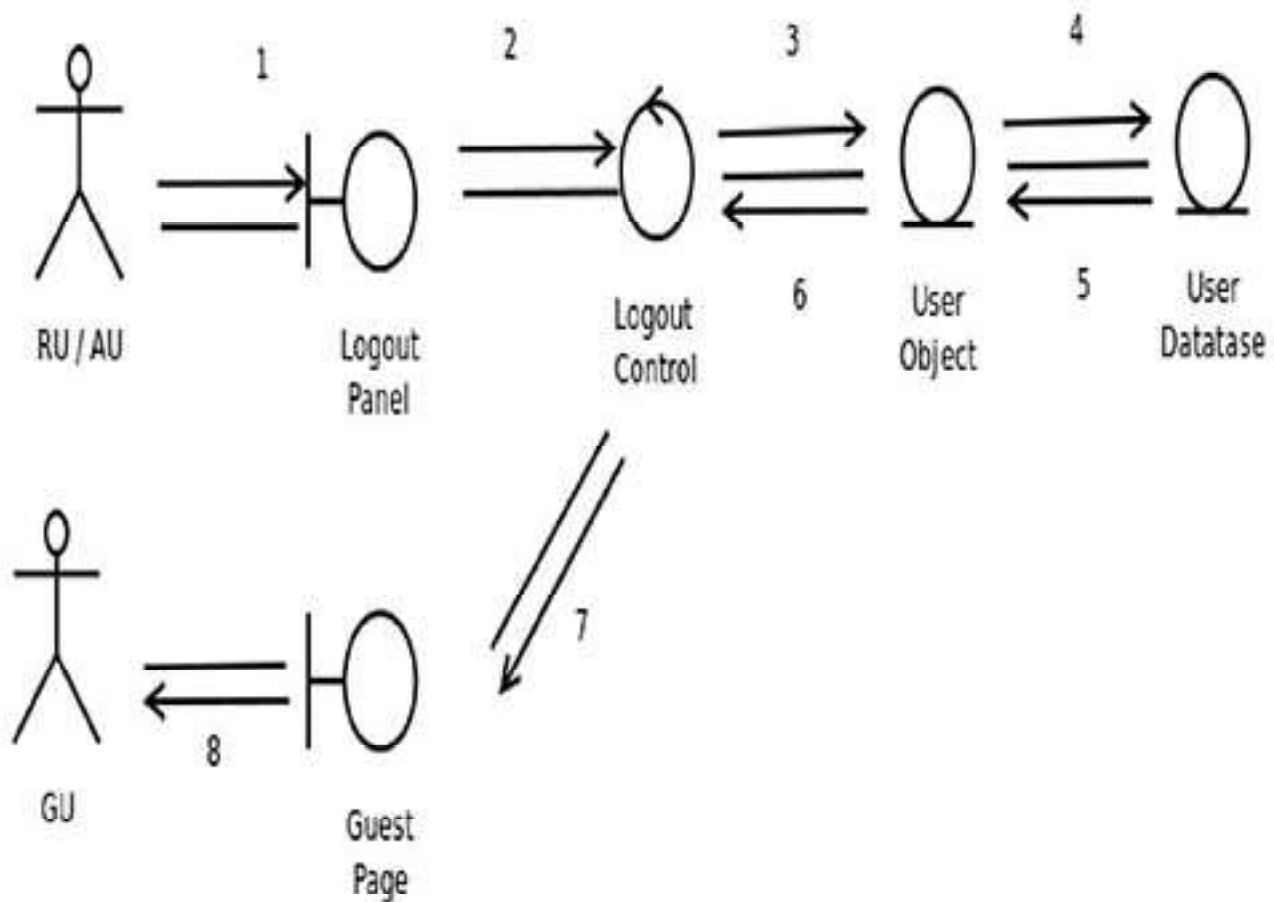
3.2.2.3. RecoverPassword

3.2.3.2. EditSchedule

3.2.3.3. RenewMembership

3.2.3.4. ChangePassword

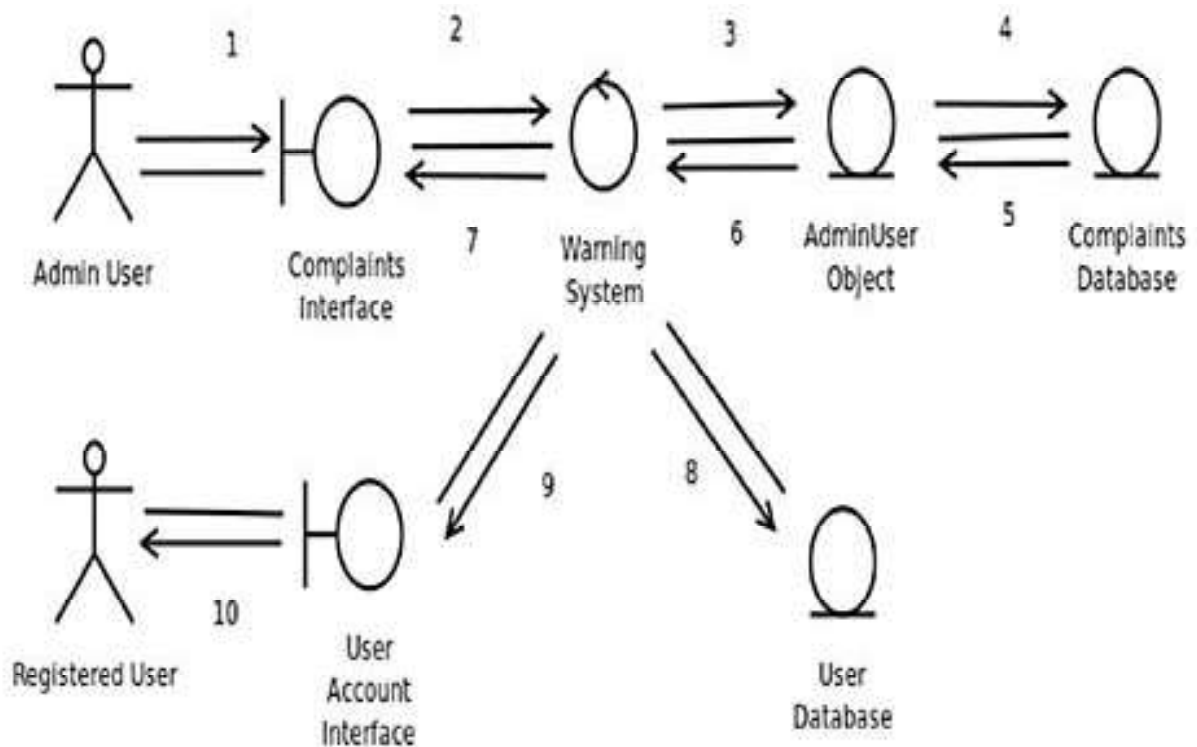
3.2.3.5. Logout



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

3.2.3.6. DeleteAccount



1. Admin-user views complaints Interface to see recent complaints
2. Admin-user clicks 'send warning' to user
3. Warning system accesses super-user object's 'warn user' method
4. The AU object sends an update to the complaints database
5. The complaint data (username, date, comment) is sent to the AU object
6. The complaint information is sent back to warning system
7. The complaint is removed from 'recent complaints'
8. The warning system sends query to increase the user's 'warnings' counter by one
9. A warning message (with date and comment) is sent to username
10. The Registered User receives warning message when he/she views their account page

Figure D3: Warn User

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

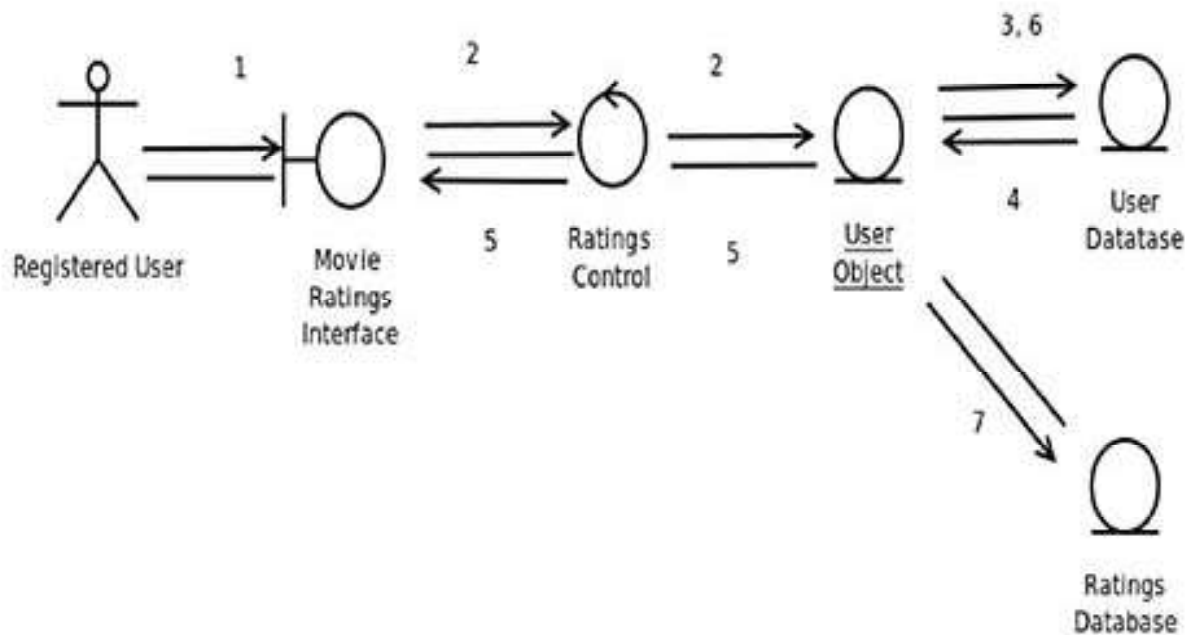
3.2.4.1. AddShareable

3.2.4.2. EditShareable

3.2.4.3. DeleteShareable

3.2.5. SHAREABLE: ANNOTATE

3.2.5.1. RateShareable



Figure

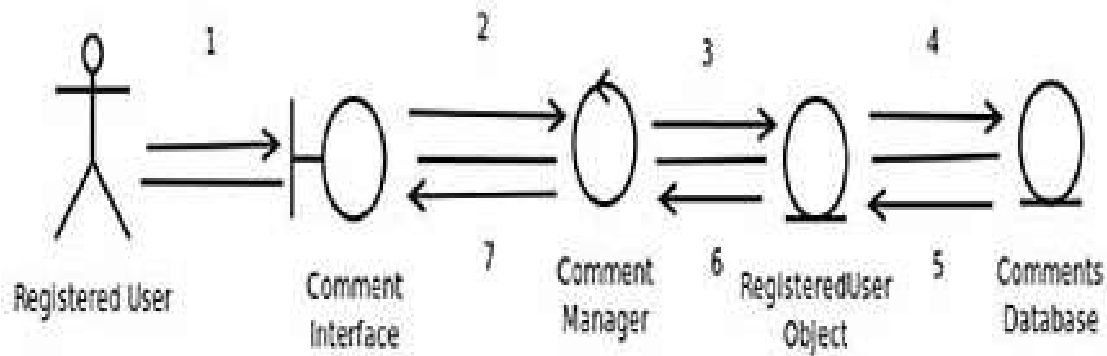
1. Registered User clicks on movie rating interface
2. Rating Interface submits UserID, MovieID, and Rating to Rating Control
3. Rating Control checks User DB to check User's ratings permissions
4. User DB returns RU's permissions
5. Ratings Control sends success or failure message back to Ratings GUI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

3.2.5.2. CommentOnShareable



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:

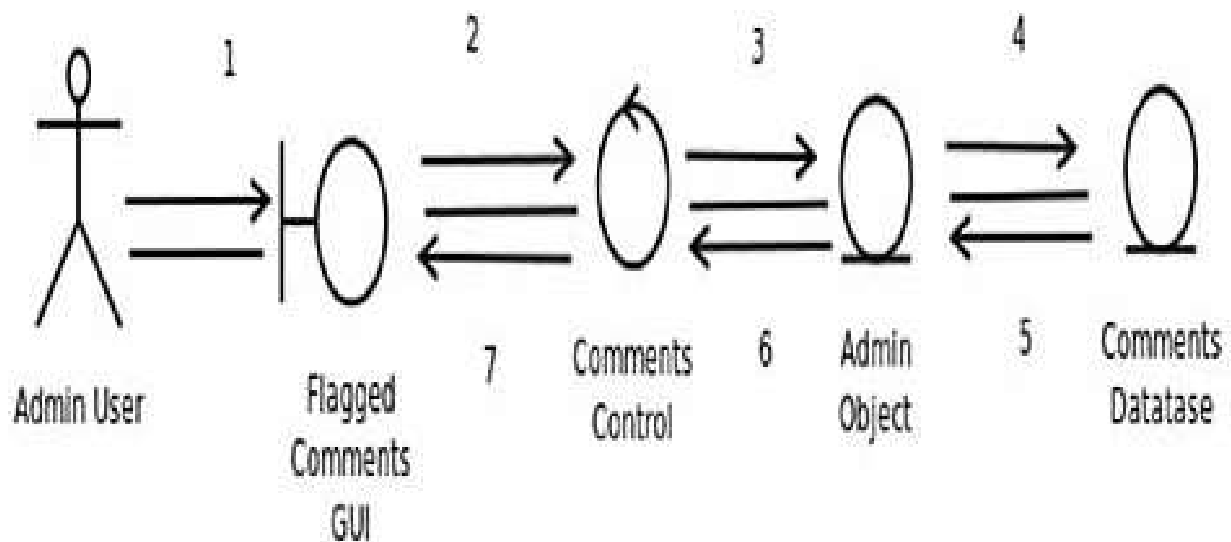
2. Blank comment is submitted
7. The comment manger displays an error message to the GUI

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

3.2.5.3. DeleteComment



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

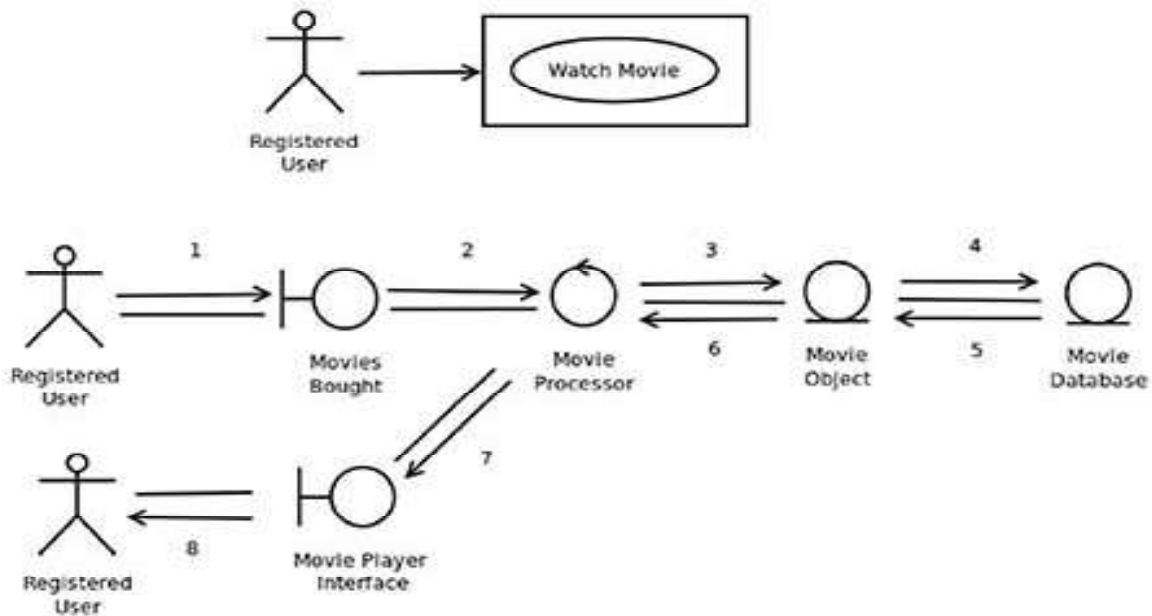
D1: Erase Comment

3.2.6. COMMUNICATIONS

3.2.6.1. SendMessage

3.2.6.3. DeleteMessage

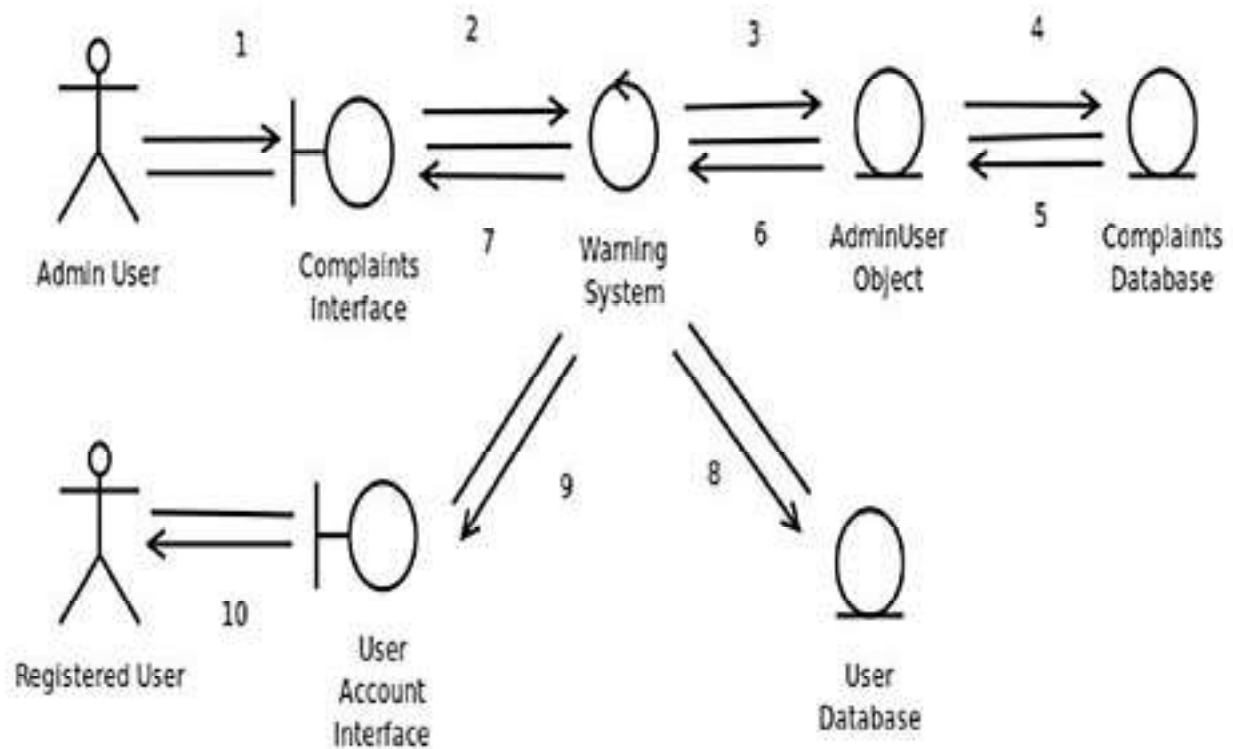
3.2.6.2. ReadMessage



1. Registered User visits 'movies bought' interface
2. RU clicks on a movie to watch
3. Movie processor creates instance of movie object
4. Movie object retrieves data on specific movie
5. Movie data is sent back to movie object
6. Movie data is sent back to movie processor
7. Movie Processor redirects data and user's location to movie player GUI
8. Movie Player Interface waits for RU to click play

C1: Watch Movie

3.2.6.4. BlockUser



1. Admin-user views complaints interface to see recent complaints
2. Admin-user clicks 'send warning' to user
3. Warning system accesses super-user object's 'warn user' method
4. The AU object sends an update to the complaints database
5. The complaint data (username, date, comment) is sent to the AU object
6. The complaint information is sent back to warning system
7. The complaint is removed from 'recent complaints'
8. The warning system sends query to increase the user's 'warnings' counter by one
9. A warning message (with date and comment) is sent to username
10. The Registered User receives warning message when he/she views their account page

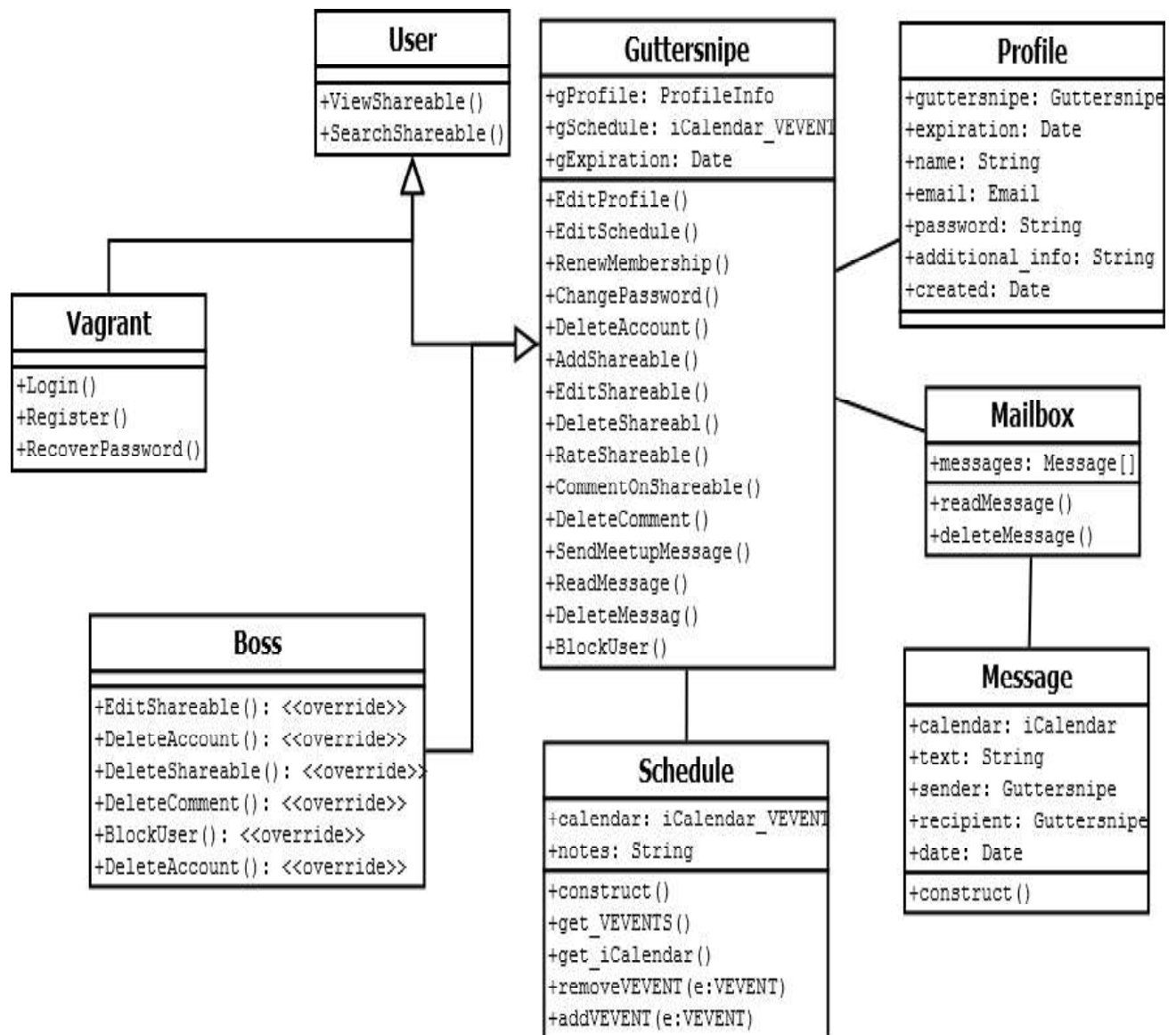
Figure D3: Warn User

Class D&D

(diagrams & definitions)

User Class

(A) Diagram



User Class (abstract)

* Methods:

- ++ViewShareable()
- ++SearchShareable()

Vagrant (extends the User Class)

* Methods:

- ++Login()
- ++Register()
- ++RecoverPassword()

Guttersnipe (extends the User Class)

* Attributes:

- ++gProfile: ProfileInfo
- ++gSchedule: iCalendar_VEVENT
- ++gExpiration: Date

* Methods:

- ++EditProfile()
- ++EditSchedule()
- ++RenewMembership()
- ++ChangePassword()
- ++DeleteAccount()
- ++AddShareable()
- ++EditShareable()
- ++DeleteShareabl()
- ++RateShareable()
- ++CommentOnShareable()
- ++DeleteCominent ()
- ++SendMeetupMessage()
- ++ReadMessage()
- ++DeleteMessage()
- ++BlockUser()

Boss

* Methods:

```
++EditShareable() : «override»  
++DeleteAccount(): «override»  
++DeleteShareable () : «override»  
++DeleteComment () : «override»  
++BlockUser () : «override»  
++DeleteAccount () : «override»
```

Attributes:

```
*** private $userID;  
*** private $username;  
*** private $accountType; *** private $userEmail;
```

Functions:

```
*** public function login($userID, $userPassword)  
// This allows a guest user to log in to the system and become a registered user or admin  
user  
*** public function register()  
// This allows a guest user to become a registered user  
*** public function complainComment($commentID)  
// This allows a guest user to send a complaint about a comment identified by  
commentID
```

RegisteredUser Class

ÿ(extends the User Class)

Attributes:

```
*** private $userID;  
*** private $username; *** private $userEmail;
```

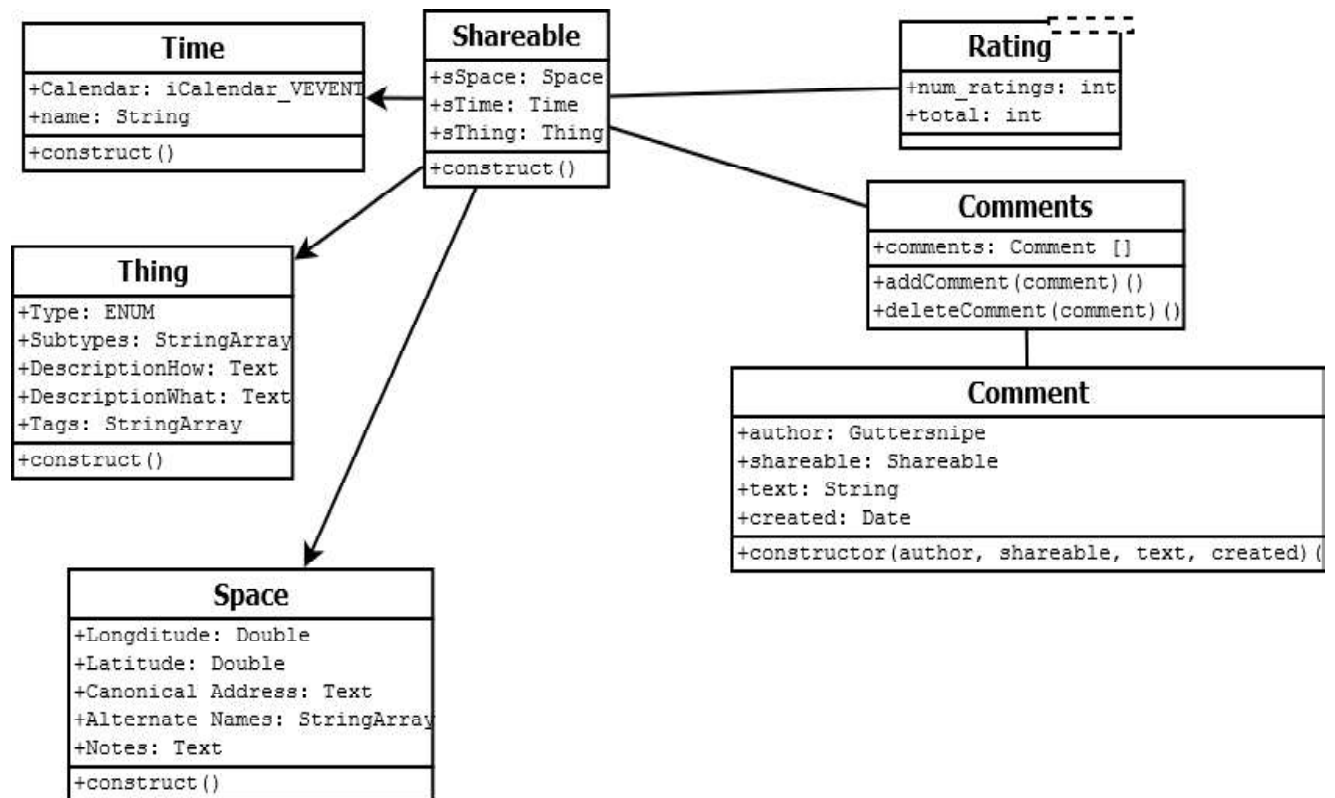
Functions:

```
*** public function getID()  
// This returns the id of the registered user  
*** public function getUsername()  
// This returns the username of the registered user  
*** public function getEmail()  
// This returns the email address of the registered user  
*** public function complainComment($commentID)  
// This allows a registered user to send a complaint about a comment identified by
```

```
commentID
*** public function login($userID, $userPassword)
// This allows a registered user to send a complaint about a comment identified by
commentID
*** public function rateMovie($userID, $movieID, $rating)
// This allows a registered user to give a rating to a movie.
*** public function submitComment($userID, $movieID, $commentText)
// This allows a registered user to post a comment to the movie page identified by
movieID
```

Shareable Class

(A) Diagram



i. Shareable Class and Parts

a. Shareable

a) *Attributes:

- (1) thing: Thing
- (2) space: Space
- (3) time: Time

b) Functions

- (1) __construct(thing, space, time) // Constructor for the Shareable class

b. Thing Class

a) *Attributes:

- (1) type: ENUM
- (2) subtypes: String []
- (3) descriptionHow: String
- (4) descriptionWhat: String
- (5) tags: StringArray []

b) Functions

- (1) __construct()

c. Space Class

a) *Attributes:

- (1) longitude: Double
- (2) latitude: Double
- (3) canonical Address: String
- (4) alternate Names: String[]
- (5) notes: String

b) Functions

(1) __construct()

d. Time Class

a) *Attributes:

(1) calendar: sCalendar_VEVENT

(2) notes: String

b) Functions

(1) __construct()

e. Rating Class

a) Attributes:

(1) num_rating: int

(2) total: int

b) Functions

(1) __construct()

f. Comments Class

a) Attributes:

(1) comments: Comment []

b) Functions

(1) __construct()

(2) addComment()

(3) deleteComment()

g. Comment Class

a) Attributes:

- (1) ++author: Guttersnipe
- (2) ++shareable: Shareable
- (3) ++text: String
- (4) ++created: Date

b) Functions

- (1) *** function __construct(\$commentID, \$commentText, \$userName, \$timestamp) // Constructor for the Comment class

B.

Managers

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

i. GUI Classes

(SearchGUI, BrowseGUI, ViewMovieGUI, LoginGUI, MainCustomerPage, MainAdminPage, RegisterGUI, RegistrationSuccessGUI, MoviesBought, MoviePlayerInterface, CommentGUI, ViewCartGUI, CartGUI, CheckoutGUI, ConfirmationGUI, ResetPasswordGUI, GuestPage, FlaggedCommentsGUI, ComplaintInterface, ComplaintsInterface, DeleteUserGUI)

ii. Database Tables

(MOVIES, R_USERS, RATINGS, PURCHASES, COMMENTS, COMPLAINTS)

Figure : Complaints.php

Figure :

Figure : register.php

Figure : firstLogin.php

Figure: ViewMovieInfo.php

Figure : browse.php

* Add functionality to current Classes

(
U,
GU, RU, AU, Movie, Cart
)

* Create new Object Classes

(
Player,
Comment,
Session?

All Database Returns Need Object Wrapper

)
* Create new Logic Classes

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

Create new GUI Classes

(
SearchGUI,
BrowseGUI, ViewMovieGUI, LoginGUI, MainCustomerPage, MainAdminPage,
RegisterGUI, RegistrationSuccessGUI, MoviesBought, MoviePlayerInterface,
CommentGUI, ViewCartGUI,
CartGUI, CheckoutGUI, ConfirmationGUI, ResetPasswordGUI, GuestPage,
FlaggedCommentsGUI, ComplaintInterface, ComplaintsInterface, DeleteUserGUI
)

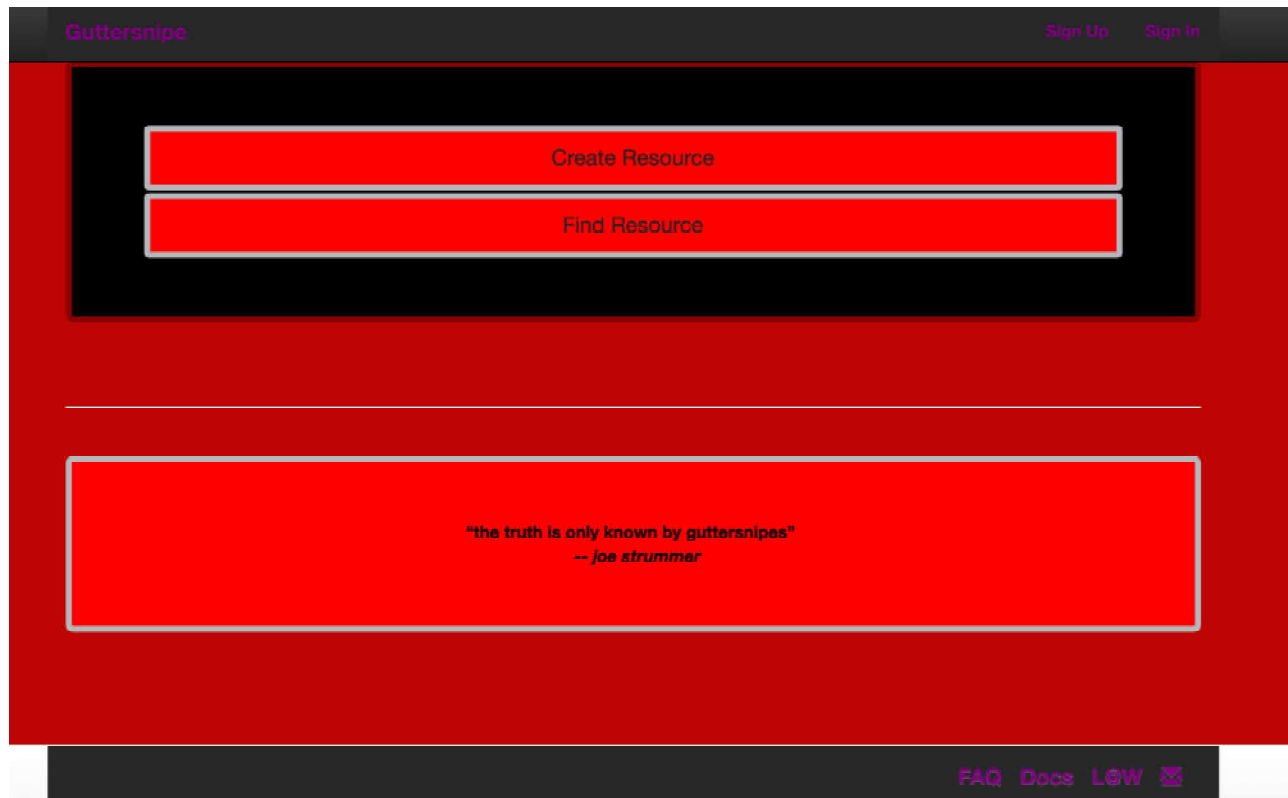
Create Database Tables

(
MOVIES,
R_USERS, RATINGS, PURCHASES, COMMENTS, COMPLAINTS

GUI COMPONENTS

1. Screenshots from 0.2 release and Future Wireframes

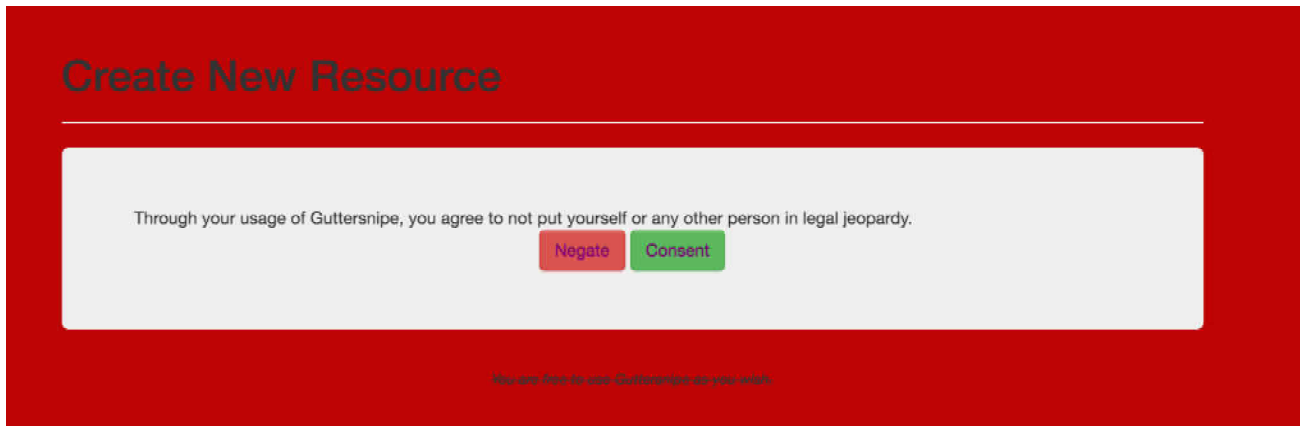
1.1. Front Page



Front Matter: Explanation

1.2. Create Shareable Wizard

1.2.1. CreateShareable: Start



The screenshot shows a red rectangular box representing a web interface. At the top left, the text "Create New Resource" is displayed in a bold, dark grey font. Below this text is a thin horizontal line. In the center of the box is a light grey rounded rectangle containing a disclaimer: "Through your usage of Guttersnipe, you agree to not put yourself or any other person in legal jeopardy." Below the text are two buttons: a red one labeled "Negate" and a green one labeled "Consent". At the bottom center of the red box, in a small, faint font, is the text "You are free to use Guttersnipe as you wish."

CreateShareable: Start

1.2.2. 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

CreateShareable: Instructions

1.2.3. 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?


Notes

Additional Notes

Confirm Description

CreateShareable: Describe

1.2.4. CreateShareable: Classify (I)



The image shows a screenshot of a web interface titled "Classify" in red text. Below the title, there is a prompt "Choose a resource type:" in red. Underneath this prompt are three white rectangular buttons with black text, arranged horizontally. The buttons are labeled "food", "medical", and "housing" from left to right. The entire interface is set against a black background and is framed by a thick red border.

CreateShareable: Classify (1)

1.2.5. CreateShareable: Classify(2)

Classify

Type

food

Edit

Subtypes

Available Subtypes

Free

Communal

Food

Donation

Dumpster

Your Selections

Food Not Bombs

Details

Add a Detail

Food is healthy

+

Your Details

vegan

Confirm Classification

CreateShareable: Classify (2)

1.2.6. CreateShareable: Map

1.2.7. CreateShareable: Schedule (1)

The screenshot shows a web form with a red background. At the top left, the word "Map" is displayed in a bold, black, sans-serif font. Below it, the text "Address of Resource :" is centered. A white text input field with a green border is positioned below the label. To the right of the input field is a blue button with the text "Find Address" in white. Below the input field is a large, empty white rectangular box with a thin black border. Below this box, the word "Notes" is displayed in a bold, black, sans-serif font. A white text input field with a green border is positioned below the "Notes" label. At the bottom left of the form is a white button with the text "Confirm Map" in black.

CreateShareable: Classify (3)

1.2.8. CreateShareable: Schedule (2)

Schedule

Sep 11 — 17 2016

	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

CreateShareable: Schedule (2)

1.2.9. CreateShareable: Schedule (3)

Choose Schedule

☒ Repeating Event

Event occurs every Monday

Start Time

▲	▲	
02	:	00 PM
▼	▼	

Duration

▲	▲	
03	:	30 PM
▼	▼	

Cancel

OK

CreateShareable: Schedule (3)

1.3. SearchShareable

1.3.1. SearchShareable: ResultList

Resources

List

Map

Calendar

Headline

La Bagel Delight

Thing

Type

- food

Subtypes

- dumpster

Place

104 Front St
Brooklyn, NY
11201

Time

- Every Monday
From Monday, 12:00 AM
To Monday, 2:59 AM
- Every Tuesday
From Tuesday, 12:00 AM
To Tuesday, 2:59 AM

Full Record

Created by Apr 16, 2015 1:21:24 PM by

Headline

Trader Joe's

Thing

Type

- food

Subtypes

- dumpster

Place

130 Court St
Brooklyn, NY
11201

Time

Full Record

Created by Apr 16, 2015 1:21:24 PM by

Headline

Peregrine

Thing

Type

- food

Subtypes

- dumpster

Place

175 Remsen St
Brooklyn, NY
11201

Time

- Every Monday
From Monday, 12:45 AM
To Monday, 2:59 AM

Full Record

Created by Apr 16, 2015 1:21:24 PM by

Filter Resources by

Thing

Place

Time

Thing

© 2015 SearchShareable, Inc.

SearchShareable: Results List

1.3. SearchShareable

1.3.2. SearchShareable: ResultCalendar

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
11am							
12pm							
1pm							
2pm		2:00 - 3:30 Event					
3pm							
4pm							
5pm							
6pm							

Schedule:

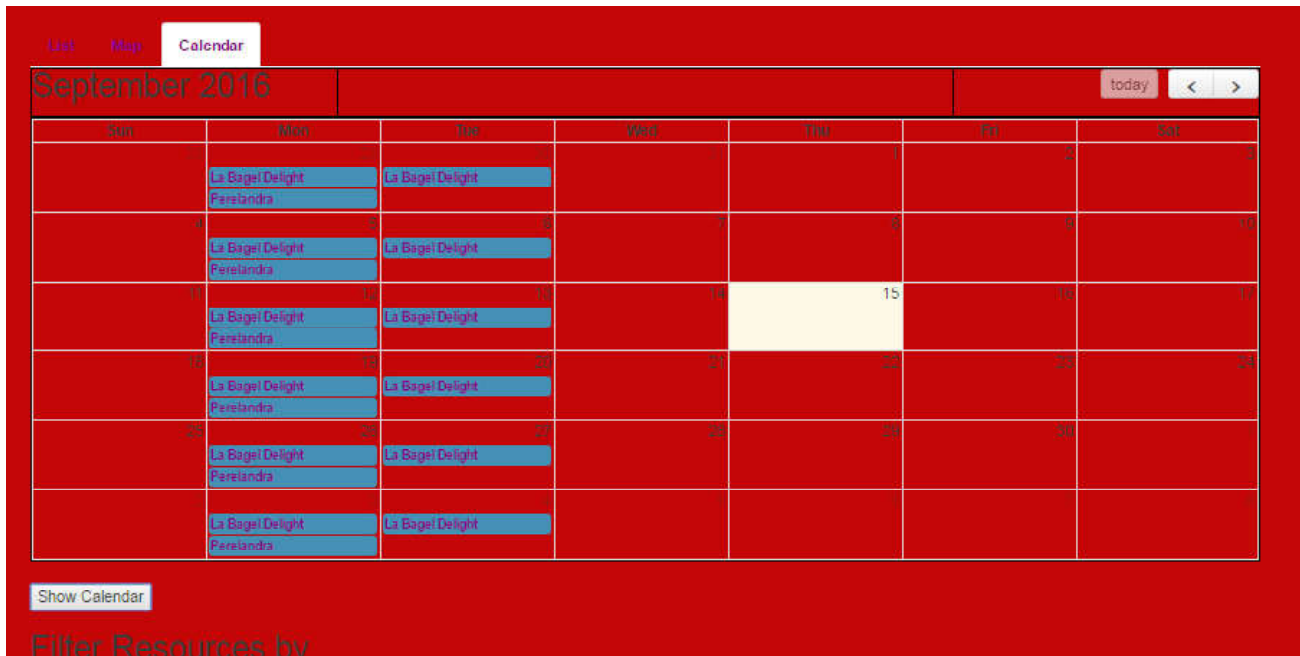
Date	Repeats	Start Time	End Time	Delete
	Every Monday	02:00 PM	03:30 PM	✕

Notes

GET THERE EARLY!!!!

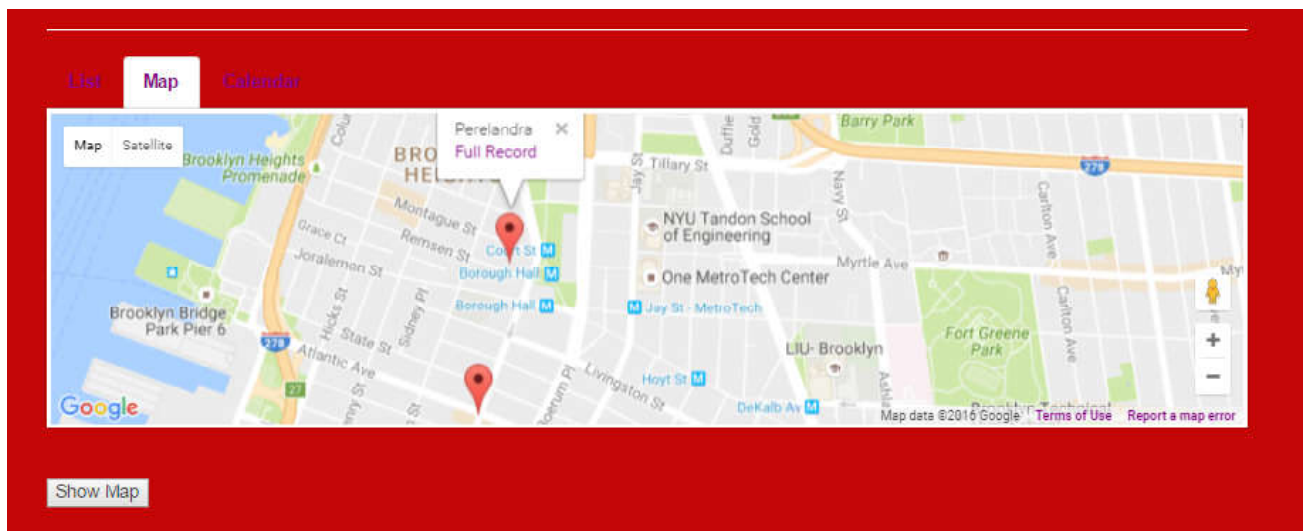
SearchShareable: Results Calendar

1.3.3. SearchShareable: ResultMap



SearchShareable: Results Calendar

1.3.4. SearchShareable: SearchByCategory



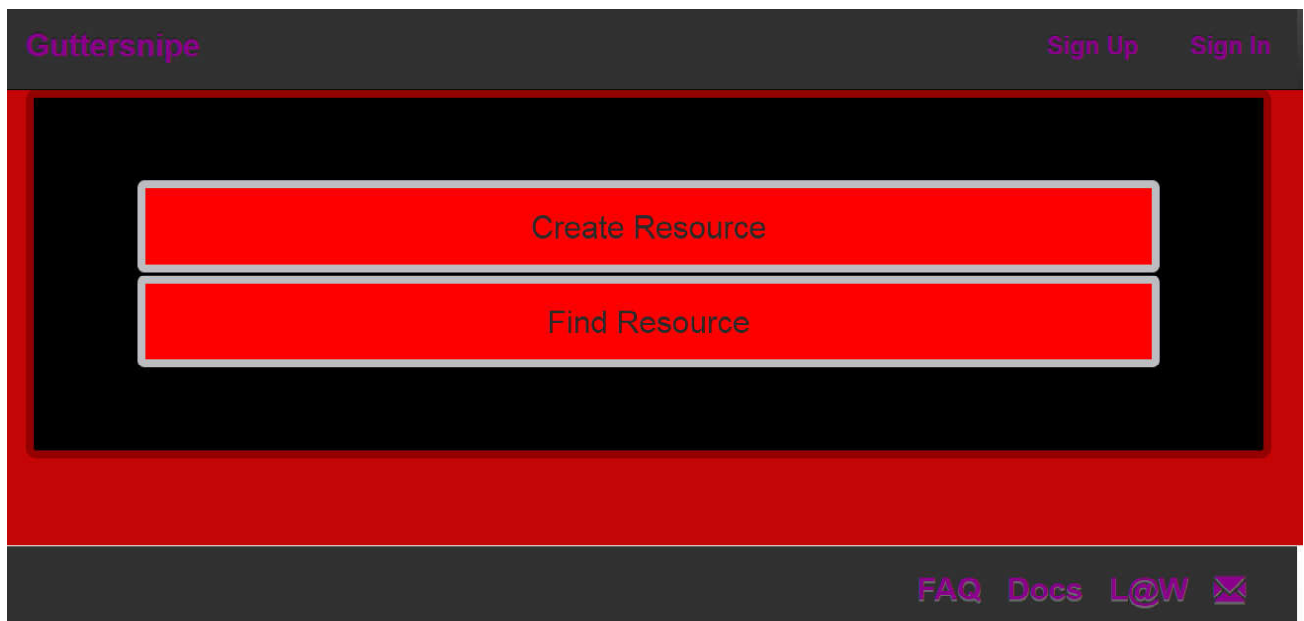
SearchShareable: Results Map

- 1.3.5. **SearchShareable: SearchByTag**
- 1.3.6. **SearchSharable: SearchByLocation**
- 1.3.7. **SearchShareable: SearchByTime**



1.4. Authentication

1.4.1. SignIn



04.04.01 Full Screen Sign In

1.4.2. SignIn Mobile

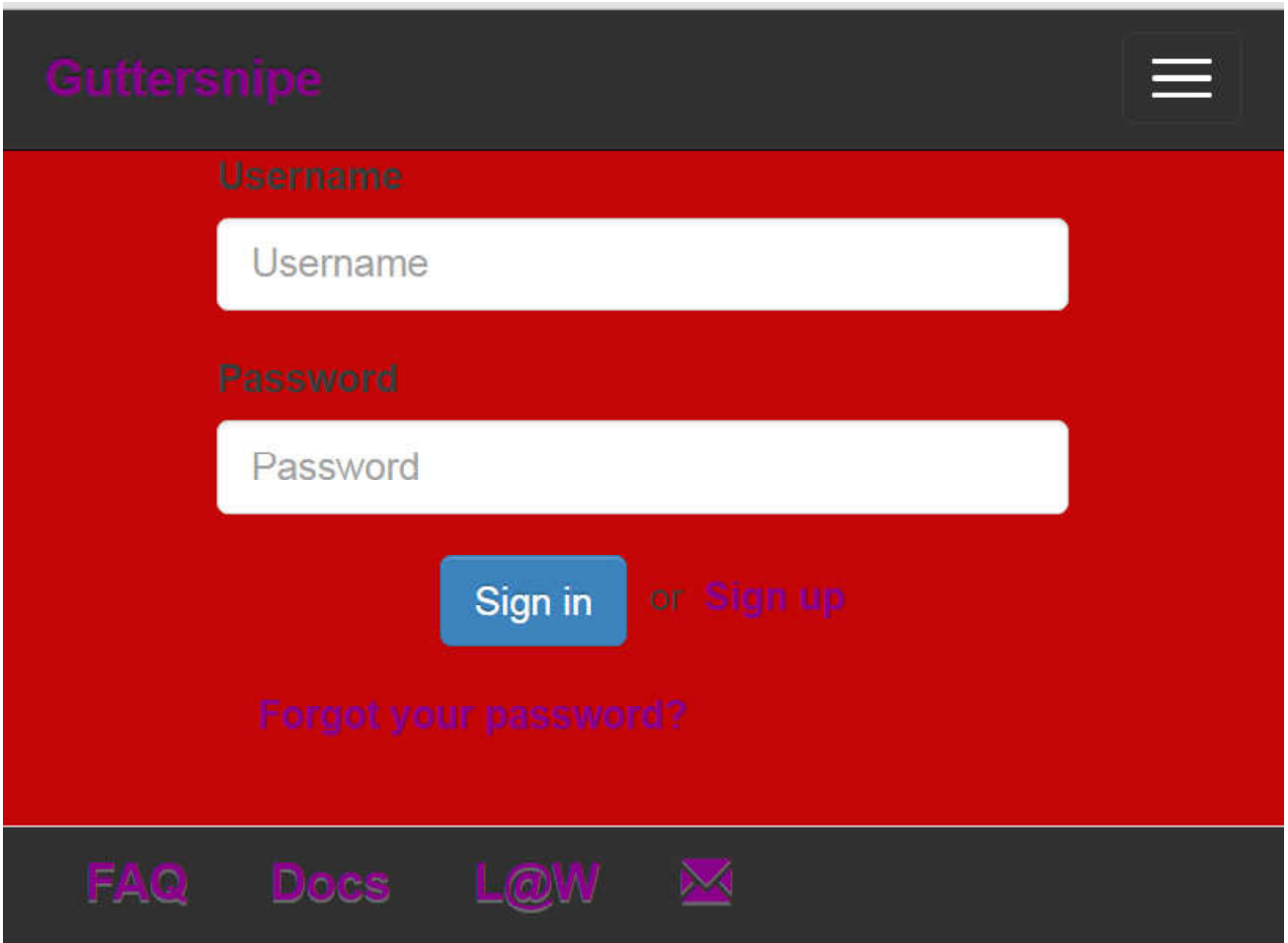


04.04.02
Burger Dropdown (2)

04.04.02
Burger Dropdown (2)



1.4.3 SignIn GUI



The image shows a web application interface for signing in. It features a dark grey header with the 'Guttersnipe' logo in purple on the left and a hamburger menu icon on the right. The main content area has a red background and contains the following elements: a 'Username' label above a white input field with 'Username' as placeholder text; a 'Password' label above a white input field with 'Password' as placeholder text; a blue 'Sign in' button followed by the text 'or Sign up' where 'Sign up' is in purple; and a purple link 'Forgot your password?'. The footer is dark grey and contains 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](#) 

1.4.4 SignUp

Sign up

First Name

Last Name

Email

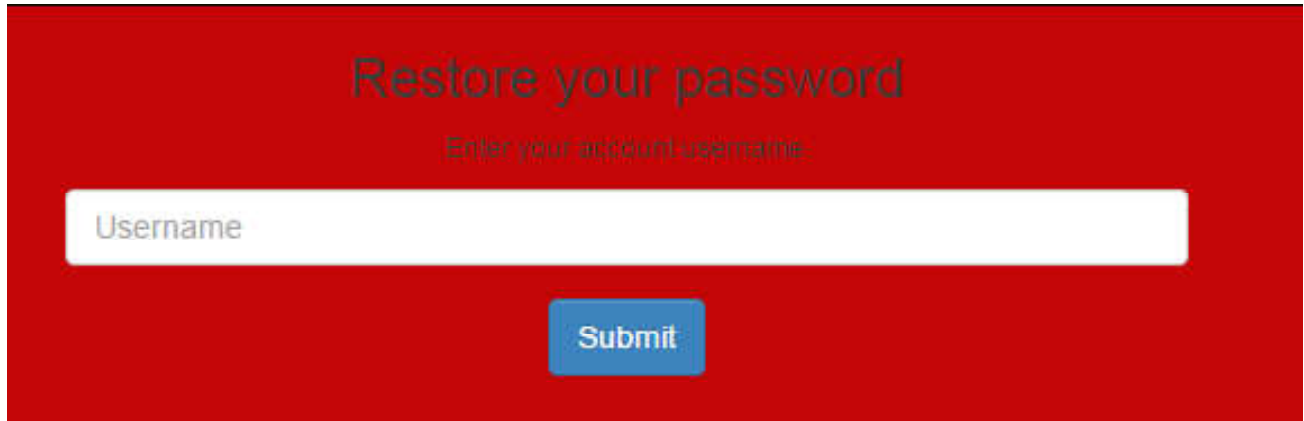
Username

Password

Sign up or Sign in

04.04.04 Guttersnipe User Pages Sign Up

1.4.5 RecoverPassword

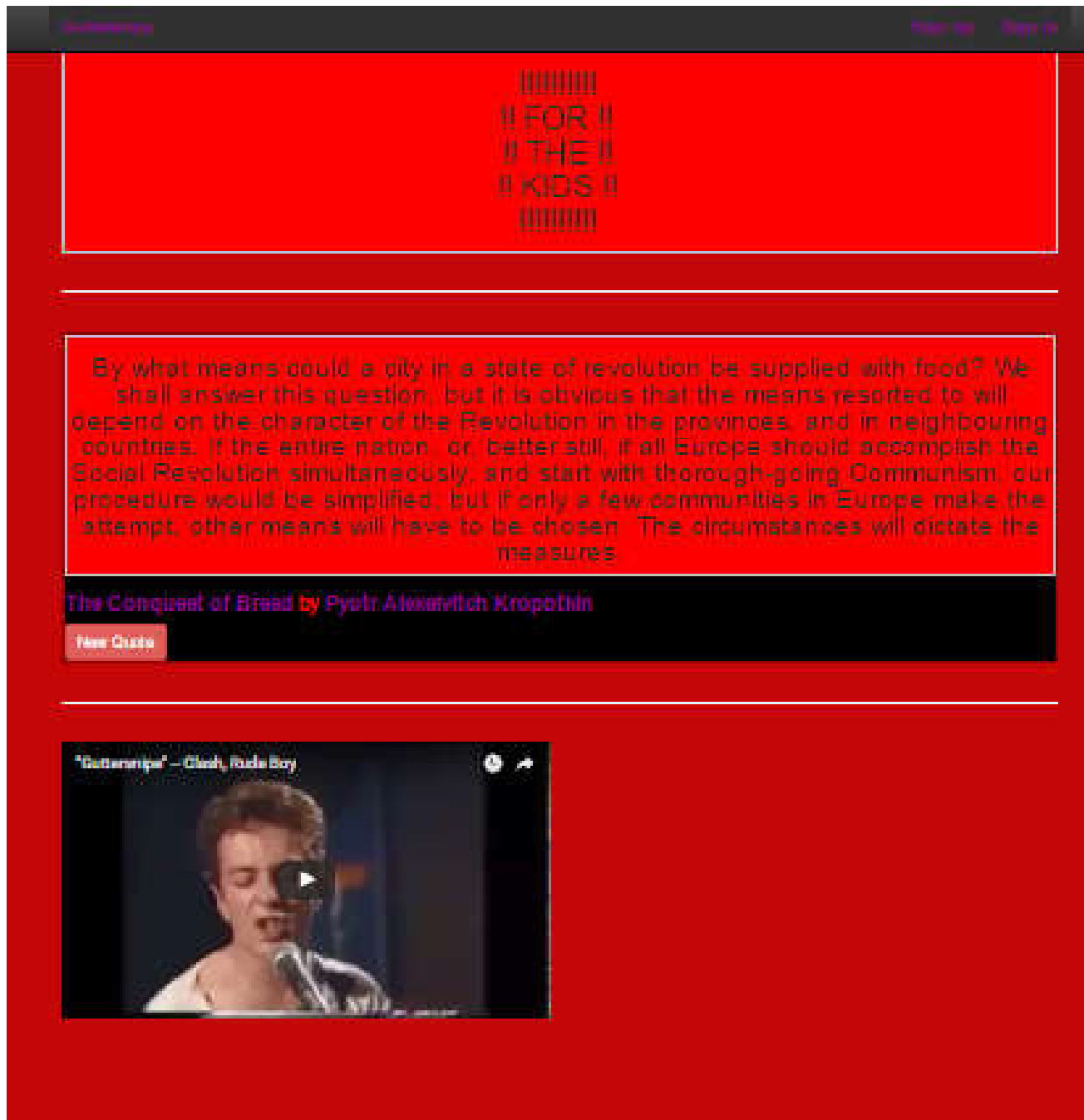


The image shows a web form for restoring a password. The form has a red background. At the top, the text "Restore your password" is displayed in a large, bold, white font. Below this, the instruction "Enter your account username" is written in a smaller, white font. There is a white text input field with the placeholder text "Username". Below the input field is a blue button with the text "Submit" in white.

04.04.06
Guttersnipe User Pages
Restore Password

1.5. Documentation

1.5.1. Mission Page



04.05.01 Mission Page

1.5.2. FAQ

PROPOSAL- GUTTERSNIPE FAQ

- [illegible]

04.05.02
FAQ Page

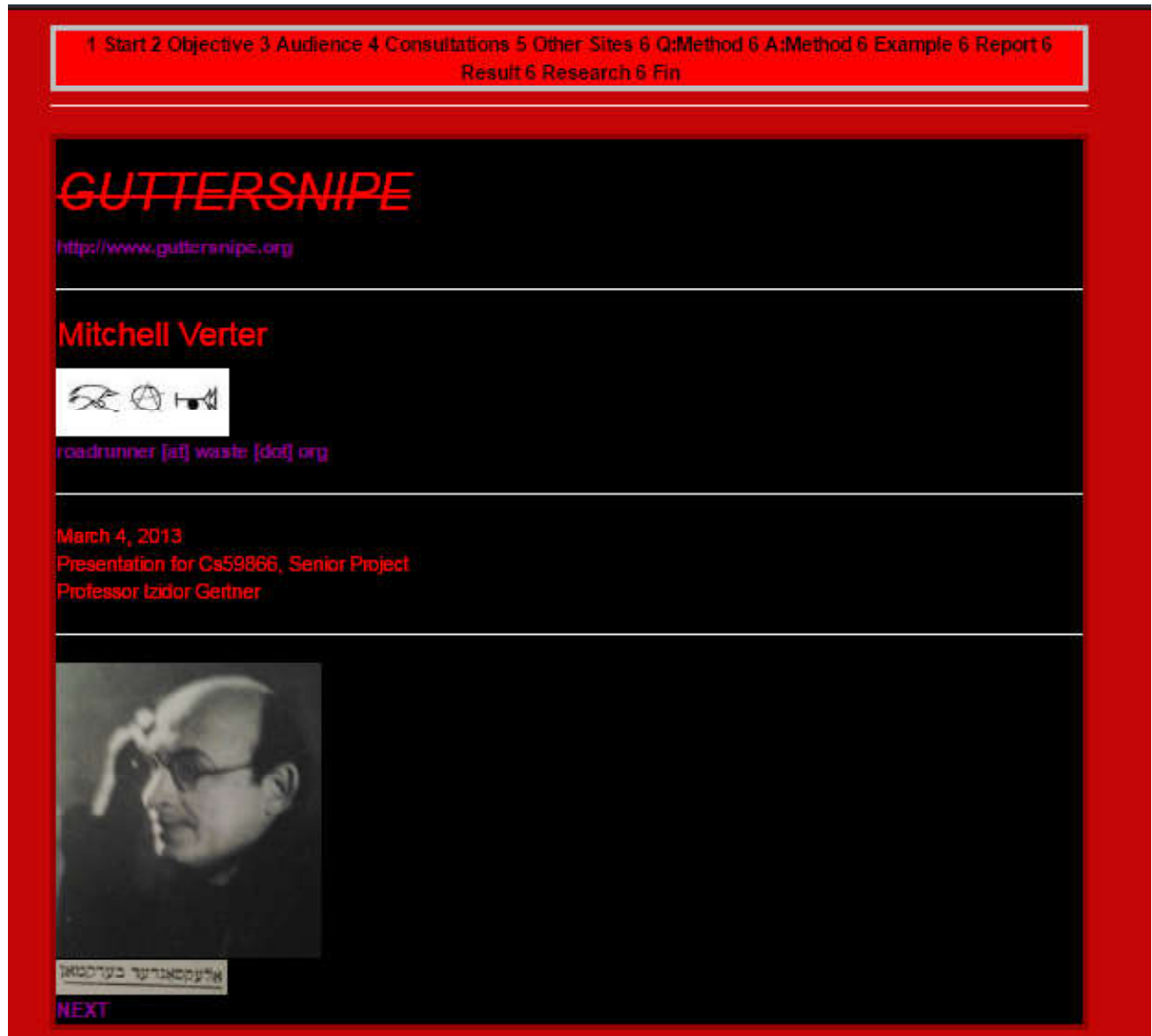
1.5.3. Presentation (2013)

1.5.3.01. Start



04.05.03.01 Presentation Start Page

1.5.3.02. Title Page



04.05.03.02 Presentation Front Page

1.5.3.03. Objective

The image is a screenshot of a presentation slide. At the top, the title 'Presentation: 2013' is displayed in a large, bold, black font. Below the title is a navigation bar with a series of numbered links: '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', and '6 Fin'. The '2 Objective' link is highlighted in red. Below the navigation bar is a large white rectangular area with a black border. Inside this area, the word 'Objective' is written in a large, bold, black font. Below 'Objective' is a bulleted list with two items: '• To overthrow capitalism by helping to establish mediums of exchange outside of the capitalist marketplace' and '• "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*'. At the bottom left of the white area, the words 'PREV' and 'NEXT' are written in red, with 'NEXT' being slightly larger and more prominent.

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

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*

PREV NEXT

04.05.03.03 Presentation Objective

1.5.3.04. Audience

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

Audience

- People who can not afford to or who choose not to participate in the capitalist marketplace
 - The Vietnamese Majority
 - Poor
 - Dislocated
 - Immigrants
 - Street people
 - Homeless
 - Traveler punks
 - Contingent/Precarious workers
 - Lumpenproletariat

PREV NEXT

04.05.03.04 Presentation Audience

1.5.3.05. Consultations

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

Consulting Organizations

- **Freogan 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

PREV NEXT

04.05.03.13 Presentation Consulting Organizations

1.5.3.06. Other Sites



04.05.03.06
Presentation
Other Sites

1.5.3.07. Q: Method



04.05.03.09 Presentation Question of Method

1.5.3.08. A: Method

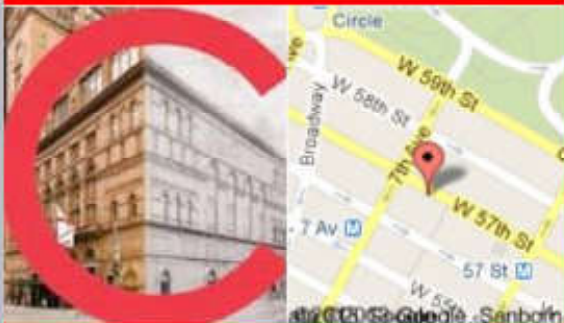
Presentation: 2013

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

- 1661 7th Ave
New York, NY 10019



PREV NEXT

04.05.03.08
Presentation
Question of Method

1.5.3.09. Example from 1999 W.A.S.T.E squat, want squat



04.05.03.09
Presentation
Example (2002)

1.5.3.10. Report (1999 GUI)

Presentation 2013

1 Start2 Objective3 Audience4 Consultations5 Other titles6 Q:Method6 A:Method6 Example6 Report6 Result6 Research6 Pin

Submit a property report

Use the following form to report 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 site code, you should try clicking on the [available codes](#).

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:

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.

Submit Report

Cancel

[What's new](#)

[What's new about this page](#)

[Find vacant properties](#)

[Look at maps](#)

[Read property's thoughts on this project](#)

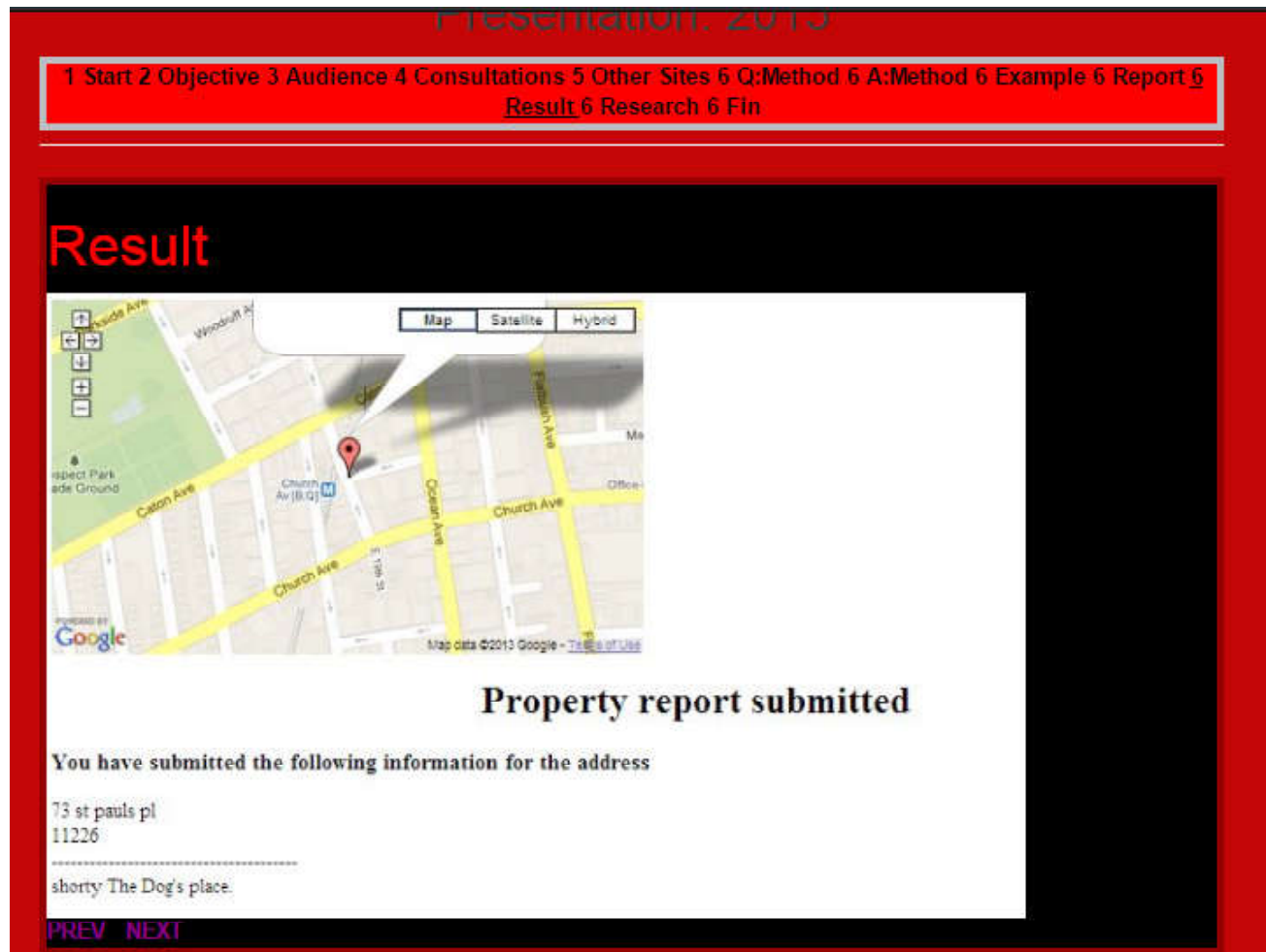
[Privacy](#)

[About](#)

04.05.03.12
Presentation

71

1.5.3.11. Result (1999 GUI)



04.05.03.11 Presentation Property Search Results (2002)

1.5.3.12. Research (1999 GUI)

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](#)

04.05.03.12 Presentation Property Report (2002)


1.5.3.13. Fin

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


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)
PREV

04.05.03.13
Presentation
FIN

1.5.4. Administrative

1.5.4.1. Legal



04.05.01
L@W Page

1.5.4.2. About



04.05.04.02 About Page

1.5.4.3. Contact



04.05.04.03 Contact

1.5.5. Kropotkins

1.5.5.1. Kropotkin Quotes

By what means could a city in a state of revolution be supplied with food? We shall answer this question, but it is obvious that the means resorted to will depend on the character of the Revolution in the provinces, and in neighbouring countries. If the entire nation, or, better still, if all Europe should accomplish the Social Revolution simultaneously, and start with thorough-going Communism, our procedure would be simplified; but if only a few communities in Europe make the attempt, other means will have to be chosen. The circumstances will dictate the measures.

The Conquest of Bread by Pyotr Alexeyevich Kropotkin

More Quotes

04.06.01 Kropotkin Quote

1.5.6. Screenshot

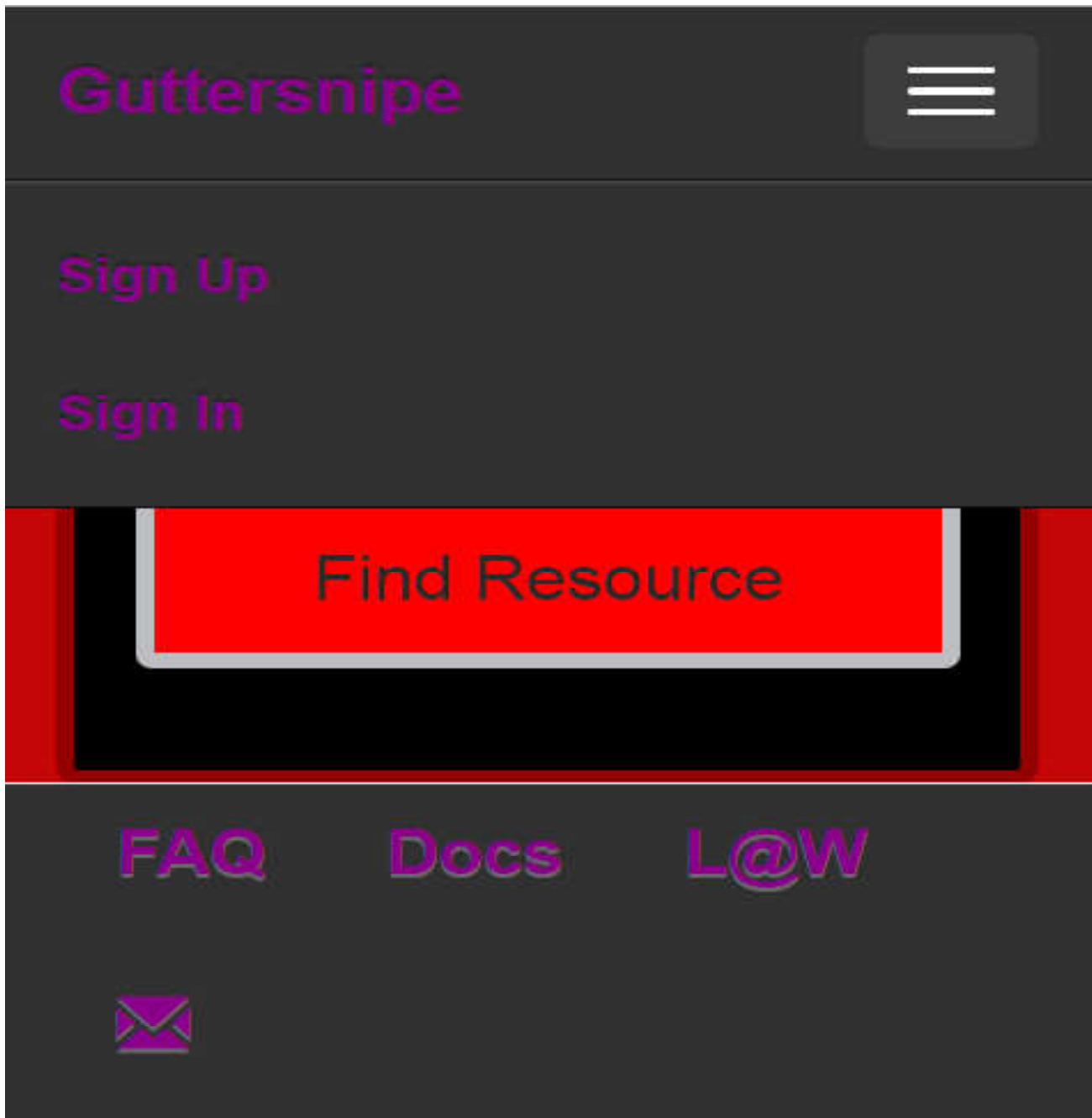
1.5.6.1. Dropdown top menu (1)



~~04.04.07~~

~~BurgerDropdown (1)~~

1.5.6.2. Droptown to menu (2)



04.07.02 BurgerDropdown (2)