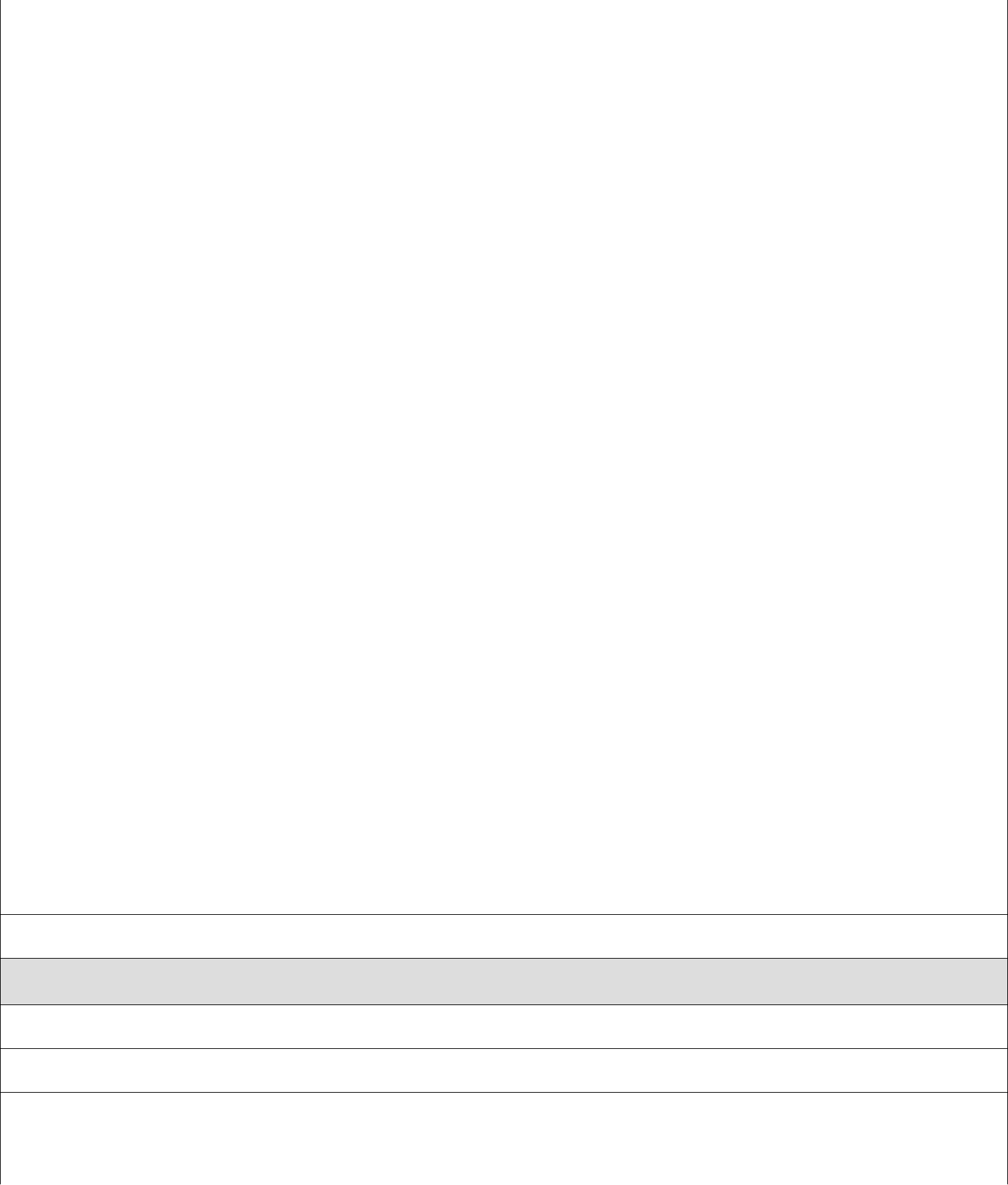
**Name:**

**Registration #: 610564**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | **MUM-Auction: Use Case Description** | |
|  |  | | |  |
| **Use Case Number: 1** | | | |  |
| **Name** | |  | User CRUD |  |
|  |  | |  |  |
| **Brief description** | | | This use case allows the user to create profile on the system | |
|  |  | |  |  |
| **Actors** | | | User(buyer-seller) |  |
|  |  | |  |  |
|  |  |  |  |  |
|  |  | |  |  |
| **Flows of Events:** | | |  |  |
| **1. Basic Flows** | | |  |  |
|  |  | | |  |
| **1.1.0 Create user profile** | | | |  |
|  |  |  |  |  |
| **Step** |  |  | **User Actions** | **System Actions** |
|  |  |  | |  |
| 1 |  | The user clicks on sign up button | | The system displays the sign-up profile form with |
|  |  |  |  | the fields for firstname, lastname, email, password. |
|  |  |  |  |  |
|  |  |  | |  |
| 2 |  | The user fills out the form and requests the system | | The system verifies that there’s no other profile in |
|  |  | to save the details | | the database with the same email address and |
|  |  |  |  | saves the user profile and returns the success message |
|  |  |  |  | on success or a fail message in case of failure. In |
|  |  |  |  | case another profile exists with the email address, |
|  |  |  |  | the system returns the message indicating a |
|  |  |  |  | duplicate entry exists. |
|  |  | |  |  |
| **Postconditions** | | |  |  |
| The user profile is persisted in the system | | | |  |



**Business Rules**

No duplicate user profiles. A unique profile is identified by email address.

**1.1.1 Read/view user Profile**

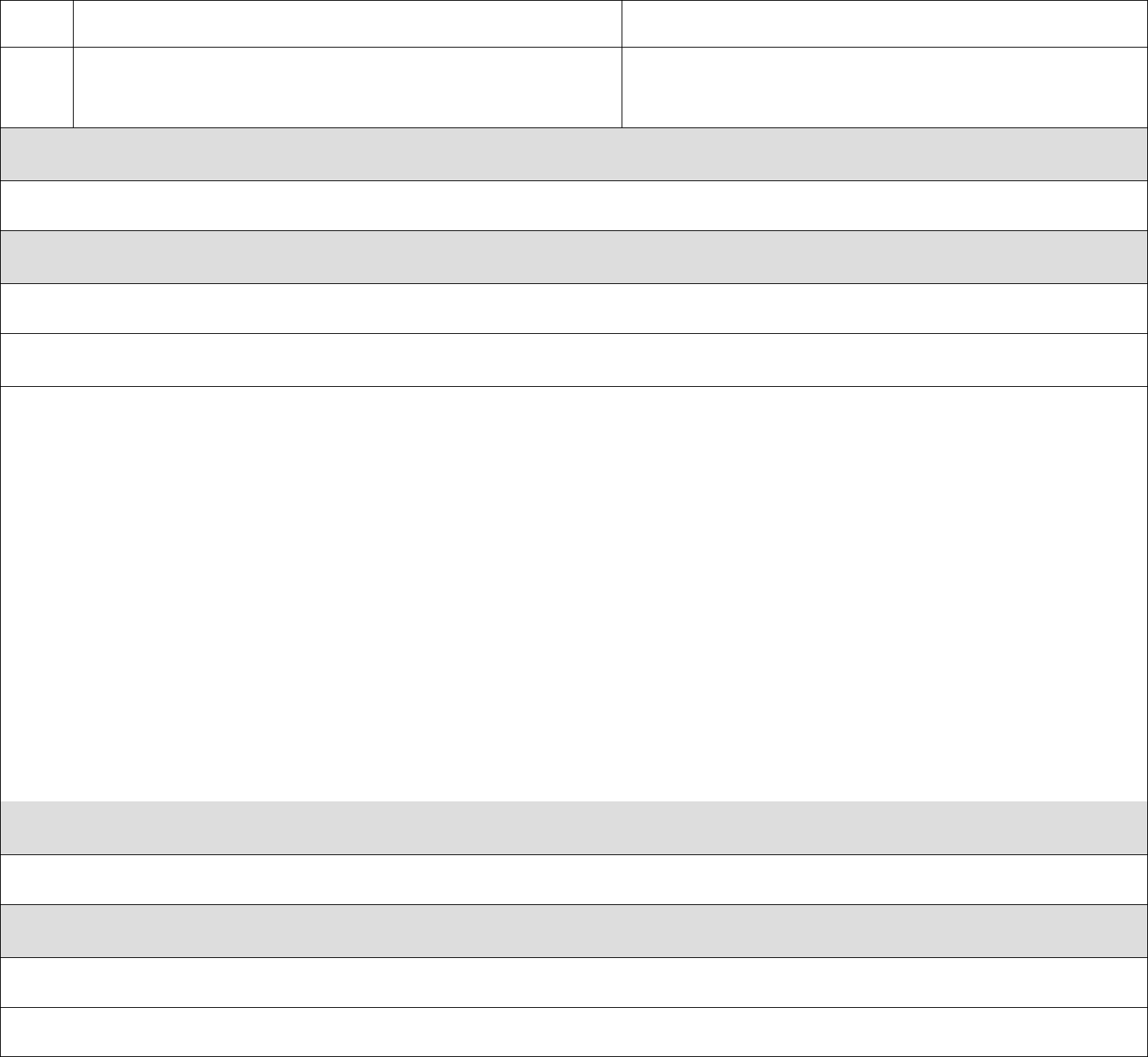
|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | The user views his profile | |  | | --- | | The system returns the profile of the user as a | | string with the names (firstname and lastname), | | Email. | |  | |
|  |  |  |
|  |  |  |
|  |  |  |
| **1.1.2 Update User Profile** | |  |
|  |  |  |
| **Step** | **User Actions** | **System Action** |
|  |  |  |
| 1 | The user view his profile | |  | | --- | | The system returns the profile of the user as a | | string with the names (firstname and lastname), | | Email. | |
|  |  |  |
| 2 | The user update profile data | The system displays an editable user profile |
|  |  |  |
|  |  |  |
|  |  |  |
| **Postconditions** | |  |
| The user profile will be updated | |  |
|  |  |  |
| **Business Rule** | |  |
| The email field should be unwritable. | |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Number: 2** | | |  |
| **Name** | | POST CRUD |  |
|  |  | |  |
| **Brief description This** use case provides a functionality for managing posts | | | |
|  |  |  |  |
| **Actors** | | User(seller) |  |
|  |  | |  |
|  |  |  |  |
| **Preconditions** | | |  |
| The user must log in to the system | | |  |
|  |  | |  |
| **Flows of Events:** | | |  |
| **1. Basic Flows** | | |  |
|  |  | |  |
| **1.0 Create Post** | | |  |
|  |  |  |  |
| **Step** |  | **User Actions** | **System Actions** |
|  |  |  |  |
| 1 |  | The user creates post | The system displays the post form with description, an option to upload at least one photo, a minimum price and expiration date. |
|  |  |  |  |
|  |  |  |  |
| 2 |  | The user fills out the form , upload photos of the product , choose expiration date and requests the system to save the post | The system saves the post and returns |
|  |  |  | a success message or a fail message in case of an exception if the size of the photo is too big or the expiration date is already past. |
|  |  | |  |
| **Postconditions** | | |  |
| The user post is persisted in the system | | |  |
|  |  | |  |
| **Business Rule** | | |  |
| Expiration date must not be passed | | |  |
|  |  | |  |
|  |  | |  |
| **1.1 Read/view post** | | |  |
|  |  |  |  |
| **Step** |  | **User Actions** | **System Actions** |
|  |  |  |  |
| 1 |  | The user requests for a list of posts | The system returns a list of all posts. |
|  |  |  |  |
| 2 |  | The user selects to view a post they want from | The system returns the post object as a string |
|  |  | the list. | with item data, item photo and expiration date with the last price. |
|  |  | |  |
|  |  | |  |
| **1.2 Update post** | | |  |
|  |  |  |  |
| 1 |  | The seller requests a list of posts | The system returns a list of posts. |
|  |  |  |  |
| 2 |  | The seller selects the post they want to update | The system displays an editable form with the |
|  |  |  | post details |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| 3 | The seller changes the parts they want to update | The system updates the post and returns a |
|  | and requests system to save the new details | success message or a fail message on exception. |
|  |  |  |
| **Postconditions** | |  |
| The post details will be updated | |  |
|  |  |  |
| **Business Rule** | |  |
|  |  |  |
| The minimum price should not be writable after the users start bidding . | |  |
|  |  |  |
| **1.1.3 Delete post** | |  |
|  |  |  |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |
| 1 | The seller requests for a list of posts | The system returns a list of all posts. |
|  |  |  |
| 2 | The seller selects to delete a post from a list of | The system displays a confirmation dialogue |
|  | posts | window |
|  |  |  |
| 3 | The admin selects OK on the confirmation dialog | The system confirms the post is not assigned to |
|  | window to confirm deleting the course | any current bidding and deletes and returns a success |
|  |  | message or a fail message on exception. |
|  |  |  |
| **Postconditions** | |  |
| The post will be deleted | |  |
|  |  |  |
|  |  |  |
| **Business Rule** | |  |
| The post must not have current buyer bidding on this item | | |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case Number: 3** | | | |  |
| **Name** | |  | Block CRUD |  |
|  |  | |  |  |
| **Brief description** | | | This use case provides a functionality for managing blocks | |
|  |  | |  |  |
| **Actors** | | | Admin |  |
|  |  | |  |  |
|  |  |  |  |  |
| **Preconditions** | | |  |  |
| The admin must logged in to the system | | | |  |
|  |  | |  |  |
| **Flows of Events:** | | |  |  |
| **1. Basic Flows** | | |  |  |
|  |  | |  |  |
| **1.0 Create Block** | | |  |  |
|  |  |  |  |  |
| **Step** |  | **User Actions** |  | **System Actions** |
|  |  |  | |  |
| 1 |  | The admin calls the create block command | | The system displays the create block form with name, |
|  |  |  |  | sequence number, start date and end date. |
|  |  |  | |  |
| 2 |  | The Admin enters the block start and end dates, the block | | The system checks if there’s no other block with same |
|  |  | identifier number, and a name and requests the system to | | name, start and end dates and saves the block and |
|  |  | save the details |  | returns a success message or a fail message on failure. |
|  |  |  |  | It should return a specific message when a duplicate |
|  |  |  |  | block exists. |
|  |  | |  |  |
| **Postconditions** | | |  |  |
| The block is persisted in the system | | | |  |
|  |  | |  |  |
|  |  |  |  |  |
| **Business Rules** | | |  |  |
| A block must be unique. A unique block is identified by the name and the start and end dates | | | | |
|  |  | |  |  |
|  |  | |  |  |
| **1.1 Read/view Block** | | |  |  |
|  |  |  |  |  |
| **Step** |  | **User Actions** |  | **System Actions** |
|  |  |  | |  |
| 1 |  | The admin requests for a list of blokcs | | The system returns a list of all blocks. |
|  |  |  | |  |
| 2 |  | The admin selects to view a block they want | | The system returns the block object as a string |
|  |  |  |  | with block name, start and end dates |
|  |  | |  |  |
|  |  | |  |  |
| **1.2 Update Block** | | |  |  |
|  |  |  | |  |
| 1 |  | The admin requests for a list of blocks | | The system returns a list of all blocks. |
|  |  |  | |  |
| 2 |  | The admin selects the block they want to update | | The system displays back an editable form with |
|  |  |  |  |  |

the block details pre-populated

* The admin changes the parts they want to update and The system updates the block record and returns a

|  |  |
| --- | --- |
| requests the system to save the new details | success message or a fail message on exception. |

**Postconditions**

The block details will be updated

**Business Rule**

The name, start and end dates should be unwritable.

**1.1.3 Delete Block**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
|  |  |  |
| 1 | The admin requests for a list of blocks | The system returns a list of all blocks. |
|  |  |  |
| 2 | The admin selects to delete the block they want | The system displays a confirmation dialogue |
|  |  | window |
|  |  |  |
| 3 | The admin selects OK on the confirmation dialog | The system checks that the block is not linked to |
|  | window to confirm deleting the block | any section and deletes the it and returns a success |
|  |  | message or a fail message on an exception. |
|  |  | Otherwise, it will return a message indicating the |
|  |  | block is associated to some sections. |
|  |  |  |

**Postconditions**

The block will be deleted

**Business Rule**

The block must not be linked to some sections