Design Specifications

for

easyRent

Version 1.0

Prepared by

Group Name: Foo-Tang Clan

Abraham Dela Cruz	**abrahamdc@gmail.com**
Raj Luhar	**rluhar1990@gmail.com**
Chris Horuk	**chris@horukmail.com**
Joey Phommasone	**joey.phom@gmail.com**
Zack Warburg	**zwarburg@hotmail.com**

Instructor: Chandra Krintz

Course: CMPSC 189A

Teaching Assistant: Stratos Dimopoulos

Date: March 7th, 2013

Contents

1 Introduction

1.1 Product Overview

easyRent is a tenant-facing mobile application which allows a user to pay rent and submit maintenance requests. The application will interact with AppFolio's existing backend database, letting the user access payment and maintenance history. The mobile application will also let the tenant schedule notifications regarding rent and other information from his or her respective property management companies.

2 App Layout

2.1 Login Pages

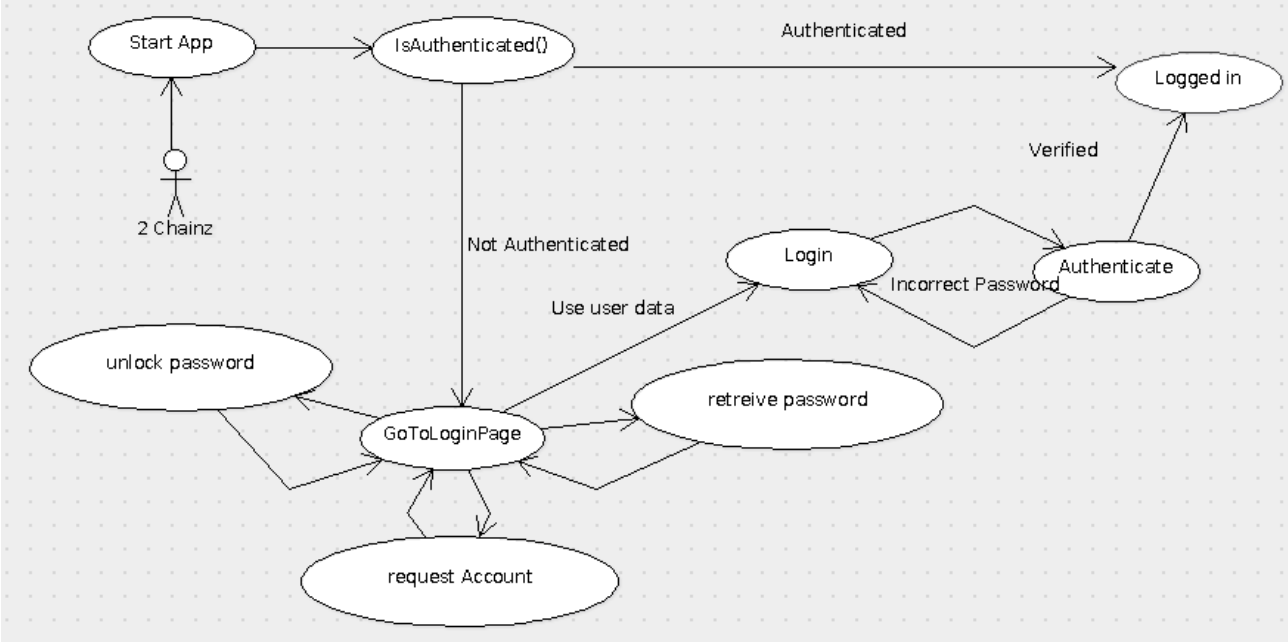
2.1.1 Logging In

Overview:

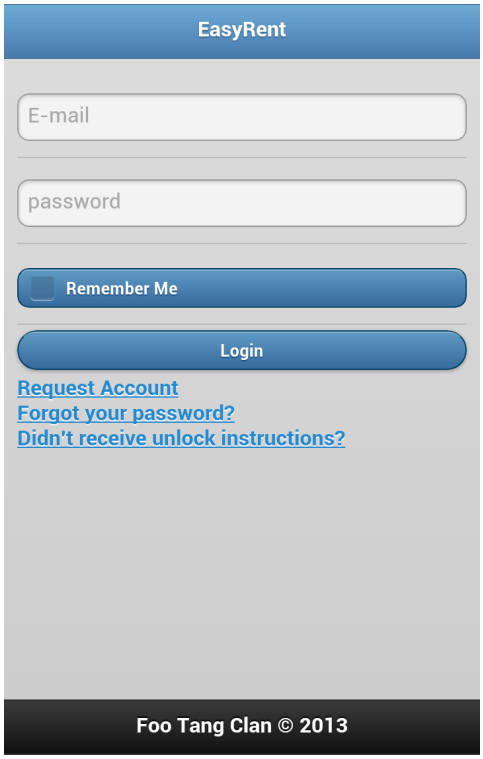
From this page a user will be able to:

- Login
- Request an account
- Deal with a forgotten password
- Resend unlock instructions

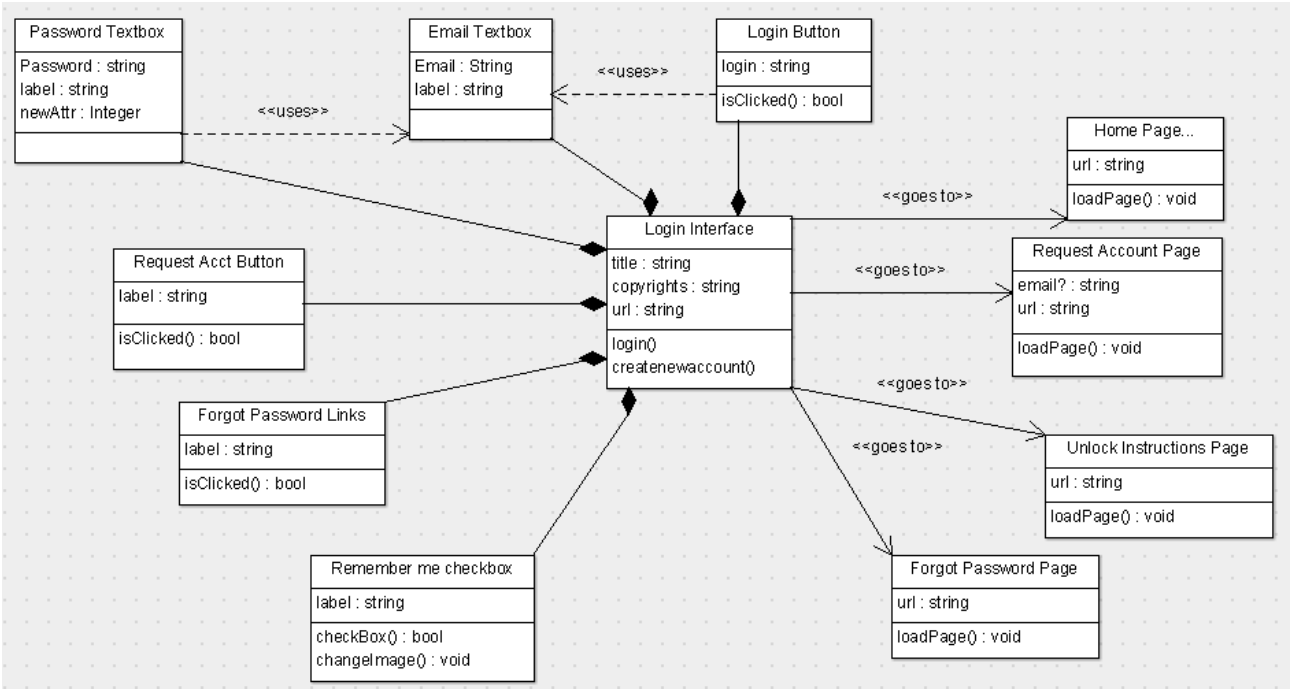
Activity Diagram



UI Mock-Up



UML Diagram



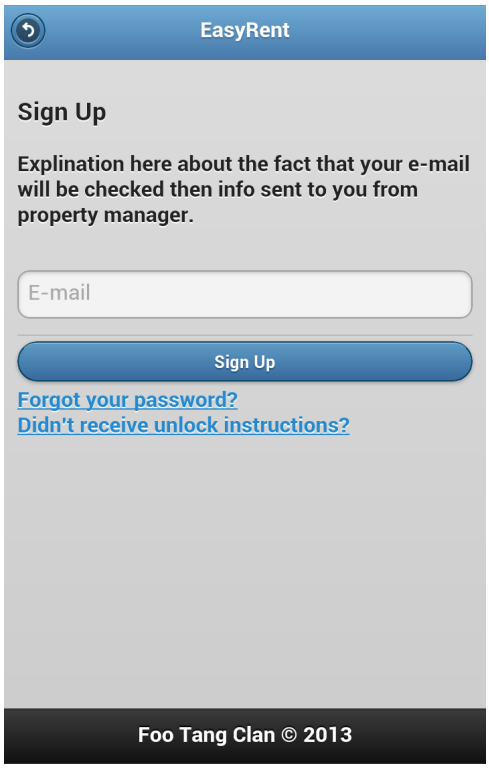
2.1.2 Request Account

Overview:

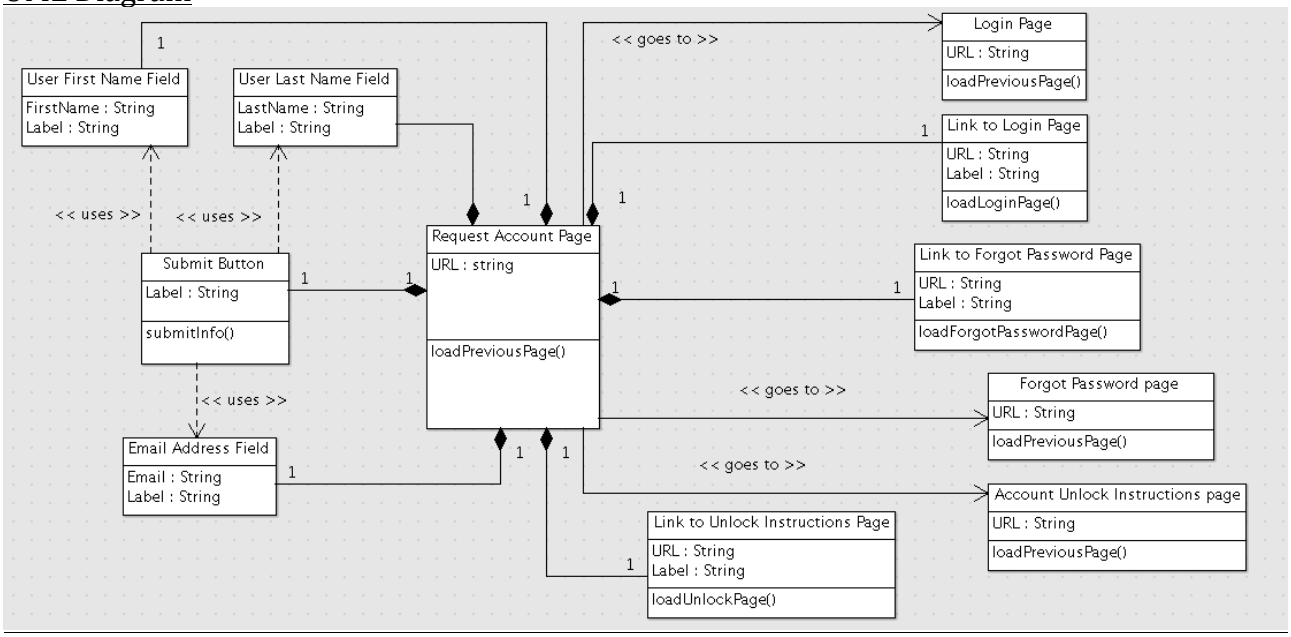
From this page a user will be able to:

- Enter an email address and full name
- Submit their new credentials and wait for verification

UI Mock-Up



UML Diagram



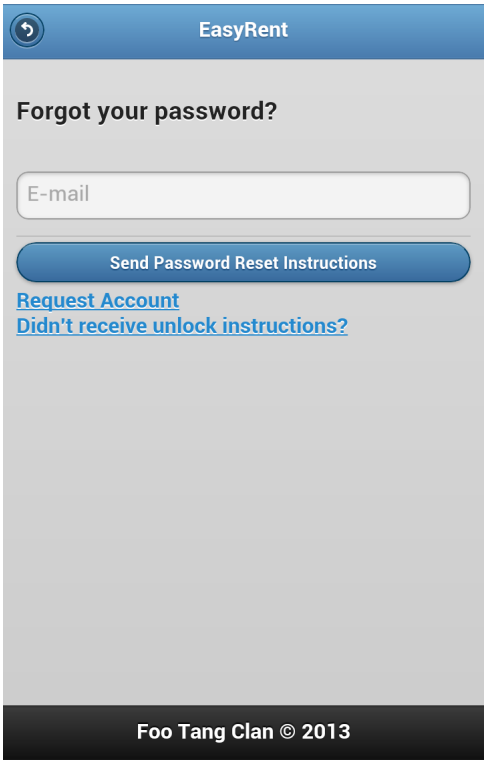
2.1.3 Reset Password

Overview:

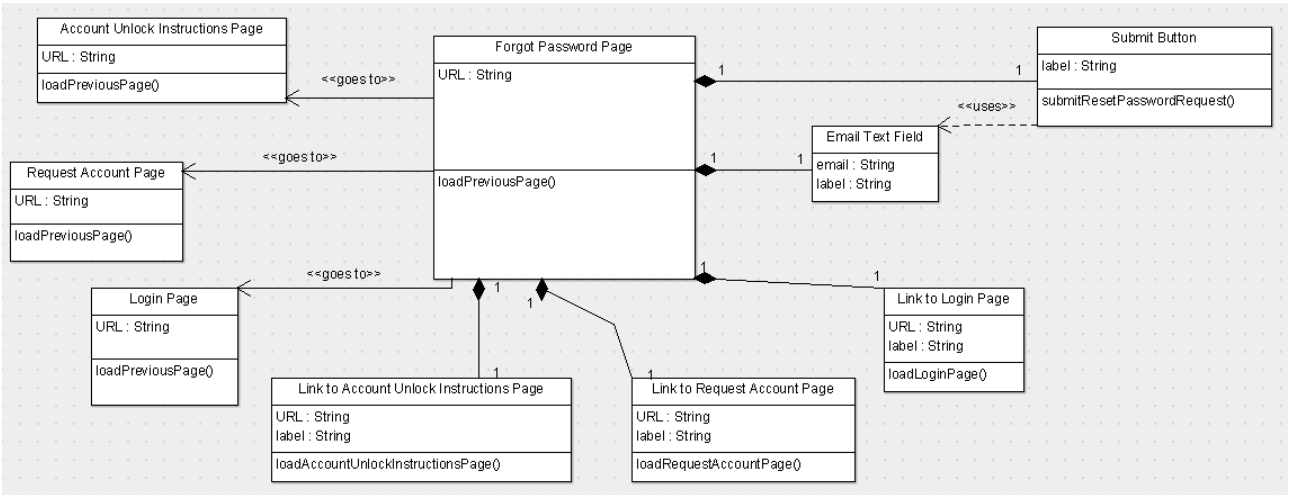
From this page a user will be able to:

- Enter account email address
- Submit the email address and have a link sent to this address to set up a new password for the user.

UI Mock-Up



UML Diagram



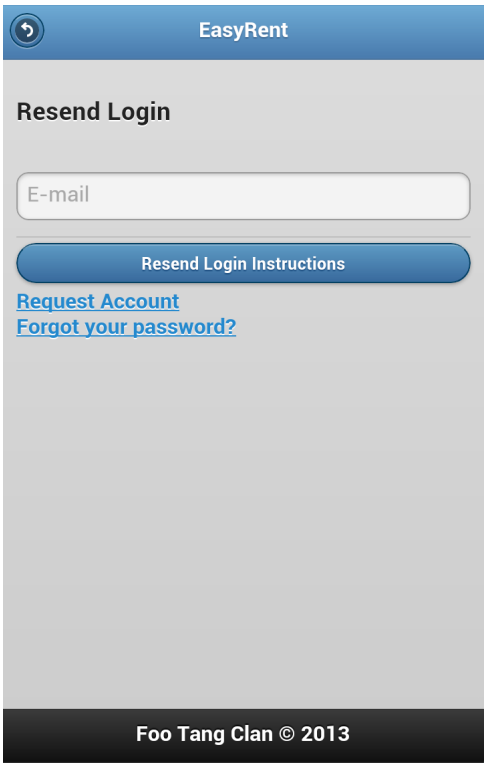
2.1.4 Resend Unlock Instructions

Overview:

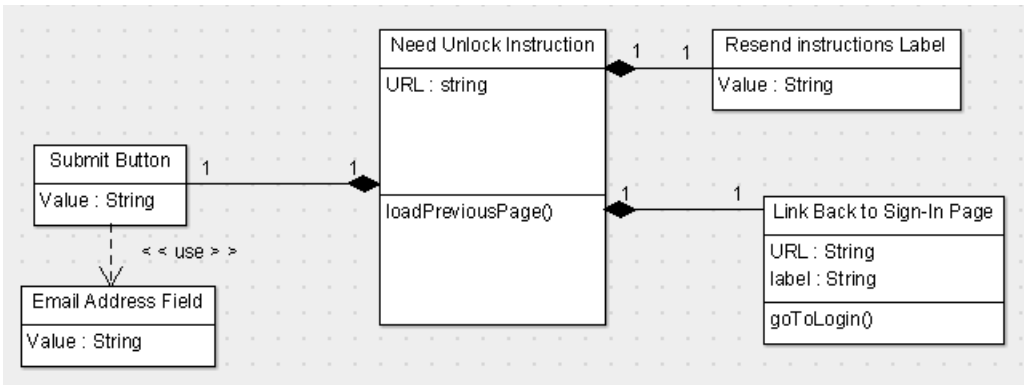
From this page a user will be able to:

- Enter account email address.
- Submit the email address and have a link sent to this address to unlock the account associated with this email address

UI Mock-Up



UML Diagram



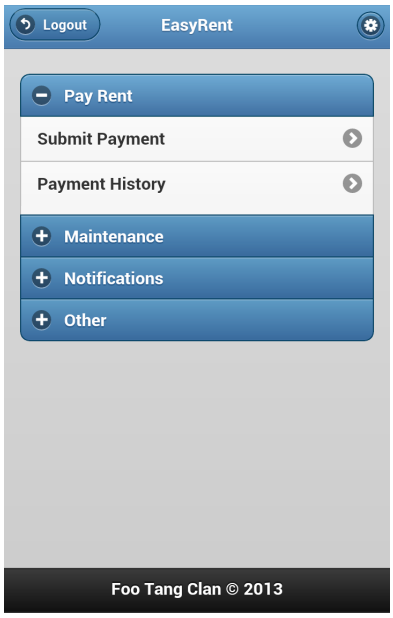
2.2 Home Page

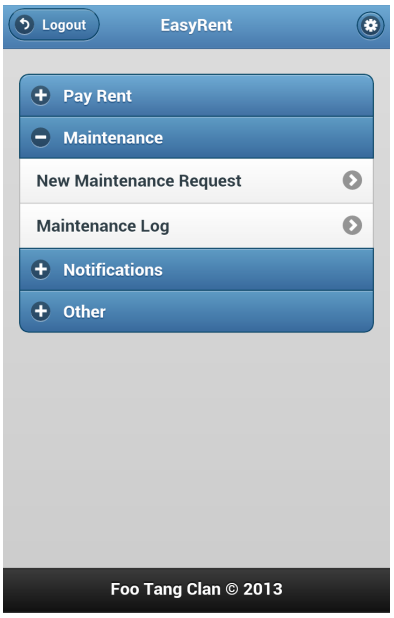
Overview:

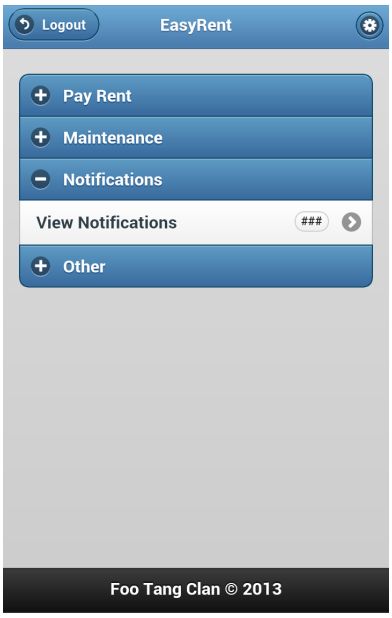
From this page a user will be able to:

- Navigate to the other sections of the app.
- Log out from the app.

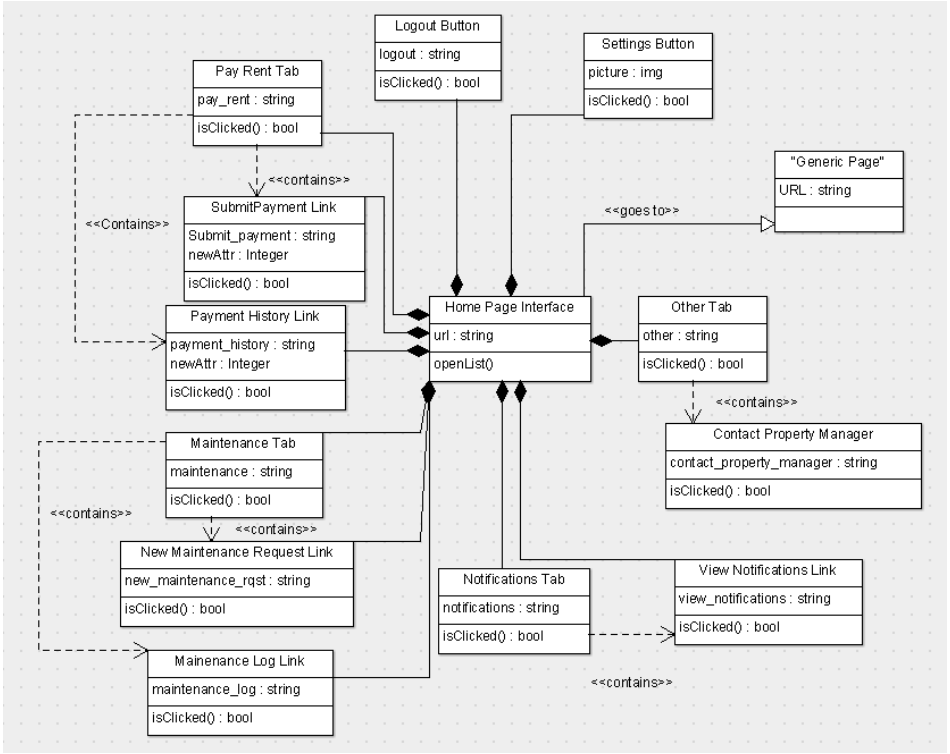
UI Mock-Up







UML Diagram



2.3 Paying Rent

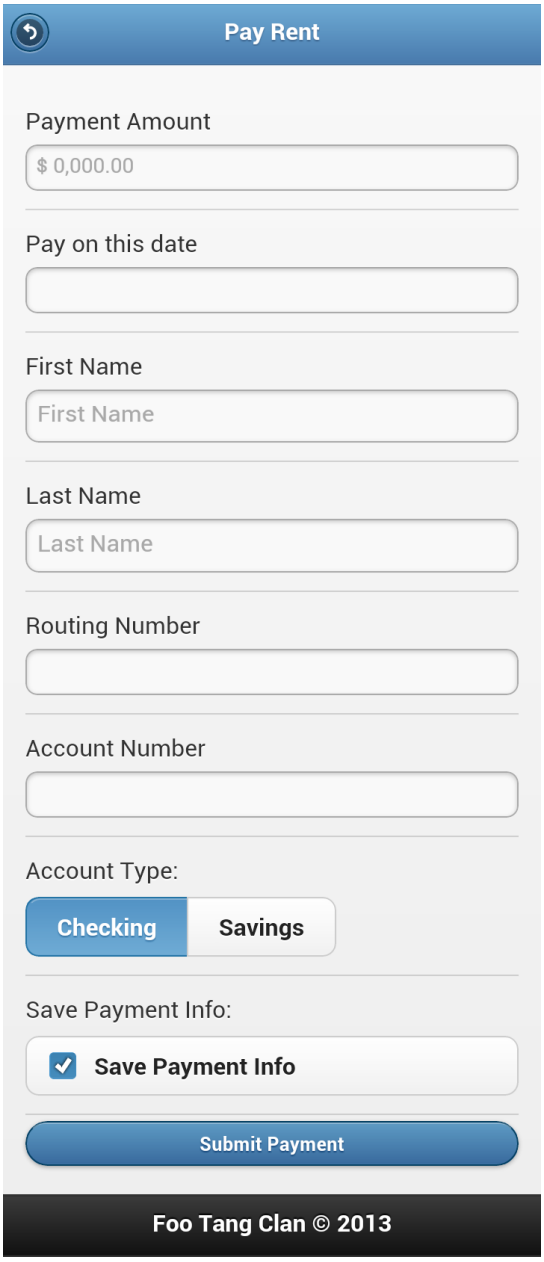
2.3.1 Submit Payment

Overview:

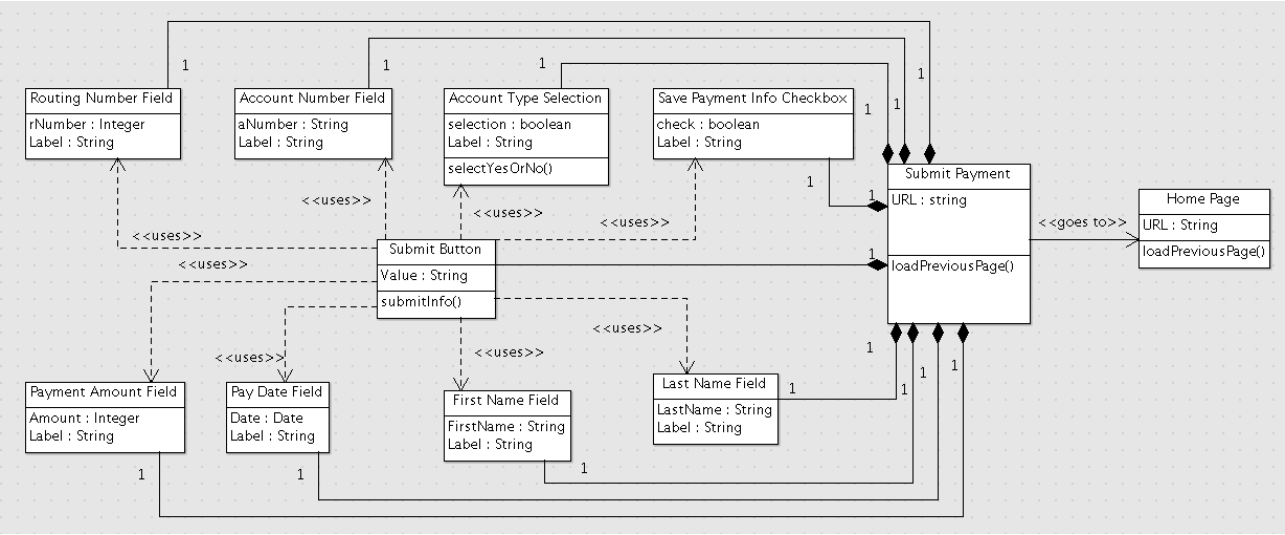
From this page a user will be able to:

- Submit a new payment.
- Input their bank account information
- See the amount that is due
- Choose how much they want to pay.

UI Mock-Up



UML Diagram



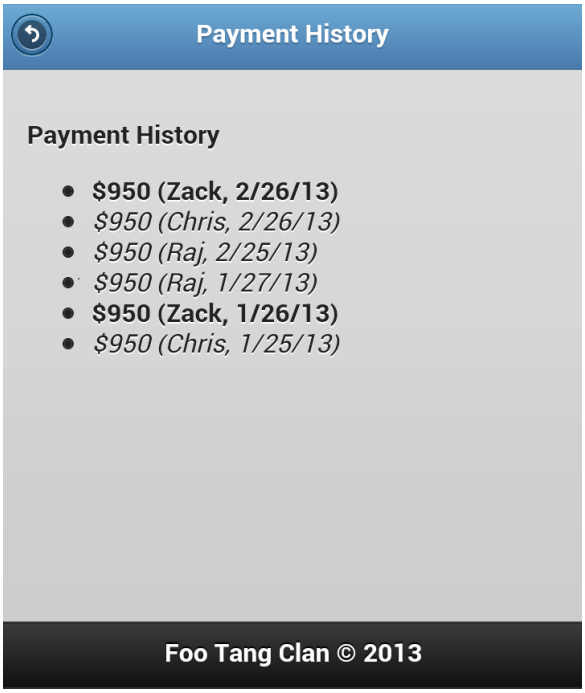
2.3.2 Payment History

Overview:

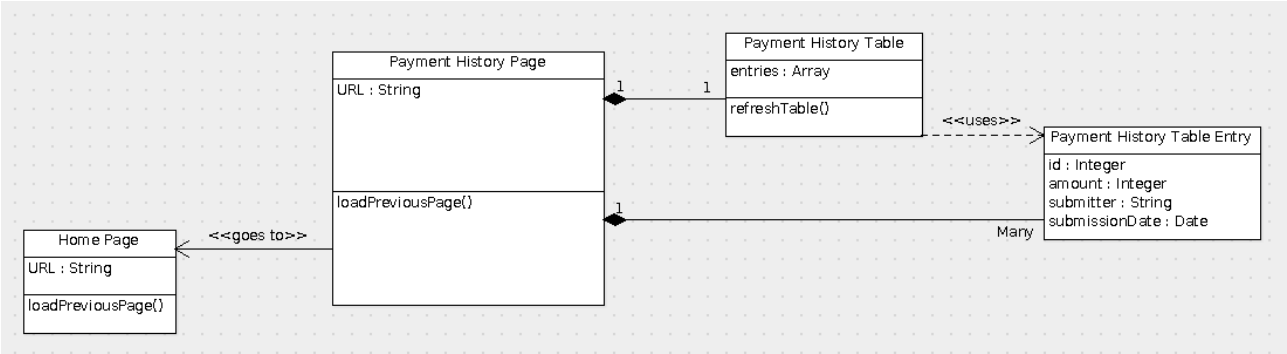
From this page a user will be able to:

- See all payments that they have made
- View payments made by their roommates
- View any fees or fines that have been incurred against them.

UI Mock-Up



UML Diagram



2.4 Maintenance

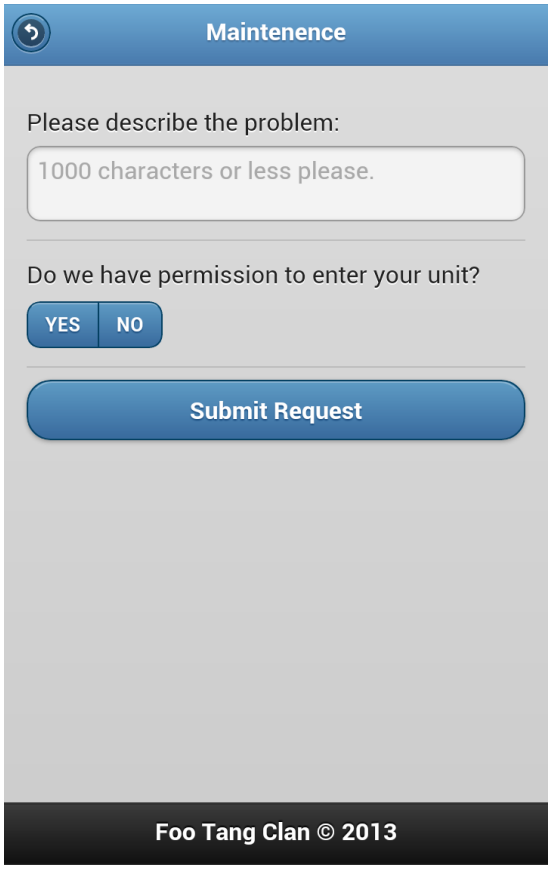
2.4.1 New Request

Overview:

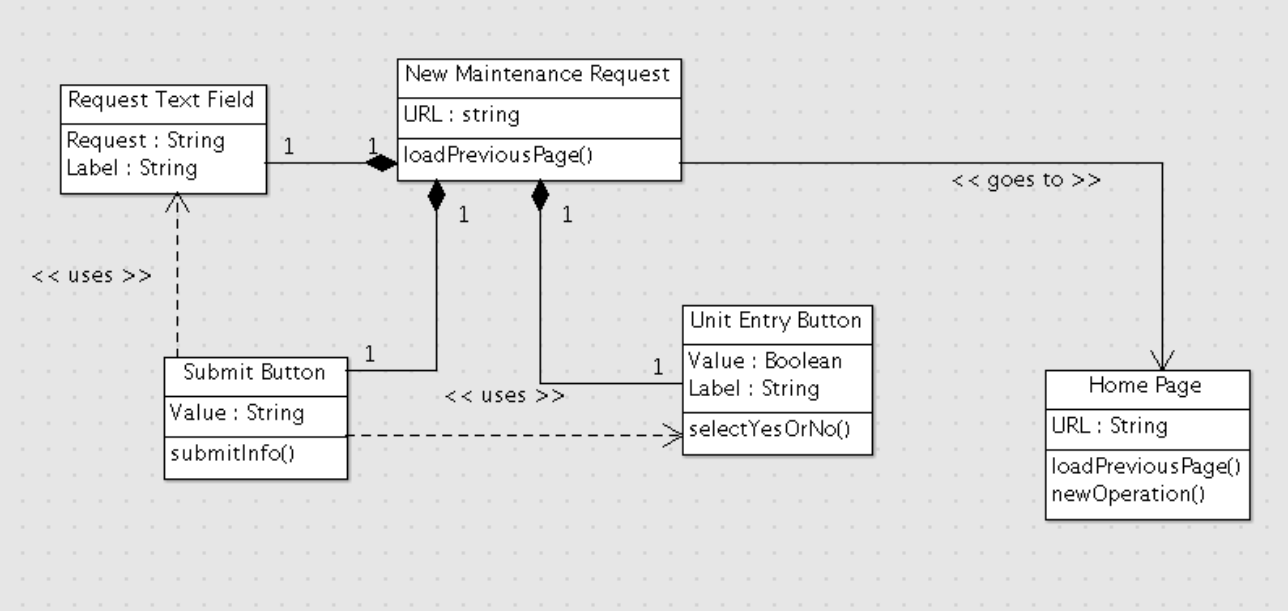
From this page a user will be able to:

- Submit a maintenance request.
- Explain what the problem is.
- Say whether maintenance personnel have permission to enter their apartment.

UI Mock-Up



UML Diagram



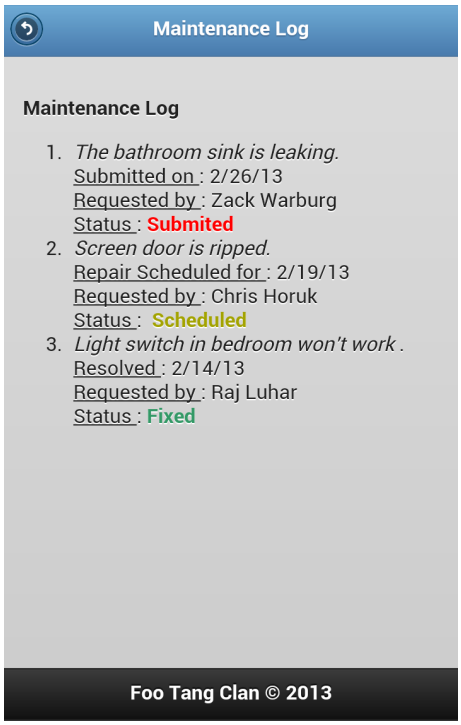
2.4.2 Maintenance Log

Overview:

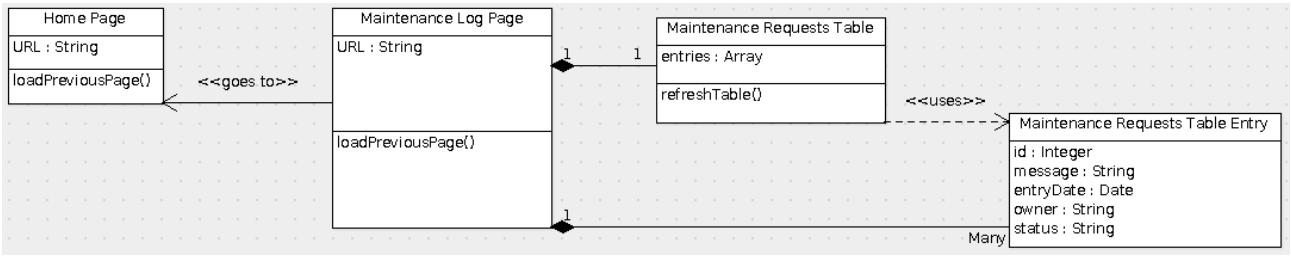
From this page a user will be able to:

- View a log of all requests that have been made by both them and their roommates.
- View if and when issues have been resolved.

UI Mock-Up



UML Diagram



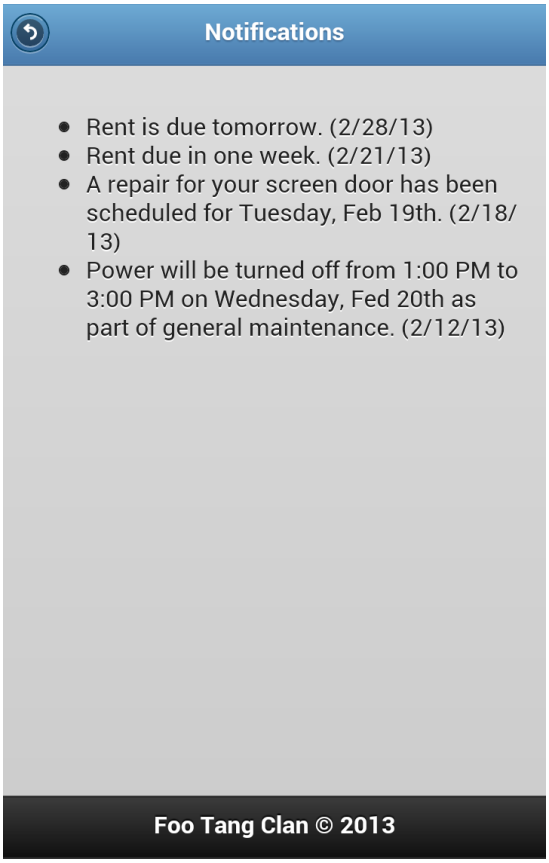
2.5 Notifications

Overview:

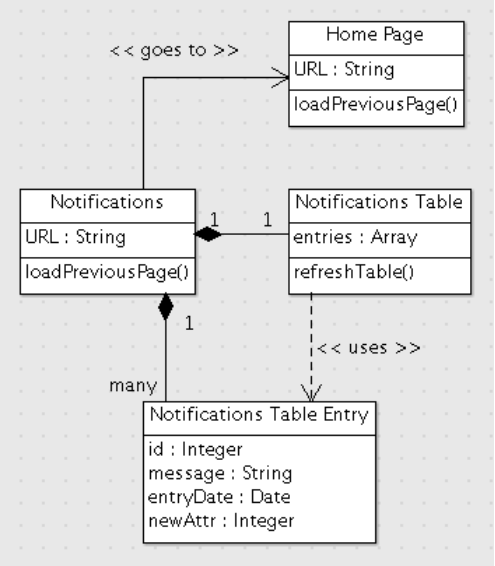
From this page a user will be able to:

- View notifications sent out by the property manager.
- View reminders about paying rent.

UI Mock-Up



UML Diagram



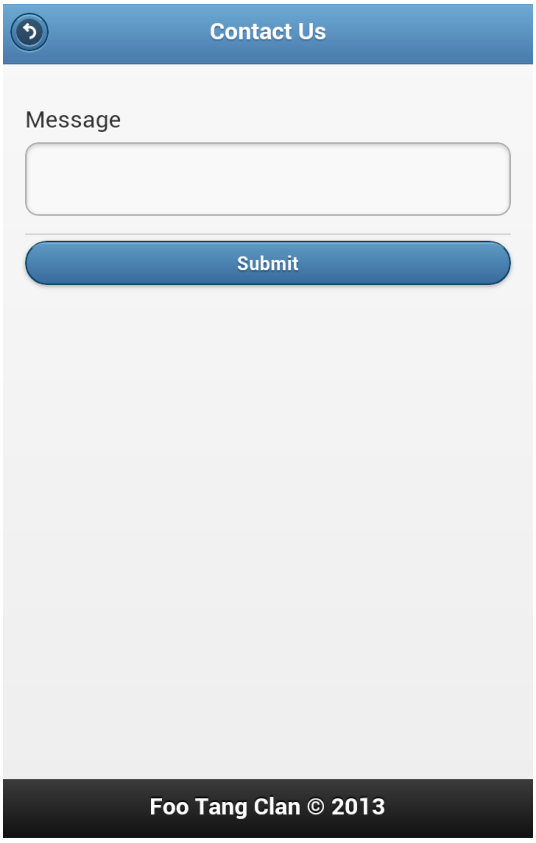
2.6 Contact

Overview:

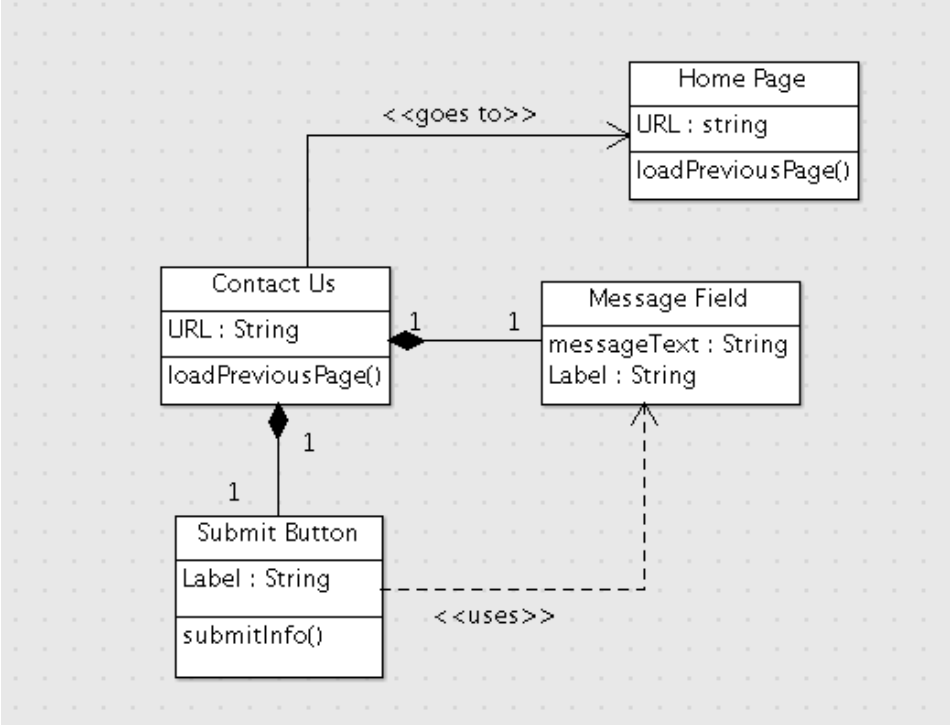
From this page a user will be able to:

- Send a new message to their property manager.

UI Mock-Up



UML Diagram



2.7 Settings

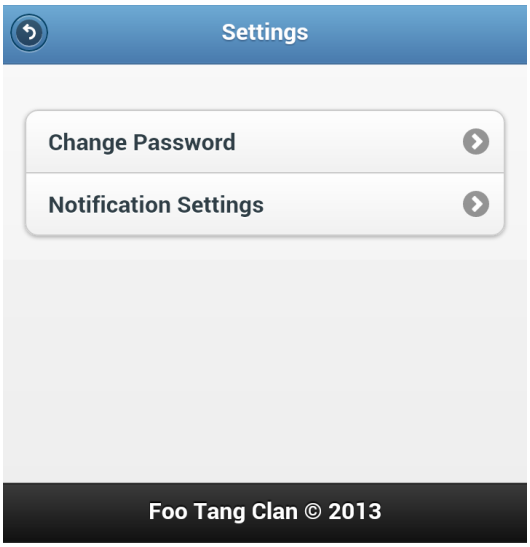
2.7.1 Main Settings

Overview:

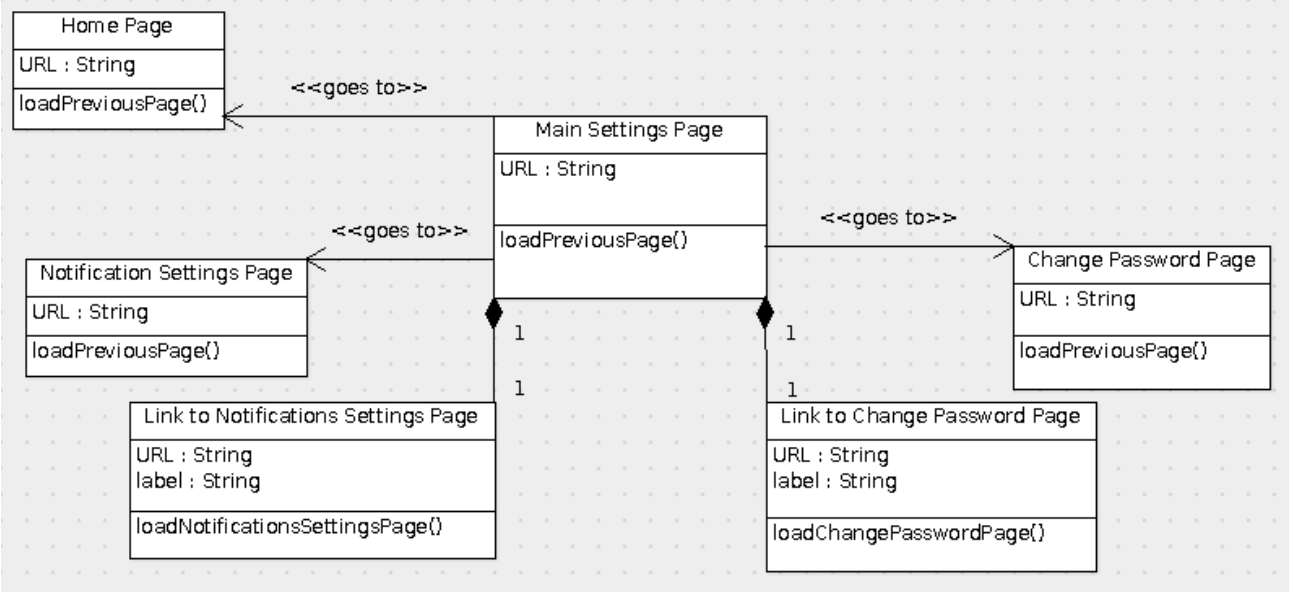
From this page a user will be able to:

- Navigate to other settings page.

UI Mock-Up



UML Diagram



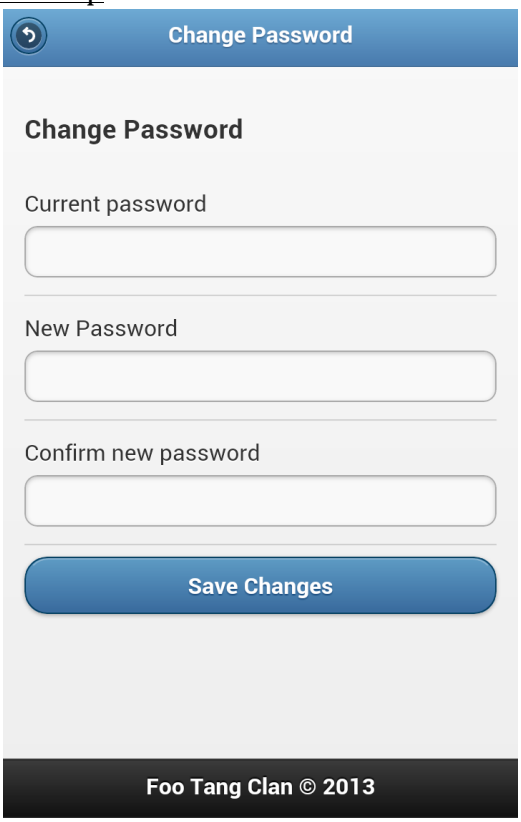
2.7.2 Change Password

Overview:

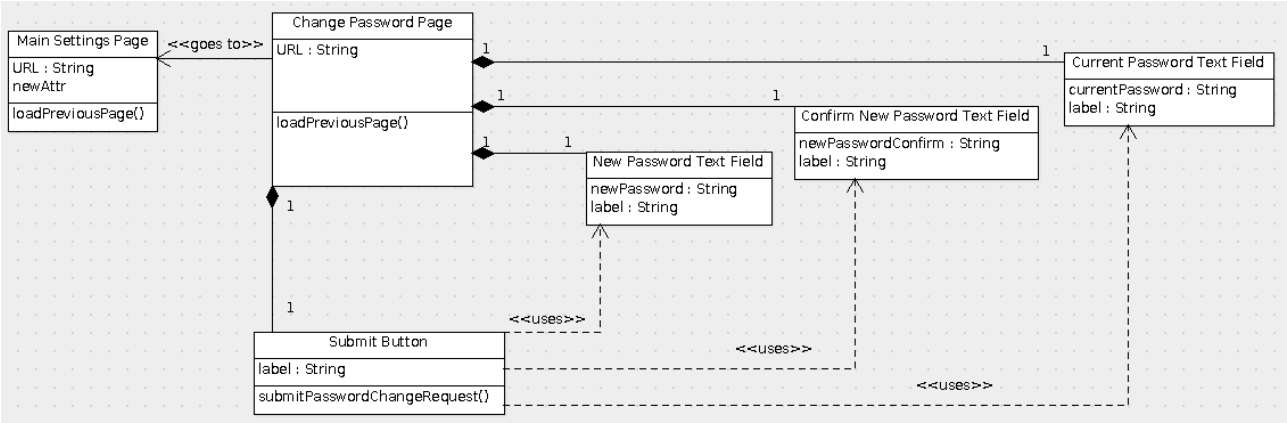
From this page a user will be able to:

- Change their password to login to the app.

UI Mock-Up



UML Diagram



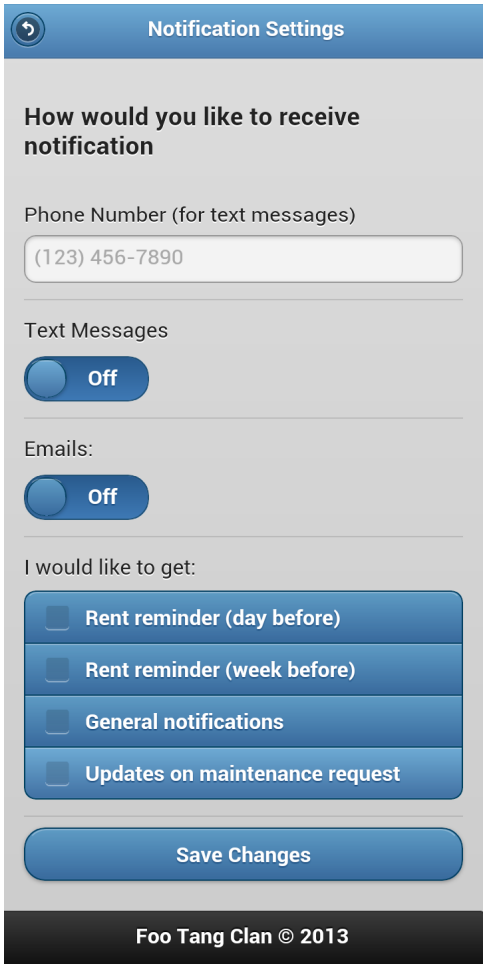
2.7.3 Notification Settings

Overview:

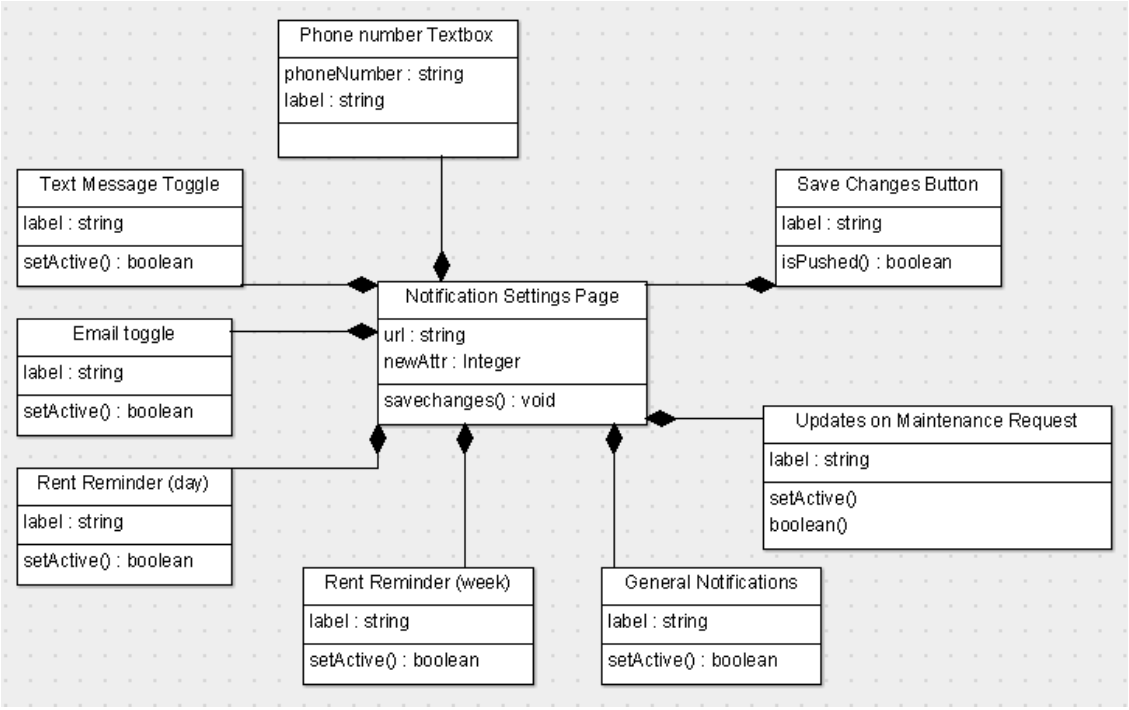
From this page a user will be able to:

- Change how they receive alerts from the app.
- Change what types of alerts they receive.

UI Mock-Up



UML Diagram


Phone number Textbox
phoneNumber : string
label : string
Text Message Toggle
label : string
setActive() : boolean
Save Changes Button
label : string
isPushed() : boolean
Notification Settings Page
url : string
newAttr : Integer
savechanges() : void
Email toggle
label : string
setActive() : boolean
Updates on Maintenance Request
label : string
setActive()
boolean()
Rent Reminder (day)
label : string
setActive() : boolean
Rent Reminder (week)
label : string
setActive() : boolean
General Notifications
label : string
setActive() : boolean

3 Testing

3.1 Logging in

Check correct credentials situation:

```
def loginAttempt(email, password)
    within 'div#login-form' do
        fill_in 'Login', with: email
        fill_in 'Password', with: password
        click_on 'Login'
    end
end
test "Check that correct credentials are accepted" do
visit "/login"
    loginAttempt('valid_user@email.com','secret')
    assert_equal current_path, user_home_path
end
```

Check incorrect credentials situation

```
test "Check that incorrect credentials are not accepted" do
    visit "/login"
    loginAttempt('invalid_user@email.com','random')
    assert_equal current_path, user_sign_in_path
end
```

Check and deal with blank credentials

```
test "Check that blank credentials doesnt break anything" do
    visit "/login"
    loginAttempt('','')
    assert_equal current_path, user_sign_in_path
end
```

Checks if links are properly working

```
test "Check request account link" do
    visit "/login"
    click_link 'Request Account'
    assert_equal current_path, user_sign_up_path
end
```

Check if the user is already logged in and redirect correctly

```
test "Check that user cant log in twice" do
    visit "/login"
    loginAttempt('valid_user@email.com','secret')
    visit "/login"
    assert_equal current_path, user_home_path
end
```

3.1.1 Request Account

Check if the user already logged in and redirects correctly

```
test “User already logged in” do
    visit “/login”
    loginAttempt(‘valid_user@email.com’,’secret’)
    visit “/users/sign_up”
    assert_equal current_path, user_home_path
click_link
```

Checks if links are properly working

```
test “Check request account link” do
    visit “/login”
    click_link ‘Request Account’
    assert_equal current_path, user_sign_up_path
end
```

3.1.2 Forgot Password

Check if the user is already logged in and redirect correctly

```
test “User already logged in” do
    visit “/login”
    loginAttempt(‘valid_user@email.com’,’secret’)
    click_link
end
```

Check if email exists

```
test “email exists”
    visit “/login”
    loginAttempt(‘valid_user@email.com’,’secret’)
    click_link
    assert_equal current_path, user_sign_up_path
end
```

Check if links are working properly

```
test “Check request account link” do
    visit “/login”
    click_link ‘Request Account’
    assert_equal current_path, user_sign_up_path
end
```

3.1.3 Request Unlock Instructions

Check if the user is already logged in and redirects correctly

```
test “User already logged in” do
    visit “/login”
    loginAttempt(‘valid_user@email.com’,’secret’)
    click_link
end
```

Check if user’s account is locked or not

```
test “Check account lock status” do
    visit “/account_lock”
    attemptUnlock(user@email.com)
end
```

Check links on the unlock page to see if they function.

```
test “Check unlock page link” do
    visit “/login”
    click_link ‘Unlock Instructions’
    assert_equal current_path, user_unlock_path
end
```

3.2 Paying Rent

3.2.1 Submit Payment

Check the amount field

```
test “Check value in amount field” do
    visit “/home/pay_rent”
    submitPayment(0,”2/2/13”,”firstName”,”lastName”,12345678910,123456789,”
        checking”,”off”)
    assert pay_rent_path reloaded with Amount Field Error.
end
```

Check account information fields(account/routing fields)

The form can’t be submitted with empty fields because of html.

Check if an account type is selected

```
submitPayment(0,”2/2/13”,”firstName”,”lastName”,12345678910,123456789,””,”off”)
assert pay_rent_path reloaded with Account Type Error.
```

Check name fields

The form can’t be submitted with the name fields empty.

3.2.2 Payment History

Check if another tenant in the same unit can see the payment history

```
Test “Unit wide payment history”
    loginAttempt(“name1@email.com,password1)
    visit “/home/payment_history”
    loginAttempt(“name2@email.com,password2)
    visit “home/payment_history”
    assert_equal payment_log_path1, payment_log_path2
end
```

Check if save payment info works

```
Test “Check save payment functions” do
    visit “/home/rent_payment”
    submitPayment(800.00,”2/2/13”,”firstName”,”lastName”,12345678910,12345678
      9,”checking”,”on”)
    visit “/home/rent_payment”
    within ‘div#payment_form input#user[first_name]’ do
        assert_has_content?(firstName) # just assume that if we saved the first name
    end                                # correctly, then we saved everything else
end
```

3.3 Maintenance

3.3.1 New Request

Can create new request and store in database

```
Test “Create new maintenance request” do
    visit “/home/maintenance_request”
    submitMaintenanceRequest(“Request Description String”,”yes”)
    assert_not_nil new_maintenance_request, “new request not created”
end
```

3.3.2 Log

Check if logs are created

```
def getRequest
    @maintenanceRequest = MaintenanceRequest.find(1)
end
test “Check if logs are created” do
    getRequest
    if (@maintenanceRequest)
        visit “/home/maintenance_log”
        within ‘div#maintenance_log li’ do
            assert_has_content?(@maintenanceRequest.content)
        end
    end
end
```

3.4 Contact

Since this page will need to interface with the AppFolio Property Manager application on the backend (code which has not yet been disclosed to us), we cannot come up with any test code for this page at this time. However, the testing that would need to occur for this page would be to ensure that once a user sends a message, their respective property manager receives this message in their log.