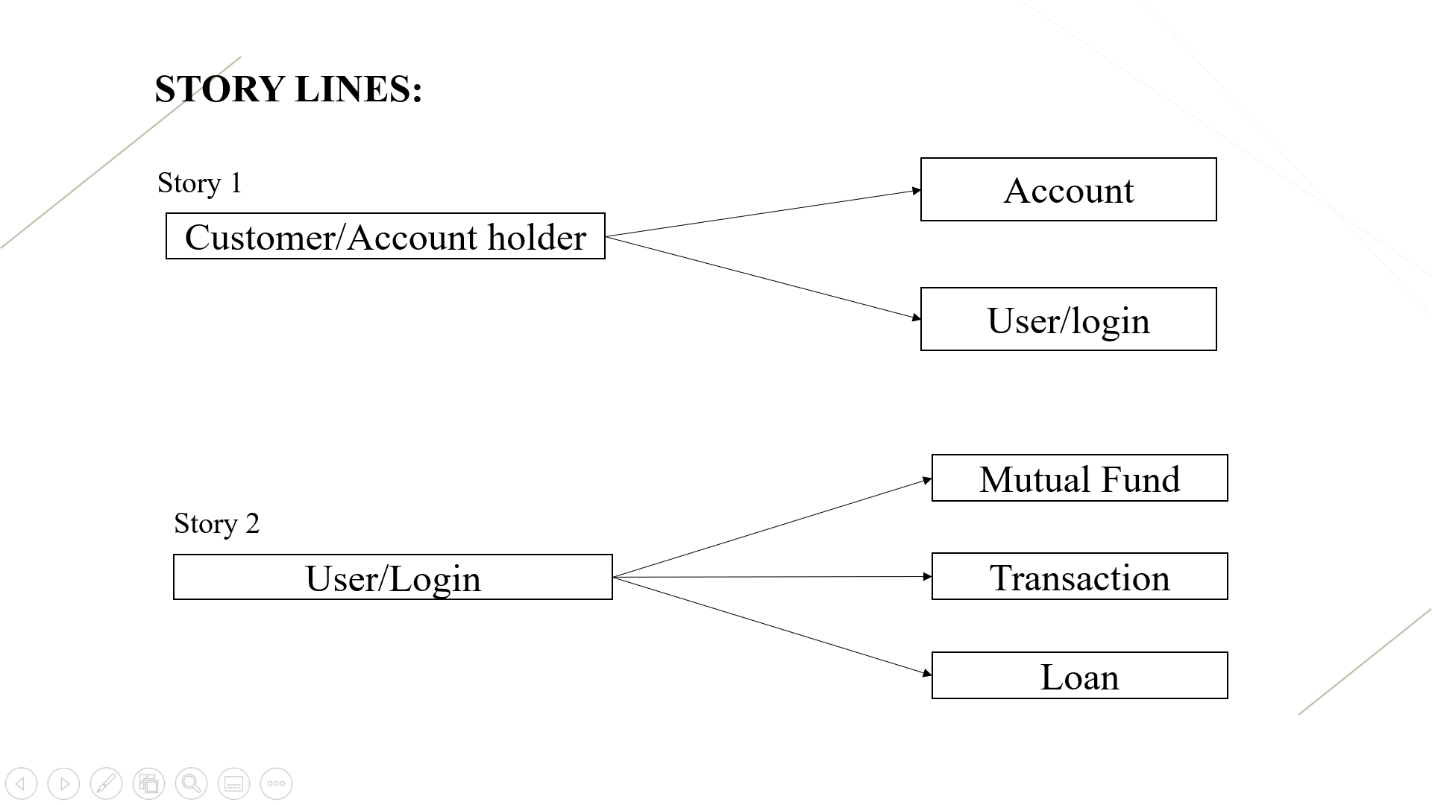
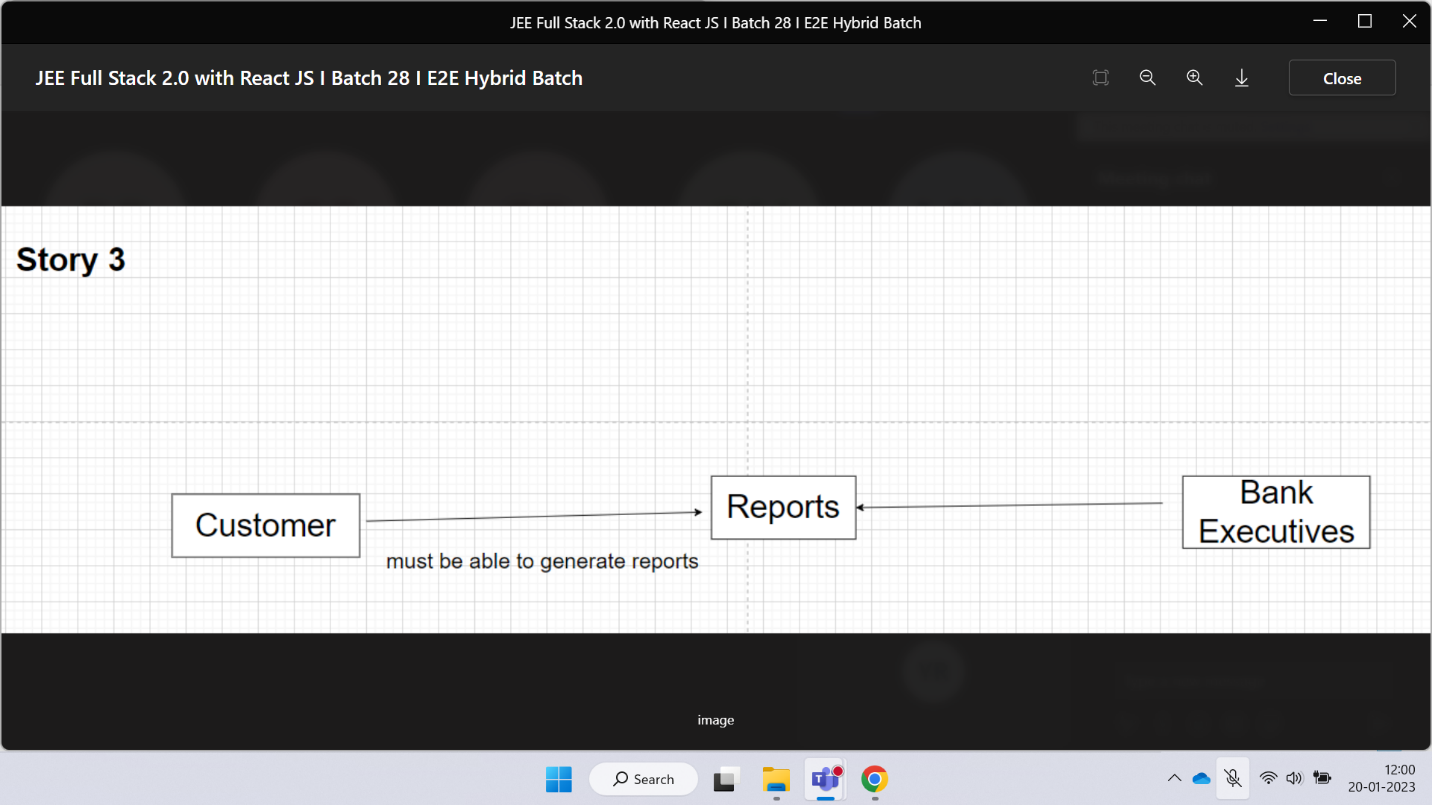
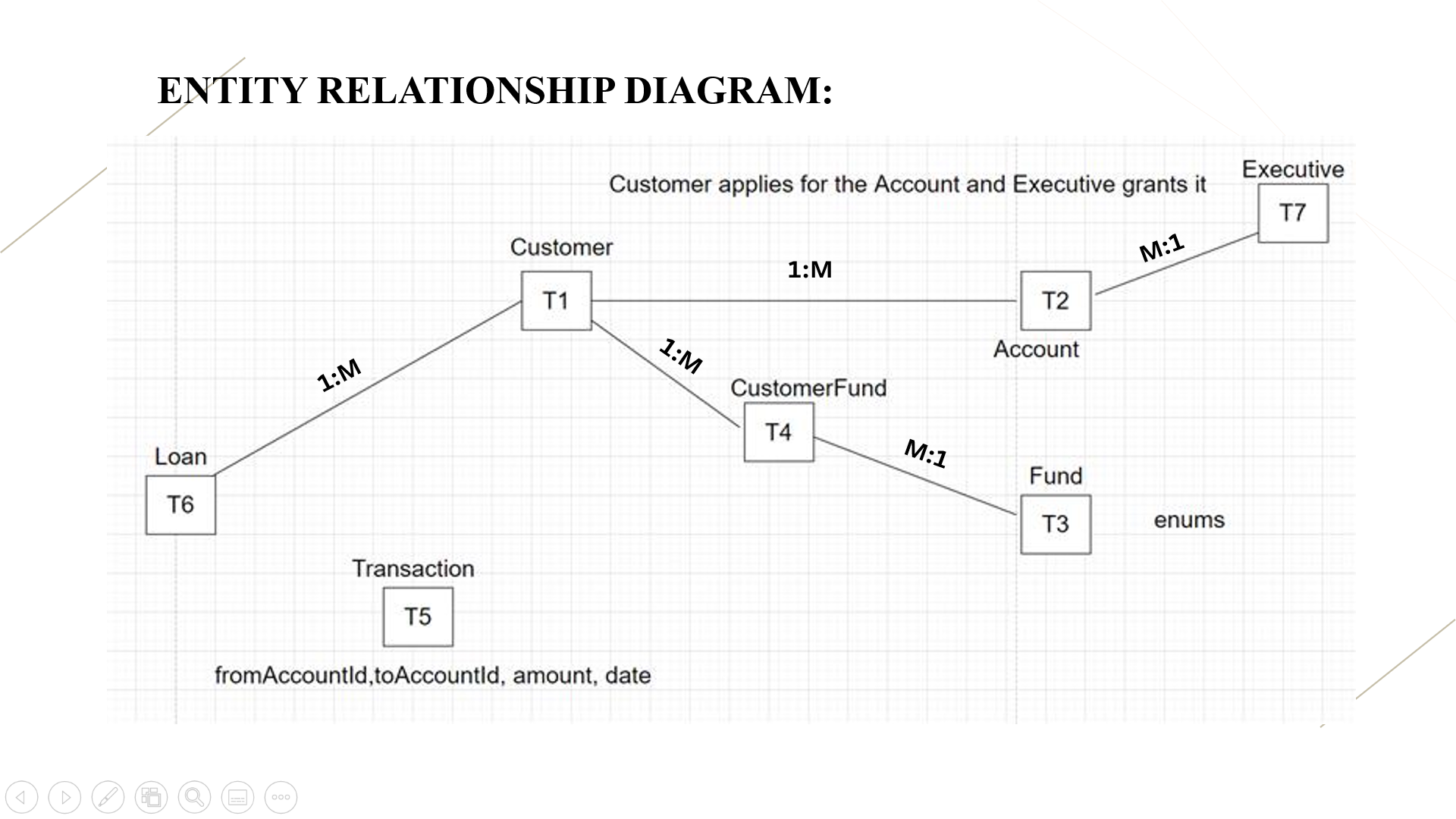
Online Banking System







* Customer [1:M] 1:M [1:1]Account : Inject Customer in Account
* Customer [1:1] 1:1 [1:1]User : Inject User in Customer
* Customer [1:M] M:M [M:1]Fund: Create a Separate class CustomerFund

relationship: CustomerFund

* If we do not want any extra columns in relationship table, we can go for @ManyToMany
* If we want extra columns in relationship table, then we must create a Relationship entity and inject the objects of Both the classes in it.

* Loan [1:1] M:1 [1:M] Customer : Inject Customer in Loan entity : @ManyToOne
* Account [1:1] M:1 [M:1] Executive: Inject Executive in Account: @ManyToOne

**APIs:**

**Customer:** POST, GETAll, GetById, PUT, DELETE

**Loan:** POST, GETAll, GetById, PUT

**Executive:** POST, GETAll, GetById

**Account:** POST[customerID,executiveId], PUT, DELETE

**Fund:** POST, PUT, GETAll, GetById, DELETE

**CustomerFund:** POST [customerId,fundId]

**Transaction:** POST

**Business API:**

* GetCustomerByLoan
* GetAccountByExecutive
* GetFundByCustomer

**Software Used:**

* Maven
* Spring tool suite 4
* Postman
* React js
* Node js
* HTML, CSS, Java Script
* VS code
* Redux
* React Thunk

**Functional Requirements:**

This software will have following functionalities

**Online balance check and transaction information:**

Customer will be able to check his balance online while sitting at home by accessing the database of the bank using his/her password and account no. allotted him by the bank.

**Balance transfer:**

This system will provide a path to the customer of the bank to transfer his balance to other account in easy steps. A small transfer fee will be applicable for this transaction.

**User Characteristics:**

There are various kinds of users for the product. Usually web products are visited by various users for different reasons.

The users include :

1. Chancellor who will be acting as the controller and he will have all the privileges of administrator.
2. All the persons who needs to perform banking.

**Generals Constraints:**

Some general constraints should be defined which will have a great part in the overall succession of the online banking project.

**Safety and Security:**

This Project must be safe and secure because customers will directly contact their account through the internet. Software will have to identify the valid customer according to his/her bank details and password. So it is a difficult task to prevent the system by major disasters by preventing the unauthorized access to the system.

**Assumptions and Dependencies:**

Following are the assumptions and dependencies which are related to this online banking project.

This project is a stand-alone project so it will not affect the system where it will be embedded.

This project is a web-based project while the staff was addict of using traditional methods of data storage and retrieval so they will be trained a bit to jump to it.

This system will not depend on any other module. It will be a web-based so everyone will independently contact it.

It is will not affect the environment at all.

Banks will feel free to adopt it because it will not be so much expensive.

As this project contains valuable and new features so it will probably remove the previous online banking systems embedded in some banks.

**Specific Requirements:**

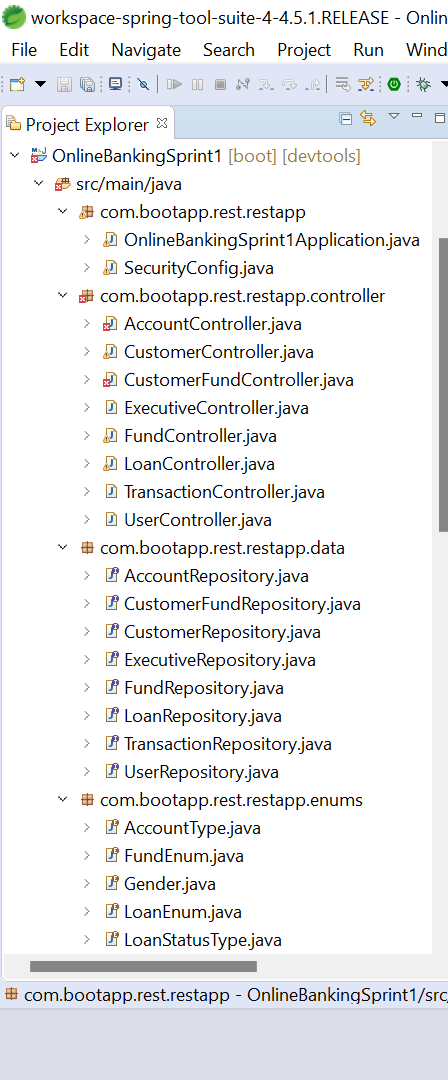
How the online banking will interact with the environment, what will be the functional and non-functional requirement. These all the steps should be defined here for providing a powerful base to the design phase. The design of the project will completely depend on the functional and non-functional requirements. So these should be defined clearly and accurately for the effectiveness.

**Functional Requirements:**

Following are the services which this system will provide. These are the facilities and functions required by the customer.

* Online balance check.
* Balance transfer(credit/debit)
* Loans
* Mutual fund/sip
* Transactions

**Package Structure:**

**=**

