**File Management System**

**Metadata** **Class**:

Owner

Created\_at

Updated\_at

Update\_timestmap()

**FileEntity**(ABC)

Name

Metadata

@abstractmethod Display()

**File**(**FileEntity**)

Super().\_\_init\_\_(name, owner)

Content

Display()

**Folder**(**FileEntity**)

Super().\_\_init\_\_(name, owner)

Children {}

Add()

Remove()

Get()

Display()

**StorageService**

Root

Upload()

Download()

Delete()

**Calender Booking**

**RecurranceType**(Enum)

NONE,DAILY, WEEKLY, MONTHLY

**USER**

User\_id

Name

**TimeSlot**

Start

End

Overlaps()

**EVENT**

Title

Slot

Owner

Recurrence

**CalenderRepository**(ABC)

@abstractmethod

Add\_event()

Get\_event\_for\_users()

**InMemoryCalendarRepo**(CR)

User\_eent = {}

Def add\_avent()

Def get\_event\_foriser()

**CaldnerService**()

Repository

Is\_conflicting()

Book\_event()

**BUDGET APP**

**User**

User\_id

Name

wallet

**Wallet**

Transaction

def add\_trans()

def get\_balance()

**Transaction**

Id

Amount

Category

Note

Timestamp

**Funding** **Strategy**(ABC)

@abstractmethod – def fund()

**BankTransfer**

Def fund

**CreditCard**

Def fund

**BudgetManager**

User = []

Create\_user()

Add\_income()

Add\_expense()

Get\_summary()

**Meeting Room**

**TimeSlot**

Start

End

Overlaps()

\_\_repr\_\_()

**USER**

User\_id

Email

**Meeting**

Meeting\_id

Title

Timeslot

Organizer

Attendees

**MeetingRoom**

Room\_id

Name

Bookings = []

Is\_available()

Book\_meeting()

**NotificationSender**(ABC)

Send()

**EmailNotifiation**(NotficationSender)

Send()

**Meeting Room - continue**

**CalenderService**

Rooms = {}

Book\_room()

**Library Management App**

**Book**

Title

Author

Isbn

Is\_available

**USER**

Name

Email

**Member**(USER)

Super().\_\_init\_\_(name, email)

Borrowed\_books = []

**Admin**(USER)

Super().\_\_init\_\_(name, email)

**PenaltyCalculator**(ABC):

@abstractmethod calculate\_pen()

**SimplePenaltybyDay**(penaltyCalculator)

Calculate\_pen()

**NOTIFIER**(ABC)

@abstractmethod send()

**EmailNotification**(NOTIFIER):

Send()

**SMSNotification**(NOTIFIER):

Send()

**Transaction**:

User

Book

Borrow\_date

Due\_date

Return\_date

Def return\_book(): datettime.now()

**Library**:

Catalog = {}

Transactions = []

Penalty\_Cal = Siple\_pen()

Notifier = EmailNotifier()

Def add\_book()

Def borrow\_book()

Def Return\_book()

Def list\_avail\_books()

**Notepad-Text Editor**

**Command(ABC)**

@abstractmethod execute()

@abstractmethod undo()

**Editor**

Text\_buffer = []

Cursor = int 0

Undo\_stack = []

Redo\_stack = []

Def Insert\_text()

Def delete\_text()

Def move\_cursor()

Def undo()

Def redo()

**InsertCommand(Command)**

Editor

Text

Position

Def execute()

Def undo()

**DeleteCommand(Command)**

Editor

Count

Deleted\_text

Position = max(0, editor.cirsor - count)

Def execute()

Def undo()

**MoveCursor(Command)**

Editor

Offset

Prev\_cursor

Def Execute()

Def undo()

**LRU Cache**

**Node**:

Key

Value

Prev

Next

**LRUCache**

Capacity

Cache = {}

Head = Node(0, 0)

Tail = Node(0, 0)

\_add\_node(node)

\_remove\_node(node)

\_move\_to\_head(node)

\_\_pop\_tail()

get()

put()

**Parking Lot**

**VehicleType(Enum):**

BIKE = ‘BIKE’, CAR, TRUCK

**Spot Type(ENUM):**

BIKE = ‘BIKE’, CAR, TRUCK

**ParkingSpot:**

Spot\_id

Spot\_type

Vehicle

Def is\_available()

Def assign\_vehicle()

Def remove\_vehicle()

**Ticket**

Ticket\_id

Vehicle

Assigned\_spot

Entry\_time

**PricingStrategy**

Self.rates = {bike:15$}

Def calculate\_price()

**ParkingFloor**

Floor\_number

spots

def get\_available\_spot()

\_can\_fit()

Def Get\_available\_counts()

**ParkingLot**

floors

active\_tickets

pricing

Def park\_vehicle()

Def unpark\_veficle()

Def get\_available\_spots()