

Chapter 1 Assignment Profile

1.1 Technology

1.2 Setup

1.3 Testing

1.1 Technology:

Front-End: HTML5, CSS3, Bootstrap, Angular9

Backend: NodeJS, MongoDB

1.2 Setup:

I have used latest version of UI framework angular and it requires angular CLI to run the project. Moreover, I have implemented backend logic as an API. Therefore, there will be two projects; one for UI and another for backend. The project setup details are as follow.

UI Project:

Name: CardGame-UI

Dependencies: Angular CLI, node

Commands:

1. **npm install -g @angular/cli** (Note: Install angular CLI)
2. **npm install** (Note: Go inside project directory and type this command; it will install required modules for project)
3. **ng serve --port 8081** (Note: This command will run project in localhost on port 8081, port specification is required because I have used **CORS** policy in backend API. So, it will not allow any request to be processed from another port except 8081)

Backend Project:

Name: CardGame-API

Dependencies: node, MongoDB

Commands:

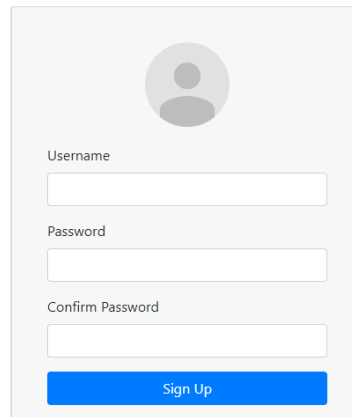
1. **npm install** (Note: Go inside project directory and type this command; it will install required modules for project)
2. **node server.js** (Note: This command will start the server on port 3000)

*Note:

Both projects are needed to be run in order to test the implemented functionalities as mentioned in the project documentation.

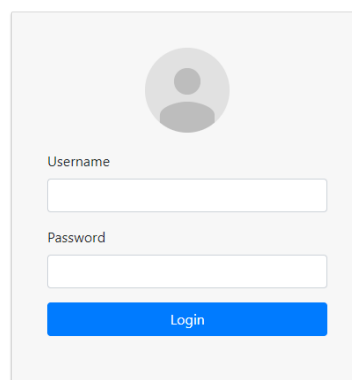
1.3 Testing:

Initially, users have to register on portal from signup form and user can go to that page by clicking on **SignUp** menu available in the top right corner of header.

A light grey rectangular form with a rounded top. At the top center is a circular placeholder for a profile picture. Below it are three input fields: 'Username', 'Password', and 'Confirm Password'. At the bottom is a blue button with the text 'Sign Up' in white.

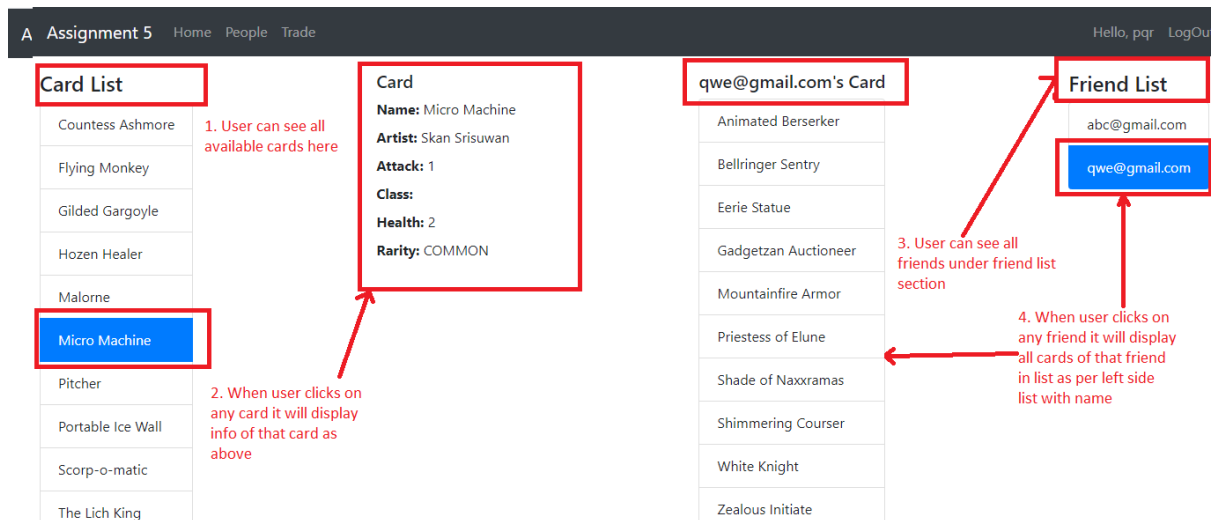
1.1 Registration Page

Next, on successful registration user will be redirected to **Login** screen with success message, users have to enter their credentials and they will be navigated to **Home** page if username and password are right, otherwise they will receive error message in top right corner of the browser.

A light grey rectangular form with a rounded top. At the top center is a circular placeholder for a profile picture. Below it are two input fields: 'Username' and 'Password'. At the bottom is a blue button with the text 'Login' in white.

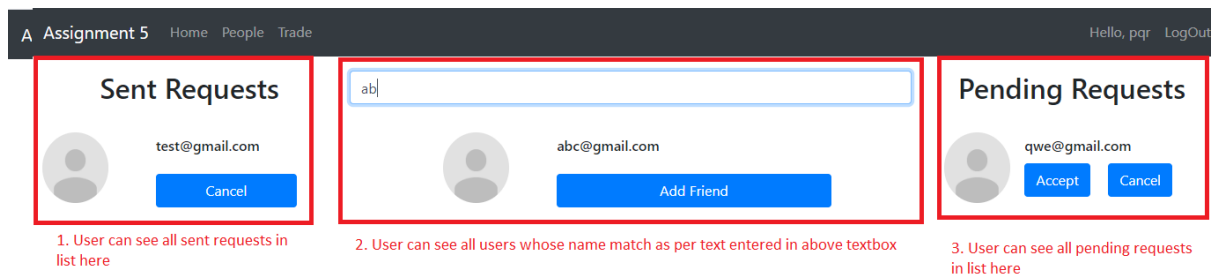
1.2 Login Page

At **Home** page, users can see their basic profile like available cards and those cards info as well as all their friend cards.



1.3 Home Page

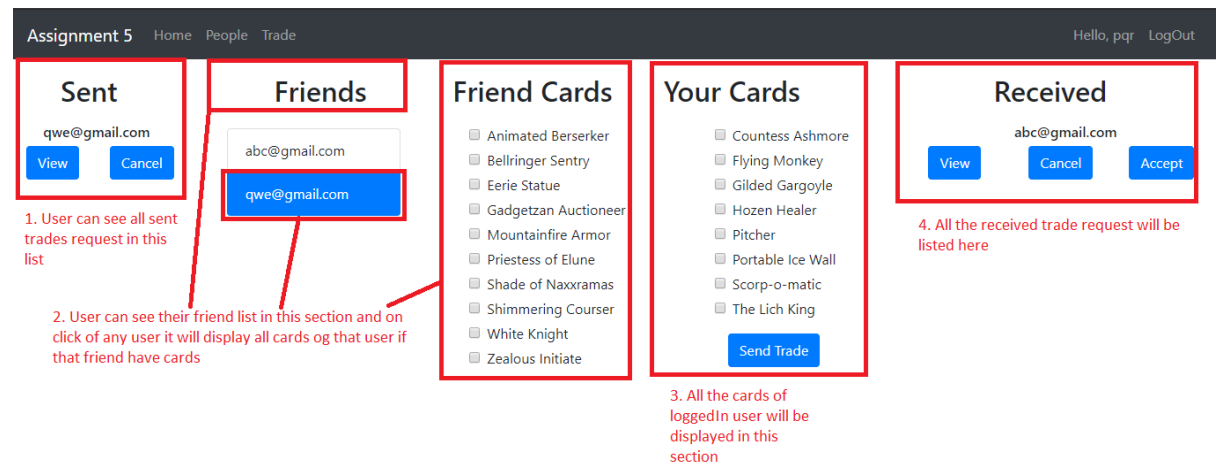
On the **People** window, users can do operations like search friend, accept or reject friend request and they can see sent requests and pending requests. In addition, user cannot send request to other people if they have already sent or received from those people and they can't send request to people, who are already their friends and they will see message in top right corner after each action.



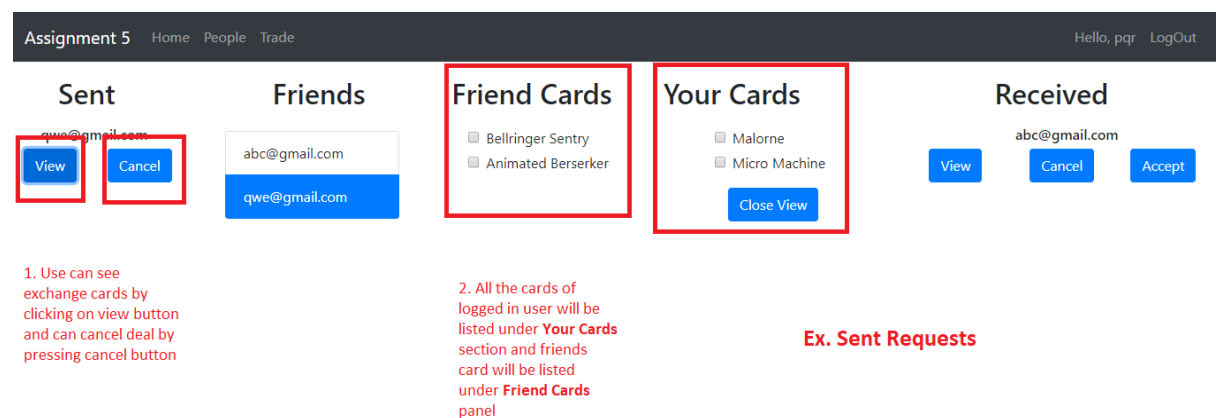
1.4 People Page

At **Trade** screen, user can see all the sent and received requests and also they can see all friends with their cards. Furthermore, user can send request using **Send Trade** button and they are able to view, cancel or accept the trade deal using available options.

***Note:** For the elimination of repeated card trading, I've used approach of remove the cards of user when they send trade request. Therefore, they will not be able to send trade request more than once for one card.



1.5 Trade Page (Default View)



1.6 Trade Page (Sent Requests View)

Assignment 5

Home

People

Trade

Hello, pqr

LogOut

Sent

qwe@gmail.com

View

Cancel

Friends

abc@gmail.com

qwe@gmail.com

Friend Cards

☐ Murloc Holmes

☐ Possessed Villager

☐ Scorp-o-matic

☐ Portable Ice Wall

Close View

Your Cards

Received

abc@gmail.com

View

Cancel

Accept

Ex. Received Requests

All the receivable cards will be listed under **friend cards** panel and request cards by friend will be shown under **Your cards** section and can close view using **Close View** button

User can see all the exchange cards by clicking on **View** button, cancel the deal using **Cancel** button and confirm the trade by **Accept** button

1.7 Trade Page (Received Requests View)

***Note:** All the data will be updated real-time as specified in document.

Chapter 2 System Profile

2.1 Design Decisions

2.2 Data Storage

2.3 Route Detail

2.1 Design Decisions:

From the provided document and video, I analyzed the requirements and functionalities of an application. Therefore, I came up with the decision of dividing UI in five pages; **register, login, home, people and trade**.

The main objective behind this bifurcation is providing particular facility from its belonging modules, for an example user can add friend from people page and can see sent and received requests.

1. Home:

This page provides functionality of check list of available cards and details of those cards. Also, it displays the information of friends of logged in user coupled with their cards.

2. People:

It provides facility of searching user using name and users can send request to people, whom with they want to be friends. In addition, they can see sent and received requests of people and they are able to confirm and accept request.

3. Trade:

It offers various functionalities like exchanging card with friends and viewing details of sent and received requests; if logged in user wants to accept or cancel selected request, they can do using available buttons.

4. Register:

This page provides facility of registration to any user, who wants to be part of card exchange game application.

5. Login:

Using this screen user can login to application and can do available operation of their choices.

*Note:

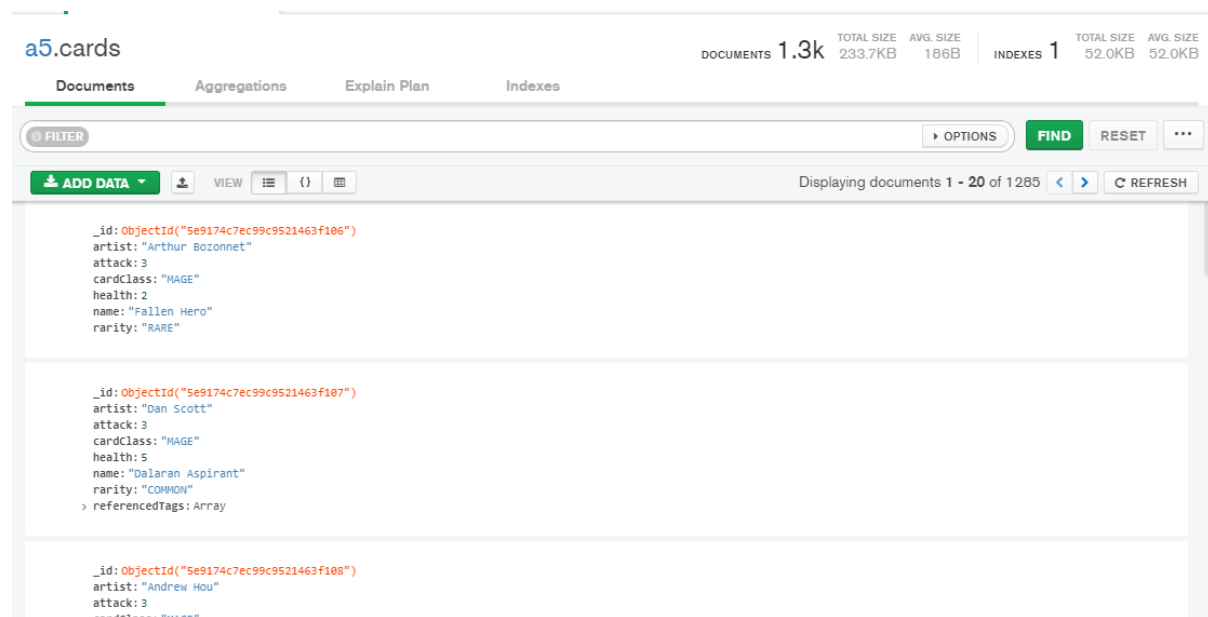
For restricting user from going back to previous page after logout, I've used concept of authentication and for that I have implemented **Json web token** in application. When user gets logged in application node API issues one token and sends in response and angular stores it in session storage of browser. So, if user is already logged in and tries to login again, they can't do it and can't go back to previous page after log out.

2.2 Data Storage:

As per the specification mentioned in assignment document, I have used mongoDB for data storage. I have created different collection for storing **user**, **friend request**, **trade request**, **friends** and **cards** detail using mongoose schema in nodejs. Moreover, I inserted all the data of cards from **databaseInitializer.js** file. Also, I have implemented concept of one to many relationship in collection for an example **user and cards**.

Firstly, when user gets registered on portal, his or her detail will be stored in **user** collection with 10 random cards, which have relationship of **one to many** with **cards** collection. Thereafter, whenever they send friend request then it will store **user** unique id in **friend requests** collection and it has also one to many and one to one relationship with **user** collection and after confirmation of request data will be stored in **friend** collection with **user Id** and **friend Id** information and contains relationship with **users**. Furthermore, when users perform any trade operation, the data will be stored in **trade requests** collection along with sender and receiver's id and their cards information and after accepting request their cards will be exchange or if user cancels trade, cards will be moved back to their **user** collection.

Following are the details of each collection, which I have used in my application.



2.1 Cards

Browser tabs: a5.users Documents

a5.users DOCUMENTS 9 TOTAL SIZE 2.4KB AVG. SIZE 273B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Documents Aggregations Explain Plan Indexes

FILTER OPTIONS FIND RESET ...

ADD DATA VIEW {}

Displaying documents 1 - 9 of 9 REFRESH

```
{
  "_id": ObjectId("5e96e21b6b870616d0c6584e"),
  "card": Array
    [0]: ObjectId("5e9174c7ec99c9521463f526")
    [1]: ObjectId("5e9174c7ec99c9521463f279")
    [2]: ObjectId("5e9174c7ec99c9521463f558")
    [3]: ObjectId("5e9174c7ec99c9521463f13e")
    [4]: ObjectId("5e9174c7ec99c9521463f395")
    [5]: ObjectId("5e9174c7ec99c9521463f228")
  },
  "username": "test@gmail.com",
  "password": "$2a$08$69bcIE9okwSehIn5yVVCjezcNSXOK1h6GN860zAA56SR3525J3UK6",
  "__v": 0
}
```

```
{
  "_id": ObjectId("5e96e22a6b870616d0c6584f"),
  "card": Array
    [0]: ObjectId("5e9174c7ec99c9521463f2f8")
    [1]: ObjectId("5e9174c7ec99c9521463f215")
    [2]: ObjectId("5e9174c7ec99c9521463f515")
    [3]: ObjectId("5e9174c7ec99c9521463f551")
    [4]: ObjectId("5e9174c7ec99c9521463f388")
    [5]: ObjectId("5e9174c7ec99c9521463f521")
  },
  "username": "abc@gmail.com",
  "password": "$2a$08$1h0TG3N1wH8UqInQ628tu1sENxf23TJDAlNCHh8ZxknZNKaGa3E6",
  "__v": 0
}
```

2.2 Users

Browser tabs: a5.friendrequests Documents

a5.friendrequests DOCUMENTS 3 TOTAL SIZE 222B AVG. SIZE 74B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Documents Aggregations Explain Plan Indexes

FILTER OPTIONS FIND RESET ...

ADD DATA VIEW {}

Displaying documents 1 - 3 of 3 REFRESH

```
{
  "_id": ObjectId("5e99d184175785335cd19eb9"),
  "people": Array
    [0]: ObjectId("5e99d177175785335cd19eb8")
  },
  "user": ObjectId("5e99d177175785335cd19eb8"),
  "__v": 0
}
```

```
{
  "_id": ObjectId("5e9aeaa434c5aa27d879fde8"),
  "people": Array
    [0]: ObjectId("5e96e21b6b870616d0c6584e")
  },
  "user": ObjectId("5e9ae2034c5aa27d879fde7"),
  "__v": 0
}
```

```
{
  "_id": ObjectId("5e9aef0634c5aa27d879fde9"),
  "people": Array
    [0]: ObjectId("5e99d177175785335cd19eb8")
  },
  "user": ObjectId("5e96e22a6b870616d0c6584f"),
  "__v": 0
}
```

2.3 FriendRequests

a5.friends Documents

a5.friends DOCUMENTS 6 TOTAL SIZE 504B AVG. SIZE 84B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Documents Aggregations Explain Plan Indexes

Displaying documents 1 - 6 of 6

```

{
  "_id": ObjectId("5e970c973f35434034fac74d"),
  "people": Array
    0: ObjectId("5e96e20d6b870616d0c6584d")
    1: ObjectId("5e96e21b6b870616d0c6584e")
    2: ObjectId("5e96e22a6b870616d0c6584f")
  "userId": ObjectId("5e96e21b6b870616d0c6584e")
  "__v": 0
}

```

```

{
  "_id": ObjectId("5e973b54f3ba7920401a9dcf"),
  "people": Array
    0: ObjectId("5e96e20d6b870616d0c6584d")
    1: ObjectId("5e96e21b6b870616d0c6584e")
    2: ObjectId("5e96e22a6b870616d0c6584f")
  "userId": ObjectId("5e96e21b6b870616d0c6584e")
  "__v": 0
}

```

```

{
  "_id": ObjectId("5e98b007657c1e30fc039f2b"),
  "people": Array
    0: ObjectId("5e98afea657c1e30fc039f29")
    1: ObjectId("5e99d177175785335cd19eb8")
    2: ObjectId("5e99d177175785335cd19eb8")
  "userId": ObjectId("5e99d177175785335cd19eb8")
  "__v": 0
}

```

2.4 Friends

a5.traderequests Documents

a5.traderequests DOCUMENTS 2 TOTAL SIZE 350B AVG. SIZE 175B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Documents Aggregations Explain Plan Indexes

Displaying documents 1 - 2 of 2

```

{
  "_id": ObjectId("5e9af68b34c5aa27d879fdeb"),
  "senderCards": Array
    0: ObjectId("5e9174c7ec99c9521463f3b7")
    1: ObjectId("5e9174c7ec99c9521463f3ea")
  "receiverCards": Array
    0: ObjectId("5e9174c7ec99c9521463f428")
    1: ObjectId("5e9174c7ec99c9521463f35e")
  "senderId": ObjectId("5e9aee2034c5aa27d879fde7")
  "receiverId": ObjectId("5e99d177175785335cd19eb8")
  "__v": 0
}

```

```

{
  "_id": ObjectId("5e9af7be34c5aa27d879fded"),
  "senderCards": Array
    0: ObjectId("5e9174c7ec99c9521463f397")
    1: ObjectId("5e9174c7ec99c9521463f568")
  "receiverCards": Array
    0: ObjectId("5e9174c7ec99c9521463f517")
    1: ObjectId("5e9174c7ec99c9521463f4c3")
  "senderId": ObjectId("5e9aee2034c5aa27d879fde7")
  "receiverId": ObjectId("5e99d177175785335cd19eb8")
  "__v": 0
}

```

2.5 TradeRequests

2.3 Route Detail:

I have developed two separate projects for implementing card exchanges game web application. Both projects have different routes and details for each are described below.

UI Application:

It contains total five pages. And those are **Register, Login, Home, People and Trade**.

1. <http://localhost:8081/register>
2. <http://localhost:8081/login>
3. <http://localhost:8081/home>
4. <http://localhost:8081/people>
5. <http://localhost:8081/trade>

***Note:** If user tries to enter any of the **URL**, he or she will be redirected to **login** page and if user is logged in then it will navigate to **home** page.

Backend Application:

It is an API and it consists of mainly three different routes. And those are **auth, user and trade**.

1. <http://localhost:3000/api/auth/signup>
2. <http://localhost:3000/api/auth/signin>
3. <http://localhost:3000/api/user/home>
4. <http://localhost:3000/api/user/addfriend>
5. <http://localhost:3000/api/user/searchuser>
6. <http://localhost:3000/api/user/acceptrequest>
7. <http://localhost:3000/api/user/cancelrequest>
8. <http://localhost:3000/api/user/getrequests>
9. <http://localhost:3000/api/auth/cancelpendingrequest>
10. <http://localhost:3000/api/trade/dashboard>
11. <http://localhost:3000/api/trade/send>
12. <http://localhost:3000/api/trade/cancel>
13. <http://localhost:3000/api/trade/accept>
14. <http://localhost:3000/api/trade/view>

***Note:** If user tries to enter any of the **URL**, he or she will get **cannot get {URL}** response.