**Friend management system design specs and API document**

1. Used tech and tools

* Use Mysql to store the friend, block and subscription data, may consider to use mongo as well.
* Use spring boot framework, log4j, mysql connector and jpa repository for data CRUD with database;
* Use docker file and deploy it to azure AKS

1. Database schema design

* 3 tables been used, **subscription, friend and block**

| **Table name: subscription, to store the subscription info** | | |
| --- | --- | --- |
| **column** | **type** | **remark** |
| id | int | DB auto increase index |
| requestor | String | Json string for target friend emails,  For example:  {“emails”:[“1@1.com”, “2@2.com“]},  This means 1 and 2 subscribe to 3 |
| target | String | Email address, for example: “3@3.com” |
| Last\_modified\_timestamp | long | Last modified timestamp for this info |
| remark | string | Remark for this info |

| **Table name: block, to store the block info** | | |
| --- | --- | --- |
| **column** | **type** | **remark** |
| id | int | DB auto increase index |
| blocked | String | Email address, for example: “3@3.com” |
| blocked\_by | String | Json string for target friend emails,  For example:  {“emails”:[“1@1.com”, “2@2.com“]},  This means 3 are been blocked by 1 and 2 |
| Last\_modified\_timestamp | long | Last modified timestamp for this info |
| remark | string | Remark for this info |

| **Table name: friend, to store the friend relationship info** | | |
| --- | --- | --- |
| **column** | **type** | **remark** |
| id | int | DB auto increase index |
| Last\_modified\_timestamp | long | Last modified timestamp for this friend relationship info |
| remark | string | Remark for this info |
| source | string | Email address, for example: “3@3.com” |
| targets | string | Json string for target friend emails,  For example:  {“emails”:[“1@1.com”, “2@2.com“]},  This means both 1 and 2 are the friend of 3.  For every new added friend relationships, 2 rows in this table would be created. For example, if 1 and 2 are been requested to add as friend, system would create 2 rows. 1) source=1 and target=2, 2) source=2 and target=1 |

1. API documents

* **Add friend API**

**Description: add 2 email address as friend relationships**

**Method: POST**

**URL: http://{ip}:9008/friend/add**

**Request parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| **parameter** | **type** | **required** | **remark** |
| friends | String list | yes | For example: {  "friends":  [  "andy@example.com",  "john@example.com"  ]  } |

**Response:**

|  |
| --- |
| {  "success": true  } |

* **Get friend list API**

**Description: get the friend list for input email**

**Method: POST**

**URL: http://{ip}:9008/friend/get**

**Request parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| **parameter** | **type** | **required** | **remark** |
| email | String | yes | For example: {  "email":  {  "andy@example.com"  } |

**Response:**

|  |
| --- |
| {  "success": true,  "friends" :  [  "john@example.com"  ],  "count" : 1  } |

* **Get common friend list API**

**Description: get the common friend list for input 2 emails**

**Method: POST**

**URL: http://{ip}:9008/friend/getcommon**

**Request parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| **parameter** | **type** | **required** | **remark** |
| friend | String list | yes | {  "friends":  [  "andy@example.com",  "john@example.com"  ]  }  Get common friend list for above 2 emails |

**Response:**

|  |
| --- |
| {  "success": true,  "friends" :  [  "common@example.com"  ],  "count" : 1  } |

* **Subscription API**

**Description: subscribe one email to another**

**Method: POST**

**URL: http://{ip}:9008/subscription**

**Request parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| **parameter** | **type** | **required** | **remark** |
| requestor | String | yes | Sample value: "requestor": "test@example.com" |
| target | String | yes | Sample value: "target": "lisa@example.com"  Test email is subscribing to lisa |

**Response:**

|  |
| --- |
| {  "success": true  } |

* **Block API**

**Description: block one email to another**

**Method: POST**

**URL: http://{ip}:9008/block**

**Request parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| **parameter** | **type** | **required** | **remark** |
| requestor | String | yes | Sample value: "requestor": "test@example.com", |
| target | String | yes | Sample value: "target": "lisa@example.com",  Test would block lisa |

**Response:**

|  |
| --- |
| {  "success": true  } |

* **Retrieve update list API**

**Description: retrieve available update list for one email**

**Method: POST**

**URL: http://{ip}:9008/retrieve**

**Request parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| **parameter** | **type** | **required** | **remark** |
| sender | String | yes | Sample value: " sender": "test@example.com", |
| text | String | No | Sample value: "text": "hello, lisa@example.com", |

**Response:**

|  |
| --- |
| {  "success": true,  "recipients":  [  "lisa@example.com",  "kate@example.com"  ]  } |