Detailed Design for Quarters

Team 6

James Anthony (anthonjb)
Wenqiang Chen (chenw25)
Carolyn Chong (chongce)
Kevin Ly (lyk2)

January 18, 2016

Contents

List of Figures

Revision History

Date	Comments
January 5, 2016	Revision 0
January 10, 2016	Revision 1
January 11, 2016	Revision 2

1 User Interface Elements Descriptions

A description of the user interface (UI) design of Quarters is presented here. This section is divided into subsections of the UI navigation flow and the major UI elements. Each UI element is explained with the support of Norman's design principles and illustrated with screen images from a mockup. Norman's principles of design are briefly stated to provide some background on the UI design principles of Quarters.

1.1 Navigation Flow

See Figure 1. Users are directed to the landing page. From there new users can sign up or returning users can login. After signing up, the new user will be prompted to either join a house or create a house in order to access the rest of the application. After logging in, the returning user will be directed to the main page of the application, called the Bulletin Board. The other pages of the application, including House Management and User Settings can be accessed from any page via the top navigation bar while House Settings, Calendar, Messages, Finances and Maintenance can all be navigated to from any page using the side navigation bar.

If a user is a member of multiple houses, they can switch between houses via House Management from the top navigation bar.

In-app notifications are visible on the top navigation bar on all pages, thus no navigation is required to access notifications.

The user can access user profile settings and log out via the top navigation bar.

1.2 Norman's Design Principles

Don Norman lists six principles to support software usability in his book *The Design of Everyday Things* [1]. These principles are visibility, affordances, mappings, constraints, feedback, and the conceptual model. Together they guide the design decisions of the UI of Quarters.

1. Visibility conveys to the user their current state and possible actions [1].
2. An affordance is a visual attribute of an object or control that helps the user determine how the object or control can be used [1].
3. A mapping is the relationship between a control and its effects [1].
4. Constraints limit the ways in which an object or control can be used by the user [1].
5. Feedback is when the user is informed of the results of their actions and indicates what actions can be taken next [1].

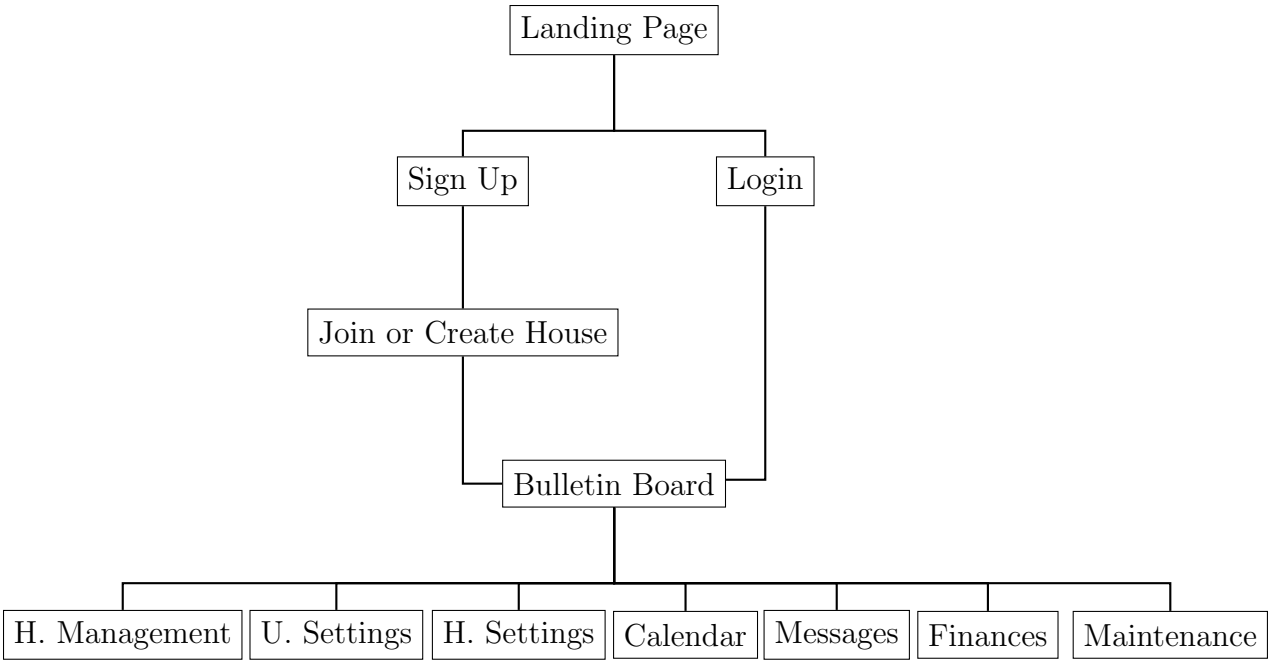

Figure 1: Diagram of Quarters UI Navigation Flow

6. The conceptual model is a physical understanding of an interface or interaction technique based on real-world experience [1].

1.3 Landing Page

The purpose of the landing page is to provide information about the web application's features, and to allow the potential user to sign up. There is also a login button for returning users. Both the signup and login buttons are designed to afford clicking and both are placed in convenient and visible locations. The landing page is designed to capture a potential user's interest in the web application by making it appear modern, secure, easy to use, and beneficial through the use of fonts, colors, and layout. A big image covers the landing page that is meant to evoke feelings for a desire of that lifestyle the user could have if they joined Quarters. See Figure 2.

1.4 Sign Up

The sign up page is designed to have strong visibility. A new user should be able to know what to do just by looking at the page. The page is simple, straightforward and uncluttered to allow for a quick process. The empty fields afford input. The sign up button is an example of a constraint because it is disabled until the user enters valid input. See Figure 3.

Figure 2: Screen image of Landing page

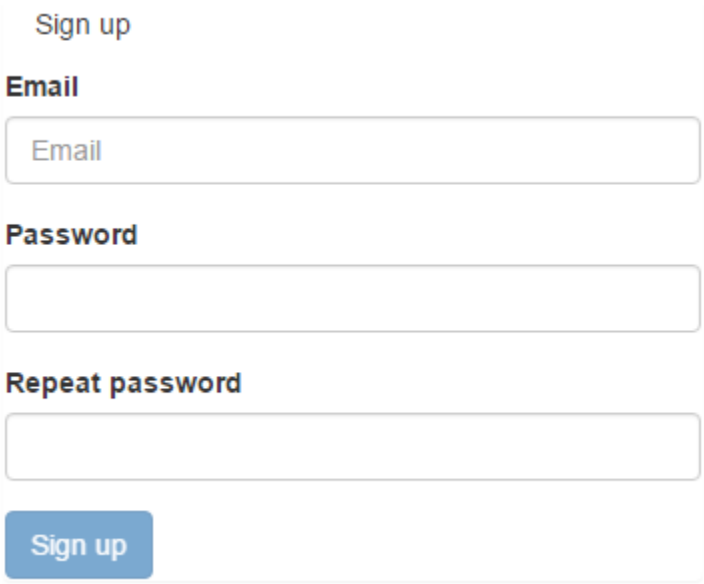

Figure 3: Screen image of Sign Up page

1.5 Join or Create House

See 1.9.

1.6 Login

Designed similarly to the sign up page, the login page has strong visibility. The empty fields afford input. The login button is disabled until the user enters valid input.

1.7 Navigation bars

Every page of the application has the same layout for consistency. Each page consists of a top navigation bar and a left navigation bar, and the remaining space is devoted to hold the content of that specific page. The structure of the navigation bars are consistent throughout the application to improve learnability. Fonts, colors and layouts are consistent, as well. The whole user experience, from signing up to logging out, is responsive. This means that regardless of the screen window size, the user will be able to access all functionality of the application. For example, the navigation bars collapse to a toggle menu in smaller screen sizes to ensure the premium screen space is occupied mainly by content with which the user will spend the most time interacting. The intuitive iconography of the navigation bars clearly indicate there are more options hiding deeper down, and thus, demonstrates strong visibility.

1.8 Bulletin Board

The bulletin board (bulletin for convenience) is the main page of the application. From here, every other page can be accessed via the navigation bar. Every user's bulletin is personalized based on their activity and their house's activity. Posts take up the majority of the screen space and are listed in chronological order to support the conceptual model. Each post is listed with a corresponding icon that symbolizes the type of activity. A large field is positioned at the top of the bulletin to allow users to share content quickly. When the user clicks on the input field it is highlighted to give feedback to the user that the field is active and input is allowed. See Figure 4.

1.9 House Management

New users are automatically directed to this page after signing up. Returning users can access House Management via the top navigation bar by clicking on the house address. This page is where users can join new houses, create new houses, or add members to a house. This page should be designed with strong feedback including whether the user's actions were successful and what actions they should take next.

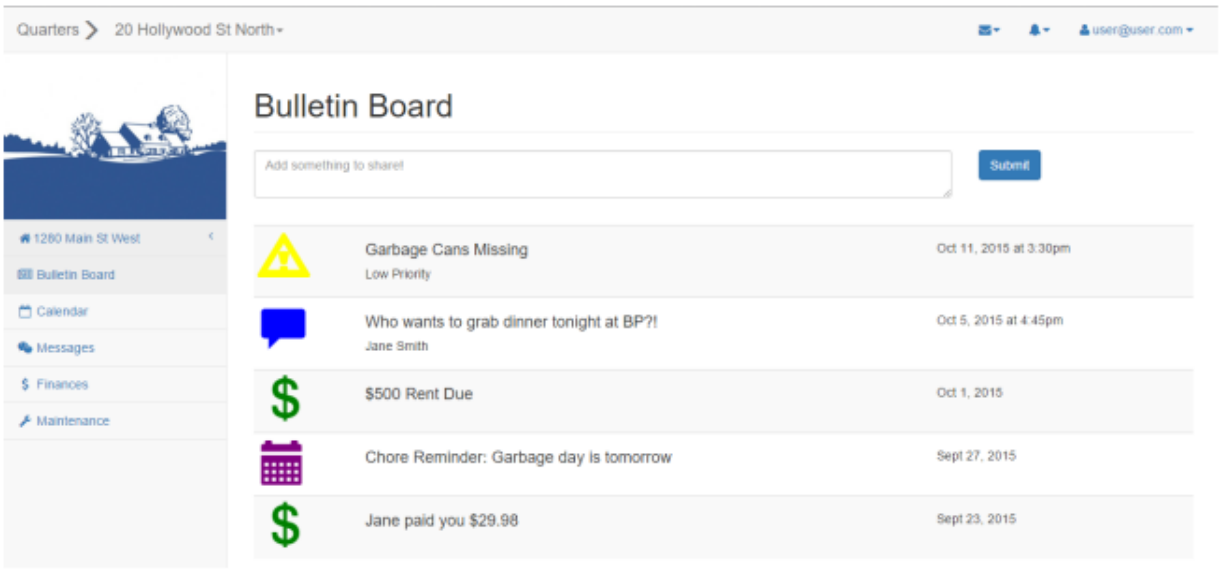

Figure 4: Screen image of Bulletin Board page

1.10 User Settings

Users can change their display name on this page. This page is accessible by clicking on the user's email in the top navigation bar.

1.11 House Settings

House Settings is accessible via the side navigation bar. Important documents uploaded by members of the house are included here. Documents should be presented following the conceptual model of a stack of "papers", that is, a scaled-down screenshot of each document should be laid out in a grid-like structure.

1.12 Calendar

The Calendar resembles the UI of Google Calendar because it is a widely used calendar that allows for some familiarity. The checkbox group showcasing different calendars improves visibility of the calendar state. The Calendar should be designed so that users can click on a day or established event to make changes, as opposed to only allowing users to make changes using the buttons positioned above the Calendar. In this way, the natural mapping allows users to interact with the Calendar in an intuitive manner. See Figure 5.

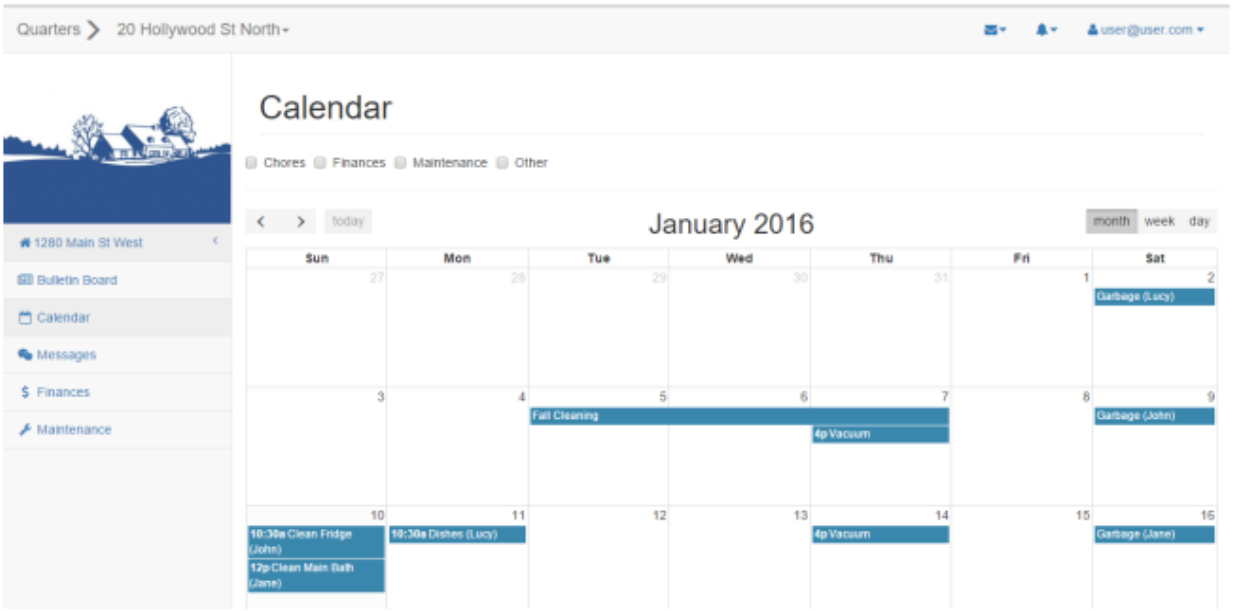

Figure 5: Screen image of Calendar page

1.13 Messages

A basic chat history of direct messages between two users is displayed here. The messages section is separate from the online users section. An image of the user accompanies each message. Again, a simple, uncluttered interface is important to ensure the user experience is as quick as possible. See Figure 6.

1.14 Finances

The main focus of this page is a table displaying all bills owed and owing. See Figure 7.

1.15 Maintenance

This section holds a list of maintenance tickets in chronological order. Each ticket is accompanied by a corresponding icon to symbolize the type of ticket. Colors are used to differentiate the priority levels of each ticket. Each ticket is displayed in its own horizontal panel. See Figure 8.

1.16 Notifications

Notifications are accessed via the bell icon displayed in the top navigation bar. Notifications are viewed by clicking on the dropdown menu. Visibility is improved because the navigation bar is displayed on every page, therefore, notifications are displayed on every page. As a result, the user is always aware of application updates. Feedback is improved because the

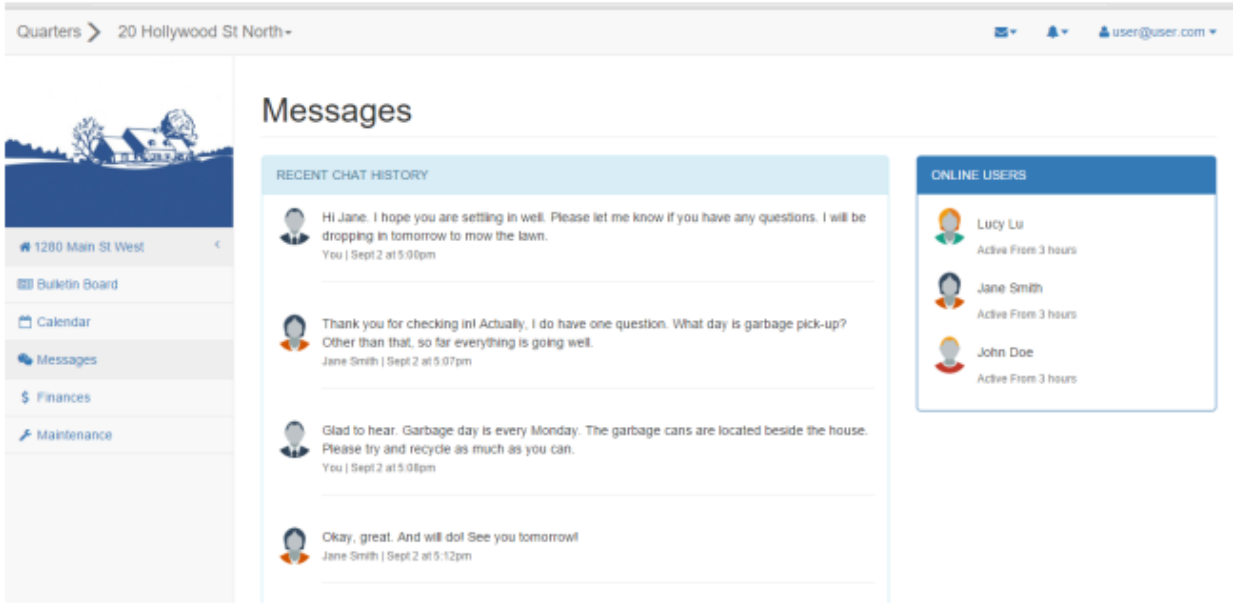

Figure 6: Screen image of Messages page

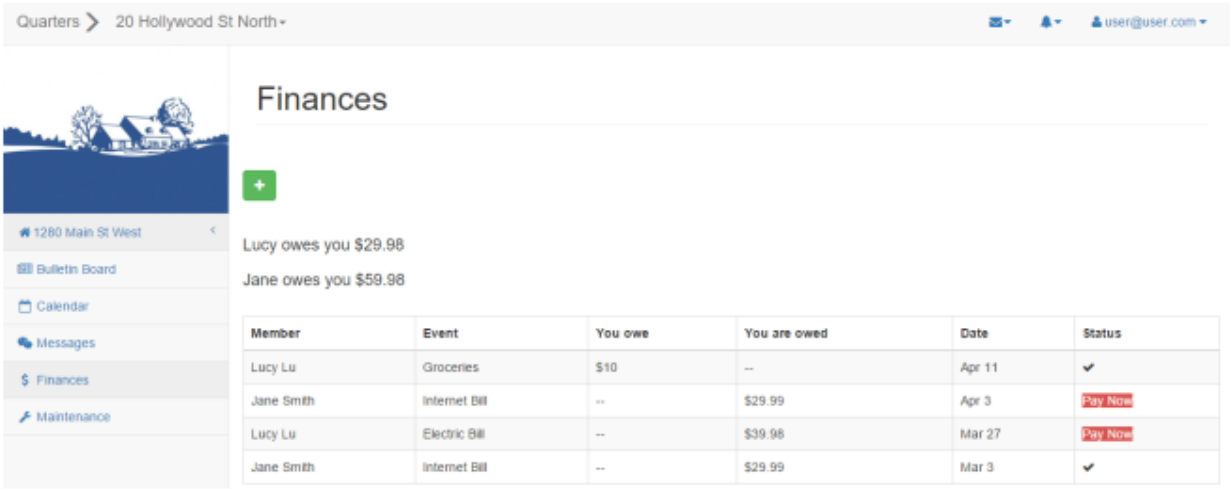

Figure 7: Screen image of Finances page

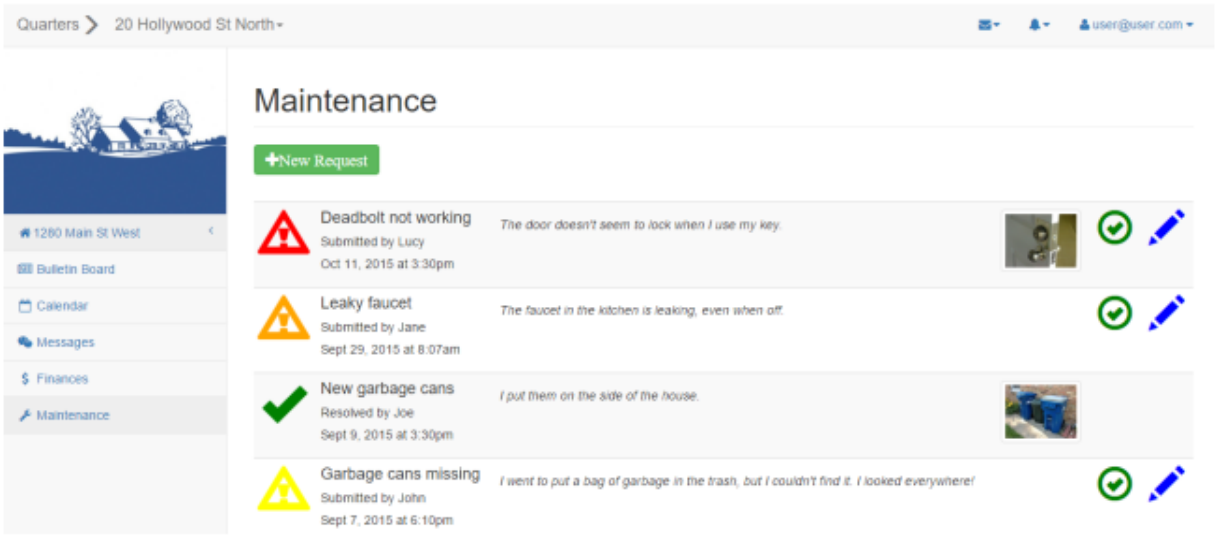

Figure 8: Screen image of Maintenance page

notification icon, the bell, changes color upon a new notification, and then reverts back to its default color to inform the user that they have read the notification.

2 Module Decomposition

2.1 Quarters application

- builds the entire application on the web.
- access different modules to build certain features.
- **startApp (feature)**
  - builds the view.
  - builds the specific feature provided by the argument.
- **logout()**
  - clears server sided session data.
  - redirects to home page.

2.2 Sign Up

- Allows user to register for an account on Quarters.
- **validateEmail(email)**

  - validate email against database for uniqueness
  - outputs error if emails is invalid

- **signup(email,pass)**
  - send sign up request with user credential to server to be added to database.
  - redirect user to main application after sign up is successful.

2.3 Login

- Allows user to login into Quarters application.
- **login(email,pass)**
  - send login request with user credential to server to be verified.
  - outputs appropriate error message if credential is incorrect.
  - redirect user to main application if login is successful.

2.4 House Management

- conceals information with regard to housing and roles.
- **houseLookup(userID)**
  - returns a list of houseIDs a user is a part of.
- **houseInfo(houseID)**
  - returns information about a house as a JSON object.
  - description, users in the house and their roles.
- **joinHouse(userID, houseID)**
  - userID is a part of the house at the corresponding houseID.
- **leaveHouse(userID, houseID)**
  - userID leaves the house at houseID.
- **updateHouseInformation(userID, houseID)**
  - checks if userID has permission to update houseID.

2.5 Bulletin Board

- conceals information in regards to bulletin board.
- **generateBoard(userID, houseID)**
  - gets all post relevant for the user.
- **displayPost(obj)**
  - generates corresponding html for the post and enables all functionality.
- **createPost(obj)**
  - adds post information to the database.
  - calls display post to display information.
- **deletePost(postID)**
  - removes postID from database.
- **addComment(obj, postID)**
  - adds comment to postID.
- **removeComment(commentID)**
  - removes comment from a post within a database.

2.6 Calendar

- conceals information in regards to the calendar
- **generateCalendar()**
  - creates empty calendar html.
- **populateCalendar(userID, HouseID)**
  - populates calendar widgets with events from the database.
- **createEvent(userId, HouseID, Obj)**
  - creates event with data constructed in the object argument.
  - data is pushed to the database.

2.7 Financial

- conceals information in regards to the financial system
- **getFinancialInfo(userid, houseID)**
  - get all financial information for userid at houseID that has not been completed.
- **display([])**
  - genereates html and creates proper event bindings for each financial item in the argument array.
  - bind click event on each financial module, that opens a window.
- **update(fID, userID)**
  - updates information in regards to the financial object by userID.
  - includes comments, details, and status.
- **newFinancial(userid, houseID, {})**
  - creates a new finance entry in the database and updates the view to reflect it.
  - information will be based throught the object argument.

2.8 Ticketing System

- contains all the methods used to display and manage the ticketing system.
- **getTicketing(houseID)**
  - get all ticket information for userid at houseID.
- **display([])**
  - genereates html and creates proper event bindings for each financial item in the argument array.
  - bind click event on each ticket module, that opens a window.
- **update(ticketID, userID, houseID)**
  - updates information in regards to the ticket object by userID.
  - includes comments, details, and status.
- **newTicket(userid, houseID, {})**
  - creates a new ticket entry in the database and updates the view to reflect it.
  - information will be based throught the object argument.

2.9 Notification

- incapsulates methods for notifications.
- **display([])**
  - generates html elements and event bindings for notifications.
  - populates the list using the array arugment.
- **getNotification(userID, HouseID)**
  - gets latest (15) notifications from the database.
- **updateStatus([notificationID])**
  - updates the statuses of the noticiations (seen, unseen).

3 Relational Database Structure

3.1 ER-Diagram

See Figure 9.

3.2 Table Descriptions

- **user** - Store user credentials(email, password, role type)
- **user_info** - Store basic user informations(name, birthday etc.)
- **house** - Store house informations (address, postal code, etc.)
- **maintenance_tickets** - Store maintenance tickets details
- **discussion** - Store discussion posts
- **priority_level** - Specify priority levels (low, medium, high)
- **role_type** - Specify role types (tenants, landlord)
- **calendar_events** - Store calender event details
- **notification** - Store notifications for users
- **notification_type** - Specify notification types (discussion, events, tickets, joined house, etc.)

4 Development Details

Languages of implementation

- NodeJS
- PostgreSQL
- Jade

Supporting frameworks

- Bootstrap
- ExpressJS

Supporting technology

- Ubuntu Server

References

[1] Don Norman. *The Design of Everyday Things - Revised and Expanded Edition*. Basic Books, New York, 10-131, 2013.

[Somewhere in this document you should mention the communication protocol used between the web interface and the server (most likely http?). You don't need to go into detail about it. —DS]

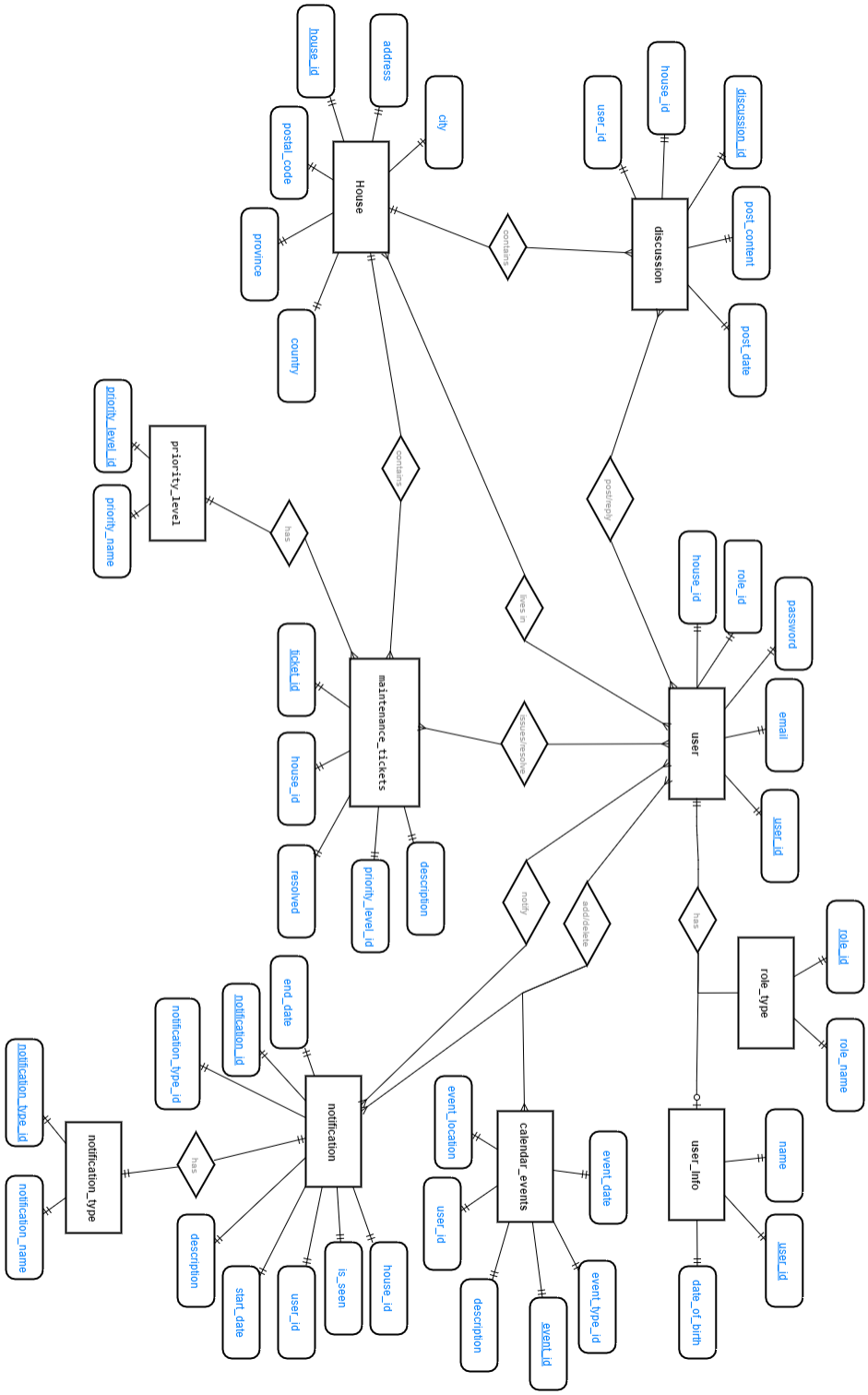

Figure 9: ER-Diagram