Sprint 1 Week 1 Poker

Issues estimated
#23 – Implementing Card Create Endpoint
#24 – Implement Authentication Middleware
#25 – Implement Card Update Endpoint
#26 – Implement Card Delete Endpoint
#27 – Implement user home page endpoint
#28 – Implement Registration endpoint
#29 – Implement Login endpoint
#30 – Implement Image Upload

Abbreviations:
P: Parit
R: Richard
S: Shang Zhe
W: Walter
Y: Yujian

#23 -- R: 1 W: 2 P: 4 Y: 2 S: 3

Parit: When you put a card into the db, you would have to place it in a user.
Shang Zhe: The middleware will handle that.
Richard: The database will be encapsulate.
Walter: Cards go into the array held by the user.
Richard: Fetch user, append card, save the array.
Walter: The document will change because the array will change because the card will change.

ShangZhe: Need to deal with plumbing, carefully handling the modification of the data. Even as the driver will handle some of the checking, the server will need to validate the card input. Will need careful validation on data coming in. Validation features on Mongoose may not be sufficient.

Yujian: It may need more finess.

Richard: I have experience implementing it at have a well thought route on how to implement it.

#23 – R: 1 S: 3 P: 3 W: 3 Y: 2 → size 3

#24 – W: 6 P: 5 Y: 7 S: 2 R: 9

Richard: Shang Zhe wrote 100 lines last semester for this purpose.
Walter: These 100 lines comes with testing, research and typing.
Shang Zhe: A majority of them are also comments… and the code from last semester was an abstraction for Vendors and stuff.

Yujian: I am unsure about how that works. If I were to do it I would have to do a lot of research for it.

Shang Zhe: see Richard’s

Parit: As far as I know, the middleware handles the login request on the user and login will pass an enc cookie__

Shang Zhe: This middleware will handle cookies from the client that it is valid and checks with the database for authentication.

Parit: Then the effort falls onto the enc-dec, expiration date will be stored in the cookie setings.

Shang Zhe:

Walter: Independent coding and research will take a while. I would be using libraries.

Shang Zhe: Using other libraries would increase the time commitment and size

Richard: You don't know the size hence you're rating may not be clear.

#24 – Y: 4 R: 3 W: 6 P: 5 S: 2

Parit: I don't know any enc-dec library. What will be used to encrypt the data?

Shang Zhe: Walter knows of crypt that can be used.

Walter: What about subtleCytpo.

Shang Zhe: CryptoLibrary in node.js can be used, Bcrypt is for hashing.

Yujian: [ditto] but given that Shang Zhe has shown this can be done easily.

Shang Zhe: Can be done tomorrow.

#24 – R: 3, W: 3, P: 5, S: 3, Y: 4 → size 3

#25 – R: 3, W: 3, P: 4, Y: 3, S: 3 → size 3

mimic of #23

#26 – S: 1, R: 3, Y: 1, P: 3, W: 3

Parit: If you can do the CREATE, EDIT, and DELETE should be identical. Given that everything is authenticated. Give it the same point as PATCH and POST. Combining all three would give the CRUD an estimated size 4 in order to port sizes.

Richard: Even though we require less lines of code that UPDATE and POST. It's fundamentally the same thing.

Walter: Fetch the user, access card array – filter the card array on card Id. And then save.

Shange Zhe: Hard part that gave 3 to PATCH and POST was because it involved validation, to check for weird sequences of characters in fields. For delete, authentication and checking that the cards exists is required. The difficulty of PATCH and POST is gone.

Yujian: Ditto Shange Zhe. Get the correct user and the correct card.

Shange Zhe: By the point the endpoint is reached, the user would have been found. Middleware will handle authentication.

#26 – W: 2, S: 1, Y: 2, R: 2, P: 2 → size 2

#27 – S: 4, W: 1, R: 3, P: 1, Y: 2

Shang Zhe: Getting the information from the db is easy. Hard part is distilling the data so that none of the databases internals is exposed. Need to erase ObjectIds, turn into flat string. Some fields require transformation, considered carefully to prevent exposure.

Richard: I don't think the data can be perfectly passed on to the data. The data will not be what required for the client. Need to filter and transform data.

Yujian: Same idea as Richard, but not as elaborate.

Walter: Just realised we need to populate the cards with the tags.
    Parit: The card entity will store the tag id and the front end will render it.
    Shang Zhe: The cards array can be sent back as is and once the object Id's have been flattened then the id's can be used to refer tags.

Parit: You can send the whole thing in one. But if security is a concern and you'll need to translate everything.

#27 – R: 3, S: 3, Y: 3, P: 3, W: 3 → size 3

28 – P: 1, R: 4, S: 2, Y: 1, W: 1

Richard: I thought we had to set up the default things for the user – to populate the default fields for the user.
    Parit: That has been set up as the user schema.
    Shang Zhe: Yeah.

Shang Zhe: A little bit of validation is involved to prevent strange characters in the username and that the username has not be used in the database. I wrote in the Api spec that the error status is returned if the username is already in use.

Yujian: I thought it was as simple as making a new document.

Walter: Ditto

Parit: Ditto

28 – W: 2, Y: 2, S: 2, P: 2, R: 2 → size 2

Parit: What about the middleware. Does it do it?
    Shang Zhe: Middleware will be used authenticate user, but this is used to make that user.

29 – W: 3 S: 2, R: 4, P: 5, Y: 2

Parit: You gotta check the hash of the password and if it's okay you create the token and send to the user. That's more processes than more issues faced so far.
    Walter: Check the hash?
    Parit: Check the password that it is correct?
    ShangZhe: Do a match check in the database.
    Yujian: Is there more to it?
    ShangZhe: Strcmp will suffice

Richard: You have to generate the token with the expiration. That could be a bit difficult. You need to get the user id (flattened), expiration, and encrypt it with the secret salt. But it could also be straightforward. Has similarity with #24, but because in endpoint needs a little more steps.

Walter: Could put expiration on cookie header.

Shang Zhe: The expiration date is to be encrypted into the body of the cookie and an expiration date on the cookie (visible on the client).

Walter: I was fixated on the cookies. Gotta create and send it back, with the required details.

Yujian: I acknowledge the cookies, but not in mind with the encrypted.

Shang Zhe: 3 steps. Find username, match password, fill out the JS Object with the database ID, expiration date, run over symmetric encryption and send back with expiration date. Need a key that is kept as env.

Richard: Is there a size limit to cookies?

Shang Zhe: Likely but not likely to be hit.

Parit: If you fail you have to send something back?

Shang Zhe: 401 response

29 – W: 3, P: 5, R: 3, S: 2, Y: 3

Parit: If you give other functionality 3, this one has a little more process, hence 5.

Yujian: A lot of concerns enc and dec. If can be done as a three step process, then the size would be smaller.

Richard: Size is comparable with creating a card. The process is rather straightforward.

Walter: Size is comparable to other issues of size 3.

Shang Zhe: It's still that very clear 3 step process.

→ Size 3

Decided that server is to have size limit, 1MB image field. Front end will offer resize capabilities for this purpose.

30 – Y: 9, S: 6, W: 6, P: 8, R: 10

Richard: That's a lot of research and how to use libraries. We gave create card a 3 and this is a lot harder than creating card. Taking unfamiliar territory.

Yujian: Same. This is about x3 the work of card creation.

Parit: There is a lot of research to be done. And changes needed to be made on api and schema to create image uploading function.

Walter: Images can be stored in another collection. For POST, create a new document and save the id and link to the card. Get you be sent the image directly from the database.

> Shang Zhe: It would suffice to store the image to the card it belongs, that is embedded into the users themselves.

Shang Zhe: Image can be stored in the Card collection. This issue concerns the client and server side, larger scope than most issues. Image compression resize is a front-end responsibility and the back-end will need to check that the image is sanitised, manually since Mongoose is ill-equipped for that security. Send bad req for miss-sized and not image.

30 – W: 7, P: 9, S: 5, Y: 8, R: 9

Parit: Bumped up to nine. Since you have to sanitise before to store. Additional worry.

Richard: This is as big as create card. It’s bigger than create card x3.

Yujian: Dropped to 8, clarification on the backend and image upload. Different from card API. Thought this is purely how to get pictures to and from and not integrated with API for cards.

Walter: Information can be sent through the body.

> Shang Zhe: Can encode cardId in URL and can use body for data.

Shang Zhe: JIMP will be able to sanitise it for us. Open the image and save.

> Walter: Will we need encoding on the image?
>
> Shang Zhe: As long as we don’t use JSON body we can send binary data. Encoding comes form the header content-type. MIME package to parse content to ordered type. Two rounds of checking, check entire request body for size capacity. JIMP will do reencoding and see if still fits. JIMP can be used for resizing.

30 – W: 6, S: 6, P: 8, Y: 8, R: 9 → size 7