



Project Report

Project Title: Inventory Management System

UE20CS352 – Object-oriented Analysis and Design with Java

Submitted by:

**Name 1: Arya
Girigoudar
PES1UG21CS806
Name 2: Priya Gawli
PES1UG21CS829
Name 3: Vishnu Patgar
PES1UG21CS844**

Under the guidance of

Prof. Bhargavi M
Professor
PES University

January - May 2023

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru – 560 085, Karnataka, India

Problem Statement

The antiquated and ineffective inventory management system in use by XYZ Company frequently causes stockouts, overstocking, erroneous inventory levels, and missed revenues. Making informed decisions about stock replenishment, purchases, and pricing is difficult for the business due to a lack of real-time access into inventory data. Spreadsheets and manual data input procedures also raise the risk of errors, which makes the system even less accurate and ineffective. A new inventory management system is therefore required, one that can offer real-time inventory data, automate inventory tracking and replenishment, decrease errors, and optimize inventory levels to enhance overall operational efficiency and customer satisfaction.

Synopsis

Any business needs an inventory management system because it is crucial to ensuring efficient operations and customer satisfaction.

The XYZ company's existing inventory management system is antiquated and ineffective, which causes a number of issues like stockouts, overstocking, erroneous inventory levels, and missed revenues.

The lack of real-

time insight into inventory data, manual data input procedures, and spreadsheets that make mistakes more likely are the main causes of these issues.

The business needs a new inventory management system that can automate inventory tracking and replenishment, offer real-time inventory data, minimize errors, and optimize inventory levels in order to handle these problems. By guaranteeing that the correct products are accessible at the right time and at the right price, the new system will increase overall operational efficiency, decrease expenses, and increase customer happiness.

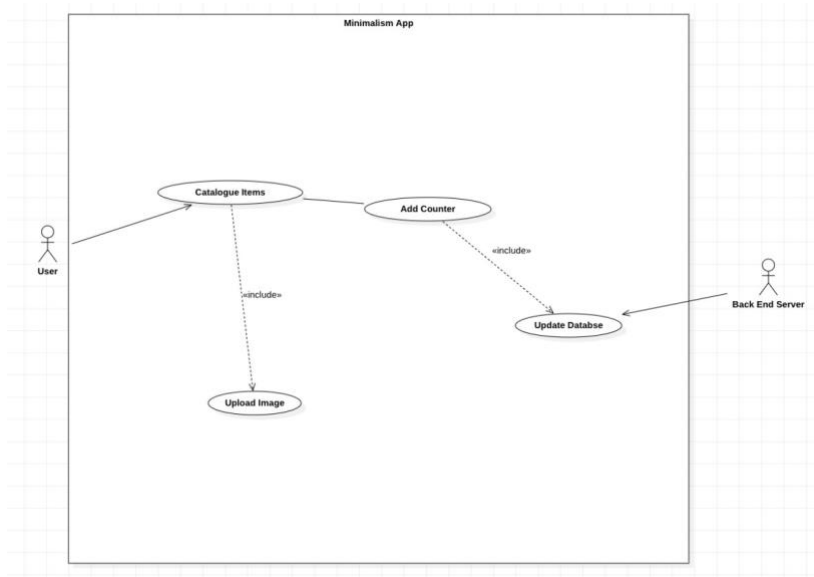
Because the suggested inventory management system would be cloud-based, anyone, anywhere, at any time, will be able to access real-time inventory data. Additionally, it will have an automated inventory tracking and replenishment feature that will lessen the need for manual data entry, cut down on errors, and guarantee that inventory levels are kept at their ideal levels. Additionally, the system will produce statistics and analytics that will help the business decide on pricing, purchasing, and stock replenishment.

Finally, the suggested inventory management system will offer a more effective, accurate, and dependable method of managing inventory, enhancing the overall effectiveness and profitability of the business.

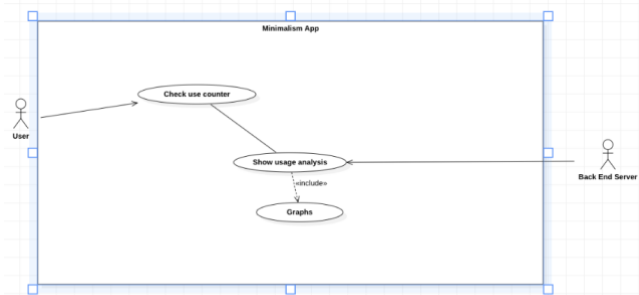
Use Case and Class models

Use case diagrams:

1. **Cataloging possessions:** Users can create a digital catalog of their possessions, allowing them to keep track of what they own and where it is located.

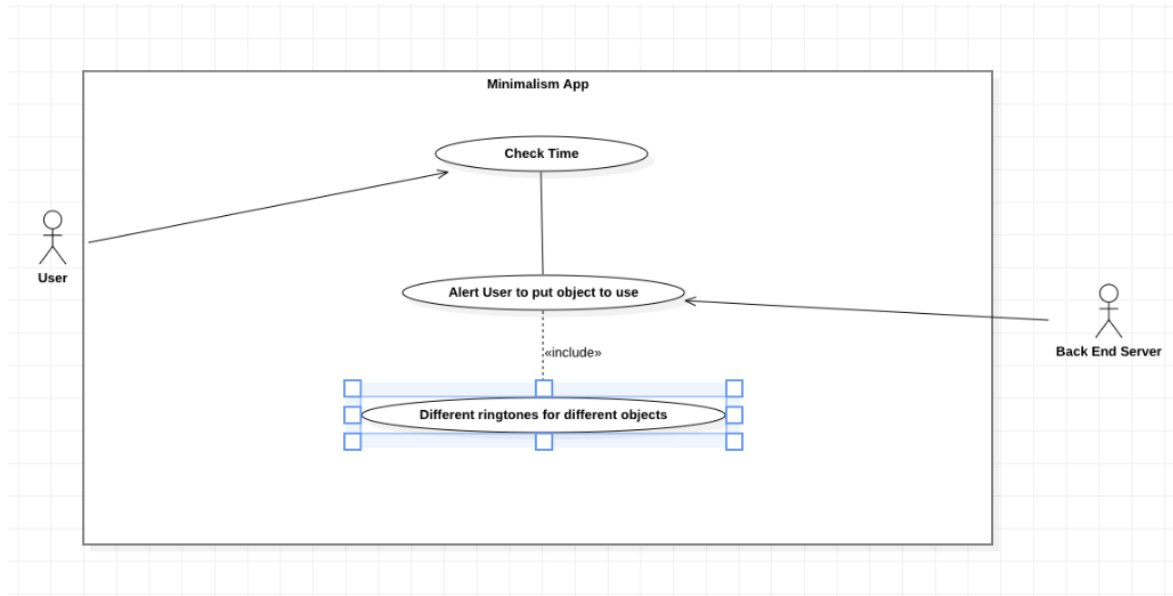


2. **Tracking Usage:** The app allows users to track how often they use each item in their possession, helping them make informed decisions about what they need and what

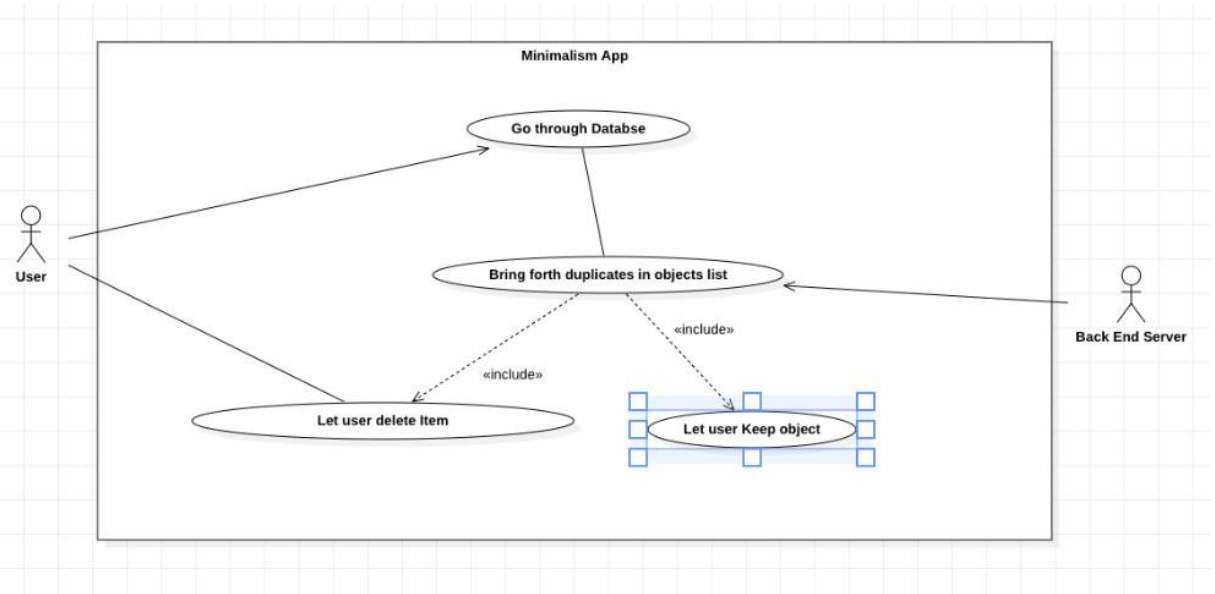


they can let go of.

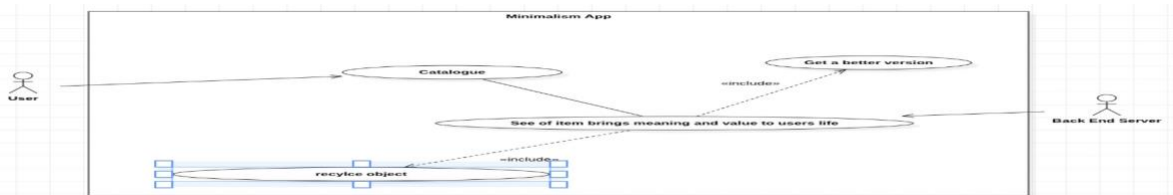
3. **Setting reminders:** Users can set reminders to use particular items, helping them get the most out of their possessions.



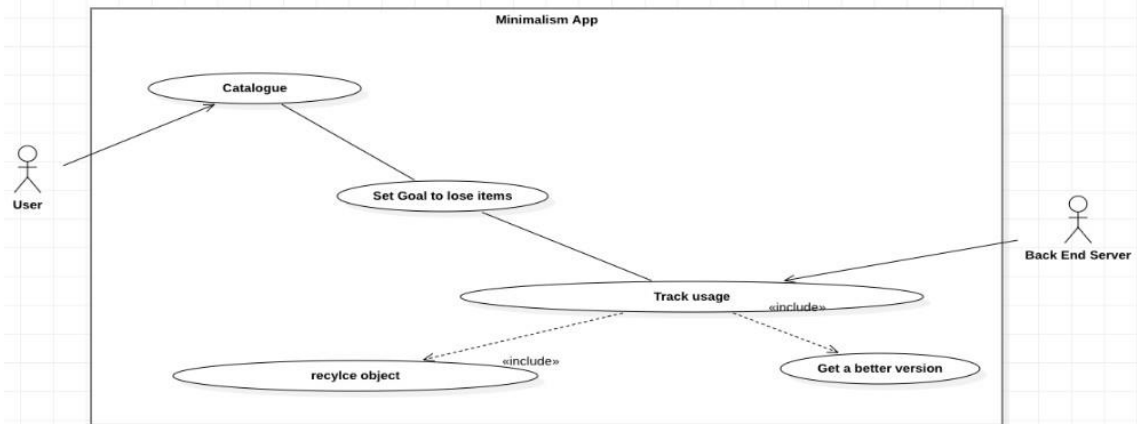
4. **Identifying duplicates:** The app can help users identify duplicate items in their possession, allowing them to streamline their possessions and minimise clutter.



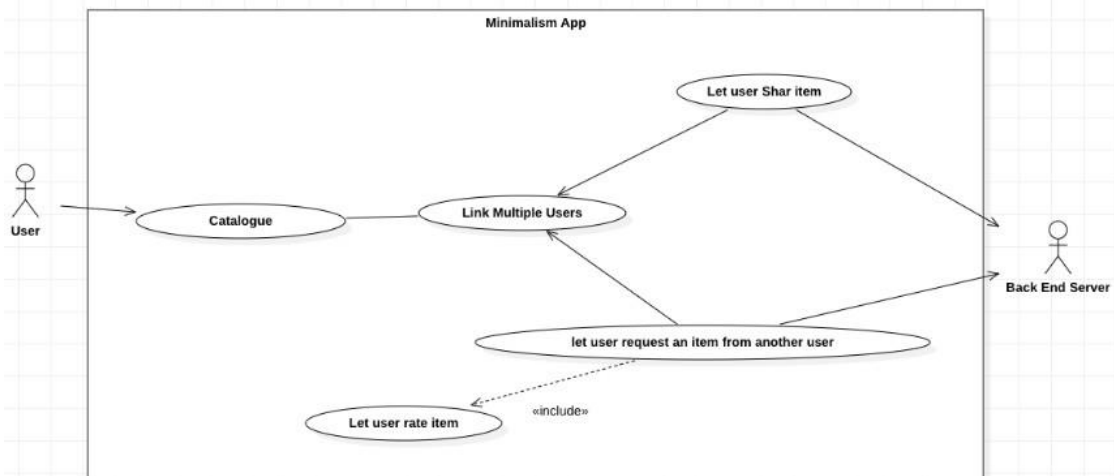
5. **Recycling and upcycling:** The app can provide tips and suggestions for recycling or upcycling items, helping users minimise waste and reduce their environmental impact.



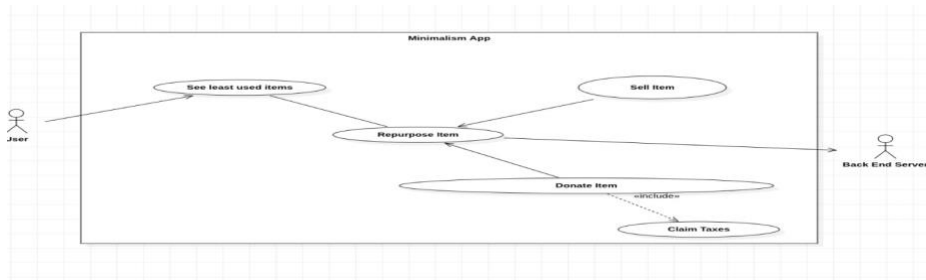
6. **Goal setting:** Users can set goals for minimising their possessions, reducing waste,ting a more minimalist lifestyle, and track their progress over time.



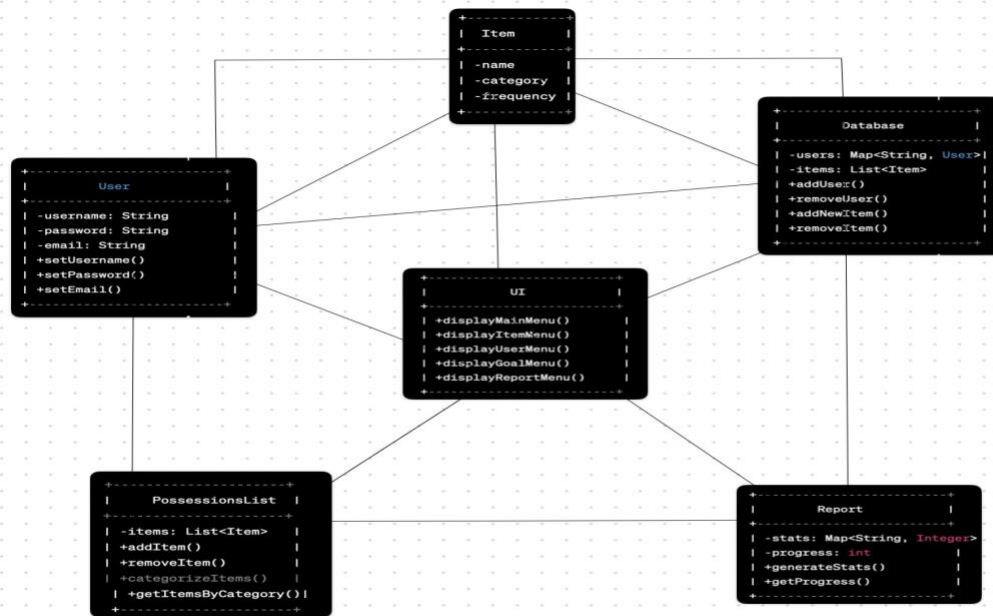
7. **Sharing Possessions:** The app can facilitate sharing possessions with others, helping users reduce waste and build a more collaborative community.



8. Donating items: The app can provide information on where to donate items that users no longer need, helping them give back to their community and reduce waste.



Class Diagram



Architecture Patterns, Design Principles and Design Patterns used

The minimalist lifestyle app follows the following design principles:

- Single Responsibility Principle (SRP): Each class has a single responsibility, and its behavior is focused on that responsibility only.
- Don't Repeat Yourself (DRY): The project avoids duplication of code by implementing reusable components such as the DatabaseHandler class, which is used throughout the project.
- Keep It Simple, Stupid (KISS): The app is designed to be simple and easy to understand, with straightforward code and minimal complexity.
- Separation of Concerns (SoC): Different concerns and functionalities are separated into different classes, allowing for better maintainability and scalability.

The app implements the following architectural patterns:

- Model-View-Controller (MVC) - This pattern is used to separate the application into three interconnected components, namely the Model, View, and Controller.
- Singleton - This pattern is used to ensure that only one instance of a class is created throughout the lifetime of an application.
- Facade - This pattern is used to provide a simplified interface to a complex system or set of classes.

The app implements the following design patterns:




- Singleton pattern: The DatabaseHandler class implements the Singleton pattern, which restricts the instantiation of a class to a single object.
- MVC design pattern: The different classes act as the View, Controller, and Model components. The View receives user input from the UI and updates the Model based on that input. It also updates the UI to reflect the changes made to the Model. The Model represents the data and business logic of the app.

- Builder pattern: The app uses the Builder pattern by utilizing the StringBuilder class to build a string character by character.

- **Individual contributions of team members**

- Arya Girigoudar: CSS/FRONT END, Connected item list to backend, Backend
- Priya Gawli: Model of project, H2 integration
- Vishnu Patgar: Schema Diagrams, Backend
- [GithubLinkHere](#)

Screenshots of the GUI

Ite...	Item Counter ▴ ▾	Item Category ▴ ▾
	70	jigrormo
	11	zun
	51	capfad
	11	dec

☰ Item List

Item Name ↕	Item Category ↕	Item Counter ↕	Ite...
zun capfad	jigrormo	70	
capfad dec	zun	11	
dec bivo	capfad	51	
bivo judbilo	dec	11	

Inventory Management

☰ Item List

Item Name ↕

zun capfad

capfad dec

dec bivo

bivo judbilo

Item Name

Item Category

Item Counter

Item Image

Upload File...

 Drop file here

Save

Cancel

