**FIDO Server (Existing Functionalities):**

I have below APIs working on the FIDO server:

**Note:** Currently Registration API and Authentication API of EBAY works as it is without any changes with an IOS device request with ECDSA Keys. Only Database code *(StorageImpl.java file updated)* is added for these two APIs to support a MySQL server Database for storage and retrieval.

**A] Login Request API**

1) **API**: v1/public/loginRequest

2) **Method**: POST

3) **Input JSON:**

{

"username": "<phone number of user in E.164 format. Ex. +919134678956>"

}

4) **Output JSON:**

{

"registrationResponse": "<OTP generated>"

}

5) **What does the API do?**

* Gets username (i.e. phone number) passed in the input JSON.
* Generates OTP and sends SMS on that phone number.
* Captures the date and time when the OTP was generated.
* Stores details in registration table (*Table name:* *registrationdb).* Detailslike: username(i.e. phone number), OTP and OTP\_creationDate.
* In registration table, checks if username already exists and returns the stored OTP for the user in the API response. Else, it returns the newly generated OTP, if the user is not existing in table.
* Finally as a response to API, it returns the OTP for the user (either stored OTP if the user is already present in DB, else the newly generated OTP if user is completely new).

**B] Login Response API**

1) **API**: v1/public/loginResponse

2) **Method**: POST

3) **Input JSON:**

{

"username": "<phone number in E.164 format. Ex. +919134678956>",

"otp": "<previously generate OTP sent to user>"

}

4) **Output JSON:**

{

"Login Status": true/false,

"Registration status": true/false

}

5) **What does the API do?**

* Gets the username and OTP passed in the input JSON.
* Gets OTP and OTP\_creationDate stored for the username from registration table.
* Gets the current system time.
* Checks if present time – otp creation time > 3 minutes, and if yes, it prints a message as “OTP Expired”
* If present time – otp creation time < 3 minutes, updates login status(*column name: loginstats in registrationdb table*) to true for the user in registration table.
* Finally as a response to API, returns Login status either true or false and also registration status, which will be false for now as user has not yet registered, but just logged in.

**C] New Vendor Request API**

**Note:** Vendor here implies to Relying Party website.

1) **API**: v1/public/newVendorRequest

2) **Method**: POST

3) **Input JSON:**

{

"vendorName": "<Name of Relying Party>",

"username": "<Name of the user>",

"phoneNumber": "<phone number of user registered during login (i.e. username in registrationdb table>",

"dateTime": "<date-time details in string>"

}

4) **Output JSON:**

{

"registrationResponse": "<OTP generated>”

}

5) **What does the API do?**

* Gets the vendor name, username, phone number and dateTime details passed in the input JSON.
* Generates OTP and sends SMS on that user’s phone number.
* Captures the date and time when the OTP was generated.
* Stored all details: vendorName, username, phoneNumber, dateTime, OTP and OTP\_creationDate in vendor table (*table name: vendordb*)
* In vendor table, check if same username with same vendor name already exists, and if yes, just prints a message as “*User already present with same vendor in Database!!!”*. Basically same user with same vendor is not allowed to be registered twice. But same user is allowed to be registered with a different vendor name.
* Finally as a response to API, returns the OTP generated.

**D] New Vendor Response API**

1) **API**: v1/public/newVendorResponse

2) **Method**: POST

3) **Input JSON:**

{

"vendorName": "<name of vendor sent previously in vendor request API>",

"username": "<name of user sent previously in vendor request API>",

"otp": "<OTP previously generated during vendor request API>"

}

4) **Output JSON:**

{

"Vendor Name": "<Name of Vendor registered>",

"Registration Status": true/false

}

5) **What does the API do?**

* Gets the username, vendor name and OTP passed in the input JSON.
* Gets OTP and OTP\_creationDate stored for the username and vendor name from vendor table.
* Gets the current system time.
* Checks if present time – otp creation time > 3 minutes, and if yes, it prints a message as “OTP Expired”
* If present time – otp creation time < 3 minutes, updates vendor registration status(*column name: vendor\_regstats in vendordb table*) to true for the given username and vendor name in vendor table.
* Gets phone number for given username and vendor name from vendordb table.
* Updates the vendor name in registration table (*vendordetails column in* *registrationdb table*) using the phone number as username, which was fetched in above the step.

**Note:** username in registration table is same as phoneNumber id vendor table, i.e.

**FOREIGN KEY(phoneNumber) from vendordb = PRIMARY KEY(username) from registrationdb**

* Finally as a response to API, returns the vendor name and the vendor registration status either as true or false, based on success of vendor registration operation.

**E] FIDO Server Request API**

1) **API**: v1/public/fidoServerRequest

2) **Method**: POST

3) **Input JSON:**

{

"vendorName": "<vendor name already registered before>",

"phoneNumber": "<phone number associated with the registered vendor name>"

}

4) **Output JSON:**

{

"Vendor Name": "<Name of Vendor registered>",

"Registration Status": true/false

}

5) **What does the API do?**

* Gets the vendor name and phone number passed in the input JSON.
* Checks if vendor registration status is true for the given vendor name and phone number in vendor table *(column vendor\_regstats from vendordb table)*
* If Vendor is registered, sends a push notification to IPhone device with a message as “*PSL FIDO server requesting you to Authenticate: <vendor name>!!!*”. Else just prints a message as *“The given vendor is not registered to use FIDO services”.*
* Finally as a response to API, returns the Vendor name and vendor registration status.

**Database details:**

**Database Name:** fidodatabase

**Tables:**

1. registrationdb ----- Registration details table.
2. vendordb ----- Vendor details table.

**Table descriptions:**

1. registrationdb table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| username | varchar(20) | No | PRI | NULL | - |
| regstats | tinyint(1) | Yes | - | NULL | - |
| publickey | longtext | Yes | - | NULL | - |
| otp | varchar(10) | Yes | - | NULL | - |
| loginstats | tinyint(1) | Yes | - | NULL | - |
| vendordetails | longtext | Yes | - | NULL | - |
| aaid\_keyid | longtext | Yes | - | NULL | - |
| deviceid | varchar(50) | Yes | - | NULL | - |
| otp\_creationdate | datetime | Yes | - | NULL | - |

1. vendordb table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| username | varchar(20) | No | - | NULL | - |
| phonenumber | varchar(20) | No | MUL (FK) | NULL | - |
| vendor | varchar(50) | Yes | - | NULL | - |
| datetime | varchar(20) | Yes | - | NULL | - |
| otp | varchar(10) | Yes | - | NULL | - |
| vendor\_regstats | tinyint(1) | Yes | - | NULL | - |
| otp\_creationdate | datetime | Yes | - | NULL | - |

**NOTE**: phonenumber in vendordb = username in registrationdb.