**1. Introduction**

**Purpose:**  
*Commerce Digital Application* is a full-stack web platform designed to provide online banking services such as account management, fund transfers, and personalized digital financial experiences.

**Objectives:** The

* Deliver a secure and user-friendly banking interface.
* Enable seamless transactions and real-time data retrieval.
* Integrate a scalable backend API and MySQL database, deployed through Docker containers for modularity and portability.

**2. Project Lifecycle & Approach**

Based on Adaptive Planning principles, this project will use short, iterative development cycles. Near-term work is planned in detail, and future iterations are refined progressively

Project\_Planning

.

| **Phase** | **Duration** | **Description** |
| --- | --- | --- |
| **Initiation** | Week 1 | Define goals, roles, and scope |
| **Planning** | Week 2 | Create WBS, estimates, and release plan |
| **Execution (Iterative)** | Weeks 3–7 | Frontend, backend, and database integration |
| **Testing & Validation** | Weeks 6–7 | Integration testing and refinements |
| **Deployment & Handover** | Week 8 | Docker deployment, documentation |

**3. Work Breakdown Structure (WBS)**

Following the WBS model from the reference document

Project\_Planning

:

**1.0 Initiation**

1.1 Define goals and objectives  
1.2 Identify stakeholders  
1.3 Establish success metrics

**2.0 Planning**

2.1 Create architecture overview (frontend, backend, database, Docker)  
2.2 Define milestones and iterations  
2.3 Develop project schedule and Gantt chart

**3.0 Frontend Development (React + TypeScript)**

3.1 Setup environment using Vite  
3.2 Create reusable UI components  
3.3 Implement pages: Dashboard, Accounts, Payments, Profile  
3.4 Integrate Axios API calls  
3.5 Testing & linting setup

**4.0 Backend Development (Node.js + Express)**

4.1 Initialize backend server  
4.2 Create routes (Accounts, Payments, Profile)  
4.3 Setup database connection using mysql2 / Sequelize  
4.4 Implement API authentication and middleware  
4.5 Unit testing

**5.0 Database Development (MySQL)**

5.1 Define schema for users, accounts, transactions  
5.2 Configure Dockerized MySQL  
5.3 Write initial seed scripts  
5.4 Validate CRUD operations

**6.0 Integration & Testing**

6.1 Integrate frontend and backend APIs  
6.2 Conduct system testing and bug tracking  
6.3 Optimize Docker Compose services

**7.0 Deployment & Documentation**

7.1 Build Docker images  
7.2 Deploy to AWS / Render  
7.3 Write README and final project report

**4. Schedule (Gantt-style Overview)**

| **Week** | **Task** |
| --- | --- |
| **Week 1** | Project initiation, setup Git & Docker environment |
| **Week 2** | Architecture design, WBS creation, iteration plan |
| **Week 3–4** | Frontend UI and routing |
| **Week 4–5** | Backend REST APIs and DB schema |
| **Week 5–6** | Integration testing, debugging |
| **Week 7** | Docker orchestration and performance tuning |
| **Week 8** | Deployment, documentation, presentation |

**5. Communication Plan**

Aligned with the principles in the PowerPoint file

Project\_Planning

:

* **Weekly team meetings** – to track progress and blockers.
* **GitHub issues** – for task assignment and bug tracking.
* **Slack/Email updates** – for day-to-day coordination.
* **Milestone reviews** – at the end of each iteration.

**6. Risk Management Plan**

| **Risk** | **Impact** | **Mitigation** |
| --- | --- | --- |
| Docker container communication issues | High | Use Docker Compose networking and .env configs |
| Database data loss | High | Mount MySQL volume and schedule daily backups |
| API request failure | Medium | Add retry logic and centralized error handling |
| Team coordination delays | Medium | Weekly progress meetings and daily commits |

**7. Quality & Metrics Plan**

* **Code Quality:** Linting, Prettier formatting
* **Testing:** Jest for backend, React Testing Library for frontend
* **Performance:** Lighthouse audits for UI; load testing for API
* **Metrics:** Velocity tracking, error rates, test coverage percentage

**8. Configuration & Change Control**

As per the configuration management guidelines in the document

Project\_Planning

:

* All code changes tracked in Git (feature branches + pull requests).
* Docker and .env files versioned.
* API and DB schema changes logged in /docs/changelog.md.

**9. Deliverables**

| **Deliverable** | **Description** |
| --- | --- |
| Frontend | React + TypeScript SPA with pages |
| Backend | Node.js REST API |
| Database | MySQL schema and Dockerized instance |
| Documentation | README, API docs, and project report |
| Deployment | Docker Compose production setup |

**10. Progressive Elaboration**

This project follows *progressive elaboration*, refining requirements and deliverables with each iteration. Each sprint produces working code, validated through stakeholder feedback