NerdBlock – Deliverable 2

Durham College, INFT-3201, Ghafouri Bakhsh, Fara

Ahmed Naveed, Alexzander Saddler, Muhammad Yasir Patel, Valentine Sah, Graeme Jeffreys

***EXECUTIVE SUMMARY***

NerdBlock’s subscriber base has jumped from a few hundred to over 20,000, which really means they need a bigger and more reliable system to handle subscriptions. This report explains our proposed solution—a strong database and application—that helps run daily tasks, makes customer interactions easier, and takes care of overstock transfers to Joe Nerds in a smoother way.

By putting our plan in place, NerdBlock should see better efficiency, less time wasted on manual tasks, and happier customers. We will deliver a normalized 3NF database, an easy-to-use interface, automated reports, plus secure connections with the stuff they already have. This document gives both the technical parts and some bigger-picture ideas so NerdBlock can keep growing and reach their future business goals.

***INTRODUCTION***

We are designing a database system for NerdBlock’s subscription service. NerdBlock is a fast growing, Canadian business where customers can subscribe to genres, and each month they will receive a box of collectables relating to the genres they picked. Any overstock is distributed to NerdBlock’s brick-and-mortar stores called Joe Nerds. They weren’t expecting a large influx of subscribers and didn’t invest in much infrastructure. Now they need to accommodate over 20,000 subscribers. Now they have reached out to us for assistance in supplying a system that can accommodate said influx of new subscribers.

***MISSION STATEMENT***

“Deliver a scalable, end-to-end subscription management solution that boosts NerdBlock’s global growth, streamlines operations, and enriches customer experiences—ensuring efficiency, reliability, and lasting business impact.”

NerdBlock’s subscribers grew much faster than we thought, so now we urgently need a stronger and more flexible system. Our goal is to remove inefficiencies and data slowdowns, so NerdBlock can keep making customers happy without any service breaks or performance issues.

We want to give a strong, future-proof database solution that can grow smoothly, making sure data is always correct and easy to access, even if subscribers and orders keep growing in big numbers. We remain committed to using the best practices in database design, data security, and user experience. We’ll make sure the final system meets NerdBlock’s everyday operations and long-term goals. Our goal is to provide a dependable way for fans to engage with their preferred genres and letting NerdBlock process monthly orders with ease and manage overstock transfers smoothly.

***INITIAL PROJECT PLAN***

Our initial plan for the project is to gather required details from the synopsis and the client then design the scope of the project, and database. These details include external sources affecting the system and attributes the client wants to see in the tables. Then we will design diagrams around the system, such as an ERD (entity relationship diagram). To design an ERD, we need to understand the scope of the entire system and what it will do. Requiring us to design a System Diagram before more technical diagrams.

Here is a roadmap of the plan of our project for what shall be delivered and when it shall be delivered to our employers.

**Deliverable #1: Initial Planning**

* **Deadline**: January 24, 2025
* **Contents**:
  + **Group Charter Document**: Define roles and responsibilities.
  + **Draft Report**:
    - Mission Statement.
    - "Early Days" Plan (brief summary of initial steps).
    - System Boundary Diagram (high-level depiction of the business case).
    - Initial Project Plan:
      * Tasks, deadlines, dependencies, and assigned resources.

**Deliverable #2: Requirements, 3NF Schema, & Draft UI Examples**

* **Deadline**: February 21, 2025
* **Contents**:
  + **Updated Deliverable #1**: Refine based on feedback.
  + **Functional & Non-Functional Requirements**:
    - Define system functionalities and constraints.
  + **Draft 3NF Relational Schema**:
    - Normalize database design to eliminate redundancy.
  + **Entity-Relationship Diagram (ERD)**:
    - Show entities, attributes, and relationships.
  + **Application Design Drafts**:
    - Visuals for navigation, forms, and reports.
  + **Updated Project Management Details**:
    - Adjust task timelines, deadlines, and milestones.

**Deliverable #3: Final Proposal Report & Presentation**

* **Deadline**: March 28, 2025
* **Contents**:
  + **Finalized Report**:
    - Updated Deliverables #1 & #2.
    - Refined 3NF schema, ERD, and application design.
  + **Business Pitch Presentation**:
    - 10-minute non-technical presentation for the client.
    - Focus on solution benefits and visuals.
  + **Prototype Demonstration**:
    - Show core functionality (e.g., subscription management, reporting).
    - Demonstrate data committing/updating and reporting features.

**Deliverable #4: Technical Presentation, Database, & Application**

* **Deadline**: April 3, 2025
* **Contents**:
  + **Final Database & Application**:
    - Submit the complete database and functioning application.
    - Provide an active URL (if applicable).
  + **Technical Presentation**:
    - Demonstrate database and application features.
    - Explain technical design and how the system meets requirements.
  + **Final Project Management Report**:
    - Summarize completed tasks, responsibilities, and challenges.

**Dependency Table**

Below is a quick reference showing which tasks depend on each other. This helps clarify the order in which deliverables must be completed:

|  |  |
| --- | --- |
| **Task** | **Depends On** |
| System Boundary Diagram | Project Scope Definition (Client & Team Input) |
| ERD | System Boundary Diagram |
| 3NF Schema | ERD |
| UI Drafts | 3NF Schema (to know final table/field structure) |
| Implementation (Coding) | 3NF Schema + Finalized UI Drafts |
| Testing (Integration) | Partial or Full Implementation |
| Final Presentation | Completed Testing & Final Implementation |

A Gantt chart will be added visualize these dependencies and dates in further deliverables.

**Contingency Plan**

* Because on-time feedback and smooth teamwork is super important, we set up these precautions:
  + If client or stakeholder feedback is delayed, we’ll keep working on tasks that dont depend on it and then regroup fast once feedback arrives.
  + We added some extra time in our projected deadlines and do weekly check-ins, so we can fix roadblocks early on.
  + Team Availability: Each big task has a backup team member who can jump in if the main person isnt available.

**Testing Phases**

* We include testing in every part of our plan, so the final product is stable:
  + Unit Testing: We check single modules (like database triggers or UI parts) on their own.
  + Integration Testing: Once the database and UI connect, we test if data flows right from start to end.
  + User Acceptance Testing (UAT): Near the end, a small group of real users or stakeholders will test the almost-finished system to see if it meets usability and features they want before we deploy.

***DEADLINES***

All of the deadlines are also available through our project management tool (Trello).

The deadlines can also be observed through our initial project plan.

***ASSIGNED RESOURCES***

|  |  |  |  |
| --- | --- | --- | --- |
| TASK | ASSIGNED RESOURCE | REQUIRED TOOLS | TIME ALLOCATION |
| Set Up Project Management Tool | Alexzander | Trello | 1 hour |
| Mission Statement | Muhammad Yasir Patel | Microsoft Word | 1 hour |
| Early Days Plan | Alexzander | Microsoft Word | 1 hour |
| System Boundary Diagram | All team members. | Visio | 4 hours |
| Initial Project Plan | All team members. | Microsoft Word | 3 hours |

***SYSTEM BOUNDARY DIAGRAM***

The System Boundary Diagram shows a broad look at how different parts of NerdBlock’s subscription service, including how they work with outside systems and entities.

**Subscription Management:** Manages customer subscriptions, genres, and accounts. It get updates from the Client Website and tells Inventory what items are needed.

**Inventory Management:** Maintains item stock levels, product details, and does overstock transfers. It also get restocks from the Supplier System and can send extra items to the Retail Store System.

**Order Fulfillment:** Creates monthly shipments based on subscription info and current inventory, then follows them till delivery.

**Reporting & Analytics:** Collects data from all modules to offer insights on customer behaviors and inventory trends, and it shares key reports with the Admin System.

External entities—**Client Website**, **Database Access App**, **Supplier System**, **Retail Store System**, and **Admin System**—all link with these core modules to support NerdBlock’s daily workflows.



***RECOMMENDATIONS***

**Scalability & Growth**  
We recommend putting the database and app on a cloud environment like AWS or Azure so we can handle tons of subscribers if they come all at once. Another thing is having a load balancer in front of the app servers, so it wont slow down or crash when traffic is high. We should also add indexes on fields like CustomerID or SubscriptionID, and maybe split up large tables by date or region, so queries run faster.

**Security & Data Protection**  
Its important to use encryption for data at rest (like Transparent Data Encryption) and while it travels (HTTPS or TLS). We also want role-based access control, so staff can only see what they need for their jobs, which can block people from messing with data they shouldn’t see. Doing security checks every few months helps make sure nobody changed permissions or that we don’t have holes in the settings.

**Automation & Efficiency**  
We can set up alerts that tell us when inventory is too low or if monthly subscription cancels get too high, so we can fix problems faster. We can also use CI/CD (maybe GitHub Actions, Jenkins, or Azure DevOps) to test and deploy automatically, so we make fewer mistakes by hand. Another good idea is to plan maintenance windows to do re-indexing, log backups, and updates so we don’t have big downtimes.

***TESTING STRATEGIES***

**Performance Testing**

* **Load & Stress Tests:** We plan to use tools like JMeter or Locust so we can simulate heavy traffic and see if the app stays fast when lots of people are using it.
* **Scalability Verification:** We’ll slowly raise the user load in a test (staging) environment to make sure the system can handle growth without slowing down.

**Reliability Testing**

* **Failover & Disaster Recovery:** If we use AWS or another cloud, we’ll test automatic failover, and if it’s on-premise, we’ll do mirrored database or replication drills to see if switching over works right.
* **Backup & Restore Procedures:** We’ll do monthly restore tests from backups just to be sure our data can be recovered quickly and everything is still intact if something goes wrong.

**Security Testing**

* **Penetration Testing:** We might hire a security expert or use automated pentest tools to check for things like SQL injection or cross-site scripting holes.
* **Vulnerability Scanning:** We’ll schedule regular scans on servers, apps, and networks to catch new issues.
* **Access Control Review:** We’ll also verify each role (like Admin or Employee) only has the rights they really need, so we dont risk people seeing stuff they shouldn’t.

***BUSINESS IMPACT***

**Enhanced Efficiency & Scalability**  
With a clearer boundary in the system, the new design cuts down on daily friction by making each module’s job more obvious. Subscription and inventory info flow smoothly to fulfillment, so you don’t have to watch everything by hand or worry about errors, even if you suddenly get a lot more subscribers.

**Informed Decision-Making & Growth**  
Putting all data in one spot means better reporting and analytics, so NerdBlock can see subscription trends, inventory movement, and different region needs in real time. Managers can shift strategies quicker, order stock more wisely, and plan for bigger expansions without guessing.

**Better Customer Experiences**  
By seeing how every step fits together (from signing up all the way to getting monthly boxes), NerdBlock can make shipping faster, avoid running out of items, and send extra stuff to Joe Nerds more quickly. This leads to happier customers who are more likely to stay subscribed.

**Long-Term Competitive Edge**  
A well-structured system doesn’t just handle todays rush—it sets NerdBlock up to grow more, whether local or abroad. This solid base also lets them link up with new partners, add more automation, and keep making data-driven changes that improve the whole subscription process.

***Functional & Non-functional Requirements***

**Functional Requirements**

|  |  |
| --- | --- |
| User Registration & Login | * What authentication methods should be supported (e.g., email/password, social login, two-factor authentication)? * Should there be different access levels for different user roles (e.g., admin, regular user)? * Is their a specific age limit to login? * What kind of username and password should I put in? * What personal details should users provide during registration? |
| Subscription Management | * What are the key subscription tiers, and how flexible should they be? * Should users receive reminders before an auto-renewal occurs? * What policies should be in place for refunds or subscription pauses? |
| Product Catalog | * How frequently does the product catalog need to be updated? * When does it update? * How often it will give updates? * Is once a week or once a month? |
| Inventory | * What inventory tracking system should be implemented to prevent overstock or shortages? * How should transfers to Joe Nerds be handled in case of overstock? |
| Payment Processing | * What payment methods should be prioritized (credit card, PayPal, cryptocurrency, etc.)? * How should users be notified of payment failures or billing issues? * Should users be able to save their payment details for future transactions? * What security measures should be in place to protect payment information? |
| Order Tracking | * What order tracking system should be integrated for real-time updates? * Will the order of the tracking system work in a good efficient way? * Should users receive real-time shipping updates via email or SMS? * Should there be an estimated delivery date displayed for each order? |
| Personalization | * How should users specify their preferences for themes or fandoms? * Should there be AI-based recommendations for subscription boxes? |
| Customer Support | * What customer support channels should be available (live chat, email, phone, chatbot)? |
| Admin Dashboard | * What key metrics should be available to admins for managing inventory and subscriptions? |
| Notifications | * How will users receive notifications from Nerdblock? * Will it be by receiving SMS messages? |
| Reviews & Ratings | * Should admins have the ability to moderate or remove inappropriate reviews? * Should users be able to edit their reviews after submission? |
| Multi-Subscription & Account Linking | * Can single user account run multiple active subscriptions at the same time. * Should users all be linked under one profile so it’s easier to manage everything? |
| Reports & Analytics | * What types of reports are most critical for business decision-making? * How often should reports be generated, and in what formats? * Should analytics be integrated with third-party tools (e.g., Google Analytics, Tableau)? |

**Non-Functional Requirements**

|  |  |
| --- | --- |
| Performance | * What is the expected maximum number of users the platform should support simultaneously? * How quickly should pages load under normal and peak traffic conditions? * Should there be performance monitoring tools in place (e.g., New Relic, Datadog)? |
| Security | * What level of encryption should be used for user data and payments? * Should there be multi-factor authentication (MFA) for user accounts? * What compliance standards must be met (GDPR, PCI-DSS, etc.)? * How should data breaches or security incidents be handled? |
| Scalability | * Should the system be able to handle a sudden spike in subscriptions (e.g., Black Friday sales)? * How should the platform scale when new product categories or services are added? * Should there be a cloud-based or on-premise hosting solution? |
| Usability | * What level of accessibility compliance should the platform meet (WCAG 2.1, ADA)? * Should there be a mobile-friendly design or a dedicated mobile app? * How should the onboarding process be designed for new users? |
| Availability | * What is the minimum uptime requirement (e.g., 99.9%)? * Should there be a backup server or disaster recovery plan in place? * How should system downtime or maintenance windows be communicated to users? |
| Compliance | * What legal and e-commerce regulations must the platform follow? * Should there be an age restriction for certain products or subscriptions? * How should user data retention and deletion policies be handled? |
| Compatibility | * What browsers and devices should the platform support (Chrome, Safari, mobile, tablet, etc.)? * Should there be integrations with third-party tools (e.g., accounting, CRM, email marketing platforms)? |
| Maintainability | * How frequently should the system receive updates and patches? * Should there be automated testing to detect bugs before deployment? * Should there be a staging environment for testing before launching new features? |

***Functional Dependencies List***

Suppliers:

* Supplier ID -> {Name, Street Address, Postal Code, Phone Number, Email Address}
* {Street Address, Postal Code} -> Postal Code

Products:

* ProductID -> {Name, Description, Price}

Inventory:

* InventoryID -> {ProductID, Product Name, Quantity}
* ProductID -> Product Name

Overstock:

* OverstockID -> {Quantity, ProductID, StoreID, Transfer Sorting Key}
* Transfer Sorting Key -> {ProductID, StoreID}

Employees:

* EmployeeID -> {StoreID, First Name, Last Name, Phone Number, Email Address, Date Joined, IsAdmin, IsStoreOwner}

Reports:

* ReportID -> {ReportName, ReportDescription, CreatedDate, CreatedBy, ReportScope, ReportFormat}
* ReportName -> {ReportID, ReportDescription, CreatedDate, CreatedBy, ReportScope, ReportFormat}

Inventory:

* InventoryID -> {ProductID, Product Name, Quantity}
* ProductID -> Product Name

Shipping Report:

* ShippingID -> {TransactionID, SubscriberID, Date, Shipping Duration}
* TransactionID -> SubscriberID

Subscriber:

* SubscriberID -> {CountryID, First Name, Last Name, Phone Number, Email Address, GenreID, Shipping Address, Date Created}
* CountryID -> Shipping Address

Country:

* CountryID -> {Country Name, Tax, Additional Fees}
* Country Name -> {CountryID, Tax, Additional Fees}

Transfer Report:

* TransferReportID -> {Date, Transfer Sorting Key}

Stores:

* StoreID -> {City, Street Address, Postal code, Phone Number}
* Street Address -> {City, Postal code}

Orders:

* OrderID -> {EmployeeID, ProductID, SupplierID, Quantity}

Genres:

* GenreID -> {Name, Description}
* Name -> {GenreID, Description}

Transfer:

* TranserID -> {Sorting key, ProductID, StoreID, Quantity}
* Sorting Key -> {ProductID, StoreID}

Boxes:

* BoxID -> {Sorting key, ProductID, StoreID, Price}

Transactions:

* TransactionID -> {box Sorting Key, SubscriberID, total Price, date }

Subscription:

* SubscriptionID -> {GenreID, Type, Price}

***Draft 3NF in Relational Schema format***

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A close-up of a list

Description automatically generated

***Entity Relationship Diagram matching 3NF***

A screenshot of a computer

Description automatically generated

***Application Design Drafts***

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

***Data Dictionary***

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

