Open Cloud Exchange Interface

by Logan Bernard and Shiran Sukumar

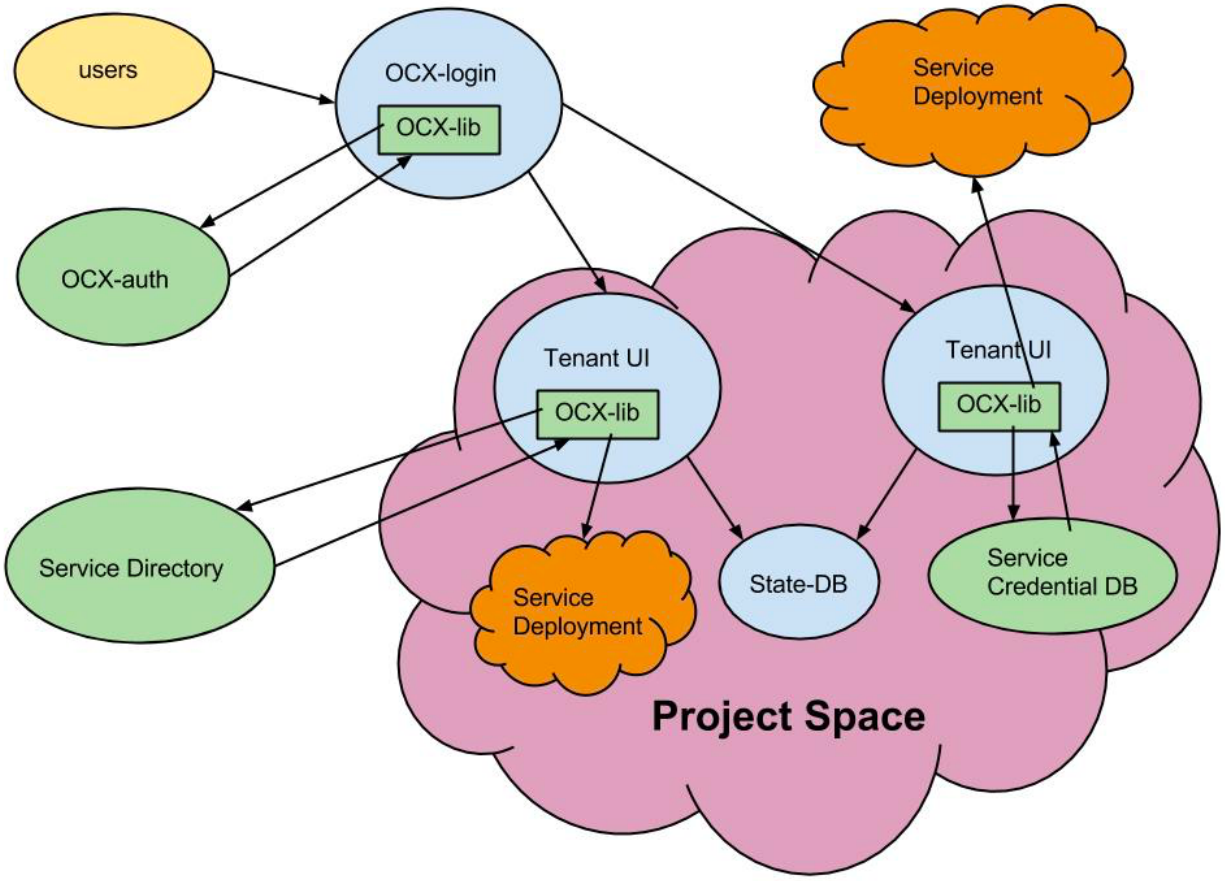
Proposal Content

High-Level Goals

I. Provide a user interface to allow multiple, potentially distrusting providers to offer their services
II. Enable a marketplace to show the high (Hadoop Service, PaaS, Appliances) and low (VMs, Storage, Computer, Networking, etc.) level customizable offerings of multiple providers
III. Develop a simple, straightforward user experience for non-expert users

Requirements

I. Security - Must secure user credentials, service and project data
II. Scalability - Must scale to large number of users, projects, and services
III. Providers - Allow multiple, potentially non-trusting providers to coexist
IV. Extensibility - Must be able to extend interface and allow third-party service interaction
V. Functionality - User must be able to create, manage, and deploy projects
VI. Performance - Must perform efficiently for production use



OCXi architecture

Key:

Green - Implementation by OCX team

Blue - Within scope of student project

Components:

- OCX Library: library used to authenticate user credentials, create project space, accesses state and credential db, accesses service directory
- OCXi : user interface for OCX
- OCX-login: interface for users to log into OCX and view project list
- OCX-auth: authentication server for user login
- Service-Directory: maintains all service endpoints; specific service information can be gathered from individual resources (ie availability, image size, SSH, CPU, price, version, name, description, etc.)
- Service Deployment: low and high level services could be a VM, appliance, or service (ie Hadoop)
- State-DB: stores a list of the project's purchased services
- Service Credential-DB (SC-DB): holds credentials for all project services
- Tenant-UI: project and marketplace interface
- Project Space: virtual space for tenants (tenant ui, state db, sc-db)

Architecture Diagram

User credentials are authenticated by OCX-auth, an authorization server. Once authenticated the Project Space is created for the user's Tenant UI. The Project Space will contain databases to keep track of the project state and service credentials. The Tenant UI uses the OCX Library to acquire information on the Project Space's services. The OCX Library calls the Service Directory to gather endpoints, which can be used to acquire each service's associated information. The OCX Library manages the credentials of the associated services in the SC-DB and stores a list of the project's purchased services in the State DB. Using the stored endpoints, the Tenant UI uses its API to connect with services. It must authenticate with the central authority to interact with its services. After this process, the Tenant UI provides a marketplace of available services and a project management interface or an extensible, third party interface (e.g. Hadoop) for interaction with services.

Design Implications and Discussion

This section briefly describes some of the key design decisions and our motivation for them.

- **Separate Per-Tenant Project Space Level**
    - The only shared central service in our design is the OCX authority used to log into the service. All subsequent interaction by a user are with services spun up in the tenant's own project space. The motivations are: 1) performance isolation since different tenants interact with different services and 2) improved security since if a project space service is penetrated, only one tenant is impacted. The OCX-login and OCX-auth still need to be designed as scalable multi-tenant services, but they are much simpler and have a reduced attack surface from the full tenant UI.

- **Tenant UI accesses DBs and services using the OCX Library**
    - In the design, all components interact with the underlying services using the OCX-library. This library is expected to expose an interface very similar to the existing OpenStack python library, but it may invoke multiple underlying OpenStack regions/deployments. For example, the interface to list all of a project's VMs may be the same as the existing library, but it will go separately to all the Nova services that the project uses and return an aggregate list. The interface of this library will isolate applications (such as the tenant UI) from changes we expect to happen to more fully support the OCX requirements in OpenStack. For example, it maintains a set of credentials for different OpenStack deployments, which will disappear when OpenStack supports federated authorization.

- **Shared Service Directory**
    - The shared service directory is part of the larger OCX project, and is not the responsibility of this project. It is being developed as a new service (rather than modifications to keystone's existing service directory) because it needs to span multiple regions/providers… and provides a single service that different providers can advertise their services to, and users can go to to find out about all the services in an OCX.

- **Project Space uses SC-DB**
    - This is a temporary kludge until OpenStack supports federated authorization. It will allow the OCX-library to obscure from most operations that we are dealing with multiple OpenStack deployments with different credentials, and will go away once we can operate on the services using a single set of credentials.

- **Project Space uses State-DB**
    - This DB holds the state of the tenant UI, so that the UI itself is stateless. New UI instances can be spun up for each user and will synchronize by going through this DB.

- **Extensible UI**
    - Rather than making the UI itself extensible using plugins (like horizon today), we instead assume that each service stands up its own UI and a common user experience is provided by common styling between the UIs as a user moves from one to another. This is necessary in a multi-provider cloud, since providers are not trusted to modify a shared service. One of the open questions is how uniform an experience we can give to users given this model.

Related Work

Through determining how OCXi will compete in the already existing market, research was required to evaluate the design. The evaluation criteria examined the aesthetic, experience, and the practicality.

AWS

Examining the Amazon Web Service, it is a basic interface that did not seem very user friendly. The AWS is the most widely used interface on the market and does provide a baseline for what one can expect to accomplish. The design was very simple and did not offer any components that could be incorporated into the design. Although AWS is not intuitive, it can be used to build example projects quickly.

JUJU

The most advanced, customizable, and intuitive user experience that provides the OCXi with a great model for presenting a unique user interface for setting up services. JuJu uses a GUI that allows users to drag and drop nodes, select their configuration and deploy. The ease of setting up OpenStack in their demo made a great case for adopting a similar interface. The model of being able to drag-and-drop or click and change components of a cloud infrastructure is very practical and user friendly. Ultimately, to provide the most customizable experience and be able to set up any configuration on the OCX, OCXi needs to be open to possibilities. Rather than limit users to ready-to-deploy packages, we aim to present choices and rather than preventing deployments if there are configuration conflicts. JuJu is a model that OCXi incorporates into its design, because in the future more people will likely want a visualization tool to see their deployments.

OCX-UI

The OCX-UI represents a standard for what the OCX needs. While examining OCX-UI the aim was to find what made the OCX-UI desirable and useful for the OCX. The ease of explanation and intuitive experience stood out as major factors. While JuJu is extremely powerful, it may be hard for some users to quickly use. OCX-UI has a universal design that almost anyone should be able to find exactly what they are looking for and purchase. Most of the design requirements were derived from understanding the OCX-UI.

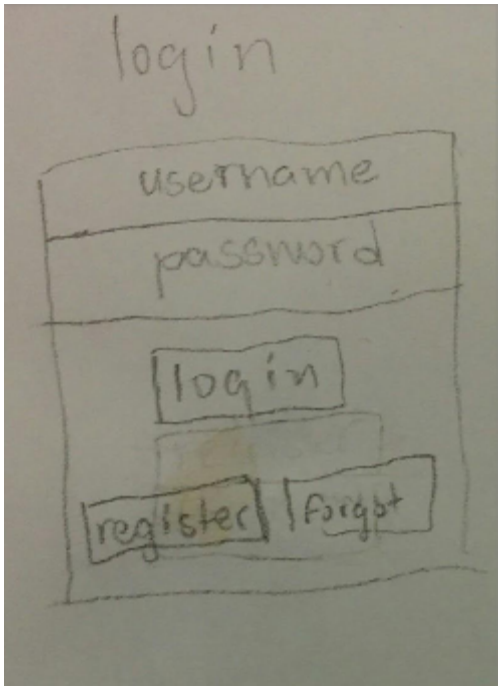
Horizon

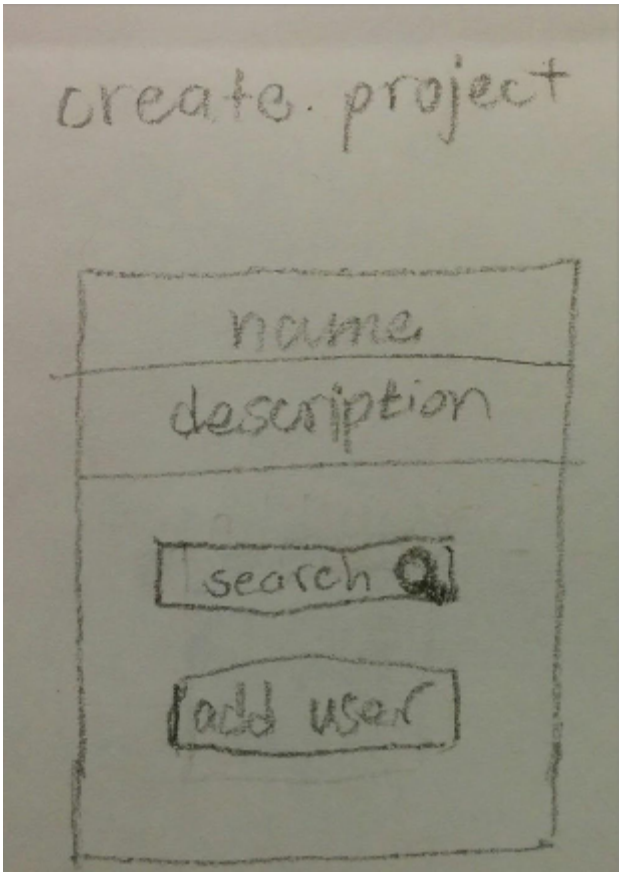
OpenStack's UI, Horizon, is a barebones interface to deploy projects. The UI is simple, and clean, but is very bland. Too bland for the OCX's needs. However, the usefulness with Horizon is the wide familiarity with the system. While OCXi will not look like Horizon, the pop up options will try to remain consistent. For example, there is a horizontal bar that represents a VM and the edit buttons should contain the same information that Horizon has in its implementation. The goal with evolving Horizon is to provide a richer experience without over complicating the pre existing user interface.

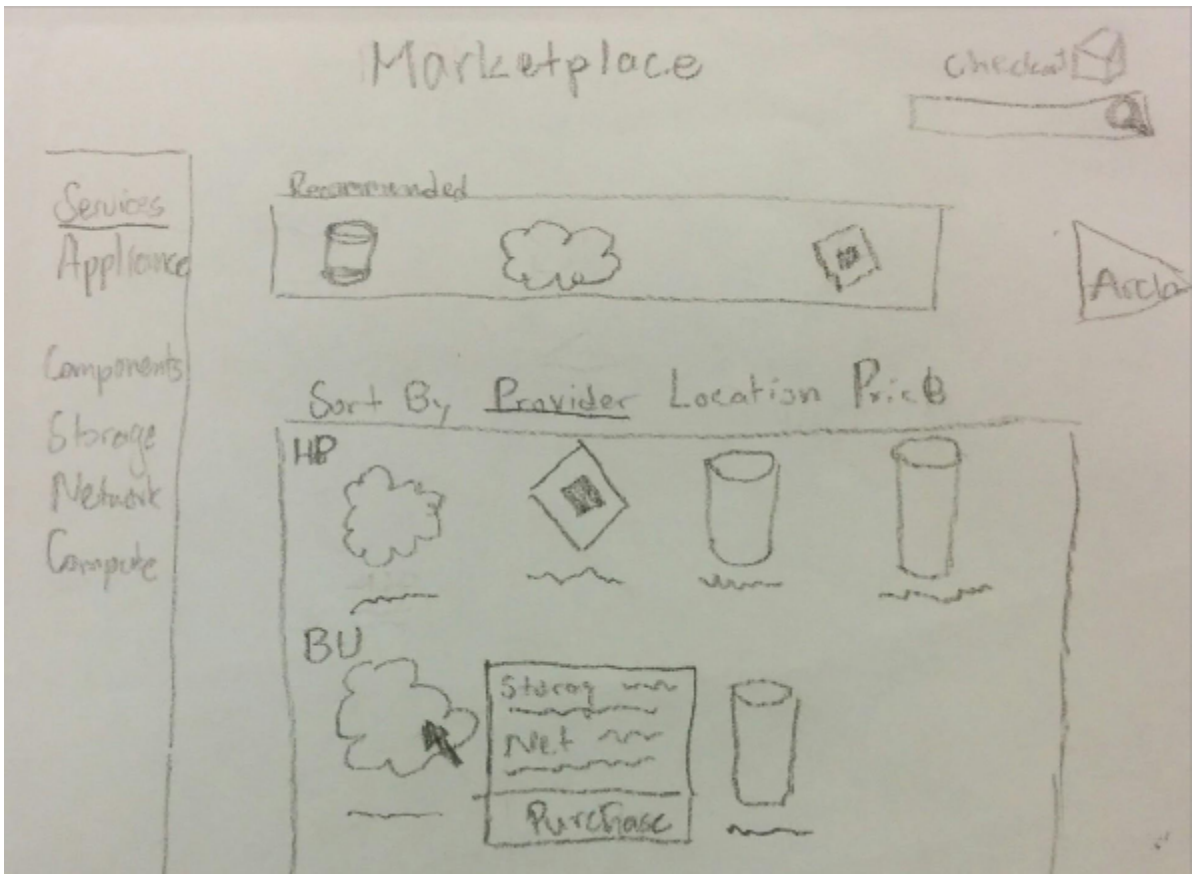
Simplifying Assumptions

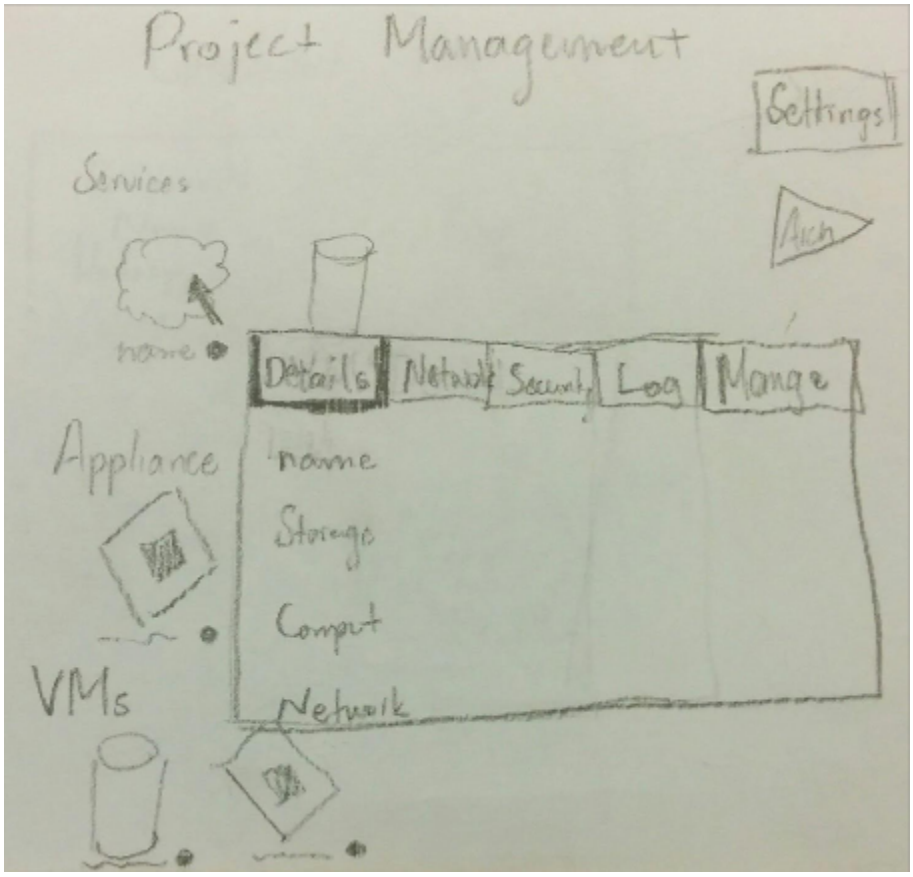
It is assumed that we will be building an interface targeting simple users. We address only the actions of tenants through simple use cases to provide a basic UI, which can be improved upon with time. Any complications will be dealt with by advanced users via the command line, and are out of the scope of this project. Additionally, there a number of simplifying assumptions made within the marketplace. First, we will only display volume storage options and limited compute customizability. We will also leave payment methods and services as out of the scope of our project.

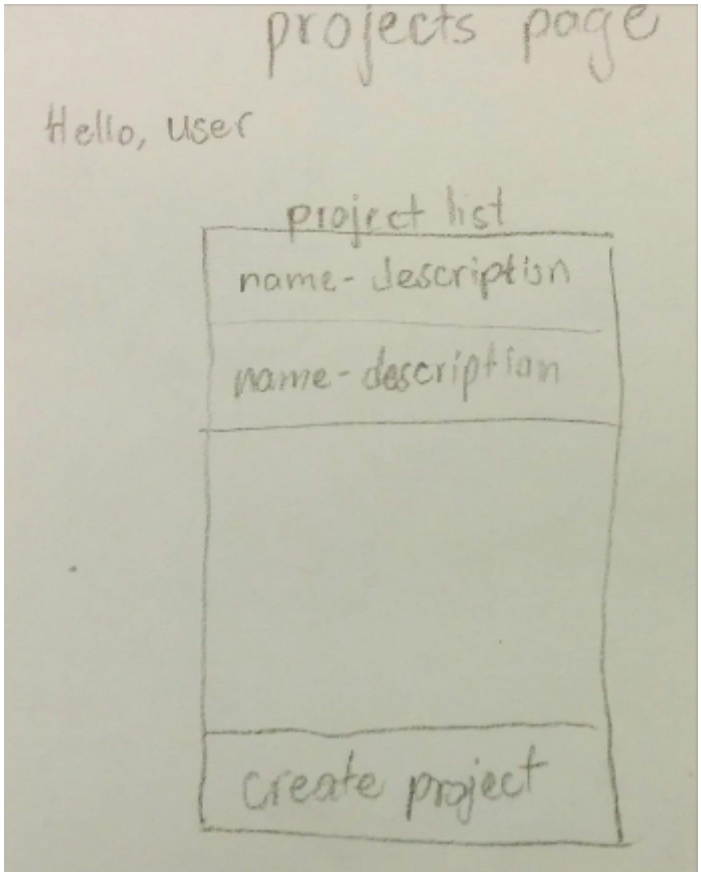
Design Diagrams








Project Management
Settings
Services
Auth
name
Details
Network
Security
Log
Manage
Appliance
name
Storage
Compute
Network
VMs


projects page
Hello, user
project list
name - description
name - description
create project

User Stories

1. New user creates a project, purchases and deploys a VM
2. User Logout
3. Returning user destroys an active VM and project
4. Returning user alters an active VM and project
5. User Renames VM
6. User adds and uses extensible service
7. User Renames Network
8. Admin adds User

1. New user creates a project, purchases and deploys a VM

- OCX-Login
    - On the OCX-Login page, the user selects 'Register' and is sent to a registration page.
- New User Registration
    - After submitting basic information (e.g. email, password, name), the user is registered with the OCX-Authority and the user is sent to the Projects Page.
- Projects Page
    - The user's available projects (names and descriptions) are listed for selection. Only one project may be selected and entered at a time, sending the user into the specified Project Space.
    - For a new user, only the 'Create New Project' option is available. After submitting a project name and description, the project is deployed and the user is sent to the new Project Space as the project's admin.
        - Also can add users to be associated with the project by OCX username
        - permissions handled on the project management page

Project Space

- OCX-I (marketplace)
    - For a new project, the user lands in the marketplace.
    - The user selects Resources
    - User Selects Storage
        - Chooses a storage option available
    - User selects Compute
        - Chooses a compute option
    - User adds Appliance
        - User searches and selects Appliance
            - Sets resources to uses (ie default storage and compute just selected)
    - User goes to Project Management
- Project Management
    - User selects their VM
    - VM Details (Overlay)

        - the user is able to see details of the VM once again, and chooses to power it on.
        - Use SSH: User clicks details and finds the IP and SSH Public Key
        - options to attach public network, add private network
        - *(stretch goal) Open Console: VNC*
- At this point, a new user, project, and VM have been created. The user can access their current VM using SSH, or continue purchasing and deploying items from the marketplace via the project management page.

2. User Logs Out

- OCX-Login
    - The user enters their credentials and selects ‘Login’
- Projects Page
    - The user views their list of project names and descriptions, and enters (is sent to the Project Space of) ‘Project 1’.

Project Space

- User lands on the Project Management page
- User clicks logout
- User is taken to the Projects Page

3. Returning user destroys an active VM and project

- OCX-Login
    - The user enters their credentials and selects ‘Login’
- Projects Page
    - The user views their list of project names and descriptions, and enters (is sent to the Project Space of) ‘Project 1’.

Project Space

- Project Management
    - The returning user lands on the project management page upon entering their project’s space.
    - the user selects the ‘Project Admin’ button
- Project Admin (user / project description management)
    - the user selects ‘Destroy Project’
    - the user enters their credentials as a confirmation of their intention and authorization to destroy the project (must also be clean)
    - error message indicates the user must delete all VMs first
    - user returns to the project management page
- Project Management
    - They see the details of their project. There is one VM, which is active; this VM is selected
    - VM Details (Overlay)
        - the user shuts down the VM.
        - the user selects ‘destroy VM’, and does a second confirmation of their action
    - user selects the ‘Projects Admin’ button
- Project Admin (user / project description management)
    - the user selects ‘Destroy Project’
    - the user enters their credentials as a confirmation of their intention and authorization to destroy the project
    - deletion confirms
    - takes user to projects page

4. Returning user alters an active VM and project

- OCX-Login
    - The user enters their credentials and selects ‘Login’
- Projects Page
    - The user views their list of project names and descriptions, and enters (is sent to the Project Space of) ‘Project 1’.

Project Space

- Project Management
    - The returning user lands on the project management page upon entering their project’s space.
    - They see the details of their project. There are several active VMs, the user wants to connect two VMs over the same network
    - User selects VM-1 to view details
    - VM Details (Overlay)
        - clicks Network tab and chooses to add a network
        - user associates VM-1 with public network “net1”
    - user selects another VM-2
    - VM Details (Overlay)
        - views the network tab; user associates “net1” with VM-2

5. Returning user renames a VM

- OCX-Login
    - The user enters their credentials and selects ‘Login’
- Projects Page
    - The user views their list of project names and descriptions, and enters (is sent to the Project Space of) ‘Project 1’.

Project Space

- Project Management
    - The returning user lands on the project management page upon entering their project’s space.
    - User selects VM-1 to view details
    - VM Details (Overlay)
        - clicks name
        - renames VM-1 to ‘VM-new’

6. User Adds and Uses Extensible Service *(stretch goal)*

- OCX-Login
    - The user enters their credentials and selects ‘Login’
- Projects Page
    - The user views their list of project names and descriptions, and enters (is sent to the Project Space of) ‘Project 1’.

Project Space

- Project Management
    - The returning user lands on the project management page upon entering their project’s space.
    - The user decides to add a Service
    - User selects ‘Marketplace’ tab
    - The user selects a Service (HPC, Hadoop, Protein Modeling, etc.)
        - User chooses from a list of possible configuration options (Size, Speed, Network)
- Project Management
    - User selects Service
    - Service Details (Overlay)
        - selects manage which opens Service Interface (e.g. Hadoop’s native interface)
- Service Interface
    - User interacts with Service Interface (3rd party)
    - Any changes are reflected upon their service, and can be seen via the Project Management page

7. Returning user renames a Network

- OCX-Login
    - The user enters their credentials and selects ‘Login’
- Projects Page
    - The user views their list of project names and descriptions, and enters (is sent to the Project Space of) ‘Project 1’.

Project Space

- Project Management
    - The returning user lands on the project management page upon entering their project’s space.
    - User selects VM-1 to view details
    - VM Details (Overlay)
        - clicks Network tab
        - clicks edit button next to ‘net1’
        - renames ‘net1’ to ‘new-net1’
            - warning thrown about other VMs attached to ‘net1’
            - these VMs will now be connected to ‘new-net1’

8. Admin adds a user

- OCX-Login
    - The user enters their credentials and selects ‘Login’
- Projects Page
    - The user views their list of project names and descriptions, and enters (is sent to the Project Space of) ‘Project 1’.

Project Space

- Project Management
    - The returning user lands on the project management page upon entering their project
    - User clicks ‘Project Admin’ button
    - Views ‘People’
    - Clicks ‘add user’
    - Fills out username and permissions (user must already be registered with OCX)
    - Saves new user information

Tasklist

The OCX Library, Service Directory, and SC-DB (service credentials database) are out of scope, although much of the Project Space's functionality is dependent upon these components. We will emulate interaction with these components. For our project's purposes, the Login Page and Project Space will be hosted on the same web server. It is the responsibility of the OCX team to implement this web server.

Our Project will attempt to make basic, list orientated pages, but we also provide our thoughts on cleaner, enhanced pages.

Login Page

- create django page (1 day)
  - make fields for : username, password
  - make button for : login, register, forgot password
- create OCX-auth pseudo database calls (3 days)
  - create apache server
  - create pseudo table of credentials
  - create example request function call
  - create example return value

Projects Page

- create django page (1 day)
  - list user's associated projects (name+description of each)
  - select project and go to management page (in project space)
- create pseudo projects-db (1 day)
  - create psuedo project's table calls to store project name and description and associated users with permissions

Project Space

- create pseudo state-db calls
  - to a project's service endpoints and project info(2 days)

Pseudo-API to emulate interaction with OCX Library (4 days)
example request for service to credentials db
example return for service from credentials db
function call to Service Directory (out of scope) for information about services

Project Management

create django page
request/function call for VMs and Services of project (2 days)
display VMs (1 week)
get information from service directory
connect to services via pseudo library, service directory
display list of VMs
display VM details (cpu, image, network, ssh, etc)
add buttons to edit VMs
Enhanced features
news feed of user changes
username, change made to vmname, timestamp
display services and their details
add button to call service interface (third party)

MarketPlace

create django page (1 week)
get available services/components via library, service directory
display categories sidebar
display services by category
add appliances
display configuration pane, detailed info
Enhanced features
shopping cart
add to cart

- checkout
- cart details on click

search functionality

filter remaining choices

static project diagram (showing connections between components)

Billing *(Enhanced Version)*

create django page (1 week)

- get billing information via library
- display overview
- display individual component charges
- add pay button
- add confirmation button
- add charge button

Milestones

Mid-to-End October

Goals: Final Proposal; Final Design; UI Implementation; Demo

Final Proposal and Visualization

Oct 16 - Oct 23

- We will create a final proposal and visualization of our design.
- The finalized proposal
- The finalized design with focus on a market-first approach

Basic Django Implementation of Design

Oct 24 - Oct 31

- Begin the implementation using Django.
- Build out wireframe design
- Third party user will be able to maneuver through the interface

Mid November

Goals: UI Implementation; OCX Library Integration with OCXi; Demo

OCX Library Integration

Nov 3 - Nov 14

- Focus on the underpinnings that will be necessary to make the OCXi function.
- Lay out the API calls necessary for: user authentication, retrieval of available resources, and service manipulation (attachment to User Cloud, billing, etc).
- Integrate a model of the OCX Library (which will be developed by the OCX)

End of November

Goals: Enhanced and Full User Experience

Visual Enhancement and Finalized User Stories

Nov 17 - Nov 25

- Click through the OCXi and understand the intended functionality
- Expose the OCX Library interaction
- Visual elements will be refined

December

Goals: Final Project Report and Presentation

Project Fixes: Odds and Ends

Dec 1 - Dec 9

- Buffer time to pick up any slack

Final Presentation

Dec 9

- Able to give a full walk-through of the OCXi