# CI/CD Implementation Guide for Your GitHub Project

Based on the repository at https://github.com/nanuchi/my-project, here's how to implement effective CI/CD functionality to maximize your marks:

## CI/CD Functionality (30 Marks)

### 1. Set Up Basic CI Pipeline
```yaml
# .github/workflows/ci.yml
name: Continuous Integration

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4
    - name: Set up Node.js
      uses: actions/setup-node@v3
      with:
        node-version: '18'
    - name: Install dependencies
      run: npm install
    - name: Run tests
      run: npm test
```

### 2. Add CD Pipeline
```yaml
```

```
# .github/workflows/cd.yml
name: Continuous Deployment

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4
    - name: Set up Node.js
      uses: actions/setup-node@v3
      with:
        node-version: '18'
    - name: Install dependencies
      run: npm install
    - name: Build project
      run: npm run build
    - name: Deploy to production
      uses: some-deployment-action@v2
      with:
        target: production
        secrets: ${{ secrets.DEPLOY_KEY }}
```

### 3. Add Multiple Environments

```yaml
# Add to cd.yml
  staging-deploy:
    needs: test
    runs-on: ubuntu-latest
    environment: staging
    steps:
    # Similar steps as above but with staging config
```

## Structure and Clarity (10 Marks)

1. **Workflow Structure**:
   - Separate files for CI and CD (ci.yml, cd.yml)
   - Clear job naming (test, deploy, staging-deploy)
   - Logical step ordering

2. **Clarity and Conciseness**:
   - Add comments to explain complex steps
   - Use descriptive step names
   - Keep workflows focused (one job per logical unit)

## Bonus Marks (10 Marks)

### 1. Best Practices
```yaml
# Add to workflows
- name: Cache node modules
  uses: actions/cache@v3
  with:
    path: ~/.npm
    key: ${{ runner.os }}-node-${{ hashFiles('**/package-lock.json') }}
    restore-keys: |
      ${{ runner.os }}-node-
```

### 2. Error Handling

```yaml
- name: Notify on failure
  if: failure()
  uses: actions/github-script@v6
  with:
    script: |
      github.issues.createComment({
        issue_number: context.issue.number,
        owner: context.repo.owner,
        repo: context.repo.repo,
        body: "🚨 Build failed! Please check the workflow run."
      })
```

### 3. Security Scanning

```yaml
- name: Run security audit
  run: npm audit
```

### 4. Automated Versioning

```yaml
- name: Bump version
  uses: phips28/gh-action-bump-version@master
  with:
    tag-prefix: 'v'
```

## Implementation Steps

1. Create `.github/workflows` directory in your repo
2. Add the YAML files above
3. Configure necessary secrets in GitHub repo settings
4. Test by pushing changes to see workflows run
5. Monitor workflow executions and refine as needed

Remember to customize the deployment steps based on your actual deployment target (AWS, Vercel, Heroku, etc.) and adjust the Node.js version to match your project requirements.