Paths

**User Controller**

**/users/login: Login page**

* Requests
  + POST(String username, String password)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| Username == null | new ResponseEntity<>("Username cannot be empty", HttpStatus.BAD\_REQUEST); |
| Password == null | new ResponseEntity<>("Password cannot be empty", HttpStatus.BAD\_REQUEST); |
| Username does not exist in users table | new ResponseEntity<>("User does not exist",HttpStatus.BAD\_REQUEST); |
| Username exists in users table but password does not match password in users table | new ResponseEntity<>("Incorrect Password", HttpStatus.BAD\_REQUEST); |
| Username and password match a value | new ResponseEntity<>(user.getId(),HttpStatus.OK); |

**/users/createUsers: Create a new user**

* Requests
  + POST(User user)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| User == null | "User cannot be null"; |
| User.getUsername()==null | “Username cannot be null” |
| user.getPassword() == null | “Password cannot be null” |
| Password is less than 8 characters and/or contains no uppercase or lowercase letters | "Password must be 8 characters or longer Password must contain at least one uppercase  Password must contain at least one lowercase level"; |
| New User has valid credentials | "New User " + user.getUsername() + " Created"; |

**/users: Display all user information**

* Requests
  + GET()
* Response

|  |  |
| --- | --- |
| **Case** | **Return Value: JSONObject[ ]** |
| All | Properties of all users |

**/users/{userId}/deleteUser: Deletes a user**

* Requests
  + DELETE()
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| User.findById(userId)==null | "User with id " + id + " not found" |
| Is valid user | "User " + user.getUsername() + " Deleted"; |

**/users/{userId}/editUser: Edit a user’s properties**

* Requests
  + POST(User user)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| User == null | “User cannot be null” |
| User.getUsername()==null | “Username cannot be null” |
| user.getPassword() == null | “Password cannot be null” |
| Password is less than 8 characters and/or contains no uppercase or lowercase letters | "Password must be 8 characters or longer Password must contain at least one uppercase  Password must contain at least one lowercase level"; |
| Is valid user | "User " + user.getUsername() + " Modified"; |

**/users/{userId}/cash: Gets the User’s current cash amount**

* Requests
  + GET()
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: Double** |
| All | User’s cash |

**/users/{adminId}/banUser: Bans a user**

* Requests
  + POST(int userId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| User with id “adminId” is not an admin | “You are not an admin” |
| Else | "User " + user.getUsername() + " Banned"; |

**users/{adminId}/unbanUser: Unbans a user**

* Requests
  + POST(int userId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| User with id “adminId” is not an admin | “You are not an admin” |
| Else | "User " + user.getUsername() + " unbanned"; |

**/users/admin/addAdmin: Gives another user admin right**

* Requests
  + POST(int userId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| User.findById(userId)==null | "User with id " + id + " not found" |
| Is valid user | "User " + user.getUsername() + " Deleted"; |

**/users/{userId}/friends**

* Requests
  + GET()
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: List<JSONObject>** |
| All | Returns the Ids and Username of the user’s friend |

**/users/addFriend**

* Requests
  + POST(userId, friendId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| Already Friends with User | "You are already friends with this user"; |
| Else | "Friend " + users.findById(friendId).getUsername() + " added"; |

**/users/removeFriend**

* Requests
  + POST(userId, friendId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| Invalid Friend Id | "Invalid Friend Id"; |
| Else | "Friend " + friend.getFriend().getUsername() + " removed"; |

**/users/friendsSearch**

* Requests
  + GET(String index)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: List<User>** |
| All | Returns a list of users whose username, first name, and/or last name start with the given index |

**Inventory Controller**

**/users/{userId}/portfolio/sell: Sell one stock as provided by stockId**

* Requests
  + POST(int stockId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| Stock id does not exist in table | "Invalid Stock Id" |
| User id does not exist in table | “Invalid User Id” |
| User does not own any shares of the stock | "User does not own any shares of" + stocks.findSymbolById(stockId); |
| Successful sell | " + num + " stock(s) of " + stock.getSymbol() + " sold" |

**/users/{userId}/portfolio/buy: Buy one stock as provided by stockId**

* Requests
  + POST(int stockId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| Inventory found by Stock Id and User Id == null | "Invalid stockId or userId" |
| Successful purchase | " + num + " stock(s) of " + stock.getSymbol() + " purchased" |

**/users/{userId}/portfolio: Displays user’s stocks**

* Requests
  + GET()
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: JSON Object[ ]** |
| All | id (stock id)  Stock.symbol (stock symbol)  Stock.company (Company Name)  Stock.price (current price)  Stock.quantity (Number of the stocks the user owns) |

**/users/{userId}/portfolioValue: Returns the value of the user’s portfolio**

* Requests
  + GET()
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: Double** |
| All | Portfolio Value |

**Stock Controller**

**/stock: Displays information about a stock**

* Requests
  + GET(int stockId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: JSONObject** |
| All | Id (Stock Id)  Symbol (Stock Symbol)  Company (Company Name)  Price (Current Value) |

**/stock/search: Displays information about stocks starting with the given index**

* Requests
  + GET(String index)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: JSONObject[ ]** |
| All | Symbol (Stock Symbol)  CompanyName (Company Name)  PrimaryExchange (Which Exchange the company belongs to)  LatestPrice (Stock Price)  Change (Increase or decrease in price since trading opened)  Change percent (Percent that the stock price has increased or decreased) |

**/stock/historicData: Returns 1 Month of price data for a stock**

* Requests
  + GET(int stockId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: JSONObject[ ]** |
| All | Date: (YYYY-MM-DD)  Close: Price at end of trading hours |

**/stock/historicData/{range}: Returns price data for a stock as determined by a range**

* Requests
  + GET(int stockId)
  + Ranges:
    - “5y”: 5 years
    - **“**2y”: 2 years
    - “1y”: 1 year
    - “ytd”: year to date
    - “6m”: 6 months
    - “3m”: 3 months
    - “1m”: 1 month
    - “1d”: 1 day
    - Any other range: 1 month
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: JSONObject[ ]** |
| All | Date: (YYYY-MM-DD)  Close: Price at end of trading hours |

**Chat Controller**

**/chat/newChat: Create a new chat**

* Requests
  + Post(int userId, int friendId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: String** |
| User with id=”userId” already have an existing chat with user with id=”friendId” | You already have an existing chat with this user |
| Else | Group Id (as String) |

**/chat/getGroupId: Create a new chat**

* Requests
  + Get(int userId, int friendId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: int** |
| All | Group Id |

**Transaction Controller**

**Path: users /{userId}/transaction**

**Return information about user’s transaction**

* Requests
  + GET(int userId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: JSONObject** |
| All | userId (User Id)  Symbol (Stock Symbol)  date (transaction date and time)  buy (boolean buy or sell)  num(quantity of transaction)  total(total value of transaction stock worth) |

**Score Controller**

**/scores/leaderboards: Get top ten users by rank**

* Requests
  + GET(java.sql.date date)
    - Date format: YYYY.MM.DD
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: List<Scores>** |
| If less than 10 users have score | List of all scores ordered by rank |
| Else | List of top ten scores by ranks |

**/scores/getAllByDate: Get all scores for a certain date**

* Requests
  + GET(java.sql.date date)
    - Date format: YYYY.MM.DD
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: List<Scores>** |
| All | List of all scores ordered by rank |

**/scores/getAllByUser: Get all scores for a certain date**

* Requests
  + GET(int userId)
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: List<Scores>** |
| All | List of all scores ordered by rank |

**/scores/getFriends: Get all scores for friends for a certain date**

* Requests
  + GET(java.sql.date date, int userId)
    - Date format: YYYY.MM.DD
* Responses

|  |  |
| --- | --- |
| **Case** | **Return Value: List<Scores>** |
| All | List of all scores of friends with id=userId |

**Web Socket Endpoints**

**/livechat/{groupId}/{username}: Group Chats**

**/chat/{groupId}/{username}: Direct Message chat with friend**