LoyalNest Full-Stack Development Study Plan (for university student - He Chengxun)

Objective

To systematically master the technical stack and architecture of LoyalNest during four years of university, enabling independent development, maintenance, and expansion of the system by graduation. By the end, the student should also be able to independently design, build, deploy, and launch a complete Shopify App (not necessarily LoyalNest) as a solo developer, while applying effective project management practices.

Learning Strategy

- **Theoretical**: Dedicate 2–6 hours/week to study books, video courses, or interactive tutorials (e.g., Linux basics, project management fundamentals), adjusting based on academic workload (e.g., 2 hours during exams, 6–8 during breaks).
- **Practical**: Use LoyalNest submodules and the solo Shopify App as practice projects, including unit, integration, and property-based tests, with Linux-based development environments.
- **Project-Based**: Complete one mini-project per semester, one collaborative open-source contribution, one hackathon participation, and one capstone Shopify App, managed with task estimation and user stories.
- **Engineering**: Learn Git, Docker, Kubernetes, VSCode, Swagger, CI/CD tools, automated testing (Jest, Playwright, Cypress), Linux system administration, and modern best practices.
- **Showcase**: Build a public GitHub portfolio, launch a Shopify App, publish a resume site, document feedback from mentors/peers, and share technical blog posts or demo videos.
- **Wellness**: Use time-blocking and Pomodoro techniques, incorporate breaks, celebrate small wins, join study groups, and schedule bi-monthly mentor check-ins to maintain motivation and avoid burnout.

Capstone Project: Independent RFM SaaS (To be built over 4 years)

Build a fully independent customer segmentation SaaS based on the RFM (Recency, Frequency, Monetary) model.

This app should

- Support CSV imports, Shopify/WooCommerce/API integration, and Webhooks/EventBridge for real-time events
- Compute RFM scores, visualize segments with heatmaps, charts, and PWAs for offline access
- Allow merchants to download segments, sync tags to email tools (Klaviyo, Mailchimp)
- Include user auth, tenant management, dashboard UI (PWA-enabled), and a scoring engine
- Dockerized, deployed on AWS ECS or Vercel, with Kubernetes basics for scalability
- Ensure GDPR-compliant security with audit logs and PII deletion

Core Modules of the RFM Application

- Data Import: CSV upload, Shopify API, REST API ingestion, Webhooks/EventBridge
- Data Modeling: Auto-detect customer ID, timestamp, amount; clean null/duplicates

- RFM Scoring Engine: Custom binning strategy, scoring rule editor, real-time processing with ClickHouse
- Visualization: RFM matrix, heatmaps, lifecycle pie charts, PWA support
- Customer Tags: Auto-generate tags by segments, support export and Shopify/CRM integration
- Auto Sync: Integrate with Klaviyo, Mailchimp, Shopify customer tag API
- Multi-Tenant Support: Tenant login and data isolation
- User Permissions: Role-based access (admin, analyst, etc.)
- Security: Rate limiting, input validation, OAuth token management

Capstone Milestones

- RFM Scoring CLI Tool (Year 3 Q2): Run on CSV orders via terminal, include property-based tests, test on Linux environment, create user stories and estimate tasks (e.g., using Trello)
- REST API Wrapper (Year 3 Q3): /api/rfm/score, /api/rfm/chart, secured with Helmet.js, document requirements and present to mentor
- Admin Dashboard (Year 4 Q1): RFM heatmap + customer segments UI, PWA-enabled, conduct risk assessment for UI development
- App Launch (Year 4 Q2): GitHub repo + docs + demo deployment on AWS ECS/Vercel via Linux-based CI/CD pipeline, with blog post or demo video, run retrospective on project

Layered Learning Modules (From Fundamentals to Advanced)

Fundamentals (Year 1)

- **Semester 1**:
  - JavaScript/TypeScript basics
  - HTML + CSS + DOM manipulation
  - Git + GitHub workflow
  - Node.js intro and runtime principles
  - Data structures & algorithms (arrays, linked lists, hash tables)
  - Time-blocking and Pomodoro techniques (Toggl, Notion)
- **Semester 2**:
  - React + Tailwind CSS
  - Arrays/linked lists/hash tables
  - Linux Basics: File system navigation (ls, cd, pwd), permissions (chmod, chown), package management (apt/yum), SSH, basic shell scripting (bash)
  - Deploy static site on a Linux server (e.g., Ubuntu WSL or DigitalOcean droplet)
  - Get peer feedback, find mentor via Shopify Dev Community

Tooling & Engineering Skills (Year 2)

- React + Vite + Tailwind CSS
- State management (useState, useEffect, Context)
- VSCode plugins, ESLint + Prettier config
- Docker / Docker Compose (run on Linux environment)
- AWS ECS or Vercel deployment with Terraform/CDK basics, using Linux for setup
- Swagger docs, Postman testing
- GitHub Actions / CI Pipeline basics (Linux-based runners)
- Task Estimation: Learn story points and sprint planning with Trello or Jira

Shopify Ecosystem (Year 2–3)

- **Shopify Basics**: App types, API scopes, Storefront API, Hydrogen
- **Admin App Development**: OAuth 2.0, App Bridge, Polaris UI
- **Embedded Integration**: Iframe embed, App Bridge v3
- **Shopify Functions**: Rust-based, bundle, API hooks
- **Billing System**: Subscriptions, one-time charges, GraphQL
- **Webhooks/EventBridge**: Real-time event handling for orders
- **App Review & Launch**: Checklist, PII checks, app listing

LoyalNest Core Modules (Year 2–3)

- NestJS framework (Controller, Service, Module)
- PostgreSQL + Prisma/SQLx operations (manage database on Linux)
- Redis: caching, queues, publish-subscribe (configure on Linux)
- Kafka for event-driven communication (microservices)
- ClickHouse for real-time RFM analytics
- Rust basics + backend logic (for Shopify Functions)
- User authentication (JWT), multi-tenant design, RBAC permissions
- Microservices: Auth, Points, Referrals, RFM, Admin Features
- Security: OWASP Top 10, secure API design with Helmet.js
- Team Collaboration: Facilitate discussions and resolve conflicts in study groups or open-source projects

System Iteration and Architecture Evolution Module (Year 3–4)

- **VIP Tiers System**: Tier strategy design, segmented permissions, data persistence
- **Multi-Store Points Sharing**: Multi-tenant architecture, Shopify Plus multi-store API, unified identity
- **Product-Level RFM**: Product data aggregation, scoring cache, Redis optimization
- **Gamification**: Leaderboard caching, Redis sorting, rules engine
- **Scalability**: Load balancing (AWS ELB), database indexing, caching strategies, Kubernetes basics (Minikube/EKS on Linux)
- **Serverless**: AWS Lambda, Vercel Functions for serverless deployment
- **AI Analytics**: ML models (PyTorch/Hugging Face) for predictive segmentation
- **Gray Releases**: Nginx routing, weight-based traffic splitting (configure on Linux)
- **GDPR Auto-Cleanup**: Audit logs, PII deletion, Webhook retry mechanism
- **Disaster Recovery**: pg_dump, Redis snapshots, RTO drills (execute on Linux)
- **A/B Testing**: User story mapping, A/B testing, feedback analysis
- **Penetration Testing**: OWASP ZAP or Burp Suite for vulnerability testing (run on Linux)
- **Web3 Basics**: Loyalty points smart contract (ethers.js/Solana Rust SDK)
- **Risk Management**: Identify and mitigate risks (e.g., technical debt, scope creep)
- **Retrospectives**: Conduct “What Went Well, What Could Be Improved” after projects

Four-Year Roadmap

Year 1

- **Semester 1**: JS/HTML/CSS, Git, basic data structures, join Shopify Dev Community, set up study group, learn time-blocking (Toggl/Notion).

- **Semester 2**: React, Tailwind, arrays/linked lists/hash tables, Linux basics (file navigation, permissions, SSH, scripting), deploy static site on Linux server, get peer feedback, find mentor via Shopify Dev Community.

Year 2

- **Semester 1**: TypeScript, NestJS, PostgreSQL, sorting/recursion/stacks/queues, write Jest unit tests, learn task estimation with Trello/Jira, mentor check-in.
- **Semester 2**: Redis, Kafka, Shopify Ecosystem intro (Storefront API, Hydrogen), Shopify App building, deploy on AWS ECS/Vercel with Terraform using Linux, contribute to open-source project, practice team collaboration and conflict resolution.

Year 3

- **Semester 1**: Shopify Functions, Webhooks/EventBridge, billing, RFM/VIP modules, CI/CD, property-based testing (fast-check), present mini-project to mentors with user stories and stakeholder communication, join Shopify hackathon.
- **Semester 2**: Behavior tracking, disaster recovery, permissions, high-traffic simulation with k6, Playwright/Cypress E2E tests, ClickHouse for real-time RFM, secure APIs (Helmet.js), publish blog post, manage Linux server for testing/deployment, conduct retrospective on mini-project.

Year 4

- **Semester 1**: Project management (Scrum, risk management), gray releases, performance optimization, serverless deployment, A/B testing, Kubernetes basics (Minikube/EKS on Linux), penetration testing (OWASP ZAP on Linux), PyTorch/Hugging Face ML model, Web3 loyalty points, PWA for Admin Dashboard.
- **Semester 2**: Complete app launch, analyze user feedback, build portfolio site, publish demo video/blog, document lessons learned, mentor check-in, run retrospective on capstone project.

Suggested Enhancements & Engineering Additions

Mini Projects Per Year

- Year 1: React component showcase, deployed on Linux server
- Year 2: Shopify admin panel clone (Hydrogen-based), planned with task estimation
- Year 3: RFM API microservice with Webhooks, documented with user stories
- Year 4: Full SaaS deployment + predictive analytics, with risk assessment and retrospective

Collaboration & Community

- Join open-source contribution (by Year 3)
- Participate in Shopify hackathon or app challenge (Year 3 or 4)
- Publish 1–2 technical blog posts or demo videos yearly (Year 3–4)
- Join Discord/Reddit Shopify or SaaS dev groups
- Bi-monthly mentor check-ins (Year 2–4) to discuss project progress and stakeholder feedback

Additional DevOps/Engineering Practices

- Add PostHog for event tracking

- Use Sentry or LogRocket for front-end error logging
- Try Unleash or Flagsmith for feature flags
- Use Terraform/CDK for cloud resource IaC on Linux
- Performance test with k6 on Linux
- Monitor with Grafana + Loki or Prometheus on Linux

Optional Tools

- OWASP ZAP or Burp Suite for penetration testing on Linux
- Workbox/Vite PWA for Admin Dashboard
- ethers.js or Solana Rust SDK for Web3 loyalty points
- Trello/Jira for task estimation and sprint planning

Final Graduation Capability Profile

- **Technical Skills**: Proficient in React, NestJS, Rust, PostgreSQL, Redis, Kafka, ClickHouse, serverless, Kubernetes, PyTorch/Hugging Face, Web3 basics, Linux system administration
- **Engineering Skills**: Can set up Dev environments (including Linux servers), write automated tests (unit, integration, E2E, property-based), deploy on AWS ECS/Vercel, recover systems, secure APIs, and document processes
- **Project Management**: Proficient in Agile/Scrum, task estimation, stakeholder communication, risk management, team collaboration, conflict resolution, and retrospectives; uses GitHub Projects, Trello/Jira for workflows
- **Product Thinking**: Can collect user feedback, conduct A/B testing, define priorities, and iterate versions
- **System Thinking**: Understands LoyalNest architecture, scalability, security, and service decomposition

Books

Year 1

- Semester 1:《You Don't Know JavaScript》(Vol. 1),《Eloquent JavaScript》
- Semester 2:《You Don't Know JavaScript》(Vol. 2),《Learning TypeScript》(Josh Goldberg),《Linux Basics for Hackers》(OccupyTheWeb) or equivalent online tutorial (e.g., Linux Journey)

Year 2

- Semester 1:《Node.js in Action》,《Agile Estimating and Planning》(Mike Cohn, lightweight introduction to task estimation)
- Semester 2:《NestJS Definitive Guide》,《PostgreSQL in Action》

Year 3

- Semester 1:《The Rust Programming Language》,《Shopify App Development》(Gavin Ballard or equivalent)
- Semester 2:《DevOps Handbook》

Year 4

- Semester 1:《Designing Data-Intensive Applications》(Martin Kleppmann),《The Art of Project Management》(Scott Berkun, for risk management and retrospectives)