

Florida International University
School of Computing and Information Sciences

CIS 4911 - Senior Capstone Project
Software Engineering Focus

The logo for HyperDesk 1.0 is displayed in a teal, cursive-style font. The text "HyperDesk 1.0" is centered within a dark blue rectangular background. The "H" is particularly large and stylized, with a long vertical stroke that extends below the baseline of the other letters.

HyperDesk 1.0

Final Deliverable

Team Members:

Rachelle Tobkes
Daniel Alvarez

Product Owner(s):

Pia Celestino

Mentor(s):

Pia Celestino

Instructor:

Masoud Sadjadi

The MIT License (MIT)
Copyright (c) <2016> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

In this document, we present the HyperDesk system; a web application for listing/renting office spaces. Unlike current systems, HyperDesk gives users the freedom of naming their own price when wanting to rent a workspace. In this document, we include a detailed description of the user stories that were implemented, as well as make note of the user stories that are still pending, supply the project plan including a discussion of each sprint and hardware/software requirements, describe the system design, and provide system validation.

Table Of Contents

[Table Of Contents](#)

[Introduction](#)

[Current System](#)

[Purpose of New System](#)

[User Stories](#)

[Implemented User Stories](#)

[Pending User Stories](#)

[Project Plan](#)

[Hardware and Software Resources](#)

[Sprint Plans](#)

[Sprint 1](#)

[Sprint 2](#)

[Sprint 3](#)

[Sprint 4](#)

[Sprint 5](#)

[Sprint 6](#)

[Sprint 7](#)

[System Design](#)

[Architectural Patterns](#)

[Deployment Diagram](#)

[Design Patterns](#)

[System Validation](#)

[Glossary](#)

[Appendix](#)

[Appendix A - UML Diagrams](#)

[Static UML Diagrams](#)

[Dynamic UML Diagrams](#)

[Appendix B - User Interface Design](#)

[Appendix C - Sprint Review Reports](#)

[Appendix D - Sprint Retrospective Reports](#)

[Appendix E - Project Management](#)

[References](#)

Introduction

In this section, we introduce HyperDesk and discuss current systems that exist for listing/renting office spaces. We explore the limitations of the present systems and describe the functionalities that our new system will offer. We begin with a discussion of the current systems where we make note of the problem that our system will attempt to solve and we conclude with a description of the new system where we contrast the new features with the existing ones.

The remainder of the document consists of the following seven sections: User stories, Project Plan, System Design, System validation, Glossary, Appendix, and References. In the user story section, we provide a comprehensive list of the user stories that we have implemented, as well as the user stories that are still pending. In project plan, we include the hardware/software requirements of the HyperDesk system, as well as the plans for our eight sprints. In the system design section, we discuss the architectural patterns, system and subsystem decomposition, deployment diagram, and design patterns utilized in our system.

In system validation, we provide system and subsystem test cases for each individual user story. In the glossary, we define terms that are used throughout the document. In the appendix, we include all of our UML diagrams, user interface designs, sprint retrospectives, and sprint plannings. Lastly, the document is concluded with a references section where we reference work that is not our own.

Current System

Very often, many businesses have empty office or desk space in their buildings that go unused. Likewise, individuals are constantly searching for temporary workspaces for their companies. Currently, several systems exist for listing/renting office spaces. Some of these systems include ShareDesk, LiquidDesk, and DeskSurfing.

Through these applications, individuals in need of office space rentals can easily find workspaces in their area. Additionally, those who have unoccupied office spaces are able to create posts for other people to find and rent their workspaces.

Although the current systems seem to already solve the problem, they do have a few limitations. Individuals looking to rent office spaces can only rent for a given price. Sometimes, this price is too expensive; causing people to forgo renting the space, which then leads to the office still being unoccupied.

Purpose of New System

The new system will include functionalities that the current systems do not currently entail. Our system will not only provide a method for individuals to list and rent office spaces, but also, allow users to name their own price. Unlike the current systems, users will have the opportunity to choose between renting an office space for the given price, or making an offer for their preferred price. Office space owners will then be able to select which offer they want to accept.

This allows users more freedom when choosing a workspace to rent. If he/she sees that one is too expensive, then he/she can make his/her own offer in hopes that the lister will accept it. Usually, if a lister has no other offers or renters for that day and time frame, the lister will grant the offer, rather than having their office space go unused. Furthermore, our system will provide a messaging system for users to message other users with office space inquiries through the application.

User Stories

In this section, we provide a comprehensive list of the user stories that have been implemented, as well as those that are still pending to be implemented at a later time. Each sprint consisted of one or more user stories per developer. Additionally, a new user story could not be begun until the previous assigned one was completely finished.

Implemented User Stories

User Story #759 - Search for Office Spaces

As a user, I want to be able to search for office spaces near me, so that I may view them and/or rent if interested.

User Story #760 - Log Into the System

As a registered user, I want to be able to log into my account so that I can post and/or rent office spaces.

User Story #761 - Register for the System

As a user, I want to be able to register for the system so that I am able to log into my account and use the member features.

User Story #762 - Log Out of the System

As a registered user, I want to be able to log out of my account so that I am able to log back in at a later time.

User Story #763 - List an Office Space

As a registered user, I want to be able to post an office space so that other users may view and/or rent my office space.

User Story #764 - View an Office Space

As a user, I want to be able to view an office space so that I am able to see it's details and rent it if interested.

User Story #765 - Forget Password

As a registered user, I want to be able to reset my password if so that I can still log into my account in case that I have forgotten it.

User Story #766 - Make an Offer

As a registered user, I want to be able to make an offer for an office space, so the can either accept it or decline it.

User Story #767 - Review an Office Space

As a registred user, I want to be able to rate/review an office space so that other users will know if it is good or bad.

User Story #768 - View Office Space Reviews

As a user, I want to be able to view the ratings/reviews for an office space so I will know beforehand if it is good or bad.

User Story #769 - Edit Profile

As a registered user, I want to be able to edit my profile information so that other users can view it and see if I am trustworthy enough to rent their office space.

User Story #772 - Filter Search Results

As a user, I want to be able to filter my search results so that I can easily find exactly what I am looking for.

User Story #818 - Accept/Decline Offers

As a registered user, I want to be able to accept or decline my offers so that people can either rent my office space or submit a new offer.

User Story #830 - Rent Office Space Immediately

As a registered user, I want to be able to rent an office space immediately, so that I do not have to wait for the seller to either accept or decline an offer that I make.

User Story #841 - Message Users

As a registered user, I want to be able to message other users to ask them any questions that I may have about their office space.

User Story #850 - Save Credit Card Details

- As a member, I want to be able to save my credit card details so that I can use them at a later time when I rent or make an offer.

User Story #851 - Save Bank Account Details

- As a member, I want to be able to set up my bank account details so I am able to get paid when users rent my office space.

Pending User Stories**User Story #770 - Edit Office Space**

As a registered user, I want to be able to edit the details of my office space so I can correct and/or update any details that I may have included.

User Story #771 - View Accepted Offers

As a registered user, I want to be able to view my accepted offers so that I can know which office spaces I have rented.

User Story #806 - Delete an Office Space

As a registered user, I want to be able to delete any office space that I have posted so that I can remove it if I no longer want people to rent it.

User Story #859 - View Sent Offers

As a registered user, I want to be able to view my offers that I have sent so I can keep track of the office spaces that I have made offers for.

User Story #860 - Receive Notifications

As a member, I want to be able to receive a notification when I have a new message, new offer, or my offer is accepted.

User Story #861 - Use Saved Card Details

As a member, I want to use my saved credit card details so that I do not have to enter them each time I want to rent an office space.

User Story #862 - Update Bank Account Information

As a member, I want to be able to update my bank account information so that I can change the bank account of where I will receive my funds.

User Story #863 - Use the Website Securely

As a user, I want to be able to use the website securely when I set up my bank account or enter credit card details.

Project Plan

In this section, we begin with a comprehensive list of the hardware and software requirements that we used to develop our system, including the reasons behind our selections. Then, we present our sprint plans. For each sprint, we list the user stories in descending order of priority and include their tasks, acceptance criteria, and UML diagrams.

Hardware and Software Resources

Handlebars: We chose to utilize handlebars in our application due to its logic less templating engine. Handlebars is known for having little to no logic in its templates that are on the HTML page. This allows us to keep our HTML pages simple, clean, and separate from our logic-based JavaScript files.

Express: We chose to use Express.js because it grants us features that will extend our server side coding. Also, it goes hand in hand with Node.js which is another reason why we selected it.

UI-kit: We chose to use UI-kit due to all of it's available components. Certain components such as auto-completion, time picker, file uploading, and image displays will be very useful to us and have a minimalist/modern design to them which is what our product owner is looking for.

Node.js: We chose to employ Node.js in our application due to it allowing both client side and server side code. This allows us to use the same patterns for both frontend and backend and also, the same libraries. Additionally, it handles the client and server requests asynchronously.

Sass: We are using Sass because it allows us more flexibility with our stylesheets and offers additional features that CSS does not have.

MongoDB: We chose to incorporate MongoDB into our web application because it is a NoSQL database, and we wanted to gain more insight into it. It allows our database schema to be more scalable and it also has a deep query ability which supports dynamic queries.

Mongoose: We chose mongoose as our middleware because it provides a schema-based solution to model application data. It includes built-in type casting, validation, query building, business logic hooks and more.

Stripe API: We chose to use the Stripe API to process payments because it provides security to our application. Instead of saving confidential information on our server, stripe generates a secure token which we are able to save on our server and use to charge payments in the future.

Webstorm 2016: We are using the WebStorm IDE because it is equipped with server-side and client-side development with Node.js. It also supports HTML, CSS, and has version control.

Sprint Plans

Sprint 1

(01/16/2016 - 01/29/2016)

User Story #761 - Register for the System

Tasks

- UML diagrams
- Form validation
- Implement email verification
- Set up bcrypt for password encryption
- Create register UI
- Unit and Integration tests

Acceptance Criteria

- The user must enter a valid email address, username, password, confirms password, first name, and last name.
- The password is encrypted.
- The user receives a verification email upon registering.

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. In our use case model, we have two actors: the user and the member. The member inherits all of the use cases from the user while taking on additional use cases that the user cannot participate in. Our sequence diagram in **Figure 2** represents the interactions in our system. The register use case begins when the user enters their information and clicks submit and ends when the register redirects the user to the verify page. Our class diagram in **Figure 3** shows each class' attributes and how they relate to one another. For example, the RegisterController depends on the RegisterModel.

User Story #760 - Log into the System

Tasks

- Develop UML diagrams
- Implement user sessions
- Validate user input
- Create login UI
- Unit and Integration tests

Acceptance Criteria

- The system notifies the user when he/she attempts to login with invalid credentials
- The system validates the username with its corresponding password
- The user can only login once he/she has verified his/her account

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 4**. And our class diagram in **Figure 5**.

User Story #762 - Log out of the System

Tasks

- UML diagrams
- Create logout button in menu
- Unit and Integration tests

Acceptance Criteria

- The user is on the login page once he/she logs out

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 6**. And our class diagram in **Figure 7**.

Sprint 2
(01/30/2016 - 02/12/2016)

User Story #763 - List an Office Space***Tasks***

- Develop UML diagrams
- Implement drag and drop to upload images
- Validate user input
- Create UI
- Integrate Google Maps API
- Unit and Integration tests

Acceptance Criteria

- The user must enter a valid title, description, address, price, contact email/number, images, and amenities.
- The user is presented with their newly posted office space once it is submitted.
- The user is presented with a map when they enter the office location

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 8**. And our class diagram in **Figure 9**.

User Story #764 - View an Office Space***Tasks***

- Develop UML diagrams
- Create UI
- Integrate Google Maps API
- Create Dummy office space data
- Unit and Integration tests

Acceptance Criteria

- The user is presented with the office space's title, description, address, price, contact email/number, and amenities
- Buttons to rent the office space or make an offer are displayed.
- A map is displayed with the location of the office space.
- The images are shown in a slideshow

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 10**. And our class diagram in **Figure 11**.

User Story #765 - Forget Password

Tasks

- Develop UML diagrams
- Create forget password form
- Unit and Integration tests
- Integrate password reset email

Acceptance Criteria

- The user supplies his/her email address
- An email is sent to the user with a link to change his/her password
- The user's email address is validated

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for rent now is presented in **Figure 1**. In our use case model, we have two actors: the user and the member. The member inherits all of the use cases from the user while taking on additional use cases that the user cannot participate in. Here, only the members can request for forgotten password which extends logging in. Our sequence diagram is shown in **Figure 12**. And our class diagram in **Figure 13**.

Sprint 3

(02/13/2016 - 02/26/2016)

User Story #759 - Search for an Office Space

Tasks

- Develop UML diagrams
- Develop algorithm to find nearby spaces
- Create UI
- Unit and Integration tests

Acceptance Criteria

- The user is provided with a list of office spaces within the area of the selected location.
- The user can click on an office space to be presented with more details.
- The items on the list will show a picture of the office space, a title, and price.

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 14**. And our class diagram in **Figure 15**.

User Story #767 - Review an Office Space***Tasks***

- Create UML diagrams
- Create rating form
- Unit and Integration tests

Acceptance Criteria

- The average rating and total ratings are updated
- The user must at least select a star (between 1 to 5)
- Writing a review in the textbox is optional

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system presented in **Figure 1**. Our sequence diagram is shown in **Figure 16**. And our class diagram in **Figure 17**.

User Story #768 - View Office Space Reviews***Tasks***

- Create UML diagrams
- Create UI
- Unit and Integration tests

Acceptance Criteria

- The reviews are displayed at the bottom of an office space page
- The review includes the username, date, star rating, profile picture, and written review (if any)

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 18**. And our class diagram in **Figure 19**.

Sprint 4
(02/27/2016 - 03/11/2016)

User Story #766 - Make an Offer***Tasks***

- Create UML diagrams
- Create UI
- Set up Pikaday for the calendars
- Integrate Stripe for payments
- Unit and Integration tests

Acceptance Criteria

- The user enters their credit card information
- The user must choose per hour, per day, or per month.
- The user must select a date and hour for per hour.
- The user must select a date and duration for per day and per month.
- The user must be able to view the current availability.
- If the offer is lower than 75% of rent now, the system notifies the user that it is too low and the offer does not go through.

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 20**. And our class diagram in **Figure 21**.

User Story #772 - Filter Search Results***Tasks***

- Create UML diagrams
- Create UI for the filters
- Develop query builder
- Unit and Integration tests

Acceptance Criteria

- The user can choose from various filter (price, features, etc)
- The search results are refreshed and updated according to the filter

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 22**. And our class diagram in **Figure 23**.

Sprint 5

(03/19/2016 - 04/01/2016)

User Story #830 - Rent an Office Space Immediately

Tasks

- Create UML diagrams
- Reuse UI from Make an Offer
- Stripe Integration
- Unit and Integration tests

Acceptance Criteria

- The user enters their credit card information
- The user must choose per hour, per day, or per month.
- The user must select a date and hour for per hour.
- The user must select a date and duration for per day and per month.
- The user must be able to view the current availability.

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 24**. And our class diagram in **Figure 25**.

User Story #818 - Accept/Decline Offers

Tasks

- Create UML diagrams
- Create UI for viewing offers
- Stripe Integration
- Unit and Integration tests

Acceptance Criteria

- Offers are shown in a list
- Offers are shown as pending or accepted

- Accept and decline buttons

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**. Our sequence diagram is shown in **Figure 26**. And our class diagram in **Figure 27**.

Sprint 6

(04/02/2016 - 04/15/2016)

User Story #841 - Message Users

Tasks

- Create UML diagrams
- Create UI for viewing messages
- Create form for sending messages
- Unit and Integration tests

Acceptance Criteria

- Previous messages are displayed.
- List of sent messages is shown.
- User must be logged in to send a message.

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for our entire system is presented in **Figure 1**.

Our sequence diagram is shown in **Figure 29**. The message user use case begins when the member enters the information for the message and clicks send. The use case ends when the sent message is displayed.

Our class diagram is depicted in **Figure 28**. Whenever a message is sent, the messageController creates a new message in the database via our messageModel, which is in charge of interacting with the database.

User Story #769 - Edit Profile

Tasks

- Create UML diagrams
- Create UI for user profiles
- Edit buttons to edit profile content
- The user's listed spaces are displayed
- User can change his/her password via profile settings
- Unit and Integration tests

Acceptance Criteria

- The user's information is displayed.
- The user's office spaces are shown
- Edit buttons.

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for the entire system is presented in **Figure 1**.

Our sequence diagram is shown in **Figure 31**. The edit profile use case begins when the member clicks an edit button and ends when the newly edited information is displayed.

Our class diagram is depicted in **Figure 30**. The profileController takes care of updating the profileModel and the profileModel is in charge of querying the database which then returns to the profileModel and back to the profileController.

Sprint 7

(04/18/2016 - 05/02/2016)

User Story #850 - Save Credit Card Details***Tasks***

- Testing
- Credit Card Details UI
- UML Diagrams
- Stripe Integration

Acceptance Criteria

- The user can edit their credit card details.
- The credit card details are shown in account settings
- The user can choose to use saved credit card at checkout.

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for message user is presented in **Figure 1**. Our sequence diagram is shown in **Figure 32** and our class diagram is depicted in **Figure 33**.

User Story #851 - Update Bank Account Information***Tasks***

- Testing

- Bank account information form
- UML Diagrams
- Stripe Integration

Acceptance Criteria

- Must enter routing number account number, address.
- Must agree to Stripe TOS.
- Must do this before listing an office space for the first time.

Modeling

Please refer to Appendix A for our UML diagrams. Our use case model for message user is presented in **Figure 1**. Our sequence diagram is shown in **Figure 35** and our class diagram is depicted in **Figure 34**.

System Design

In this section, we begin by discussing the architectural patterns that we used in our system. We then present the system and subsystem decomposition where we describe the functionality provided by each subsystem. Next, we depict our deployment diagram where we illustrate which subsystems will reside on each hardware component and show how the different pieces are connected. Lastly, we conclude with a discussion of our selected design patterns.

Architectural Patterns

The architectural patterns that we used in our system are Model View Controller (MVC) and Client Server. We chose to implement MVC due to it allowing us to separate the business logic, database access, and presentation of our system in a clear and concise manner. Additionally, using MVC reduces the complexity of our code, allows for code reuse, and ensures fewer overall dependencies.

We chose to utilize the client server architecture because our system is a web application and by design, the browser is our client and the application is the server. The client will access the application stored on the server, which prevents the user from having to worry about running anything on the client side; this improves the response time for the user and allows for the scalability of the application.

System and Subsystem Decomposition

Our major subsystems consist of the client and the server **Figure 35**. Within the client we have the browser and within the server we have our models, views, and controllers. The model is used to access the database. It returns all query results to the controller. The controller interacts with both the model and the view. It sends commands to the model to update its state and also sends commands to the view to change its presentation. Our models, views, and controllers can be seen in **Figure 36**.

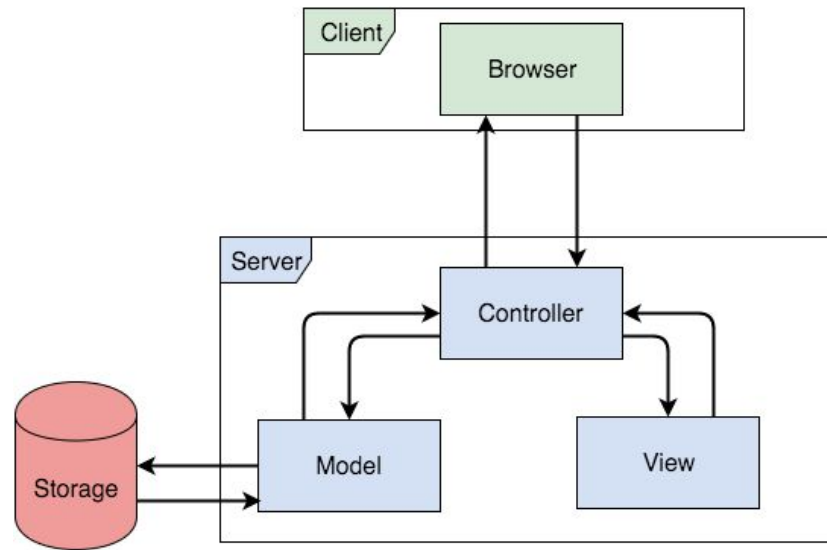


Figure 35 - System and subsystem decomposition

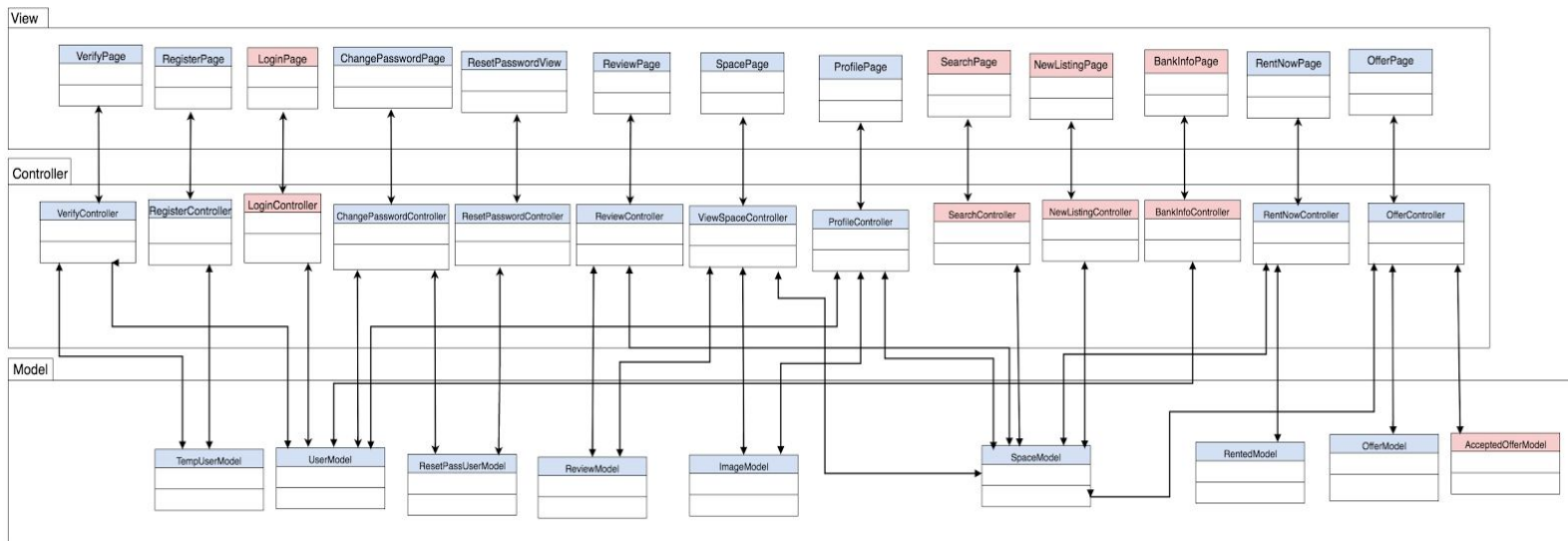


Figure 36 - Models, views, and controllers package diagram.

Deployment Diagram

The DBServer consists of our database, MongoDB. Mongoose is used as the middleware to communicate between the WebServer and DBServer. On the WebServer, we have NodeJS, Express, and Handlebars. Lastly, on the client side, there is the Browser that contains HTML5. Our deployment diagram can be seen below in **Figure 37**.

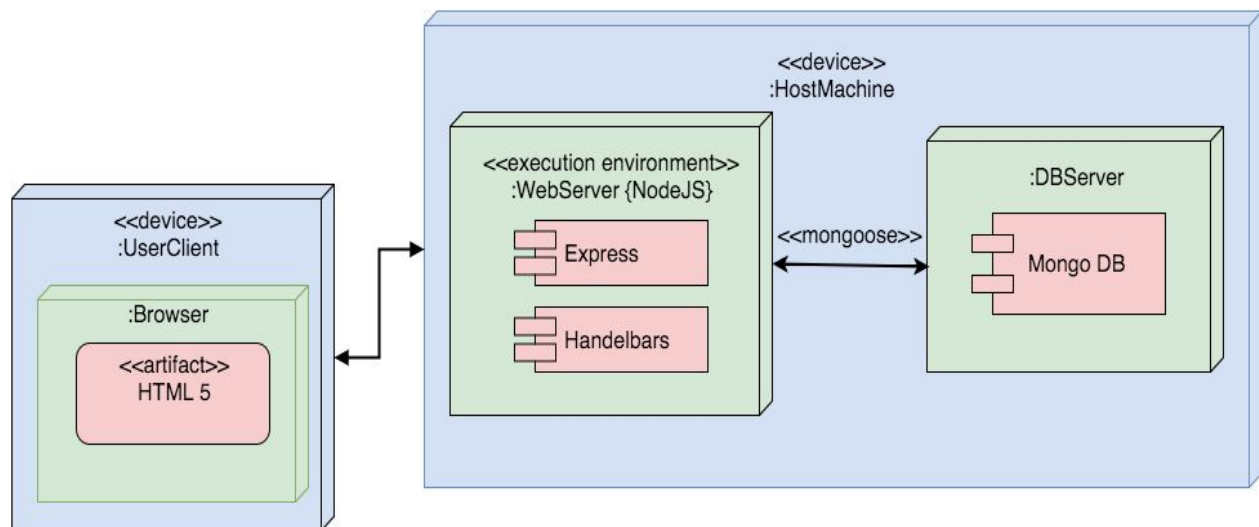


Figure 37 - Deployment Diagram

Design Patterns

The design patterns that we use in our system are dependency injection and singleton.

We are using dependency injection because when we need to use a dependency, we include it by requiring the modules that are needed. In our templates, the dependencies are injected using Handlebars partials. In other words, we incorporate dependencies in partials and then insert them into templates that need to utilize them. Examples include our Stripe.js and Moment.js dependencies in our Offer and RentNow views.

We decided to utilize the singleton pattern due to it allowing us to restrict the instantiations of our classes. For example, all of our models such as Space, Offer, Review and Image only exist as a single instance and are reused in several of our other classes such as ViewSpace, NewListing, and RentNow. This gives us the benefit of keeping our code clean and concise.

System Validation

In this section, we present our system tests and subsystem tests for each user story. We tested all of our user stories using both Selenium web browser automation and manual testing in order to ensure that our system does not contain any faults or defects. Additionally, we utilized the TestNG framework which allowed us to run our tests in arbitrarily big thread pools.

User Story #760 - Log into the System

System Tests:

Purpose: To test the functionality of use case: Login_002: Login.

Test ID: Login_001 (Sunny Day):

Purpose:

- To test the functionality when a user logs in with a valid username and password.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must have a registered account
- The user must be on the login page

Input:

- username: jdoe
- password: 123456

Expected Output:

The user is redirected to the homepage and the "Logout" button is now visible

Test ID: Login_002 (Rainy Day):

Purpose:

- To test the functionality when a user attempts to log in with an invalid username.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user must have a registered account
- The user must be on the login page

Input:

username: test

password: 123456

Expected output:

The message “invalid username” should appear

Test ID: Login_003 (Rainy Day):

Purpose:

- To test the functionality when a user attempts to log in with an invalid password.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user must have a registered account
- The user must be on the login page

Input:

username: jdoe

password: 12345678

Expected output:

The message “invalid password” should appear

Test case 4 (Rainy Day):

Purpose:

- To test the functionality when the user clicks “Login” with empty text fields

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user must have a registered account
- The user must be on the login page

Input:

username:

password:

Expected output:

The message “invalid credentials” should appear

Subsystem Tests

Purpose: To test the functionality of use case: Login_002: Login.

Test ID: Login_004 (Sunny Day):

Purpose:

- To test the functionality when a user logs in and then logs out.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must have a registered account
- The user must be on the login page

Input:

- username: JohnDoe
- password: 123456

Expected Output:

The user is back on the login page after logging in and then logging out.

Test ID: Login_005 (RainyDay):Purpose:

- To test the functionality when a user logs in and then tries to login again by clicking the back button, without logging out.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must have a registered account
- The user must be on the login page

Input:

- username: JohnDoe
- password: 123456

Expected Output:

The user is redirected to the homepage and the **Logout** button is visible

Test ID: Login_006 (Rainy Day):Purpose:

- To test the functionality when a user registers, verifies his/her email, and tries to login with the wrong password.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the register page.

Input:

- first name: John
- last name: Doe
- username: JohnDoe
- email: dalvarez0312@gmail.com

- password: 123456
- confirm password: 123456

Expected output:

The system presents the user with a message “Invalid password.”

User Story #761 - Register for the System

System Tests:

Purpose: To test the functionality of use case: Register_003: Register

Test ID: Register_001 (Sunny Day):

Purpose:

- To test the functionality when a user enters valid data in all of the fields.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the register page.

Input:

- first name: Rachelle
- last name: Tobkes
- username: rtobkes
- email: rtobkes@gmail.com
- password: 1234
- confirm password: 1234

Expected Output:

The user is redirected to the verify notification page and receives an email to verify his/her account.

Test ID: Register_002 (Rainy Day):Purpose:

- To test the functionality when a user attempts to register with a username that already exists

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the register page.

Input:

- first name: Rachelle
- last name: Tobkes
- username: rtobkes
- email: rtobk001@fiu.edu
- password: 1234
- confirm password: 1234

Expected output:

The message "Sorry, that username is already taken" should appear

Test ID: Register_003 (Rainy Day):Purpose:

- To test the functionality when a user attempts to register with an email that already exists

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user must be on the register page

Input:

- first name: Rachelle

- last name: Tobkes
- username: rtobkes24
- email: rtobkes@gmail.com
- password: 1234
- confirm password: 1234

Expected output:

The message “An account with that e-mail already exists.” should appear

Test ID: Register_004 (Rainy Day):

Purpose:

- To test the functionality when the user clicks “Register” with an empty field

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user must be on the Register page

Input:

- first name: Rachelle
- last name:
- username: rtobkes24
- email: rtobk001@fiu.edu
- password: 1234
- confirm password: 1234

Expected output:

The messages “<field> is required” should appear under any field left blank

Test ID: Register_005 (Rainy Day):

Purpose:

- To test the functionality when the user tries to register with a first name that's beyond the allowed length (20 characters).

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user must be on the Register page

Input:

- first name: ThisNameIsTooLongToRegister
- last name: Tobkes
- username: rtobkes24
- email: rtobkes@fiu.edu
- password: 1234
- confirm password: 1234

Expected output:

The messages "name must have up to 20 characters" should appear under each blank field

Subsystem Tests:

Test ID: Register_006 (Sunny Day):

Purpose:

- To test the functionality when a user registers, validates his/her account, logs into the system, and then logs out.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the register page.

Input:

- first name: Rachelle
- last name: Tobkes
- username: rtobkes
- email: rtobk001@fiu.edu

- password: 123456
- confirm password: 123456

Expected output:

The user is back on the login page after he/she logs out.

Test ID: Register_007 (Rainy Day):

Purpose:

- To test the functionality when a user registers, does not validate his/her account, and tries to log into the system.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the register page.

Input:

- first name: Rachelle
- last name: Tobkes
- username: rtobkes
- email: rtobk001@fiu.edu
- password: 123456
- confirm password: 123456

Expected output:

The system presents the user with a message "Invalid username."

Test ID: Register_008 (Rainy Day):

Purpose:

- To test the functionality when a user registers, verifies his/her email, and tries to login with the wrong password.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the register page.

Input:

- first name: Rachelle
- last name: Tobkes
- username: rtobkes
- email: rtobk001@fiu.edu
- password: 123456
- confirm password: 123456

Expected output:

The system presents the user with a message "Invalid password."

User Story #762 - Log out of the system

System Tests:

Test ID: Logout_001 (Sunny Day):Purpose:

- To test the functionality of when a user logs out of his/her account

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the homepage
- The user must be logged in

Input:

- Click logout button

Expected output:

The user is back on the login page after he/she logs out. No logout button should be visible.

Test ID: Logout_002 (Rainy Day):**Purpose:**

- To test the functionality of when the user goes back without logging out

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the homepage
- The user must be logged in

Input:

- Click back button

Expected output:

The user is back on the login page with the logout button visible

Test ID: Logout_003 (Rainy Day):**Purpose:**

- To test the functionality that a user can logout from a different page

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the homepage
- The user must be logged in

Input:

- navigate to new page

Expected output:

The user is on a different page with the logout button visible, when he/she clicks logout they are redirected to the login page

Subsystem Tests:Purpose:

- To test the functionality when a user registers, validates his/her account, logs into the system, and then logs out.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the register page.

Input:

- first name: John
- last name: Doe
- username: JohnDoe
- email: dalvarez0312@gmail.com
- password: 123456
- confirm password: 123456

Expected output:

The user is back on the login page after he/she logs out.

Test ID: Logout_004 (Sunny Day):Purpose:

- To test the functionality when a user logs in and then logs out.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must have a registered account

- The user must be on the login page

Input:

- username: JohnDoe
- password: 123456

Expected Output:

The user is back on the login page after logging in and then logging out.

User Story #763 - List an Office Space

System Tests:

Purpose: To test the functionality of use case: ListSpace_006: List an Office Space

Test ID: ListSpace_001 (Sunny Day):

Purpose:

- To test the functionality when the user navigates to the Add New Space link

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user is on the home page

Input:

- Click Add New Space on the Menu.

Expected Output:

The system displays the form to add a new office space. With the 4 tabs available: Details, Location, Photos, and Done.

Test ID: ListSpace_002 (Sunny Day):

Purpose:

- To test the functionality when the user clicks the Location Tab.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user is on the Add New Space page.

Input:

- click Location Tab

Expected output:

The system shall display a map to the user with a search box to enter their address.

Test ID: ListSpace_003 (Sunny Day):Purpose:

- To test the functionality when the user clicks the Photos Tab.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user is on the Add New Space page.

Input:

- click Photos Tab.

Expected output:

The system displays a template to the user where they can drag and drop up to 7 images.

Test ID: ListSpace_004 (Sunny Day):Purpose:

- To test the functionality when the user drags and drops pictures into the drag and drop template.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user is on the Add New Space page in the Photos Tab.

Input:

- Drag up to 7 images over the drag and drop template and release them.

Expected output:

The system displays a slideshow of the images that were dropped.

Subsystem Tests:

Purpose: To test the functionality of use case: ListSpace_006: List an Office Space

Integrated with other use cases.

Test ID: ListSpace_005 (Sunny Day):

Purpose:

- To test the functionality when the user navigates to the Add New Space link after logging into the system

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user is on the Login Page.

Input:

- Username: JDoe
- Password: 123456
- Click Add New Space on the Menu

Expected Output:

The system displays the form to add a new office space. With the 4 tabs available: Details, Location, Photos, and Done.

Test ID: ListSpace_006 (Sunny Day):

Purpose:

- To test the functionality when the user attempts to add a new space from the search page.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user is on the search page.

Input:

- click Add New Space.

Expected output:

The system displays the form to add a new office space. With the 4 tabs available: Details, Location, Photos, and Done.

User Story #764 - View Office Space

System Tests:

Purpose: To test the functionality of use case: ViewSpace_004: View an Office Space

Test ID: ViewSpace_001 (Sunny Day):

Purpose:

- To test the functionality when the user navigates to the URL viewspace/id.
(id is the id string of the office space that was clicked).

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- A space with the id 571f7bfb7a154e551b893e74 should exist in the database.

Input:

- Navigate to URL viewspace/571f7bfb7a154e551b893e74

Expected Output:

The system displays the information of the specified office space to the user. Including the space details, hours of operation, pricing, features, images, and location on a map.

Test ID: ViewSpace_002 (Rainy Day):

Purpose:

- To test the functionality when the user attempts to navigate to a viewspace URL with a space id that does not exist.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services

Input:

- Navigate to URL viewspace/5000

Expected output:

The user is redirected to the homepage.

Test ID: ViewSpace_003 (Rainy Day):

Purpose:

- To test the functionality when the user attempts to navigate to a viewspace URL with a space id that is not an object id.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services

Input:

- Navigate to URL `viewspace/rrksks`

Expected output:

The user is redirected to the homepage.

User Story #765 - Forgot Password**System Tests:****Test ID: ForgotPass_001 (Sunny Day):**Purpose:

- To test the functionality when a user requests to reset his/her password with a valid registered account and valid new password credentials.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the Sign-in page.
- The user must have a registered account.

Input:

- To request password change:
 - email: rtobk001@fiu.edu
- To change the password
 - new password: Rt12345668
 - confirm password: Rt12345668

Expected output:

The user should receive an email with a link to reset his/her password. When the link is clicked, the user enters the new password and confirms it, and then is redirected to the sign-in page.

Test ID: ForgotPass_002 (Rainy Day):Purpose:

- To test the functionality when a user requests to reset his/her password with an unregistered account.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the Sign-in.

Input:

- email: rtobk003@fiu.edu

Expected output:

The system presents the user with a message "E-mail does not exist."

Test ID: ForgotPass_003 (Rainy Day):Purpose:

- To test the functionality when a user tries to reset his/her password with passwords that do not match.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the Sign-in.
- The user must have a registered account.

Input:

- To request password change:
 - email: rtobk001@fiu.edu
- To change the password
 - new password: rt12345668
 - confirm password: Rt12345668

Expected output:

The system presents the user with a message "Password do not match."

Subsystem Tests:

Test ID: ForgotPass_004 (Sunny Day):

Purpose:

- To test the functionality when a user successfully resets his/her password and then logs in with his/her new password.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the Sign-in page.
- The user must have a registered account.

Input:

- To request password change:
 - email: rtobk001@fiu.edu
- To change the password
 - new password: Rt12345668
 - confirm password: Rt12345668
- To sign in:
 - username: rtobkes
 - password: Rt12345668

Expected output:

The user should be logged in on the homepage, the logout button should be visible.

Test ID: ForgotPass_005 (Rainy Day):

Purpose:

- To test the functionality when a user successfully resets his/her password and attempts to log in with his/her old password.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the Sign-in.
- The user must have a registered account.

Input:

- To request password change:
 - email: rtobk001@fiu.edu
- To change the password
 - new password: Rt12345668
 - confirm password: Rt12345668
- To sign in:
 - username: rtobkes
 - password: 123456

Expected output:

The system presents the user with a message “Invalid password.”

Test ID: ForgotPass_006 (Rainy Day):

Purpose:

- To test the functionality when a user registers and then attempts to reset his/her password without verifying his/her account.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the register page.

Input:

- To register:
 - first name: Rachelle
 - last name: Tobkes
 - username: rachel
 - email: rtobkes@gmail.com

- password: 123456
- confirm password: 123456
- To request password change:
 - email: rtobkes@gmail.com

Expected output:

The system presents the user with a message "Email does not exist."

User Story #759 - Search for Office Space

System Tests:

Purpose: To test the functionality of use case: Search_008: Search for Office Spaces
Test ID: Search_001 (Sunny Day):

Purpose:

- To test the functionality when a user searches for offices spaces by entering their zip code.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the homepage.

Input:

- 33186

Expected Output:

The system displays a list of office spaces within 20 miles of the zip code.

Test ID: Search_002 (SunnyDay):

Purpose:

- To test the functionality when a user makes a search with the name of a city.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user should be on the homepage.

Input:

Miami

Expected output:

The system displays a list of office spaces that are located in Miami.

Test ID: Search_003 (Sunny Day):

Purpose:

- To test the functionality when a user searches for office spaces with a state.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user must be on homepage.

Input:

Florida

Expected output:

The system displays a list of office spaces that are located in Florida.

Test ID: Search_004 (Rainy Day):

Purpose:

- To test the functionality when the user attempts to search with an invalid city/zipcode/state.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user is on the homepage.

Input:

thisIsInvalid

Expected output:

The system displays 0 search results and displays the message “no spaces found”.

Test ID: Search_005 (Rainy Day):

Purpose:

- To test the functionality when the user attempts to search with an empty search field.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user is on the homepage.

Input:

Expected output:

The system displays 0 search results and displays the message “no spaces found”.

User Story #767 - Review an Office Space

System Tests:

Purpose: To test the functionality of use case: WriteReview_007: Review an Office Space

Test ID: WriteReview_001 (Sunny Day):

Purpose:

- To test the functionality when the user reviews an office space with a star rating selected and a written review.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged into the system
- The user should be on the review page for an office space

Input:

- Star selected: 4
- Review: This office space was the best one I've ever rented!!

Expected Output:

The user is redirected to the office space page.

Test ID: WriteReview_002 (Sunny Day):

Purpose:

- To test the functionality when the user reviews an office space with a star selected and an empty review.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged into the system
- The user should be on the review page for an office space

Input:

- Star selected: 3

Expected output:

The user is redirected to the view office space

Test ID: WriteReview_003 (Rainy Day):

Purpose:

- To test the functionality when the user attempts to leave a review without selecting a star rating.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user should be logged into the system
- The user should be on the review page for an office space

Input:

- Review: This office was terrible!!!! There was no WIFI!!!

Expected output:

The system displays "Please select a rating" to the user.

Subsystem Tests:**Test ID: WriteReview_004**Purpose:

- To test the functionality to see if the average rating displayed is updated after a user leaves a review.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the review page for an office space
- The user should be logged into the system
- The review has no ratings

Input:

- Star rating: 4

Expected Output:

The average rating updates to 4.

Test ID: WriteReview_005 (Sunny Day):Purpose:

- To test the functionality when a user leaves a review and then wants to see it.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the review page for an office space
- The user should be logged into the system.

Input:

- Star rating: 5
- Review: This was great!!!!!! LOVED IT!

Expected output:

The user is redirected to the office space page with the review showed as the most recent. The username, date, star rating, and written review is displayed.

Test ID: WriteReview_006 (Sunny Day):

Purpose:

- To test the functionality when the user attempts to leave a review without being logged in.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user is viewing an office space.

Input:

- click: Leave a Review

Expected output:

The system redirects the user to the login page.

User Story #768 - View Office Space Reviews

System Tests:

Purpose: To test the functionality of use case: ViewOfficeReviews_009: View Office Space Reviews/Rating

Test ID: ViewReviews_001 (Sunny Day):Purpose:

- To test the functionality when a user clicks on an office space to read its reviews.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should have searched for office spaces.

Input:

- Click on an office space listing

Expected Output:

The reviews from the office space should be displayed at the bottom of the office space page. It should show the username, date, star rating, and written review for each review.

Test ID: ViewReviews_002 (Sunny Day):

Purpose:

- To test the functionality when a user clicks on an office space to see its average rating.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the review page for an office space

Input:

- Click on an office space listing

Expected output:

The average rating should be displayed towards the top of the page with the number of reviews under it.

Test ID: ViewReviews_003 (Rainy Day):Purpose:

- To test the functionality when the user attempts to see reviews from an office space that has no reviews/ratings.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user is viewing an office space.

Input:

- Click on an office space listing.

Expected output:

The system displays no reviews at the bottom of the page and the average rating should show as 0/5.

Subsystem Tests

Purpose: To test the functionality of use case: ViewOfficeReviews_009: View Office Space Reviews/Rating integrated with other functionalities

Test ID: ViewReviews_004 (Sunny Day):Purpose:

- To test the functionality when a user leaves a review and then views his/her review

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the review page for an office space

Input:

- star rating: 3
- review: It was alright

Expected Output:

The user should be redirected to the office space page after he/she clicks post and the review should be displayed at the bottom of the office space as the most recent.

Test ID: ViewReviews_005 (Rainy Day):

Purpose:

- To test the functionality when a user clicks on an office space when not logged in, and then logs in so he/she can leave the review.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be viewing an office space.

Input:

- Click "Leave a Review Link"

To log in:

- username: rtobkes
- password: 123456

Expected output:

The user is redirected to the login page when he/she clicks "leave a review", then when they sign in, they are brought back to the review page.

Test ID: ViewReviews_006 (Rainy Day):

Purpose:

- To test the functionality when the user attempts to see reviews from an office space that has no reviews/ratings.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user is viewing an office space.

Input:

- Click on an office space listing.

Expected output:

The system displays no reviews at the bottom of the page and the average rating should show as 0/5.

User Story #766 - Make an Offer**System Tests:**

Purpose: To test the functionality of use case: MakeOffer_011: Make an Offer

Test ID: MakeOffer_001 (Sunny Day):

Purpose:

- To test the functionality when the user makes an offer by choosing per hour, and entering all of the fields correctly.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the "Make an Offer" modal

Input:

- Date: 4/21/16
- Hours: 12:00pm-1:00pm, 1:00pm-2:00pm, 2:00pm-3:00pm
- Offer: 500
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 4242424242424242
- Expiration Date: 04/19
- CVC: 564

Expected Output:

The system displays a notification asking the user if he/she wants to proceed with the offer. If the user clicks yes, they are taken back to the space page, otherwise they are taken back to the form.

Test ID: MakeOffer_002 (Sunny Day):Purpose:

- To test the functionality when the user makes an offer by choosing per day, and entering all of the fields correctly.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the “Make an Offer” modal

Input:

- Date: 4/21/16
- Duration: 5
- Offer: 500
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 4242424242424242
- Expiration Date: 04/19
- CVC: 564

Expected Output:

The system displays a notification asking the user if he/she wants to proceed with the offer. If the user clicks yes, they are taken back to the space page, otherwise they are taken back to the form.

Test ID: MakeOffer_003 (Sunny Day):Purpose:

- To test the functionality when the user makes an offer by choosing per month, and entering all of the fields correctly.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the “Make an Offer” modal

Input:

- Date: 4/21/16
- Duration: 2
- Offer: 5000
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 4242424242424242
- Expiration Date: 04/19
- CVC: 564

Expected Output:

The system displays a notification asking the user if he/she wants to proceed with the offer. If the user clicks yes, they are taken back to the space page, otherwise they are taken back to the form.

Test ID: MakeOffer_004 (Rainy Day):Purpose:

- To test the functionality when the user makes an offer by choosing per hour, without selecting any hours.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the "Make an Offer" modal

Input:

- Date: 4/21/16
- Hours:
- Offer: 500
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 4242424242424242
- Expiration Date: 04/19
- CVC: 564

Expected Output:

The system displays an error message: "Please select at least 1 hour."

Test ID: MakeOffer_005 (Rainy Day):Purpose:

- To test the functionality when the user makes an offer by choosing per day, with an offer that is less than 75% of the Rent Now price.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the "Make an Offer" modal

Input:

- Date: 4/21/16
- Duration: 3
- Offer: 5
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 4242424242424242
- Expiration Date: 04/19
- CVC: 564

Expected Output:

The system displays an error message: "Sorry, your offer is too low."

Subsystem Tests:

Test ID: MakeOffer_006 (Sunny Day):

Purpose:

- To test the functionality when the user views an office space and then attempts to make an offer while being logged in.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on an office space page.

Input:

- Click: Make an Offer

Expected Output:

The system opens the make an offer form/modal.

Test ID: MakeOffer_007 (Sunny Day):

Purpose:

- To test the functionality when the user searches for an office space, views it, and attempts to make an offer while being logged in.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user is on the homepage.

Input:

- Search: Miami

- Click: Wynwood Office Space
- Click: Make an Offer

Expected Output:

The system opens the make an offer form/modal.

Test ID: MakeOffer_008 (RainyDay):

Purpose:

- To test the functionality when the user attempts to make an offer without being logged in.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the view space page for an office space.

Input:

- Click: Make an Offer

Expected Output:

The system redirects the user to the login page. Once the user is logged in, they are taken back the office space page that they were originally at.

User Story #772 - Filter Results

System Tests:

Purpose: To test the functionality of use case: FilterResults_012: Filter Search Results

Test ID: Filter_001 (Sunny Day):

Purpose:

- To test the functionality when the user selects the price filter.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on search results page.

Input:

- Click: Price filter
- Enter \$50 for minimum
- Enter \$100 for maximum

Expected Output:

The system updates the search result list and only displays office spaces that cost between \$50 and \$100 per day.

Test ID: Filter_002 (Sunny Day):Purpose:

- To test the functionality when the user selects the rating filter.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on search results page.

Input:

- Click: Rating filter
- Click: 4 stars

Expected Output:

The system updates the search result list and only displays office spaces that have at least 4 stars.

Test ID: Filter_003 (Sunny Day):Purpose:

- To test the functionality when the user selects the desks filter.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on search results page.

Input:

- Click: Desks filter
- Click: 10 desks

Expected Output:

The system updates the search result list and only displays office spaces that have at least 10 desks.

Test ID: Filter_004 (Rainy Day):Purpose:

- To test the functionality when the user filters search when the search field is empty.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on search results page.

Input:

- Click: Price
- \$70

Expected Output:

The empty search list should remain empty and not display any results.

Subsystem Tests:

Purpose: To test the functionality of use case: FilterResults_012: Filter Search Results

Test ID: Filter_005 (Sunny Day):Purpose:

- To test the functionality when the user searches by location, and then selects a filter.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the homepage.

Input:

- Search: Coral Gables
- Click: Price Filter
- Click \$50

Expected Output:

The system first displays all offices in the coral gables area. Then, updates the search result list and only displays office spaces that cost under \$50 per day when the filter is chosen.

Test ID: Filter_006 (Sunny Day):Purpose:

- To test the functionality when the user views an office space, then goes back to the search page and chooses a filter.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services

- The user should be on search results page.

Input:

- Click an office space
- Click: back button
- Click filter: Rating
- Click: 4 stars

Expected Output:

The system updates the search result list and only displays office spaces that have at least 4 stars.

Test ID: Filter_007 (Rainy Day):Purpose:

- To test the functionality when the user searches a nonexistent location and then chooses a filter.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on search results page.

Input:

- Search: hfksksjd
- Click: Filter Desks
- Click: 10 desks

Expected Output:

The list remains empty.

User Story #818 - Accept/Decline Offers**System Tests:**

Purpose: To test the functionality of use case: Accept_Decline013: Accept/Decline Offer

Test ID: AcceptDecline_001 (Sunny Day):Purpose:

- To test the functionality when the user accepts one of their offers.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services

- The user should be logged in.
- The user should have new offers available.
- The user should be on the pending offers tab.

Input:

- Click: Accept

Expected Output:

The offer is moved from Pending to Accepted. The availability for the office space is updated.

Test ID: AcceptDecline_002 (Sunny Day):

Purpose:

- To test the functionality when the user declines an offer.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged in.
- The user should have new offers available.
- The user should be on the pending offers tab.

Input:

- Click: Decline

Expected Output:

The system removes the offer from the offer view. The availability for the office space is unchanged.

Test ID: AcceptDecline_003 (Rainy Day):

Purpose:

- To test the functionality when the user attempts to accept an offer that has already been accepted.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged in.
- The user should have previously accepted an offer.
- The user should be on the pending offers tab.

Input:

Expected Output:

The offer is no longer visible under the Pending Offers tab. Once a user accepts an offer, they do not have the option to accept it again.

Subsystem Tests

Purpose: To test the functionality of use case: Accept_Decline013: Accept/Decline Offer integrated with other features.

Test ID: AcceptDecline_004 (Sunny Day):

Purpose:

- To test the functionality when the user logs in, views his/her offers, and then accepts one of their offers.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the Sign-in.
- The user should have new offers available.

Input:

- Username: dalva
- Password: 123456
- Click: Menu
- Click: My Offers
- Click: Recieved Offers
- Click: Pending
- Click: Accept

Expected Output:

The offer is moved from Pending to Accepted. The availability for the office space is updated.

Test ID: AcceptDecline_005 (Sunny Day):

Purpose:

- To test the functionality when the user logs in, views his/her offers, and then declines an offer.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be the Sign-in page.
- The user should have new offers available.

Input:

- Username: dalva
- Password: 123456
- Click: Menu
- Click: My Offers
- Click: Recieved Offers
- Click: Pending
- Click: Decline

Expected Output:

The system removes the offer from the offer view. The availability for the office space is unchanged.

Test ID: AcceptDecline_006 (Rainy Day):Purpose:

- To test the functionality when the user attempts to accept an offer without being logged in.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services

Input:

- Click: Menu

Expected Output:

The My Offers option is not available in the menu.

User Story #830 - Rent Now**System Tests:**

Purpose: To test the functionality of use case: RentNow_015: Rent Now

Test ID: RentNow_001 (Sunny Day):Purpose:

- To test the functionality when the user rents an office space by choosing per hour, and entering all of the required fields correctly.

Precondition:

- The user should have access to the application

- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the “Rent Now” modal

Input:

- Date: 4/25/16
- Hours: 11:00am-12:00pm, 2:00pm-3:00pm
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 4242424242424242
- Expiration Date: 09/22
- CVC: 564

Expected Output:

The system displays a notification asking the user if he/she wants to proceed with the transaction. If the user clicks yes, they are taken back to the space page, otherwise they are taken back to the form. Also, the user receives an email regarding the reservation details.

Test ID: RentNow_002 (Sunny Day):

Purpose:

- To test the functionality when the user rents an office space by choosing per day, and enters all of the fields correctly.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the “Rent Now” modal

Input:

- Date: 4/27/16
- Duration: 2
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 4242424242424242
- Expiration Date: 09/22
- CVC: 564

Expected Output:

The system displays a notification asking the user if he/she wants to proceed with their transaction. If the user clicks yes, they are taken back to the space page, otherwise they are taken back to the form. Also, the user receives an email with the transaction details if they click yes.

Test ID: RentNow_003 (Sunny Day):Purpose:

- To test the functionality when the user rents an office space by choosing per month, and entering all of the fields correctly.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the “Rent Now” modal

Input:

- Date: 4/27/16
- Duration: 2
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 4242424242424242
- Expiration Date: 009/22
- CVC: 564

Expected Output:

The system displays a notification asking the user if he/she wants to proceed with the transaction. If the user clicks yes, they are taken back to the space page, otherwise they are taken back to the form. Also, the user receives an email with the transaction details if they click yes.

Test ID: RentNow_004 (Rainy Day):Purpose:

- To test the functionality when the user rents an office space per day without selecting a date from the calendar.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the “Rent Now” modal

Input:

- Duration: 2
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 4242424242424242
- Expiration Date: 09/22
- CVC: 564

Expected Output:

The system displays the error message: "Please correct the highlighted fields."

Test ID: RentNow_005 (Rainy Day):Purpose:

- To test the functionality when the user rents an office space per hour with an invalid credit card number.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the "Rent Now" modal

Input:

- Date: 4/25/16
- Hours: 11:00am-12:00pm, 2:00pm-3:00pm
- Name on Card: Rachelle Ellen Tobkes
- Card Number: 6473837484738272
- Expiration Date: 09/22
- CVC: 564

Expected Output:

The system displays the error message: "The credit card appears to be invalid".

Subsystem Tests:

Purpose: To test the functionality of use case: RentNow_015: Rent Now

Test ID: RentNow_006 (Sunny Day):Purpose:

- To test the functionality when the user rents an office space while being logged in.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the view for an office space.

Input:

- Click: Rent Now

Expected Output:

The system opens the Rent Now modal that contains the Rent Now form.

Test ID: RentNow_007 (Sunny Day):Purpose:

- To test the functionality when the user searches for an office space, views an office space, and then rents the office space.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user is on the search page.
- The user is logged in

Input:

- Search criteria: Miami, Florida
- Click: first listed space
- Click: Rent Now

Expected Output:

The system opens the Rent Now modal that contains the Rent Now form.

Test ID: RentNow_008 (Rainy Day):Purpose:

- To test the functionality when the user attempts to rent an office space without being logged in.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on the view for an office space.

Input:

- Click: Rent Now

Expected Output:

The system should redirect the user to the login page. Once he/she logs in, they are taken back to the office space page of which they originally wanted to rent.

User Story #841 - Message Users**System Tests:****Test ID: Message_001 (Sunny Day):**Purpose:

- To test the functionality of when a user sends a valid message to another user.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must have the message modal opened.
- The user must be logged in

Input:

- Username: rachellet
- Message: "Hey! I want to rent your office space."

Expected output:

The message is displayed as sent and the receiver receives it in their inbox under received.

Test ID: Message_002 (Sunny Day):Purpose:

- To test the functionality of when the user sends consecutive messages to another user.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must have the message modal opened.
- The user must be logged in

Input:

- Username: rachellet

- Message: “hey!”
- Message: “I want to rent your office space.”

Expected output:

Both messages are shown as sent, and the receiver receives the 2 messages in the correct order.

Test ID: Message_003 (Rainy Day):

Purpose:

- To test the functionality when the user attempts to send a message to himself

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must have the message modal opened.
- The user must be logged in

Input:

- Username: dalva
- Message: “Hey! I want to rent your office space.”

Expected output:

The system notifies the user that he cannot send a message to himself.

Test ID: Message_004 (Rainy Day):

Purpose:

- To test the functionality when the user sends a blank message to another user.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must have the message modal opened.

- The user must be logged in

Input:

- Username: rachellet
- Message:

Expected output:

The system notifies the user that the message cannot be blank.

Subsystem Tests

Test ID: Message_005 (Sunny Day):

Purpose:

- To test the functionality of when a user sends a valid message to another user after searching for an office space, viewing an office space, and viewing the user's profile.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the search page.
- The user must be logged in

Input:

- Search: Miami, Florida
- Username: rachellet
- Message: "Hey! I want to rent your office space."

Expected output:

The message is displayed as sent and the receiver receives it in their inbox.

Test ID: Message_006 (Sunny Day):

Purpose:

- To test the functionality of when the user logs in and then sends a message to another user.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the sign in page

Input:

- Username: dalva
- Password: 123456
- Username: rachellet
- Message: "I want to rent your office space."

Expected output:

The message is displayed as sent and the receiver receives it in his/her inbox.

Test ID: Message_007 (Rainy Day):

Purpose:

- To test the functionality of when a user tries to send a message without being logged in.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services

Input:

Expected output:

The MailBox option is not shown in the menu

User Story #769 - Edit Profile**System Tests:**

Purpose: To test the functionality of use case: EditProfile_003: Edit Profile

Test ID: EditProfile_001 (Sunny Day):

Purpose:

- To test the functionality when the user edits his/her name

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged in.
- The user should be on their profile page.

Input:

- Click: Edit name icon
- Enter: Rachelle Ellen Tobkes
- Click: Save name icon

Expected Output:

The system updates and displays the user's name as "Rachelle Ellen Tobkes."

Test ID: EditProfile_002 (Sunny Day):

Purpose:

- To test the functionality when the user edits his/her profile picture.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged in.
- The user should be on their profile page.

Input:

- Click: Edit photo icon
- Select: new_image.jpg
- Click: Save photo icon

Expected Output:

The system updates and displays the user's profile picture as new_image.jpg.

Test ID: EditProfile_003 (Sunny Day):Purpose:

- To test the functionality when the user edits his/her username.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged in.
- The user should be on their profile page

Input:

- Click: Edit username icon
- Enter: rtobkes94
- Click: Save username icon

Expected Output:

The username is updated and displayed as "rtobkes94."

Test ID: EditProfile_004 (Rainy Day):Purpose:

- To test the functionality when the user edits his/her username to a username that already exists.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged in.
- The user should be on their profile page

Input:

- Click: Edit username icon
- Enter: Dalva
- Click: Save username icon

Expected Output:

The system displays the message "Sorry that username is already taken."

Test ID: EditProfile_005 (Rainy Day):Purpose:

- To test the functionality when the user edits his/her username to an invalid username.

Precondition:

- The user should have access to the application

- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged in.
- The user should be on their profile page

Input:

- Click: Edit username icon
- Enter: rtobkes 94
- Click: Save username icon

Expected Output:

The system displays the message "Sorry, your username cannot have whitespace."

Subsystem Tests:

Purpose: To test the functionality of use case: EditProfile_003: Edit Profile

Test ID: EditProfile_006 (Sunny Day):Purpose:

- To test the functionality when the user views a space they've posted, navigates to their profile from the link on the space page, and then edits their profile.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged in.
- The user should be viewing a space that they have posted.

Input:

- Click: View owner's profile
- Click: Edit name icon
- Enter: Rachelle Ellen Tobkes
- Click: Save name icon

Expected Output:

The system displays icons to edit the user's information. When a field is edited, the system updates and displays the user's name as "Rachelle Ellen Tobkes."

Test ID: EditProfile_007 (Sunny Day):Purpose:

- To test the functionality when the user logs in, navigates to his/her profile, and then edits their profile.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user is on the Sign-in page.

Input:

- Username: rtobkes
- Password: 123456
- Click: Profile
- Click: Edit Bio icon
- Enter: "I love senior project!"
- Click: Save bio icon

Expected Output:

The system displays the edit icons on the user's profile. Once the information is saved, the system updates and displays the user's bio as "I love senior project!"

Test ID: EditProfile_008 (Rainy Day):

Purpose:

- To test the functionality when the user wants to edit his/her profile without being logged in.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be on their profile page.

Input:

Expected Output:

No edit icons are visible on the user's profile page.

Test ID: EditProfile_009 (Rainy Day):

Purpose:

- To test the functionality when the user attempts to edit another user's profile.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user should be logged in.
- The user should be on someone else's profile page.

Input:

Expected Output:

The system does not display any edit icons; therefore, the user cannot edit another person's profile.

User Story #850 - Save Card Details**System Tests:**

Purpose: To test the functionality of use case: Save_Card019: Save Card Details

Test ID: SaveCard_001 (Sunny Day):Purpose:

- To test the functionality when the user saves their credit card details for the first time.

Precondition:

- The user should have access to the application
- The user should be logged in.
- The user should be on their account settings.

Input:

- Click: Card Details
- Name: Rachelle Tobkes
- Card Number: 4242424242424242
- Expiration: 09/22
- CVC: 232

Expected Output:

The system displays the message "Your details have been saved."

Test ID: SaveCard_002 (Sunny Day):Purpose:

- To test the functionality when the user views their saved card details after saving.

Precondition:

- The user should have access to the application
- The user should be logged in.
- The user should be on their account settings.

Input:

- Click: Card Details

Expected Output:

The system displays the user's last 4 from their credit card, in this case 4242 and the type of the card, which is Visa here. Also, the system displays "Edit card details" message with check box.

Test ID: SaveCard_003 (Sunny Day):Purpose:

- To test the functionality when the user updates their saved card details

Precondition:

- The user should have access to the application
- The user should be logged in.
- The user should be on their account settings.

Input:

- Click: Card Details
- Click: check box
- Name: Rachelle Tobkes
- Card Number: 5555555555554444
- Expiration: 09/22
- CVC: 232

Expected Output:

The system notifies the user that their card details have been updated/saved.

Test ID: SaveCard_004 (Rainy Day):Purpose:

- To test the functionality when the user attempts to save card details with an invalid credit card number.

Precondition:

- The user should have access to the application
- The user should be logged in.
- The user should be on their account settings.

Input:

- Click: Card Details
- Click: check box
- Name: Rachelle Tobkes
- Card Number: 1234567891234567
- Expiration: 09/22
- CVC: 232

Expected Output:

The credit card box is highlighted in red.

Subsystem Tests

Purpose: To test the functionality of use case: Save_Card019: Save Card Details integrated with other features.

Test ID: SaveCard_005 (Sunny Day):Purpose:

- To test the functionality when the user saves their credit card details after viewing their profile page.

Precondition:

- The user should have access to the application
- The user should be logged in.

Input:

- Click: My Profile Link
- Click: Account Settings
- Click: Card Details
- Name: Rachelle Tobkes
- Card Number: 4242424242424242
- Expiration: 09/22
- CVC: 232

Expected Output:

The system displays the message "Your details have been saved."

Test ID: SaveCard_006 (Rainy Day):Purpose:

- To test the functionality when the user attempts to save card details without being logged in.

Precondition:

- The user should have access to the application
- The user should be on the homepage

Input:Expected Output:

There is no "Account Settings" visible since the user is not logged in and therefore, they cannot save/update card details

User Story #851 - Save Bank Account Information**System Tests:**

Purpose: To test the functionality of use case: (Bank_Account) - Set Up Bank Acc.

Test ID: SaveBank_001 (Sunny Day):Purpose:

- To test the use case when the user sets up his/her bank account with valid credentials.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services
- The user must be on the bank account info page

Input:

- Street address: 3119 Maple Lane
- City: Miami
- Zip Code: 33328
- State: Florida
- DOB: 09/22/1994
- Last 4 ssn: 4545
- Routing Number: 110000000
- Account Number: 000123456789

Expected Output:

The user is redirected to the New Listing page.

Test ID: SaveBank_002 (Rainy Day):Purpose:

- To test the functionality when a user attempts to set up their bank account with an invalid account number.

Precondition:

- The user should have access to the application

- The database needs to properly bind to the application and accessible to the web services
- The user must be on the bank account page

Input:

- Street address: 3119 Maple Lane
- City: Miami
- Zip Code: 33328
- State: Florida
- DOB: 09/22/1994
- Last 4 ssn: 4545
- Routing Number: 110000000
- Account Number: 111

Expected output:

The account number text box is highlighted in red.

Test ID: SaveBank_003 (Rainy Day):

Purpose:

- To test the functionality when a user attempts to set up their bank account with an empty field.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services
- The user must be on the bank account page.

Input:

- Street address: 3119 Maple Lane
- City: Miami
- Zip Code: 33328
- State: Florida
- DOB:

- Last 4 ssn: 4545
- Routing Number: 110000000
- Account Number: 000123456789

Expected output:

The system highlights the DOB text box in red.

Subsystem Tests

Test ID: SaveBank_004 (Sunny Day):

Purpose:

- To test the functionality when a user attempts to list a new space with already having a bank account set up.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and accessible to the web services

Input:

- Click: List New Space

Expected output:

The user is taken to the new listing page.

Test ID: SaveBank_005 (Rainy Day):

Purpose:

- To test the use case when the user attempts to list a new space without having set up their bank account.

Precondition:

- The user should have access to the application
- The database needs to properly bind to the application and be accessible to the web services

Input:

- Click: List New Space

Expected Output:

The system should redirect the user to the bank account information page.

Glossary

Client/Server Architecture - An architectural pattern where the server provides services to instances of other subsystems called the clients, which are responsible for interacting with the user.

Deployment diagram - Represents runtime components and their assignments to hardware nodes

Design pattern - Provides a schema for refining the subsystems or components of a software system, or relations between them.

Lister - A person who lists an office space on the HyperDesk system

Member - A registered user for the HyperDesk System. Inherits the functionalities of the user.

Persistent data - Data that outlives a single execution of the system.

Renter - A person who is renting an office space on the HyperDesk system

Sequence diagram - An interaction diagram that emphasizes the time-ordering of messages.

Subsystem - Is a well-defined software component that provides a number of services to other subsystems, like manages persistent data, the interaction with the user, and the communication with other subsystems over the network.

User - A general user for the HyperDesk system.

Appendix

Appendix A - UML Diagrams

Static UML Diagrams

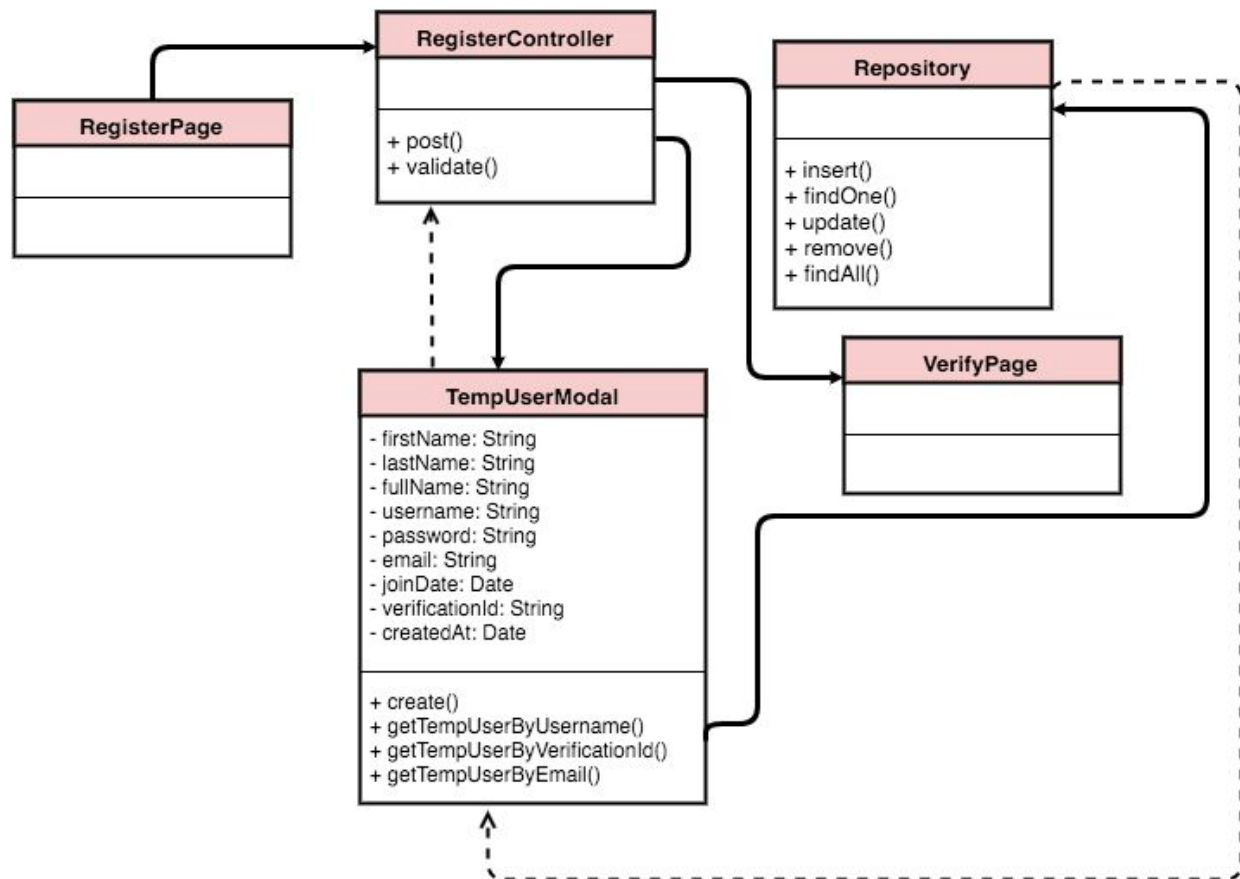


Figure 3 - Register class diagram User Story #761

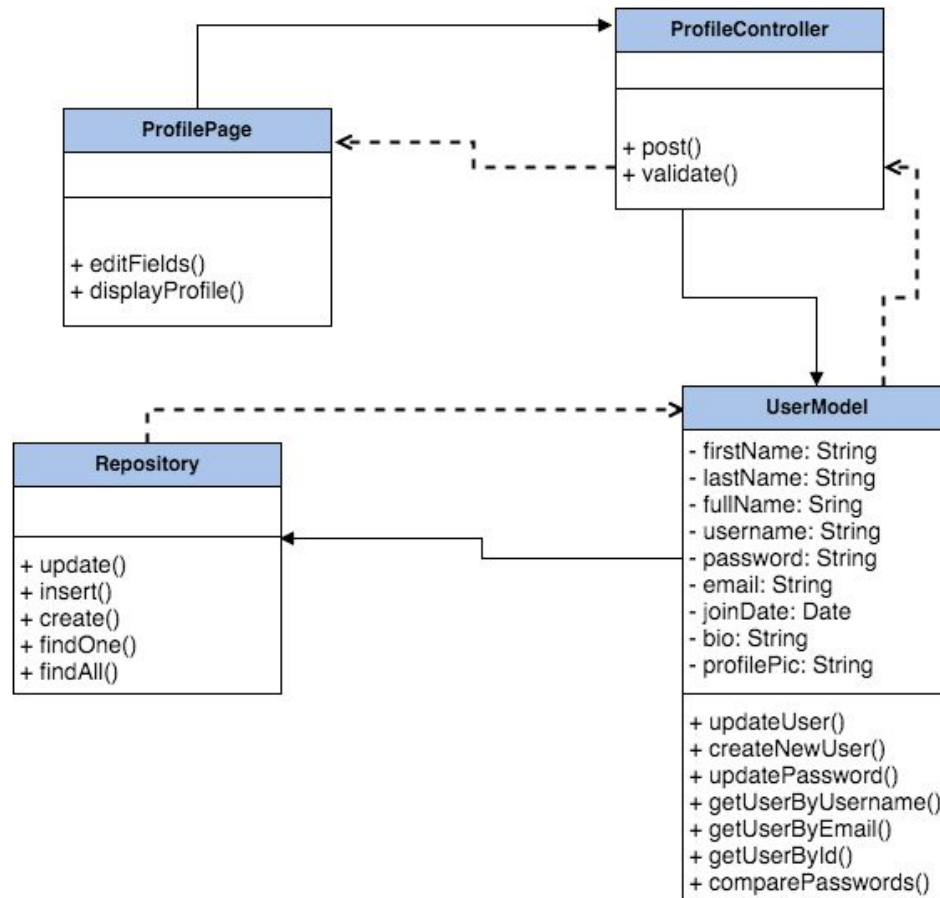


Figure 30 - Edit Profile class diagram for User Story #769

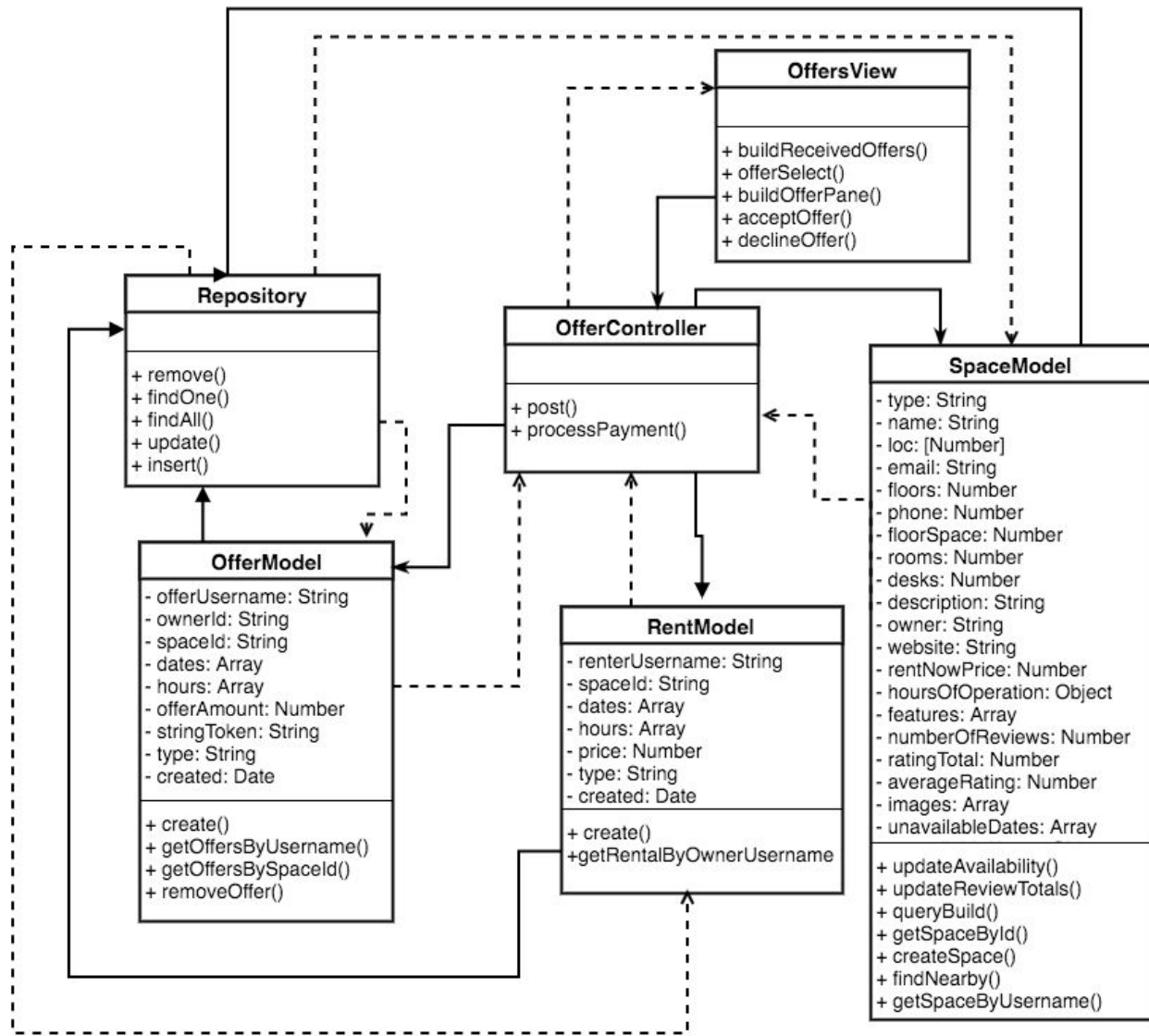


Figure 27- Accept offer class diagram for User Story #818

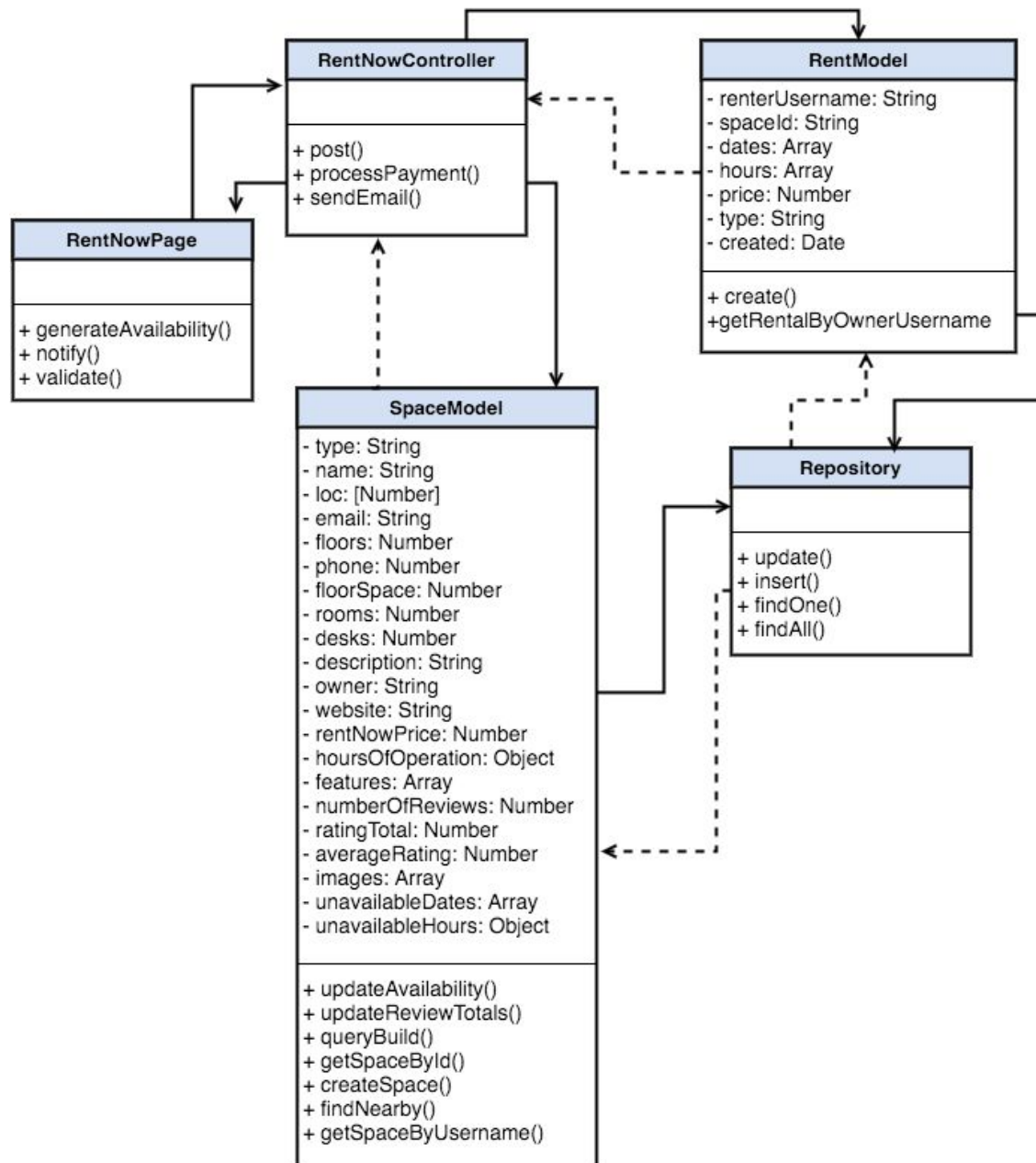


Figure 25 - Rent now class diagram for User Story #830

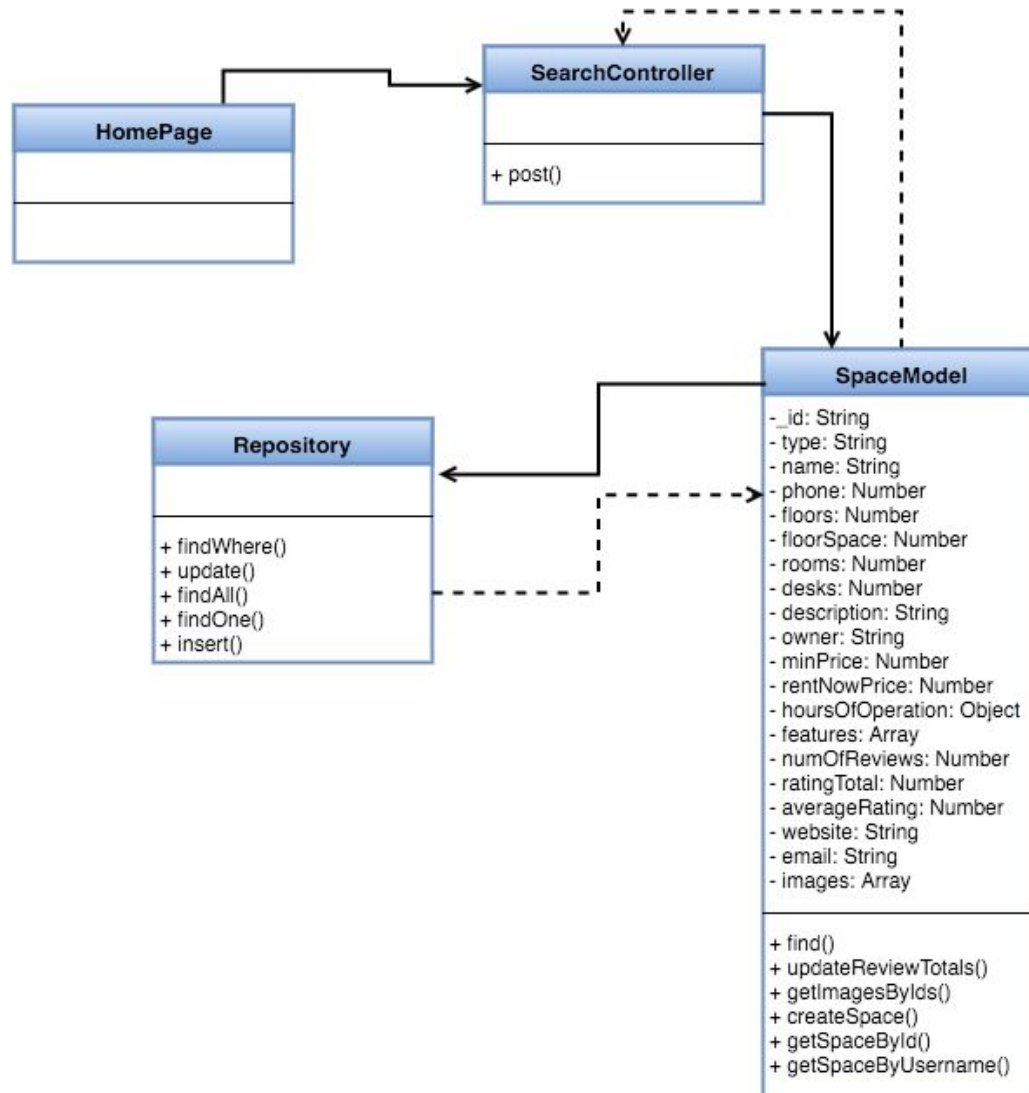


Figure 23 - Filter results class diagram for User Story #772

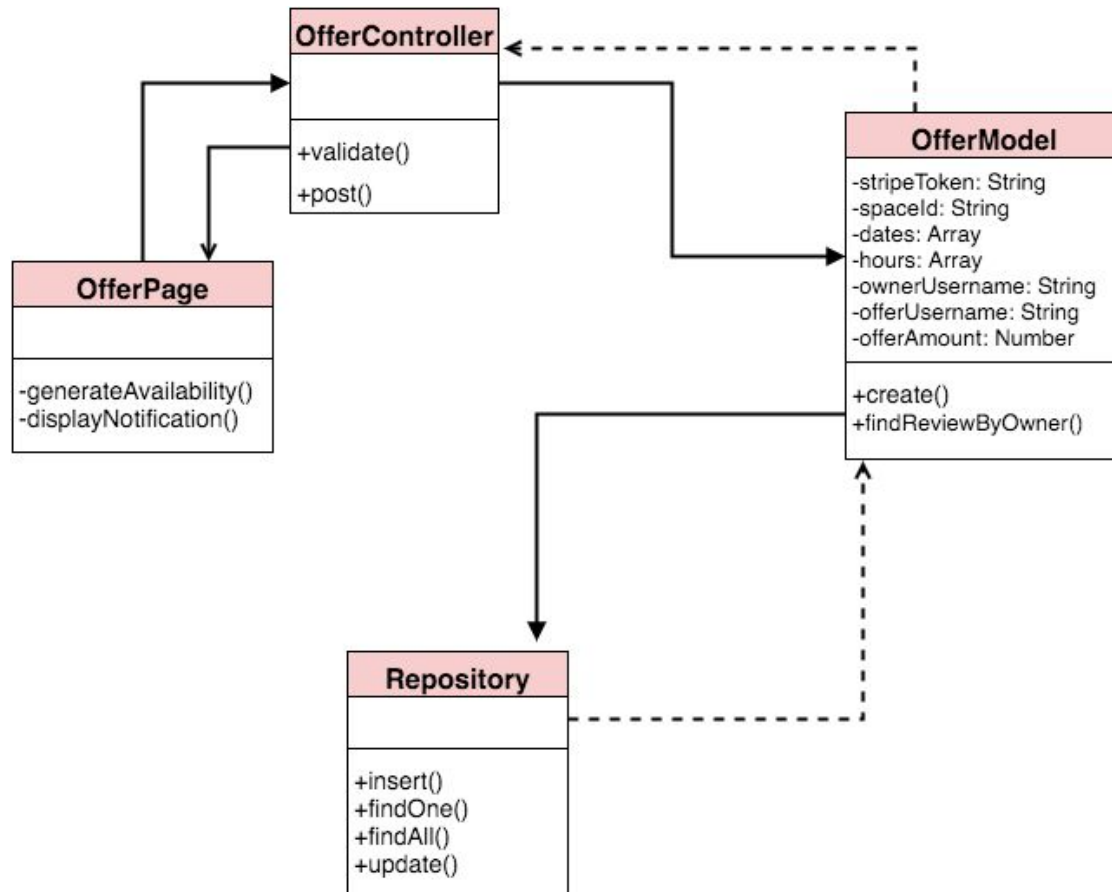


Figure 21 - Make an Offer class diagram for User Story #766

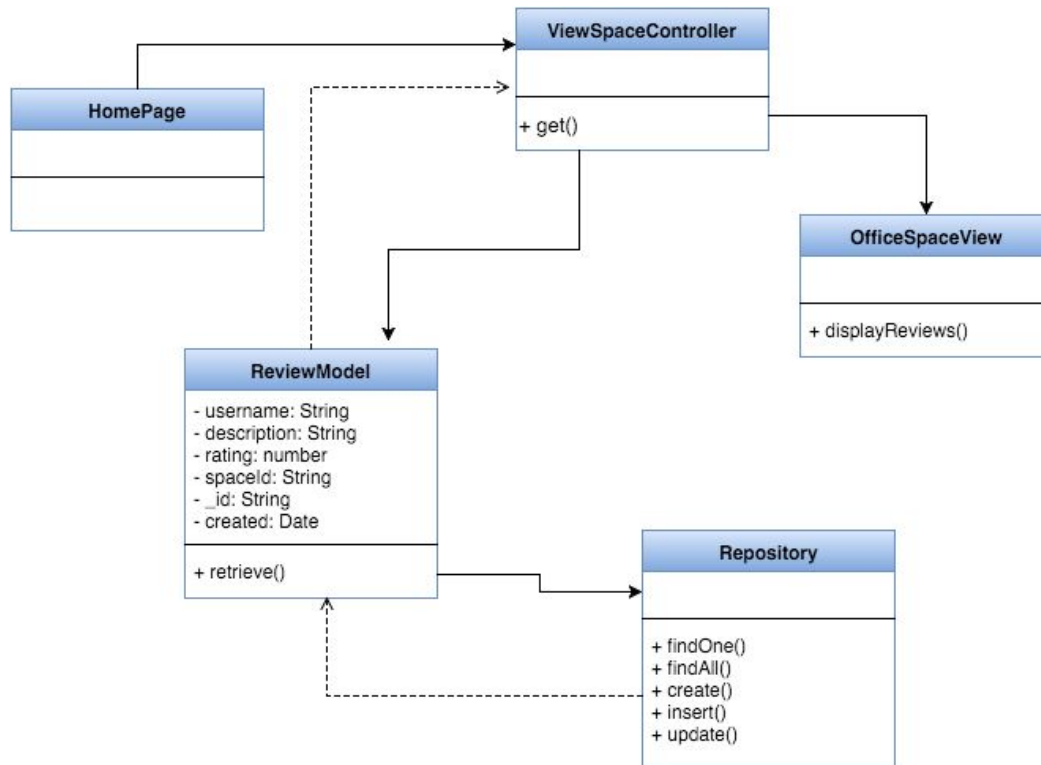


Figure 19 - View Reviews class diagram for User Story #768

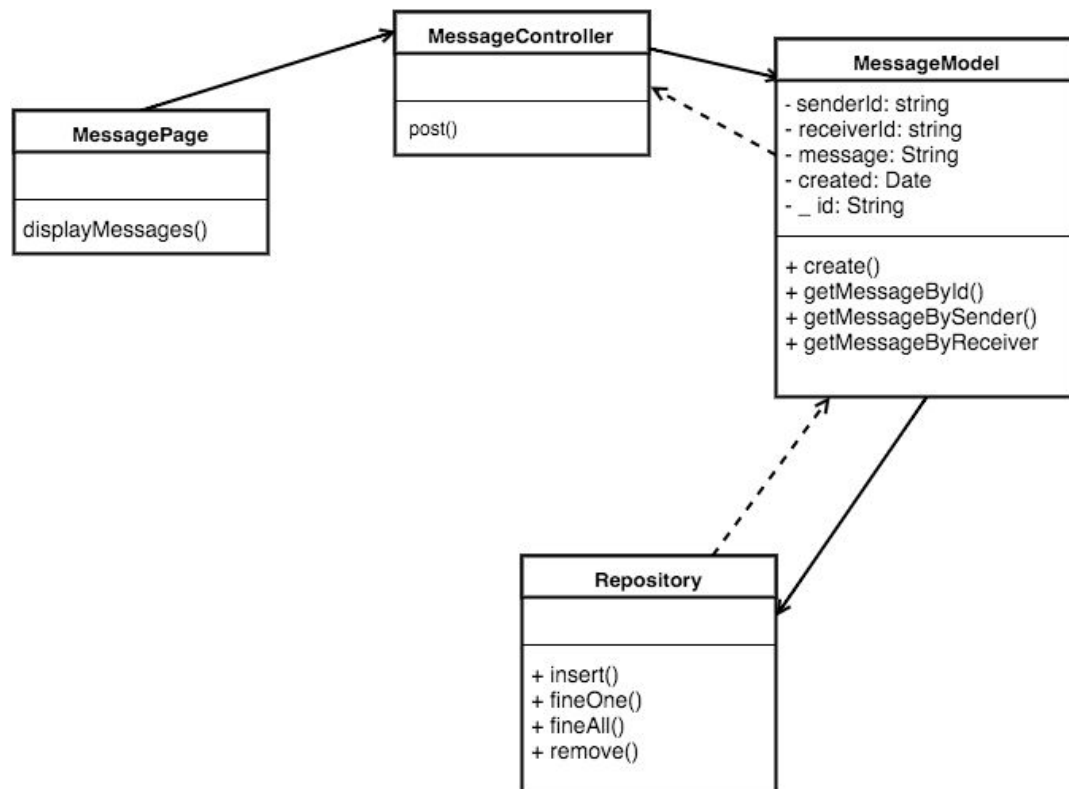


Figure 28 - Message User class diagram for User Story #841

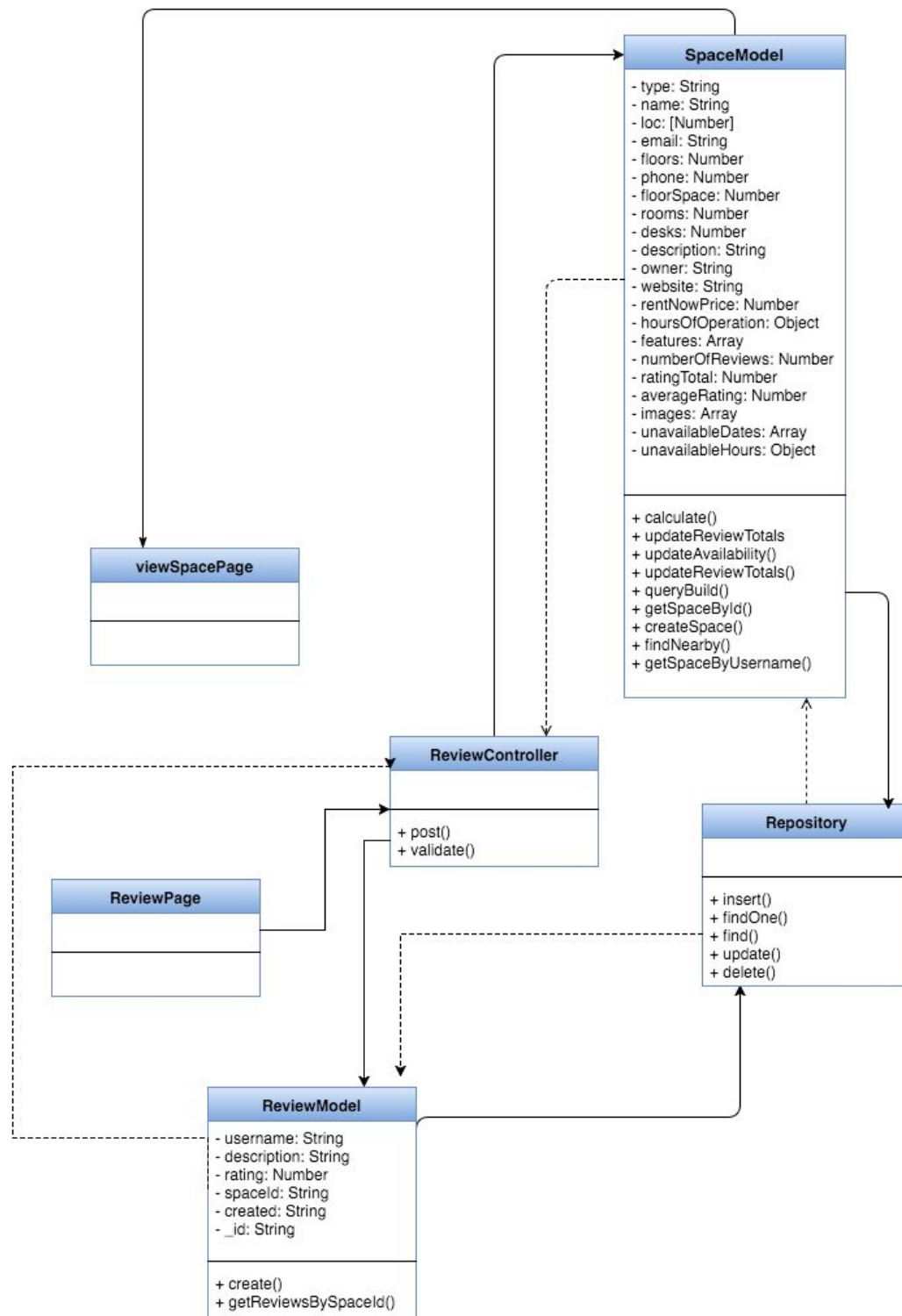


Figure 17 - Write Review class diagram for User Story #767

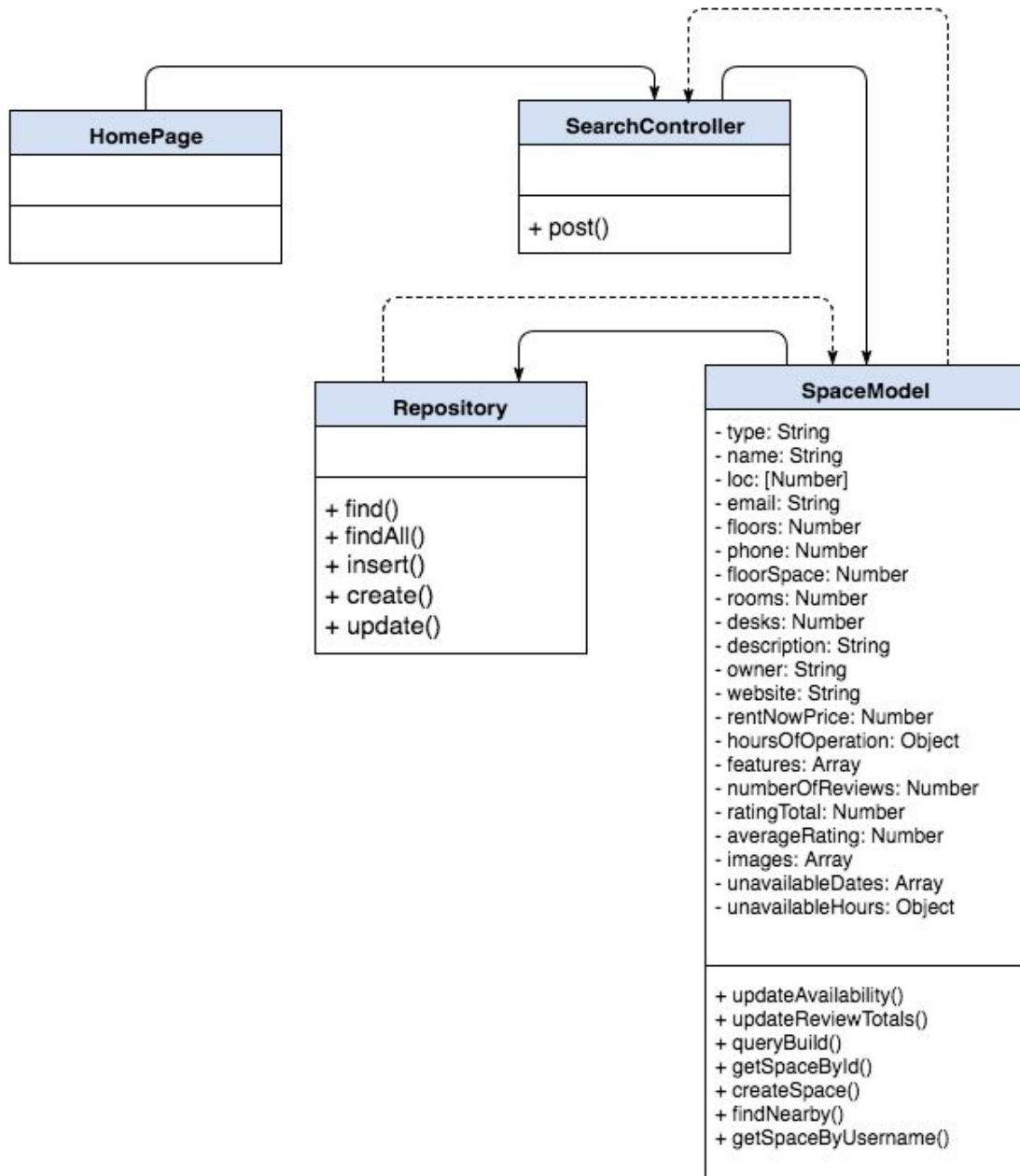


Figure 15 - Search class diagram for User Story #769

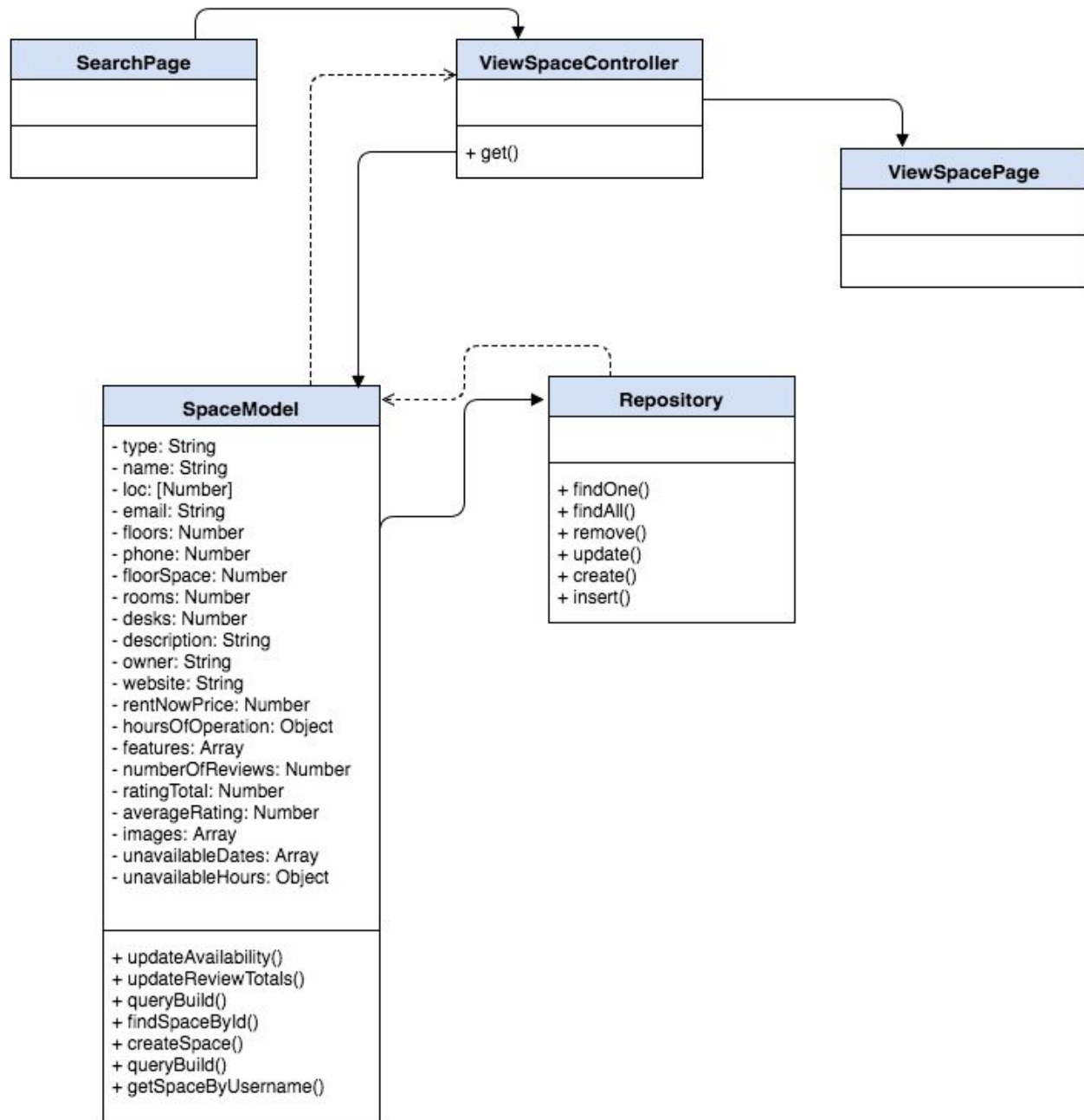


Figure 10 - View Office Space class diagram for User Story #764

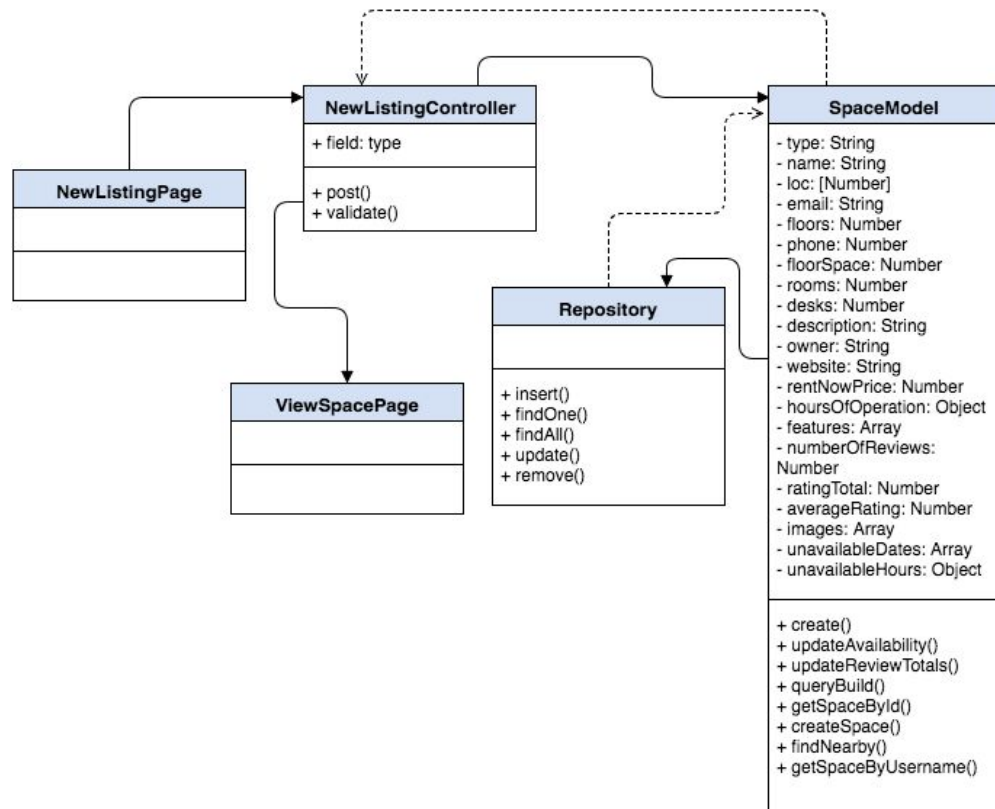


Figure 8 - New Listing class diagram for User Story #763

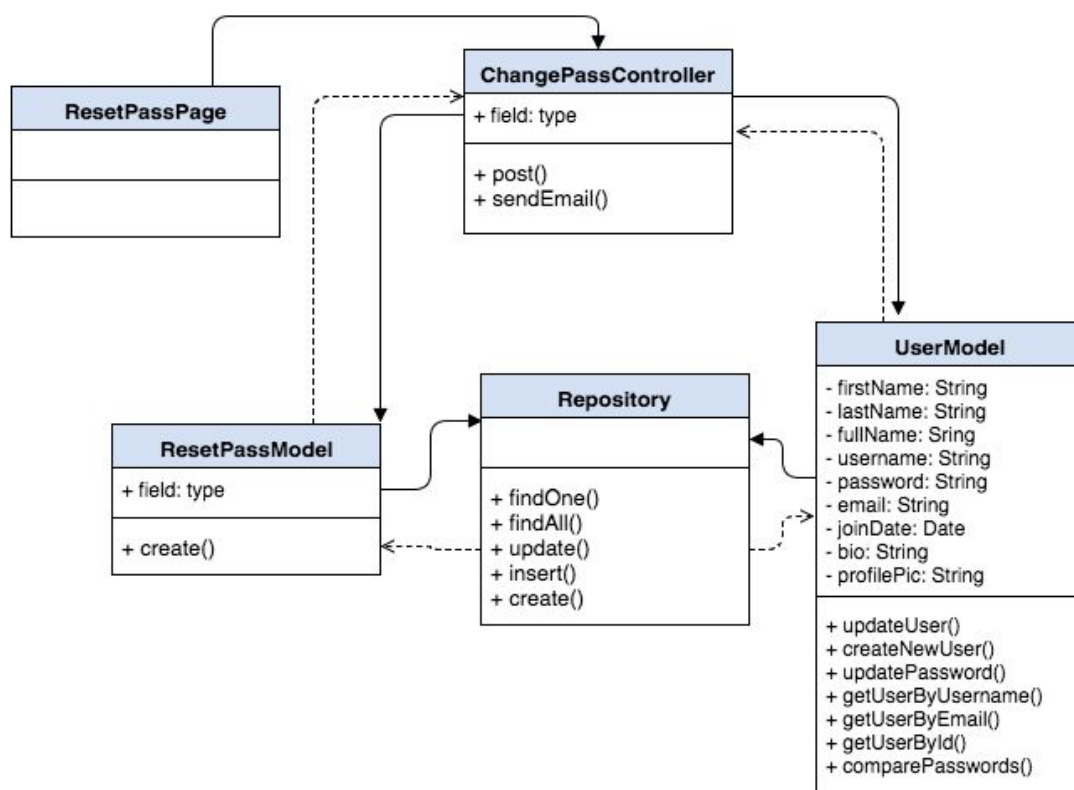


Figure 13 - Forgot Password class diagram for User Story #765

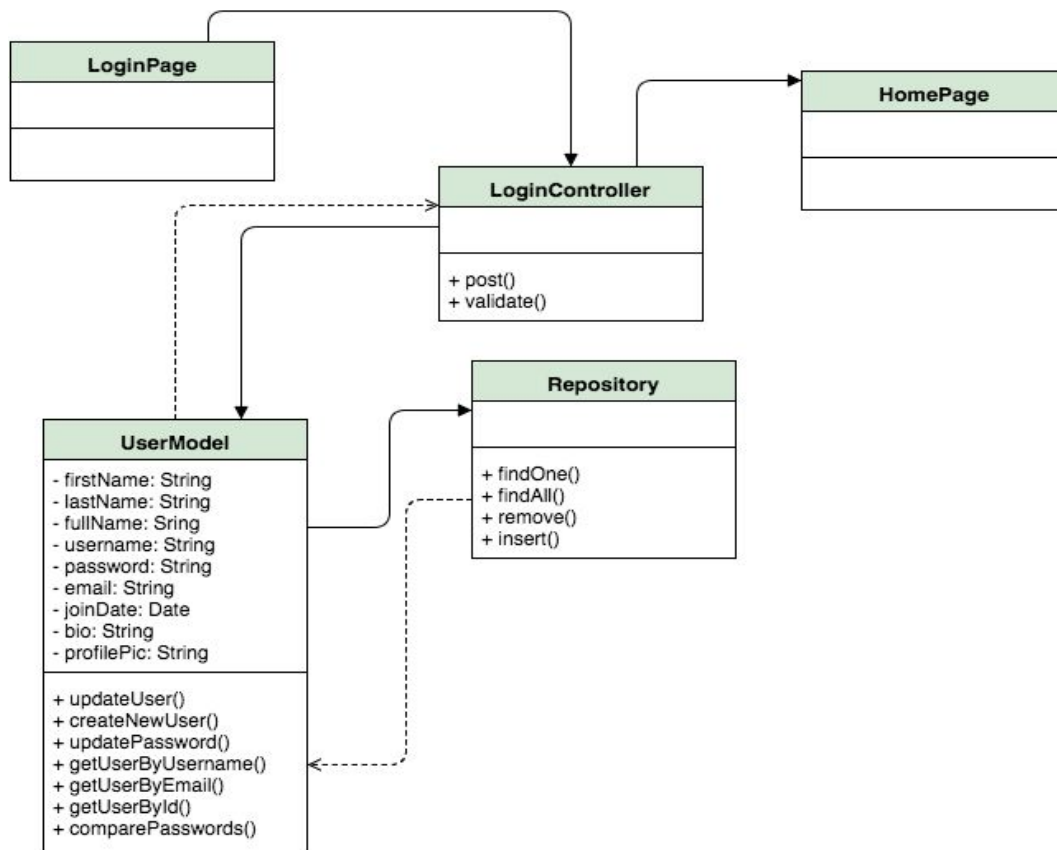


Figure 4 - Login class diagram for User Story #760

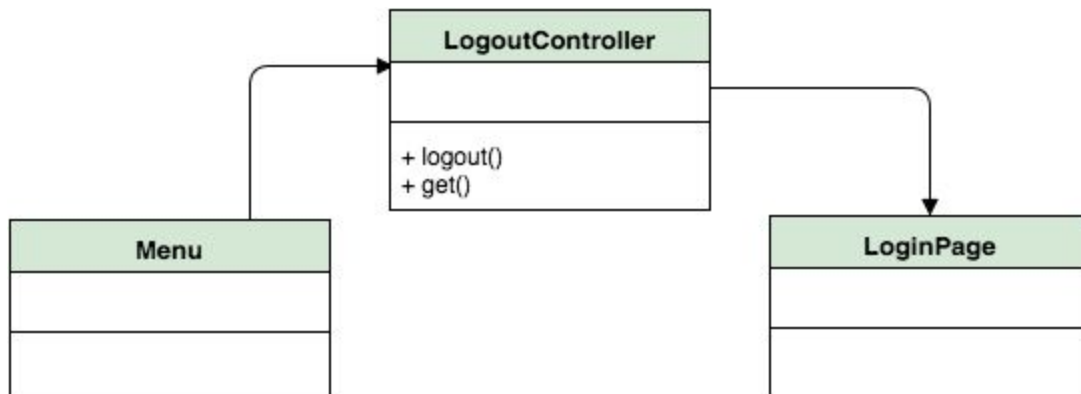


Figure 7 - Logout class diagram for User Story #762

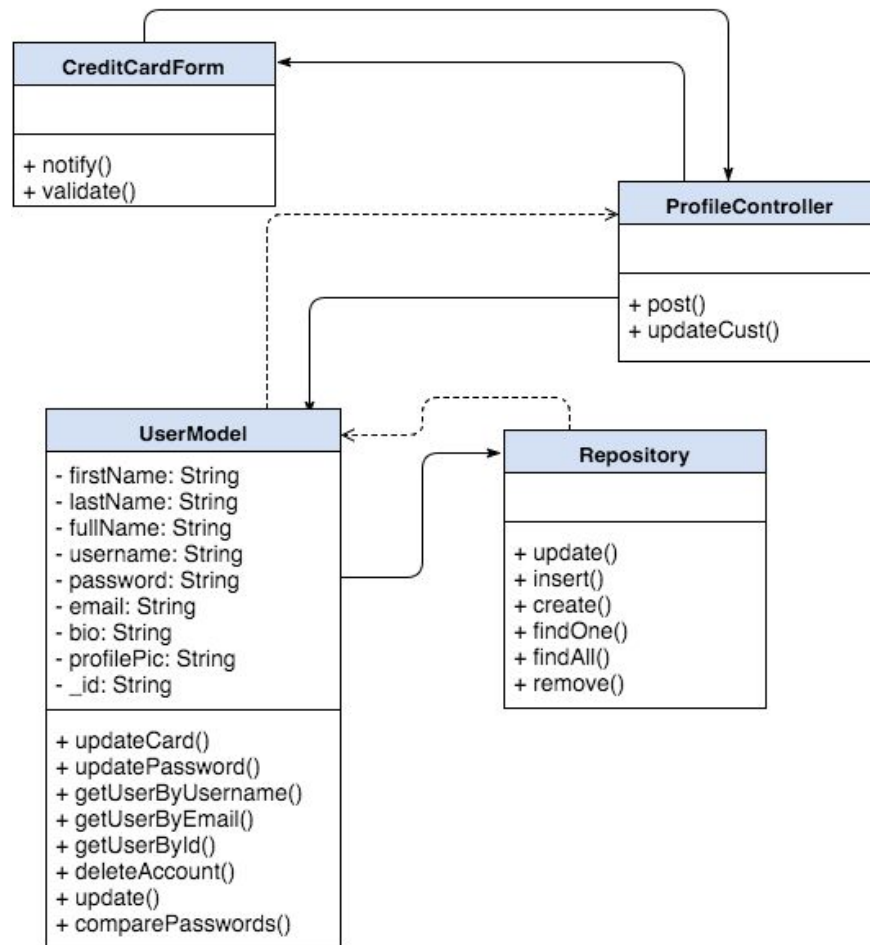


Figure 33 - Save Card details class diagram for User Story #850

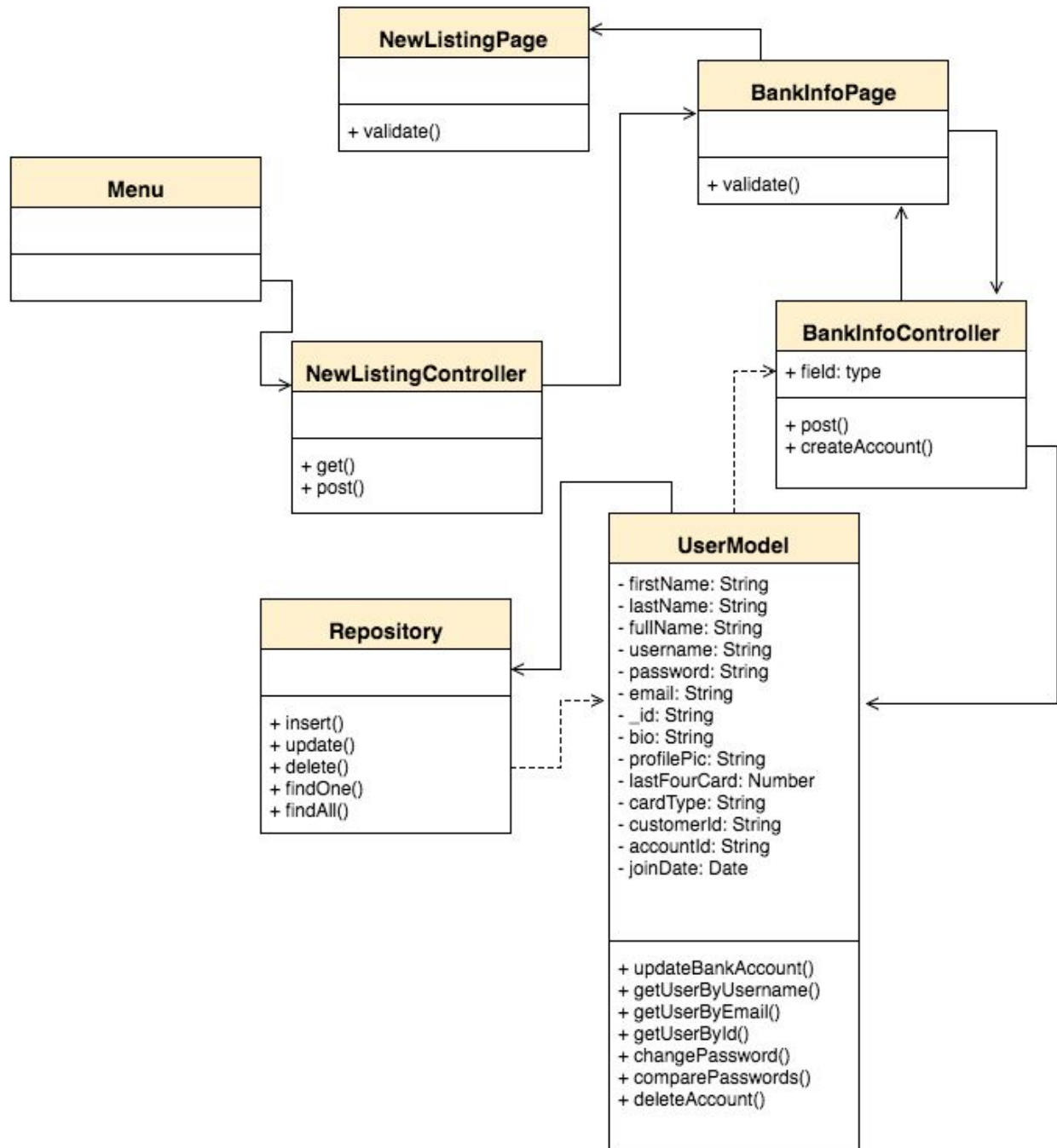


Figure 34 - Save Bank Account details diagram for User Story #851

Dynamic UML Diagrams**Figure 1 - Use Case Model for the entire system**

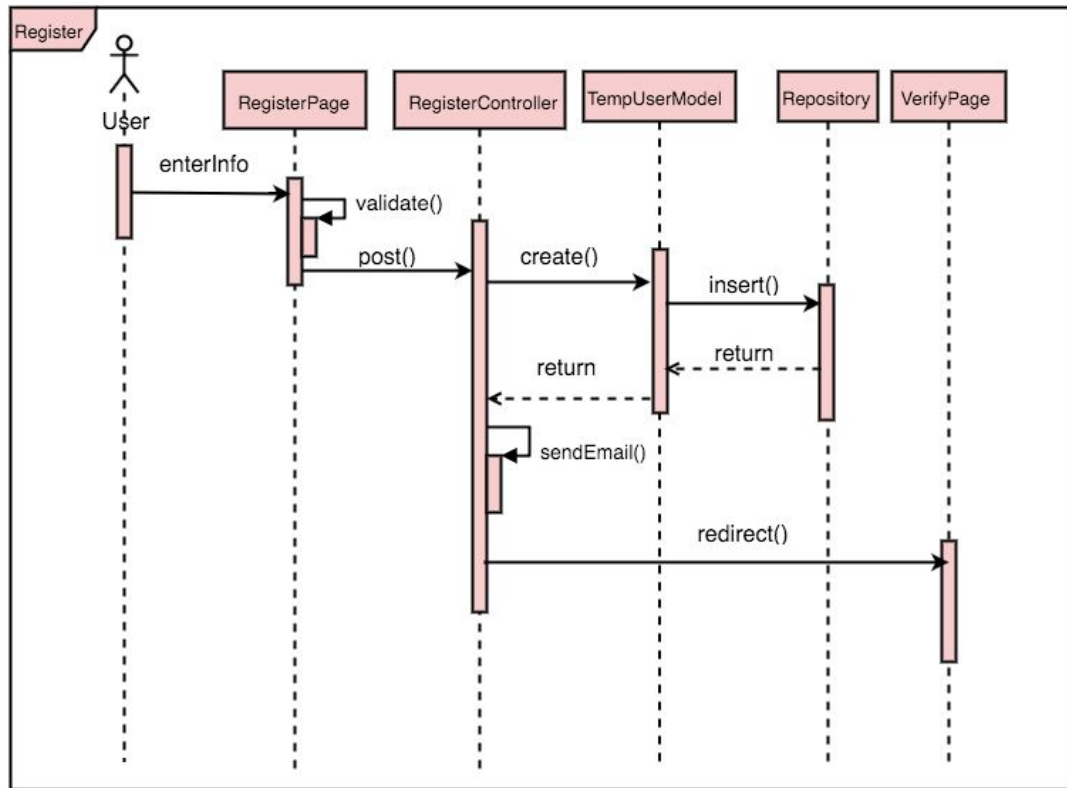


Figure 2 - Register sequence diagram for User Story #761

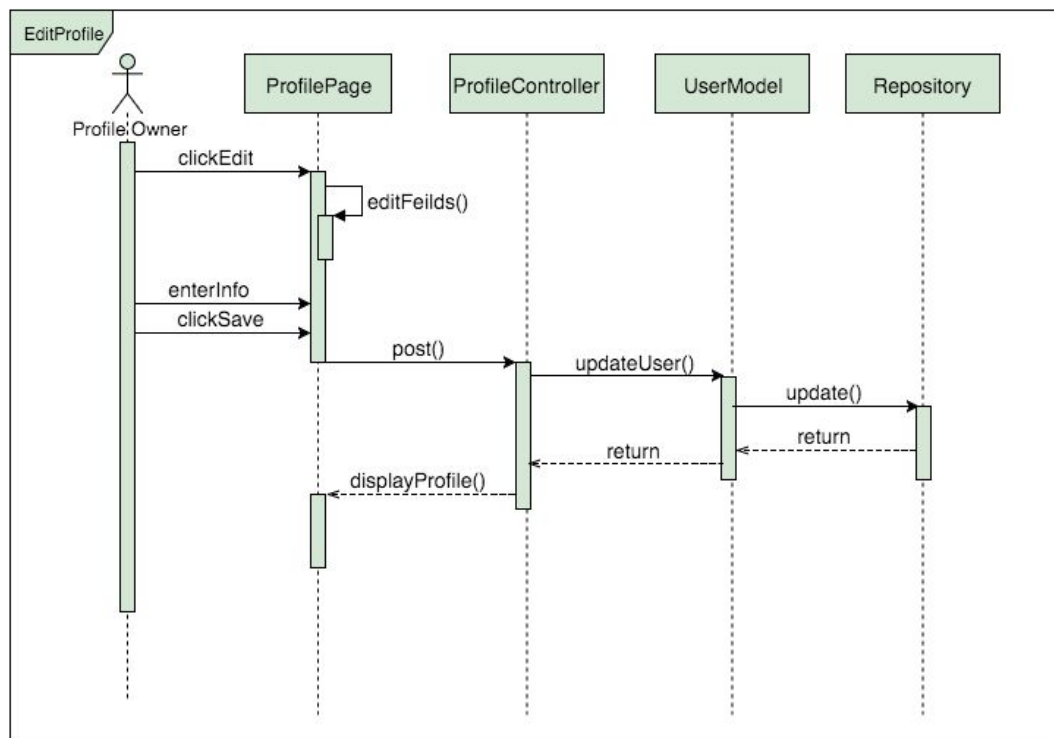


Figure 31 - Edit Profile sequence diagram for User Story #769

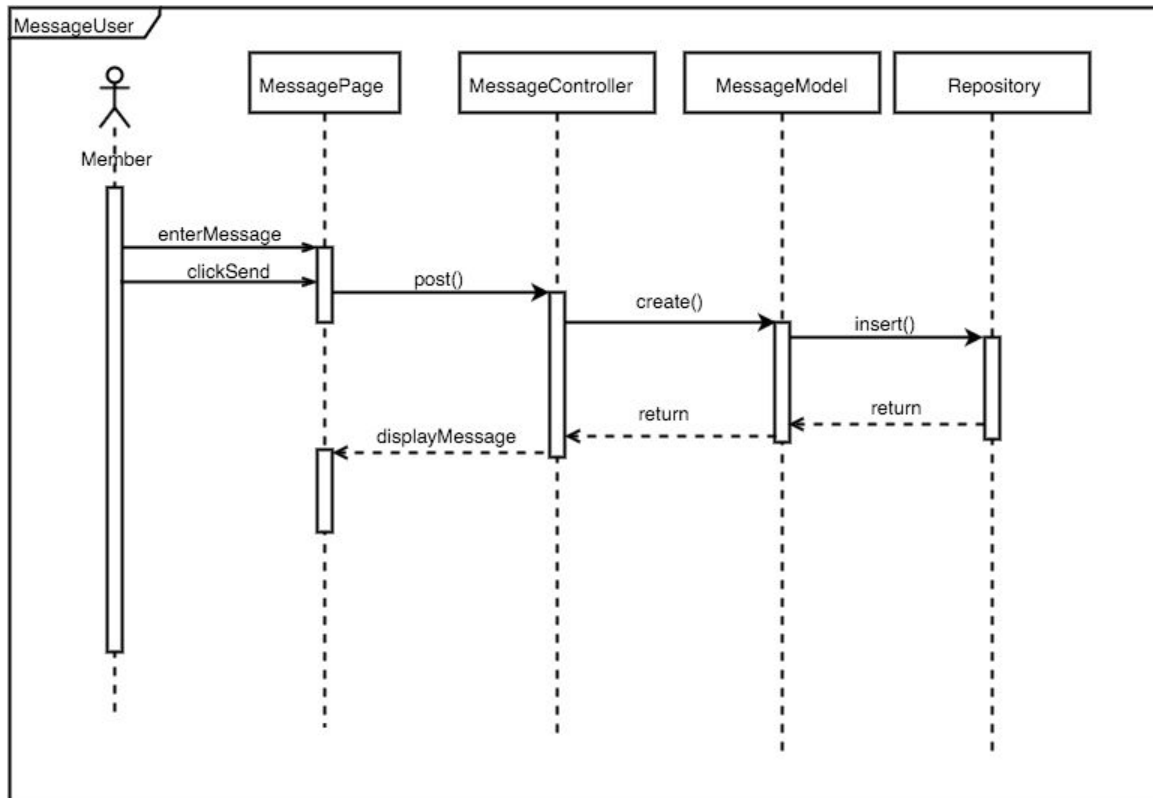


Figure 29 - Message user sequence diagram for User Story #841

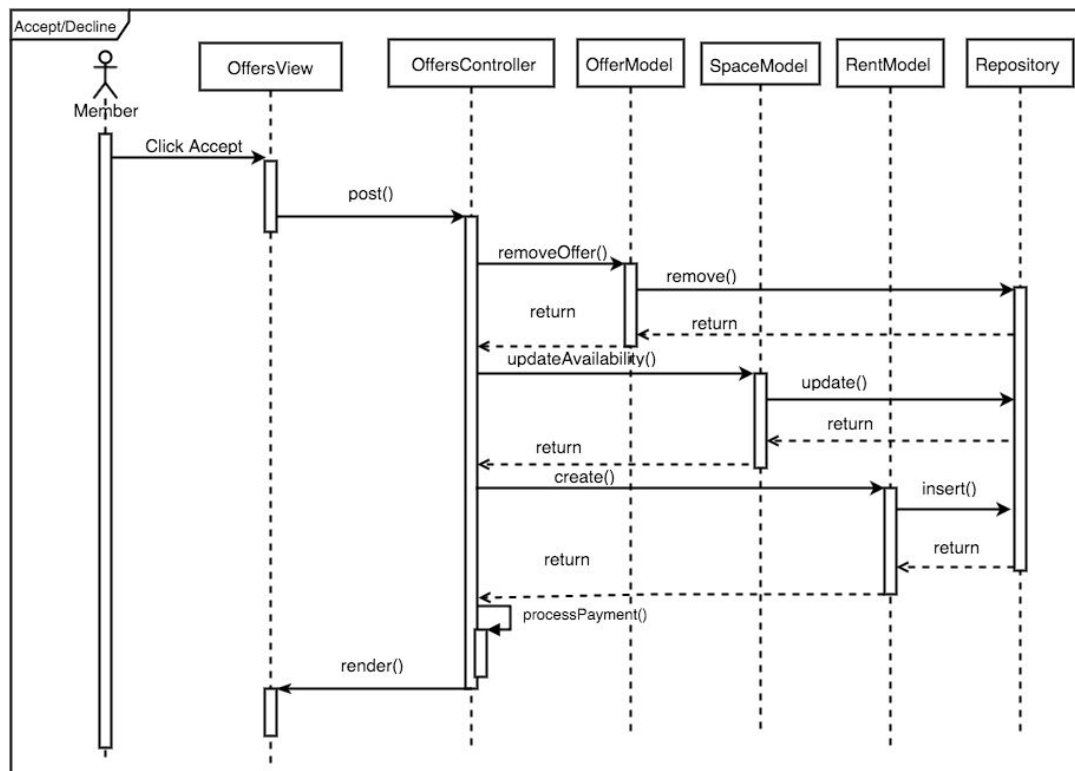


Figure 26 - Accept offer sequence diagram for User Story #818

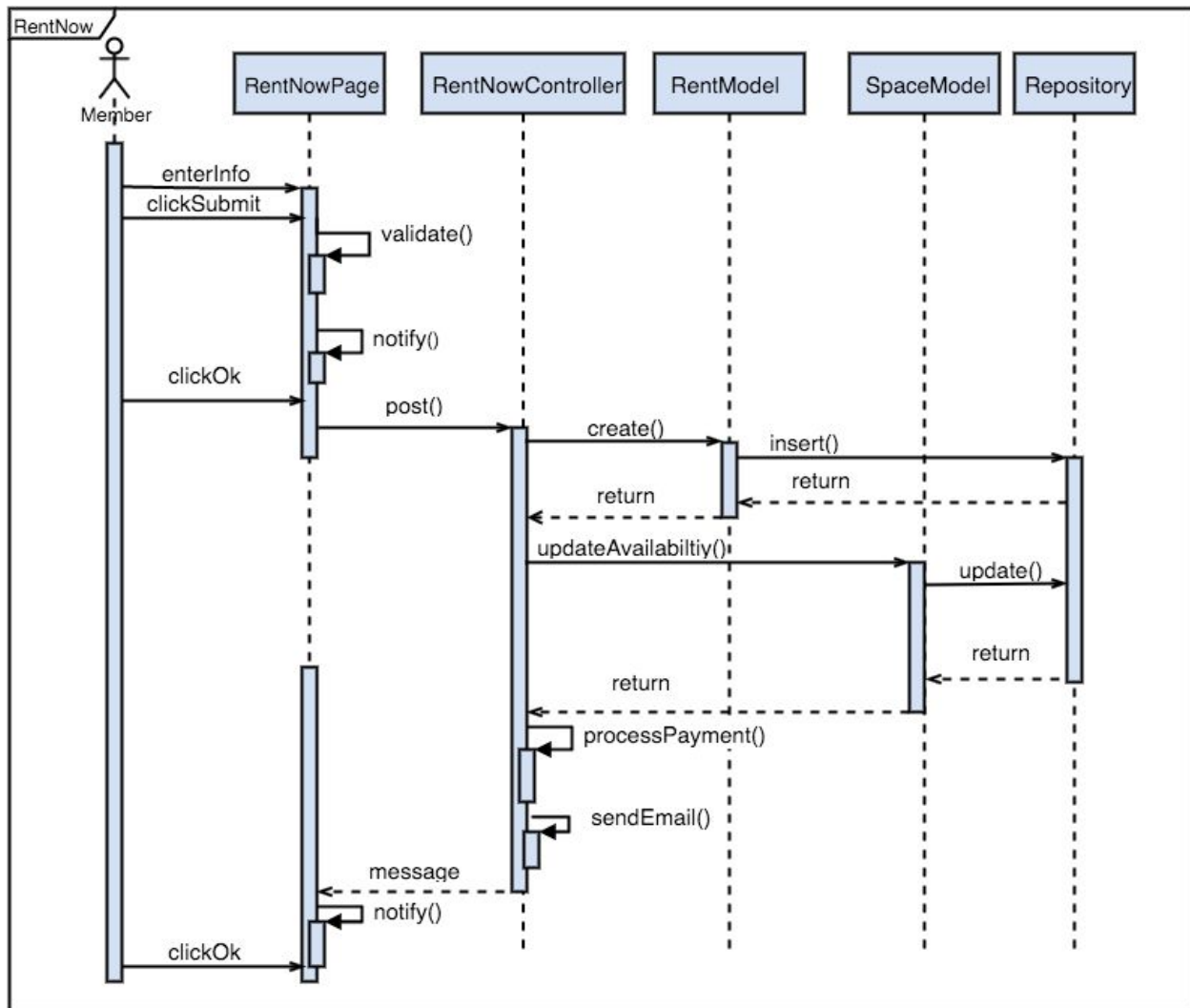


Figure 24 -Rent Now sequence diagram for User Story #830

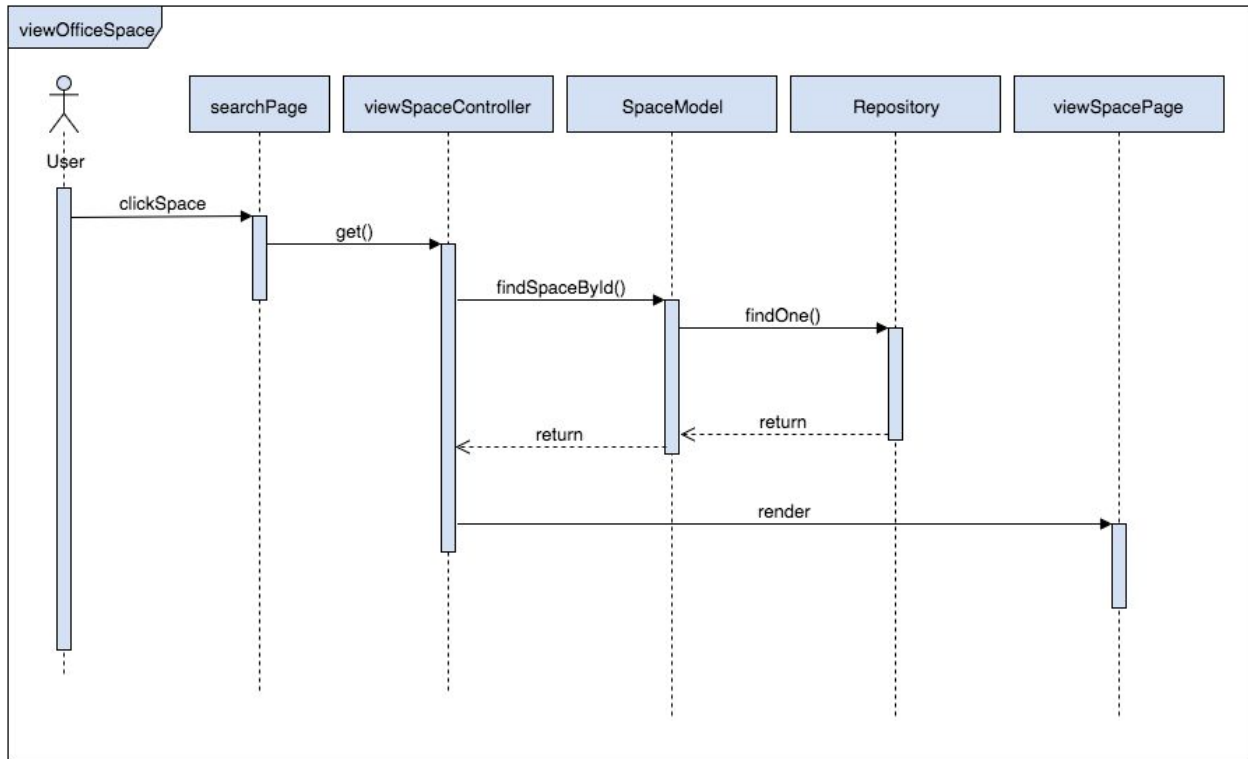


Figure 11 - View Office Space sequence diagram for User Story #764

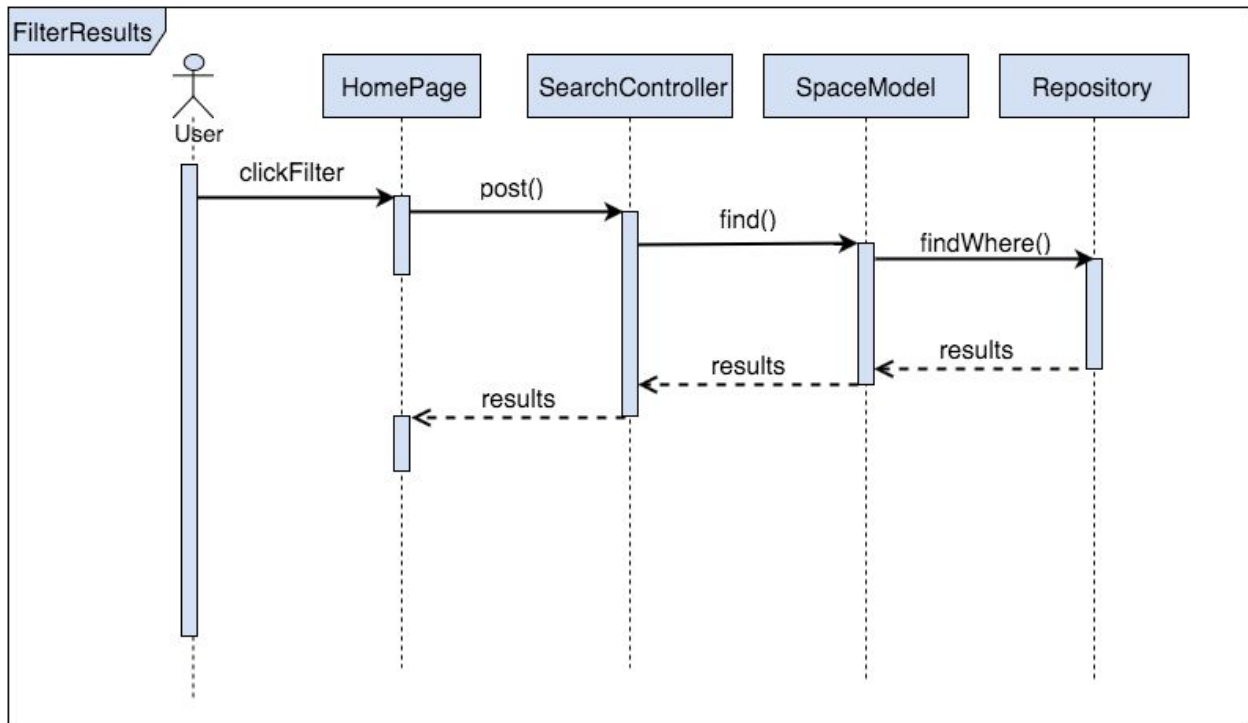


Figure 22 - Filter results sequence diagram for User Story #772

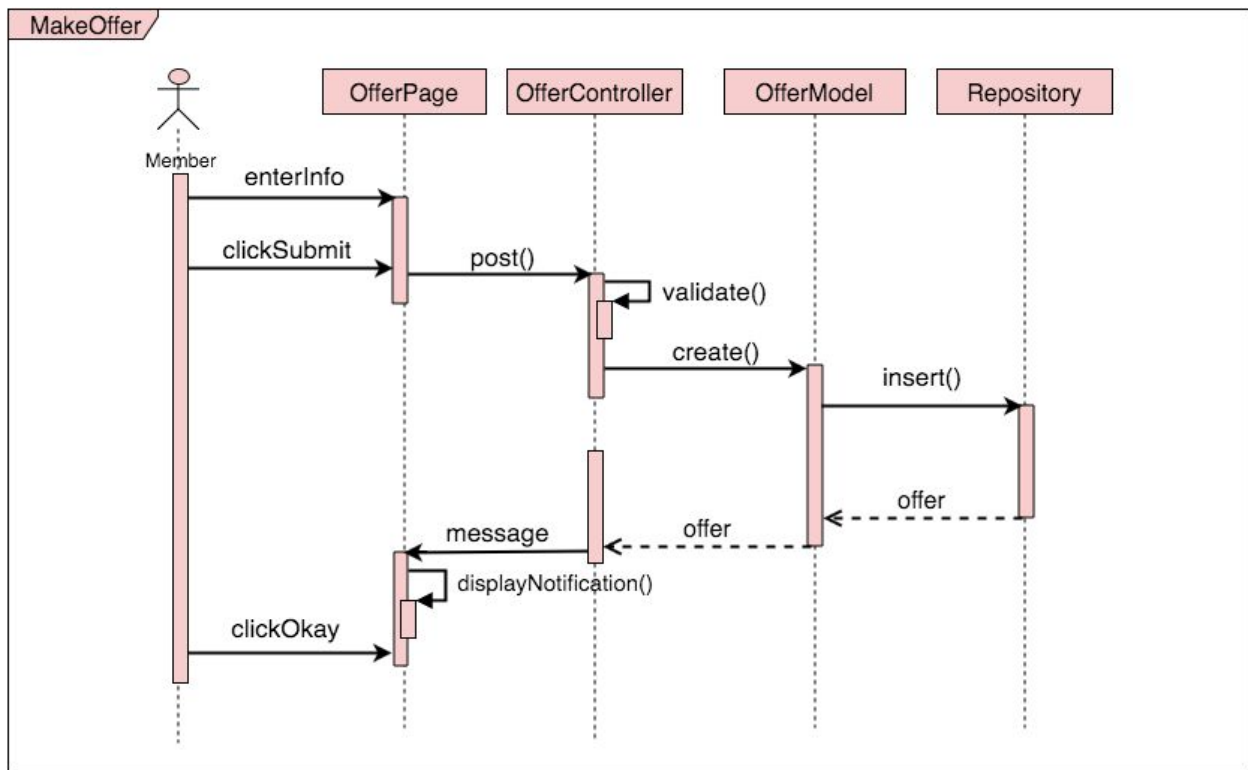


Figure 20 - Make an Offer sequence diagram for User Story #766

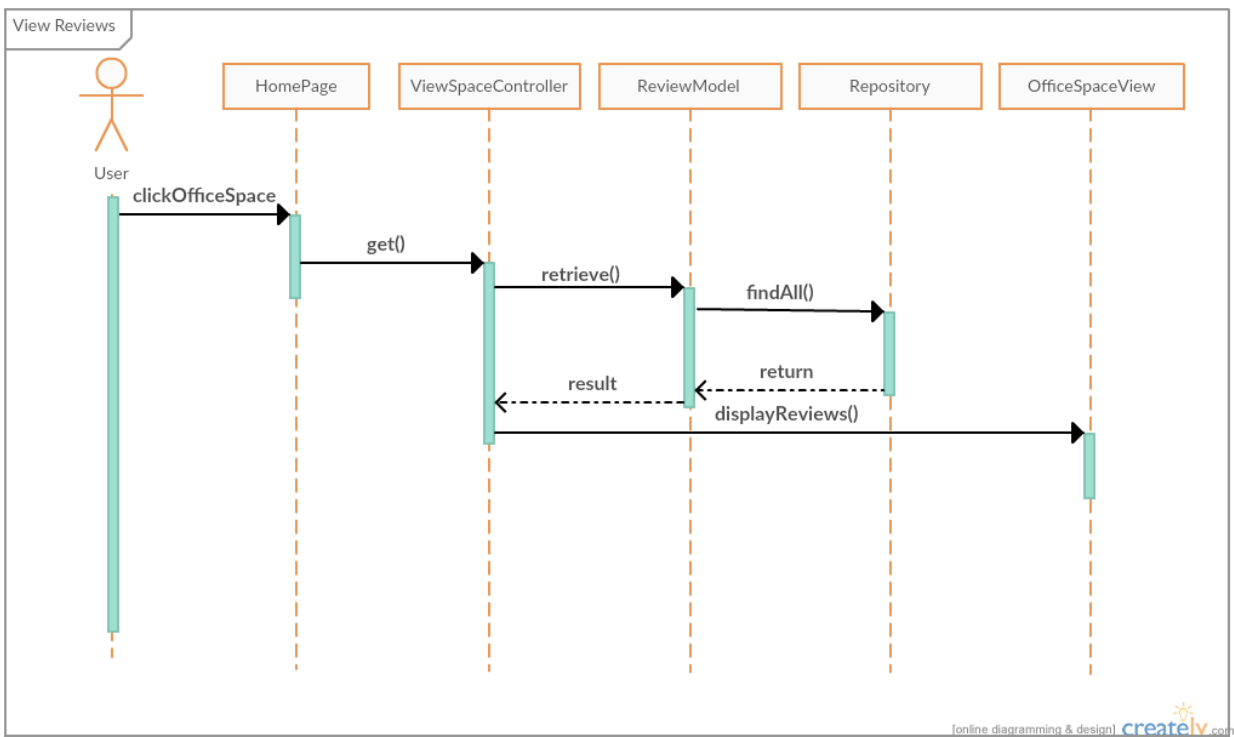


Figure 18 - View Reviews sequence diagram for User Story #768

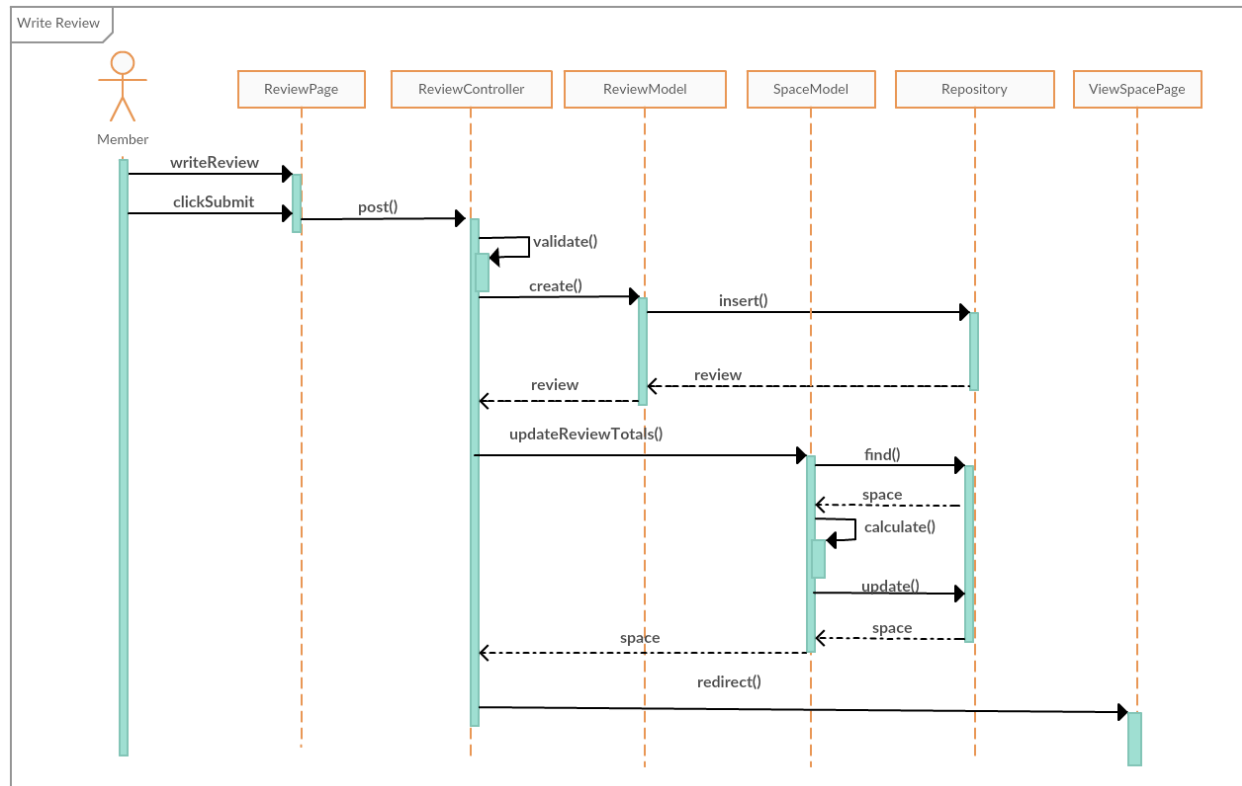


Figure 16 - Write Review sequence diagram for User Story #767

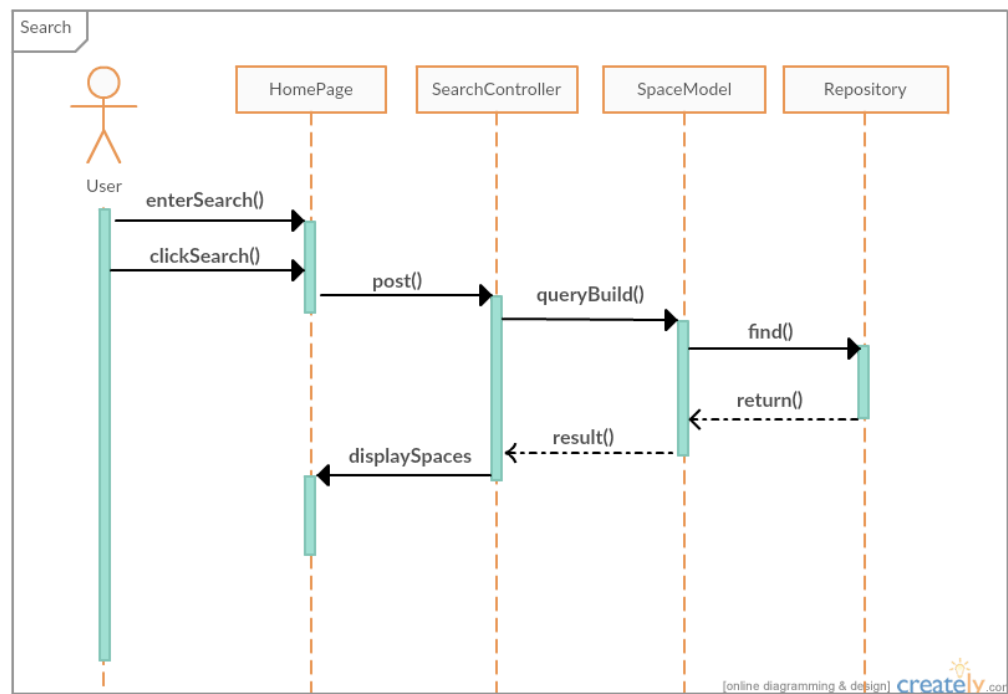


Figure 14 - Search sequence diagram for User Story #759

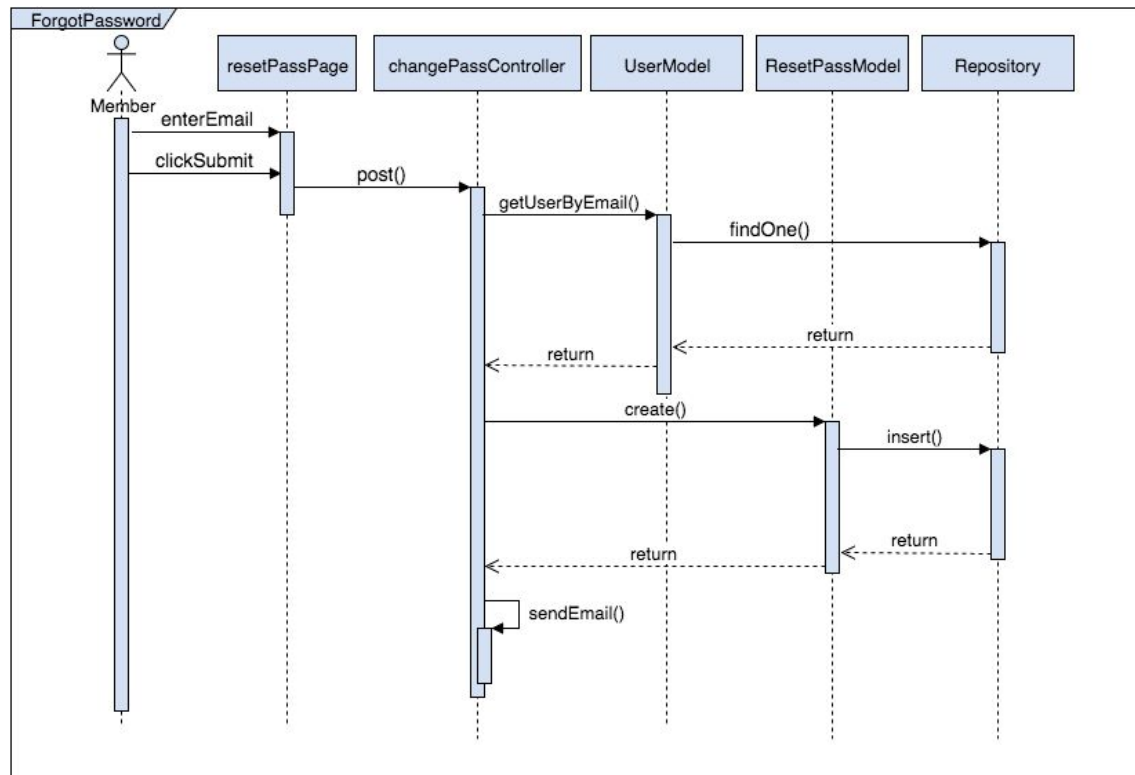


Figure 12 - Forgot password sequence diagram for User Story #765

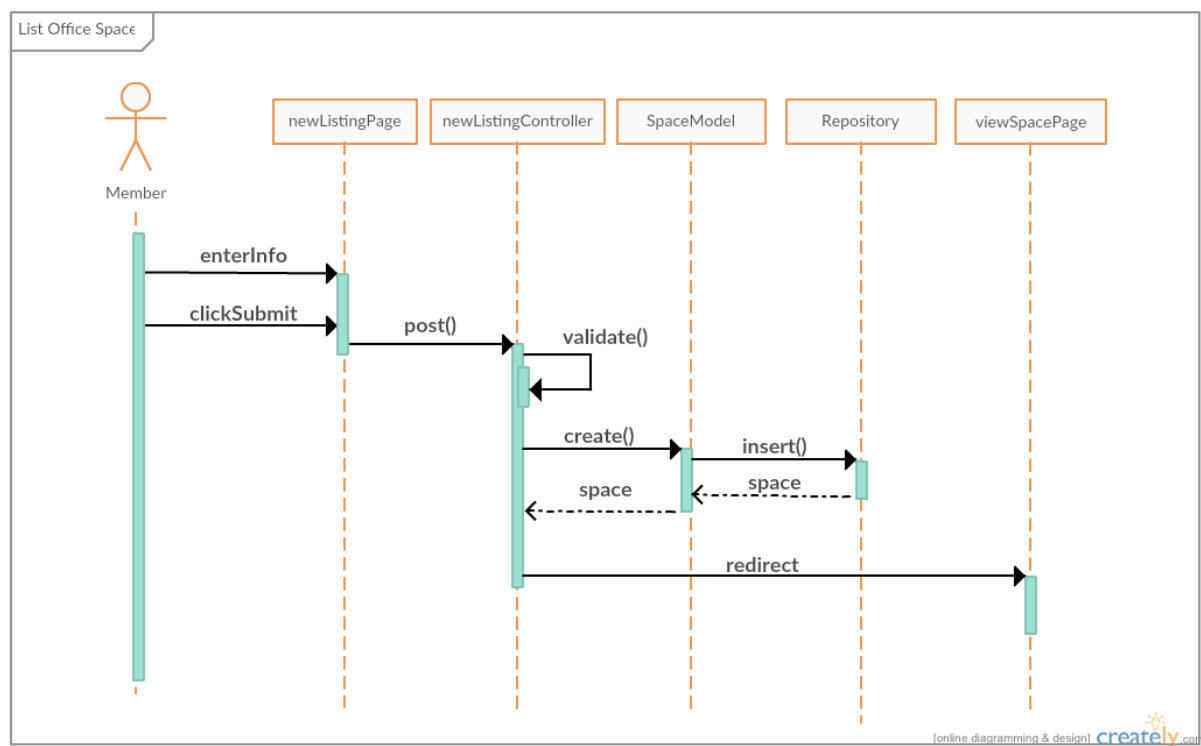


Figure 9 - New Listing sequence diagram for User Story #763

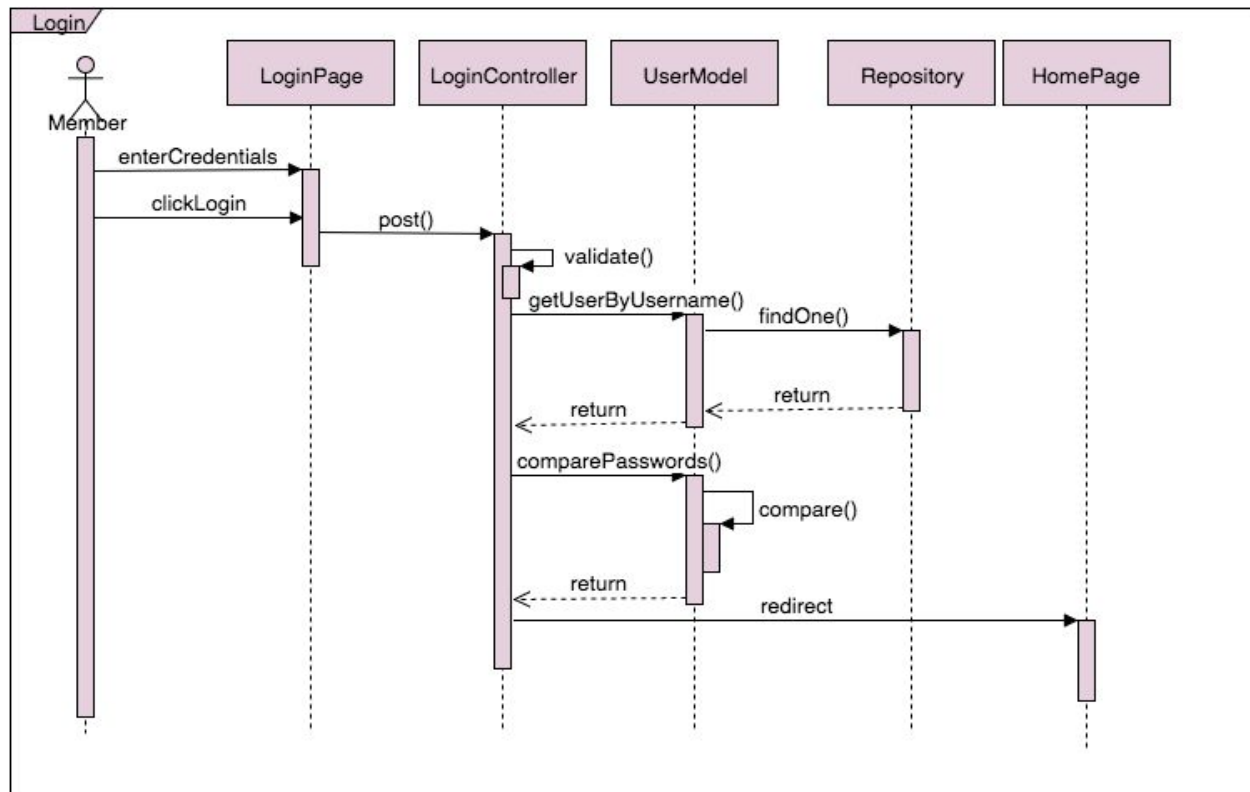


Figure 5 - Login sequence diagram for User Story #760

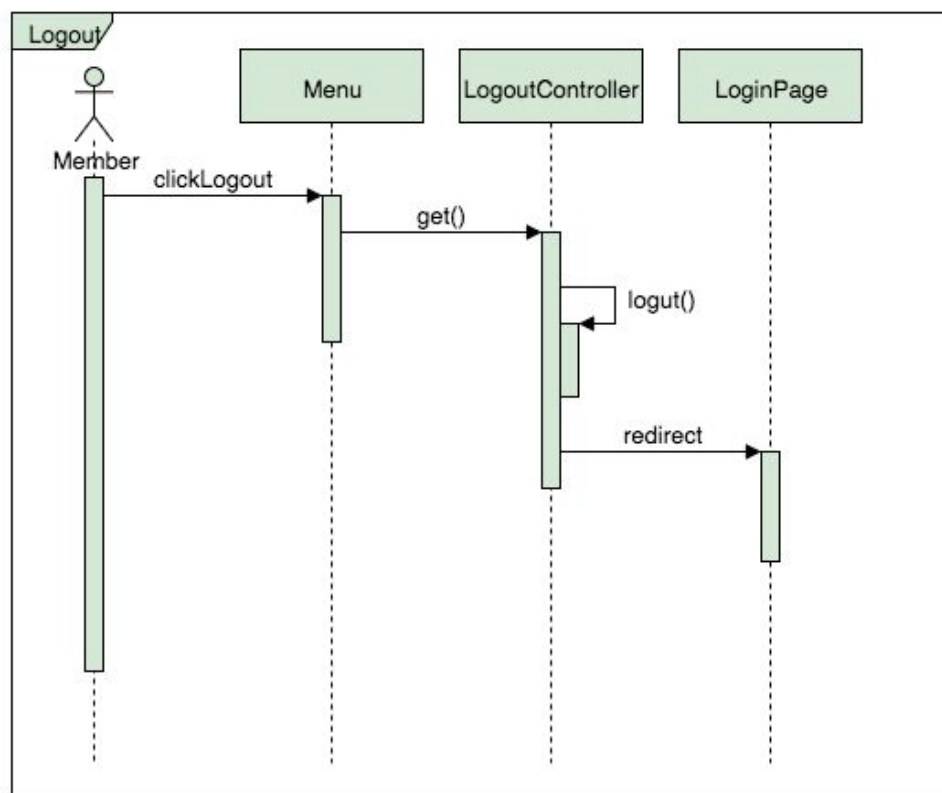


Figure 6 - Logout sequence diagram for User Story #762

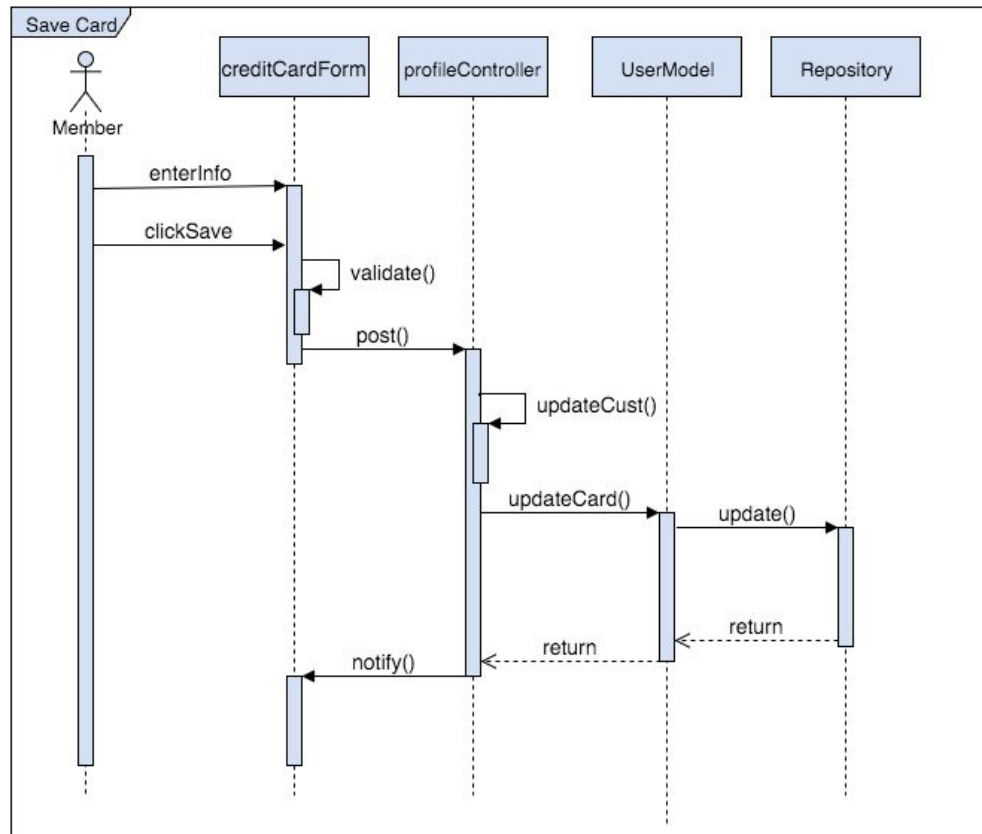


Figure 32 - Save Card sequence diagram for User Story #850

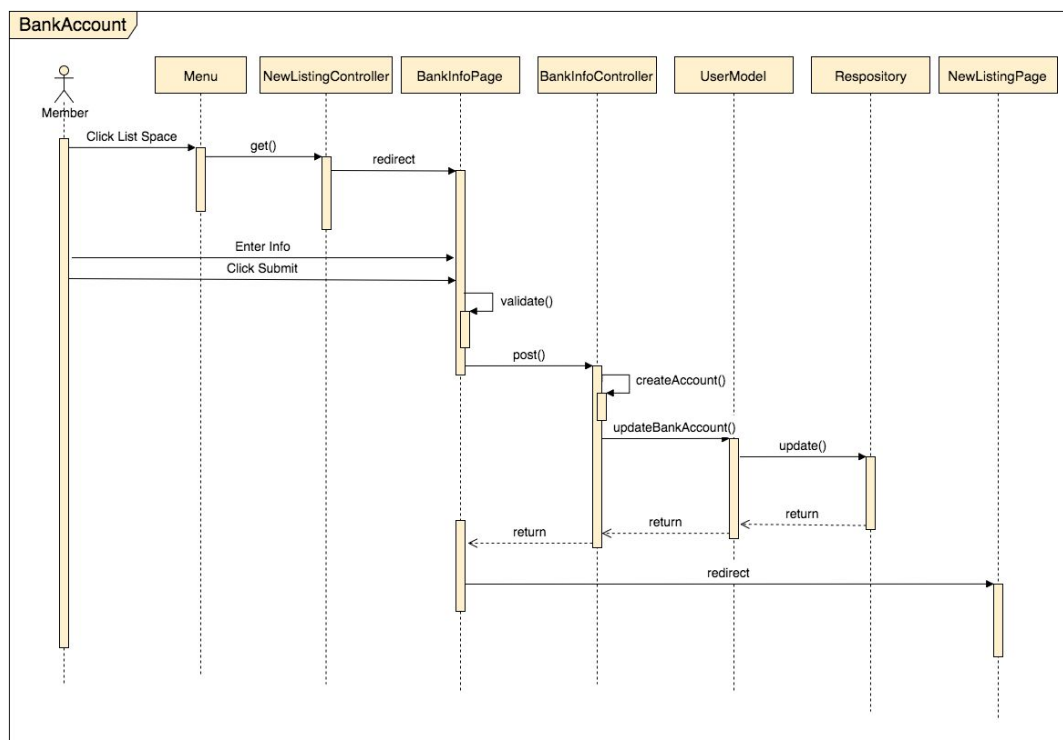
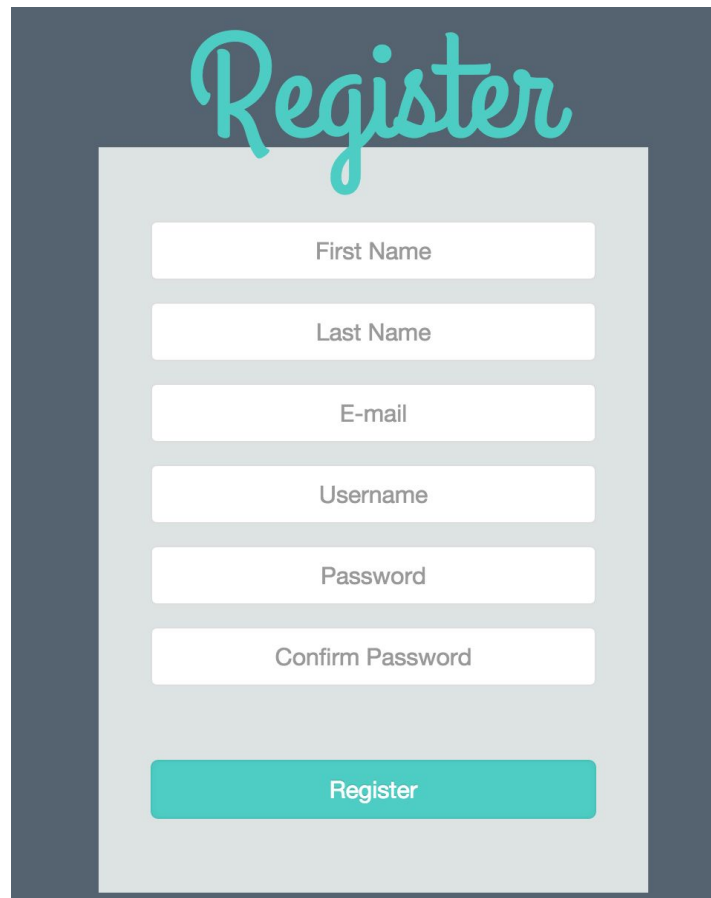


Figure 35 - Save Bank Account Details sequence diagram for User Story #851

Appendix B - User Interface Design



The image shows a 'Register' form with a dark blue background. The word 'Register' is written in a large, teal, cursive font at the top. Below it, there is a light gray rectangular area containing six white input fields stacked vertically. The labels for these fields are 'First Name', 'Last Name', 'E-mail', 'Username', 'Password', and 'Confirm Password'. At the bottom of this gray area is a teal button with the word 'Register' in white text.

Figure 38 - Register (User story #761)

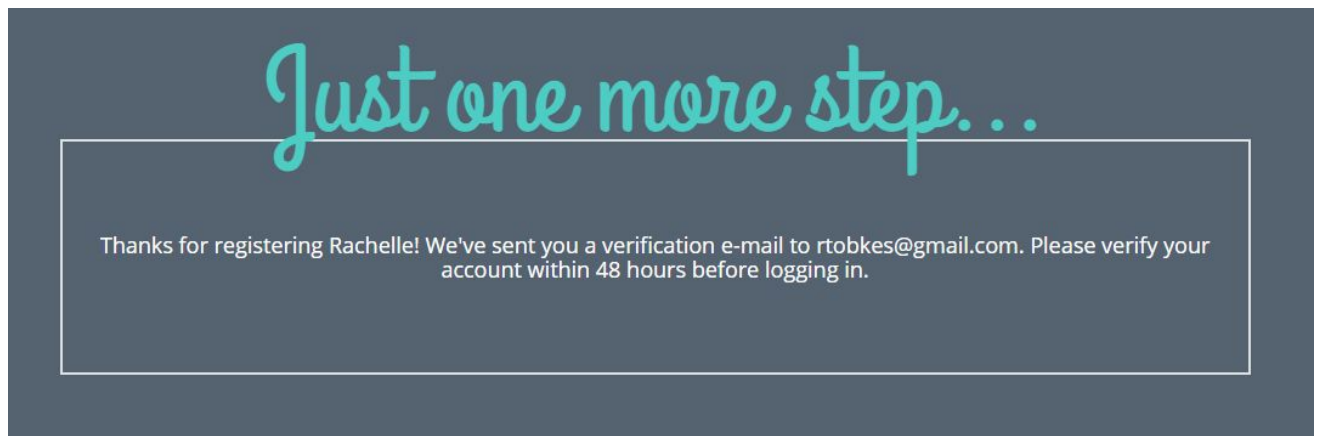
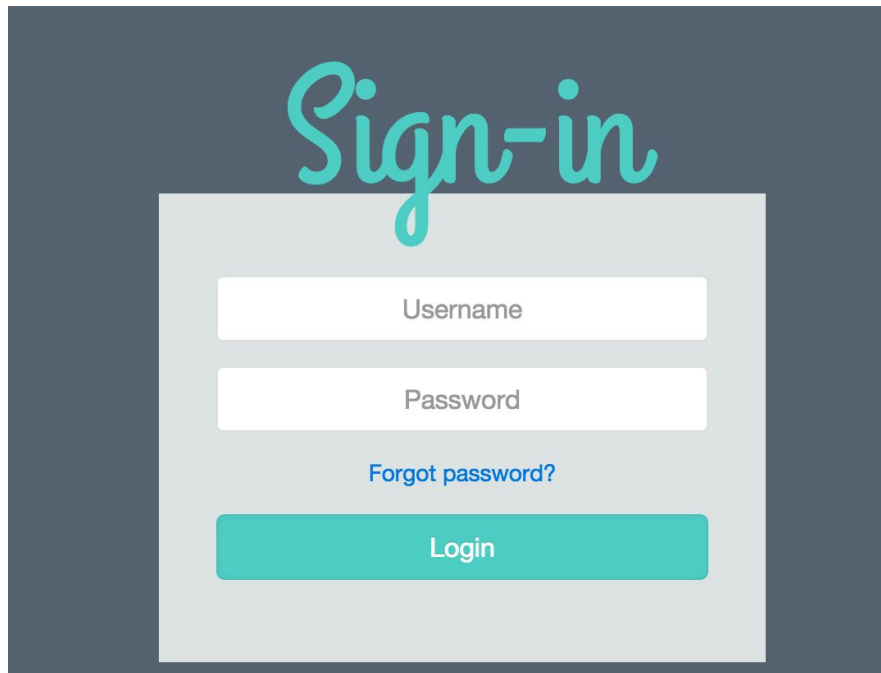


Figure 39 - Email verification notification when registering (User story #761)A sign-in form with a dark blue background. At the top, the word "Sign-in" is written in a large, teal, cursive font. Below it, there are two white input fields: "Username" and "Password". Under the "Password" field, there is a link that says "Forgot password?" in blue text. At the bottom, there is a teal button with the word "Login" in white text.

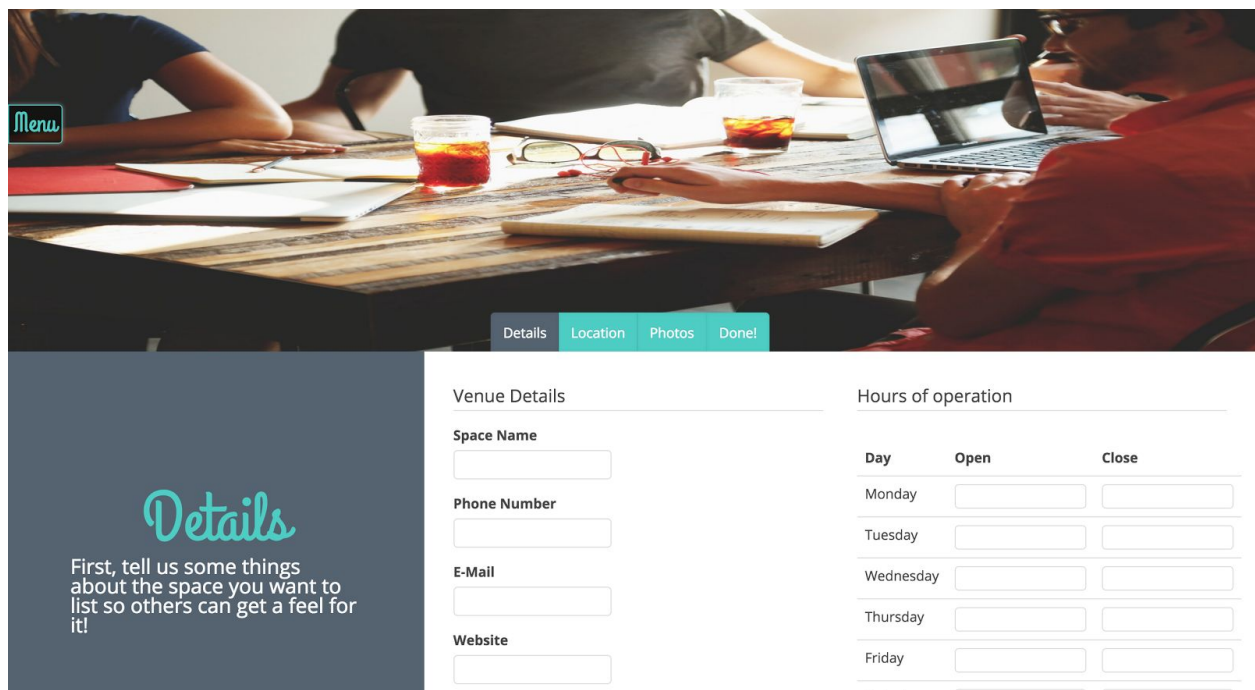
Sign-in

Username

Password

[Forgot password?](#)

Login

Figure 40 - Login (User story #760)A screenshot of a web application showing a form for listing an office space. The background is a photo of people working at a table. The form has a dark blue header with the word "Menu" in a teal box. Below the header, there are four tabs: "Details", "Location", "Photos", and "Done!". The "Details" tab is selected. The form is divided into two columns. The left column is titled "Venue Details" and contains four input fields: "Space Name", "Phone Number", "E-Mail", and "Website". The right column is titled "Hours of operation" and contains a table with three columns: "Day", "Open", and "Close". The table has five rows for the days of the week: Monday, Tuesday, Wednesday, Thursday, and Friday. Each row has input fields for the "Open" and "Close" times.

Menu

Details Location Photos Done!

Venue Details

Space Name

Phone Number

E-Mail

Website

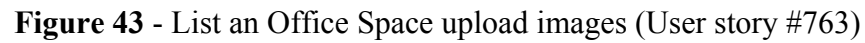
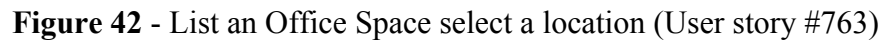
Hours of operation

Day	Open	Close
Monday		
Tuesday		
Wednesday		
Thursday		
Friday		

Details

First, tell us some things about the space you want to list so others can get a feel for it!

Figure 41 - List an Office Space details (User story #763)



5 / 5
1 Reviews
[Leave a Review](#)

Phone Number
9546439966

Email
rtobk001@fiu.edu

Website
rtobkes.com

Type
Private Office

Floors
45

Square Feet
45

Number of Rooms
45

Number of Desks
40

Description
love this space :)

Day	Open	Close
Monday	05:00 AM	11:00 PM
Tuesday	04:00 AM	11:00 PM
Wednesday	04:00 AM	11:00 PM
Thursday	04:00 AM	11:00 PM
Friday	04:00 AM	11:00 PM
Saturday	04:00 AM	11:00 PM
Sunday	04:00 AM	11:00 PM

Features

- Wi-Fi
- Furniture
- Accessibility
- Mail
- Coffee

Pricing

Starting Bid: \$45 [Make An Offer](#)

Rent Now: \$45 [Rent Now](#)

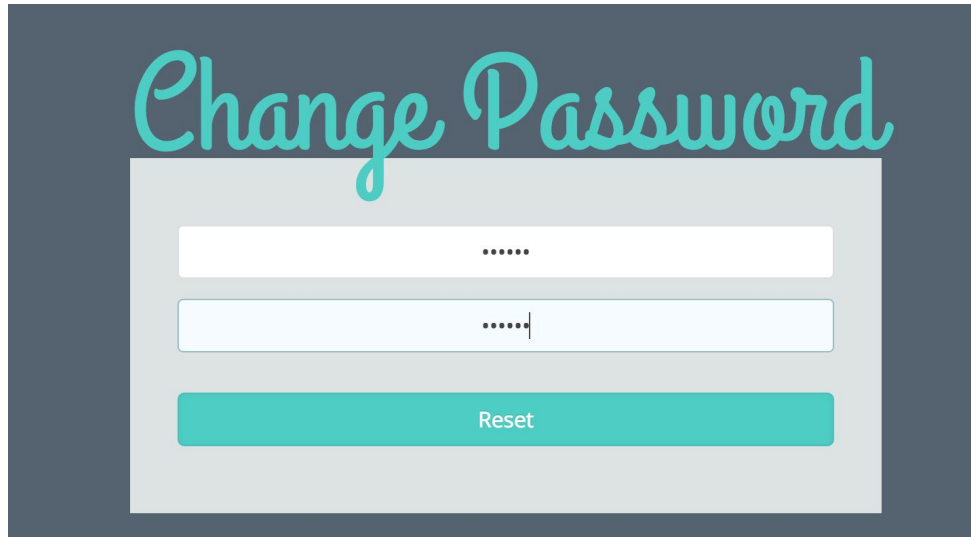
Figure 44 - Viewing an office space (User story #764)

Reset Password

E-mail

Reset

Figure 45 - Reset password form to enter email address (User story #765)



A screenshot of a 'Change Password' form. The title 'Change Password' is written in a large, teal, cursive font at the top. Below the title, there are two input fields for passwords, each containing six dots. A teal button labeled 'Reset' is positioned below the input fields. The entire form is centered on a dark gray background.

Figure 46 - Form to set up a new password (User story #765)

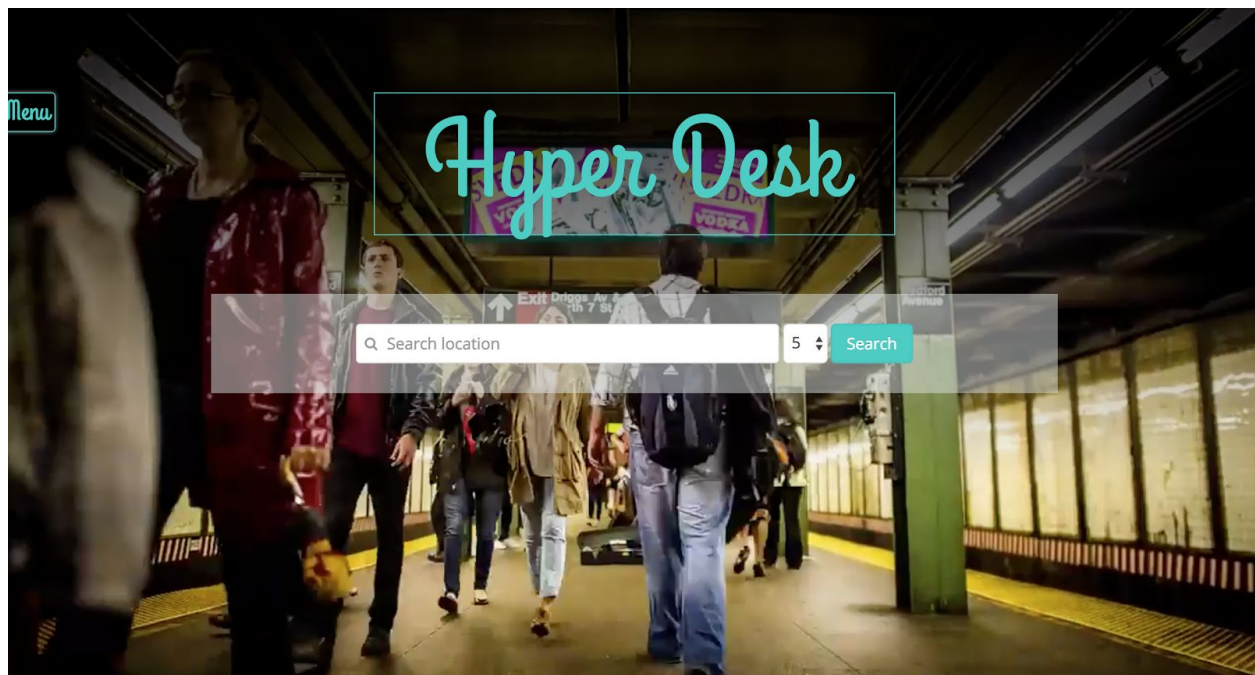


Figure 47 -Homepage and search (User story #759)

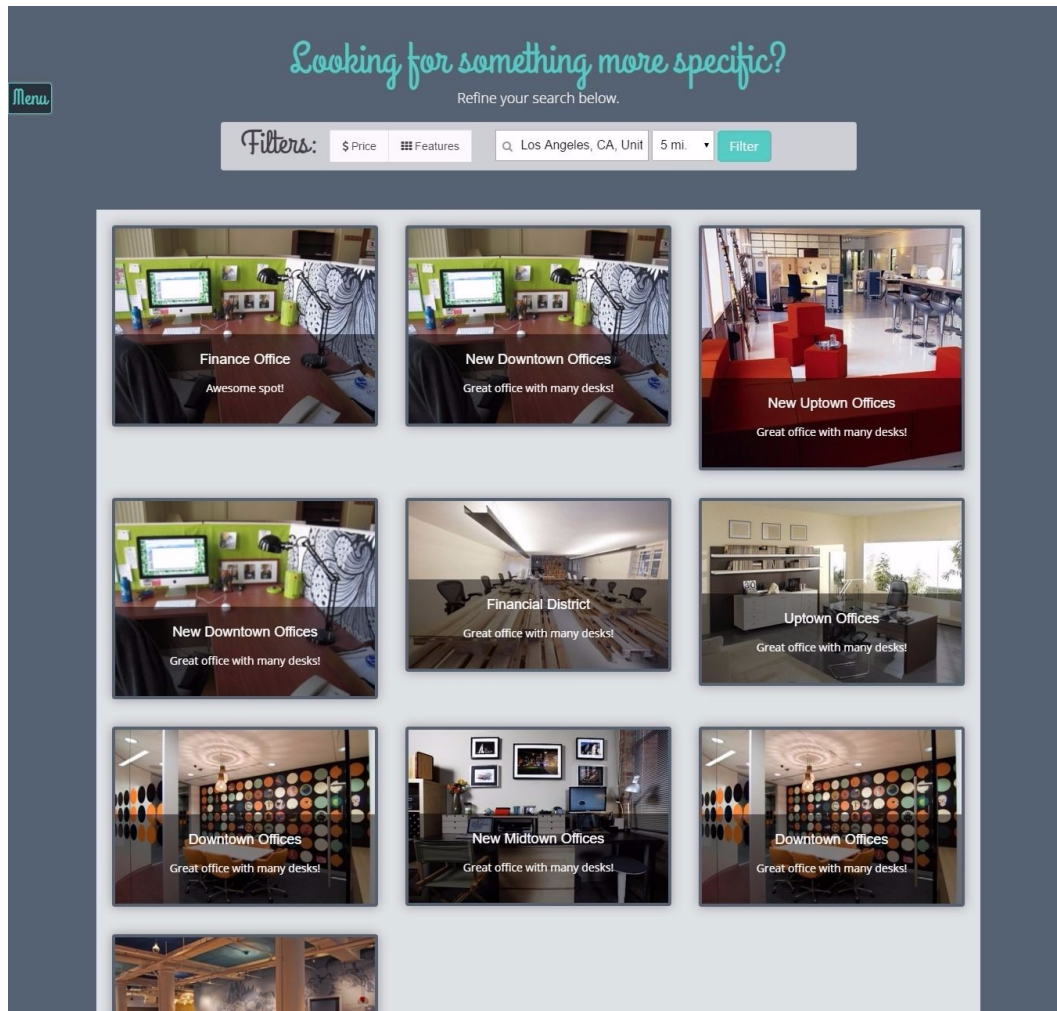


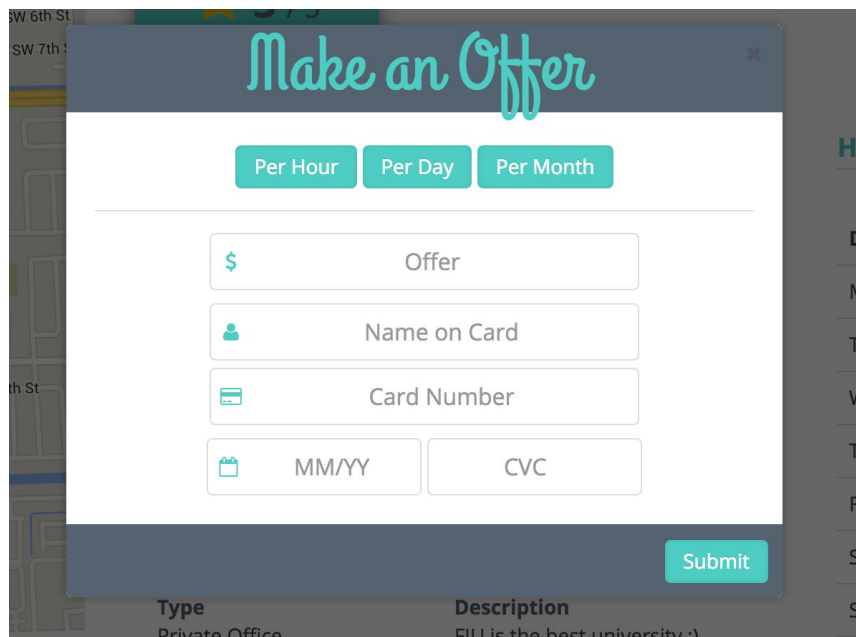
Figure 48 - Search results display (User story #759)

Figure 49 - Filter search results (User story #772)



The image shows a 'Write a Review' form. At the top, the title 'Write a Review' is in a large, teal, cursive font. Below it, the text 'Select your rating:' is in a smaller, black, cursive font. Underneath is a row of five stars; the first three are yellow and the last two are grey. Below the stars is a text input field containing the text 'IT WAS ALRIGHT :/'. At the bottom of the form, there are two buttons: a blue 'Go Back' button and a teal 'Post' button.

Figure 50 - Review an office space (User story #767)



The image shows a 'Make an Offer' form. At the top, the title 'Make an Offer' is in a large, teal, cursive font. Below it, there are three buttons: 'Per Hour', 'Per Day', and 'Per Month'. Below these buttons is a form with four input fields: a dollar sign icon followed by 'Offer', a person icon followed by 'Name on Card', a card icon followed by 'Card Number', and a calendar icon followed by 'MM/YY'. To the right of the 'MM/YY' field is a 'CVC' field. At the bottom right of the form is a teal 'Submit' button. The background of the form is a dark grey map.

Figure 51 - Making an offer (User story #766)

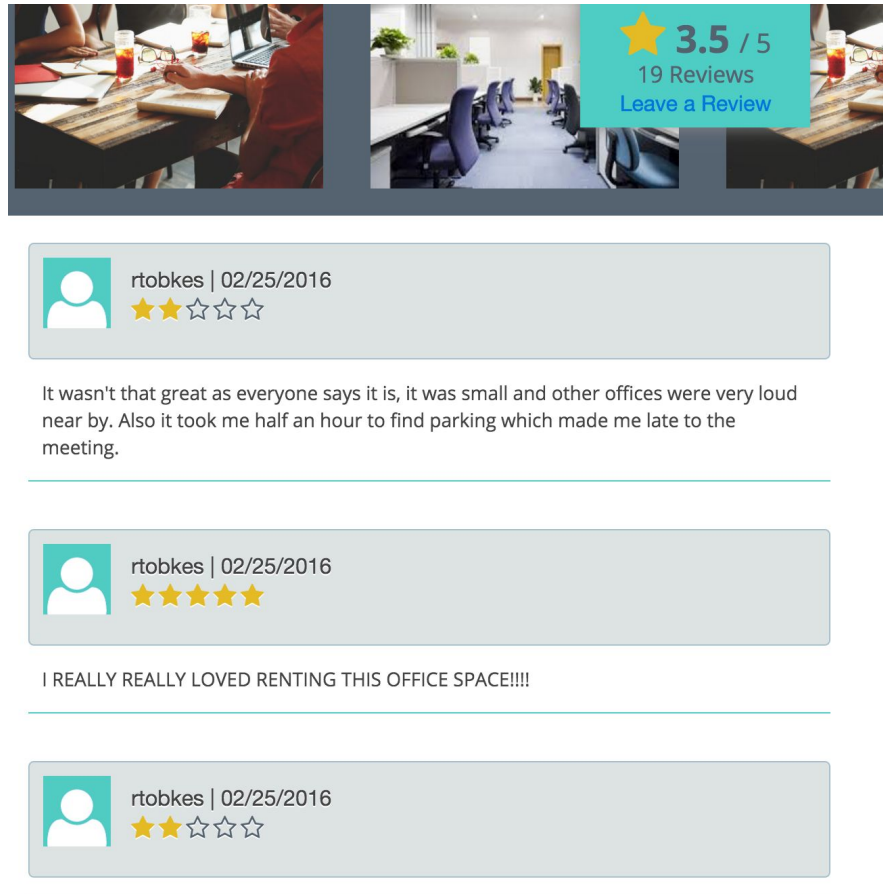


Figure 52 - View reviews (User story #768)

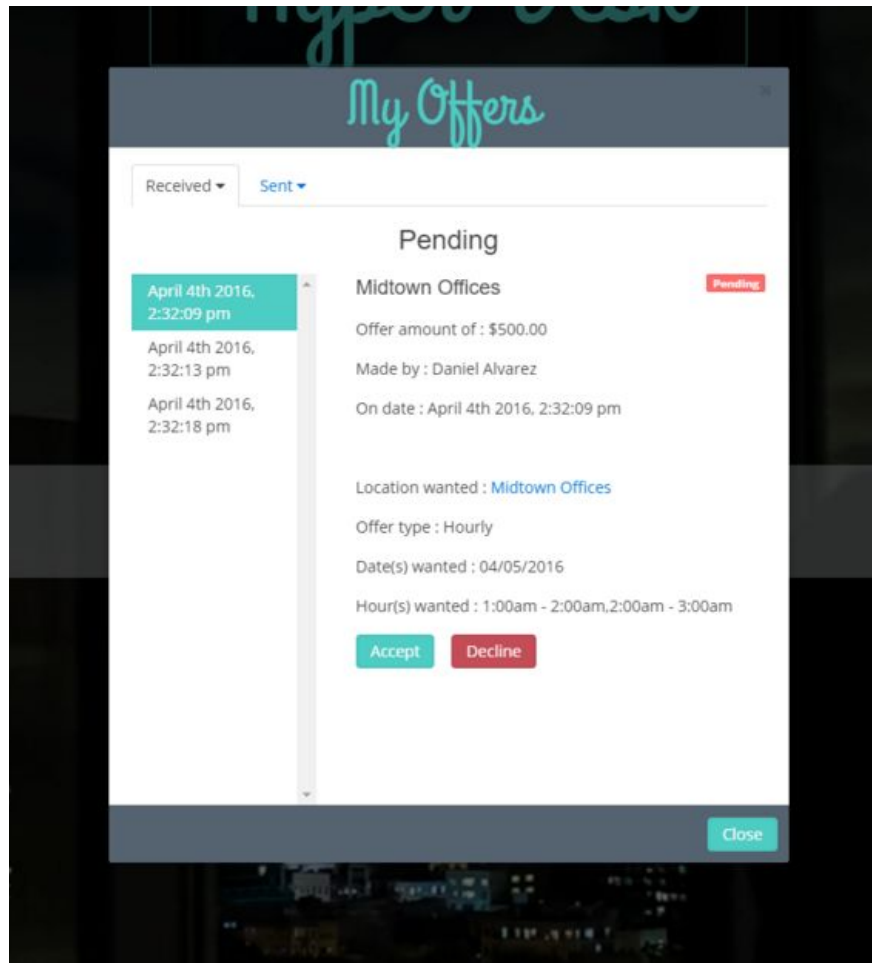


Figure 52 - Accepting/declining offers and viewing offers (User story #818)

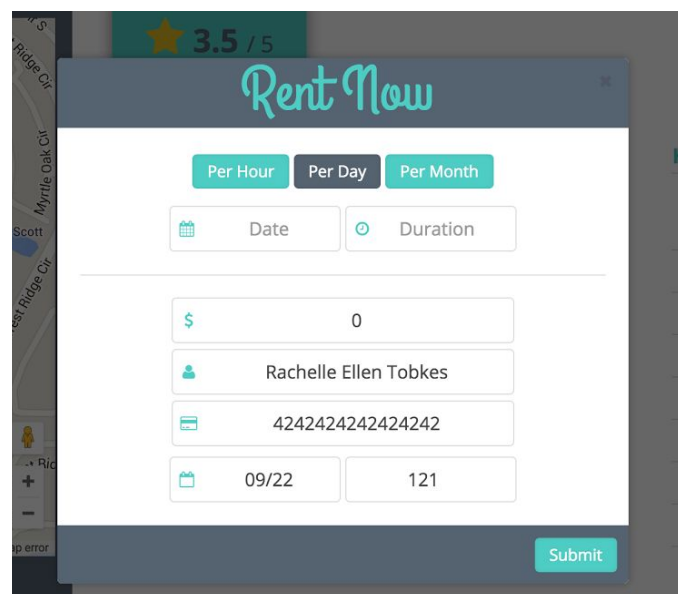


Figure 54 - Rent office space for given price form (User story #830)

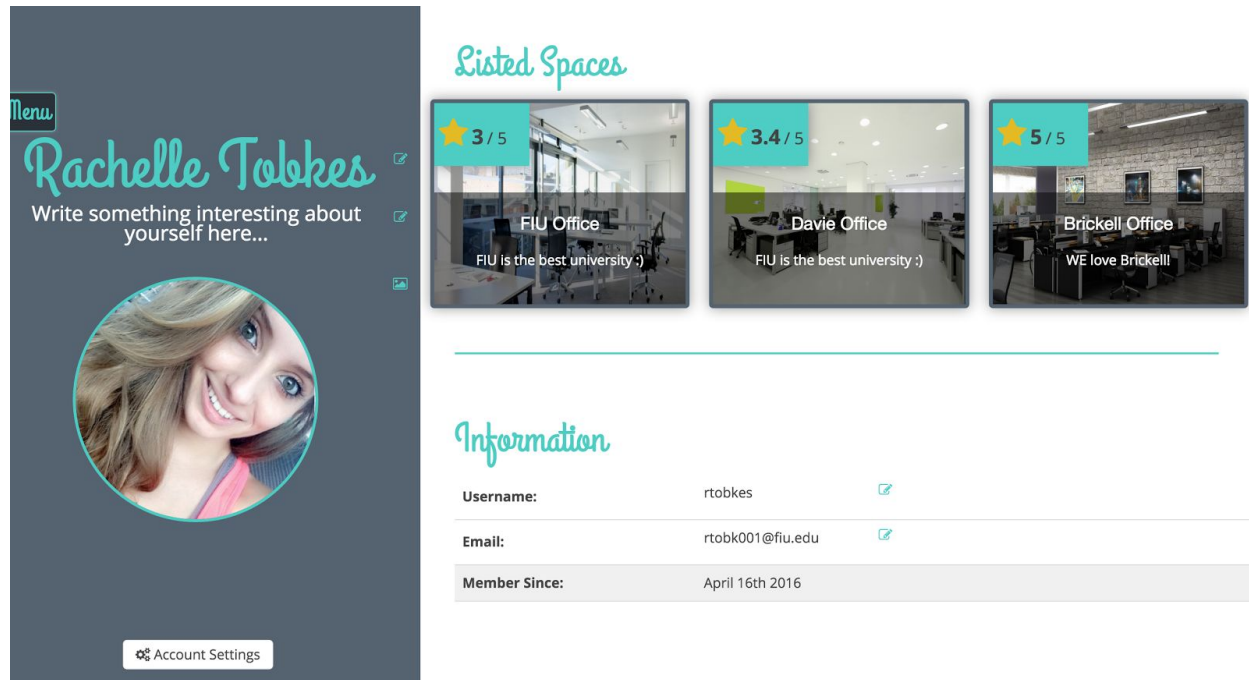


Figure 55 -Viewing a user's profile and editing your profile (User story #769)

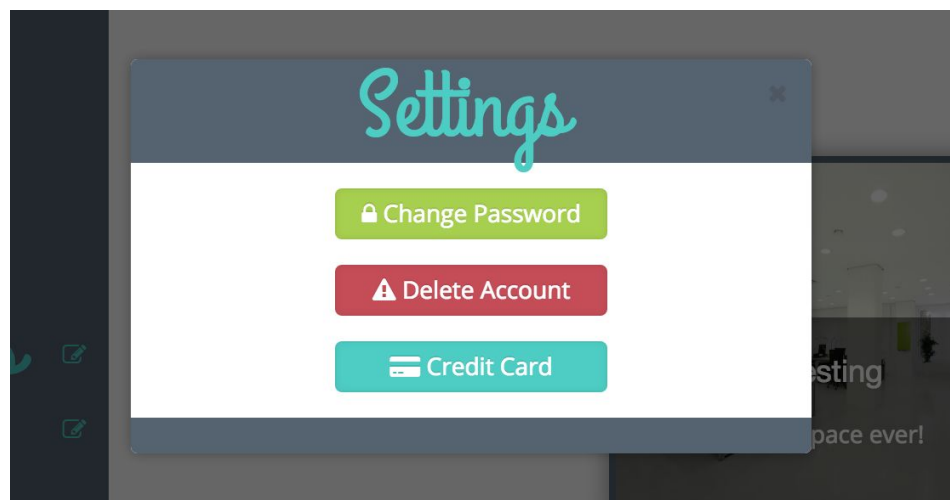
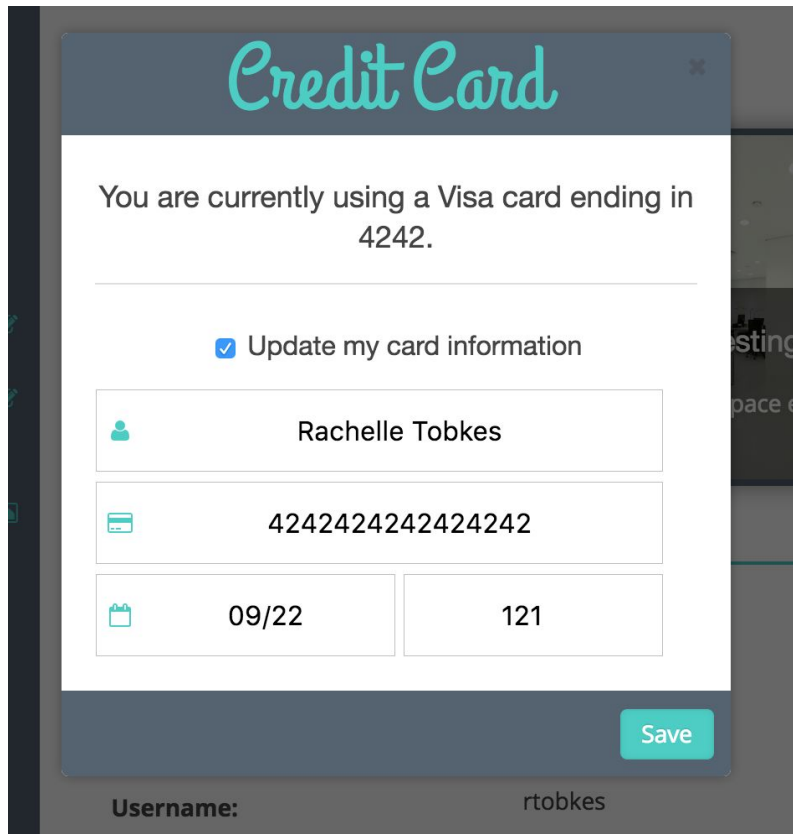


Figure 56 -Account Settings

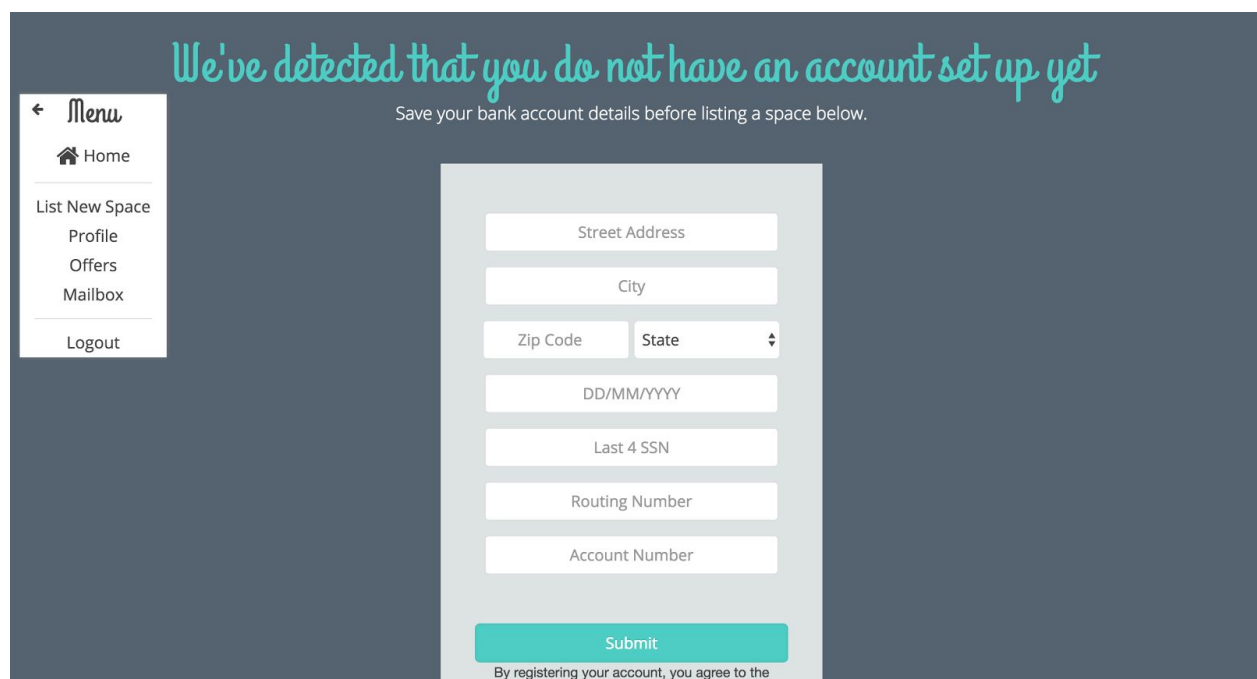


Credit Card

You are currently using a Visa card ending in 4242.

☒ Update my card information

Username: rtobkes

Figure 57 - Save Card Details user story #850

We've detected that you do not have an account set up yet

Save your bank account details before listing a space below.

Menu

- Home
- List New Space
- Profile
- Offers
- Mailbox
- Logout

By registering your account, you agree to the

Figure 58 -Save Bank Account Details user story #851

Appendix C - Sprint Review Reports

Sprint 1 Report

Date: 1/29

Attendees: Rachelle Tobkes, Daniel Alvarez, Pia Celestino

Start time: 1:00PM

End time: 2:00PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #760
- User Story #761
- User Story #762

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- None

Sprint 2 Report

Date: 2/12

Attendees: Rachelle Tobkes, Daniel Alvarez, Pia Celestino

Start time: 2:00PM

End time: 3:00PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #765
- User Story #760
- User Story #764

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- None

Sprint 3 Report

Date: 2/26

Attendees: Rachelle Tobkes, Daniel Alvarez, Pia Celestino

Start time: 9:00PM

End time: 9:30PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #767
- User Story #767
- User Story #759

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- None

Sprint 4 Report

Date: 3/18

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 2:00PM

End time: 2:30PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #768
- User Story #772
-

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- None

Sprint 5 Report

Date: 04/01

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 2:00PM

End time: 2:30PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #818
- User Story #830

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- None

Sprint 6 Report

Date: 04/15

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 2:00PM

End time: 2:30PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #769
- User Story #841

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- None

Sprint 7 Report

Date: 04/29

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 2:00PM

End time: 2:30PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #850
- User Story #851

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- None

Appendix D - Sprint Retrospective Reports

Sprint 1 Retrospective

Date: 1/29

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 2:30

End time: 3:00

What went wrong?

- Did we do a good job estimating our team's velocity?
The velocity should have been greater.
- Did we do a good job estimating the points (time required) for each user story?
Points should have been greater.
- Did each team member work as scheduled?
Yes

What went right?

- Login, Register, and Logout user stories were completely finished, including all documentation.
- Pia accepted our user stories

How to address the issues in the next sprint?

- How to improve the process?
 - Try to finish the use cases and diagrams the first monday of each sprint so that we have more time for refinement, coding, and testing.
- How to improve the product?
 - We will first come up with a design with the help of Pia.

Sprint 2 Retrospective

Date: 2/12

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 4:30

End time: 5:00

What went wrong?

- Did we do a good job estimating our team's velocity?
The velocity should have been a lot greater.
- Did we do a good job estimating the points (time required) for each user story?

Points should have been a lot greater. User stories took longer than we had expected. More researching and troubleshooting was needed than we had anticipated.

- Did each team member work as scheduled?

Yes

What went right?

- Forgot password user story were completely finished, including all documentation. All diagrams were completed for List an Office Space and View an Office Space. Google Maps API was incorporated properly into List an Office Space as well as file upload and front ends completely finished for both user stories.

How to address the issues in the next sprint?

- How to improve the process?
 - Add more points to the user stories to have a better estimate.
- How to improve the product?
 - Redesign some templates to match the overall theme of the website.

Sprint 3 Retrospective

Date: 2/26

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 4:30

End time: 5:00

What went wrong?

- Did we do a good job estimating our team's velocity?
The velocity should have been a lot greater.
- Did we do a good job estimating the points (time required) for each user story?
Points should have been a lot greater. User stories took longer than what we had expected. More researching and troubleshooting was needed than we had anticipated.
- Did each team member work as scheduled?
Yes

What went right?

- Rating system was fully implemented.
- Search algorithm created.
- Search implemented for retrieving nearby office spaces.

How to address the issues in the next sprint?

- How to improve the process?
 - Break up user stories into sub user stories.

How to improve the product?

- Make sure our views are responsive so our application can be viewed on mobile phones and still look nice.

Sprint 4 Retrospective

Date: 3/18

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 2:30

End time: 3:00

What went wrong?

- Did we do a good job estimating our team's velocity?
The velocity should have been a lot greater.
- Did we do a good job estimating the points (time required) for each user story?
Points should have been a lot greater. User stories took longer than what we had expected. More researching and troubleshooting was needed than we had anticipated.
- Did each team member work as scheduled?
Yes

What went right?

- Availability was fully implemented
- User's can now make offers for office spaces per hour, per day, or per month.
- Stripe was integrated to accept payments
- Search filters were implemented.
- The homepage was designed.

How to address the issues in the next sprint?

- How to improve the process?
 - Break up user stories into sub user stories.
- How to improve the product?
 - Discuss with one another algorithms for more efficient implementation.

Sprint 5 Retrospective

Date: 4/01

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 2:30

End time: 3:00

What went wrong?

- Did we do a good job estimating our team's velocity?
Yes, we did a good job estimating the velocity.
- Did we do a good job estimating the points (time required) for each user story?

Yes, points for user stories were estimated correctly.

- Did each team member work as scheduled?

Yes

What went right?

- Rent Now fully implemented
- Credit card Integration fully implemented
- Accepting offers fully implemented.
- Declining offers fully implemented.
- Notifications of offers implemented
- Product owner is satisfied and happy.

How to address the issues in the next sprint?

- How to improve the process?
 - Continue discussions about algorithms and implementations.

How to improve the product?

- Ensure that the overall design matches website theme

Sprint 6 Retrospective

Date: 4/16

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 2:30

End time: 3:00

What went wrong?

- Did we do a good job estimating our team's velocity?

Yes, we did a good job estimating the velocity.
- Did we do a good job estimating the points (time required) for each user story?

Yes, points for user stories were estimated correctly.
- Did each team member work as scheduled?

Yes

What went right?

- Edit profile fully implemented
 - You can change password or delete account from settings in profile.
 - You can edit personal information
- Messaging fully implemented.
 - Receive or send messages to other users

How to address the issues in the next sprint?

- How to improve the process?
 - Continue discussions about algorithms and implementations.
 - Reuse/modify templates to keep similar UI design and conserve time

How to improve the product?

- Ensure that all the requirements are met for each user story
- Ensure that we are following Pia's guidelines

Sprint 7 Retrospective

Date: 4/16

Attendees: Rachelle Tobkes, Daniel Alvarez

Start time: 2:30

End time: 3:00

What went wrong?

- Did we do a good job estimating our team's velocity?
Yes, we did a good job estimating the velocity.
- Did we do a good job estimating the points (time required) for each user story?
Yes, points for user stories were estimated correctly.
- Did each team member work as scheduled?
Yes

What went right?

- Save credit card details was completed
 - User can save and update their credit card details
 - A customer is created on Stripe for the user
- Set up Bank Account completed
 - A user must now set up their bank account with HyperDesk before being able to list a space.
 - A stripe account is created for the user behind the scenes

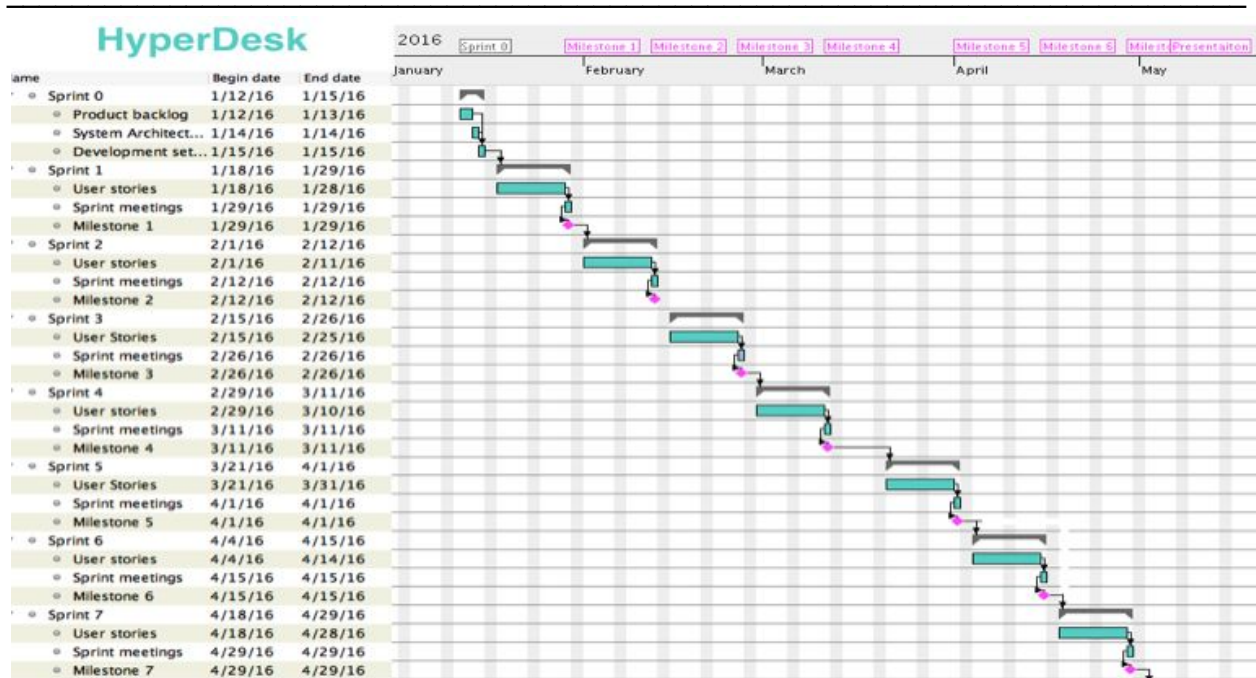
How to address the issues in the next sprint?

- How to improve the process?
 - Continue discussions about algorithms and implementations.
 - Reuse/modify templates to keep similar UI design
 - Fix any bugs from previous user stories that might affect new user stories

How to improve the product?

- Ensure that all the requirements are met for each user story
- Ensure that we are following Pia's guidelines

Appendix E - Project Management



References

"API Reference." Stripe. N.p., n.d. Web. 26 Apr. 2016.

"Dbushell/Pikaday." GitHub. N.p., n.d. Web. 26 Apr. 2016.

"Moment.js 2.13.0." Moment.js. N.p., n.d. Web. 26 Apr. 2016.

"Stripe." Documentation. N.p., n.d. Web. 26 Apr. 2016.