EXP. NO: 1
9/8/2024

Identifying a Software System – Stock Inventory Management

Aim:

To identify a software system that needs to be developed

Objective:

- To formulate the requirements to develop a stock inventory management system for shops
- Define the modules of the application
- Define the functional and non – functional requirements for the application

Introduction:

The reliance on computer applications for everyday tasks is ever increasing. Particularly, the involvement of software systems in managing businesses is very effective in cutting costs and making tedious tasks easier. Stock inventory management is one such task for shopkeepers. Managing the number of available items in the shop and tracking them during sale can be difficult to manage when the volume of sale is large. An automated system which tracks incoming and outgoing items during sale will make this task a lot easier.

Our stock inventory management system will address this issue by integrating billing and stock management, such that when an item is sold, the database of items is automatically updated to reflect the number of outgoing items. The system will also provide an interface to add, edit and manage new and existing items in the shop.

Additionally, this system will have provisions for user authentication to enable multiple employees to manage the available items. For example: the cashier will only have access to outgoing items and billing, while the manager can also update new items when they are added to the shop inventory. This will facilitate a secure workflow in the shop so that every employee only has access to what they need to do the job.

Modules:

Stock In: The Stock In page will allow users to add new items to the inventory. The page will get details of the items to be added from the user and update the database with the items.

Stock Out: The Stock Out page will allow users to track the items which were sold or from the shop inventory. It will contain the details of customers to whom the items were sold, along with time of sale, items sold and price paid by the customer. The database will automatically be updated when items are sold to remove the items from the shop inventory.

Sales Charts: Using the data from the stock out database, the number of items sold can be plotted in charts to show the number of sales in a user-friendly format. This will help the user to track sales within a time period and profits earned.

Billing: The integrated billing system will enable users (the cashier) to generate a bill for the customer. The cashier will be able to enter the details of the customer, add the items to be sold from the shop inventory database, and generate the invoice. When the sale is confirmed, the billing system will generate a bill in PDF format which can be printed out.

Authentication: Since a shop can have different types of employees, it will be unsafe to give full access to every employee. Instead, each employee will have their own account linked to their email, using which they can login to the application to access the information and services which they need while being restricted access to other information. For example, the cashier will have access to the stock out and billing services only, while the shop manager will have access to all services of the application.

Functional Requirements:

Hardware Requirements:

- Any modern desktop system
- Quad Core Intel / AMD Processor
- Minimum of 4GB RAM
- Internet connection
- Backend server to host the database, or a cloud hosting service such as Firebase

Software Requirements:

This section will outline the tech stack that will be used to develop the shop inventory management system.

Frontend: Bootstrap 5 (HTML / CSS Framework)

Bootstrap 5 is the latest major release of the popular open-source front-end framework, widely used for building responsive and mobile-first web applications. This version continues to build on the success of its predecessors by offering a robust set of tools and components that streamline the development process, enabling developers to create visually appealing, consistent, and functional user interfaces with minimal effort. Bootstrap 5 also embraces modern CSS techniques, including the use of custom properties (CSS variables), which offer greater flexibility in theming and customization. This allows developers to easily adjust the design system without needing to dive deep into the core code. The new utility API further enhances this customization by providing a powerful way to generate utility classes on the fly, tailored to the specific needs of a project.

The frontend of the stock inventory management system will be designed using Bootstrap 5, which will provide an appealing and user friendly interface for the user to interact with the application.

Backend: Ruby on Rails

Ruby on Rails, often simply called Rails, is a powerful and popular open-source web application framework written in the Ruby programming language. It follows the Model-View-Controller (MVC) architectural pattern, which helps organize and structure code in a way that separates the data layer (Model), the user interface (View), and the application logic (Controller). Rails is designed to make web development faster, more straightforward, and more efficient by providing developers with a set of conventions and built-in tools that minimize the need for repetitive code. One of Rails' most notable features is its emphasis on "convention over configuration," meaning that it provides sensible defaults and predefined structures, allowing developers to focus on writing application-specific code rather than spending time on boilerplate setup. This approach significantly accelerates the development

process, especially for startups and small teams. Rails also supports "Don't Repeat Yourself" (DRY), a principle that encourages code reuse and modularity, further reducing redundancy and errors. The framework comes with a wide array of built-in features, such as database handling, form validations, routing, and an integrated testing framework, which help streamline the entire development cycle from inception to deployment.

For our project, Ruby on Rails will be used to create a dynamic webpage which automatically updates when changes are made to the database. Rails provides a full package to develop the application from frontend to backend while also simplifying the development process. There are plenty of services which provide hosting for Ruby on Rails applications, such as Render. This can help deploy our application to the web at low cost.

Database: Firebase Cloud Firestore and Firebase Realtime Database

Firebase is a comprehensive app development platform by Google that offers a suite of tools and services for building, managing, and scaling web and mobile applications. It is particularly known for its real-time database and backend-as-a-service (BaaS) capabilities, which allow developers to focus on building user-facing features without worrying about server management or infrastructure. Firebase's database functions include two main types of databases: Firebase is a comprehensive app development platform by Google that offers a suite of tools and services for building, managing, and scaling web and mobile applications. It is particularly known for its real-time database and backend-as-a-service (BaaS) capabilities, which allow developers to focus on building user-facing features without worrying about server management or infrastructure.

Firebase Realtime Database is a NoSQL cloud database that stores data in JSON format and syncs it across all clients in real-time. This makes it ideal for applications requiring live data updates, such as chat apps, live feeds, or collaborative tools. Data in the Realtime Database is stored as a large JSON tree, and it allows developers to listen for changes and update data automatically across connected devices.

Cloud Firestore, on the other hand, is a more advanced and scalable NoSQL database. It supports richer data structures like collections and documents, offers stronger querying capabilities, and is better suited for complex and large-scale applications. Firestore also provides offline support, enabling applications to function seamlessly even without an active internet connection, and synchronizes changes once the connection is restored.

The details of items in shop inventory will be stored in the Cloud Firestore database, while the transaction details will be stored in the Firebase Realtime Database which allows quick and easy regular updating of data while being a low cost service.

Authentication: Firebase Authentication

Firebase Authentication is a service provided by Google’s Firebase platform that simplifies the process of adding user authentication and management to web and mobile applications. It supports a wide range of authentication methods, including email and password, phone numbers, and federated identity providers like Google, Facebook, Twitter, and GitHub. This flexibility allows developers to easily integrate various sign-in options into their apps, catering to diverse user preferences. One of the key advantages of Firebase Authentication is its ease of use. Firebase handles the heavy lifting of securely storing user credentials, managing

user sessions, and implementing best practices in security, such as password hashing and multi-factor authentication. This allows developers to focus on building the core features of their applications without worrying about the complexities of user authentication. Firebase Authentication also integrates seamlessly with other Firebase services, such as Firestore and Firebase Realtime Database, allowing developers to easily control access to data based on the authenticated user's identity. Additionally, it provides built-in UI libraries for common authentication flows, which can be customized to match the app's design.

Firebase authentication will be implemented so that employees can have their own accounts with only the necessary privileges granted to them, while the manager retains full access over the application and database management.

Non – Functional Requirements:

Security: The security and safety of our application will be guaranteed by the use of Firebase Authentication service. It will ensure that unprivileged users do not have the permission to modify the database, while only granting them the access to view items and perform other basic operations such as viewing sales and performing billing.

Quality: The quality of our application will be guaranteed by using standard and popular development frameworks such as Bootstrap and Ruby on Rails. They are tried and tested frameworks which have been used to develop large scale applications while also being highly secure. This will also allow us to regularly update our application with the latest version of the framework, which will address security issues and provide a more appealing interface with every version of the software.

Optimisation: The application will be incredibly lightweight by ensuring that no unnecessary code is present. Since the frameworks used to develop our application are already lightweight and provide only the bare necessities, our inventory management application will be fast and responsive to any task, even on mobile devices.

Bootstrap's responsive design allows users to use the application on the go on mobile as well, which makes stock management easier than ever while also being secure.

Speed: As discussed above, Ruby on Rails and Bootstrap are lightweight frameworks which only provide the bare necessities for a fully functional application. This makes our application fast to access and work with, while also being modular. Each functionality of the application will be optimised for its task and dependencies will be reduced to ensure high performance.

Loading: Our application will be deployed to the web on a cloud hosting service such as Render or Firebase. This will enable users to access the website from anywhere at anytime while still being secure. An internet connection will be the only requirement to access the application.

Conclusion:

Thus, we have defined all the requirements and functionalities of our stock inventory management system. This application will be developed with the above-mentioned frameworks and services, while keeping in mind the non – functional requirements to ensure user friendliness and responsiveness of the application.

Result:

Thus, the software system to be developed is identified and its modules, functional and non - functional requirements defined.