**Group Project Vehicle loan Application**

1. Application will be developed in following -
2. Spring Boot + Rest Controller + JPA with Hibernate
3. Design
4. Generate the skeleton as per class design provided

5. Implement the case study in incremental manner

**This project has all the necessary functionality that a Loan Management application should have. It provides support to all roles. The whole project is design using springboot.In the Backend, it is using Spring Boot, at the data access layer the project is using JPA(Hibernate) Framework.**

As we know, the requirement for taking loans has increased tremendously in the last few years, whether it is an education loan, home loan, car loan, and etc. We want to have a system that could automate this activity at the same time it should be secured. From a loan provider’s point of view, there should be a system that should be easy to handle with no technical difficulties.

**Modules:**

1. Account module
2. Admin module
3. Loan application module
4. Login module
5. Userdetails module
6. User register

POJO CLASSES:DTO

**com.vechileloanapplication.dto**

**Account:**

========

accountNo:int

UserDetailsEntity : user;

**Admin:**

=======

String email;

String name;

String password;

**LoanApplication:**

================

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

String chassisNo;

**double** existingEMI;

**int** tenure;

**double** interest;

**double** amount;

LocalDate appdate;

String status="PENDING";

String brand;

String colour;

String model;

**int** type;

**double** exShowPrice;

**double** onRoadPrice;

UserDetailsEntity user;

**Login:**

**====**

String email;

String password;

**UserDetails:**

============

String address;

String state;

String city;

String pin;

String emptype;

**double** salary;

String aadhar;

String pan;

String salarySlip;

String addressProof;

**UserRegister:**

=============

String email;

String name;

String gender;

String mobile;

**int** age;

**Model classes:**

============

**com.vechileloanapplication.entity**

create realted model classes for entities which we have

1. Account Entity model class
2. Admin Entity model class
3. Approved loans Entity class
4. Emiclass
5. LoanApplication Entity class
6. Userdetails entity
7. User registration

**Exceptions:**

**============**

**com.vechileloanapplication.exceptions**

1.DuplicateRecordException

2.EditUserDetailsException

3.ExceptionResponse

4.GlobalExcptionhandler

5.InvalidDetailsException

6.LoanApplicationException

7.PasswordResetException

8.RecordNotFoundException

9.RegistrationException

**ServiceLayer:**

**===========**

**com.vechileloanapplication.service**

**1.AccountService**

**public** AccountEntity getAccountByEmail(String email) **throws** RecordNotFoundException;

**public** List<AccountEntity> AddAccount(AccountEntity account) **throws** RecordNotFoundException;

**public** List<AccountEntity> UpdateAccount(AccountEntity account)**throws** RecordNotFoundException;

**2. AdminService**

================

**public** AdminEntity adminRegister(AdminEntity admin) **throws** RecordNotFoundException;

**public** AdminEntity showAdminDetailsByEmail(String email) **throws** RecordNotFoundException;

**public** List<AdminEntity> updateAdmin(AdminEntity admin) **throws** RecordNotFoundException;

**public** List<AdminEntity> getAllAdmins() **throws** RecordNotFoundException;

AdminEntity logout(AdminEntity admin) **throws** RecordNotFoundException;

**3. ApprovedLoanService**

**======================**

**public** List<ApprovedLoansEntity> showAllApprovedByEmail(String email) **throws** RecordNotFoundException;

**public** ApprovedLoansEntity showApprovedByLoanId(**int** loanId) **throws** RecordNotFoundException;

**public** List<ApprovedLoansEntity> addApprovedLoan(ApprovedLoansEntity approved) **throws** RecordNotFoundException, DuplicateRecordException, InvalidDetailsException, LoanApplicationException;

List<ApprovedLoansEntity> showall();

**4.EmiService:**

**===========**

**public** **double** EMICalculate(**double** loanAmount,**int** terminYears,**double** interestRate);

**5.LoanApplicationService:**

**====================**

**public** List<LoanApplicationEntity> showAllLoanApplicationByEmail( String email) **throws** RecordNotFoundException;

**public** List<LoanApplicationEntity> showRejectedLoansByEmail(String email) **throws** RecordNotFoundException ;

**public** List<LoanApplicationEntity> modifyLoanApplicationStatus(LoanApplicationEntity loanapp) **throws** RecordNotFoundException;

**public** Optional<List<LoanApplicationEntity>> applyLoan(LoanApplicationEntity loanapp,**int** userId);

**public** Optional<LoanApplicationEntity> showLoanApplicationByChassis(String chassisNo) **throws** RecordNotFoundException;

**public** List<LoanApplicationEntity> showAllLoanApplications() **throws** RecordNotFoundException;

**public** List<LoanApplicationEntity> getAllPendingLoans() **throws** RecordNotFoundException;

**public** List<LoanApplicationEntity> getAllRejectedLoans() **throws** RecordNotFoundException;

**public** List<LoanApplicationEntity> getAllAcceptedLoans() **throws** RecordNotFoundException;

**7.LoginDtoService:**

===================

**public** UserRegistrationEntity authenticateUser(String userEmail,String password) **throws** RegistrationException, RecordNotFoundException ;

**public** List<UserRegistrationEntity> getAllUsersDetails();

**public** AdminEntity authenticateAdmin(String adminEmail,String password) **throws** RegistrationException, RecordNotFoundException;

**public** List<AdminEntity> getAllAdminDetails();

**8.LogoutService :**

====================

**public** UserRegistrationEntity logout(UserRegistrationEntity userbasic) **throws** RecordNotFoundException;

**9. UserDetailsService :**

**=======================**

**public** List<UserDetailsEntity> addUserDetails(UserDetailsEntity useradvanced, String email) **throws** RecordNotFoundException;//userDao

**public** UserDetailsEntity showUserDetailsInformationByUserId(**int** userId) **throws** RecordNotFoundException;//userDao

**public** List<UserDetailsEntity> editUserDetails(UserDetailsEntity user) **throws** RecordNotFoundException;//userDao

**public** List<UserDetailsEntity> showAllUserDetails() **throws** RecordNotFoundException;//adminDao

**10. UserRegisterService :**

**=============================**

**public** UserRegistrationEntity userRegister(UserRegistrationEntity userbasic) **throws** DuplicateRecordException;

**public** List<UserRegistrationEntity> passwordReset(UserRegistrationEntity userbasic)**throws** RecordNotFoundException;

**public** UserRegistrationEntity showUserRegistrationInformationByEmail(String email)**throws** RecordNotFoundException;

**public** List<UserRegistrationEntity> getAllUsers()**throws** RecordNotFoundException;

**Controllerlayers:**

**===================**

**com.vechileloanapplication.service**

1. AccountController

findAccount(String email);

insertAccount(AccountEntity account);

updateAccount(AccountEntity account);

2. AdminController:

adminRegister(AdminEntity admin);

showAdminDetailsByEmail(String mail);

updateAdmin(AdminEntity admin);

getAllAdmins();

logout(AdminEntity admin);

3. ApprovedLoanController

showAllApprovedByEmail(String email)

showApprovedByLoanId(**int** loanId)

addApprovedLoan(ApprovedLoansEntity approved)

getAllProducts()

4. LoanApplicationController

showLoanApplicationByEmail(String email)

showLoanApplicationByChassis(String chassisNo)

RejectedLoansByEmail(String email)

applyLoan(LoanApplicationEntity loans, **int** id)

updateLoanApplication(LoanApplicationEntity loanApp)

getAllPendingLoans()

getAllRejectedLoans()

getAllApprovedLoans()

5. LoginController

verifyUser(String password);

adminVerfication(String password)

6. logout controller :

Logout(UserRegistrationEntity userbasic);

7. UserDetailsController

updateUser(UserDetailsEntity user)

findUser(Integer userId)

getAllUsers()

addUserDetails(String email, UserDetailsEntity user)

8. UserRegisterController

userRegister(UserRegistrationEntity userbasic)

showUserRegistrationInformation(String email)

getAllUsers()

passwordReset(UserRegistrationEntity userbasic)