EXP. NO: 9

IMPROVING REUSABILITY AND MAINTAINABILITY

Aim:

To improve the reusability and maintainability of the stock management system by applying appropriate design patterns.

Reusability:

Frontend (Bootstrap & JavaScript):

- Bootstrap provides a modular, reusable component structure. Common UI components like buttons, forms, and tables can be reused across multiple pages.
- The various modules of the app are separated into different .js files containing functionality for each of the modules.
- This provides modularity by allowing us to reuse the same functions in different sections of the app, and also for different apps.
- It consumes very less resources to deploy. Hence the cost to deploy this across multiple servers is also low.
- The use of standard web technologies (HTML, CSS, JavaScript) ensures compatibility across multiple browsers and devices, allowing the same codebase to be reused for web applications tailored to different platforms or user groups.
- Bootstrap's grid system and pre-designed layouts allow for easy reuse of responsive design patterns. For example, a single layout for "Stock In" can be adapted for "Stock Out" with minimal changes.
- Since the app is lightweight and modular, new users can customize branding, themes, or specific modules without needing to alter the entire codebase, making it reusable for various business cases.

Backend (Firebase):

- Using Firebase Hosting or other CDN-based platforms ensures the app is distributed across global servers, enabling fast and reusable deployment for users in different geographic regions.
- Firebase SDK supports offline data persistence, making the app reusable for users in low-connectivity environments without requiring additional infrastructure changes.
- Firebase Authentication provides a plug-and-play solution for user management. The same authentication logic can be reused across different apps or modules, regardless of the app's specific features or domain.
- Firebase Firestore's flexible schema and hierarchical structure allow collections and subcollections to be reused for similar applications with minimal changes. For instance, a users collection or inventory collection can easily fit into other stock or inventory-related apps.

- Firebase SDKs support web, Android, iOS, and backend platforms, making backend functionalities reusable across multiple apps, irrespective of their platform or frontend stack.
- Firebase Realtime Database's real-time syncing feature is reusable for any app requiring instant updates (e.g., collaborative apps, dashboards). The same implementation logic used for stock updates can be reused in a chat application or a collaborative editor.
- Firebase integrates easily with other services like Stripe, Twilio, and Google APIs. These integrations can be configured once and reused for similar use cases in other projects.
- Firebase's configuration and deployment can be templated using JSON or YAML files, allowing reusable and consistent setup for future projects or app variations.
- Firebase Storage can be reused for storing shared assets (e.g., images, PDFs) across apps. For example, the same storage bucket can serve as a centralized repository for multiple stock management or e-commerce systems.

Maintainability:

Frontend (Bootstrap & JavaScript):

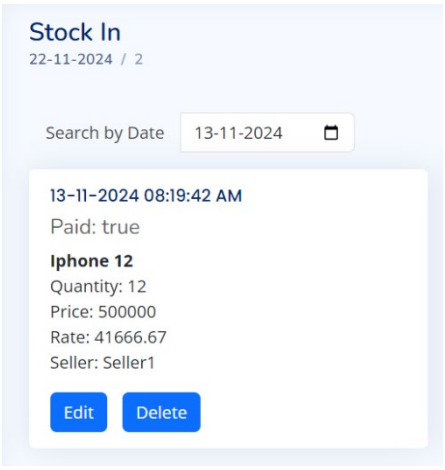
- Organizing the codebase by separating HTML, CSS, and JavaScript ensures that each layer of the frontend is independently manageable, making updates and debugging easier.
- By dividing the frontend into components or modules (e.g., separate .js files for "Stock In," "Billing," etc.), updates can be made to individual parts without affecting the entire application.
- Following consistent coding practices, such as adhering to a linter (e.g., ESLint) or style guide, makes the code easier to read and maintain over time, especially for teams.
- Using a widely adopted framework like Bootstrap ensures long-term support, clear documentation, and compatibility with newer versions, reducing maintenance efforts when updating styles.
- Storing shared assets (e.g., icons, styles, templates) in a single location ensures updates are reflected across the app without the need for duplicate changes.
- Bootstrap's responsive grid system ensures compatibility with various device sizes, reducing the need for future maintenance as new screen sizes or devices emerge.
- Including clear comments and maintaining a README.md for the frontend modules helps future developers (or yourself) quickly understand the purpose and structure of the code.
- Using Git for tracking changes to HTML, CSS, and JavaScript ensures that any issues can be traced and resolved quickly.

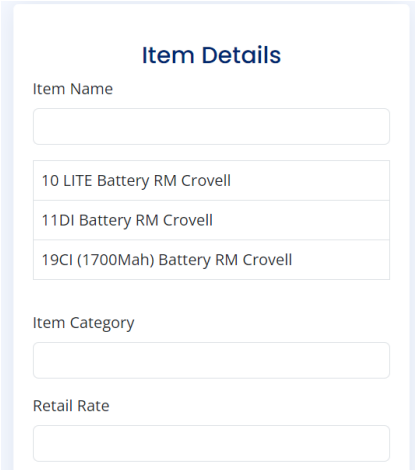
Backend (Firebase):

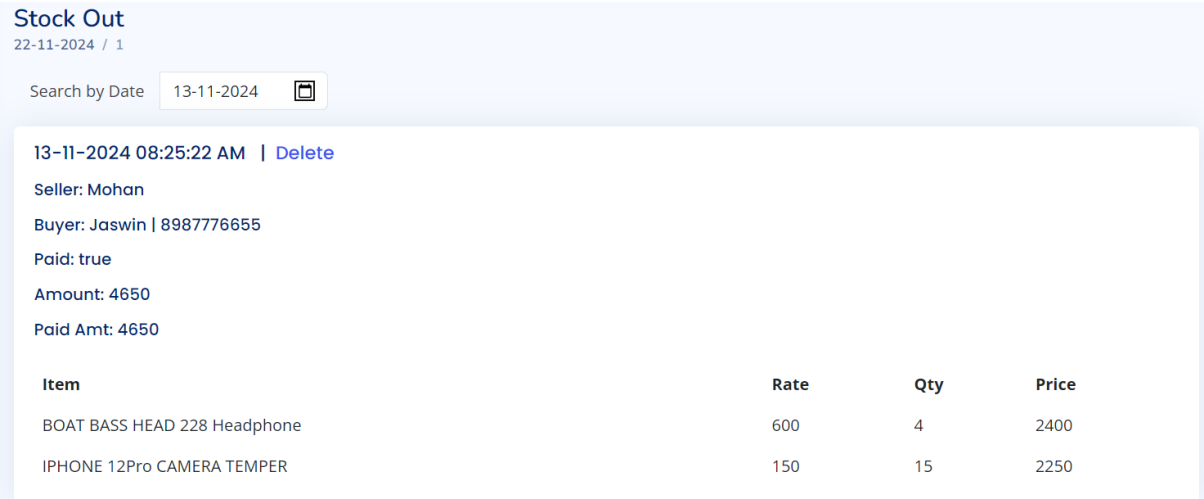
- Maintaining clear documentation of the Firestore or Realtime Database schema (e.g., collections, document formats) ensures future developers understand data relationships and structures.
- Writing clear and optimized Firebase Security Rules reduces the risk of errors during updates and makes the app easier to maintain as new features are added.
- Modular Cloud Functions for backend logic (e.g., stock updates, billing calculations) make the backend easier to maintain and extend. Updates to one function do not affect unrelated parts of the app.
- Firebase provides monitoring and debugging tools (e.g., Crashlytics, Performance Monitoring) that help identify and resolve backend issues efficiently.
- Writing efficient Firestore queries (e.g., indexed searches, limited document reads) ensures performance remains stable, reducing maintenance when scaling the app for more users.
- Staying up-to-date with Firebase SDK versions ensures compatibility with the latest features and reduces the risk of deprecated functions causing issues.
- Using environment variables or configuration files (e.g., .env) for Firebase project settings (API keys, database URLs) allows for easier maintenance when switching between development, staging, and production environments.
- Configuring regular backups of Firestore and Realtime Database ensures data integrity and simplifies recovery in case of accidental data loss or errors.
- Using Firebase Remote Config to manage and update app behavior without requiring redeployment makes maintaining live applications more efficient.
- Implementing detailed error logs in Cloud Functions and frontend Firebase calls makes debugging easier, reducing the effort required to maintain the backend

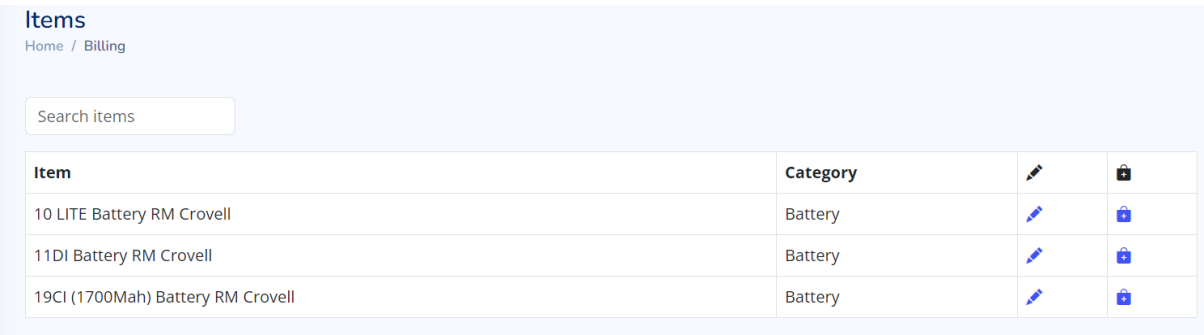
Modular JS Components:

Bootstrap Components:









Description	Allotted Marks	Obtained Marks
Preparation	**20**	
Design / Implementation	**20**	
Viva	**15**	
Output	**10**	
Record	**10**	
Total	**75**	

RESULT:

Thus, the Stock Inventory Management System is improved for maintainability and reusability.