Features: Core and additional

Split:

Core features:

* Register/Login (tentative have)
  + 3 – admin,customer, public
* Top up balance
  + customer
* Add,
* --------------------------------
* delete vehicle – admin features (add on)

-------------------------------------------------------------------------------------------------

* Get available vehicles (transaction)

---------------------------------------------

* Reserving a vehicle (transaction)
* Make payment (uid, deduct, updatetransation)
* Rent (
  + At page my bookings,
  + Start odo, status = rent
* Return (
  + At page my bookings,
  + End time, end odo,
* Cancel (



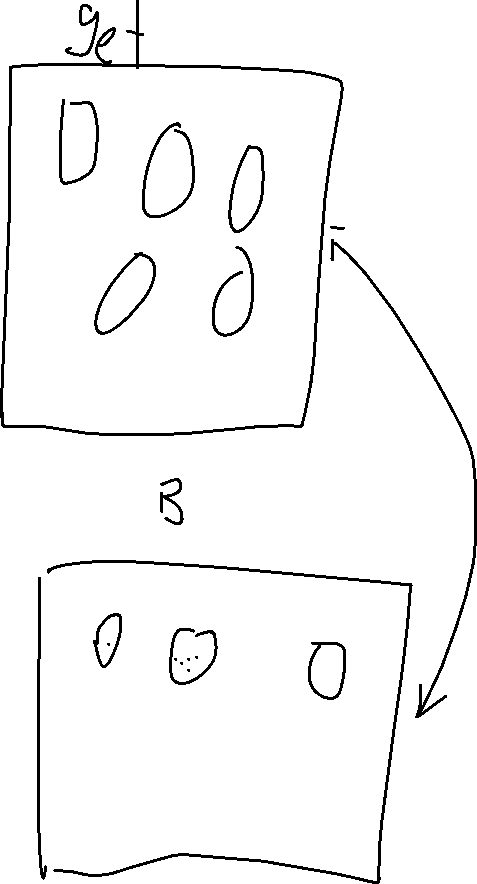
Transaction db

* Id
* Uid
* Vid
* Start time
* Duration
* End time
* Start odo
* End odo
* State: [reserved, confirmed, rent, returned]

Vehicle db : build up

Additional Feature:

* Edit (add, deduct days)
* Demerit points
* Voucher
* Vehicle on service



User

* Goes to app
* Register
* Login in
* Top up balance (customer)(min balance)
* Get available vehicles (Vehicles)
* Reserved -create transaction (transaction)
  + make payment(customer)
  + updates transaction (transaction) – booking confirm
  + updates transaction (transaction) – rented
  + updates transaction (transaction) – returned
* user got into trouble
* add demerit (Transaction)
  + deduct from balance (admin)
  + notify user (admin)
  + add demerit to customer (admin)

-----------------------------------------------------------------TBC--------------------------

* user receives voucher
  + create voucher object (Voucher)
  + customer uses voucher (Customer)
    - apply Voucher, check validity (Customer)
* vehicle on service (admin)
  + …

User needs to be able to:

1. view vehicles from a date and duration
2. query cost of vehicles
3. and rent from available vehicles
4. cancel bookings
5. make payment using in app balance
6. top up to balance

Transaction – vehicle booked, and duration.

User method

1. Get all vehicles available
   1. Input: date, duration, vehicle type
   2. Returns: all available vehicles with cost of rental
      1. Vehicle object needs to store rental cost
   3. Behaviour: query for availability (need to think about how to store)
2. Select vehicle to rent
   1. Input: vehicle id, date, duration, (userid)
   2. Returns: confirmation of availability
   3. Behaviour: reserved car id at the specified duration for a period of time
      1. Need to revert back if payment not made
3. Cancel booking
   1. Input: booking id, (user id)
   2. Returns: confirmation of cancellation
   3. Behaviour: unreserved bookings with booking id
4. Make payment
   1. Input: booking id, amount
   2. Returns: confirmation of payment
   3. Behaviour: check balance, if enough, deduct, booking to confirmed
5. Top up balance
   1. Input: amount, self.balance
   2. Returns: confirmation of top up
   3. Behaviour: update balance

User attributes:

1. Balance
2. Userid

Questions: do we need to import all of the columns in the User database

Vehicles needs to be able to:

1. Get available vehicles (from db)
2. Create new vehicle (when retrieved from db)

Vehicles method

1. Get available vehicles (from db)
   1. Input: date, duration, vehicle type
   2. Returns: all available vehicles with cost of rental
      1. Vehicle object needs to store rental cost
   3. Behaviour: query from transaction database
      1. Get all available vehicle of the vehicle type, where its date and duration not in transaction

Vehicle attributes:

1. A

Transactions needs to be able to:

1. Create transactions
2. Upload new transactions to db
3. Query transactions from uid
4. Update transaction
   1. {reserved, confirmed, trip start, strip end}
5. Query transactions from vehicle type and time

Transactions methods:

1. Create transactions:
   1. Input: required attribute to create class Transaction
   2. Return: transaction object
2. Upload new transactions to db
   1. Input: require columns in transaction db
   2. Return: uploaded transaction
   3. Behaviour: gather all information and insert to db
3. Query transactions from uid
   1. Input: uid
   2. Return: all transactions from uid
   3. Behaviour: query from database base
4. Update transaction:
   1. Input: transaction id, status
   2. Return: updated transaction
   3. Behaviour: call Transaction method to update (or just update in database straight)

Current Task:

There are 3 types, vehicle, transaction,

|  |  |
| --- | --- |
| Vehicles   * Handle vehicle fleet | |
| Attributes | Description |
| vehicles | Store the vehicle fleet details in an array |
|  | |
| Methods | Descriptions |
| getAvailableVehicles(type=all, start, duration) | Input:  start – (str) starting time  duration – (str) duration to borrow  (str) vehicle type (optional)  Returns: List of vehicles of type  Might need to add date and duration to rent to retrieve |
| addVehicle(params) | Input: Vehicle params  Behaviour: create vehicle and append to vehicles |
| removeVehicle(id) | Input: (str) vehicle id  Behaviour: remove vehicle with vehicle id from array  Returns: removed vehicle |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Vehicle   * vehicle Object | |
| Attributes | Description |
| ID | (int) PrimaryKey |
| VehicleModel | (str) Vehicle model |
| VehicleNumber | (str) Vehicle number |
| RegistrationYear | (int) Registration Year |
| FuelEconomy | (int) Fuel Economy |
| DailyRate | (float 2dp) Daily Rate to rent car |
| HourlyRate | (float 2dp) Hourly Rate to rent car |
| WeelyRate | (float 2dp) Weekly Rate to rent car |
| EngineType | {‘Diesel’,’Petrol’,’Hybrid’,’Electric’} |
| Availability | ?? how to store |
|  |  |
| NOT SO IMPT Attributes |  |
| InsuranceEnd | (datetime) date when insurance ends |
| RoadTaxEnd | (datetime) date when roadtax ends |
| Milleage | (int) Milleage |
| Seats | (int) from {2,4,7} |
| CarLogs | Array that store Object Event  { Date, EventId }  Event can be a rental activity or misc activity done to car like servicing, change tire etc |
|  |  |
| Methods |  |
| isAvailable(start, duration) | Input:  start – (str) starting time  duration – (str) duration to borrow |
| Set get Milleage |  |
| Get set Availability |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Set get Insurance |  |
| Set get RoadTax |  |
| Set get Car Logs |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Car  inherit from Vehicle | |
| Attribute |  |
| Type | Car |
|  |  |
| Lorry  Inherit from Vehicle | |
| Attribute | Description |
| Type | Lorry |
| Cargosize |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Customers | |
| Attribute | Description |
| ID |  |
| Balance |  |
| Rental History |  |
|  |  |
| Method | Description |
| RentCar(id, start, duration) | Input:   * (int) car id * (str) starting time * (str) duration to borrow   Behaviour:   * check for sufficient balance * check for availability of car * create new transaction if pass test * update vehicle availability |
| Get set balance |  |
| Get set history |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Transactions   * Store and handle all transaction |  |
| Attribute | Description |
| Transactions | Store all transaction objects |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| AddTransction(params) | Params needed to create transaction  {vehicleId, start, duration, mileageStart} |
| EditTransaction(id, key, value) | Edit to update transaction upon completion of rental |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| Transaction | |
| Attribute | Description |
| ID | PRIMARY KEY |
| Customer ID |  |
| Vehicle ID |  |
| Start | (str) Start timing |
| Duration | (str) Duration booked |
| status | {Booked, Collected, Returned} |
| MileageStart |  |
| MileageEnd |  |
|  |  |
| Method | Description |
| Get set mileage start | Property to set and get starting mileage |
| set mileage end | Property to set ending mileage |
|  |  |
| Returned(milleageEnd) | Method to indicate transaction ended  Behaviour:   * Check if duration is lesser than or equal to intended * If not trigger penalty * Update mileage end * Update returned status |
| Penalise(duration) | Method to penalize upon late return  Behaviour:   * Deduct from custormer Balance * Check if any booked slot after |