**Database Schemas**

1. New DB tables required for Enterprise Policy Management Website.

We have decided it together to keep the UI module (design) and the backend module (main business logic) separate for future convenience. As of now, I’m thinking of creating this website using HTML5 and CSS3 (node.js as frontend) with JAVA as the backend. So I will need to create a different Database say [“**enterprisePolicyDb**”] which will store below data in a table name as “**userDetails**”.

1. **Username** [Type: String] ---- PRIMARY KEY
2. **Password** [Type: String]
3. **Role** [Type: String]
4. **Grade** [Type: Float]
5. **Employee** **ID** (optional) [Type: Long]

The **Role** field will have only two values: “**User**” for normal employee and “**Admin**” for system admin. The values for all user types will be pre-populated in the DB, which will be fetched as and when required for validations and other stuff. **Grade** will be stored with values as: **3.x, 5.x, 7.x, 9.x and 13.x.**

**B.** To store the authenticators allowed for each employee grade group, a new table will be created in this same DB with name as “**gradePolicies”** with following details stored in it:

1) **Grade** [Type: String] ---- PRIMARY KEY

2) **AllowedAuthenticators** [Type: String []] - Should be exact names mention in **authenticators.Name** field**.**

The **Grade** field again will store values as: **3.x, 5.x, 7.x, 9.x and 13.x** only**.** The **AllowedAuthenticators** field should contain list of names that are exactly same as the name field in **authenticators.Name** field for each authenticator.

**C**. To store the authenticators allowed to be used (from the list of allowed set of authenticators for each grade) for each app provided by the organization, a new table will be created in the same DB with name as “**appPolicies**” with following details stored in it:

1) **Grade** [Type: String] ----- FOREIGN KEY to **gradePolicies.Grade**

2) **App** **Name** [Type: String]

3) **AuthenticatorsSupported** [Type: String []] - Should be exact names mention in **authenticators.Name** field**.**

The **Grade** field again will store values as: **3.x, 5.x, 7.x, 9.x and 13.x** only**.** The **AuthenticatorsSupported** field should contain list of names that are exactly same as the name field in **authenticators.Name** field for each authenticator.

**D**. To get the AAIDs, KeyIDs and other information related to a particular authenticator, I’m thinking of storing a pre-populated DB for the same on the FIDO server side DB [“**fidodatabase**”] with name as, “**authenticators**”, having below details (all these fields are as per FIDO protocol document):

1. **Name** [Type: String] ---- PRIMARY KEY
2. **AAID** [Type: String [] ]
3. **Vendor** **ID** [Type: String [] ]
4. **Key** **ID** [Type: String [] ]
5. **User** **Verification** [Type: Long]
6. **Key** **Protection** [Type: Integer]
7. **Matcher** **Protection** [Type: Integer]
8. **Attachment** **Hint** [Type: Long]
9. **Tc** **Display** [Type: Integer]
10. **Authentication** **Algorithms** [Type: Integer [] ]
11. **Assertion** **Schemes** [Type: String [] ]
12. **Attestation** **Types** [Type: Integer [] ]
13. **Authenticator** **Version** [Type: Integer]
14. **Extensions ------ >**
15. **ID** [Type: String]
16. **Data** [Type: String]
17. **Fail**\_**If**\_**Unknown** [Type: Boolean]

We will need to decide what data to set in all the above fields for authenticators we are supporting currently (pin-code and fingerprint), and for the new authenticators we will develop in future.

**E**. Next, we need to think how to link the device details with this user policy information. Currently we don’t save any field for user’s name or phone number in the “**devicedetails**” table in DB. Below are the current details stored in “devicedetails” table:

1. **deviceid** [Type: String] --- PRIMARY KEY
2. **devicetoken** [Type: String]
3. **rpaccountname** [Type: String]
4. **regstats** [Type: Boolean]
5. **publickey** [Type: String]
6. **aaid**\_**keyid** [Type: String]
7. **rpaccountid** [Type: String] -- FOREIGN KEY to [**vendordb. accountid**]

On install of the app, when user opens the app the first time, he accepts the push notifications feature to be enabled for that app to receive notifications from server in future. During this call, we fire an API, which sends the server these two fields: **deviceid** and **devicetoken**. At this time, the user doesn’t have a username or phonenumber stored on app side and sent across to the server. We will need to maybe add an extra field “username” in here, send the value of it to server as part of input JSON of this API, and the server can store it in the above DB table with a field name as “**username**”. Also once the policy is created, we will need it while making the regRequest call. So we need to store the policy too in this table with name as “policy”

So new field that can be added from server side is:

1. **username** [Type: String] FOREIGN KEY to [**vendordb. email**]
2. **policy** [Type: String]

**F.** The device details/user details will be linked with the RP website details with the **deviceid** as main key; just what we used to do before. This flow will not be changed. The RP website details are stored in a table named “**vendordb**” which has below fields:

1. **accountid** [Type: String] --- PRIMARY KEY
2. **displayname** [Type: String]
3. **rpdisplayname** [Type: String]
4. **email** [Type: String]

***NOTE: [We should keep this “email” same as “username” in the above devicedetails table].***

1. **otp** [Type: String]
2. **vendor**\_**regstats** [Type: Boolean]
3. **otp**\_**creationdate** [Type: DateTime]

**G**. The “**transactiondb**” table will also won’t change and will work as it used to work like before.

1. accountid [Type: String] --- PRIMARY KEY, FOREIGN KEY to [**devicedetails. rpaccountid**]
2. contents [Type: String]
3. appid [Type: String]

So To summarize in terms of a pictorial diagrams,

FIDO Database

|  |
| --- |
| **authenticators** |
| name **\*** |
| aaid |
| vendorId |
| keyId |
| userVerification |
| keyProtection |
| matcherProtection |
| attachmentHint |
| tcDisplay |
| authnAlgorithm |
| assertnSchemes |
| attestationTypes |
| authVersion |
| extensions |

|  |
| --- |
| **devicedetails** |
| deviceid **\*** |
| devicetoken |
| rpaccountname |
| regstats |
| publickey |
| aaid\_keyid |
| rpaccountid **#** |
| username |
| policy |

|  |
| --- |
| **vendordb** |
| accountid **\*** |
| displayname |
| rpdisplayname |
| email |
| otp |
| vendor\_regstats |
| otp\_creationdate |

|  |
| --- |
| **transactiondb** |
| accountid **#** |
| contents |
| appid |

**FIDO** **Server** **Database**.

**Enterprise** **Policy** **Management** **Website** **Database**:

Enterprise Policy DB

|  |
| --- |
| **gradePolicies** |
| grade **\*** |
| allowedAuthenticators |

|  |
| --- |
| **appPolicies** |
| grade **#** |
| appName |
| authenticatorsSupported |

|  |
| --- |
| **userdetails** |
| username **\*** |
| password |
| role |
| grade |
| employeeId |

PRIMARY KEY (**\***)

FOREIGN KEY (**#**)

Note:

* **devicedetails.username** will be same as **userdetails.username**.
* **userdetail.grade,** **gradePolicies.Grade** and **appPolicies.Grade** all should have same values (3.x, 5.x, 7.x, 9.x and 13.x).
* **gradePolicies.allowedAuthenticators []** and **appPolicies.authenticatorsSupported []** should contain names of authenticators which should be exact same to the names in **authenticators.name** field for each authenticator.
* **vendor.email** will be same as **devicedetails.username** and **userdetails.username**. (For now thinking of keeping username as his unique email address. Haven’t kept any strict constraint here in MYSQL DB, but it will be better if all these 3 fields are same for fetching data from respective tables and databases in server).