Molecular Playground Expansion

Team Dropout High Level Design Document

Table of Contents

1.0 Overview

1.1 System Overview

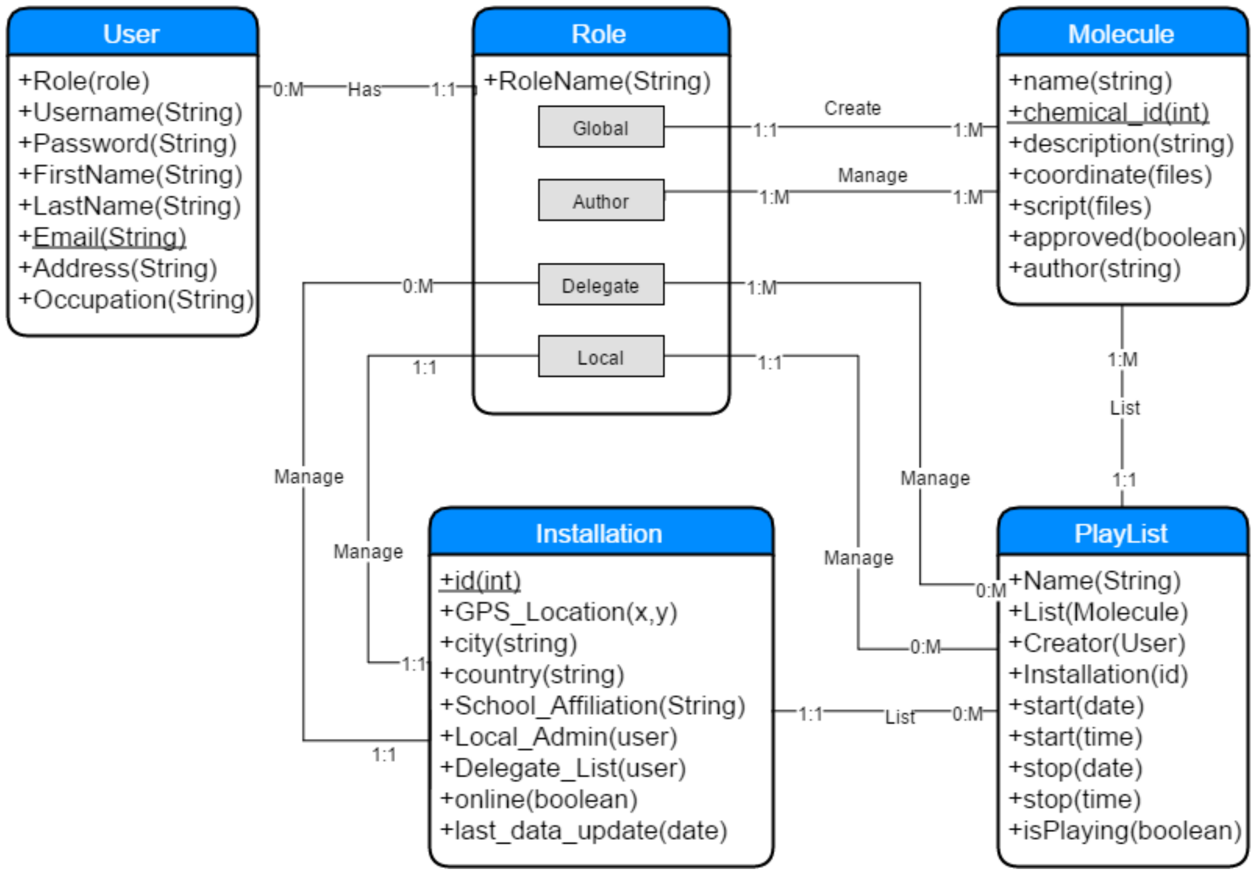
The goal of Molecular Playground Expansion is to centralize the data in the existing Molecular Playground installations all over the world. The typical workflow of the expansion would begin with an author uploading a molecule using the application. A global administrator would then approve the molecule submission. Once the molecule was approved, it would be pushed on to every local installation of Molecular Playground allowing local administrators and delegates to create playlists with this new molecule.

Additional functionality of the system includes account hierarchy controls. Global administrators will be able to manage all local administrators and also have an option to create a new local administrator account. Local administrators will be able to manage all the delegate accounts assigned to their local installation and similarly will have the ability to create new delegates. Author accounts are self created because they merely have the ability to upload content and request approval from the global administrator.

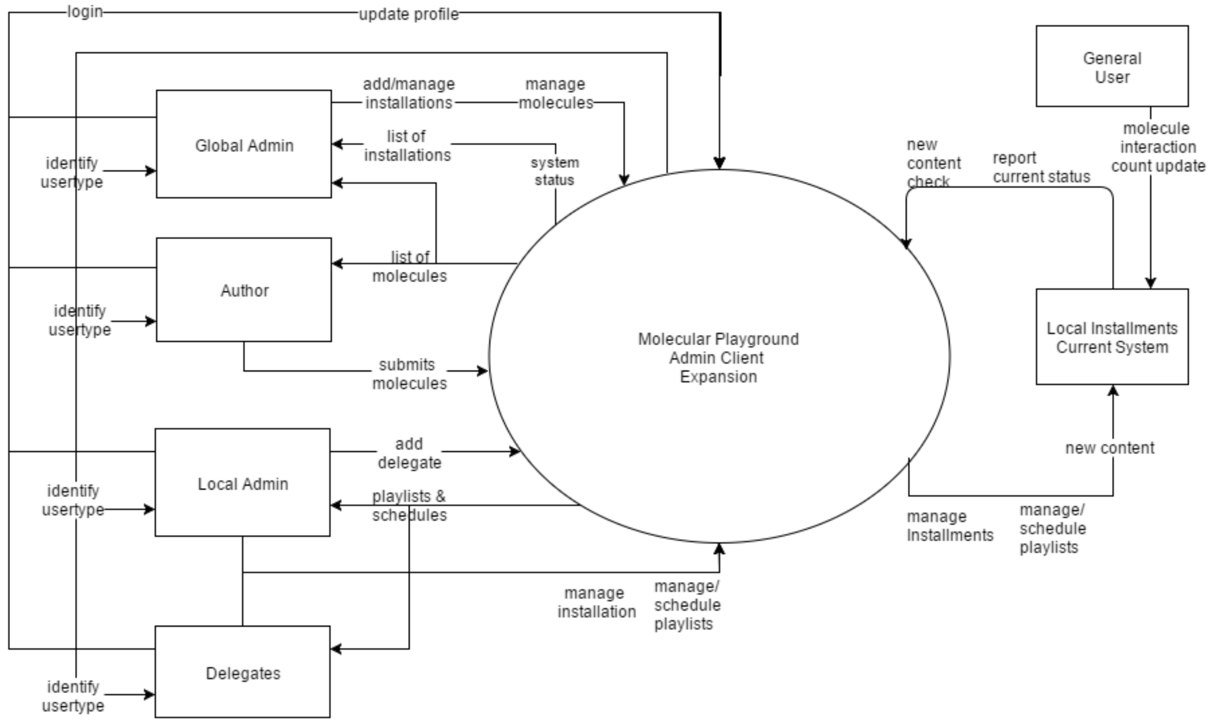
1.2 System Context

This section shows an ER diagram, context DFD, and level-1 DFD which help provide details about our system and how data flows through our system.

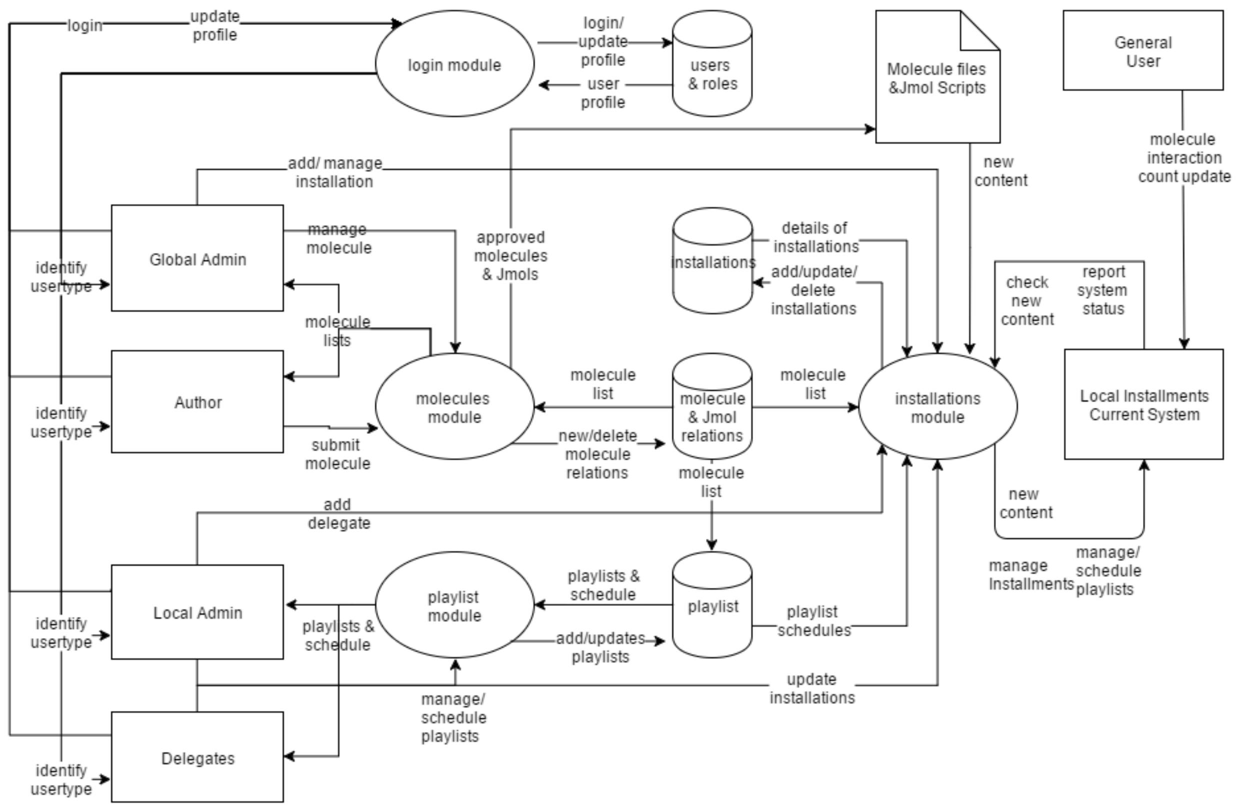
1.2.1 System ER diagram



1.2.2 Context Level DFD



1.2.3 System Level-1 DFD



1.3 Data Dictionary

All users:			***Global admin:***	
login	-username -password		add installments	-system id -GPS Location -city -country -School Affiliation -Local Admin -Delegate List -online -last data update
update profile	-username -changes		manage installments	-system id -changes
Identity usertype	-role		new content	-jmol file -time
author:			manage molecules	-molecule id -approval
submit molecules	-jmol file -molecule file -username		approve molecule	-molecule -approval

	-time			
Local Installments Current System:			***local admin:***	
new content check	-system id -current version #		create local admin	-username -email -password
new content	-data file -latest version #		update profile	-username -changes
manage installment	-installation param -start/stop		manage delegate	-username -changes
manage/ schedule playlists	-playlist -time		manage installment	installation param -start/stop
report system status	-online -played counts		manage/ schedule playlists	-playlist -time
installation details	-installation param		add delegate	-delegate info
update installations	-system id -changes		playlist schedule	-playlist -time
molecule list	-molecule		add/update playlist	-playlist -changes

1.4 Scope of Document

This document provides an overview of the system architecture including, an explanation and visual representation of the data flow, UML diagrams documenting system artifacts, details regarding the user interface design, and definitions of all system components.

1.5 Scope of Application

The Molecular Playground Expansion will be implementing the client-server structure. There will be a central database containing all of our data where clients of the system, both admins and authors, will be able to modify the database contents using our web application. The application will be adapting the model-view-controller (MVC) architectural pattern. The back-end of the application will be written using Javascript with a PostgreSQL database. These languages and patterns will make our system scalable and allow for future expansion including mobile application development.

1.6 Assumptions

1.6.1 Design Considerations

The site will be functional across all current major browsers (Chrome, Safari, Firefox, and Internet Explorer). The jMol application is currently functional on all modern operating systems and will still be compatible with them after our new features are added.

1.6.2 Constraints

The largest constraint on this project is the time we are allotted to work on it. Considering that the project is for a single class and we have less than a month until it is over, we must be realistic in the goals we set for the application concerning how much we can accomplish.

There are also constraints on the server hardware allotted to this system. Were we given unlimited resources, we would attempt to make the application highly resistant in software, but with the limited computation and storage power allocated, we do not have many options in this domain.

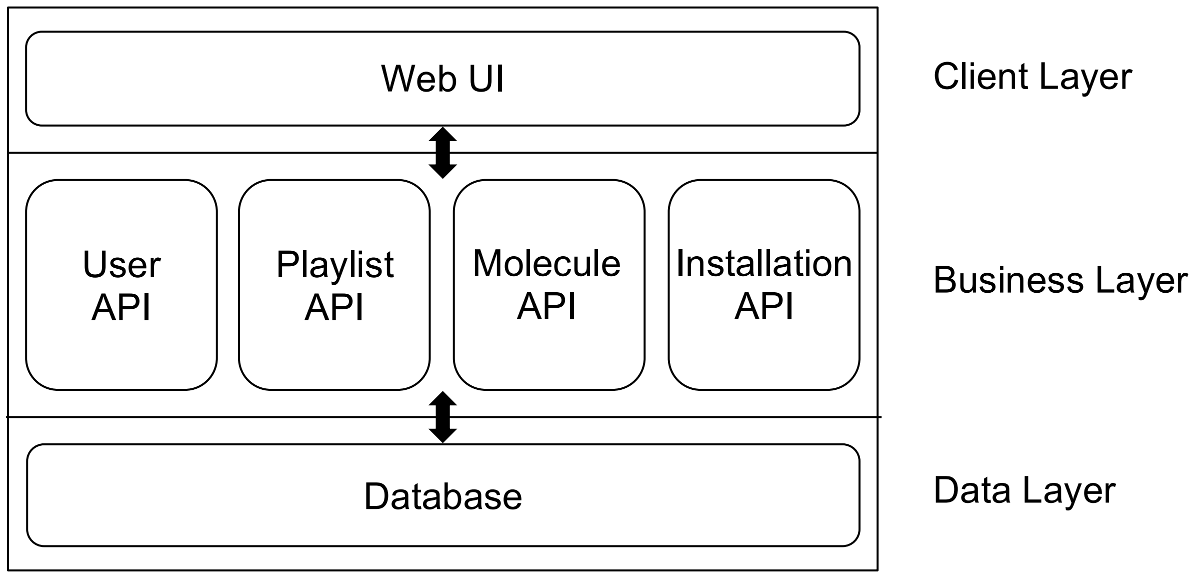
1.6.3 Risks

Since we are going to be handling sensitive user data (notably names and email addresses), we are inevitably at some small risk of cyberattack. It is unlikely we will be attacked considering that there will not (for the foreseeable future) be a user-base large enough to make an attack worth it. In addition, we are not storing personally identifiable information or payment information of any kind. Any system that stores client information is undeniably at this risk, however it is not something that we will be too worried about for the reasons just mentioned.

2.0 Architecture Design

This section outlines in further detail many of our high level implementation details such as the system layout, rationale for our architecture, packages, and User Interface Design. Since this is a web based application we will be using a client-server model to structure our code. More specifically we will be using Model-View-Controller to divide our code as seen in the UML diagram in section 2.2.

2.1 System Layout


Web UI
Client Layer
User API
Playlist API
Molecule API
Installation API
Business Layer
Database
Data Layer

2.1.1 System components

Component	**Layer**	**Description**
Web UI	Client	Consists of HTML pages and front-end Javascript code. This component renders HTML pages based on data sent from and transfer user input to Business Layer to be processed.
User API	Business	Processes requests from the Web UI component that involve user-related data functionality. This component requests user data from Data Layer and sends processed data to Client Layer. It is not exposed to the end-user. Sample functions: createUser (uid, username, password, email) authenticate (username, password) changePassword (uid, username)
Playlist API	Business	Processes requests from the Web UI component that involve playlist-related functionality. This component requests playlist data from Data Layer and sends processed data to Client Layer. Sample functions: savePlaylist (playlistId, moleculeIds) createPlaylist(playlistId, moleculeIds)
Molecule API	Business	Processes requests from the Web UI component that involve molecule-related functionality. This component requests molecule data from Data Layer and sends processed data to Client Layer. Sample functions: approveMolecule (moleculeId, files) editMolecule (moleculeId, files)
Installation API	Business	Processes requests from the Web UI component that involve installation-related functionality. This component requests installation data from Data Layer and sends processed data to Client Layer. Sample functions: addInstallation (installationId, latitude, longitude) removeInstallation (installationId)
Database	Data	A relational database that holds data about users, playlists, molecules and installations. This component provides a data API that supports CRUD (Create, Read, Update, Delete) operations.

2.1.2 Architecture Rationale

Our system architecture will be built in the RESTful style that is common for web applications, and will use HTTP requests (get/post/put/delete) for interaction between client and server. The main functions of our system are user actions such as creating/editing playlists, uploading new molecule data, and managing user accounts. These functions will be triggered by user action on the client side, processed by our API on the server side, and any changes to the system will be stored in a relational database. The logic of our application will follow the MVC (model, view, controller) paradigm.

Our system will be built in a modular style, where User, Playlist, Molecule, and Installation API components are organized separately as individual “models,” which are abstractions of relational database tables. For example, the User model “has many” Playlists, so multiple playlist objects stored in the database can have a relationship with a single user object where the user “owns” the playlists. In this way, the core user features of the system will easily be managed.

2.1.3 Packages

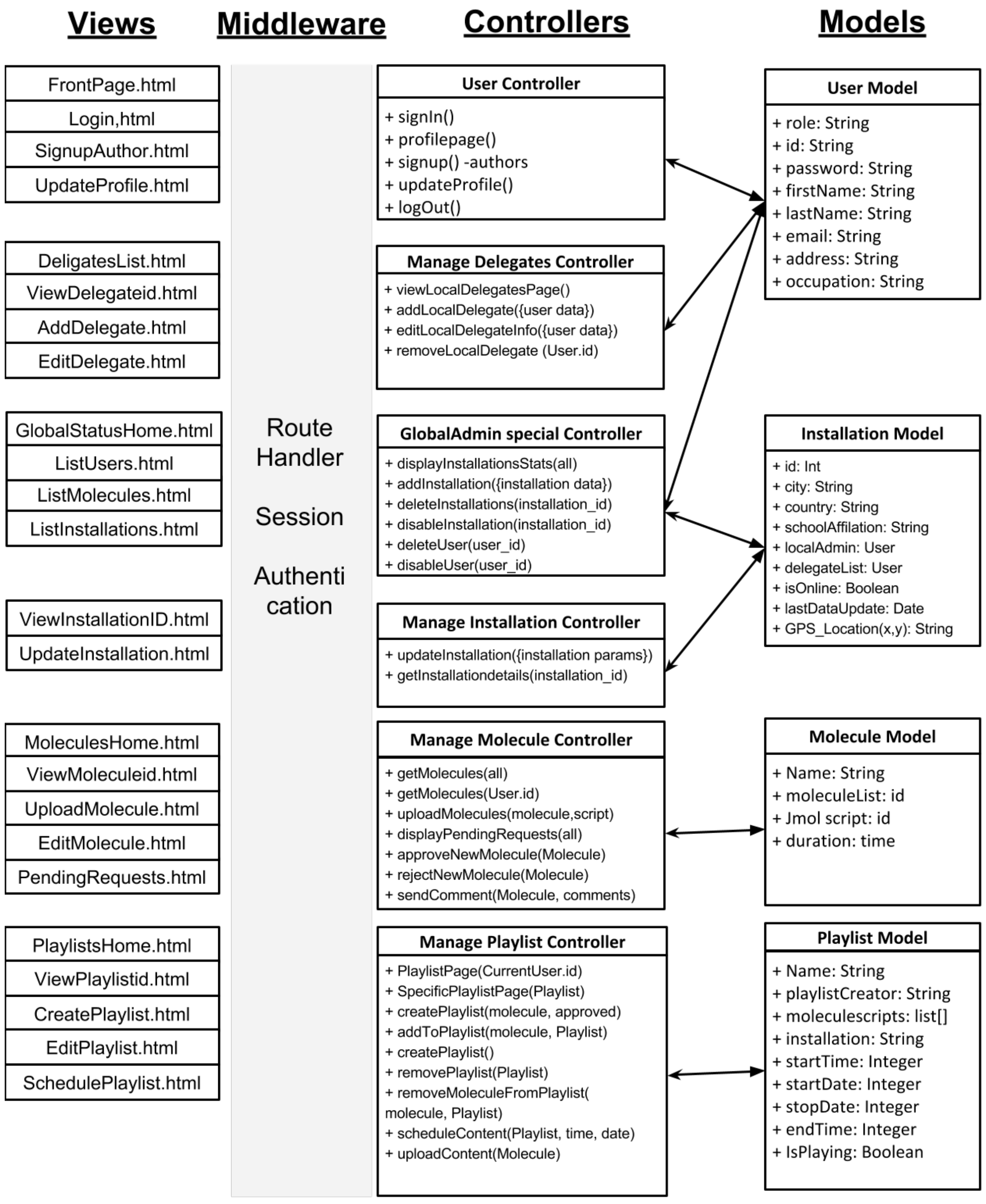
Front end:

- jQuery - HTML event handling, animation, and ajax processing
- AngularJS - interactive visualization with drag-and-drop and animations
- Bootstrap CSS - standard template for views and design
- Google Map - display installation location in google map
- NPM.js libraries:
    - express-login - simple express auth login
    - videojs-playlist-ui - angular display and ordering of playlists
    - angular-upload-file - angular display data in nice table format
    - table - angular display data in nice table format

Back end:

- aws/google auth - angular display data in nice table format
- aws/google storage - store sql database and file system for molecules
- cron - bash script for existing system push/pull data to server(aws/google storage)
- json - data format for passing info between server and client views
- NPM.js libraries
    - session - stores user login data use in controllers for routing
    - version - compares data version on server with pull requests from local installations
    - loopback - Node.js framework for RESTful API design
    - pg -PostgreSQL db integration with framework

2.2 UML Diagram



2.2.1 Model-View-Controller Overview

As seen from our UML diagram our system implements the software architectural pattern known as Model-View-Controller or MVC. Our views are HTML files that will render the pages on our website. Our middleware handles routing between our views and controller functions. It allows data to be passed from the views to the controllers and vice versa. Models within an MVC architectural pattern can be thought of as the underlying, logical structure of data in a software application and the high-level classes associated with it. The controllers then manipulate the data through the models and the changes are then seen in the views.

2.3 User Interface Design

The following table lists all pages accessible on our web application. For more detailed information, please refer to the use case listed for each page in the software requirement specifications document. The description field explains the function of the corresponding page. A page is defined as an interactive framework intended to help the user navigate to other pages and/or enter information into the web application and/or view information stored in the web application. Buttons on pages help the user achieve this purpose.

Page	Use Case	Description
Front Page		The applications landing page before a user logs into the system. The user can sign up to go to Sign Up page or login to go to the Login page. Figure 2.3.1
Home Page: Global Admin	1.1	Global administrator can see current installations and can choose various options in right-hand tab. Figure 2.3.2
Manage Jmol	1.2	Global administrator can edit the molecule list here. Molecules are listed in a drop down menu when the arrow is clicked. A molecule can be removed by selecting it and clicking remove. Figure 2.3.3
Manage Pending Requests	1.3	Global administrators can view and accept requests for new molecules here. Pending requests are listed in a drop down menu when the arrow is clicked. A request may be accepted or rejected by selecting it and clicking the corresponding button. Figure 2.3.4-1. When accepting or rejecting a request, the application will prompt for correctness. Figure 2.3.4-2
Manage Installations	1.5	Global administrator can either select add an installation to go to the Add New Installations page or choose to edit installation to go to Edit Installations page. Figure 2.3.5
Add new Installations	1.4	Global administrator can add a new installation by entering the needed information. Figure 2.3.6-1. When information

		is entered, the application prompts for correctness. Figure 2.3.6-2.
Edit Installations	1.5	Global administrator can edit existing installations. Figure 2.3.7-1. When information is entered, the application prompts for correctness. Figure 2.3.7-2.
Update Profile	5.2	Global administrator/local administrator/delegate can update their profile information. Figure 2.3.8.
Home Page: Local Admin/Delegate		Local administrator/delegate can choose corresponding options in right hand tab. Figure 2.3.9
Playlists	2.1	Local administrator/delegate can view a list of all current playlists in the installation. They can click a specific one to go to View Playlist page or click the create new playlist button to go to the Create Playlist page. Figure 2.3.10
Create Playlist	2.3	Local administrator/delegate creates a new playlist by providing a name, description, and searching for/adding molecules to it. Figure 2.3.11
View Playlist	2.2	Local administrator/delegate can view the content on the selected playlist or click edit to go to Edit Playlist page. Figure 2.3.12
Edit Playlist	2.4	Local administrator/delegate can change the name, description, or content involved in a specific playlist by removing unwanted molecules and searching for/adding new ones. Figure 2.3.13
Manage Delegate	3.1	Local administrator can view all delegates. Delegates can be clicked to go to Edit Delegate page or “Add delegate” button can be clicked to go to New Delegate page. Figure 2.3.14
Edit Delegate	3.3	Local administrators can change the username, email, or password involved in a delegate's account. Changes are saved by clicking Save Delegate. The delegate can also be

		removed by clicking remove delegate. Figure 2.3.15
New Delegate	3.2	Local Administrators can create a delegate by creating a username/password and entering an email address, and clicking Save Delegate. Figure 2.3.16
Schedule Content		Local Administrator/delegate can see all playlists on installation. Each playlist can be selected to bring user to Time Input page. Figure 2.3.17
Schedule Content-Time Input Prompt	2.5	Local Administrator/delegate can choose a starting date/time for the playlist selected. Figure 2.3.18
Login Page	5.1	A user logins to the system by providing a username and password. Figure 2.3.19
Sign Up page	4.1	A user can sign up to be an author on this page by providing a username, password, country, and occupation then selecting Continue. Figure 2.3.20
Upload Content	2.6	A user can upload a jmol file here in order to submit a potential new molecule for the system. Figure 2.3.21

2.3.1 Navigation

The application always opens to the landing page from where the user can either log into the system if they have an account, or sign up for an author account. Clicking login will bring the user to the login page. Once logged on, the user will a unique homepage corresponding to the user's role. If sign up is clicked the user will be brought to the author sign up page where they can create an author account.

After signing on, a **global administrator** will have the fIollowing options in a right hand tab.

- Home: brings user back to corresponding homepage. Figure 2.3.2
- Manage Content : brings user to manage content page. Figure 2.3.3, Figure 2.3.4
- Manage Installations: brings user to manage installations page. Figure 2.3.5
    - once on the Manage Installations page, New Installation can be clicked to bring user to the Add New Installation page. Figure 2.3.6
    - once on the Manage installations page, Edit Installation can be clicked to bring user to the Edit Installation page. Figure 2.3.7
- Update Profile: brings user to update information page. Figure 2.3.8
- Sign Out: brings user to front page. Figure 2.3.1

After signing on, a **local administrator** and **delegate** will have the following options in the right hand tab (manage delegate will not be an option for delegates).

- Home: brings user back to corresponding homepage. Figure 2.3.9
- Playlists: brings user to Playlists page. Figure 2.3.10
    - once on playlist page user can either click add new playlist which will bring the user to the Create Playlist page. Figure 2.3.11
    - once on playlist page, user can click specific specific playlist to go to view specific playlist page. Figure 2.3.12.
        - On view specific playlist page, the edit button can be selected to go to edit specific playlist page. Figure 2.3.13.
- Manage delegate: brings local administrator to the Manage Delegate page. Figure 2.3.14
    - A specific delegate can be clicked to go to Edit Delegate page. Figure 2.3.15
    - The Add Delegate button can be clicked to go to New Delegate page. Figure 2.3.16
- Update Profile: brings user to update information page. Figure 2.3.8
- Schedule Content: brings user to Schedule content page. Figure 2.3.17
    - A specific playlist can be clicked and this will initiate a scheduling prompt. Figure 2.3.18
- Upload: brings user to Upload Content page. Figure 2.3.21
- Sign Out: brings user to front page. Figure 2.3.1

After signing on, an **author** can upload content. Figure 2.3.21

2.3.2 UI Figures

Figure 2.3.1: Front Page

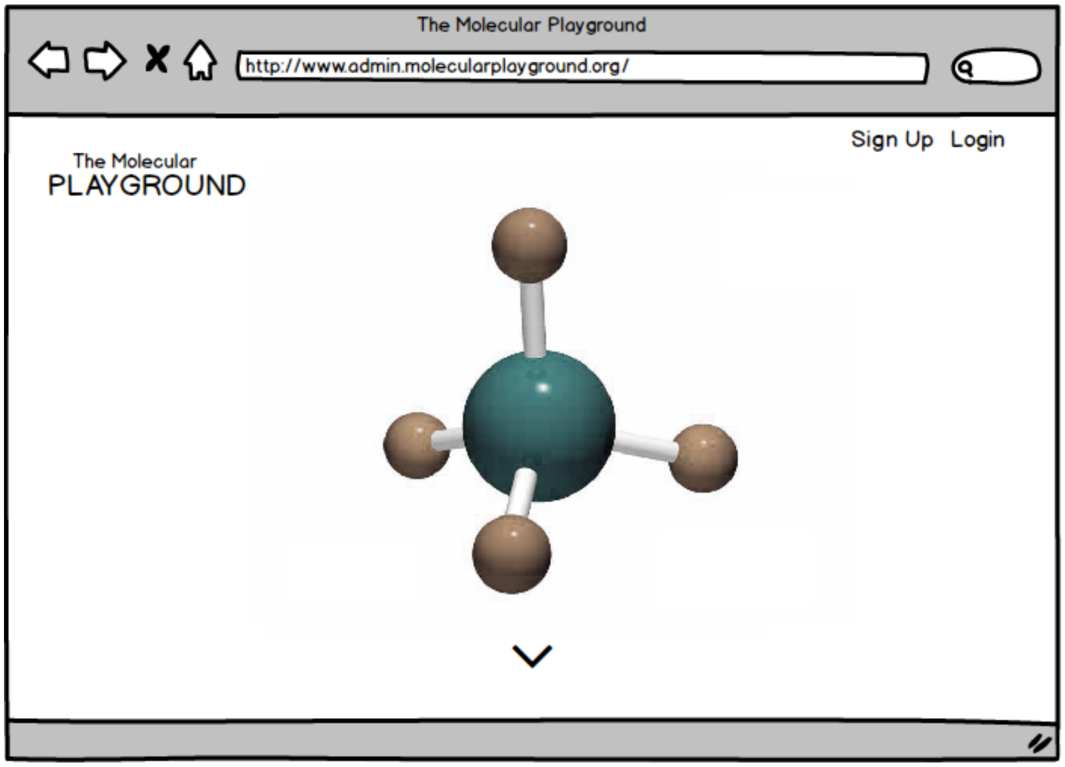


Figure 2.3.2: Home Page for Global Admin

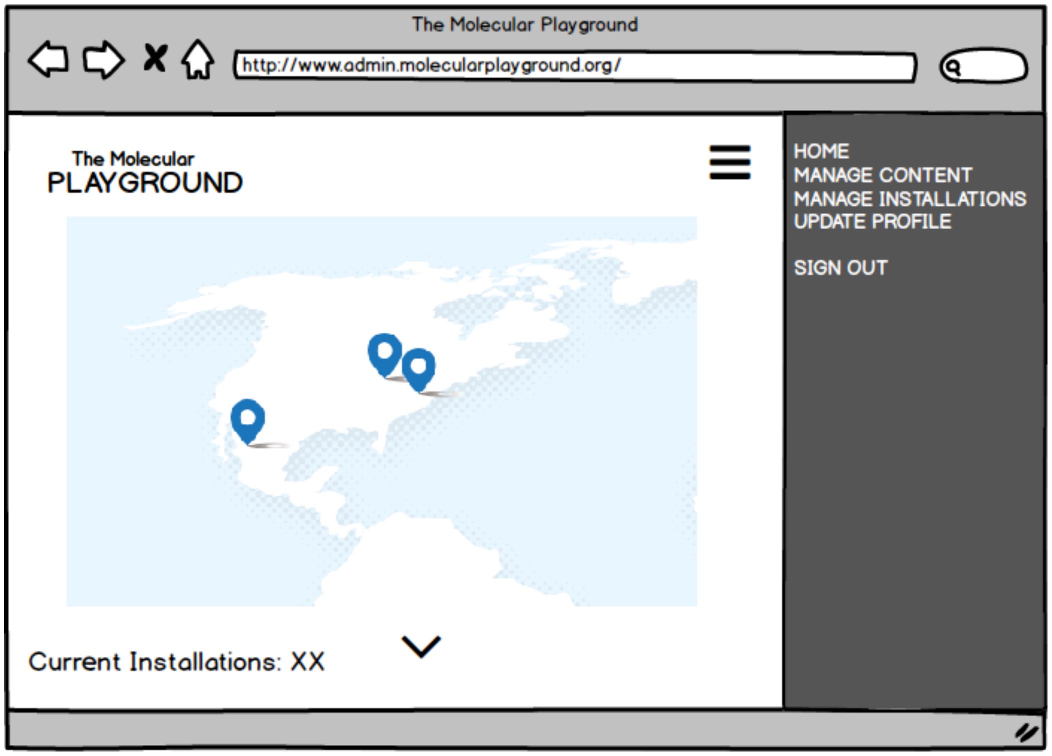


Figure 2.3.3: Global Administrator Manage Content - Manage jmol

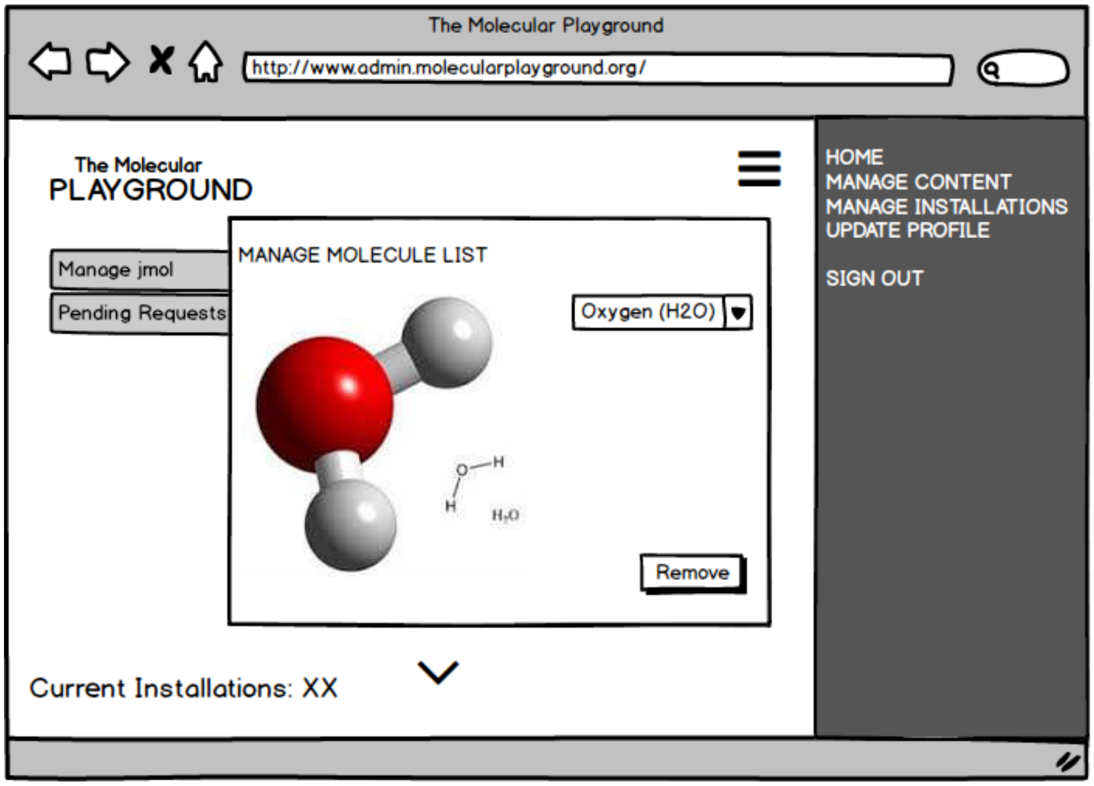


Figure 2.3.4-1: Global Administrator - Manage Content - Manage Pending Requests

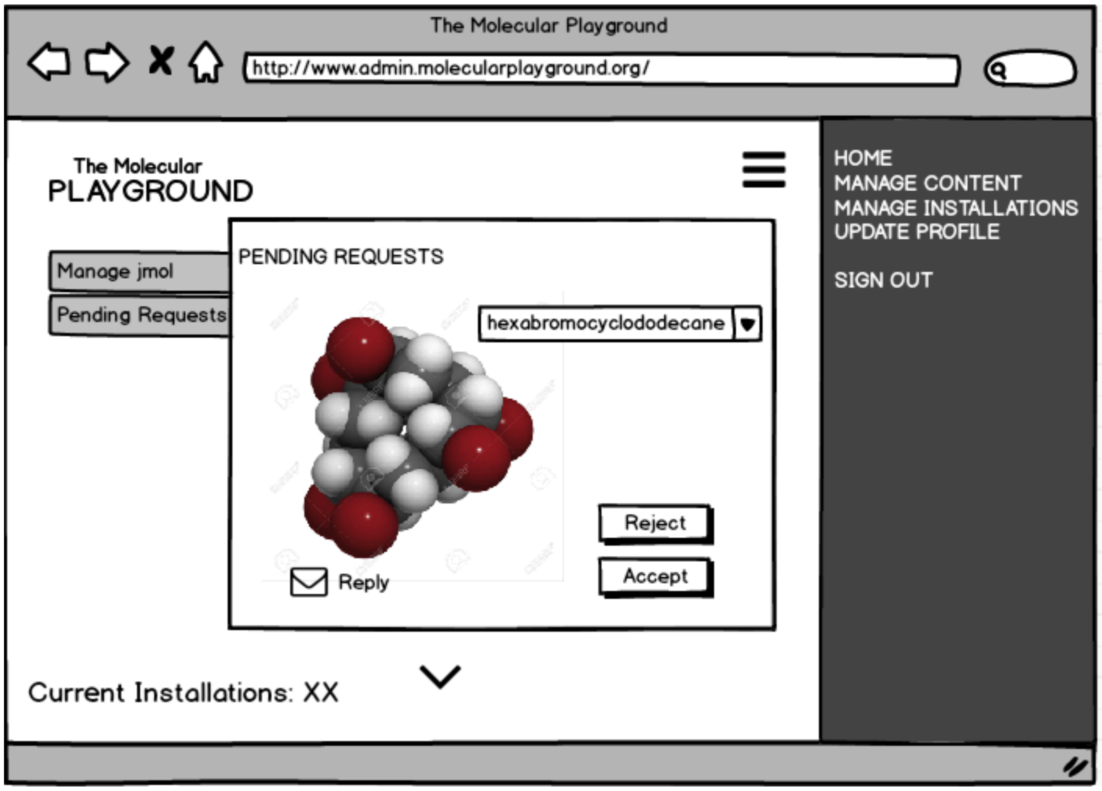
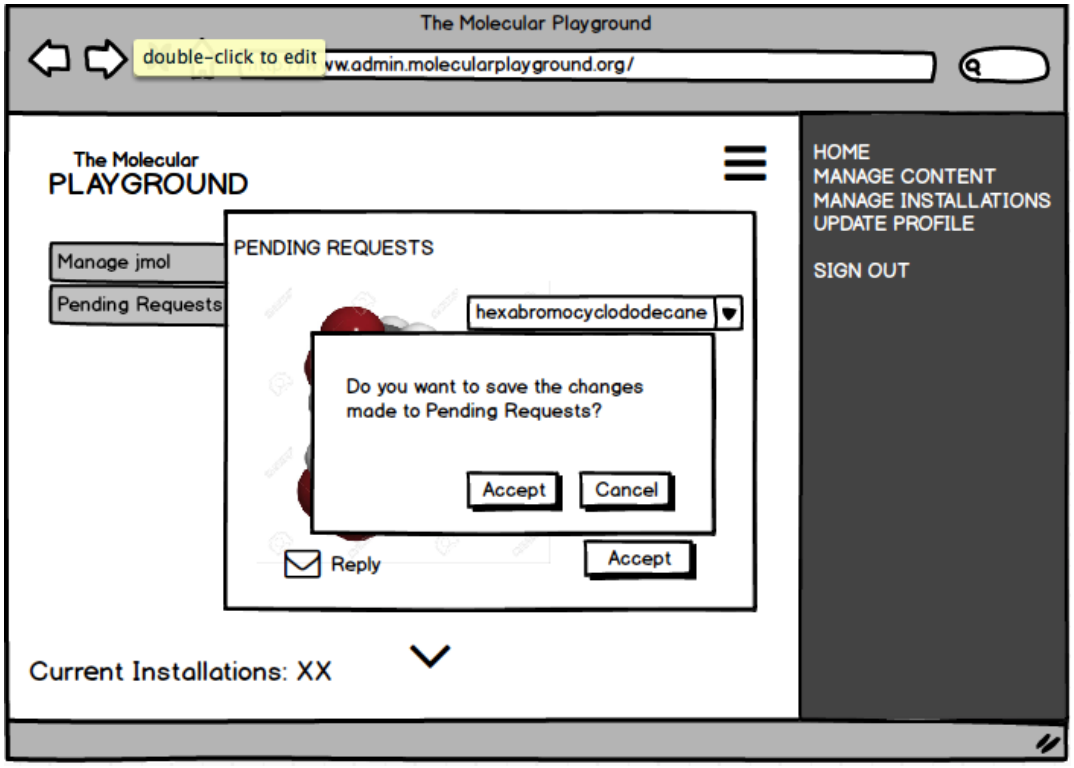


Figure 2.3.4-2: Global Administrator - Manage Content - Manage Pending Requests Prompt



Do you want to save the changes made to Pending Requests?

Figure 2.3.5: Global Administrator - Manage Installations

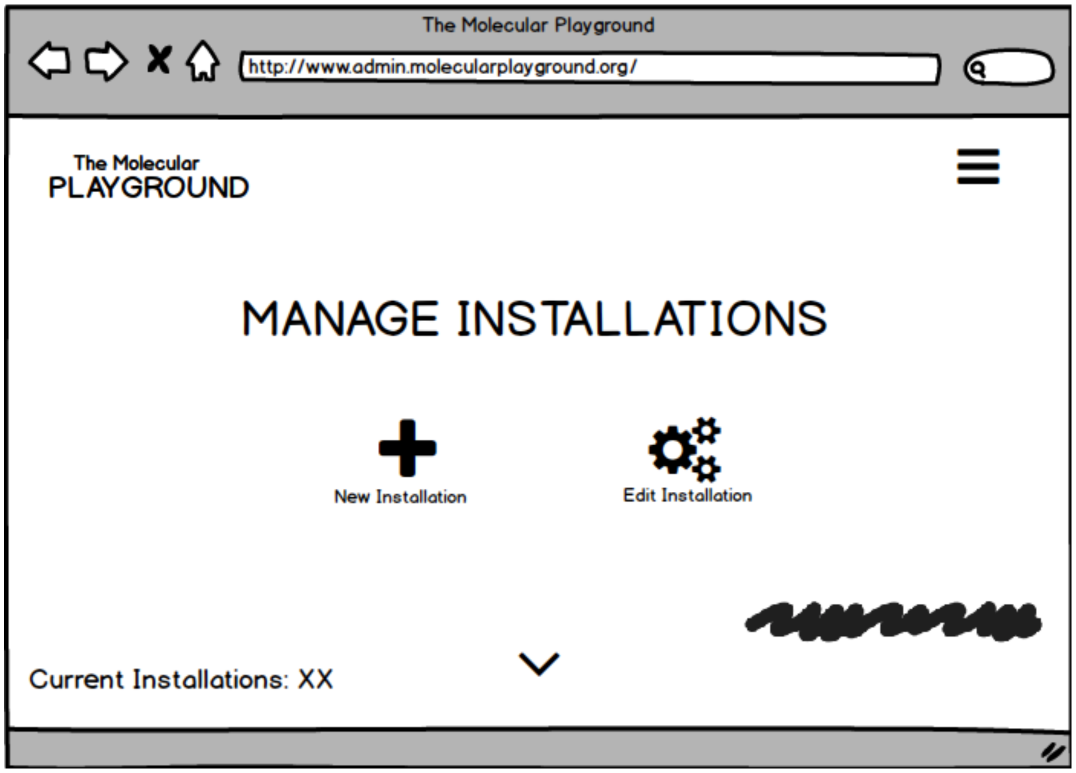
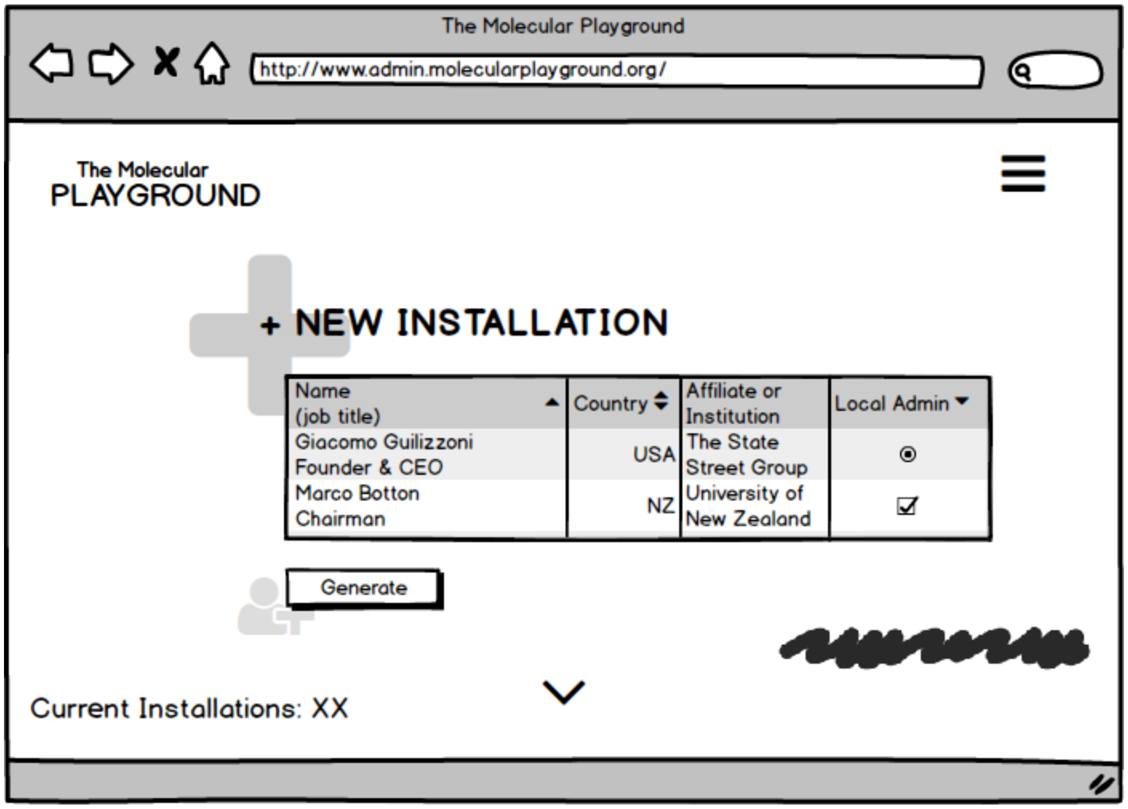
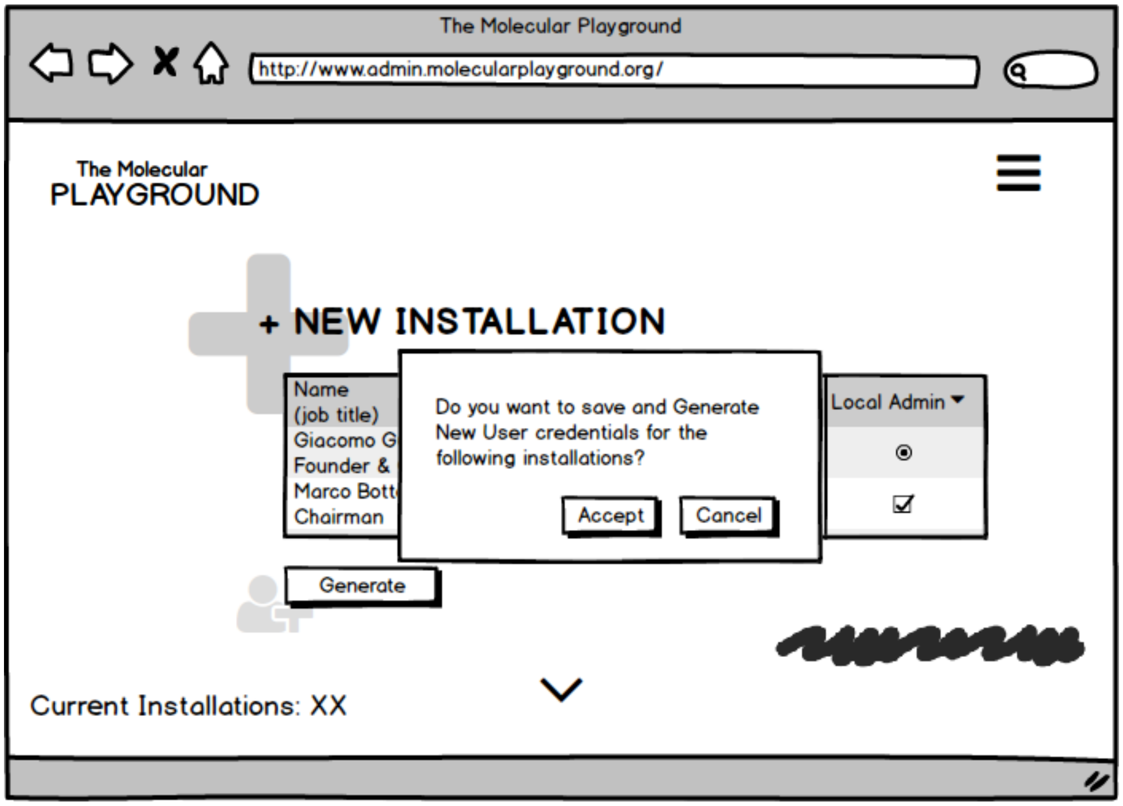


Figure 2.3.6-1: Global Administrator - Manage Installations - New Installation



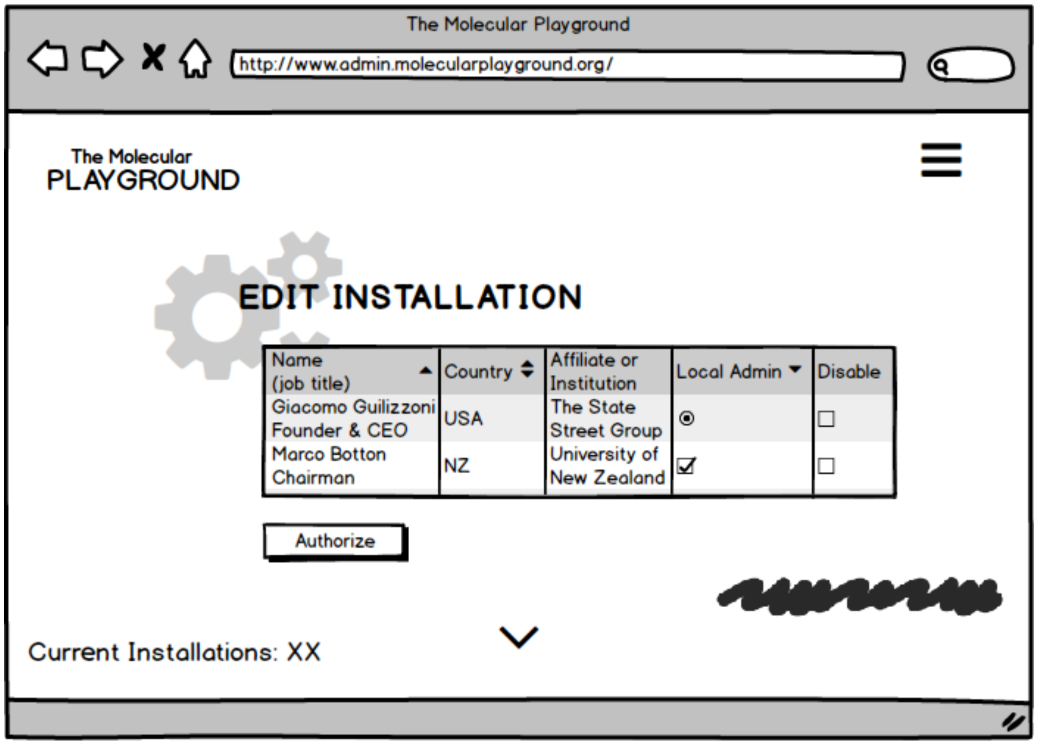
Name (job title)	Country	Affiliate or Institution	Local Admin
Giacomo Guilizzoni Founder & CEO	USA	The State Street Group	●
Marco Botton Chairman	NZ	University of New Zealand	✓

Figure 2.3.6-2: Global Administrator - Manage Installations - New Installation Prompt



Do you want to save and Generate New User credentials for the following installations?

Figure 2.3.7-1: Global Administrator - Manage Installations - Edit Installation

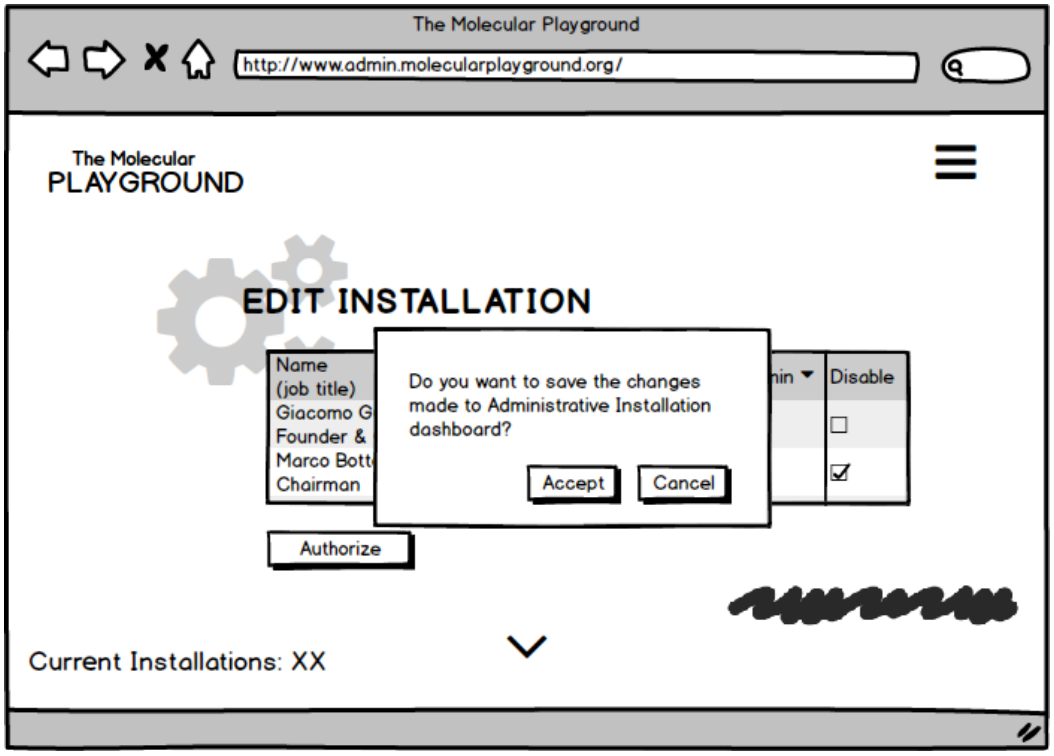


EDIT INSTALLATION

Name (job title)	Country	Affiliate or Institution	Local Admin	Disable
Giacomo Guilizzoni Founder & CEO	USA	The State Street Group	◉	☐
Marco Botton Chairman	NZ	University of New Zealand	☑	☐

Current Installations: XX

Figure 2.3.7-2: Global Administrator - Manage Installations - Edit Installation Prompt



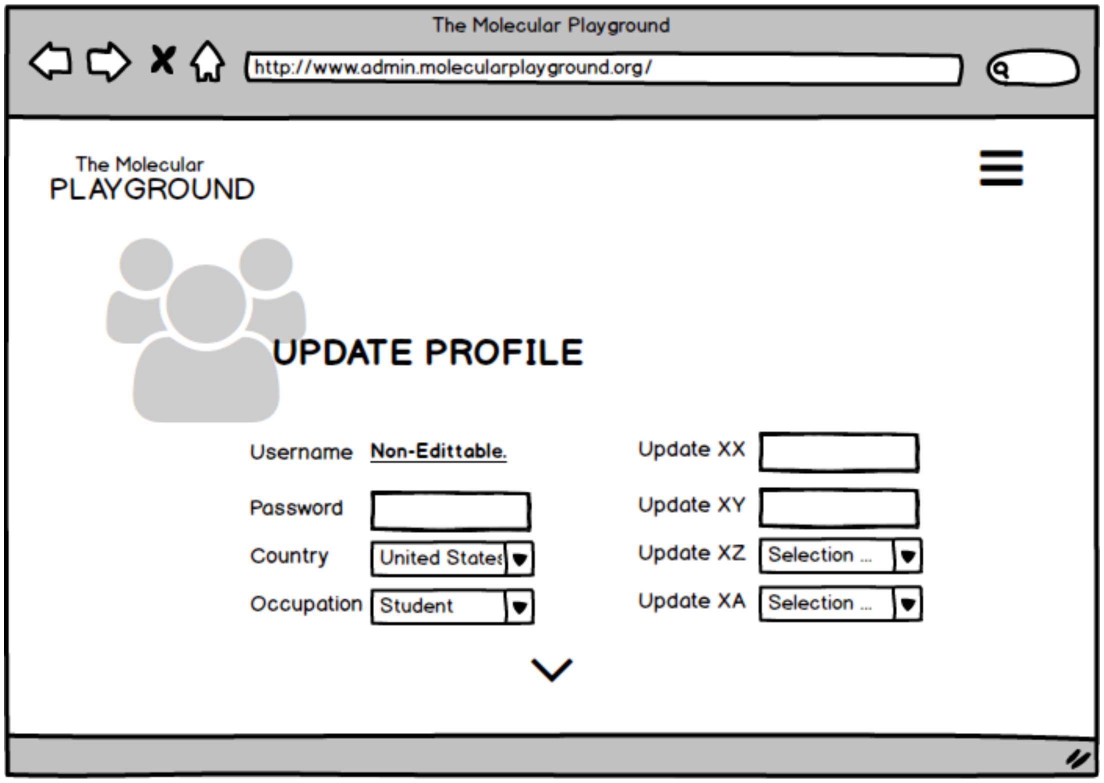
EDIT INSTALLATION

Name (job title)	in	Disable
Giacomo G Founder &		☐
Marco Bott Chairman		☑

Do you want to save the changes made to Administrative Installation dashboard?

Current Installations: XX

Figure 2.3.8: Update Profile



Username	Non-Edittable.	Update XX	
Password		Update XY	
Country	United States	Update XZ	Selection ...
Occupation	Student	Update XA	Selection ...

Figure 2.3.9: Home page for local administrator/delegate(excluding manage delegate option)

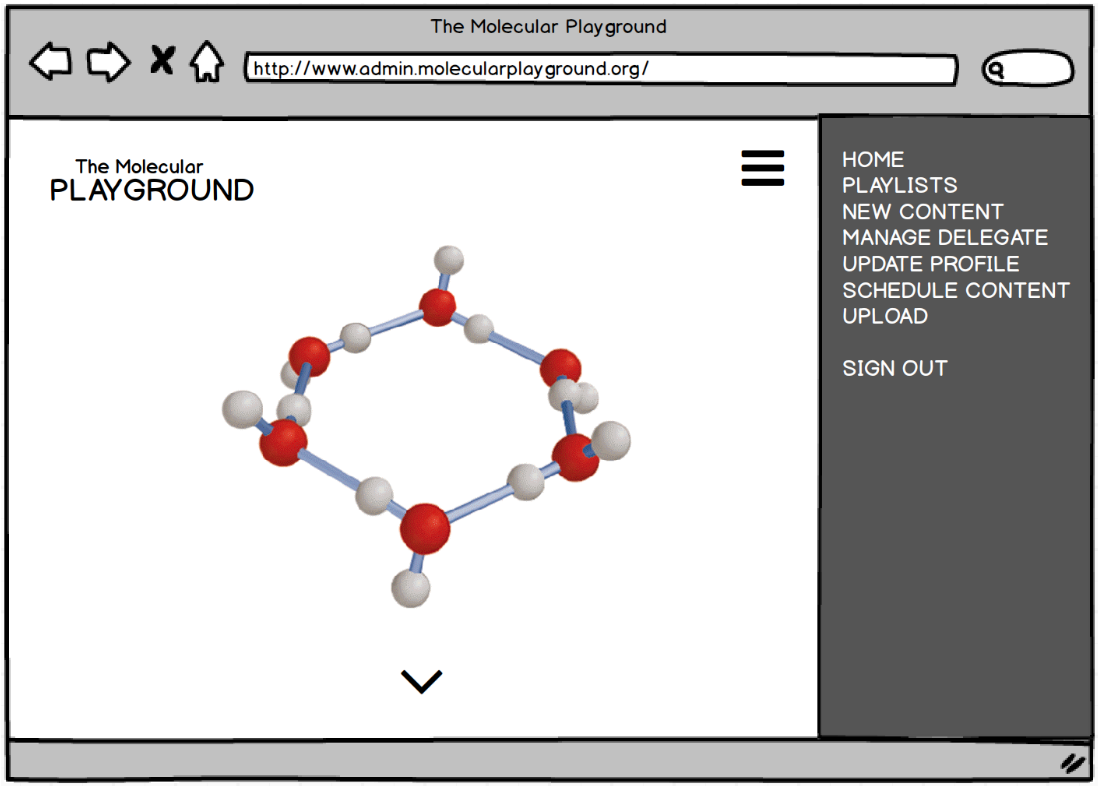
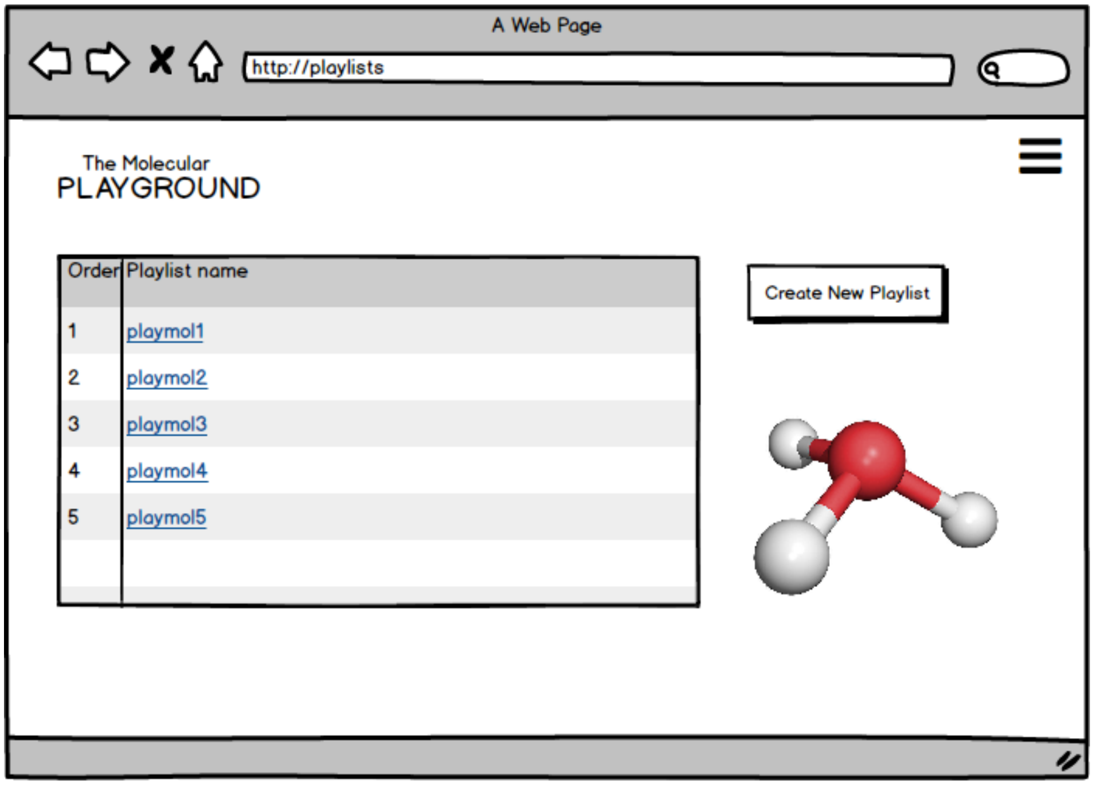
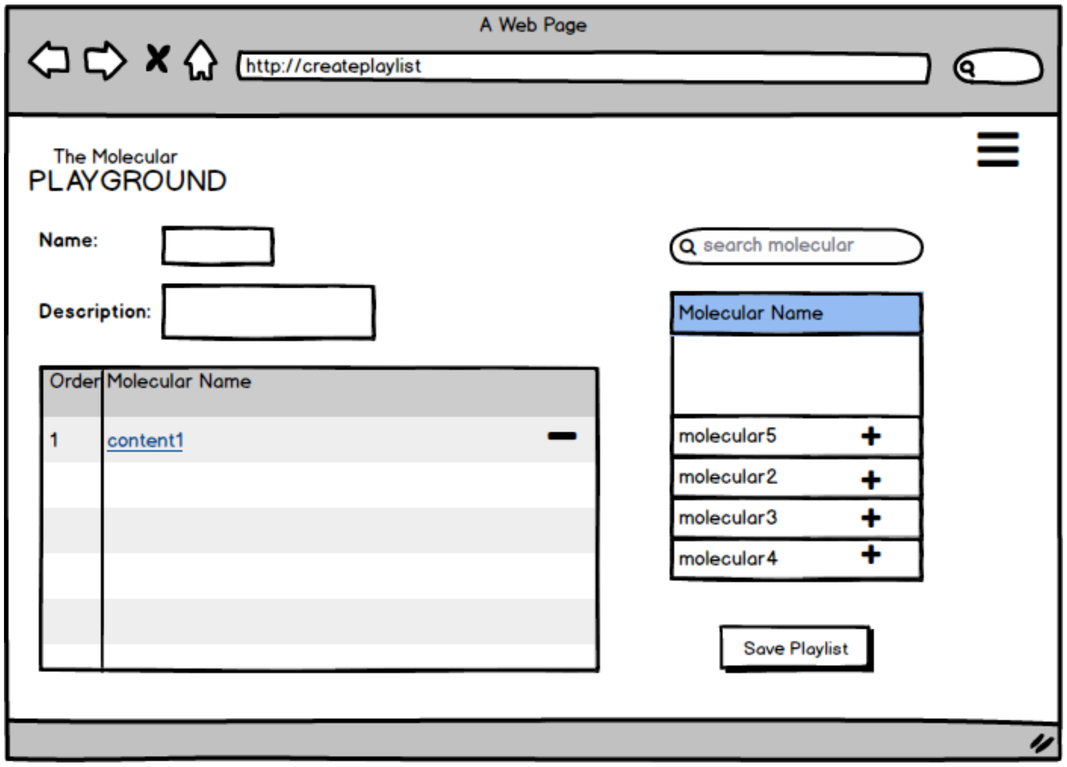


Figure 2.3.10: Local administrator/Delegate - Lists of Playlist



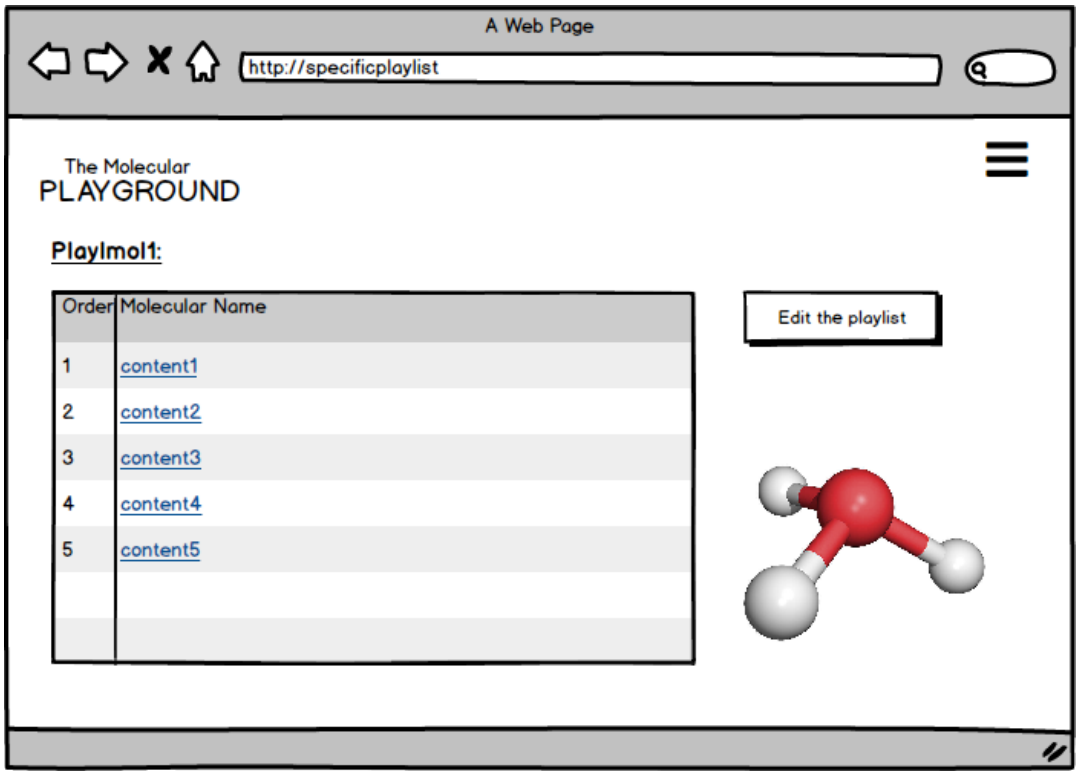
Order	Playlist name
1	playmol1
2	playmol2
3	playmol3
4	playmol4
5	playmol5

Figure 2.3.11: Local administrator/Delegate - Playlist - Create new Playlist



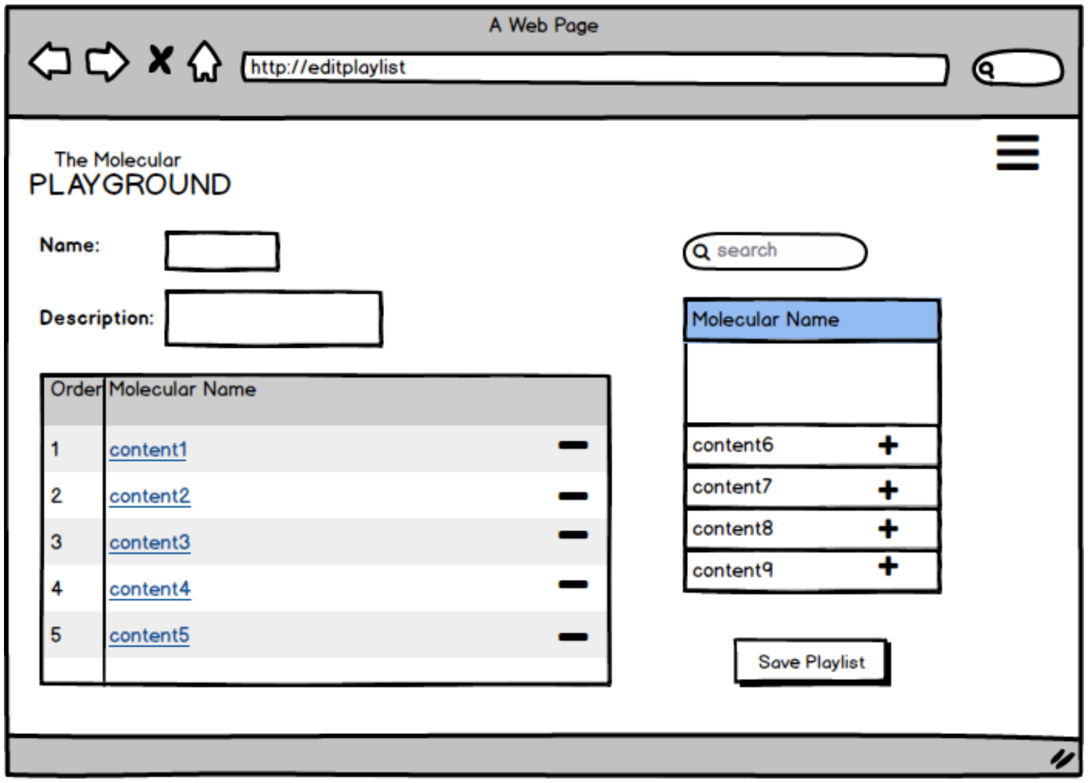
Order	Molecular Name
1	content1

Figure 2.3.12: Local administrator/Delegate - Playlist - Click Specific Playlist



Order	Molecular Name
1	content1
2	content2
3	content3
4	content4
5	content5

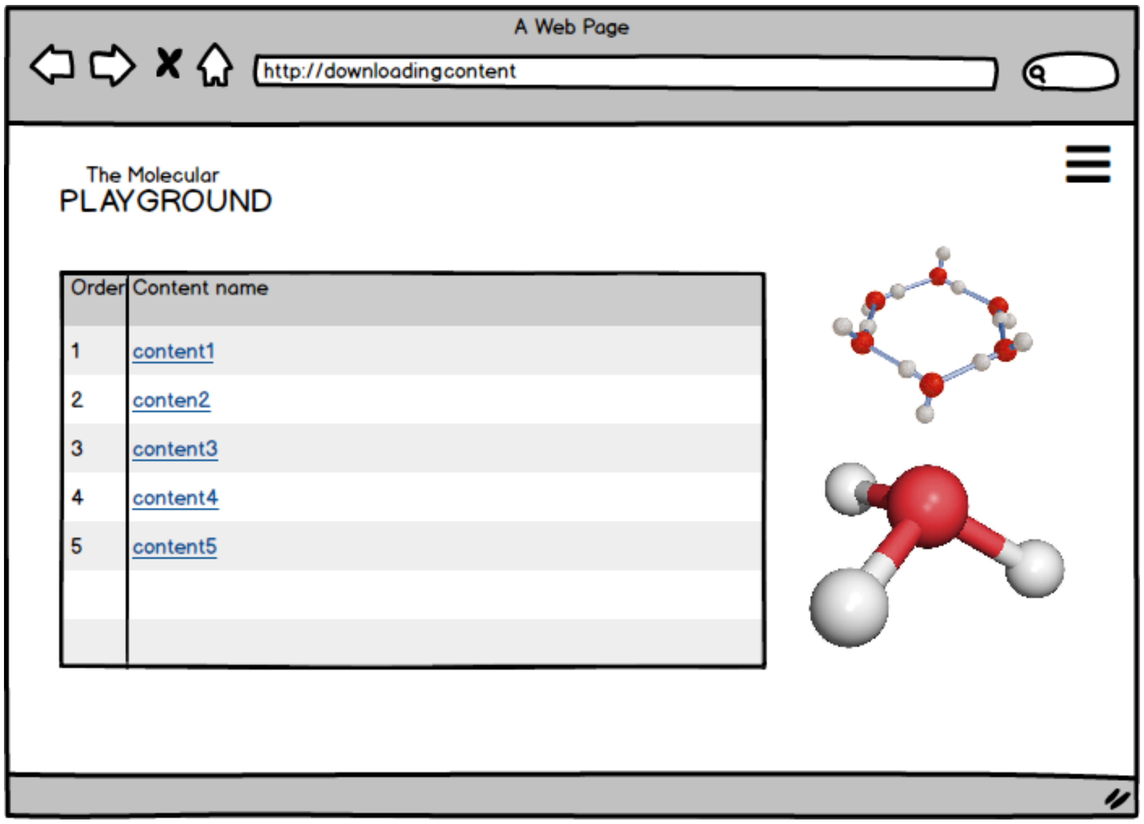
Figure 2.3.13: Local administrator/Delegate - Playlist - Edit Specific Playlist



Order	Molecular Name
1	content1
2	content2
3	content3
4	content4
5	content5

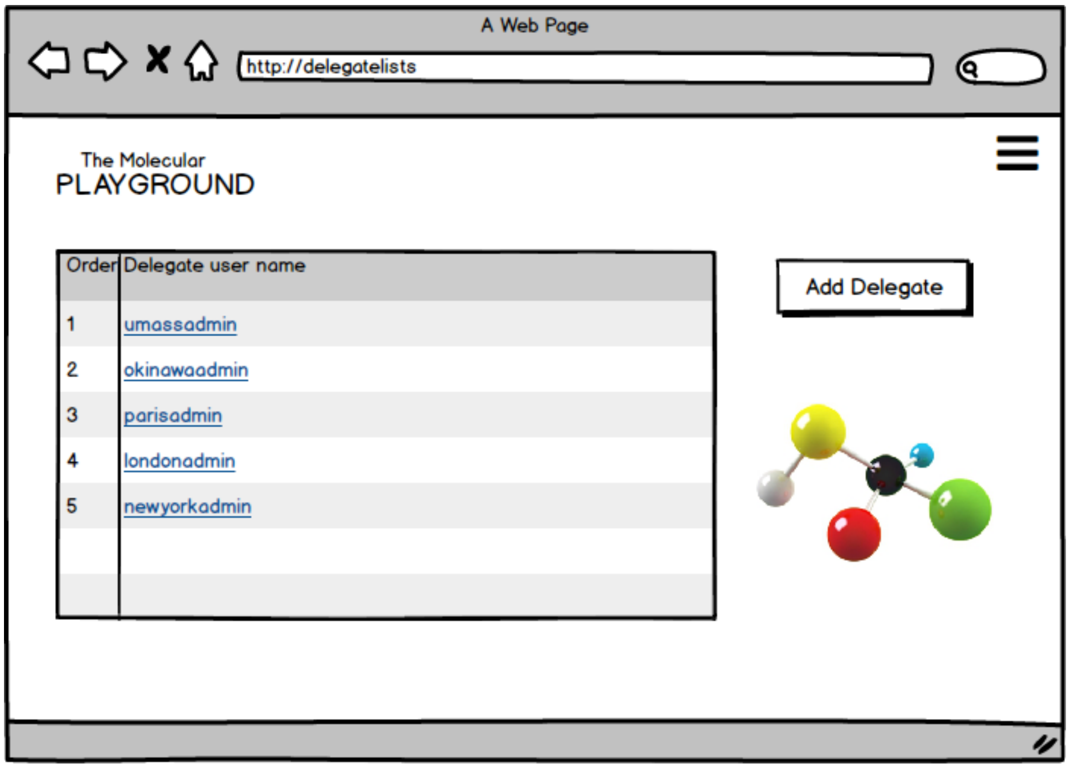
Molecular Name
content6
content7
content8
content9

Figure 2.3.14: Local Administrator/ Delegate - Download Content



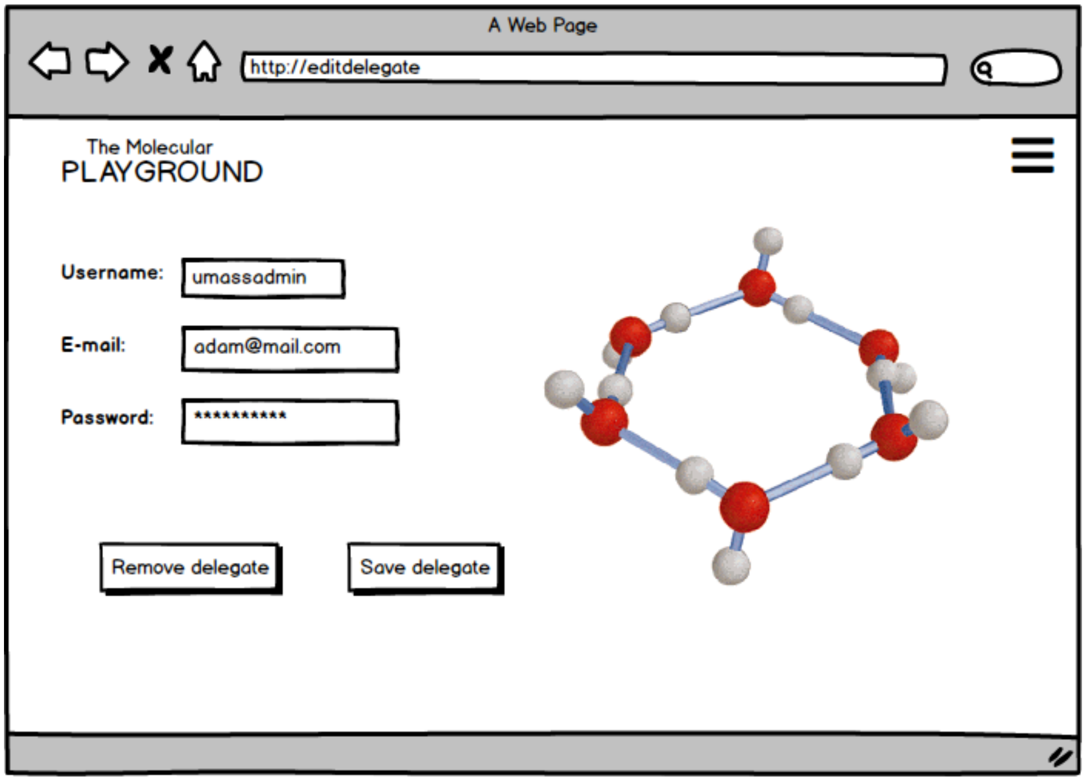
Order	Content name
1	content1
2	conten2
3	content3
4	content4
5	content5

Figure 2.3.15: Local Administrator- Manage Delegate



Order	Delegate user name
1	umassadmin
2	okinawaadmin
3	parisadmin
4	londonadmin
5	newyorkadmin

Figure 2.3.16: Local Administrator - Edit Specific Delegate



Username: umassadmin

E-mail: adam@mail.com

Password: **********

Figure 2.3.17: Local Administrator - Add Delegate

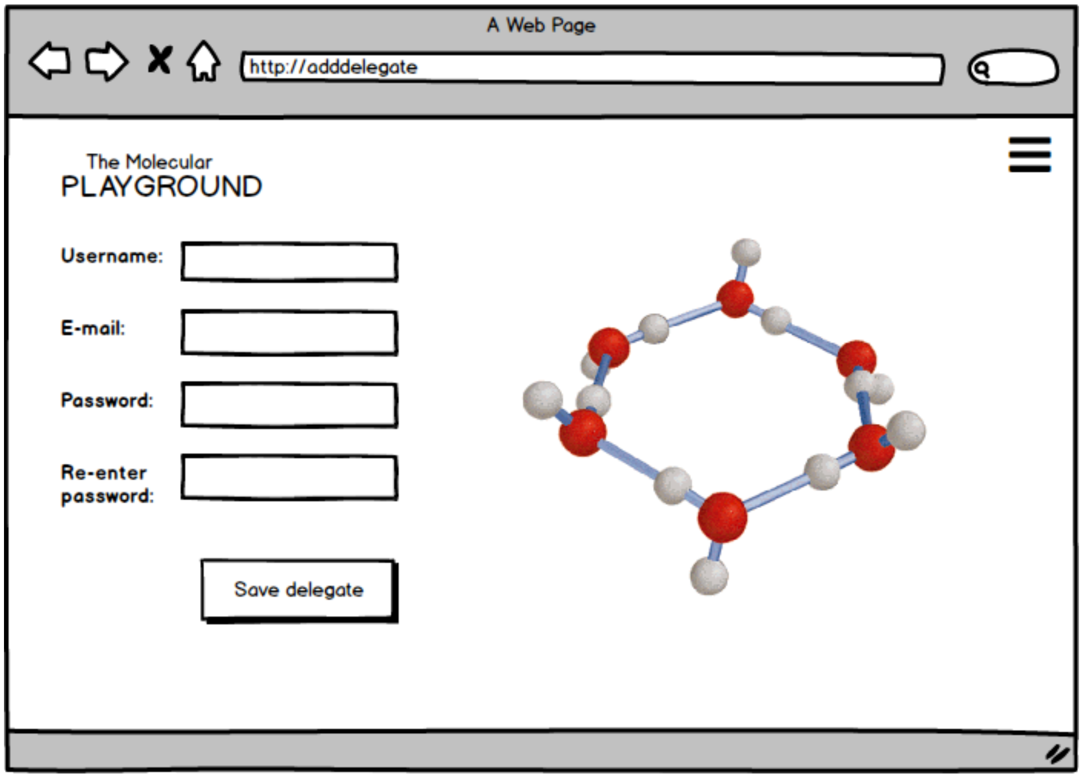


Figure 2.3.18: Local Administrator/Delegate - Schedule Content

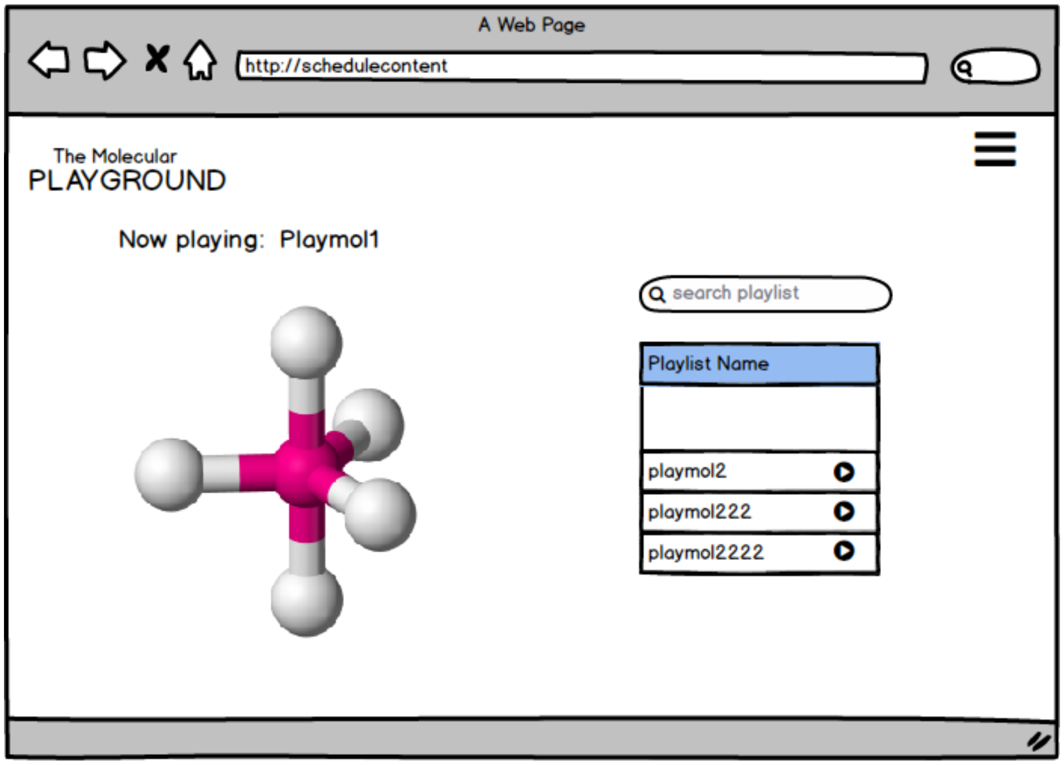


Figure 2.3.19: Local Administrator/Delegate - Schedule Content

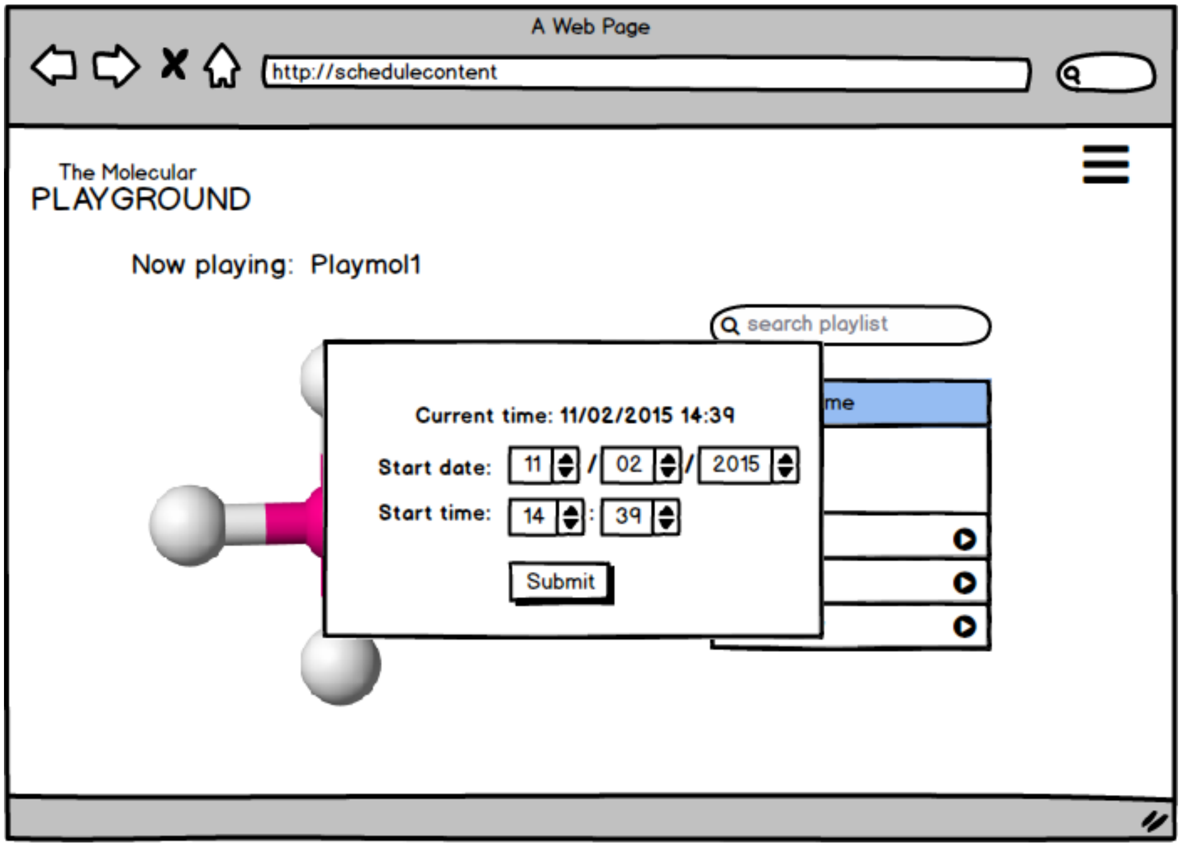


Figure 2.3.20: Login Page

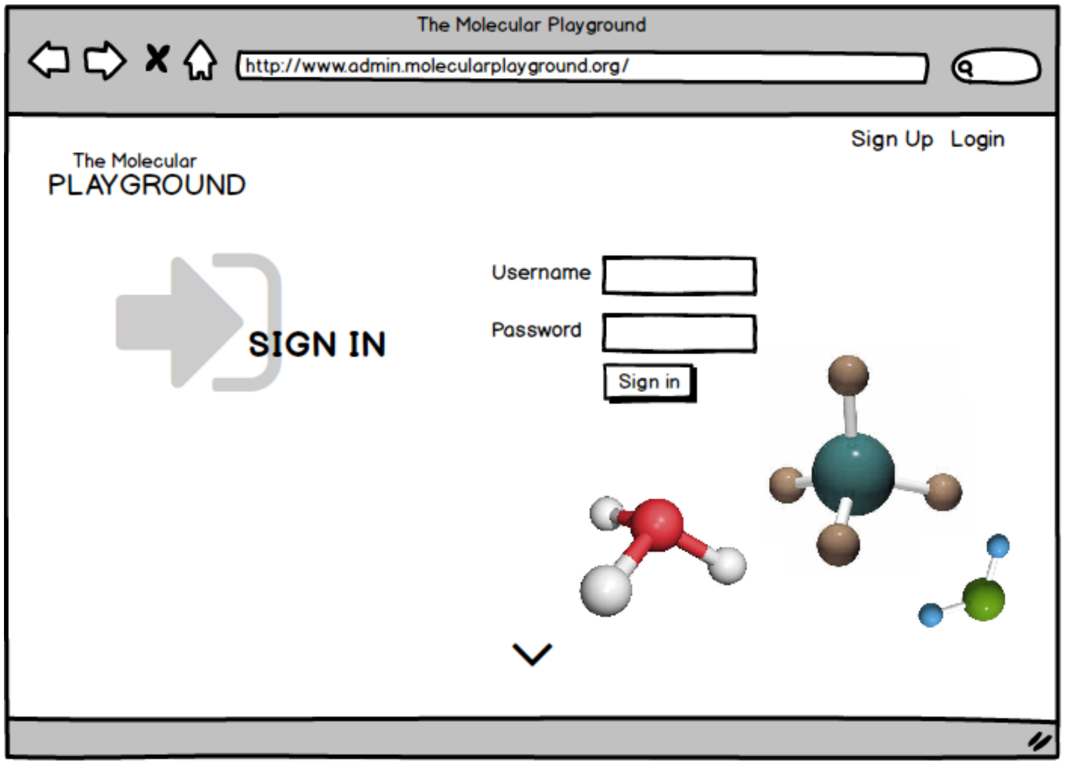
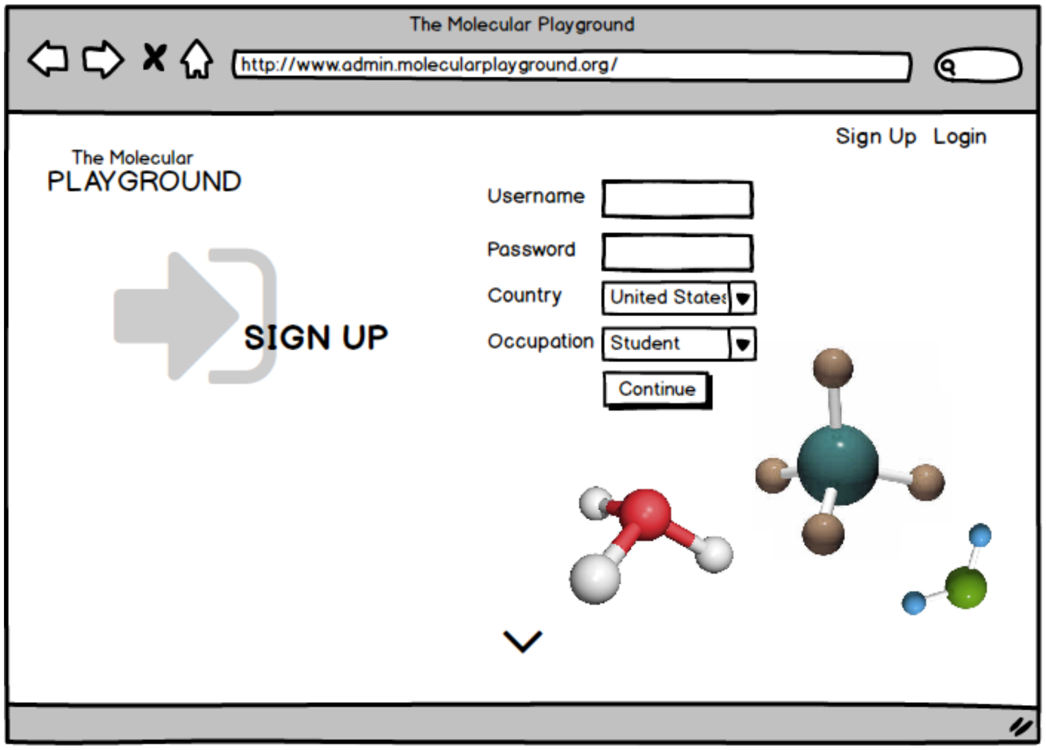


Figure 2.3.21: Sign Up



3.0 Summary

Molecular Playground Expansion will be an accessible system that can handle the user's needs in a responsive and dynamic manner. We made our interface as intuitive and clean as possible to support our accessibility goals. Authors will be able to upload new content and browse the system. Administrators will be able to manage certain users of the system. The system is a standard client-server model that will allow all installations using Molecular Playground around the world to access our database and receive new content with ease.

This document details the system we've designed that is cohesive with the Molecular Playground as it exists today, with usability at the forefront of our priority. High level communication detail with both client and server has been established, as well as encompassing elaborations on the high-level architecture. Our Model-View-Controller approach to development will enable the scalability in our system as Molecular Playground's audience grows. Using JavaScript intimately throughout our implementation will allow a tighter and more integrated product.

In conclusion, the Molecular Playground will be expanded in such a way that will allow users to access a centralized web application to manage their accounts fluidly and easily from locations around the world. Our goal is to increase its functionality so that this tool remains valuable in the accelerated worlds of education and science.