CASE STUDY 1: Smart Subscription Billing System

A SaaS company wants to build a **Subscription Billing Engine** that:

Must include:

- A `User` and `Subscription` class
- Methods & validation with error handling
- Custom errors when:
    - Invalid user details
    - Invalid subscription prices
- Timer logic: `setInterval` to simulate monthly billing
- Closures for discount logic
- Spread operator for combining features
- Arrow functions for internal operations
- Type conversion (price validation)

Step by step implementation

1. Create classes:

- `User`
- `Subscription`
- `PremiumSubscription` (inherits from Subscription)

2. Use:

- **Constructors**
- **Methods**
- **Static properties**
- **Error handling**
- **Custom error classes**

3. Billing logic:

- Bills user every second (simulating monthly billing)
- Uses closures for discount application
- Returns detailed billing summary

FULL CODE IMPLEMENTATION

```
// ---------------------
// CUSTOM ERRORS
// ---------------------
class ValidationError extends Error {
  constructor(message) {
    super(message);
    this.name = "ValidationError";
  }
}

// ---------------------
// USER CLASS
// ---------------------
class User {
  constructor(name, email) {
    if (!name || !email) {
      throw new ValidationError("Name and Email are
required");
    }

    this.name = name;
    this.email = email;
    this.createdAt = new Date();
  }
}

// ---------------------
// SUBSCRIPTION CLASS
// ---------------------
class Subscription {
  static company = "TechFlow Inc"; // Static property

  constructor(user, price, features = []) {
    if (!(user instanceof User)) {
      throw new ValidationError("Invalid user object");
```

```
    }

    price = Number(price); // type coercion

    if (isNaN(price) || price <= 0) {
      throw new ValidationError("Price must be a valid
positive number");
    }

    this.user = user;
    this.price = price;
    this.features = [...features]; // Spread operator to
clone array
  }

  calculateMonthlyBill() {
    return this.price;
  }
}

// ----------------------
// PREMIUM SUBSCRIPTION (INHERITANCE)
// ----------------------
class PremiumSubscription extends Subscription {
  constructor(user, price, features = [], bonusFeatures = [])
{
    super(user, price, features);
    this.bonusFeatures = [...bonusFeatures];
  }

  calculateMonthlyBill() {
    return this.price + 500; // premium fee
  }
}

// ----------------------------
// CLOSURE: DISCOUNT ENGINE
// ----------------------------
function discountEngine(discountPercent) {
  return function(price) {
    return price - (price * discountPercent) / 100;
  };
}

// ----------------------------
```

```
// BILLING ENGINE WITH TIMERS
// ----------------------------
function startBilling(subscription) {
  let count = 0;

  const apply10Percent = discountEngine(10);

  const timer = setInterval(() => {
    count++;

    let original = subscription.calculateMonthlyBill();
    let afterDiscount = apply10Percent(original);

    console.log(`----- BILL #${count} -----`);
    console.log("User:", subscription.user.name);
    console.log("Subscription:", Subscription.company);
    console.log("Original Bill:", original);
    console.log("After Discount:", afterDiscount);
    console.log("------------------------");

    if (count === 3) {
      clearInterval(timer); // Stop after 3 cycles
    }
  }, 1000);

  return timer;
}

// ----------------------------
// EXECUTION (SCENARIO)
// ----------------------------
try {
  const user = new User("Bhuvana", "bhuvana@example.com");

  const premium = new PremiumSubscription(
    user,
    "1500",                    // String → number conversion
    ["Analytics", "Dashboard"],
    ["Priority Support", "AI Reports"]
  );

  startBilling(premium);

} catch (error) {
  if (error instanceof ValidationError) {
```

```
    console.error("Validation failed:", error.message);
  } else {
    console.error("Unexpected error:", error);
  }
}
```

Key Notes:

1. User Object Creation

- If user name/email is missing → throws custom `ValidationError`
- Good for data correctness

2. Subscription Validation

- Price is passed as a *string* → converted using `Number()`
- Spread operator used to copy features
- Ensures immutability

3. Premium Subscription

- Extends Subscription
- Adds additional features
- Overrides `calculateMonthlyBill()`

4. Discount Engine (Closure)

```
discountEngine(10) → returns function that applies 10% discount
```

- Closure remembers `discountPercent`

5. Billing Using setInterval

- Every second = simulate monthly billing
- Stops after 3 cycles
- Uses calculated price + discount closure

SAMPLE OUTPUT

```
----- BILL #1 -----
User: Bhuvana
Subscription: TechFlow Inc
Original Bill: 2000
After Discount: 1800
-------------------------

----- BILL #2 -----
User: Bhuvana
Subscription: TechFlow Inc
Original Bill: 2000
After Discount: 1800
-------------------------

----- BILL #3 -----
User: Bhuvana
Subscription: TechFlow Inc
Original Bill: 2000
After Discount: 1800
-------------------------
```

(Premium price = 1500 + 500 = 2000)