Store Management System

Developer Guide

Adal Khan

April 23, 2025

[A01799729507@gmail.com](mailto:A01799729507@gmail.com)

Index

|  |  |  |
| --- | --- | --- |
| Chapter | Description | Page |
| 01 | Software Requirements | 03 |
| 02 | Data Model | 04 |
| 03 | Data Flow | 05 |
| 04 | Application Logic | 06 |

|  |  |
| --- | --- |
| C-01 | Software Requirements |

Front End: HTML, CSS, Typescript, Svelte & Svelte Kit

Backend: Rust, Axum

Database: MySQL.

Testing: JEST & Build-in Rust Unit Testing Library

The store management system is a database-intensive web application that must handle heavy concurrent requests and provide easy access to data. To achieve optimal performance, Rust is used along with Axum, a framework well-suited for managing high traffic with minimal overhead. BIAM has its own server. Rust is extremely fast and uses minimal resources, which is why we chose it for the backend.

|  |  |
| --- | --- |
| C-02 | Data Model |

Table Name: Products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SL | Produc\_UID | Product\_TYPE | Product\_NAME | PHOTO |
| Primary\_Key | Unique, Integer  Code Format  Product\_Type Code + Serial | * Assets - 10 * Medicine - 11 * Stationary – 12 * Electric-13 * Electronics-14 * Accessories-15 * Cloths-16 * Furniture-17 * Administrative\_Supply-18 * IT-19 * Other-20 etc * product name is dynamic and user should be able to add product name based on Product Type | Name of the Product as string | Image link as string. this column is optional and may contains null value and can be updated later |

Table Name: users (login system for users)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | EID | email | password | Name (Full) | Phone | DoB | NID | Desgination | Status (Active/inactive) | Department | Grade |  | Other |
| PKey | Unique, Integer, | Unique | hased | String | Number | Date | number | String | String (default inactive) | String | Number |  | json – store extra user info as json format |

During the authentication process, we will verify user information from this table. A newly created user must be approved by a system administrator and assigned a designation. Based on the assigned designation, the user will be able to perform specific tasks. Note that user’s activity must be recorded in other section as json data. in any suspicious activity is detected shutdown or block the account for a while.

This table is used for a separate login system exclusively for administrators. It must be secure by default, including features such as a hidden login URL.

There will be a one-time setup URL, which can only be used once. When accessed, this setup process will allow the creation of the Super Admin account.

Table Name: System

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | username | password | phone | email | role | auth\_info |
| PKey | string | string | Number | string, Unique | string (sadmin, store\_keeper, adadmin, ddadmin…) | json |

Table Name: Activity\_log (Tracking activity per users and only admin can see this)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | TaskName | Initated\_by | Date | other\_info |
| Pkey | New product added | md@sazib | 12/12/25 | json |

Table Name: Ttracker (Transection Tracker)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Transection ID | Transection Type | Description/ File No. | Date and Time | UserID | Other |
| PKey | String (Unique Ramdon hex code) | Entry, Dispatch, Auction | String | Transection time and Date | initiated by | JSON contains all information about entry or dispatch |

Table Name: Requision\_map

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Initiated\_by | department | Designation | authorizing\_officer | Status | Dispatch\_Code |
| PKey | String |  |  |  | onQueue/  Approved/  Dispatch | Random Code shown to Client |
|  |  |  |  |  |  |  |

Table Name: My\_Activity (This table is created once when a new user is approved by the admin, this keep track all of transection of a particular user: username, phone and info as json)

Table Name: App

|  |  |  |
| --- | --- | --- |
| id | key | value |
| 1 | app\_initialized | 0/1 |

Summary:

|  |  |  |
| --- | --- | --- |
| SL | Table Name | Action |
| 01 | Products | Store production information & Types |
| 02 | Users | Store user information including login info & Personal details |
| 03 | System | Store administrative & Setup information |
| 04 | App | Global App settings |
| 05 | Activity | Store user activities. Accessible to **admin only.** |
| 06 | Transection | Store User Transection details. Dispatch, Entry, Auction ect |
| 07 | Requisition | Initiate a requisition and store its flow. |
| 08 | Map | Map for Authorizing officers |

|  |  |
| --- | --- |
| C-04 | Application Logic |

------------------------------------------------ Administrator -----------------------------------------------------------------------

fetch (GET) <https://bsms.io/admin/init>

1. if DB: App.key.app\_iniatiazed != 1 && DB: System is empty

show administrator setup form

‘Please provide 100% correct information and store it in safe place’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| username | password | phone | email | role | auth\_info |
| user defined | UD | UD | UD | sadmin | if any |

onSubmit() -> Update DB: App.key.app\_initialized = 1 then redirect to <https://bsms.io/admin>

else

redirect ‘requested page was not found’

fetch (GET) <https://bsms.io/admin>

1. if DB: App.key.app\_iniatiazed != 1 && DB: System is empty

message ‘Please contact with System administrator’

redirect to self with flash message

else

show login form for admin

onSubmit() fetch (POST) <https://bsms.io/admin>

1. Verify login info
2. on Success() Redirect to <https://bsms.io/admin/>mystore otherwise redirect to self with error message

|  |  |
| --- | --- |
| URL | <https://bsms.io/admin/mystore> (GET) [Tabs /?tab=&profile] |
| 1 | Show a Store Application Window |
| MENU | 1. My Store (Show User information and and Requisition submssion and progress) |
|  | 1. Requision (Initiated, Withdraw (if not approved yet)) |
|  |  |
|  |  |
|  |  |

User’s Activity (Route)

|  |  |
| --- | --- |
| Front End (Svelte) | Backend |
| #/login Display login form   1. Identification (Username/Phone) 2. Passkey (Password/ Private Key) 3. If Authentication is successful. Redirect to dashboard #/users/Dashboard | bsms.io/login (POST)   1. Accept the Request 2. Validate User Info. DB: users 3. Generate jwt token 4. Send redirect info |
| #/users/Dashboard | bsms.io/users/:id/action – ensure the id is the serial no only not the user id itself. |

|  |  |
| --- | --- |
| C-03 | Data Flow |