Online Sale of Goods Requirements Document Implementation plan

Group Members:

- Anıl Abdullah İnce – 21070006006
- Cavit Utku Tipici – 21070006043
- Bora Tarhan - 21070006044
- Atakan Yıldız - 21070006055
- Ceyda Oymak - 22070006012
- Zeynep Özdemir – 22070006076

Table of Contents

1 Overview

This document outlines the Implementation Plan for the *Online Sale of Computer and Computer Parts* system, a lightweight e-commerce platform that enables users to browse products, filter items by category, add components to a shopping cart, and complete a simplified checkout process. The system also includes an admin interface for adding and managing products.

The project schedule spans from **October 2025 to January 2026**, divided into three iterations:

- **Iteration 1 (Nov 1–10):** User registration, login, product listing
- **Iteration 2 (Nov 10–Dec 10):** Category filtering and shopping cart
- **Iteration 3 (Dec 11–Jan 10):** Checkout simulation and admin panel

The goal of this plan is to define:

- Project milestones
- Team staffing and responsibilities
- Development and iteration process
- Coding standards
- Testing strategy
- Key measurement metrics
- Identified risks and mitigation steps
- Required tools, hardware, and software support

By following this plan, the development team ensures a structured, iterative, and predictable software engineering workflow leading to the timely completion of the final product.

2 Milestones

The project development will be divided into three major milestones:

2.1 Milestone 1 – Core Functionality / Week 1 – Week 2 (Nov 1 – Nov 10)

This milestone delivers the essential backend operations and the first working prototype of the system.
The following features will be completed:

- User registration and login
- Product listing and category data retrieval
- Basic product detail view
- Initial database setup using **SQLite**
- Project architecture setup (backend + frontend skeleton)

Outcome:
A functional prototype where users can log in and view all available computer parts.

2.2 Milestone 2 – User Interface & Persistence / Week 2 – Week 6 (Nov 10 – Dec 10)

This milestone focuses on interactive user features and frontend functionality.

Deliverables include:

- Category-based product filtering
- Fully functional shopping cart (add/remove/update)
- Basic UI enhancements for product pages
- Backend operations for cart management
- Data persistence through SQLite
- Initial admin page layout

Outcome:
Users can browse products, filter by category, and manage items in their shopping cart.

2.3 Milestone 3 – Final Release Package / Week 6 – Week 10 (Dec 11 – Jan 10)

This milestone finalizes the system and prepares the product for submission.

Deliverables include:

- Checkout simulation (order confirmation message)
- Fully working admin panel to add new products
- Improved UI design and responsive layout
- Bug fixes and functional polishing
- System-wide testing and integration validation
- Final documentation (PDF)

Outcome:
A stable, functional e-commerce prototype that demonstrates the complete user flow from login to checkout.

3 Staffing

3.1 Requirements Team

Responsible for understanding and documenting all functional requirements of the e-commerce system.

Developers: Cavit Utku Tipici, Ceyda Oymak

3.2 Design Team

Responsible for database schema design, system architecture, and basic UI wireframes.

Developers: Bora Tarhan, Atakan Yıldız, Zeynep Özdemir

3.3 Iteration 1 – Core Logic

Focus on backend logic, user registration/login, product listing APIs, and basic frontend structure.

Developers: Bora Tarhan, Zeynep Özdemir, Ceyda Oymak, Anıl Abdullah İnce

3.4 Iteration 2 – Interface & Storage

3.4.1 Interface Coding Group

Responsible for implementing the frontend pages category filtering UI, shopping cart interface, and admin panel layout.

Developers: Ceyda Oymak, Atakan Yıldız, Cavit Utku Tipici

3.4.2 Persistent Storage Group

Responsible for SQLite database structure, backend API integration, and data persistence.

Developers: Anıl Abdullah İnce, Bora Tarhan

3.4.3 Testers

Responsible for manual/system testing and bug reporting.

- Zeynep Özdemir, Anıl Abdullah İnce, Atakan Yıldız

4 Process

4.1 Check-in / Check-out (Version Control)

We will use **Git and GitHub** for version control.

Feature workflow:

1. Create a feature branch
2. Commit regularly
3. Open a Pull Request
4. Reviewed by a teammate
5. Merge into the main branch after approval

4.2 Build Process

- **Frontend:** HTML/CSS/JavaScript, no build step required
- **Backend:** Runs directly with Node.js (node server.js)
- **Database:** SQLite, no additional build process
- **Weekly Build:** A general project check will be done every Monday

5 Coding Standard

5.1 Coding Standard Verification

- Weekly code checks will be performed by team members.
- Any inconsistencies in formatting, naming, or structure will be noted in GitHub comments.
- Violations must be corrected before merging into the main branch.

5.2 Code Reviews

- The team will hold **bi-weekly code reviews** to ensure consistency.
- Every feature must be submitted through a **Pull Request (PR)**.
- At least **one reviewer approval** is required before merging.

5.3 Testing

5.3.1 Unit Testing

Since the project uses **simple JavaScript and SQLite**, formal automated testing is minimal.

- Basic unit tests will be written for backend utility functions.
- Console-based tests and manual validation will be used for route handlers.

5.3.2 Integration Testing

Integration testing will be performed manually using tools such as:

- **Postman** for API testing
- Browser-based testing for frontend interactions

Test coverage includes:

- Product listing
- Category filtering
- Shopping cart operations
- Admin product creation

5.3.3 System Testing

System-level testing verifies that the complete user flow works correctly:

- Register / Login
- Browse products
- Filter categories
- Add to cart
- Remove/update items
- Checkout simulation
- Admin adds a new product

5.4 Coding Standard

5.4.1 Spelling & Naming

- **camelCase** → variables and functions
- **PascalCase** → backend classes (if any)
- **UPPER_CASE** → constants
- File names must be descriptive and consistent.

5.4.2 Punctuation & White Space

- **2-space indentation**
- No trailing whitespace
- Meaningful line breaks for readability
- Common formatting rules followed manually or using Prettier (optional)

5.4.3 Documentation & Style

- Each file begins with a brief header (purpose, creation date).
- Functions include a short description of behavior and parameters.
- Complex logic (e.g., cart updates, admin operations) must be explained with inline comments.

5.4.4 Comments

Comments should be added to explain:

1. Business logic structure
2. Non-obvious calculations
3. API input/output behavior
4. Any limitations or assumptions

6 Measurements

6.1 SLOC (Source Lines of Code)

Tracked at each weekly build to measure progress.

6.2 Class Count

Number of classes in backend (models/controllers/services) tracked per build.

6.3 Methods per Class

Average method count measured to identify complexity.

7 Risks

7.1 Support Personnel

Delays in requirement clarification or design feedback may slow down development.

7.2 Tool Usage

Some team members may need extra time to become comfortable with Git, Node.js, or SQLite, which could cause minor delays.

7.3 Technical Issues

Unexpected errors or misconfigurations in the development environment (Node.js, database file, or local server) may require additional debugging time.

8 Tools

8.1 Tools Used

Git & GitHub – version control and team collaboration

VS Code – primary development environment

Node.js + Express (or basic Node server) – backend development

SQLite – lightweight, file-based database used for product and user data

HTML / CSS / JavaScript – frontend implementation

Figma – UI sketches & wireframes

Postman – testing and validating API endpoints

Google Drive / Docs – documentation and report collaboration

9 Support

9.1 Hardware

Developer laptops (minimum 8 GB RAM, stable internet connection)

Local development environment (Node.js + SQLite running on each member's machine)

9.2 Software

- **Node.js** – backend runtime
- **SQLite** – lightweight database
- **VS Code** – main development editor
- **Git & GitHub** – version control
- **Google Chrome / Firefox** – browser testing
- **Postman** – API testing

9.3 People

Course instructor – feedback and milestone approval

Team members – collaboration and development

Testers – manual testing & QA

10 Class Diagram