

TODO List Chi Tiết Cụ Thể - Website Dịch Vụ Toán Học

Detailed Specific TODO List - Math Service Website

Tác giả / Author: Manus AI

Phiên bản / Version: 2.3 (Đã bổ sung QR Payment)

Ngày cập nhật: 14/08/2025

Phase 1: Infrastructure và DevOps Foundation (3 tuần / 3 weeks)

Phase 1.1: Thiết lập Môi trường Phát triển / Development Environment Setup (1 tuần / 1 week)

Task 1.1.1: Tạo Docker Compose file cho toàn bộ hệ thống / Create Docker Compose file for entire system

- ☐ Tạo file docker-compose.yml trong root directory / Create docker-compose.yml file in root directory
- ☐ Định nghĩa service cho User Service (port 8001) / Define service for User Service (port 8001)
- ☐ Định nghĩa service cho Payment Service (port 8002) / Define service for Payment Service (port 8002)
- ☐ Định nghĩa service cho Math Solver Service (port 8003) / Define service for Math Solver Service (port 8003)
- ☐ Định nghĩa service cho Content Service (port 8004) / Define service for Content Service (port 8004)
- ☐ Định nghĩa service cho Admin Service (port 8005) / Define service for Admin Service (port 8005)
- ☐ Định nghĩa service cho Next.js Frontend (port 3000) / Define service for Next.js Frontend (port 3000)
- ☐ Cấu hình networks: backend-network, frontend-network / Configure networks: backend-network, frontend-network
- ☐ Cấu hình volumes: postgres-data, redis-data / Configure volumes: postgres-data, redis-data

Task 1.1.2: Thiết lập Dockerfile templates cho FastAPI services / Setup Dockerfile templates for FastAPI services

- ☐ Tạo Dockerfile.fastapi template với Python 3.11 / Create Dockerfile.fastapi template with Python 3.11
- ☐ Cấu hình requirements.txt với: FastAPI, SQLAlchemy, Alembic, psycpg2, redis, pytest / Configure requirements.txt with: FastAPI, SQLAlchemy, Alembic, psycpg2, redis, pytest
- ☐ Thiết lập working directory /app / Setup working directory /app
- ☐ Cấu hình EXPOSE port 8000 / Configure EXPOSE port 8000
- ☐ Thiết lập CMD với uvicorn / Setup CMD with uvicorn
- ☐ Copy template cho User Service → services/user-service/Dockerfile / Copy template for User Service → services/user-service/Dockerfile
- ☐ Copy template cho Payment Service → services/payment-service/Dockerfile / Copy template for Payment Service → services/payment-service/Dockerfile
- ☐ Copy template cho Math Solver Service → services/math-solver-service/Dockerfile / Copy template for Math Solver Service → services/math-solver-service/Dockerfile
- ☐ Copy template cho Content Service → services/content-service/Dockerfile / Copy template for Content Service → services/content-service/Dockerfile
- ☐ Copy template cho Admin Service → services/admin-service/Dockerfile / Copy template for Admin Service → services/admin-service/Dockerfile

Task 1.1.3: Thiết lập Dockerfile cho Next.js frontend / Setup Dockerfile for Next.js frontend

- ☐ Tạo frontend/Dockerfile với Node.js 20 / Create frontend/Dockerfile with Node.js 20
- ☐ Cấu hình multi-stage build: dependencies, builder, runner / Configure multi-stage build: dependencies, builder, runner
- ☐ Thiết lập package.json với: Next.js 14, React 18, TypeScript, Tailwind CSS / Setup package.json with: Next.js 14, React 18, TypeScript, Tailwind CSS
- ☐ Cấu hình EXPOSE port 3000 / Configure EXPOSE port 3000
- ☐ Thiết lập CMD với npm start / Setup CMD with npm start

Task 1.1.4: Cấu hình PostgreSQL databases cho từng service / Configure PostgreSQL databases for each service

- ☐ Tạo PostgreSQL container trong docker-compose với port 5432 / Create PostgreSQL container in docker-compose with port 5432
- ☐ Tạo database user_service_db cho User Service / Create database user_service_db for User Service

- ☐ Tạo database payment_service_db cho Payment Service / Create database payment_service_db for Payment Service
- ☐ Tạo database math_solver_db cho Math Solver Service / Create database math_solver_db for Math Solver Service
- ☐ Tạo database content_service_db cho Content Service / Create database content_service_db for Content Service
- ☐ Tạo database admin_service_db cho Admin Service / Create database admin_service_db for Admin Service
- ☐ Cấu hình user postgres với password postgres123 / Configure user postgres with password postgres123
- ☐ Thiết lập persistent volume cho PostgreSQL data / Setup persistent volume for PostgreSQL data

Task 1.1.5: Thiết lập Redis cho caching và message queue / Setup Redis for caching and message queue

- ☐ Tạo Redis container trong docker-compose với port 6379 / Create Redis container in docker-compose with port 6379
- ☐ Cấu hình Redis database 0 cho User Service caching / Configure Redis database 0 for User Service caching
- ☐ Cấu hình Redis database 1 cho Payment Service caching / Configure Redis database 1 for Payment Service caching
- ☐ Cấu hình Redis database 2 cho Math Solver Service caching / Configure Redis database 2 for Math Solver Service caching
- ☐ Cấu hình Redis database 3 cho Message Queue / Configure Redis database 3 for Message Queue
- ☐ Thiết lập persistent volume cho Redis data / Setup persistent volume for Redis data
- ☐ Cấu hình Redis password redis123 / Configure Redis password redis123

Task 1.1.6: Tạo database initialization scripts / Create database initialization scripts

- ☐ Tạo scripts/init-databases.sql để tạo tất cả databases / Create scripts/init-databases.sql to create all databases
- ☐ Tạo scripts/user-service/init.sql với bảng: users, user_sessions, user_balance, user_roles / Create scripts/user-service/init.sql with tables: users, user_sessions, user_balance, user_roles
- ☐ Tạo scripts/payment-service/init.sql với bảng: transactions, payment_methods, transaction_logs, balances / Create scripts/payment-service/init.sql with tables: transactions, payment_methods, transaction_logs, balances

☐ Tạo scripts/math-solver-service/init.sql với bảng: math_problems, solutions, solution_history / Create scripts/math-solver-service/init.sql with tables: math_problems, solutions, solution_history

☐ Tạo scripts/content-service/init.sql với bảng: pages, faqs, content_categories, translations / Create scripts/content-service/init.sql with tables: pages, faqs, content_categories, translations

☐ Tạo scripts/admin-service/init.sql với bảng: admin_users, system_settings, audit_logs / Create scripts/admin-service/init.sql with tables: admin_users, system_settings, audit_logs

Task 1.1.7: Thiết lập development environment variables / Setup development environment variables

☐ Tạo .env.development với DATABASE_URLs cho tất cả services / Create .env.development with DATABASE_URLs for all services

☐ Cấu hình

USER_SERVICE_DB_URL=postgresql://postgres:postgres123@postgres:5432/user_service_db / Configure

USER_SERVICE_DB_URL=postgresql://postgres:postgres123@postgres:5432/user_service_db

☐ Cấu hình

PAYMENT_SERVICE_DB_URL=postgresql://postgres:postgres123@postgres:5432/payment_service_db / Configure

PAYMENT_SERVICE_DB_URL=postgresql://postgres:postgres123@postgres:5432/payment_service_db

☐ Cấu hình

MATH_SOLVER_DB_URL=postgresql://postgres:postgres123@postgres:5432/math_solver_db / Configure

MATH_SOLVER_DB_URL=postgresql://postgres:postgres123@postgres:5432/math_solver_db

☐ Cấu hình

CONTENT_SERVICE_DB_URL=postgresql://postgres:postgres123@postgres:5432/content_service_db / Configure

CONTENT_SERVICE_DB_URL=postgresql://postgres:postgres123@postgres:5432/content_service_db

☐ Cấu hình

ADMIN_SERVICE_DB_URL=postgresql://postgres:postgres123@postgres:5432/admin_service_db / Configure

ADMIN_SERVICE_DB_URL=postgresql://postgres:postgres123@postgres:5432/admin_service_db

☐ Cấu hình REDIS_URL=redis://:redis123@redis:6379 / Configure REDIS_URL=redis://:redis123@redis:6379

☐ Cấu hình JWT secrets: JWT_SECRET_KEY, JWT_REFRESH_SECRET_KEY / Configure JWT secrets: JWT_SECRET_KEY, JWT_REFRESH_SECRET_KEY

☐ Cấu hình API keys: VNPAY_API_KEY, MOMO_API_KEY / Configure API keys: VNPAY_API_KEY, MOMO_API_KEY

Task 1.1.8: Cấu hình code quality tools / Configure code quality tools

☐ Tạo pyproject.toml với cấu hình Black formatter / Create pyproject.toml with Black formatter configuration

☐ Cấu hình flake8 trong .flake8 với max-line-length=88, ignore=E203,W503 / Configure flake8 in .flake8 with max-line-length=88, ignore=E203,W503

☐ Cấu hình mypy trong mypy.ini với strict mode cho tất cả services / Configure mypy in mypy.ini with strict mode for all services

☐ Thiết lập isort cho import sorting / Setup isort for import sorting

☐ Cấu hình pytest trong pytest.ini với test directories cho từng service / Configure pytest in pytest.ini with test directories for each service

Task 1.1.9: Thiết lập pre-commit hooks / Setup pre-commit hooks

☐ Tạo .pre-commit-config.yaml / Create .pre-commit-config.yaml

☐ Cấu hình hook cho Black formatting / Configure hook for Black formatting

☐ Cấu hình hook cho flake8 linting / Configure hook for flake8 linting

☐ Cấu hình hook cho mypy type checking / Configure hook for mypy type checking

☐ Cấu hình hook cho isort import sorting / Configure hook for isort import sorting

☐ Cấu hình hook cho pytest unit tests / Configure hook for pytest unit tests

☐ Install pre-commit hooks trong development environment / Install pre-commit hooks in development environment


Task 1.1.10: Tạo development documentation / Create development documentation

☐ Tạo README.md với hướng dẫn setup project / Create README.md with project setup instructions

☐ Tạo docs/development-setup.md với chi tiết setup từng service / Create docs/development-setup.md with detailed setup for each service

☐ Tạo docs/database-schema.md với ERD cho tất cả databases / Create docs/database-schema.md with ERD for all databases

☐ Tạo docs/api-endpoints.md với danh sách endpoints cho từng service / Create docs/api-endpoints.md with endpoint list for each service

- ☐ Tạo docs/environment-variables.md với giải thích tất cả env vars / Create docs/environment-variables.md with explanation of all env vars
- ☐  [BỔ SUNG] Tạo thư mục và file mẫu cho Hồ sơ Quyết định Kiến trúc (docs/adr/ADR-001-template.md)


Phase 1.2: Thiết lập CI/CD Pipeline / CI/CD Pipeline Setup (1 tuần / 1 week)

Task 1.2.1: Thiết lập GitHub Actions workflows / Setup GitHub Actions workflows

- ☐ Tạo .github/workflows/user-service.yml cho User Service / Create .github/workflows/user-service.yml for User Service
- ☐ Tạo .github/workflows/payment-service.yml cho Payment Service / Create .github/workflows/payment-service.yml for Payment Service
- ☐ Tạo .github/workflows/math-solver-service.yml cho Math Solver Service / Create .github/workflows/math-solver-service.yml for Math Solver Service
- ☐ Tạo .github/workflows/content-service.yml cho Content Service / Create .github/workflows/content-service.yml for Content Service
- ☐ Tạo .github/workflows/admin-service.yml cho Admin Service / Create .github/workflows/admin-service.yml for Admin Service
- ☐ Tạo .github/workflows/frontend.yml cho Next.js Frontend / Create .github/workflows/frontend.yml for Next.js Frontend
- ☐ Cấu hình triggers: push to main, pull requests / Configure triggers: push to main, pull requests

Task 1.2.2: Cấu hình automated testing pipeline / Configure automated testing pipeline

- ☐ Thiết lập unit tests cho User Service với pytest / Setup unit tests for User Service with pytest
- ☐ Thiết lập unit tests cho Payment Service với pytest / Setup unit tests for Payment Service with pytest
- ☐ Thiết lập unit tests cho Math Solver Service với pytest / Setup unit tests for Math Solver Service with pytest
- ☐ Thiết lập unit tests cho Content Service với pytest / Setup unit tests for Content Service with pytest
- ☐ Thiết lập unit tests cho Admin Service với pytest / Setup unit tests for Admin Service with pytest
- ☐ Thiết lập integration tests cho API endpoints / Setup integration tests for API endpoints

- ☐ Cấu hình test coverage reporting với codecov / Configure test coverage reporting with codecov
- ☐ Thiết lập frontend tests với Jest và React Testing Library / Setup frontend tests with Jest and React Testing Library
- ☐  [BỔ SUNG] Cấu hình CI/CD để từ chối merge nếu test coverage giảm xuống dưới ngưỡng 90%


Task 1.2.3: Thiết lập Docker image building / Setup Docker image building

- ☐ Cấu hình build Docker image cho User Service → user-service:latest / Configure build Docker image for User Service → user-service:latest
- ☐ Cấu hình build Docker image cho Payment Service → payment-service:latest / Configure build Docker image for Payment Service → payment-service:latest
- ☐ Cấu hình build Docker image cho Math Solver Service → math-solver-service:latest / Configure build Docker image for Math Solver Service → math-solver-service:latest
- ☐ Cấu hình build Docker image cho Content Service → content-service:latest / Configure build Docker image for Content Service → content-service:latest
- ☐ Cấu hình build Docker image cho Admin Service → admin-service:latest / Configure build Docker image for Admin Service → admin-service:latest
- ☐ Cấu hình build Docker image cho Frontend → frontend:latest / Configure build Docker image for Frontend → frontend:latest
- ☐ Thiết lập image tagging với commit SHA / Setup image tagging with commit SHA

Task 1.2.4: Cấu hình container registry / Configure container registry

- ☐ Thiết lập Docker Hub registry cho public images / Setup Docker Hub registry for public images
- ☐ Tạo repositories: mathservice/user-service, mathservice/payment-service / Create repositories: mathservice/user-service, mathservice/payment-service
- ☐ Tạo repositories: mathservice/math-solver-service, mathservice/content-service / Create repositories: mathservice/math-solver-service, mathservice/content-service
- ☐ Tạo repositories: mathservice/admin-service, mathservice/frontend / Create repositories: mathservice/admin-service, mathservice/frontend
- ☐ Cấu hình authentication với Docker Hub tokens / Configure authentication with Docker Hub tokens
- ☐ Thiết lập automated image pushing sau successful tests / Setup automated image pushing after successful tests

Task 1.2.5: Thiết lập deployment scripts / Setup deployment scripts

- ☐ Tạo scripts/deploy-development.sh cho development environment / Create scripts/deploy-development.sh for development environment
- ☐ Tạo scripts/deploy-staging.sh cho staging environment / Create scripts/deploy-staging.sh for staging environment
- ☐ Tạo scripts/deploy-production.sh cho production environment / Create scripts/deploy-production.sh for production environment
- ☐ Cấu hình environment-specific variables trong deployment scripts / Configure environment-specific variables in deployment scripts
- ☐ Thiết lập health checks sau deployment / Setup health checks after deployment
- ☐ Cấu hình rollback mechanism / Configure rollback mechanism
- ☐  [BỔ SUNG] Tạo script rollback.sh để quay về phiên bản trước khi có lỗi

Task 1.2.6: Cấu hình environment-specific deployments / Configure environment-specific deployments

- ☐ Tạo k8s/development/ với Kubernetes manifests cho dev / Create k8s/development/ with Kubernetes manifests for dev
- ☐ Tạo k8s/staging/ với Kubernetes manifests cho staging / Create k8s/staging/ with Kubernetes manifests for staging
- ☐ Tạo k8s/production/ với Kubernetes manifests cho production / Create k8s/production/ with Kubernetes manifests for production
- ☐ Cấu hình different resource limits cho từng environment / Configure different resource limits for each environment
- ☐ Thiết lập different replica counts: dev(1), staging(2), production(3) / Setup different replica counts: dev(1), staging(2), production(3)

Task 1.2.7: Thiết lập security scanning / Setup security scanning


- ☐ Cấu hình Snyk security scanning cho dependencies / Configure Snyk security scanning for dependencies
- ☐ Thiết lập Docker image vulnerability scanning / Setup Docker image vulnerability scanning
- ☐ Cấu hình SAST (Static Application Security Testing) / Configure SAST (Static Application Security Testing)
- ☐ Thiết lập secret scanning để detect API keys, passwords / Setup secret scanning to detect API keys, passwords
- ☐ Cấu hình security alerts và notifications / Configure security alerts and notifications

Task 1.2.8: Tạo CI/CD documentation / Create CI/CD documentation

- ☐ Tạo docs/ci-cd-pipeline.md với workflow explanation / Create docs/ci-cd-pipeline.md with workflow explanation
- ☐ Tạo docs/deployment-guide.md với deployment procedures / Create docs/deployment-guide.md with deployment procedures
- ☐ Tạo docs/troubleshooting-ci-cd.md với common issues / Create docs/troubleshooting-ci-cd.md with common issues

Phase 1.3: API Gateway và Service Discovery / API Gateway and Service Discovery (0.5 tuần / 0.5 week)

Task 1.3.1: Cấu hình Traefik API Gateway / Configure Traefik API Gateway

- ☐ Tạo traefik/traefik.yml với cấu hình cơ bản / Create traefik/traefik.yml with basic configuration
- ☐ Cấu hình entrypoints: web (port 80), websecure (port 443) / Configure entrypoints: web (port 80), websecure (port 443)
- ☐ Thiết lập Docker provider cho service discovery / Setup Docker provider for service discovery
- ☐ Cấu hình dashboard trên port 8080 / Configure dashboard on port 8080
- ☐ Thêm Traefik service vào docker-compose.yml / Add Traefik service to docker-compose.yml
- ☐  [BỔ SUNG] Cấu hình middleware cho logging và tracing (ví dụ: Jaeger)

Task 1.3.2: Thiết lập routing rules cho từng service / Setup routing rules for each service

- ☐ Cấu hình route /api/users/* → User Service (port 8001) / Configure route /api/users/* → User Service (port 8001)
- ☐ Cấu hình route /api/payments/* → Payment Service (port 8002) / Configure route /api/payments/* → Payment Service (port 8002)
- ☐ Cấu hình route /api/math/* → Math Solver Service (port 8003) / Configure route /api/math/* → Math Solver Service (port 8003)
- ☐ Cấu hình route /api/content/* → Content Service (port 8004) / Configure route /api/content/* → Content Service (port 8004)
- ☐ Cấu hình route /api/admin/* → Admin Service (port 8005) / Configure route /api/admin/* → Admin Service (port 8005)
- ☐ Cấu hình route /* → Next.js Frontend (port 3000) / Configure route /* → Next.js Frontend (port 3000)

Task 1.3.3: Cấu hình load balancing / Configure load balancing

- ☐ Thiết lập round-robin load balancing cho User Service / Setup round-robin load balancing for User Service
- ☐ Thiết lập round-robin load balancing cho Payment Service / Setup round-robin load balancing for Payment Service
- ☐ Thiết lập round-robin load balancing cho Math Solver Service / Setup round-robin load balancing for Math Solver Service
- ☐ Thiết lập round-robin load balancing cho Content Service / Setup round-robin load balancing for Content Service
- ☐ Thiết lập round-robin load balancing cho Admin Service / Setup round-robin load balancing for Admin Service
- ☐ Cấu hình health checks cho load balancer / Configure health checks for load balancer

Task 1.3.4: Thiết lập SSL termination / Setup SSL termination

- ☐ Cấu hình Let's Encrypt ACME cho automatic SSL certificates / Configure Let's Encrypt ACME for automatic SSL certificates
- ☐ Thiết lập HTTP to HTTPS redirect / Setup HTTP to HTTPS redirect
- ☐ Cấu hình SSL certificates cho domain mathservice.com / Configure SSL certificates for domain mathservice.com
- ☐ Thiết lập SSL certificates cho subdomains: api.mathservice.com, admin.mathservice.com / Setup SSL certificates for subdomains: api.mathservice.com, admin.mathservice.com

Task 1.3.5: Cấu hình rate limiting / Configure rate limiting

- ☐ Thiết lập rate limiting 100 requests/minute cho User Service authentication endpoints / Setup rate limiting 100 requests/minute for User Service authentication endpoints
- ☐ Thiết lập rate limiting 50 requests/minute cho Payment Service transaction endpoints / Setup rate limiting 50 requests/minute for Payment Service transaction endpoints
- ☐ Thiết lập rate limiting 200 requests/minute cho Math Solver Service / Setup rate limiting 200 requests/minute for Math Solver Service
- ☐ Thiết lập rate limiting 500 requests/minute cho Content Service / Setup rate limiting 500 requests/minute for Content Service
- ☐ Thiết lập rate limiting 20 requests/minute cho Admin Service / Setup rate limiting 20 requests/minute for Admin Service

Task 1.3.6: Thiết lập service discovery / Setup service discovery

- ☐ Cấu hình Docker labels cho User Service: traefik.http.services.user-service.loadbalancer.server.port=8001 / Configure Docker labels for User Service: traefik.http.services.user-service.loadbalancer.server.port=8001

- ❑ Cấu hình Docker labels cho Payment Service: `traefik.http.services.payment-service.loadbalancer.server.port=8002` / Configure Docker labels for Payment Service: `traefik.http.services.payment-service.loadbalancer.server.port=8002`
- ❑ Cấu hình Docker labels cho Math Solver Service: `traefik.http.services.math-solver-service.loadbalancer.server.port=8003` / Configure Docker labels for Math Solver Service: `traefik.http.services.math-solver-service.loadbalancer.server.port=8003`
- ❑ Cấu hình Docker labels cho Content Service: `traefik.http.services.content-service.loadbalancer.server.port=8004` / Configure Docker labels for Content Service: `traefik.http.services.content-service.loadbalancer.server.port=8004`
- ❑ Cấu hình Docker labels cho Admin Service: `traefik.http.services.admin-service.loadbalancer.server.port=8005` / Configure Docker labels for Admin Service: `traefik.http.services.admin-service.loadbalancer.server.port=8005`
- ❑ Cấu hình Docker labels cho Frontend: `traefik.http.services.frontend.loadbalancer.server.port=3000` / Configure Docker labels for Frontend: `traefik.http.services.frontend.loadbalancer.server.port=3000`

Phase 1.4: Message Broker và Monitoring Setup / Message Broker and Monitoring Setup (0.5 tuần / 0.5 week)

Task 1.4.1: Cấu hình Redis cho message queuing / Configure Redis for message queuing

- ❑ Thiết lập Redis Pub/Sub cho User Service → Admin Service notifications / Setup Redis Pub/Sub for User Service → Admin Service notifications
- ❑ Thiết lập Redis Pub/Sub cho Payment Service → User Service balance updates / Setup Redis Pub/Sub for Payment Service → User Service balance updates
- ❑ Thiết lập Redis Pub/Sub cho Payment Service → Admin Service transaction alerts / Setup Redis Pub/Sub for Payment Service → Admin Service transaction alerts
- ❑ Cấu hình Redis channels: `user.created`, `user.updated`, `user.deleted` / Configure Redis channels: `user.created`, `user.updated`, `user.deleted`
- ❑ Cấu hình Redis channels: `payment.completed`, `payment.failed`, `balance.updated` / Configure Redis channels: `payment.completed`, `payment.failed`, `balance.updated`
- ❑ Cấu hình Redis channels: `math.solved`, `admin.alert` / Configure Redis channels: `math.solved`, `admin.alert`

Task 1.4.2: Thiết lập Prometheus cho metrics collection / Setup Prometheus for metrics collection

- ❑ Tạo `prometheus/prometheus.yml` configuration file / Create `prometheus/prometheus.yml` configuration file

- ☐ Cấu hình scrape targets cho User Service metrics endpoint /metrics / Configure scrape targets for User Service metrics endpoint /metrics
- ☐ Cấu hình scrape targets cho Payment Service metrics endpoint /metrics / Configure scrape targets for Payment Service metrics endpoint /metrics
- ☐ Cấu hình scrape targets cho Math Solver Service metrics endpoint /metrics / Configure scrape targets for Math Solver Service metrics endpoint /metrics
- ☐ Cấu hình scrape targets cho Content Service metrics endpoint /metrics / Configure scrape targets for Content Service metrics endpoint /metrics
- ☐ Cấu hình scrape targets cho Admin Service metrics endpoint /metrics / Configure scrape targets for Admin Service metrics endpoint /metrics
- ☐ Thêm Prometheus service vào docker-compose.yml (port 9090) / Add Prometheus service to docker-compose.yml (port 9090)
- ☐ Cấu hình retention period 30 days / Configure retention period 30 days

Task 1.4.3: Cấu hình Grafana dashboards / Configure Grafana dashboards

- ☐ Thêm Grafana service vào docker-compose.yml (port 3001) / Add Grafana service to docker-compose.yml (port 3001)
- ☐ Tạo grafana/provisioning/datasources/prometheus.yml / Create grafana/provisioning/datasources/prometheus.yml
- ☐ Tạo dashboard User Service Metrics với panels: active users, login rate, registration rate / Create dashboard User Service Metrics with panels: active users, login rate, registration rate
- ☐ Tạo dashboard Payment Service Metrics với panels: transaction volume, success rate, payment methods usage / Create dashboard Payment Service Metrics with panels: transaction volume, success rate, payment methods usage
- ☐ Tạo dashboard Math Solver Metrics với panels: problems solved, solution time, error rate / Create dashboard Math Solver Metrics with panels: problems solved, solution time, error rate
- ☐ Tạo dashboard System Overview với panels: CPU usage, memory usage, disk usage, network I/O / Create dashboard System Overview with panels: CPU usage, memory usage, disk usage, network I/O
- ☐ Tạo dashboard API Performance với panels: response time, request rate, error rate per endpoint / Create dashboard API Performance with panels: response time, request rate, error rate per endpoint

Task 1.4.4: Thiết lập basic alerting rules / Setup basic alerting rules

- ☐ Tạo prometheus/alerts/user-service.yml với rules: high login failure rate, user registration spike / Create prometheus/alerts/user-service.yml with rules: high login failure rate, user registration spike

- ☐ Tạo prometheus/alerts/payment-service.yml với rules: payment failure rate > 5%, transaction volume spike / Create prometheus/alerts/payment-service.yml with rules: payment failure rate > 5%, transaction volume spike
- ☐ Tạo prometheus/alerts/math-solver.yml với rules: solution time > 10s, error rate > 1% / Create prometheus/alerts/math-solver.yml with rules: solution time > 10s, error rate > 1%
- ☐ Tạo prometheus/alerts/system.yml với rules: CPU > 80%, memory > 90%, disk > 85% / Create prometheus/alerts/system.yml with rules: CPU > 80%, memory > 90%, disk > 85%
- ☐ Cấu hình Alertmanager với email notifications / Configure Alertmanager with email notifications
- ☐ Thiết lập Slack webhook cho critical alerts / Setup Slack webhook for critical alerts

Task 1.4.5: Cấu hình centralized logging / Configure centralized logging





- ☐ Thêm Elasticsearch service vào docker-compose.yml (port 9200) / Add Elasticsearch service to docker-compose.yml (port 9200)
- ☐ Thêm Logstash service vào docker-compose.yml (port 5044) / Add Logstash service to docker-compose.yml (port 5044)
- ☐ Thêm Kibana service vào docker-compose.yml (port 5601) / Add Kibana service to docker-compose.yml (port 5601)
- ☐ Cấu hình log shipping từ User Service → Logstash / Configure log shipping from User Service → Logstash
- ☐ Cấu hình log shipping từ Payment Service → Logstash / Configure log shipping from Payment Service → Logstash
- ☐ Cấu hình log shipping từ Math Solver Service → Logstash / Configure log shipping from Math Solver Service → Logstash
- ☐ Cấu hình log shipping từ Content Service → Logstash / Configure log shipping from Content Service → Logstash
- ☐ Cấu hình log shipping từ Admin Service → Logstash / Configure log shipping from Admin Service → Logstash
- ☐ Tạo Kibana index patterns cho từng service / Create Kibana index patterns for each service
- ☒ [BỔ SUNG] Cấu hình logstash.conf để parse log

Task 1.4.6: Tạo monitoring documentation / Create monitoring documentation

- ☐ Tạo docs/monitoring-setup.md với Prometheus configuration / Create docs/monitoring-setup.md with Prometheus configuration
- ☐ Tạo docs/grafana-dashboards.md với dashboard descriptions / Create docs/grafana-dashboards.md with dashboard descriptions

- ☐ Tạo docs/alerting-rules.md với alert conditions và responses / Create docs/alerting-rules.md with alert conditions and responses
 - ☐ Tạo docs/logging-guide.md với log formats và search queries / Create docs/logging-guide.md with log formats and search queries
-

Phase 2: User Service Development / Phát triển Dịch vụ Người dùng (4 tuần / 4 weeks)

- ☐  [BỔ SUNG MỚI] Task 2.0: Thiết kế và Tài liệu hóa
- ☐  Viết đặc tả API cho User Service bằng OpenAPI (hoàn thiện các Pydantic model và docstring)
- ☐  Viết Tài liệu Thiết kế Cấp cao cho luồng xác thực và quản lý người dùng
- ☐  Vẽ Sơ đồ Cơ sở dữ liệu (ERD) cho User Service

Phase 2.1: Core Authentication System / Hệ thống Xác thực Cốt lõi (1 tuần / 1 week)

Task 2.1.1: Thiết kế và implement User database models / Design and implement User database models

- ☐ Tạo bảng users với columns: id (UUID), email (unique), username (unique), password_hash, first_name, last_name / Create table users with columns: id (UUID), email (unique), username (unique), password_hash, first_name, last_name
- ☐ Thêm columns: phone_number, date_of_birth, is_active (boolean), is_verified (boolean), created_at, updated_at / Add columns: phone_number, date_of_birth, is_active (boolean), is_verified (boolean), created_at, updated_at
- ☐ Tạo bảng user_sessions với columns: id, user_id (FK), refresh_token_hash, expires_at, created_at, is_revoked / Create table user_sessions with columns: id, user_id (FK), refresh_token_hash, expires_at, created_at, is_revoked
- ☐ Tạo bảng user_balance với columns: id, user_id (FK), balance (decimal), currency (default 'VND'), updated_at / Create table user_balance with columns: id, user_id (FK), balance (decimal), currency (default 'VND'), updated_at
- ☐ Tạo bảng user_roles với columns: id, user_id (FK), role_name (enum: 'user', 'admin'), assigned_at, assigned_by / Create table user_roles with columns: id, user_id (FK), role_name (enum: 'user', 'admin'), assigned_at, assigned_by
- ☐ Thiết lập foreign key constraints và indexes / Setup foreign key constraints and indexes

Task 2.1.2: Tạo Alembic migrations cho User Service / Create Alembic migrations for User Service

- Initialize Alembic trong services/user-service/ / Initialize Alembic in services/user-service/
- Tạo migration 001_create_users_table.py / Create migration 001_create_users_table.py
- Tạo migration 002_create_user_sessions_table.py / Create migration 002_create_user_sessions_table.py
- Tạo migration 003_create_user_balance_table.py / Create migration 003_create_user_balance_table.py
- Tạo migration 004_create_user_roles_table.py / Create migration 004_create_user_roles_table.py
- Tạo migration 005_add_indexes_and_constraints.py / Create migration 005_add_indexes_and_constraints.py
- Test migrations với alembic upgrade head và alembic downgrade base / Test migrations with alembic upgrade head and alembic downgrade base

Task 2.1.3: Implement password hashing với bcrypt / Implement password hashing with bcrypt

- Tạo services/user-service/app/core/security.py / Create services/user-service/app/core/security.py
- Implement function hash_password(password: str) -> str sử dụng bcrypt với cost=12 / Implement function hash_password(password: str) -> str using bcrypt with cost=12
- Implement function verify_password(password: str, hashed: str) -> bool / Implement function verify_password(password: str, hashed: str) -> bool
- Implement function generate_salt() -> str / Implement function generate_salt() -> str
- Tạo password strength validation: minimum 8 chars, 1 uppercase, 1 lowercase, 1 digit, 1 special char / Create password strength validation: minimum 8 chars, 1 uppercase, 1 lowercase, 1 digit, 1 special char
- Write unit tests cho password functions trong tests/test_security.py / Write unit tests for password functions in tests/test_security.py

Task 2.1.4: Develop JWT token generation và validation / Develop JWT token generation and validation

- Implement function create_access_token(user_id: str, expires_delta: timedelta) -> str / Implement function create_access_token(user_id: str, expires_delta: timedelta) -> str
- Implement function create_refresh_token(user_id: str) -> str với expiry 30 days / Implement function create_refresh_token(user_id: str) -> str with expiry 30 days
- Implement function decode_token(token: str) -> dict với error handling / Implement function decode_token(token: str) -> dict with error handling

☐ Implement function `verify_token(token: str) -> bool` / Implement function `verify_token(token: str) -> bool`

☐ Cấu hình JWT settings: `ACCESS_TOKEN_EXPIRE_MINUTES=30`, `REFRESH_TOKEN_EXPIRE_DAYS=30` / Configure JWT settings: `ACCESS_TOKEN_EXPIRE_MINUTES=30`, `REFRESH_TOKEN_EXPIRE_DAYS=30`

☐ Implement JWT payload với claims: `user_id`, `email`, `role`, `exp`, `iat` / Implement JWT payload with claims: `user_id`, `email`, `role`, `exp`, `iat`

Task 2.1.5: Implement token refresh mechanism / Implement token refresh mechanism

☐ Tạo endpoint `POST /api/users/auth/refresh` / Create endpoint `POST /api/users/auth/refresh`

☐ Implement function `refresh_access_token(refresh_token: str) -> dict` / Implement function `refresh_access_token(refresh_token: str) -> dict`

☐ Implement refresh token validation và blacklisting / Implement refresh token validation and blacklisting

☐ Tạo database table `refresh_token_blacklist` / Create database table `refresh_token_blacklist`

☐ Implement automatic cleanup cho expired refresh tokens / Implement automatic cleanup for expired refresh tokens

Task 2.1.6: Tạo authentication middleware / Create authentication middleware

☐ Implement middleware `authenticate_user()` để verify JWT tokens / Implement middleware `authenticate_user()` to verify JWT tokens

☐ Implement middleware `require_role(role: str)` cho role-based access / Implement middleware `require_role(role: str)` for role-based access

☐ Implement middleware `rate_limit_auth()` cho authentication endpoints / Implement middleware `rate_limit_auth()` for authentication endpoints

☐ Tạo custom exceptions: `InvalidTokenError`, `ExpiredTokenError`, `InsufficientPermissionsError` / Create custom exceptions: `InvalidTokenError`, `ExpiredTokenError`, `InsufficientPermissionsError`

Phase 2.2: User Registration và Login Endpoints / User Registration and Login Endpoints (1 tuần / 1 week)

Task 2.2.1: Implement user registration endpoint / Implement user registration endpoint

☐ Tạo endpoint `POST /api/users/auth/register` / Create endpoint `POST /api/users/auth/register`

- ☐ Implement Pydantic models: UserRegistrationRequest, UserRegistrationResponse / Implement Pydantic models: UserRegistrationRequest, UserRegistrationResponse
- ☐ Implement email uniqueness validation / Implement email uniqueness validation
- ☐ Implement username uniqueness validation / Implement username uniqueness validation
- ☐ Implement password strength validation / Implement password strength validation
- ☐ Tạo user record trong database với hashed password / Create user record in database with hashed password
- ☐ Tạo initial user_balance record với balance=0 / Create initial user_balance record with balance=0
- ☐ Gán default role 'user' cho new users / Assign default role 'user' for new users
- ☐ Return JWT tokens (access + refresh) sau successful registration / Return JWT tokens (access + refresh) after successful registration

Task 2.2.2: Implement user login endpoint / Implement user login endpoint

- ☐ Tạo endpoint POST /api/users/auth/login / Create endpoint POST /api/users/auth/login
- ☐ Implement Pydantic models: UserLoginRequest, UserLoginResponse / Implement Pydantic models: UserLoginRequest, UserLoginResponse
- ☐ Implement login với email hoặc username / Implement login with email or username
- ☐ Verify password với bcrypt / Verify password with bcrypt
- ☐ Check user is_active status / Check user is_active status
- ☐ Generate và return JWT tokens / Generate and return JWT tokens
- ☐ Create user_session record / Create user_session record
- ☐ Implement login attempt tracking / Implement login attempt tracking
- ☐ Implement account lockout sau 5 failed attempts / Implement account lockout after 5 failed attempts

Task 2.2.3: Implement user logout endpoint / Implement user logout endpoint

- ☐ Tạo endpoint POST /api/users/auth/logout / Create endpoint POST /api/users/auth/logout
- ☐ Implement logout logic: invalidate current session / Implement logout logic: invalidate current session
- ☐ Add refresh token to blacklist / Add refresh token to blacklist
- ☐ Update user_session.is_revoked = true / Update user_session.is_revoked = true
- ☐ Return success response / Return success response

Task 2.2.4: Implement password reset functionality / Implement password reset functionality

- ☐ Tạo endpoint POST /api/users/auth/forgot-password / Create endpoint POST /api/users/auth/forgot-password
- ☐ Tạo endpoint POST /api/users/auth/reset-password / Create endpoint POST /api/users/auth/reset-password
- ☐ Generate secure password reset tokens / Generate secure password reset tokens
- ☐ Implement email sending cho password reset (mock implementation) / Implement email sending for password reset (mock implementation)
- ☐ Implement token expiration (15 minutes) / Implement token expiration (15 minutes)
- ☐ Validate reset token và update password / Validate reset token and update password
- ☐ 🟡 [BỔ SUNG] Task 2.2.5: User Profile Management Endpoints
- ☐ 🟡 Implement endpoint GET /api/users/me để lấy thông tin cá nhân
- ☐ 🟡 Implement endpoint PUT /api/users/me để cập nhật hồ sơ
- ☐ 🟡 Implement endpoint POST /api/users/me/change-password

Phase 2.3: User Profile Management / Quản lý Hồ sơ Người dùng (1 tuần / 1 week)

Task 2.3.1: Implement get user profile endpoint / Implement get user profile endpoint

- ☐ Tạo endpoint GET /api/users/me / Create endpoint GET /api/users/me
- ☐ Implement Pydantic model UserProfileResponse / Implement Pydantic model UserProfileResponse
- ☐ Return user information (exclude password_hash) / Return user information (exclude password_hash)
- ☐ Include user balance information / Include user balance information
- ☐ Include user roles / Include user roles
- ☐ Require authentication middleware / Require authentication middleware

Task 2.3.2: Implement update user profile endpoint / Implement update user profile endpoint

- ☐ Tạo endpoint PUT /api/users/me / Create endpoint PUT /api/users/me
- ☐ Implement Pydantic model UserProfileUpdateRequest / Implement Pydantic model UserProfileUpdateRequest
- ☐ Allow updates: first_name, last_name, phone_number, date_of_birth / Allow updates: first_name, last_name, phone_number, date_of_birth

- ☐ Validate phone_number format / Validate phone_number format
- ☐ Validate date_of_birth (must be 18+ years old) / Validate date_of_birth (must be 18+ years old)
- ☐ Update user record trong database / Update user record in database
- ☐ Return updated user profile / Return updated user profile

Task 2.3.3: Implement change password endpoint / Implement change password endpoint

- ☐ Tạo endpoint POST /api/users/me/change-password / Create endpoint POST /api/users/me/change-password
- ☐ Implement Pydantic model ChangePasswordRequest / Implement Pydantic model ChangePasswordRequest
- ☐ Require current_password verification / Require current_password verification
- ☐ Validate new_password strength / Validate new_password strength
- ☐ Hash new password và update database / Hash new password and update database
- ☐ Invalidate all existing sessions / Invalidate all existing sessions
- ☐ Return success response / Return success response

Task 2.3.4: Implement profile picture upload / Implement profile picture upload

- ☐ Tạo endpoint POST /api/users/me/avatar / Create endpoint POST /api/users/me/avatar
- ☐ Implement file upload validation (size, format) / Implement file upload validation (size, format)
- ☐ Implement image resizing và optimization / Implement image resizing and optimization
- ☐ Store image trong cloud storage hoặc local filesystem / Store image in cloud storage or local filesystem
- ☐ Update user profile với avatar URL / Update user profile with avatar URL
- ☐ Implement profile picture upload / Implement profile picture upload
- ☐ Tạo endpoint POST /api/users/me/change-password / Create endpoint POST /api/users/me/change-password
- ☐ Implement current password verification / Implement current password verification

Phase 2.4: User Balance và Transaction History / User Balance and Transaction History (1 tuần / 1 week)

Task 2.4.1: Implement balance management endpoints / Implement balance management endpoints

- Tạo endpoint GET /api/users/me/balance để get current balance / Create endpoint GET /api/users/me/balance to get current balance
- Implement balance update function update_user_balance(user_id, amount, transaction_type) / Implement balance update function update_user_balance(user_id, amount, transaction_type)
- Implement balance validation để prevent negative balance / Implement balance validation to prevent negative balance
- Tạo balance transaction logging / Create balance transaction logging
- Implement Redis caching cho balance queries / Implement Redis caching for balance queries

Task 2.4.2: Implement transaction history / Implement transaction history

- Tạo bảng user_transactions với columns: id, user_id, amount, transaction_type, description, created_at / Create table user_transactions with columns: id, user_id, amount, transaction_type, description, created_at
- Tạo endpoint GET /api/users/me/transactions để get transaction history / Create endpoint GET /api/users/me/transactions to get transaction history
- Implement pagination cho transaction history / Implement pagination for transaction history
- Implement filtering by date range và transaction type / Implement filtering by date range and transaction type
- Implement transaction export to CSV / Implement transaction export to CSV

Task 2.4.3: Implement user statistics / Implement user statistics

















- Tạo endpoint GET /api/users/me/stats để get user statistics / Create endpoint GET /api/users/me/stats to get user statistics
- Calculate total spent, total problems solved, account age / Calculate total spent, total problems solved, account age
- Implement monthly spending trends / Implement monthly spending trends
- Implement Redis caching cho statistics / Implement Redis caching for statistics

Task 2.4.4: Write comprehensive tests / Write comprehensive tests

- Write unit tests cho authentication functions / Write unit tests for authentication functions
- Write integration tests cho registration/login flow / Write integration tests for registration/login flow
- Write integration tests cho password reset flow / Write integration tests for password reset flow

- ☐ Write integration tests cho balance management / Write integration tests for balance management
 - ☐ Write performance tests cho high-load scenarios / Write performance tests for high-load scenarios
 - ☐ Achieve >90% test coverage / Achieve >90% test coverage
-

[BỔ SUNG MỚI] Phase 3: Payment Service

- ☐  Phát triển các tính năng liên quan đến nạp tiền, thanh toán và quản lý giao dịch.
- ☐  Task 3.0: Thiết kế và Tài liệu hóa
- ☐  Viết đặc tả API cho Payment Service bằng OpenAPI
- ☐  Viết Tài liệu Thiết kế Cấp cao, bao gồm sơ đồ luồng cho việc nạp tiền và xử lý thanh toán
- ☐  Vẽ Sơ đồ Cơ sở dữ liệu (ERD) cho Payment Service
- ☐  Task 3.1: Core Payment Logic
- ☐  Thiết kế và implement các model database: transactions, payment_methods
- ☐  Tạo Alembic migrations cho Payment Service
- ☐  Implement logic tạo giao dịch nạp tiền
- ☐  Implement logic trừ tiền khi người dùng sử dụng dịch vụ (giao tiếp với User Service)
- ☐  Task 3.2: Payment Gateway Integration
- ☐  Implement logic tích hợp với VNPAY (tạo request, xử lý callback)
- ☐  Implement logic tích hợp với Momo
- ☐  **Task 3.2.1: Manual Deposit via QR Code (Nạp tiền thủ công qua QR Code)**
- ☐  **Tạo VietQR Code generator cho từng user**
 - Generate QR với nội dung: "NAPTIENMATHPRO U[USER_ID]"
 - Lưu QR code vào database hoặc generate động
 - Endpoint: GET /api/payment/qr-code/{user_id}
- ☐  **Implement endpoint tạo QR Code động**
 - Tạo endpoint GET /api/payment/qr-code/{user_id}
 - Generate VietQR với bank account cố định
 - Nội dung chuyển khoản: "NAPTIENMATHPRO U{user_id}"
 - Return QR code image hoặc QR data string

❑ ● **Tạo bảng manual_deposits trong database**

- Columns: id, user_id, amount, bank_transaction_ref, status, created_at, processed_at, processed_by
- Index trên user_id và bank_transaction_ref
- Status: PENDING, COMPLETED, REJECTED

❑ ● **Implement Admin Manual Deposit Interface**

- Endpoint POST /api/admin/manual-deposit
- Admin input: user_id, amount, bank_transaction_ref, notes
- Tự động cộng tiền vào user_balance
- Ghi log transaction với type="MANUAL_DEPOSIT"
- Validation: kiểm tra user tồn tại, amount > 0

❑ ● **Tạo Bank Transaction Verification System**

- Endpoint GET /api/admin/pending-deposits
- Admin xem danh sách chuyển khoản cần xử lý
- Match transaction với user dựa trên nội dung chuyển khoản
- Hiển thị: user_id, amount, transaction_time, bank_ref

❑ ● **Implement Notification System cho Manual Deposit**

- Gửi thông báo cho user khi tiền được cộng thành công
- Email/SMS confirmation cho giao dịch nạp tiền
- Push notification nếu có mobile app
- Template: "Tài khoản của bạn đã được cộng {amount}đ"

❑ ● **Tạo Admin Audit Trail cho Manual Deposits**


- Log đầy đủ mọi thao tác manual deposit
- Ghi nhận: admin_id, action, user_id, amount, timestamp
- Endpoint GET /api/admin/deposit-audit-logs
- Export audit logs ra CSV/Excel

❑ ● **Implement Webhook endpoint để nhận trạng thái giao dịch từ các cổng thanh toán**




❑ ● **Implement cơ chế đối soát giao dịch định kỳ**

❑ ● **Task 3.3: Testing**

❑ ● **Viết Unit Test cho logic tính toán và xử lý giao dịch**


- ☐  Viết Integration Test cho luồng nạp tiền và thanh toán (sử dụng mock payment gateway)
-

Phase 4: Math Solver Service Development / Phát triển Dịch vụ Giải Toán (3 tuần / 3 weeks)

- ☐  [BỔ SUNG MỚI] Task 4.0: Thiết kế và Tài liệu hóa
- ☐  Viết đặc tả API cho Math Solver Service bằng OpenAPI
- ☐  Vẽ Sơ đồ Cơ sở dữ liệu (ERD) cho Math Solver Service

Phase 4.1: Core Math Solving Engine / Công cụ Giải Toán Cốt lõi (2 tuần / 2 weeks)

Task 4.1.1: Implement quadratic equation solver / Implement quadratic equation solver

- ☐  [BỔ SUNG] Implement API endpoint và logic cho "Giải phương trình bậc nhất"
- ☐ Tạo endpoint POST /api/math/quadratic / Create endpoint POST /api/math/quadratic
- ☐ Implement Pydantic models: QuadraticEquationRequest, QuadraticEquationResponse / Implement Pydantic models: QuadraticEquationRequest, QuadraticEquationResponse
- ☐ Implement quadratic formula: $ax^2 + bx + c = 0$ / Implement quadratic formula: $ax^2 + bx + c = 0$
- ☐ Handle special cases: $a=0$, $\text{discriminant} < 0$, $\text{discriminant} = 0$ / Handle special cases: $a=0$, $\text{discriminant} < 0$, $\text{discriminant} = 0$
- ☐ Return detailed solution steps / Return detailed solution steps
- ☐ Store solution trong database cho history / Store solution in database for history

Task 4.1.2: Implement system of linear equations solver / Implement system of linear equations solver

- ☐ Tạo endpoint POST /api/math/linear-system / Create endpoint POST /api/math/linear-system
- ☐ Implement Pydantic models: LinearSystemRequest, LinearSystemResponse / Implement Pydantic models: LinearSystemRequest, LinearSystemResponse
- ☐ Implement Gaussian elimination algorithm / Implement Gaussian elimination algorithm
- ☐ Handle 2x2, 3x3, và NxN systems / Handle 2x2, 3x3, and NxN systems
- ☐ Detect inconsistent và dependent systems / Detect inconsistent and dependent systems
- ☐ Return step-by-step solution process / Return step-by-step solution process

Task 4.1.3: Implement polynomial operations / Implement polynomial operations

- ☐ Tạo endpoint POST /api/math/polynomial/add / Create endpoint POST /api/math/polynomial/add
- ☐ Tạo endpoint POST /api/math/polynomial/multiply / Create endpoint POST /api/math/polynomial/multiply
- ☐ Tạo endpoint POST /api/math/polynomial/divide / Create endpoint POST /api/math/polynomial/divide
- ☐ Implement polynomial parsing từ string input / Implement polynomial parsing from string input
- ☐ Implement polynomial simplification / Implement polynomial simplification
- ☐ Return formatted polynomial output / Return formatted polynomial output

Task 4.1.4: Implement calculus operations / Implement calculus operations

- ☐ Tạo endpoint POST /api/math/derivative / Create endpoint POST /api/math/derivative
- ☐ Tạo endpoint POST /api/math/integral / Create endpoint POST /api/math/integral
- ☐ Implement symbolic differentiation rules / Implement symbolic differentiation rules
- ☐ Implement basic integration techniques / Implement basic integration techniques
- ☐ Handle trigonometric, exponential, logarithmic functions / Handle trigonometric, exponential, logarithmic functions
- ☐ Return step-by-step derivation / Return step-by-step derivation

Phase 4.2: Solution History và User Interaction / Solution History and User Interaction (1 tuần / 1 week)

Task 4.2.1: Implement solution history / Implement solution history

- ☐ Tạo bảng math_problems với columns: id, user_id, problem_type, input_data, solution_data, created_at / Create table math_problems with columns: id, user_id, problem_type, input_data, solution_data, created_at
- ☐ Tạo endpoint GET /api/math/history để get user's solution history / Create endpoint GET /api/math/history to get user's solution history
- ☐ Implement pagination và filtering by problem type / Implement pagination and filtering by problem type
- ☐ Implement solution bookmarking / Implement solution bookmarking
- ☐ Implement solution sharing với unique URLs / Implement solution sharing with unique URLs




Task 4.2.2: Implement solution export / Implement solution export

- ☐ Tạo endpoint GET /api/math/export/pdf để export solutions to PDF / Create endpoint GET /api/math/export/pdf to export solutions to PDF
- ☐ Implement LaTeX formatting cho mathematical expressions / Implement LaTeX formatting for mathematical expressions
- ☐ Generate PDF với proper mathematical notation / Generate PDF with proper mathematical notation
- ☐ Implement solution export to image format / Implement solution export to image format

Task 4.2.3: Write comprehensive tests / Write comprehensive tests

- ☐ Write unit tests cho math algorithms / Write unit tests for math algorithms
 - ☐ Write integration tests cho API endpoints / Write integration tests for API endpoints
 - ☐ Write performance tests cho complex calculations / Write performance tests for complex calculations
 - ☐ Test edge cases và error handling / Test edge cases and error handling
 - ☐ Achieve >95% test coverage cho math logic / Achieve >95% test coverage for math logic
-

Phase 5: Content và Admin Service Development / Content and Admin Service Development (2 tuần / 2 weeks)

- ☐  [BỔ SUNG MỚI] Task 5.0: Thiết kế và Tài liệu hóa
- ☐  Viết đặc tả API cho Content Service và Admin Service bằng OpenAPI
- ☐  Vẽ Sơ đồ Cơ sở dữ liệu (ERD) cho Content Service và Admin Service

Phase 5.1: Content Management System / Hệ thống Quản lý Nội dung (1 tuần / 1 week)

Task 5.1.1: Implement content models / Implement content models

- ☐ Tạo bảng pages với columns: id, slug, title, content, meta_description, is_published, created_at, updated_at / Create table pages with columns: id, slug, title, content, meta_description, is_published, created_at, updated_at
- ☐ Tạo bảng faqs với columns: id, question, answer, category, order, is_published / Create table faqs with columns: id, question, answer, category, order, is_published
- ☐ Tạo bảng content_categories với columns: id, name, description, parent_id / Create table content_categories with columns: id, name, description, parent_id
- ☐ Thiết lập relationships và indexes / Setup relationships and indexes

Task 5.1.2: Implement content CRUD endpoints / Implement content CRUD endpoints

- ☐ Tạo endpoint GET /api/content/pages để list all published pages / Create endpoint GET /api/content/pages to list all published pages
- ☐ Tạo endpoint GET /api/content/pages/{slug} để get specific page / Create endpoint GET /api/content/pages/{slug} to get specific page
- ☐ Tạo endpoint GET /api/content/faqs để get FAQ list / Create endpoint GET /api/content/faqs to get FAQ list
- ☐ Implement content caching với Redis / Implement content caching with Redis
- ☐ Implement content search functionality / Implement content search functionality

Task 5.1.3: Implement content versioning / Implement content versioning

- ☐ Tạo bảng content_versions để track changes / Create table content_versions to track changes
- ☐ Implement content revision history / Implement content revision history
- ☐ Implement content rollback functionality / Implement content rollback functionality

Phase 5.2: Admin Dashboard Backend / Admin Dashboard Backend (1 tuần / 1 week)

Task 5.2.1: Implement admin authentication / Implement admin authentication

- ☐ Tạo bảng admin_users với enhanced permissions / Create table admin_users with enhanced permissions
- ☐ Implement admin login với 2FA / Implement admin login with 2FA
- ☐ Implement role-based access control / Implement role-based access control
- ☐ Implement admin session management / Implement admin session management

Task 5.2.2: Implement admin content management / Implement admin content management

- ☐ Tạo endpoint POST /api/admin/content/pages để create pages / Create endpoint POST /api/admin/content/pages to create pages
- ☐ Tạo endpoint PUT /api/admin/content/pages/{id} để update pages / Create endpoint PUT /api/admin/content/pages/{id} to update pages
- ☐ Tạo endpoint DELETE /api/admin/content/pages/{id} để delete pages / Create endpoint DELETE /api/admin/content/pages/{id} to delete pages
- ☐ Implement bulk operations cho content management / Implement bulk operations for content management
- ☐ Implement content preview functionality / Implement content preview functionality

Task 5.2.3: Implement admin analytics / Implement admin analytics

- ☐ Tạo endpoint GET /api/admin/analytics/users để get user statistics / Create endpoint GET /api/admin/analytics/users to get user statistics
- ☐ Tạo endpoint GET /api/admin/analytics/transactions để get transaction statistics / Create endpoint GET /api/admin/analytics/transactions to get transaction statistics
- ☐ Tạo endpoint GET /api/admin/analytics/math-usage để get math service usage / Create endpoint GET /api/admin/analytics/math-usage to get math service usage
- ☐ Implement real-time dashboard data / Implement real-time dashboard data
- ☐ Implement data export functionality / Implement data export functionality

Task 5.2.4: Write admin service tests / Write admin service tests

- ☐ Write unit tests cho admin authentication / Write unit tests for admin authentication
 - ☐ Write integration tests cho content management / Write integration tests for content management
 - ☐ Write security tests cho admin endpoints / Write security tests for admin endpoints
 - ☐ Test admin permissions và access control / Test admin permissions and access control
-

Phase 6: Frontend Development / Phát triển Giao diện Người dùng (3 tuần / 3 weeks)

Phase 6.1: Nền tảng Giao diện & Layout chung / Foundation & Common Layout (1 tuần / 1 week)

- ☐ ● Task 6.1.1: Thiết lập và cấu hình API Client (API Layer)
- ☐ ● Cài đặt và cấu hình một instance axios (hoặc thư viện tương tự) làm client API chính
- ☐ ● Thiết lập base URL để trỏ đến địa chỉ của API Gateway (lấy từ biến môi trường)
- ☐ ● Cấu hình interceptor để tự động đính kèm JWT access_token vào header của các request
- ☐ ● Cấu hình interceptor để xử lý các lỗi API phổ biến (ví dụ: 401 Unauthorized - tự động refresh token)
- ☐ ● Task 6.1.2: Tạo Layout và Navigation chính
- ☐ ● Tạo component Header với logo "MathPro", menu navigation (Trang chủ, Dịch vụ, Giới thiệu, Liên hệ)
- ☐ ● Tạo component Footer với thông tin công ty, liên kết dịch vụ, thông tin liên hệ
- ☐ ● Implement responsive navigation menu cho mobile và desktop

- ☐ ● Tạo component Layout chung để wrap tất cả các trang
- ☐ ● Thiết lập routing cơ bản với Next.js App Router

Phase 6.2: Luồng Xác thực người dùng / Authentication Flow (1 tuần / 1 week)

- ☐ ● Task 6.2.1: Trang và Form Đăng nhập
- ☐ ● Tạo trang /login với form đăng nhập (email/username + password)
- ☐ ● Implement validation cho form đăng nhập
- ☐ ● Tích hợp với API POST /api/users/auth/login
- ☐ ● Xử lý lưu trữ JWT token vào localStorage hoặc httpOnly cookies
- ☐ ● Implement redirect sau khi đăng nhập thành công
- ☐ ● Task 6.2.2: Trang và Form Đăng ký
- ☐ ● Tạo trang /register với form đăng ký đầy đủ (email, username, password, confirm password, họ tên)
- ☐ ● Implement validation phía client (email format, password strength, confirm password match)
- ☐ ● Tích hợp với API POST /api/users/auth/register
- ☐ ● Hiển thị thông báo thành công và hướng dẫn xác thực email
- ☐ ● Task 6.2.3: Trang và Form Thay đổi mật khẩu
- ☐ ● Tạo trang /forgot-password với form nhập email
- ☐ ● Tạo trang /reset-password với form nhập mật khẩu mới
- ☐ ● Tạo component thay đổi mật khẩu trong trang profile (mật khẩu hiện tại + mật khẩu mới)
- ☐ ● Tích hợp với các API tương ứng cho forgot/reset password

Phase 6.3: Trang Dịch vụ Toán học / Core Service Pages (1 tuần / 1 week)

- ☐ ● Task 6.3.1: Trang Giải phương trình bậc 2
- ☐ ● Tạo trang /services/quadratic với form nhập hệ số a, b, c
- ☐ ● Implement validation để đảm bảo $a \neq 0$
- ☐ ● Tích hợp với API POST /api/math/quadratic
- ☐ ● Hiển thị kết quả với các bước giải chi tiết
- ☐ ● Implement tính năng "Lưu vào danh sách" và "Thêm vào danh sách"

- ☐ ● Task 6.3.2: Trang Giải hệ phương trình
- ☐ ● Tạo trang /services/linear-system với form nhập hệ phương trình (2 ẩn số, có thể mở rộng)
- ☐ ● Implement dynamic form để thêm/bớt phương trình
- ☐ ● Tích hợp với API POST /api/math/linear-system
- ☐ ● Hiển thị kết quả với ma trận và các bước giải
- ☐ ● Hiển thị lịch sử các hệ phương trình đã lưu

Phase 6.4: Trang Quản lý Người dùng / User Management Pages (1 tuần / 1 week)

- ☐ ● Task 6.4.1: Trang Hồ sơ cá nhân
- ☐ ● Tạo trang /profile với thông tin cá nhân (họ tên, email, số điện thoại, ngày sinh)
- ☐ ● Implement form chỉnh sửa thông tin cá nhân
- ☐ ● Tích hợp với API GET /api/users/me và PUT /api/users/me
- ☐ ● Hiển thị thống kê cá nhân (số bài toán đã giải, tổng chi tiêu, thời gian sử dụng)
- ☐ ● Task 6.4.2: Trang Quản lý Số dư & Lịch sử Giao dịch
- ☐ ● Tạo trang /balance với hiển thị số dư hiện tại
- ☐ ● Implement form nạp tiền với các mức tiền định sẵn (50k, 100k, 200k, 500k)
- ☐ ● Tích hợp với Payment Service để tạo giao dịch nạp tiền
- ☐ ● Hiển thị lịch sử giao dịch với phân trang
- ☐ ● Implement filter theo loại giao dịch và khoảng thời gian

● Bổ sung QR Payment UI vào Task 6.4.2:

- ☐ ● **Implement QR Code Payment Interface**
 - Hiển thị QR Code động với user_id
 - Form chọn số tiền: 50k, 100k, 200k, 500k, tùy ý
 - Button "Tạo QR Code" để generate mới
 - Hướng dẫn: "Quét mã QR bằng app ngân hàng"
 - Copy button cho nội dung chuyển khoản
- ☐ ● **Implement Payment Status Tracking**
 - Thông báo "Đang chờ xử lý..." sau khi chuyển khoản
 - Countdown timer: "Xử lý trong 5-30 phút"
 - Button "Liên hệ admin" nếu quá 30 phút

- Trạng thái: PENDING → PROCESSING → COMPLETED

☐ ● **Real-time Balance Update System**

- WebSocket connection cập nhật balance real-time
- Polling API mỗi 30 giây check balance changes
- Toast notification popup khi thành công
- Animation effect khi số dư thay đổi
- Sound notification (có thể tắt/bật)

☐ ● **QR Payment History & Receipt**

- Tab riêng "Lịch sử nạp QR" trong transaction history
- Hiển thị: QR đã tạo, số tiền, thời gian, trạng thái
- Button "Tạo lại QR" cho giao dịch failed
- Export receipt PDF cho giao dịch thành công
- Filter: Tất cả, Thành công, Đang chờ, Thất bại

Phase 6.5: Trang Quản trị / Admin Panel (1 tuần / 1 week)

☐ ● Task 6.5.1: Dashboard Quản trị

☐ ● Tạo trang /admin với dashboard tổng quan

☐ ● Hiển thị các thống kê chính: tổng người dùng, doanh thu, số bài toán được giải

☐ ● Implement các biểu đồ thống kê (doanh thu theo tháng, người dùng mới, sử dụng dịch vụ)

☐ ● Tích hợp với API GET /api/admin/analytics/*

☐ ● Task 6.5.2: Quản lý Người dùng

☐ ● Tạo trang /admin/users với danh sách người dùng

☐ ● Implement tìm kiếm và filter người dùng

☐ ● Implement các thao tác: xem chi tiết, khóa/mở khóa tài khoản, reset mật khẩu

☐ ● Hiển thị thông tin chi tiết người dùng và lịch sử hoạt động

☐ ● Task 6.5.3: Quản lý Giao dịch

☐ ● Tạo trang /admin/transactions với danh sách giao dịch

☐ ● Implement filter theo trạng thái, phương thức thanh toán, khoảng thời gian

☐ ● Implement xuất báo cáo Excel

☐ ● Hiển thị chi tiết giao dịch và khả năng hoàn tiền

☐ ● Task 6.5.4: Quản lý Nội dung Trang

- ☐ ● Tạo trang /admin/content với danh sách các trang nội dung
- ☐ ● Implement CRUD cho các trang: Trang chủ, Giới thiệu, FAQ
- ☐ ● Tạo rich text editor để chỉnh sửa nội dung
- ☐ ● Implement preview trang trước khi publish
- ☐ ● Tích hợp với Content Service APIs

Phase 6.6: Hoàn thiện và Kiểm thử Frontend / Frontend Finalization and Testing (1 tuần / 1 week)

- ☐ ● Task 6.6.1: Testing và Optimization
- ☐ ● Viết unit tests cho các components chính
- ☐ ● Viết integration tests cho các luồng quan trọng (đăng nhập, đăng ký, giải toán)
- ☐ ● Implement error boundaries và error handling
- ☐ ● Optimize performance (lazy loading, code splitting, image optimization)
- ☐ ● Test responsive design trên các thiết bị khác nhau
- ☐ ● Implement SEO optimization (meta tags, structured data)
- ☐ ● Test accessibility (WCAG guidelines)
- ☐ ● Perform cross-browser testing
- ☐ ● [BỔ SUNG MỚI] Task 6.1.1: Thiết lập và cấu hình API Client (API Layer)
- ☐ ● Cài đặt và cấu hình một instance axios (hoặc thư viện tương tự) làm client API chính
- ☐ ● Thiết lập base URL để trỏ đến địa chỉ của API Gateway (lấy từ biến môi trường)
- ☐ ● Cấu hình interceptor để tự động đính kèm JWT access_token vào header của các request
- ☐ ● Cấu hình interceptor để xử lý các lỗi API phổ biến (ví dụ: 401 Unauthorized - tự động refresh token)

Phase 7: Integration Testing và Deployment / Integration Testing and Deployment (1 tuần / 1 week)

Phase 7.1: End-to-End Testing / End-to-End Testing (0.5 tuần / 0.5 week)

Task 7.1.1: Setup E2E testing environment / Setup E2E testing environment

- ☐ Thiết lập Playwright hoặc Cypress cho E2E testing / Setup Playwright or Cypress for E2E testing

- ☐ Tạo test database và test environment / Create test database and test environment
- ☐ Implement test data seeding / Implement test data seeding
- ☐ Cấu hình CI/CD để run E2E tests / Configure CI/CD to run E2E tests

Task 7.1.2: Write critical user journey tests / Write critical user journey tests

- ☐ Test complete user registration → email verification → login flow / Test complete user registration → email verification → login flow
- ☐ Test math problem solving → payment → solution history flow / Test math problem solving → payment → solution history flow
- ☐ Test admin login → user management → content management flow / Test admin login → user management → content management flow
- ☐ Test payment flow với mock payment gateways / Test payment flow with mock payment gateways

Task 7.1.3: Performance và load testing / Performance and load testing

- ☐ Setup load testing với k6 hoặc Artillery / Setup load testing with k6 or Artillery
- ☐ Test API performance under load / Test API performance under load
- ☐ Test database performance với concurrent users / Test database performance with concurrent users
- ☐ Identify và fix performance bottlenecks / Identify and fix performance bottlenecks

Phase 7.2: Production Deployment / Production Deployment (0.5 tuần / 0.5 week)

Task 7.2.1: Production environment setup / Production environment setup

- ☐ Setup production Kubernetes cluster / Setup production Kubernetes cluster
- ☐ Configure production databases với backup strategies / Configure production databases with backup strategies
- ☐ Setup production Redis cluster / Setup production Redis cluster
- ☐ Configure production monitoring và alerting / Configure production monitoring and alerting

Task 7.2.2: Security hardening / Security hardening

- ☐ Implement rate limiting trên production / Implement rate limiting on production
- ☐ Configure WAF (Web Application Firewall) / Configure WAF (Web Application Firewall)
- ☐ Setup SSL certificates và HTTPS enforcement / Setup SSL certificates and HTTPS enforcement
- ☐ Implement security headers và CSP / Implement security headers and CSP

☐ Configure backup và disaster recovery / Configure backup and disaster recovery

Task 7.2.3: Go-live preparation / Go-live preparation

☐ Perform final security audit / Perform final security audit

☐ Setup production monitoring dashboards / Setup production monitoring dashboards

☐ Prepare rollback procedures / Prepare rollback procedures




☐ Train support team / Train support team

☐ Execute production deployment / Execute production deployment




☐ Perform post-deployment verification / Perform post-deployment verification

[BỔ SUNG MỚI] Ghi chú quan trọng:




Về Tài liệu hóa:

-  Mỗi service phải có đặc tả API đầy đủ bằng OpenAPI/Swagger
-  Tất cả quyết định kiến trúc quan trọng phải được ghi lại trong ADR (Architecture Decision Records)
-  Sơ đồ cơ sở dữ liệu (ERD) phải được tạo trước khi implement

Về Kiểm thử:

-  Mục tiêu test coverage: >90% cho logic nghiệp vụ quan trọng
-  CI/CD sẽ từ chối merge nếu test coverage giảm
-  Ưu tiên Test-Driven Development (TDD) cho các logic phức tạp

Về Chất lượng Code:

-  Sử dụng pre-commit hooks để thực thi code formatting và linting
 -  Tất cả code phải tuân thủ Black, isort, flake8, mypy
 -  Mọi endpoint phải có Pydantic model cho request/response
-

TỔNG KẾT BỔ SUNG QR PAYMENT:

- **Backend:** Task 3.2.1 với 7 sub-tasks
- **Frontend:** Task 6.4.2 bổ sung với 4 sub-tasks
- **Tổng cộng:** 11 tasks mới được bổ sung

- **Chi phí:** 0đ (hoàn toàn miễn phí)
 - **Thời gian triển khai:** 1-2 tuần
 - **Phù hợp:** Giai đoạn đầu startup (0-1000 users)
-

Tổng thời gian dự kiến: 18 tuần / 18 weeks

Ước tính nhân lực: 2-3 developers (1 Backend, 1 Frontend, 1 DevOps/Full-stack)

Milestone chính:

- Week 3: Infrastructure hoàn thành
- Week 7: User Service hoàn thành
- Week 10: Payment Service hoàn thành
- Week 13: Math Solver Service hoàn thành
- Week 15: Content & Admin Services hoàn thành
- Week 18: Frontend và Deployment hoàn thành