Pinak Ganatra **1210339**

Here is my summary of the project we made for the subject, Embedded systems and software design. The summary written here on my side of the project i.e. Embedded systems and software design.

Summary of the Project:

This project involves a comprehensive inventory management system for a retail business. The system is designed and built using Raspberry Pi, SQLite as the database, and Python for scripting.

Components Used:

1. **Raspberry P**i: A low-cost, credit-card-sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is used as the primary device to run the Python scripts and interact with the database.
2. **SQLite:** A C library that provides a lightweight, disk-based database that doesn’t require a separate server process. This is used to store product and customer data for easy access and manipulation.
3. **Python:** A high-level, interpreted programming language that is easy to learn and includes multiple libraries for a broad range of applications. This is used to interact with the SQLite database and the barcode scanner.
4. **Barcode Scanner:** An input device used to scan the barcodes present on the products.

Purpose of the Project:

The project can be used to manage inventory in a retail environment effectively. With the help of a barcode scanner, each product can be tracked and updated in real-time in the SQLite database. It allows adding, deleting, or updating product details, such as the name, the number of available pieces, the buying price, etc. It also supports renting a product if it's marked as rentable.

Problem Solved:

The project addresses several challenges in managing a retail store:

1. **Inventory Management:** Tracking the stock of different products can be difficult, especially as the number of distinct items in the inventory increases. This project allows easy and efficient management of all products in real-time.
2. **Product Identification:** By using a barcode scanner, each product can be uniquely identified, which mitigates the risk of human error during manual entry.
3. **Rental System:** If a product is available for rent, the program can track the rented products, the customer details, and the return date.
4. **Database Management:** With SQLite, the program can efficiently manage and manipulate the data in the database.

In conclusion, the project streamlines the process of inventory management and enhances the efficiency of the retail store's operations.

Databank

The SQLite database you created in your project contains several tables, each with a specific purpose. Here's a summary of each:

1. **Artikel (Products):** This table is used to store all the information about the products. Each product has unique identifiers like `ArtID` and `barcode`. The table also includes fields like `ArtName` (product name), `Anzahllager` (stock quantity), `buying_price` (purchase price), `PreisProTag` (price per day), `PreisGesamt` (total price), `AblaufsDatum` (expiry date), `Image` (image), `NaechstePruefDatum` (next test date), `BestandLimit` (stock limit), `Ausleihbar` (rentable), and `LagerPlatz` (storage place).

2. **Benutzer (Users):** This table contains information about the users who are authorized to use the system. `BenId` is the unique identifier for each user. `Benutzername` (username) and `Passwort` (password) are used for authentication. This table also has a foreign key `KundID` referencing the customer table.

3. **Ausleihe (Rentals):** This table is used to store information about product rentals. It includes `AusleiheID` (unique identifier), `ArtID` (reference to the product being rented), `BenId` (reference to the user who rented the product), `KundID` (reference to the customer who rented the product), `ArtName` (name of the product), `AusleiheDatum`

(rental date), `AbgabeFrist` (return date), `ArtAnzahl` (quantity of products rented), and `Pfand` (deposit).

4. **Defekt (Defects):** This table contains information about the defective items. It includes `DefektID` (unique identifier), `ArtID` (reference to the defective product), `Anmerkung` (notes), `Anzahl` (number of defective units), `ArtName` (name of the defective product).

5. **Einkauf (Purchase):** This table is used to store information about the purchases. It includes `BestellID` (unique identifier), `ArtID` (reference to the purchased product), `BestellDatum` (order date), `BestellAnzahl` (order quantity), `Anmerkung` (notes).

6. **Kunde (Customers):** This table contains information about the customers. `KundID` is the unique identifier for each customer. Other fields include `KundName` (customer name), `Matrik` (matrix), `Email` (email address), `Vermerk` (notes).

Overall, the structure of the database enables efficient management of products, rentals, defects, purchases, and customer data, which are essential elements in a retail business.

Github

GitHub was an invaluable part of this project. As a collaborative platform built on top of Git, it provided us with the necessary tools to work together efficiently, manage our codebase, and ensure that our project stayed on track. Here's how:

1. **Version Control:** One of the fundamental features of GitHub is its robust version control system, built on Git. This allowed us to keep track of all changes made to the project, who made them, and when they were made. If a bug was introduced, we could easily trace back to the problematic changes and rectify them. It also ensured that none of our work was ever lost - if we ever needed to go back to a previous version of our code, we could do so effortlessly.

2. **Collaboration:** GitHub made it easy for the entire team to work together. We could all work on different features or bug fixes in parallel, using branches, without stepping on each other's toes. The Pull Request feature allowed us to review each other's code before merging it into the main codebase, ensuring code quality and consistency.

3. **Issue Tracking:** GitHub's issue tracker helped us manage bugs and feature requests efficiently. It allowed us to assign issues to specific team members, label them for easy

categorization, and even link them to specific commits or pull requests. This made it easy for everyone to stay updated on what's being worked on, and what needs to be worked on next.

4. **Documentation:** GitHub's built-in wiki and README features allowed us to maintain comprehensive documentation for our project. This included explanations of our code, instructions on how to use our software, and even meeting notes and project plans. By centralizing our documentation on GitHub, we ensured that it was always up-to-date and easily accessible to the entire team.

5. **Integration:** GitHub's wide range of integration options meant we could seamlessly connect it to our other tools. Whether it was linking it to our continuous integration server to automatically run tests on every commit, or to our project management software to keep track of progress, GitHub sat at the heart of our development process.

In summary, GitHub proved to be not just a code hosting platform for us but a comprehensive tool that played a significant role in project management, team collaboration, code quality assurance, and documentation. The benefits we gained from using GitHub were enormous and definitely contributed to the successful completion of our project.