**Individual Assignment 1 – Barrett Long**

**DMIT2028 – System Analysis and Design II**

**Section: A01**

**Instructor: Dwayne Rurka**

**Part One: Case Study: Real Estate Multiple Listing Service (REMLS)**

*The Real Estate Multiple Listing Service (REMLS) supplies information that local real estate agents use to help them sell houses to their clients.*

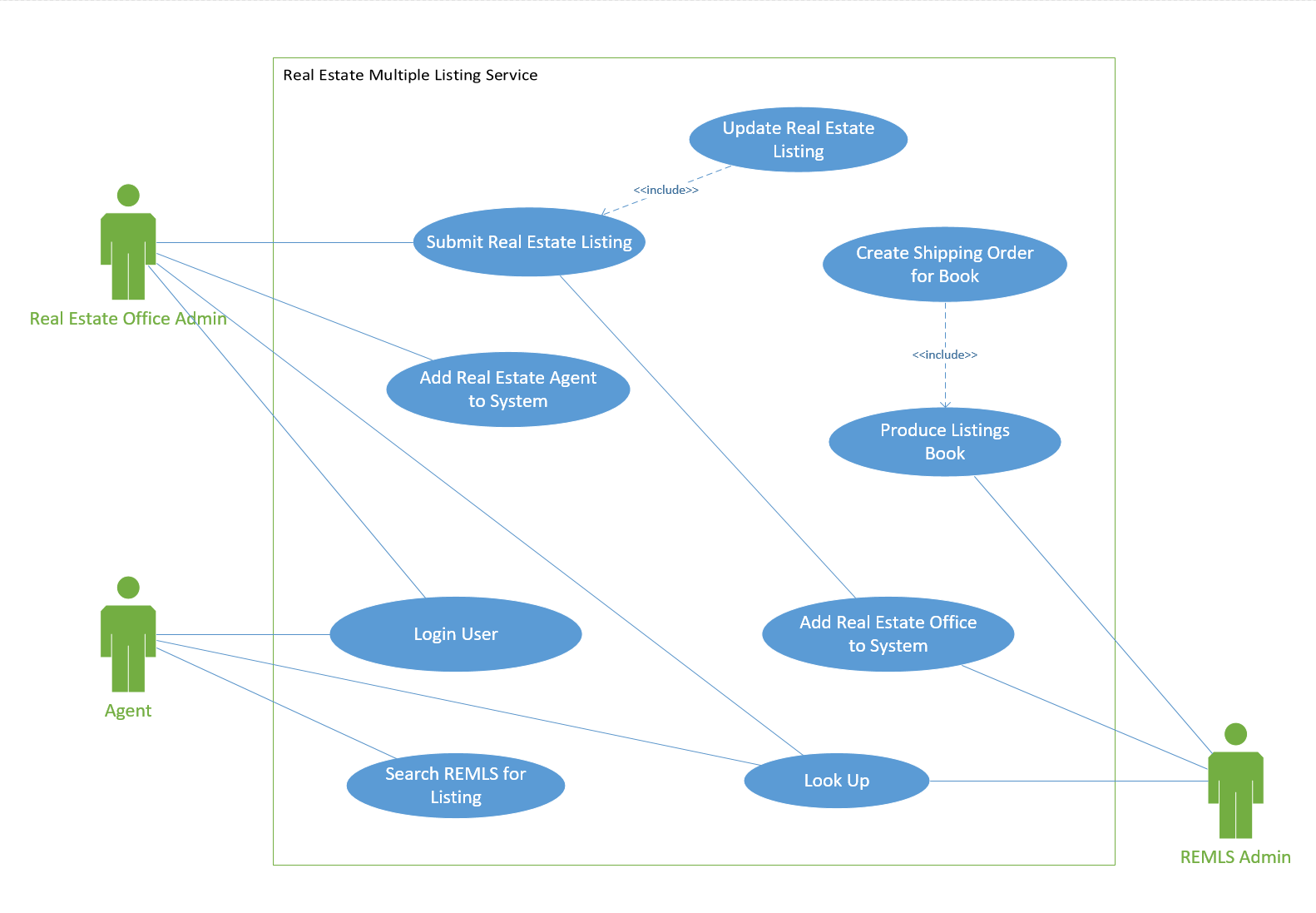
*Agents prepare listings for houses for sale by contracting with homeowners. The agent works for a real estate office, which sends the agents’ listing information to REMLS. Information on a listing includes the address, year built, square footage/meters, number of bedrooms, number of bathrooms, owner name, owner phone number, asking price, and status code.*

*An agent can search the REMLS system to obtain information on a listing that matches client requirements. Information on the house, on the agent who listed the house, and on the real estate office the agent works for is provided. Sometimes agents use this information to call the listing agent or the homeowner directly to make an appointment to show the house.*

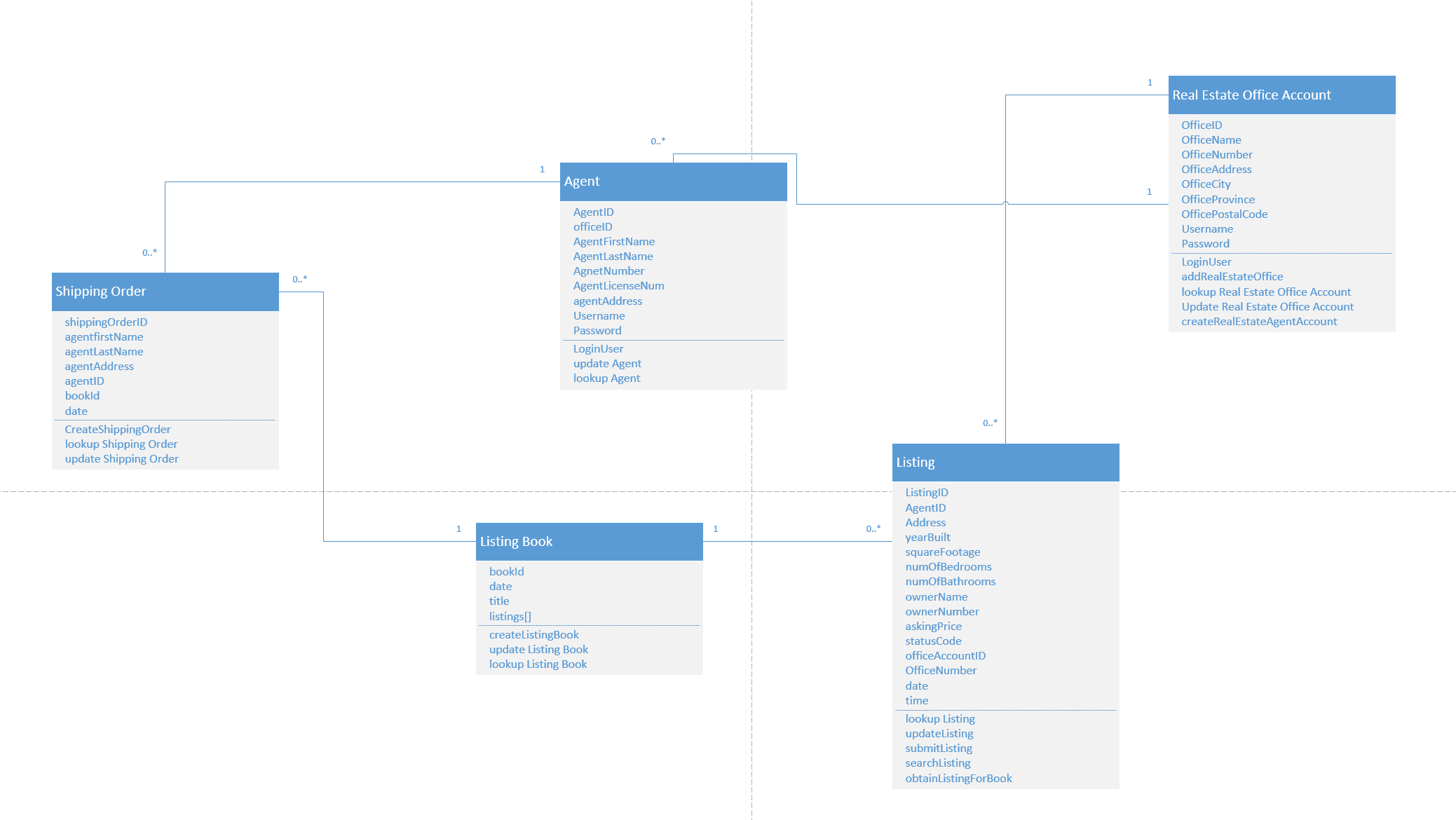
*Twice each month (on the 15th and 30th), the REMLS produces a listing book that contains information on all listings. Many agents want the books (which are easier to flip through), so they are provided even through the information is often out of date. These books are sent to all of the real estate agents even though the information is available online.*

*Sometimes agents and owners decide to change information about a listing, by reducing the price, correcting previous information on the house, or indicating that the house is sold, for example. The real estate offices send in these change requests to REMLS when an agent requests them to do so.*

1. **Use Case Model**



1. **Class Diagram**



1. **Use Case Narrative**

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Submit Real Estate Listing | |
| **Description:** | This method allows a real estate office to submit a real estate listing to the REMLS system on behalf of an agent. | |
| **Actor(s):** | Real Estate Office Admin | |
| **Preconditions:** | Real Estate Office must be in the system  Real Estate Agent must be in the system | |
| **Trigger:** |  | |
| **Normal flow of events:** | **Actor action** | **System Response** |
| 1. LoginUser  (  Username, Password  )    3. LookUp( officeID, agentName )  4. submitListing  (  AgentID, Address, yearBuilt,  squareFootage,  numOfBedrooms,  numOfBathrooms,  ownerName, ownerNumber,  askingPrice, statusCode,  officeAccountID, officeNumber,  date, time  ) | 2. Login User – Return  OfficeAccount:OfficeAccounts  4. Agents[]        5. add listing confirmation |
| **Alternate flow of events:** |  |  |
| **Post-conditions:** | A real estate listing is added to the system | |
| **Outstanding Issues:** |  | |

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Update Real Estate Listing | |
| **Description:** | Used by Real Estate Office to update aspects of a current listing | |
| **Actor(s):** | Real Estate Office Admin | |
| **Preconditions:** | Real Estate Office must be in the system  Real Estate Listing must be in the system | |
| **Trigger:** |  | |
| **Normal flow of events:** | **Actor action** | **System Response** |
| 1. LoginUser  (  Username, Password  )    3. LookUp(address)  4. updateListing  (  AgentID, Address, yearBuilt,  squareFootage,  numOfBedrooms,  numOfBathrooms,  ownerName, ownerNumber,  askingPrice, statusCode  ) | 2. Login User – Return  OfficeAccount:OfficeAccounts  4. Listing:Listings        5. Update Listing Confirmation |
| **Alternate flow of events:** |  |  |
| **Post-conditions:** | A real estate listing is updated | |
| **Outstanding Issues:** |  | |

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Add Real Estate Agent to system | |
| **Description:** | Used to add a real estate agent that works for an real estate office to the system | |
| **Actor(s):** | Real Estate Office Admin | |
| **Preconditions:** | Real Estate Office must be in the system | |
| **Trigger:** |  | |
| **Normal flow of events:** | **Actor action** | **System Response** |
| 1. LoginUser  (  Username,  Password  )  3. createRealEstateAgentAccount  (  officeID, agentFirstName,  AgentLastName, AgentNumber,  AgentLicenseNum, agentAddress,  Username, Password  ) | 2. Login User – return  officeAccount:officeAccounts  4. Confirmation |
| **Alternate flow of events:** |  |  |
| **Post-conditions:** | A Real Estate Agent Account is added to the system under the office account with username and password | |
| **Outstanding Issues:** |  | |

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Add Real Estate Office to System | |
| **Description:** | Adds a Real Estate Office with a unique ID, username and password to access the REMLS system | |
| **Actor(s):** | REMLS Admin | |
| **Preconditions:** | No Preconditions | |
| **Trigger:** |  | |
| **Normal flow of events:** | **Actor action** | **System Response** |
| 1. addRealEstateOffice  (  OfficeName, OfficeNumber,  OfficeAddress, OfficeCity,  officeProvince, officePostalCode,  Username, Password  ) | 2. confirmation |
| **Alternate flow of events:** |  |  |
| **Post-conditions:** | Real Estate Office added to the system | |
| **Outstanding Issues:** |  | |

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Search REMLS for Listing | |
| **Description:** | Used to Search the listings in the system | |
| **Actor(s):** | Agent | |
| **Preconditions:** | There must be listings in the system  The real estate agent must be in the system | |
| **Trigger:** |  | |
| **Normal flow of events:** | **Actor action** | **System Response** |
| 1. LoginUser  (  Username, Password  )    3. searchListings  (  DesiredLocation,  SquareFootRange, DesiredAge,  DesiredNumOfBedrooms,  DesiredNumOfBathrooms,  PriceRange, statusCode  )    5. lookup(listingID) | 2. Agent:Agents  4. Listings[] – Any listings that falls in  the desired ranges.  6. return listing:listings |
| **Alternate flow of events:** |  |  |
| **Post-conditions:** | Returns a single listing that fits the desired search criteria from the agent | |
| **Outstanding Issues:** |  | |

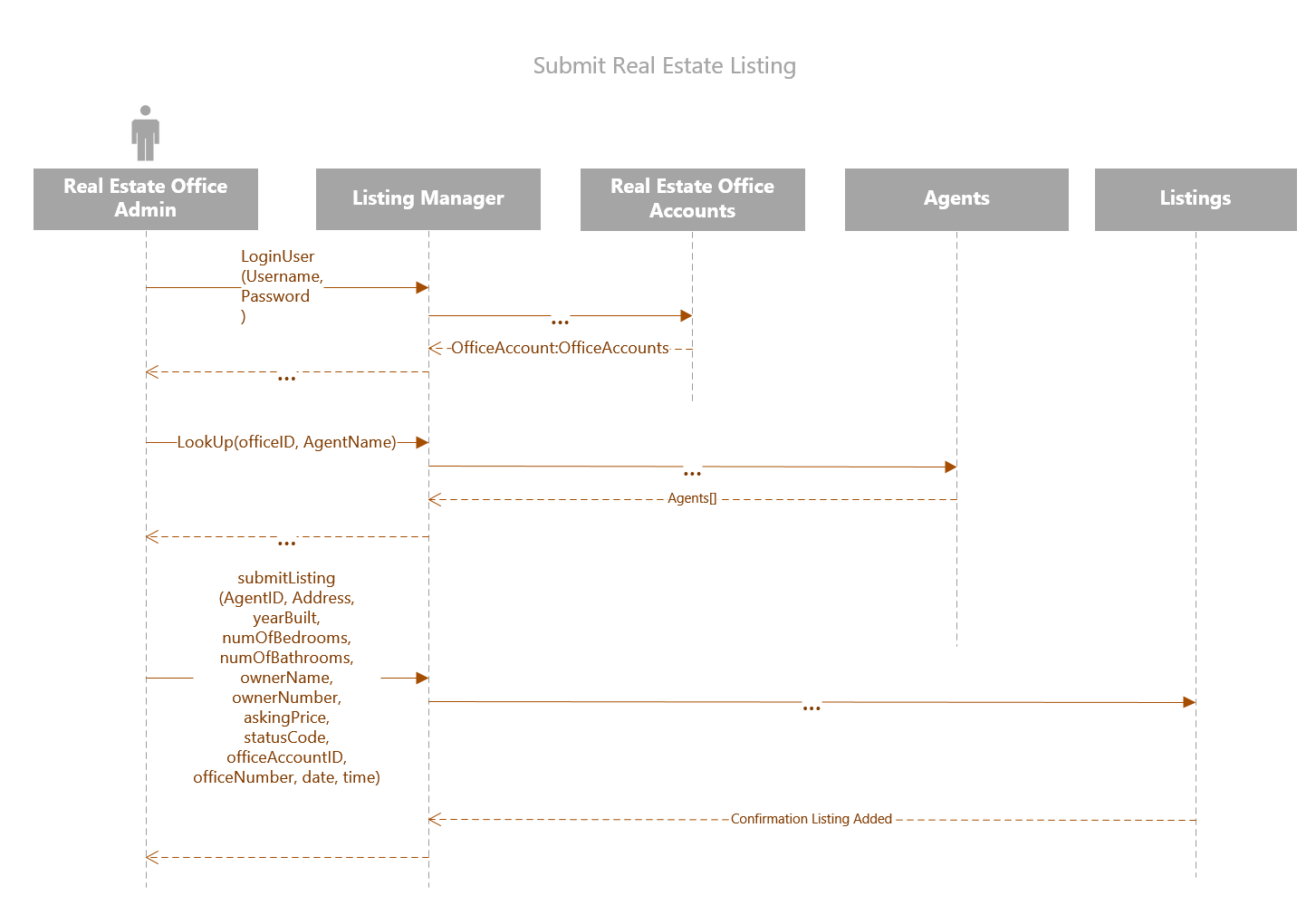
|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Produce Listing Book | |
| **Description:** | Selects and sorts all listings for a bi monthly book | |
| **Actor(s):** | REMLS Admin | |
| **Preconditions:** | There must be listings in the system to add to the book | |
| **Trigger:** |  | |
| **Normal flow of events:** | **Actor action** | **System Response** |
| 1. obtainListingsForBook  (  statusCode  )  3. CreateListingBook  (  Listings[],  date,  title  ) | 2. listings[]  4. confirmation – new book created with  unique bookID |
| **Alternate flow of events:** |  |  |
| **Post-conditions:** | Creates a collection of available listings and adds them to a book to be printed | |
| **Outstanding Issues:** |  | |

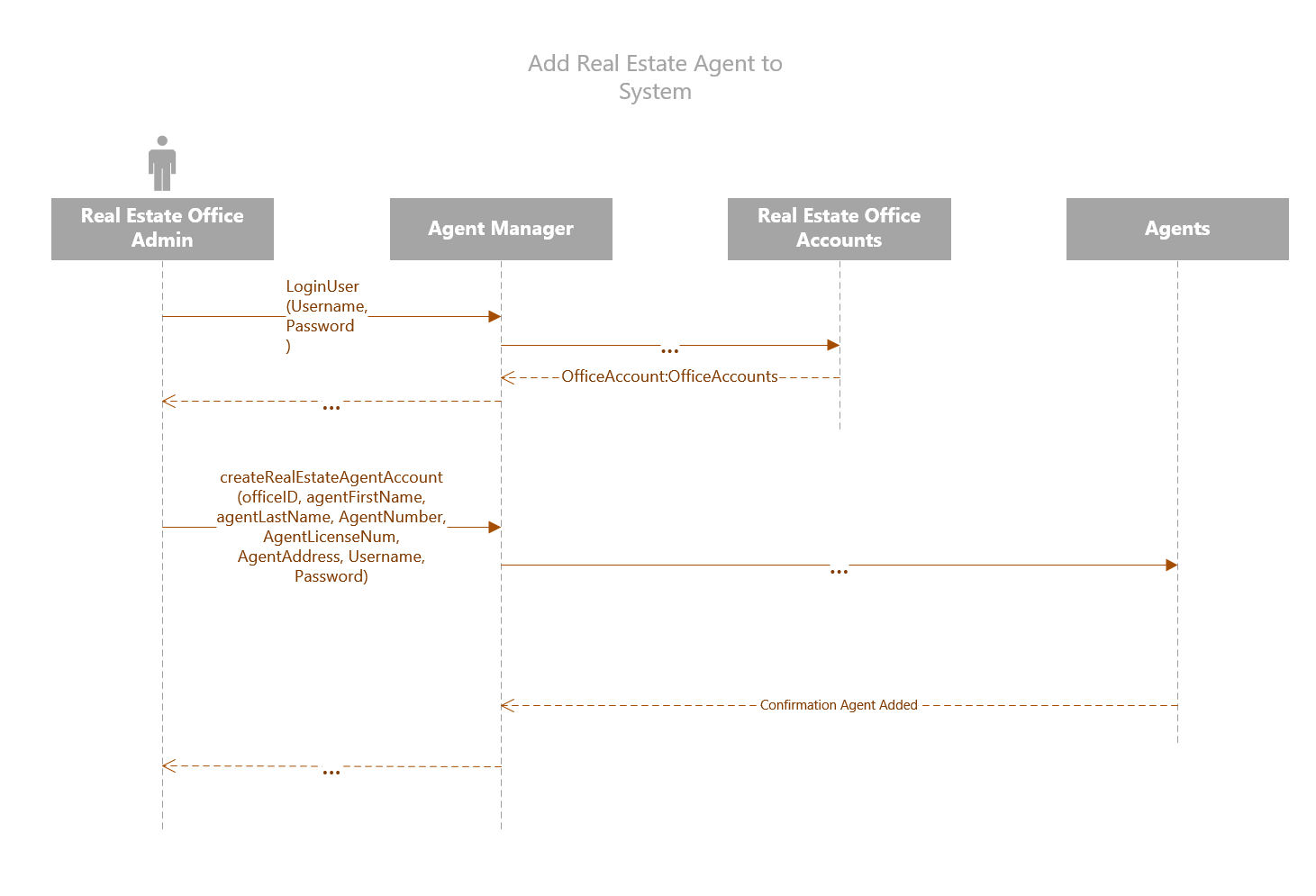
|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Create Shipping Order for Book | |
| **Description:** | Creates a shipping order to track the shipping of books to agents | |
| **Actor(s):** | REMLS Admin | |
| **Preconditions:** | There must be a book in the system  There must be Real Estate Agents in the system | |
| **Trigger:** |  | |
| **Normal flow of events:** | **Actor action** | **System Response** |
| 1. lookup(Agents)  3. lookup(date)  5. createShippingOrder  (  agentFirstName,  agentLastName,  agentAddress,  agentID,  bookID, date  ) | 2. return Agents[]  4. return Books[]  6. Confirmation – new shippingOrder  created with unique ID |
| **Alternate flow of events:** |  |  |
| **Post-conditions:** | A new shipping order has been created in the system | |
| **Outstanding Issues:** |  | |

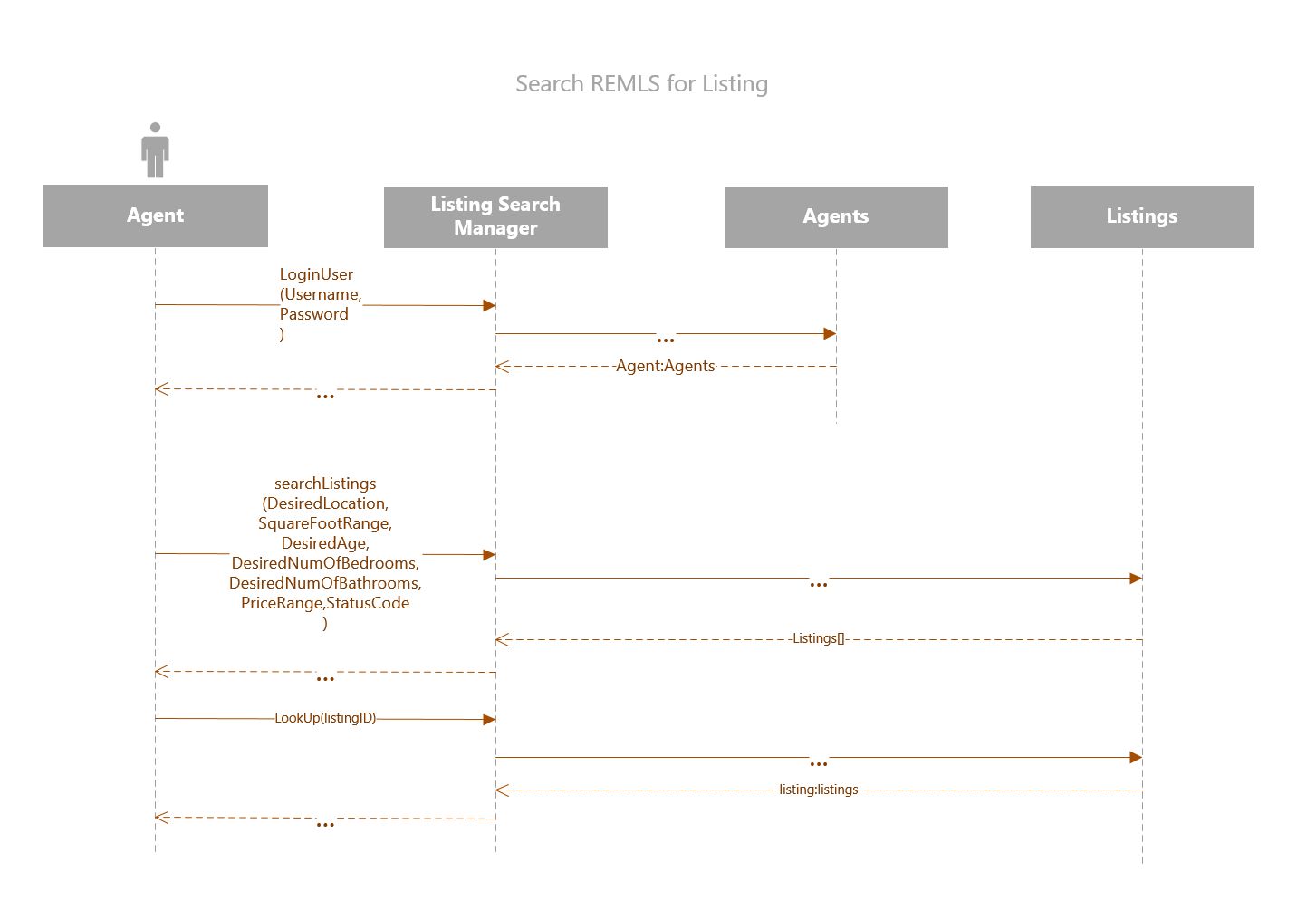
|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Login User | |
| **Description:** | Used to log users into the system | |
| **Actor(s):** | Agent  Real Estate Office Admin | |
| **Preconditions:** | Agent Must be in the system,  Real Estate Office must be in the system | |
| **Trigger:** |  | |
| **Normal flow of events:** | **Actor action** | **System Response** |
| 1. LoginUser  (  Username,  Password  ) | 2. return Object() |
| **Alternate flow of events:** |  |  |
| **Post-conditions:** | The user is logged in and the user info is returned | |
| **Outstanding Issues:** | * Agents User * Real Estate Office User | |

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | LookUp | |
| **Description:** | Used to look up information in the system | |
| **Actor(s):** | Agent  Real Estate Office Admin  REMLS Admin | |
| **Preconditions:** |  | |
| **Trigger:** |  | |
| **Normal flow of events:** | **Actor action** | **System Response** |
| 1. LookUp  (  Argument(s)  ) | 2. return Object() or <List>Objects |
| **Alternate flow of events:** |  |  |
| **Post-conditions:** | Depending on arguments will return either a specific object or a list of objects | |
| **Outstanding Issues:** | * OfficeID * AgentName or AgentID or <list>Agents * listingID * date | |

1. **Sequence Diagram**







**Part Two: Research**

1. ***Discuss the similarities and differences between Rational Unified Process and Agile Unified Process***

The Rational Unified Process and the Agile Unified Process are similar in that they are both used as ways of structuring the SDLC in a logical and maintainable way. Each outlines the procedures needed to take a project from Inception to Delivery. The Rational Unified Process differs from the Agile Unified Process in that relies upon a series of strongly defined milestones with team members taking on distinct and specific roles within the process. Each milestone (Inception, Elaboration, Construction, and Transition) flow into each other and are structured in a very sequential manner. The Agile Unified Process does not have such rigid guidelines and works as a much quicker and iterative process. When using AUP member of a team tackle all aspects of the SDLC and work to constantly iterate and add functionality to the project in much shorter time periods.

***References:***

*Waterfall, RUP and Agile: Which is right for you?* <http://www.ebizq.net/topics/dev_tools/features/11821.html>

*The Agile Unified Process (AUP) Homepage*

<http://www.ambysoft.com/unifiedprocess/agileUP.html>

*Rational Unified Process: Best Practices for Software Development Teams*

<http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf>

1. ***Describe the Scrum approach to systems development.***

The Scrum approach to system development is a subset of the Agile Unified Process that is a widely-adopted framework for development. Scrum is designed around the practise of code Sprints. Sprints focus on a single requested aspect of functionality and are implemented quickly to the overall system. Scrum processes allow for increased productivity and help organizations adjust to rapidly changing system requirements.

***References:***

*What is AGILE? What is Scrum? cPrime*

<https://www.cprime.com/resources/what-is-agile-what-is-scrum/>

*What is Scrum? An Agile Framework for Completing Complex Projects*

<https://www.scrumalliance.org/why-scrum>

1. ***List the developers of UML and the group’s popular nickname.***

UML was developed as a new approach to software design development in the early 90’s by Grady Booch, Ivar Jacobson and James Rumbaugh from Rational Software. In 1997 it was taken over by the Object Management Group and has been under there control ever since. The group of Grady Booch, Ivan Jacobson and James Rumbaugh also went by the moniker of The Three Amigos.

***References:***

*Unified Modeling Language – Wikipedia*

<https://en.wikipedia.org/wiki/Unified_Modeling_Language>

*The Three Amigos*

<https://en.wikipedia.org/wiki/The_Three_Amigos>