Spécifications pour les projets de développement en IFT785

1. Objectifs pédagogiques

- Développer une application orientée objet
- Maîtriser les concepts de tests logiciels
- Appliquer les bonnes pratiques de développement
- Utiliser des design patterns appropriés
- Pratiquer le refactoring et la gestion de versions

2. Contraintes techniques

2.1 Technologies

- Langage : Au choix parmi **Java, PHP, Python, C++, Scala, ou langage OO**
- Framework : Obligation d'utiliser un framework ou une structure API
- Base de données : Au choix, adaptée au langage sélectionné
- Tests : Framework de test adapté au langage choisi
- Versioning : (repository du laboratoire, ou alors me donner l'accès à votre github)

2.2 Architecture et design

- Architecture en couches (minimum 3 couches)
- Implémentation de **6 à 8 designs patterns différents**
- Documentation des choix architecturauxx
- Respect des principes SOLID

2.3 Tests

- Tests unitaires obligatoires
- Tests d'intégration obligatoires
- Tests end-to-end obligatoires
- Couverture de tests minimale : 50%
- Documentation des scénarios de test

3. Organisation et planning

3.1 Durée et charge de travail

- **Durée totale : 6 semaines (début 1er mars 2025 au 11 avril 2025)**
- Charge hebdomadaire : 3 heures
- Total : 18 heures de travail effectif

3.2 Jalons

- Semaine 2 : Architecture et design patterns définis
- Semaine 4 : Première version fonctionnelle avec tests
- Semaine 6 : Version finale avec documentation complète

4. Livrables

4.1 Documentation

- Document d'architecture détaillé
- Description des design patterns utilisés
- Documentation de l'API (si applicable)
- Guide de déploiement
- Rapport de test avec métriques de couverture

4.2 Code

- Code source commenté
- Tests automatisés
- Fichiers de configuration

4.3 Présentation

- Support de présentation technique
- Démonstration fonctionnelle
- Justification des choix techniques

5. Critères d'Évaluation

5.1 Qualité du Code (50%)

- Respect des principes SOLID
- Pertinence des design patterns utilisés
- Qualité du refactoring
- Lisibilité et maintenabilité
- Gestion des erreurs

5.2 Tests (30%)

- Atteinte de la couverture minimale
- Pertinence des scénarios de test
- Qualité des assertions
- Organisation des tests

5.3 Architecture (10%)

- Cohérence de l'architecture
- Séparation des responsabilités
- Qualité de la documentation
- Respect des contraintes techniques
- Utilisation appropriée du framework

5.4 Méthodologie (10%)

- Utilisation de Git
- Qualité des commits
- Respect des délais
- Organisation du code
- Qualité de la présentation