

# ប្រព័ន្ធគ្រប់គ្រងទ្រព្យសម្បត្តិ (Asset Management System)

## 1 . គោលបំណង (Objective)

ប្រព័ន្ធគ្រប់គ្រងទ្រព្យសម្បត្តិ ត្រូវបានបង្កើតឡើងដើម្បីគ្រប់គ្រងទ្រព្យសម្បត្តិផ្នែករូបវន្ត និងឌីជីថលក្នុងស្ថាប័នមួយឲ្យមានរបៀបរៀបរយ និងមានភាពទុកចិត្តបាន។ ប្រព័ន្ធនេះគ្របដណ្តប់លើវដ្តជីវិតទ្រព្យសម្បត្តិទាំងមូល ចាប់ពីការចុះបញ្ជីទ្រព្យសម្បត្តិ ការផ្តល់ប្រើប្រាស់ ការត្រួតពិនិត្យស្ថានភាព ការថែទាំ និងបញ្ចប់ដោយការប្រគល់ត្រឡប់ ឬការលុបចោល (Retire)។

គោលបំណងសំខាន់ៗនៃគម្រោងនេះមានដូចជា៖ - រួមបញ្ចូលព័ត៌មានទ្រព្យសម្បត្តិទាំងអស់នៅក្នុងប្រព័ន្ធតែមួយ - កាត់បន្ថយកំហុសដែលកើតឡើងពីការគ្រប់គ្រងដោយដៃ - អនុវត្តច្បាប់អាជីវកម្ម (Business Rules) ក្នុងការប្រើប្រាស់ទ្រព្យសម្បត្តិ - បង្ហាញការអនុវត្ត Design Patterns ក្នុងប្រព័ន្ធពិត - បង្កើតកម្មវិធី Desktop ដែលងាយស្រួលថែទាំ និងពង្រីកបន្ថែម

## 2 . ទិដ្ឋភាពទូទៅនៃប្រព័ន្ធ (System Overview)

ប្រព័ន្ធគ្រប់គ្រងទ្រព្យសម្បត្តិគឺជាកម្មវិធី Desktop ដែលប្រើប្រាស់ក្នុងស្ថាប័នតែមួយ។ ប្រព័ន្ធត្រូវបានគ្រប់គ្រងដោយអ្នកគ្រប់គ្រង (Administrator) ដែលមានភារកិច្ចចុះបញ្ជី និងគ្រប់គ្រងទ្រព្យសម្បត្តិទាំងអស់។

ប្រព័ន្ធនេះអនុវត្តរចនាសម្ព័ន្ធជាថ្នាក់ (Layered Architecture) និងប្រើ Design Patterns ដើម្បីបំបែកភារកិច្ចរវាង UI, Business Logic និង Data Access ឲ្យច្បាស់លាស់។

## 3 . ទម្រង់ប្រព័ន្ធ និងមុខងារ (System Forms and Functions)

ប្រព័ន្ធត្រូវបានបែងចែកជាទម្រង់ (Forms) ផ្សេងៗ ដើម្បីគ្រប់គ្រងភារកិច្ចនីមួយៗ។

### 3 . 1 ទម្រង់ទី១៖ ការចុះបញ្ជីទ្រព្យសម្បត្តិ (Asset Registration Form)

ទម្រង់នេះប្រើសម្រាប់គ្រប់គ្រងព័ត៌មានមូលដ្ឋានរបស់ទ្រព្យសម្បត្តិ។

**មុខងារ៖** - បញ្ចូលទ្រព្យសម្បត្តិថ្មី - កែប្រែព័ត៌មានទ្រព្យសម្បត្តិ - លុបទ្រព្យសម្បត្តិ - បង្ហាញបញ្ជីទ្រព្យសម្បត្តិ

**ច្បាប់អាជីវកម្ម៖** - លេខសម្គាល់ទ្រព្យសម្បត្តិ (Asset ID) មិនអាចស្អាត - ព័ត៌មានសំខាន់មិនអាចទុកទេ

### 3 . 2 ទម្រង់ទី២៖ ការផ្តល់ទ្រព្យសម្បត្តិ (Asset Assignment Form)

ទម្រង់នេះប្រើសម្រាប់ផ្តល់ទ្រព្យសម្បត្តិឲ្យបុគ្គលិក។

**មុខងារ៖** - ផ្តល់ទ្រព្យសម្បត្តិ - កែប្រែការផ្តល់ - ប្រគល់ទ្រព្យសម្បត្តិត្រឡប់

**ច្បាប់អាជីវកម្ម៖** - ទ្រព្យសម្បត្តិដែលមានស្ថានភាព Available ប៉ុណ្ណោះអាចផ្តល់បាន - ទ្រព្យសម្បត្តិ Retired ឬ Under Repair មិនអាចផ្តល់បាន

---

### 3.3 ទម្រង់ទី៣៖ ស្ថានភាព និងការថែទាំ (Asset Status and Maintenance Form)

ទម្រង់នេះប្រើសម្រាប់គ្រប់គ្រងស្ថានភាពទ្រព្យសម្បត្តិ។

**មុខងារ៖** - ប្តូរស្ថានភាពទ្រព្យសម្បត្តិ - កត់ត្រាព័ត៌មានការថែទាំ

---

### 3.4 ទម្រង់ទី៤៖ ស្វែងរក និងរបាយការណ៍ (Search and Reporting Form)

ទម្រង់នេះប្រើសម្រាប់ត្រួតពិនិត្យ និងស្វែងរកព័ត៌មាន។

**មុខងារ៖** - ស្វែងរកតាម Asset ID - ស្វែងរកតាមឈ្មោះ - តម្រៀបទ្រព្យសម្បត្តិតាមស្ថានភាព

---

## 4 . ច្បាប់អាជីវកម្ម និងកម្រិតកំណត់ (Business Rules and Constraints)

- ទ្រព្យសម្បត្តិត្រូវចុះបញ្ជីជាមុន មុននឹងអាចផ្តល់បាន
  - ទ្រព្យសម្បត្តិកំពុងជួសជុល មិនអាចផ្តល់បាន
  - ទ្រព្យសម្បត្តិ Retired មិនអាចផ្តល់ ឬប្រគល់ត្រឡប់បាន
  - ទ្រព្យសម្បត្តិត្រូវប្រគល់ត្រឡប់ មុននឹងផ្តល់ឡើងវិញ
  - មិនអនុញ្ញាតឲ្យលុបទ្រព្យសម្បត្តិដែលកំពុងត្រូវបានប្រើប្រាស់
- 

## 5 . ករណីប្រើប្រាស់ (Use Case Descriptions)

**ករណី៖ ចុះបញ្ជីទ្រព្យសម្បត្តិ** - អ្នកប្រើប្រាស់៖ Administrator - លក្ខខណ្ឌដំបូង៖ Asset ID មិនមានស្រាប់ - ដំណើរការ៖ បញ្ចូលព័ត៌មានទ្រព្យសម្បត្តិ → រក្សាទុក - លទ្ធផល៖ ទ្រព្យសម្បត្តិត្រូវបានរក្សាទុកក្នុង Database

**ករណី៖ ផ្តល់ទ្រព្យសម្បត្តិ** - អ្នកប្រើប្រាស់៖ Administrator - លក្ខខណ្ឌដំបូង៖ ស្ថានភាព = Available - ដំណើរការ៖ ជ្រើសទ្រព្យសម្បត្តិបុគ្គលិក → ផ្តល់ - លទ្ធផល៖ ស្ថានភាពប្តូរទៅ In Use

**ករណី៖ ប្រគល់ទ្រព្យសម្បត្តិត្រឡប់** - អ្នកប្រើប្រាស់៖ Administrator - លក្ខខណ្ឌដំបូង៖ ទ្រព្យសម្បត្តិកំពុងត្រូវបានផ្តល់ - ដំណើរការ៖ ជ្រើសការផ្តល់ → ប្រគល់ត្រឡប់ - លទ្ធផល៖ ស្ថានភាពប្តូរទៅ Available

---

## 6 . Design Patterns ដែលបានប្រើប្រាស់ក្នុងប្រព័ន្ធ

ប្រព័ន្ធនេះបានអនុវត្ត Design Patterns ចំនួន 14 ដើម្បីធ្វើឲ្យរចនាសម្ព័ន្ធប្រព័ន្ធមានភាពច្បាស់លាស់ ងាយថែទាំ និងអាចពង្រីកបាន។ ខាងក្រោមនេះគឺជាការរៀបចំ **លេខរៀងត្រឹមត្រូវ** ( 6 . 1 → 6 . 1 4 ) ជាមួយឧទាហរណ៍កូដ C # ។

### 6.1 Singleton Pattern – គ្រប់គ្រង Database Connection

```
public class DatabaseConnection
{
    private static DatabaseConnection instance;
    private SqlConnection connection;

    private DatabaseConnection()
    {
        connection = new SqlConnection("connection_string");
    }

    public static DatabaseConnection Instance
    {
        get
        {
            if (instance == null)
                instance = new DatabaseConnection();
            return instance;
        }
    }

    public SqlConnection GetConnection() => connection;
}
```

### 6.2 Factory Pattern – បង្កើត Asset Objects

```
public class AssetFactory
{
    public static Asset CreateAsset(string type)
    {
        return type switch
        {
            "IT" => new ITAsset(),
            "Office" => new OfficeAsset(),
            _ => new GeneralAsset()
        };
    }
}
```

### 6.3 Abstract Factory Pattern – បង្កើត UI Components

```
public interface IUIFactory
{

```

```

    Button CreateButton();
    TextBox CreateTextBox();
}

```

#### 6.4 Builder Pattern – បង្កើត Asset ជាជំណាក់កាល

```

public class AssetBuilder
{
    private Asset asset = new Asset();
    public AssetBuilder SetName(string name)
    {
        asset.Name = name;
        return this;
    }
    public Asset Build() => asset;
}

```

#### 6.5 Prototype Pattern – Clone Asset

```

public class Asset : ICloneable
{
    public object Clone() => this.MemberwiseClone();
}

```

#### 6.6 Strategy Pattern – ច្បាប់ផ្អែកលើស្ថានភាព

```

public interface IAssetStatusStrategy
{
    bool CanAssign();
}

```

#### 6.7 State Pattern – រដ្ឋជីវិត Asset

```

public abstract class AssetState
{
    public abstract bool AssignAllowed();
}

```

## 6 . 8 Observer Pattern – Update UI ស្វ័យប្រវត្តិ

```
public interface IObserver
{
    void Update();
}
```

## 6 . 9 MVC Pattern – បំបែក Model / View / Controller

```
public class AssetController
{
    public void AddAsset(Asset asset) { }
}
```

## 6 . 1 0 Repository Pattern – Data Access Layer

```
public class AssetRepository
{
    public void Add(Asset asset) { }
}
```

## 6 . 1 1 Adapter Pattern – ភ្ជាប់ Legacy System

```
public class LegacyAssetAdapter
{
    private LegacyAsset legacy;
}
```

## 6 . 1 2 Facade Pattern – Simplify Operations

```
public class AssetService
{
    public void RegisterAsset(Asset asset) { }
}
```

## 6 . 1 3 Command Pattern – Encapsulate Actions

```
public interface ICommand
{
}
```

```

    void Execute();
}

```

#### 6.1.4 Template Method Pattern – Workflow ថេរ

```

public abstract class AssetProcess
{
    public void Process()
    {
        Validate();
        Save();
    }
    protected abstract void Validate();
    protected abstract void Save();
}

```

#### 6.2 Factory Pattern (បង្កើត Asset Object)

```

public class AssetFactory
{
    public static Asset CreateAsset(string type)
    {
        switch(type)
        {
            case "IT": return new ITAsset();
            case "Office": return new OfficeAsset();
            default: return new GeneralAsset();
        }
    }
}

```

#### 6.3 Strategy Pattern (ច្បាប់ផ្អែកលើស្ថានភាព Asset)

```

public interface IAssetStatusStrategy
{
    bool CanAssign();
}

public class AvailableStrategy : IAssetStatusStrategy
{
    public bool CanAssign() => true;
}

```

```

}

public class RetiredStrategy : IAssetStatusStrategy
{
    public bool CanAssign() => false;
}

```

#### 6.4 State Pattern (រដ្ឋដ៏រឹង Asset)

```

public abstract class AssetState
{
    public abstract bool AssignAllowed();
}

```

#### 6.5 Observer Pattern (Update UI ស្វ័យប្រវត្តិ)

```

public interface IObservable
{
    void Update();
}

public class AssetSubject
{
    private List<IObservable> observers = new();
    public void Attach(IObservable obs) => observers.Add(obs);
    public void Notify()
    {
        foreach(var obs in observers)
            obs.Update();
    }
}

```

#### 6.6 MVC Pattern (បំបែក Model / View / Controller)

```

public class AssetController
{
    private AssetRepository repo;

    public void AddAsset(Asset asset)

```

```
    {  
        repo.Add(asset);  
    }  
}
```

---

## 6 . 7 Repository Pattern (Abstract Data Access)

```
public class AssetRepository  
{  
    public void Add(Asset asset)  
    {  
        // Database logic here  
    }  
}
```

---

## 6 . 8 Command Pattern (Encapsulate User Actions)

```
public interface ICommand  
{  
    void Execute();  
}  
  
public class AddAssetCommand : ICommand  
{  
    public void Execute()  
    {  
        // Add asset logic  
    }  
}
```

---

## 6 . 9 Template Method Pattern (ដំណើរការតែមួយមានជំហានថេរ)

```
public abstract class AssetProcess  
{  
    public void Process()  
    {  
        Validate();  
        Save();  
        Log();  
    }  
}
```



```

    }

    protected abstract void Validate();
    protected abstract void Save();
    protected virtual void Log() { }
}

```

## 7 . ការផ្គូផ្គង Design Patterns ជាមួយកូដ

Design Pattern នីមួយៗត្រូវបានភ្ជាប់ទៅនឹង Class និង Form ដែលពាក់ព័ន្ធ ដើម្បីបង្ហាញពីការអនុវត្តពិតក្នុងប្រព័ន្ធ។

## 8 . តារាងសង្ខេប Design Patterns

តារាងនេះបង្ហាញពី Design Pattern ទាំង 1 4 មុខងារ និងទីតាំងប្រើប្រាស់របស់វា។

## 9 . រចនាសម្ព័ន្ធមូលដ្ឋានទិន្នន័យ (Database Design)

ប្រព័ន្ធប្រើ Database ប្រភេទ Relational ដើម្បីរក្សាទុកទិន្នន័យទ្រព្យសម្បត្តិ និងការផ្តល់។

- Assets (AssetID, Name, Category, PurchaseDate, Status)
- Assignments (AssignmentID, AssetID, EmployeeID, AssignDate, ReturnDate)

## 1 0 . ការគ្រប់គ្រងកំហុស និងការផ្ទៀងផ្ទាត់ (Error Handling and Validation)

- មិនអនុញ្ញាត Asset ID ស្អប់
- ពិនិត្យព័ត៌មានចាំបាច់
- ទប់ស្កាត់ស្ថានភាពមិនត្រឹមត្រូវ
- បង្ហាញសារកំហុសដល់អ្នកប្រើប្រាស់

## 1 1 . ការធ្វើតេស្ត និងផ្ទៀងផ្ទាត់ (Testing and Validation)

ក្នុងការធ្វើ Demo ប្រព័ន្ធ ត្រូវបង្ហាញការធ្វើការពិតតាមទម្រង់ (Forms) ខាងក្រោម។ ក្នុងឯកសារនេះ រូបភាពដាក់ Demo (Screenshot) នៅតាមទីតាំងដែលបានបញ្ជាក់។

### Demo 1 : Asset Registration Form

ទីតាំងដាក់រូបភាព៖ បន្ទាប់ពីការពិពណ៌នាទម្រង់ទី១ - Screenshot បង្ហាញការបញ្ចូល Asset ថ្មី - Screenshot បង្ហាញ Asset បន្ទាប់ពីរក្សាទុក

(ដាក់រូបភាព Demo ទីនេះ)

---

## Demo 2 : Asset Assignment Form

**ទីតាំងដាក់រូបភាព:** បន្ទាប់ពីការពិពណ៌នាទម្រង់ទី២ - Screenshot បង្ហាញការជ្រើស Asset - Screenshot បង្ហាញការផ្តល់ Asset ទៅបុគ្គលិក

(ដាក់រូបភាព Demo ទីនេះ)

---

## Demo 3 : Asset Status and Maintenance Form

**ទីតាំងដាក់រូបភាព:** បន្ទាប់ពីការពិពណ៌នាទម្រង់ទី៣ - Screenshot បង្ហាញការប្តូរស្ថានភាព (Available → Under Repair / Retired)

(ដាក់រូបភាព Demo ទីនេះ)

---

## Demo 4 : Search and Reporting Form

**ទីតាំងដាក់រូបភាព:** បន្ទាប់ពីការពិពណ៌នាទម្រង់ទី៤ - Screenshot បង្ហាញការស្វែងរក Asset - Screenshot បង្ហាញលទ្ធផលស្វែងរក

(ដាក់រូបភាព Demo ទីនេះ)

---

**សេចក្តីចំណាំ:** រូបភាព Demo គួរតែជាប្រភេទ Screenshot ពិតពីកម្មវិធី ដើម្បីបញ្ជាក់ថាប្រព័ន្ធអាចដំណើរការបានជាក់ស្តែង។

---

## 1 2 . កម្រិតកំណត់ និងការពង្រីកអនាគត (Limitations and Future Enhancements)

**កម្រិតកំណត់:** - ប្រព័ន្ធអ្នកប្រើតែមួយ (Admin) - ជាកម្មវិធី Desktop - មិនគណនាការបាត់តម្លៃស្វ័យប្រវត្តិ

**ការពង្រីកអនាគត:** - Role-based Access Control - Web-based System - Audit Log និង Depreciation - Barcode

---

## 1 3 . សន្និដ្ឋាន (Conclusion)

ប្រព័ន្ធគ្រប់គ្រងទ្រព្យសម្បត្តិនេះ បង្ហាញពីការអនុវត្ត Design Patterns ចំនួន 1 4 ក្នុងប្រព័ន្ធពិត ដោយមានច្បាប់អាជីវកម្ម និងចរនាសម្ព័ន្ធច្បាស់លាស់។ ប្រព័ន្ធនេះអាចថែទាំ ងាយពង្រីក និងអាចអភិវឌ្ឍបន្ថែមនៅអនាគតបាន។

គម្រោងនេះបានភ្ជាប់ទ្រឹស្តី Design Patterns ជាមួយការអនុវត្តជាក់ស្តែង ដោយបំពេញតាមតម្រូវការផ្នែកសិក្សា និងវិស្វកម្មកម្មវិធី។