**Entities:**

* User(String name, String username, Address address, String password):
* Long userId;
* String name, username, password
* Address address;
* InsurancePolicy(String type, String policyholderUsername, int numberofClaims, Car car, MyDate expirtyDate):
* Long id;
* String insuranceType, policyHolderUsername;
* Int numberofClaims;
* User user;
* Car car;
* MyDate expiryDate;
* ComprehensivePolicy **extends** InsurancePolicy (super + driverAge, level)
* ThirdPartyPolicy **extends** InsurancePolicy (super + comments)
* Address (String address, String suburb, String state, int postcode)
* Car (CarType type, String model, int manufacturingYear, double carPrice) :
* enum CarType {SUV, SED, LUX. HATCH, TRUCK}
* MyDate (int day, int month, int year)

**Services:**

* InsurancePolicyService.java
* List<InsurancePolicy> getAllInsurancePolicies()

Retrieve all policies from db.

* InsurancePolicy getInsurancePolicyById (Long id)

Retrieve policy by policy’s id

* List<InsurancePolicy> getInsurancePolicyByType(String insuranceType)

Retrieve all insurance from specified type

* InsurancePolicy saveInsurancePolicy(InsurancePolicy insurancePolicy)

Save policy into the db.

* Static double calcTotalPayment(List<InsurancePolicy> policies, int flatRate)

Calculate total payment from a list of policies along with provided flat rate, return in double

* Double calcSinglePayment(Long policyId, int flatRate)

Calculate cost for single policy

* List<InsurancePolicy> getPoliciesByUsername(String username)

Retrieve all by policies owned by specified username

* InsurancePolicy createInsurancePolicy (InsurancePolicyDTO insurancePolicyDTO) throws InvalidCarTypeException

Create new insurance policy either comprehensive or third party

* List<InsurancePolicy> filterByCarsModel(String carModel) 🡪 accessing database directly
* List<InsurancePolicy> filterByCarsModel(List policies, String carModel)
* List<InsurancePolicy> filterByCarType(String type) 🡪 accessing database directly
* List<InsurancePolicy> filterByCarType(List policies, String type)
* UserService.java
* List<User> findAllUsers() – retrieve all users
* Optional<User> findUserById(Long id) – retrieve user by id
* User saveUser(User user) – save user into db
* Void deleteUser(Long id) –

Delete user from db along with the insurance policies in the db

* User findByUsername(String username) – retrieve user by username
* Boolean addPolicy(InsurancePolicy, Long userId)

addPolicy to user with the specified id.

* Double calcUserPayment(String username, int flatRate)

Calculate total payment for specified user.

* Double calcCostByType(String username, String policyType, int flatRate)

Calculate total payment of specified user and policy type.

* List<InsurancePolicy> filterByCarsModel(Long userId, String model)
* List<InsurancePolicy> filterByCarType(Long userId, String type) 🡪

Find the user and take the policies to call insurance policy function for filtering.

**Controllers / APIs: {localhost:8080}**

* InsurancePolicyController.java {localhost:8080/policies}

GET Method:

* getAllPolicies() 🡪 return policies AND status OK

({/policyId})

* getPolicyById(pathVariable Long policyId) 🡪 return policy AND status OK

(/price?id=”” & rate = “”)

* calcSinglePayment(Long policyId as id, int flatRate as rate) 🡪

return policyType, policyHolderUsername, policyId, and paymentCost

({usnername?username=””})

* getPolicyByUsername(String username) 🡪 return policies of the username

(/model?model= “ ”)

* list<InsurnacePolicy> filterByCarModel(String carModel as model)

(/carType?type=“ ”)

* list<InsurancePolicy> filterByCarType(String type as type)

POST Method:

* addPolicy(InsurancePolicyDTO insurancePolicyDTO) 🡪 example

{

    "insuranceType": "Comprehensive",

    "policyHolderUsername": "Jimmy",

    "numberOfClaims": 1,

    "car": {

        "manufacturingYear": 2024,

        "carPrice": 35000,

        "model": "Range Rover",

        "type": "SUV"

    },

    "expiryDate": {

        "day": 19,

        "month": 5,

        "year": 2030

    },

    "driverAge": 30,

    "level": 12,

    "comments": "Accidents Cover"

}

DELETE Method:

(/{policyId})

* deletePolicy(Long policyId)
* UserController.java

GET Method:

* getAllUsers() 🡪 return users AND status OK

(/id?id=””)

* getUserById(Long userId) 🡪 return user by the ID AND status OK

(/{username})

* findByUsername(PathVariable String username) 🡪 return user by usernam

POST Method:

* createUser(User user) 🡪save user into db

{

  "name": "Jimmy",

  "username": "Jimmy",

  "password": "123123",

  "flatRate": 12,

  "address": {

    "address": "Jimny st",

    "suburb": "Jeepeepe",

    "state": "IS",

    "postcode": "2500"

  }

}

(/payment)

* CalcUserTotalPayment(userPaymentDTO) 🡪 takes username and flatRate as the body

{

    "username" : "Jimmy",

    "flatRate" :  10

}

(/paymentByType)

* calcPaymentByType(userPaymentDTO) 🡪 takes username, flatRate and insuranceType as the body

{

"username" : "Jimmy",

"flatRate" :  10,

"policyType" : "Comprehensive"

}

DELETE Method:

(/{userId})

* deleteUser(Long userId) 🡪 delete user along with all of the policies belong to the user.

~~DELETE CONTROLLER NEED TO BE FIXED FOR DELETING THE POLICIES NOT JUST THE USER.~~

~~PASSWORD ISSUE – NOT BEING SAVED INTO DB.~~

~~UPDATING INFORMATION FUNCTIONS AS WELL. 🡪 ONLY UPDATE USER DATA~~

FILTER FUNCTIONS (EXPIRY DATE, ~~CAR~~)

FROM THE FILTER FUNCTIONS 🡪 GENERATE REPORT FROM CAR MODELS/TYPE REPORT AND SOME OTHER KIND OF REPORT (e.g Average holder age, average payment)

WRITE FILE INTO .CSV FILE.