**#**There are two API. One is e\_Wallet and other is Bank. Both of them are fully functional. As per we need an Interface to show them working we used our own RDS system. In reality it can be attached with any organization or companies own website. Now a days everyone (every company) has there on account system. e\_Wallet API is capable of working with any of those system.

For payment, obviously we need a Banking system, so we made a bank API. We preserved the actual data integrity. We talked to some software engineers who are responsible for maintaining their banking system. We took some suggestions from them to make our Bank API. We developed only those parts, that will be needed for us.

We made our API such a way, SQL injection cannot be injected to it. Also, it can detect malicious invocation of API. At that case, it will reject that request.

For every api call we checked every single credential for validation. For this, we used two class “PureData” and “Tools”, which is implemented on a file called “Nirapotta” (a Bengali word).

* For e\_Wallet API

1. **Createacc:** This is responsible for creating new e\_wallet account. At the very beginning no user will have any e\_Wallet account. From RDS they will create their account. This will call Bank API’s “Create” api to make user entry to that Bank.
2. **ChangePin:** This is responsible for changing users e\_Wallet pin if needed. Given during creation of account.
3. **Switchtransaction:** This API is responsible for changing user’s RFID card status. They can on/off their cards transaction using this one.
4. **Balance:** Using this one user can check their balance. This api will call Bank API, then Bank API will return balance for that particular account.
5. **ViewAllHistory:** This API is responsible for getting all transaction history for a particular account. It will call Bank API and get result from it.
6. **SingleHistory:** Using this API user can see details for a single transaction.
7. **Userlist:** Using this API, list of valid user’s will be invoked. Any user, who turned off his/her transaction, will not be included to that list.
8. **Purchase:** This API is responsible for handling any transaction request.

\*\*API 1,2,3,4,5 and 6 will be invoked from user account (for our case RDS). No matter which interface.

\*\*API 7 and 8 will be invoked from merchant site. Bookshop, café, Gym, Parking etc. (for our kase Kasundi vendor machine). Again, no matter which interface.

\*\*API 1,4 ,5 and 8 will call Bank API as they needed.

* For Bank API

1. **Create:** For making a new user entry to banking system it will be used.
2. **Load:** This api is responsible for load balance to a particular account.
3. **Balance:** This api will return particular user’s balance.
4. **History:** While invoked, this api will return full list of transaction history of a user.
5. **Transaction:** This will be used for making transaction.

\*\* In reality Bank API has many more functionality, for our case we only need those functionalities. So we only implemented this five.

\*\*Our transaction mechanism is implemented exactly as a real Banking system does.