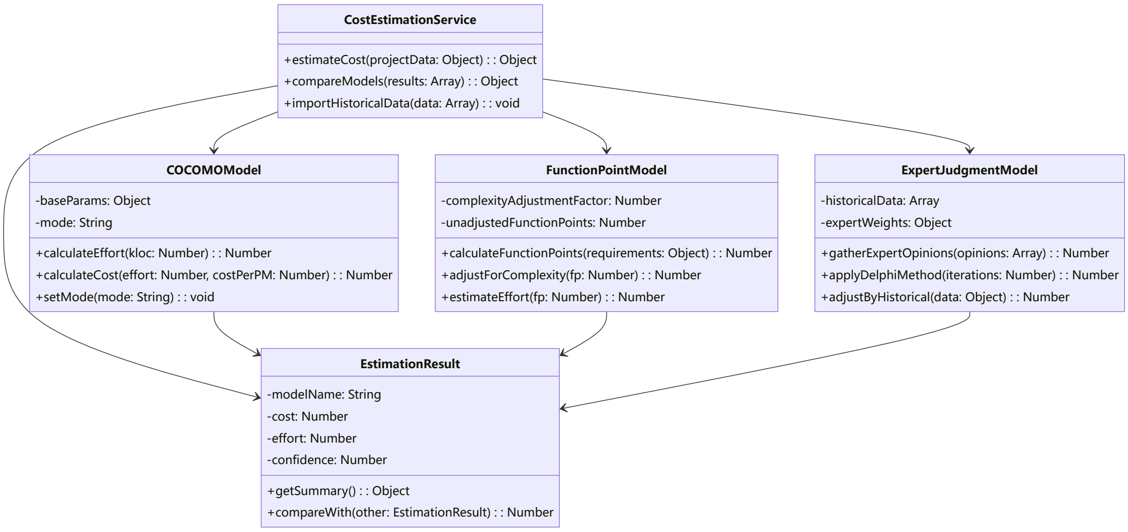
1. # Detailed Design Document for Software Project Economic Analysis and Decision-Making Tool

1. Detailed Design of Cost Estimation Module

1.1 Class Design

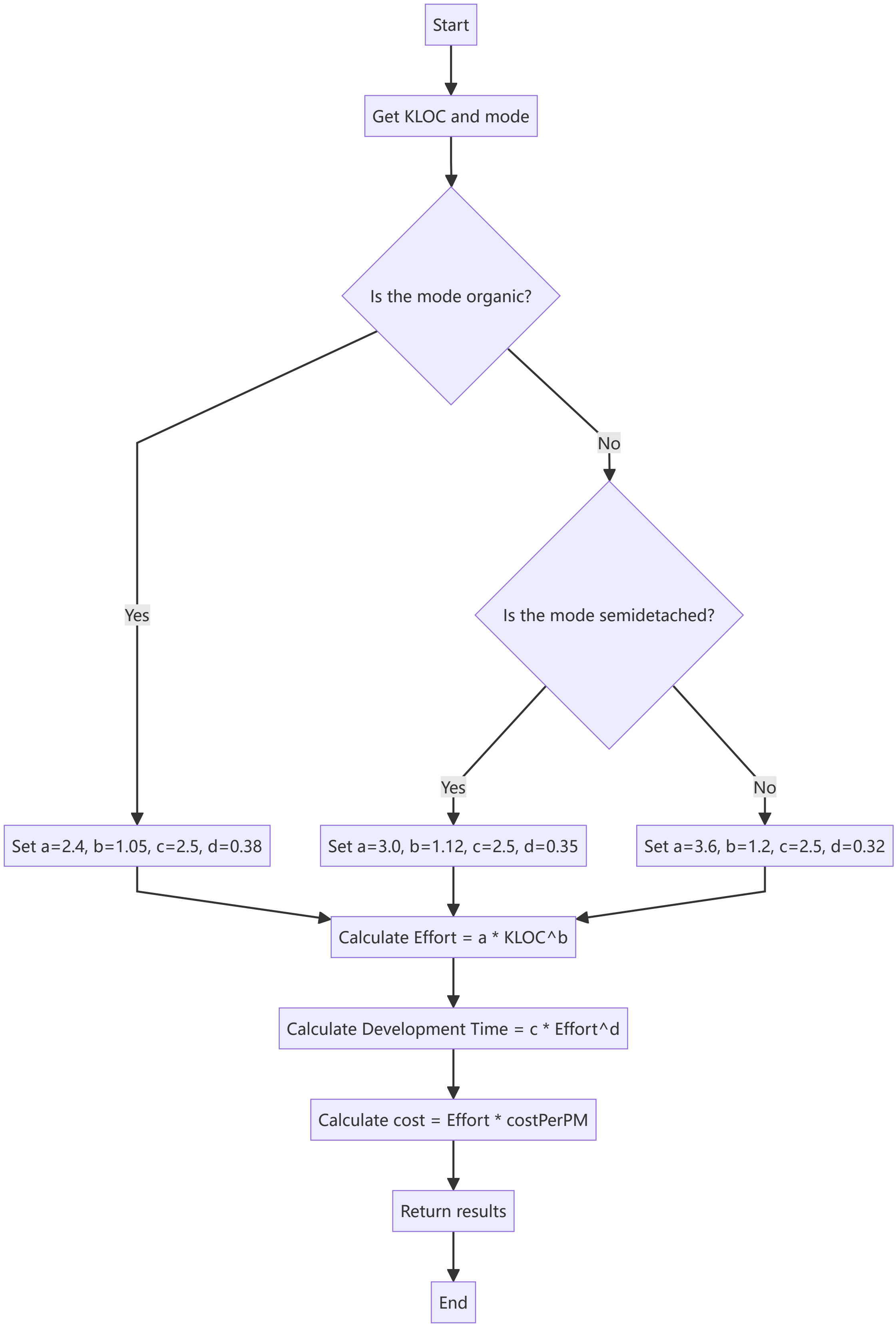


1.2 COCOMO Model Algorithm Implementation

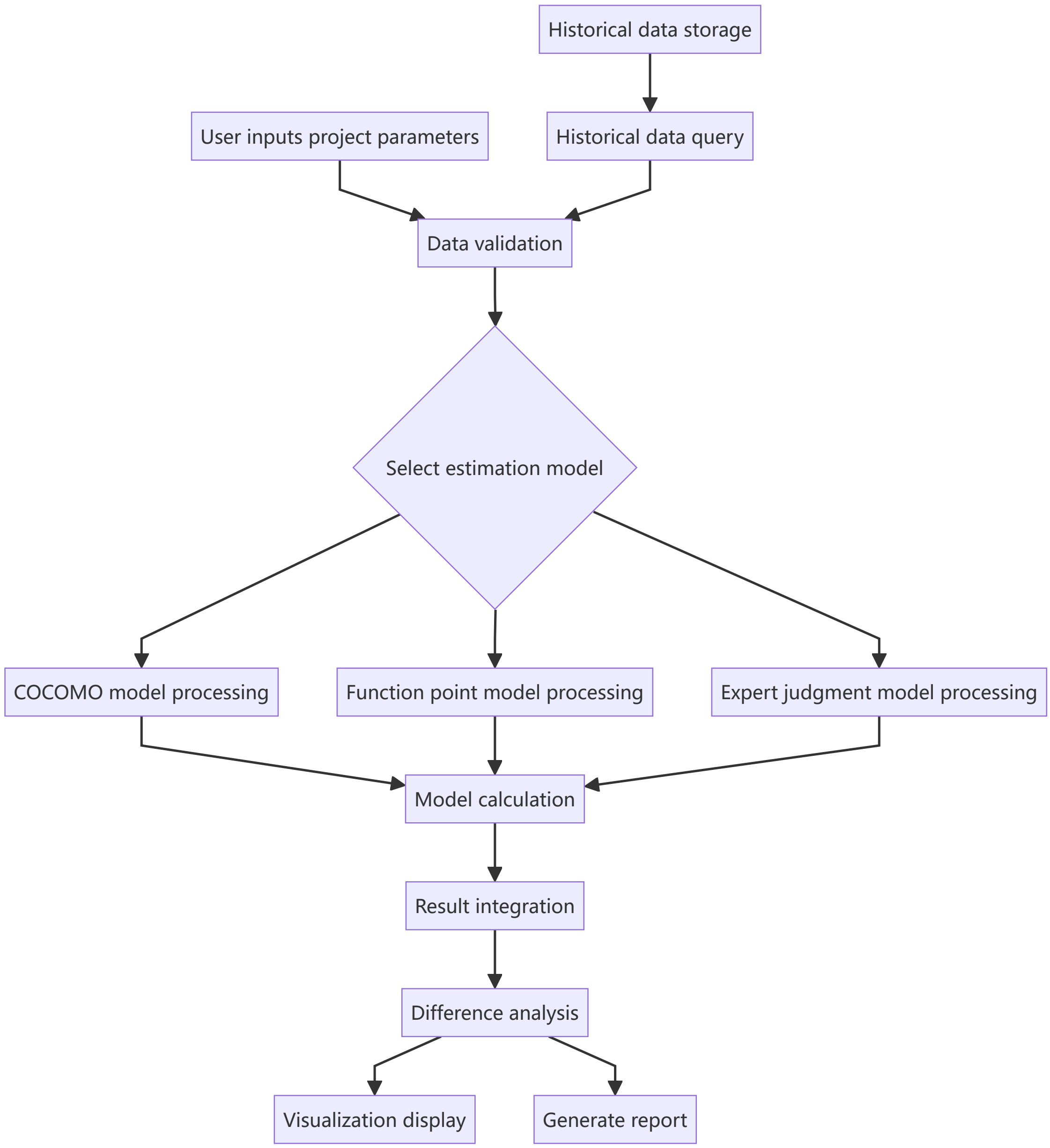
```
Effort = a * (KLOC)^b
Development Time = c * (Effort)^d
```

Parameter values (organic mode):

- a = 2.4
- b = 1.05
- c = 2.5
- d = 0.38

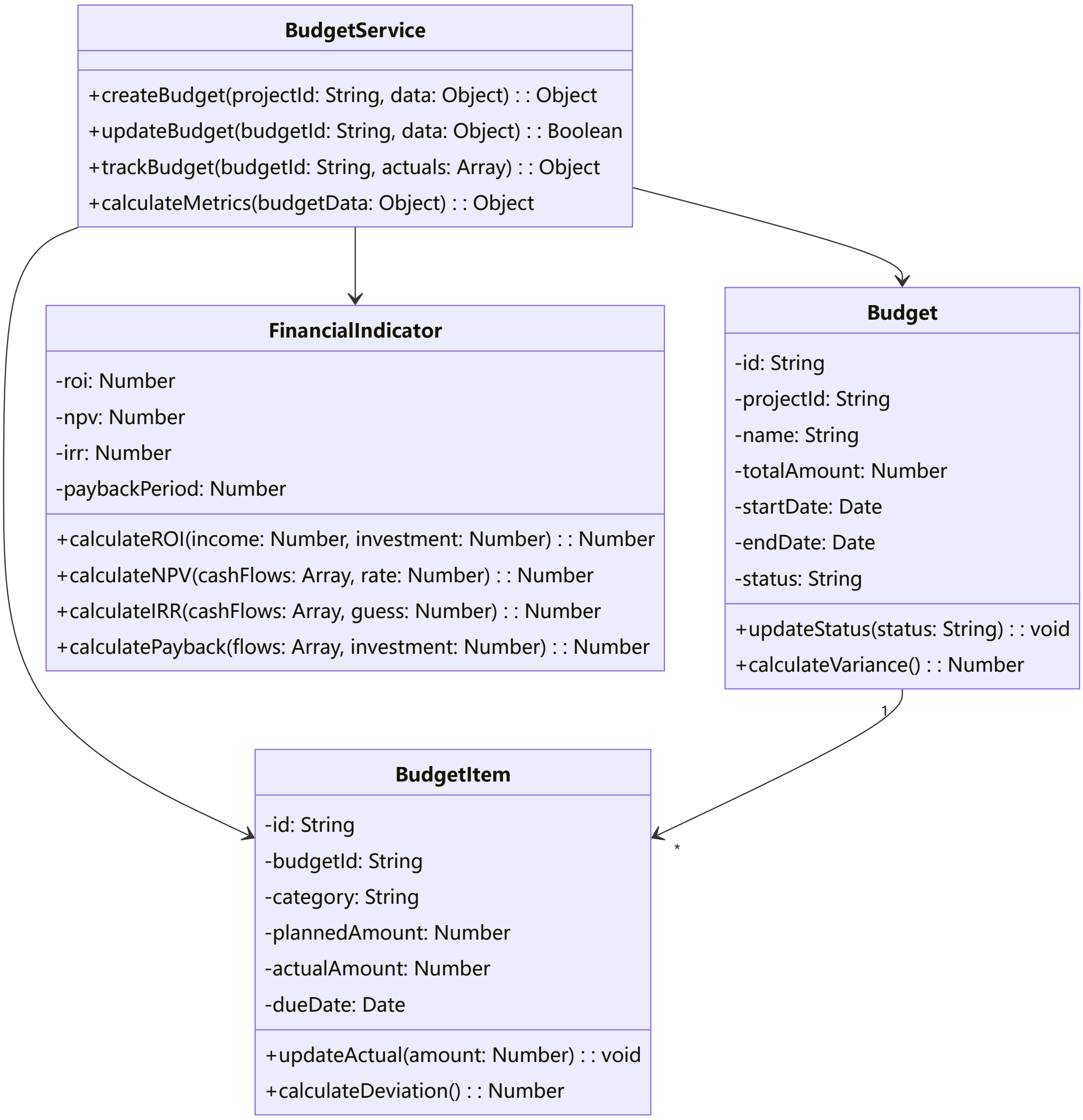

Start
Get KLOC and mode
Is the mode organic?
Yes
No
Is the mode semidetached?
Yes
No
Set a=2.4, b=1.05, c=2.5, d=0.38
Set a=3.0, b=1.12, c=2.5, d=0.35
Set a=3.6, b=1.2, c=2.5, d=0.32
Calculate Effort = a * KLOC^b
Calculate Development Time = c * Effort^d
Calculate cost = Effort * costPerPM
Return results
End

1.3 Data Flow Diagram



2. Detailed Design of Budget Management Module

2.1 Class Design



2.2 Financial Indicator Calculation Algorithm

2.2.1 ROI Calculation

```
ROI = (Total Income - Total Investment) / Total Investment * 100%
```

2.2.2 NPV Calculation

```
NPV = Σ (CashFlow_t / (1 + r)^t) - Initial Investment
```

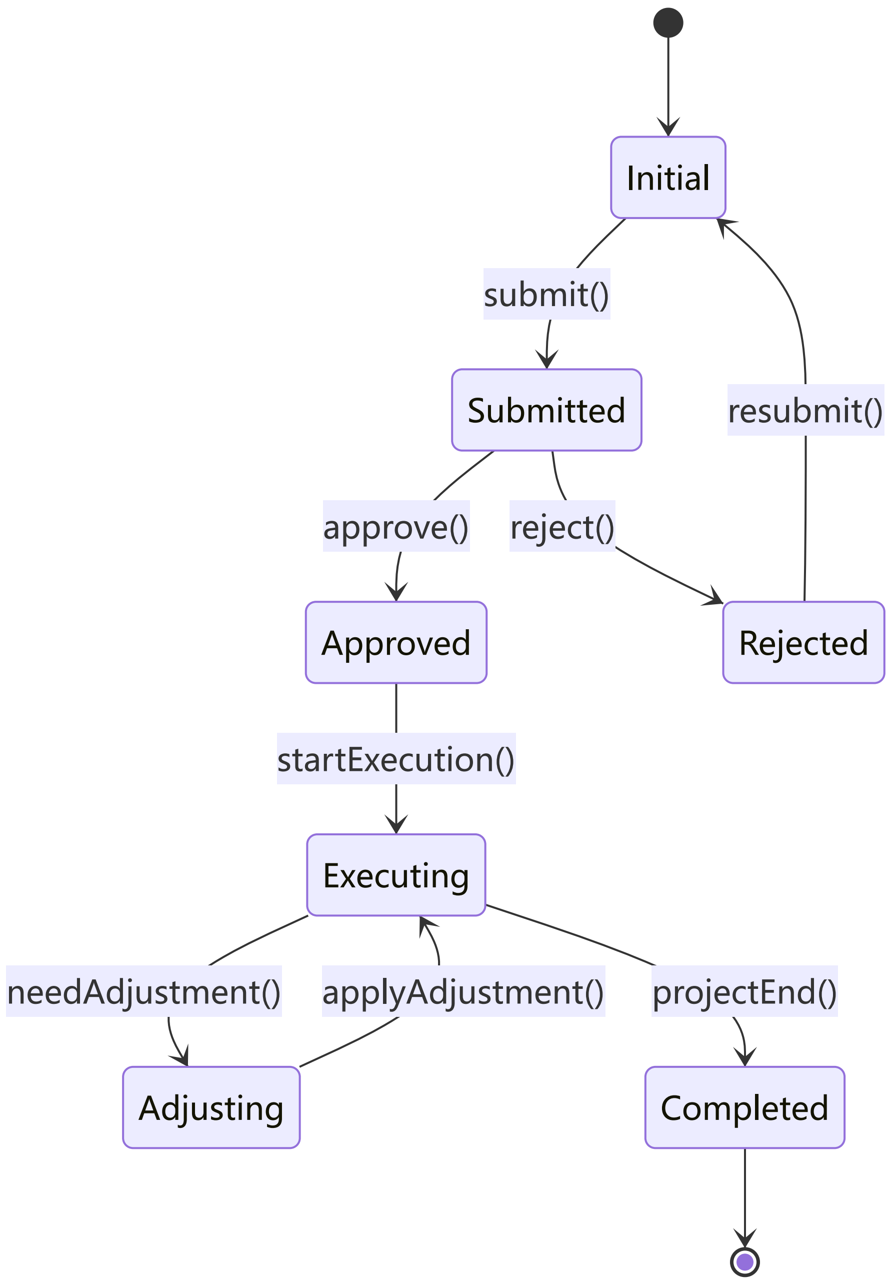
2.2.3 IRR Calculation (Newton-Raphson Method)

1. Initialize IRR guess value r0
2. Calculate NPV(r0)
3. Calculate derivative dNPV/dr
4. Iteratively update r = r0 - NPV(r0)/dNPV/dr
5. Repeat until NPV approaches 0

2.2.4 Payback Period Calculation

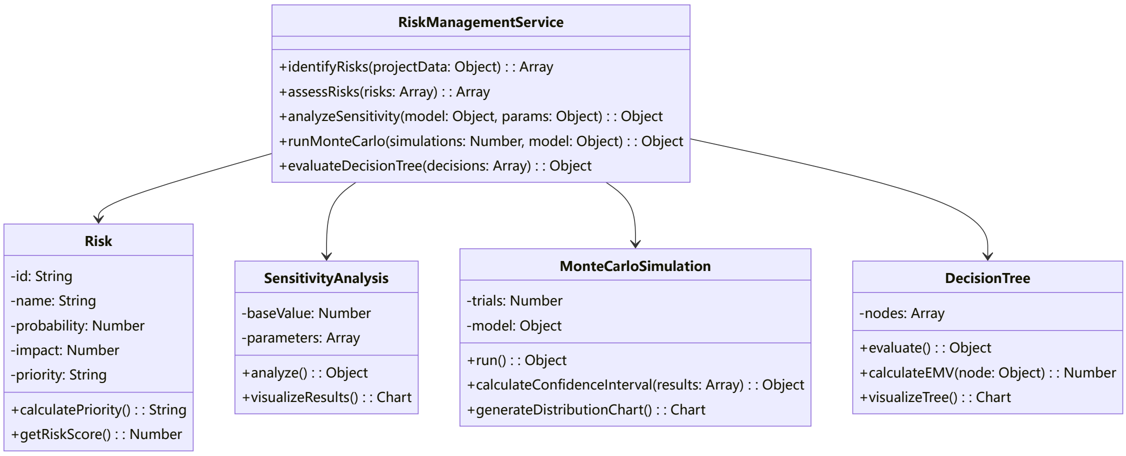
1. Cumulate cash flows until exceeding initial investment
2. Payback period = Last period with negative cumulative cash flow + (Initial investment - Cumulative cash flow) / Next period cash flow

2.3 State Transition Diagram



3. Detailed Design of Risk Management Module

3.1 Class Design

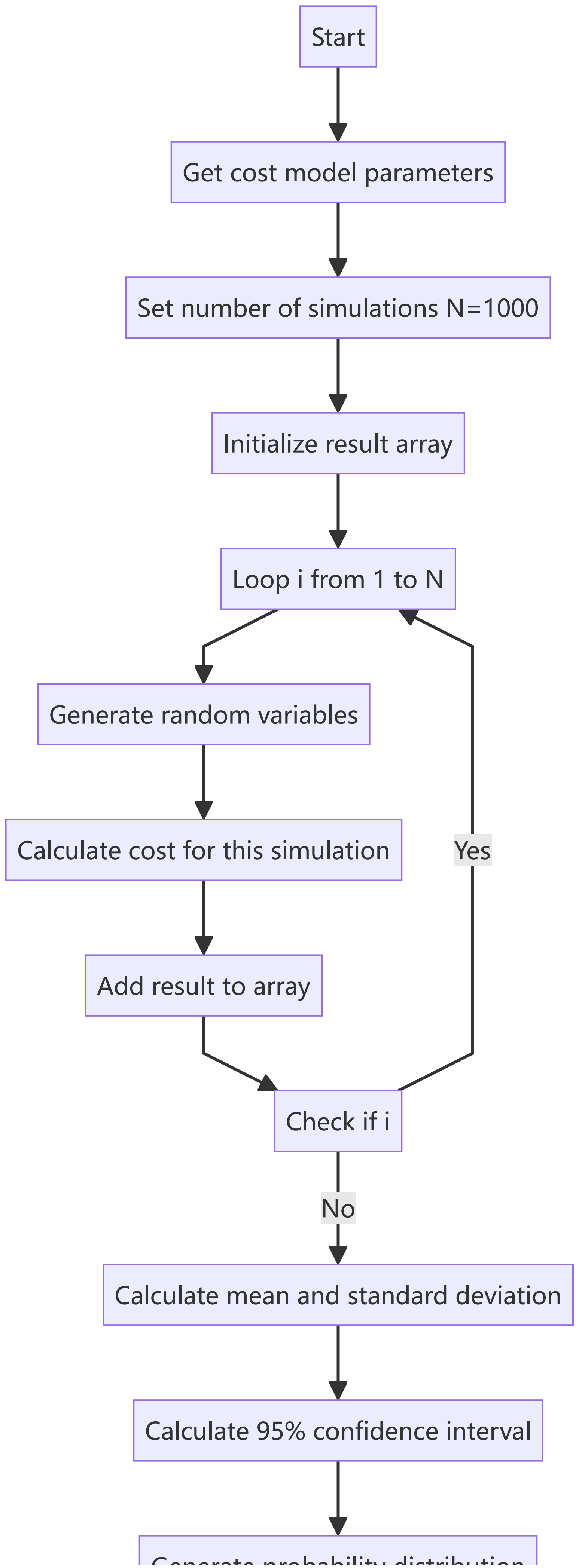


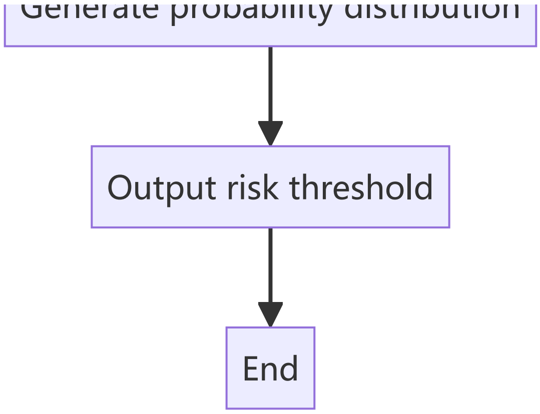
3.2 Monte Carlo Simulation Algorithm

Algorithm steps:

1. Define cost distribution parameters (mean, standard deviation)
2. Generate specified number of random samples
3. Calculate project cost for each sample
4. Statistically analyze result distribution
5. Calculate confidence interval
6. Generate probability distribution chart

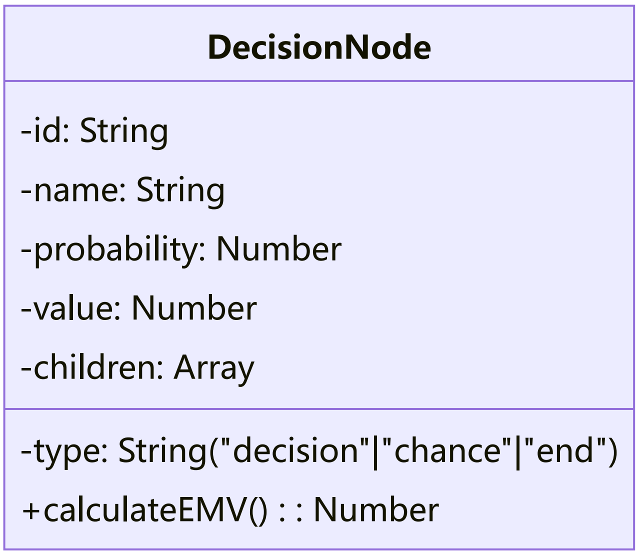
Flow chart:


Start
Get cost model parameters
Set number of simulations N=1000
Initialize result array
Loop i from 1 to N
Generate random variables
Calculate cost for this simulation
Yes
Add result to array
Check if i
No
Calculate mean and standard deviation
Calculate 95% confidence interval



3.3 Decision Tree Analysis

Decision tree node class:

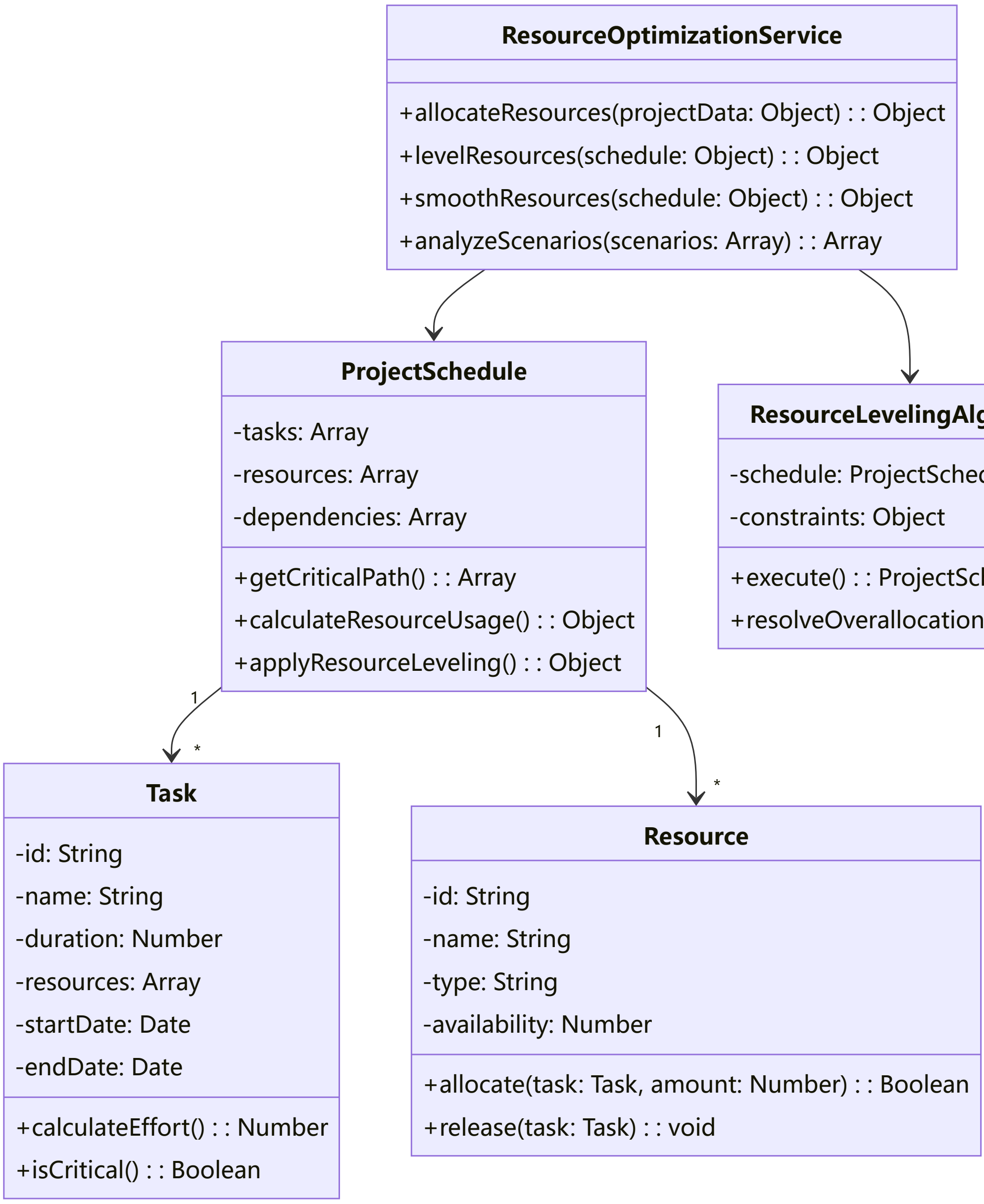


EMV calculation algorithm:

```
function calculateEMV(node) {
    if (node.type === "end") {
        return node.value;
    } else if (node.type === "chance") {
        return node.children.reduce((sum, child) => {
            return sum + child.probability * calculateEMV(child);
        }, 0);
    } else { // decision
        return Math.max(...node.children.map(child => calculateEMV(child)));
    }
}
```

4. Detailed Design of Resource Allocation and Optimization Module

4.1 Class Design

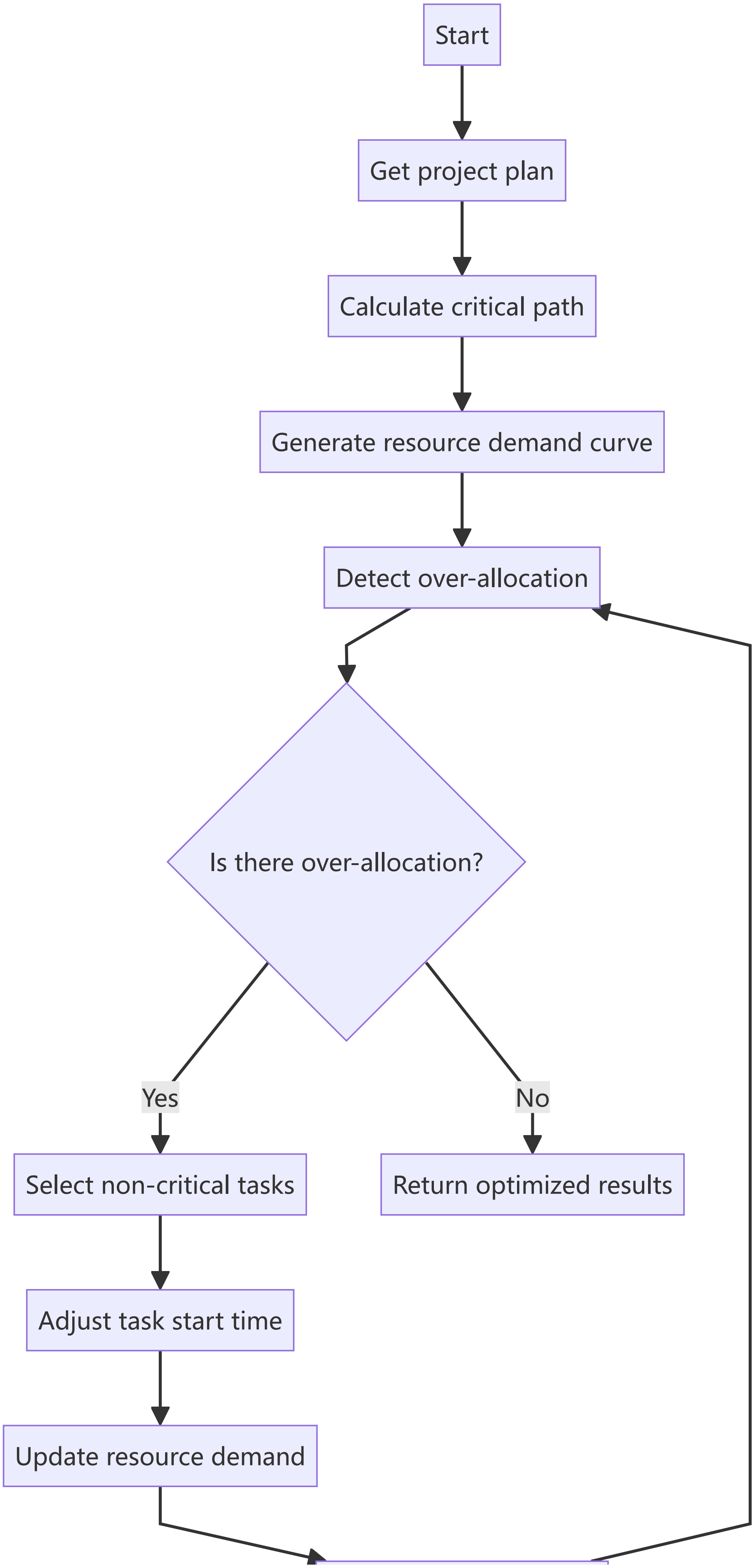


4.2 Resource Leveling Algorithm

Resource leveling algorithm based on Critical Path Method (CPM):

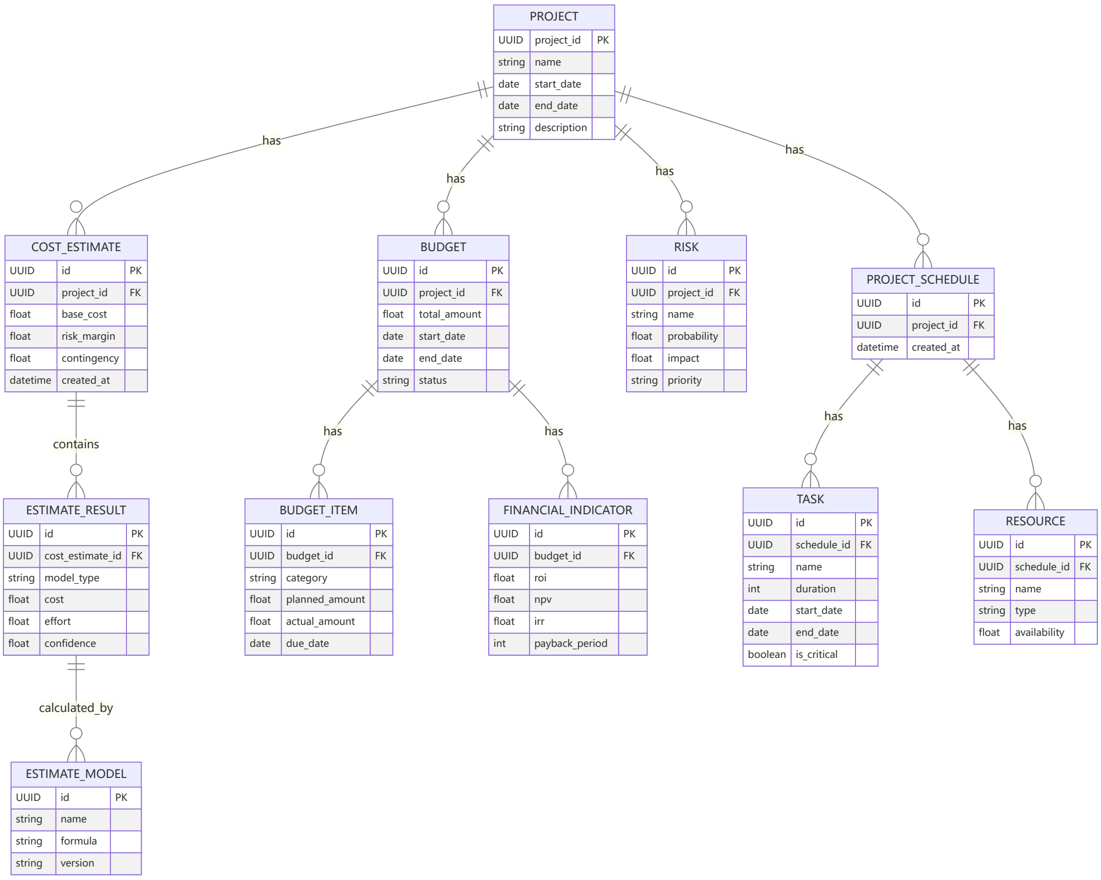
1. Construct project network diagram and determine critical path
2. Calculate initial resource demand curve
3. Identify over-allocated resources
4. Adjust non-critical path tasks:
    - Delay start time (using float time)
    - Adjust resource allocation amount
5. Repeat until resource demand is within limits

Algorithm flow chart:


Start
Get project plan
Calculate critical path
Generate resource demand curve
Detect over-allocation
Is there over-allocation?
Yes
No
Select non-critical tasks
Return optimized results
Adjust task start time
Update resource demand

5. Database Design

5.1 Entity-Relationship Diagram



6. Interface Design

6.1 Cost Estimation Module API

6.1.1 Estimate Cost

```
POST /api/v1/cost/estimate
Request Body:
{
    "projectId": "string",
    "modelType": "string",
    "parameters": {
        "kloc": number,
        "costPerPM": number,
        "mode": "string"
```

```
    }
}

Response:
{
    "cost": number,
    "effort": number,
    "modelName": "string",
    "confidence": number
}
```

6.1.2 Compare Estimation Models

```
POST /api/v1/cost/compare
Request Body:
{
    "projectId": "string",
    "modelTypes": ["string"]
}

Response:
[
    {
        "modelName": "string",
        "cost": number,
        "effort": number,
        "diffFromAverage": number
    }
]
```

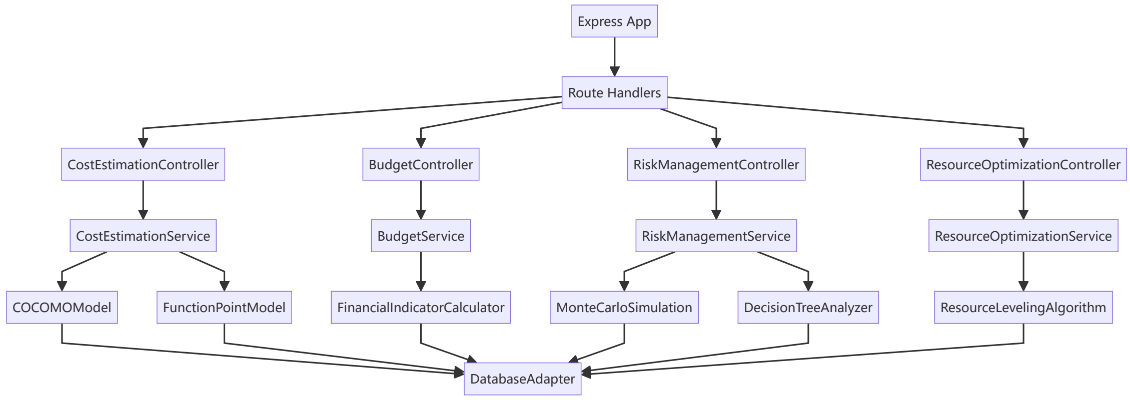
7. Frontend Component Design

7.1 Cost Estimation Page Component Structure

```
graph TD
    A[CostEstimationPage] --> B[ModelSelector]
    A --> C[ParameterForm]
    A --> D[EstimationResults]
    A --> E[ModelComparison]
    B --> F[COCOMOModelComponent]
    B --> G[FunctionPointComponent]
    B --> H[ExpertJudgmentComponent]
    D --> I[CostChart]
    D --> J[EffortTable]
    E --> K[ComparisonChart]
```

8. Backend Implementation Details

8.1 Core Module Architecture



9. Algorithm Optimization and Performance Considerations

9.1 Monte Carlo Simulation Optimization

- Use Web Workers for parallel computing to avoid UI blocking
- Implement incremental calculation to support mid-process cancellation and result preview
- Adopt stratified sampling to reduce required number of simulations

9.2 Resource Optimization Algorithm Optimization

- Use bitwise operations and matrix operations to optimize constraint solving
- Implement adaptive adjustment of algorithm parameters
- Cache frequently used calculation results to avoid repeated computations

10. Security Design

10.1 Data Encryption Scheme

- User passwords: Stored with bcrypt hashing and salt processing
- Sensitive data: Stored with AES-256 encryption, keys rotated regularly
- Communication security: All APIs encrypted via TLS 1.3

10.2 Access Control Implementation