A
Major Project Report
ON

PC Forge

for the Partial fulfilment of the award of
Degree of
Master of Computer Applications (MCA)

UNDER THE GUIDANCE OF:
Ms. Ruma Srivastava
Assistant Professor

SUBMITTED BY:
Kunal Sharma
24MCAN0014

School of Computer Applications
JECRC UNIVERSITY, JAIPUR
December 2025

Candidate's Declaration

I hereby declare that the project work, which is being presented in the Project Report, entitled, **PC Forge** in partial fulfilment for the award of Degree of "Master of Computer Application" in Department. of Computer Application, JECRC University is a record of my own investigations carried under the guidance of Ms. Ruma Srivasta. I have not submitted the matter presented in this Project Report anywhere for the award of any other Degree.

Name of Supervisor/Guide
Ms. Ruma Srivastava

Name of Candidate:
Kunal Sharma
Registration No: 24MCAN0014

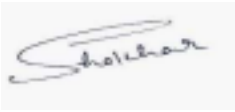
Signature

JECRC University
School of Computer Application
Major Project Completion Certificate

This is to certify that **Kunal Sharma**, Registration No. **24MCAN0014** has successfully completed the Major Project as a part of the requirements for the Master of Computer Applications (MCA) Program at JECRC University, Jaipur, India.

Project Title: PC Forge
Technology Used: HTML, CSS, JavaScript, PHP, MySQL
Project Supervisor: Ms. Ruma Srivastava

This project demonstrates student's ability to apply theoretical and practical knowledge gained during his/her academic studies to practical real-world scenarios. His/her dedication, hard work, and proficiency in the subject matter is commendable.



Mr. Shekhar Chander
HOD
Ms. Ruma Srivastava
Project Supervisor.

ACKNOWLEDGEMENT

I am deeply grateful for the support, guidance, and encouragement I received throughout the development of my project **"PC Forge."**

First and foremost, I would like to express my sincere thanks to my mentor/guide **Ms Ruma Srivastava**, whose continuous support, valuable feedback, and expert insights played a crucial role in shaping this project. Their guidance helped me overcome challenges and enhanced the overall quality of my work.

I extend my heartfelt gratitude to the faculty members of **Computer Application** for providing a strong academic foundation and a motivating environment that encouraged me to explore, learn, and innovate.

I would also like to thank my classmates, friends, and peers for their consistent encouragement, constructive suggestions, and help throughout the project development process.

A special note of appreciation goes to my family, whose constant motivation, patience, and emotional support have always been a source of strength and inspiration for me.
Finally, I acknowledge all the authors, online resources, and tools that contributed insights, knowledge, and technical support essential for building this project.

This project, **PC Forge**, would not have been possible without the collective support of everyone mentioned above, and I express my deepest gratitude to all.

Thank You

Table of Contents

Problem Definition

Introduction

PC Forge is a comprehensive, web-based system designed to simplify and streamline the process of building a custom personal computer. In today's dynamic hardware ecosystem, users—especially beginners—often face challenges such as understanding component compatibility, selecting performance-appropriate parts, and managing budget constraints.

PC Forge addresses these challenges by providing an interactive, guided environment where users can select PC components step-by-step (CPU → Motherboard → RAM → GPU → Storage → PSU, etc.) while the system automatically filters incompatible parts. This compatibility-driven workflow ensures that users avoid common mistakes such as mismatched CPU sockets or incorrect RAM types.

Additionally, the platform integrates real-time pricing, a budget tracker, user authentication, build saving capabilities, and an administrative backend for managing component inventory. Through its intuitive UI and backend logic, PC Forge serves as a digital assistant for PC builders, enabling a smooth experience from exploration to final build creation.

Need

Building a custom PC provides better performance optimization and cost efficiency than buying pre-built systems. However, for many users, understanding hardware specifications and ensuring compatibility can be overwhelming. Mistakes—such as selecting the wrong motherboard socket or incompatible RAM—can lead to financial losses and wasted time.

The platform fulfills several key needs:

- **Eliminating Compatibility Errors:** Automatically restricts component suggestions based on user selections (e.g., choosing an Intel CPU only shows compatible motherboards).
- **Real-Time Budget Awareness:** Displays cumulative build cost instantly, allowing users to stay within financial limits.
- **Centralized & Updated Part Information:** Streamlines the decision-making process by providing a unified, up-to-date inventory of parts.
- **User Convenience:** Enables storing builds for later review, comparison, or PDF export.
- **Administrative Efficiency:** Admins can manage hardware listings dynamically without altering source code.

Due to the complexity of hardware combinations and the growing interest in custom PC building, a tool like PC Forge is essential for making the process accurate, informative, and user-friendly.

Purpose

The primary purpose of PC Forge is to democratize and simplify the custom PC building process for users of all skill levels. The system aims to:

- **Automate Compatibility Checking:** Ensure users are only shown components that match previously selected hardware attributes, preventing technical errors.
- **Provide Budget Management Support:** Offer a dynamic budget tracker with real-time warnings when the selected components exceed the user's set budget.
- **Enhance User Experience:** Deliver an intuitive, clean, dark-themed UI for easy navigation and component comparison.
- **Enable Build Persistence:** Allow registered users to save, manage, and export their builds.
- **Support Efficient Administration:** Equip administrators with tools for CRUD operations on components, enabling quick updates to inventory and pricing.
- **Improve Data Accuracy & Transparency:** Ensure that all selections and cost calculations are logically processed and accurately presented.

Ultimately, PC Forge seeks to create a reliable, user-centric system that bridges the knowledge gap between hardware complexity and consumer understanding. It empowers users to create fully compatible, performance-optimized PC builds with confidence.

Objective

The PC Forge project is developed with the goal of providing a streamlined, error-free, and user-friendly platform for custom PC building. The system integrates intelligent compatibility logic, real-time pricing, user account management, and administrative tools to offer a comprehensive solution for both end-users and system administrators.

The major objectives of the PC Forge system are:

1. To provide a user-friendly interface for selecting PC components

The system aims to simplify the component selection process by offering a clean, intuitive UI where users can browse categories and choose parts in a guided, step-by-step manner.

2. To implement a compatibility engine for error-free component selection

PC Forge ensures that only compatible components are shown to the user. For example, selecting an Intel CPU automatically filters compatible motherboards. This prevents costly mistakes and enhances user confidence.

3. To provide real-time price calculation and budget management

As users select components, the system continuously updates the total price. A budget tracker visually alerts users if they exceed their predefined spending limit.

4. To offer secure user authentication for saving and managing builds

Users can create accounts, save multiple builds, and revisit or download their configurations. Authentication is handled securely using hashed passwords to protect user data.

5. To develop an admin panel for inventory and data management

Administrators can add, edit, delete, and manage hardware components, ensuring that the system's database stays accurate and up to date without modifying backend code.

6. To enable build summary export in PDF format

Users can generate a printer-friendly PDF version of their custom build for review, purchase, or documentation purposes.

7. To design a modular and maintainable system architecture

The project follows a structured, modular approach where the frontend, backend, and database are well-organized for easy maintenance and scalability.

8. To support a modern dark-theme user interface

The system adopts a visually appealing dark-mode theme aligned with modern web standards to enhance readability and user experience, targeting hobbyists, gamers, and developers.

9. To ensure seamless flow from build creation to final output

From starting a build → selecting compatible parts → calculating price → saving → exporting PDF, the objective is to make the process smooth, guided, and error-free.

System Study

System Study provides an in-depth understanding of how PC Forge works, why it is needed, and how its design aligns with software engineering principles. It covers system overview, data gathering, feasibility evaluation, and cost–benefit considerations.

System Overview

PC Forge is a modular, web-based application designed to guide users through building a custom PC by ensuring compatibility between selected components. The system uses PHP for backend logic, MySQL for database operations, and JavaScript for real-time interactivity.

The project is divided into two main modules:

1. User Module

- Component selection through a guided step-by-step builder (CPU → Motherboard → RAM → etc.).
- Automatic filtering of incompatible parts (e.g., socket type, RAM type).
- Real-time price updates and budget tracking.
- User authentication, allowing login, registration, and saving builds.
- PDF export of completed configurations.

2. Admin Module

- Dashboard for viewing overall statistics (users, builds, parts).
- Complete CRUD operations for hardware components (Add, Edit, Delete).
- Dynamic forms adapt based on component category (e.g., socket type for CPUs, wattage for PSUs).

The architecture follows a clean separation between frontend, backend, and database layers, ensuring maintainability and scalability.

Facts Finding Techniques

To design a system that accurately reflects real-world PC builder workflows, several fact-finding techniques were used:

1. Market Analysis

Existing platforms like PCPartPicker were studied to understand:

- Feature expectations
- Compatibility rules
- User interface patterns

This analysis helped define essential features such as smart filtering and real-time pricing.

2. Hardware Research

Component attributes like:

- CPU socket types (e.g., LGA1700, AM4/AM5)
- RAM generations (DDR4, DDR5)
- Form factors and wattage requirements

were studied to create accurate compatibility rules and database schema.

3. User Feedback

Potential users (students, PC enthusiasts) provided insights on:

- Need for a dark theme
- Importance of budget tracking
- Requirement to save builds and export them

These findings shaped the UX and key features of the application.

Feasibility Study

A feasibility study was conducted to ensure the project can be developed effectively with available resources.

Economic Feasibility

PC Forge is cost-effective because:

- Built using **open-source technologies** (PHP, MySQL, Apache).
- Requires no licensing fees.
- Hosting and deployment costs are minimal.

Primary investment is developer time, which makes the system economically viable for academic and practical use.

Technical Feasibility

The project uses well-established technologies:

- PHP backend
- MySQL database (PDO integration)
- Vanilla JavaScript for UI logic
- HTML/CSS for dark-mode visual design

The team has the required skill set, and no advanced or experimental technologies are involved.

Therefore, the project is fully technically feasible.

Operational Feasibility

PC Forge is easy to understand and operate due to:

- Guided component selection
- Familiar dark UI theme
- Clear instructions and real-time feedback
- No training required for users

For admins, CRUD operations are simple and efficient. Hence, operational feasibility is high.

Schedule Feasibility

The project was divided into manageable tasks:

- Requirements + DB design
- Backend module development
- UI development
- Integration & Testing
- Documentation

This modular approach ensured timely completion, making it schedule-feasible.

Motivational Feasibility

The motivation for this project is naturally high because:

- PC building is a growing interest area.
- Users enjoy customizing their systems.
- The project solves real-world problems faced by beginners and enthusiasts.

This motivation supported consistent progress throughout development.

Cost and Benefit Analysis

Costs

- Initial development time (UI, backend, testing).
- Minimal hosting and server requirements.
- Optional cost for maintaining or expanding part inventory.

Benefits

- Prevents costly hardware compatibility mistakes.
- Saves user time through automated filtering.
- Helps admins maintain an organized database.
- Provides downloadable build reports via PDF.
- Can be extended for commercial use or affiliate marketing.

The benefits significantly outweigh the minimal development and hosting costs, making this system highly valuable.

System Analysis

System Analysis focuses on understanding how the PC Forge system operates internally, how different functions interact, and what requirements must be met for smooth execution. This chapter breaks down the builder workflow, system functions, testing strategy, and functional requirements.

PC Builder Flow Overview

1. **Start Build**
   Users begin creating a new PC configuration.

2. **Set Budget (Optional)**
   The system activates budget tracking and visual warnings if the spending limit is exceeded.

3. **Select CPU**

   - User chooses a processor.
   - The system records the CPU's **socket type**.

4. **Select Motherboard**

   - The system filters motherboards compatible with the CPU socket.
   - Only valid options are shown.

5. **Select RAM**

   - Motherboard-selected RAM type (DDR4/DDR5) determines available options.
   - Prevents mismatched RAM generation.

6. **Select GPU, Storage, PSU, Case**

   - Each module updates total price in real time.
   - Compatibility constraints (e.g., PSU wattage, interface type) guide options.

7. **Review & Save Build**

   - Users authenticate (login/register) if not already logged in.
   - Build details are stored in the database for future access.

8. **Export PDF**

   - Final configuration can be downloaded as a formatted PDF.

This flow ensures a *guided*, *error-free*, and *user-friendly* PC building experience.

Function Details

The system consists of multiple functional modules, each designed to perform a specific role in the PC-building and administrative workflow.

1. Authentication Module

- Handles **user registration and login**.
- Passwords are stored securely using `password_hash()` in PHP.
- Ensures that build-saving features are restricted to valid users.

2. Smart Compatibility Engine

The core logic that differentiates PC Forge from traditional catalog-based systems.

- CPU selection automatically filters motherboard options.
- Motherboard selection filters RAM type.
- Additional constraints (e.g., PSU wattage) guide further steps.
- Dynamic SQL queries ensure only compatible components are shown.

3. Real-Time Pricing and Budget Tracking

- Every selected component updates the total build price instantly.
- Visual indicators warn users if the budget is exceeded.
- Highly useful for users planning builds within financial limits.

4. Build Management

- Users can save multiple builds to their account.
- Builds are retrieved from the database and displayed in the "My Builds" section.
- PDF export provides a physical or printable summary of builds.

5. Admin Module

Admins have full control over system data:

- Manage hardware components (Add / Edit / Delete).
- Access inventory through a categorized interface.
- Admin dashboard displays system statistics, including:
  - Total users
  - Total parts
  - Total builds
- Dynamic forms adapt to the component category.

6. Inventory & Category Management

- All parts belong to specific categories (CPU, RAM, GPU, etc.).

- Admins can update component attributes such as price, socket type, RAM generation, wattage, etc.

Testing

Testing ensures the system functions correctly under normal and edge-case scenarios. PC Forge uses multiple types of testing to validate reliability, performance, and usability

Types of Testing

1. Unit Testing

- Database connection functions
- Authentication helpers
- Component filtering utilities

Tested individually to ensure correctness

2. Integration Testing

Ensures modules work together seamlessly:

- Builder ↔ Database
- Authentication ↔ User Build Storage
- Admin CRUD ↔ Component Database

3. System Testing

End-to-end validation of:

- Registration → Build Creation → Save → PDF Export
- Full builder flow using multiple component combinations

4. Compatibility Testing

- Ensures UI works across browsers: Chrome, Firefox, Edge.
- Verifies that compatibility filtering handles various component constraints.

5. User Acceptance Testing (UAT)

- Conducted with actual users to validate:
- Dark theme usability
- Navigation clarity
- Step-by-step guided flow

Functional Requirements

Functional requirements define what the system **must** do to operate effectively.

User-Side Requirements

1. System must allow users to register using a unique username.
2. System must allow login/logout to manage build-related features.
3. Users must be able to start a new build and select parts step-by-step.
4. System must automatically prevent incompatible component selections.
5. System must display real-time pricing and budget alerts.
6. System must allow users to save builds and export them as PDF.

Admin-Side Requirements

1. System must allow admins to add, edit, delete hardware components.
2. Admin panel must provide statistics such as number of users, builds, and parts.
3. Category-based dynamic forms must be available for managing various attributes.
4. System must enable complete control over the inventory database.

System Requirements

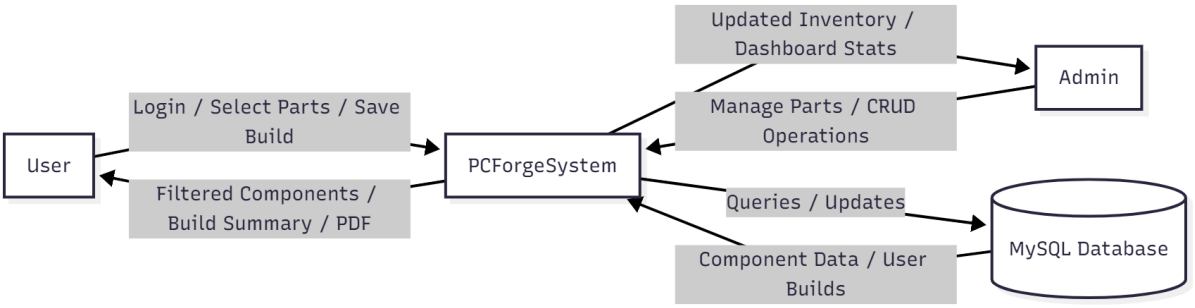
1. System must support PHP backend and MySQL database operations via PDO.
2. UI must be responsive and compatible with major browsers.
3. System must follow a modular structure for maintainability.
4. Dark theme must be consistent across frontend pages.
5. System must support secure password handling.

System Development Cycle

The System Development Cycle describes how PC Forge processes data, interacts with users, and structures internal operations. This chapter represents the complete data flow using **Context Level DFD**, **Zero Level DFD**, **First Level DFD**, and the **ER Diagram**, all in Mermaid notation..

Context Level DFD (Level 0)

The Context DFD visualizes PC Forge as a single high-level process interacting with three primary entities: **User**, **Admin**, and **Database**.



Explanation

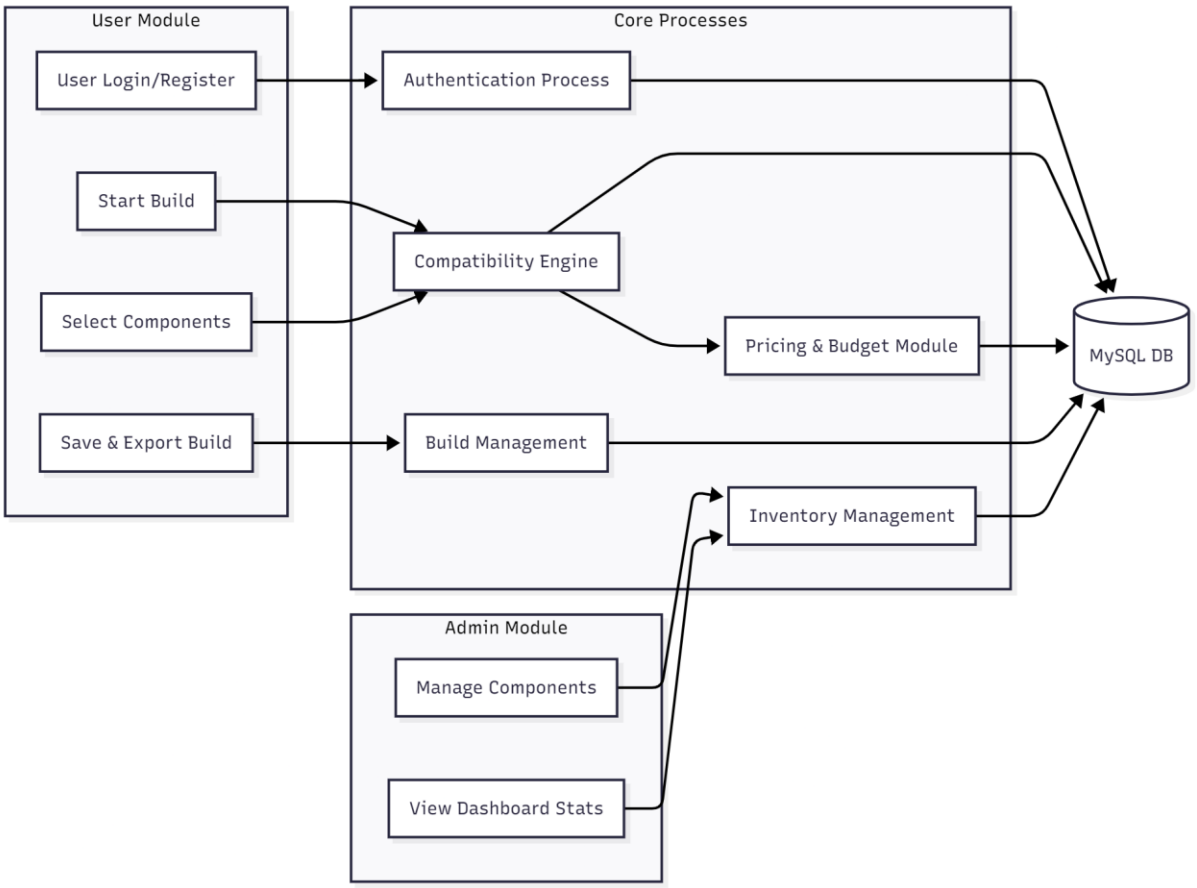
- Users interact with the system to build PCs, view compatible components, and save builds

- Admins manage component inventory and system data.

- The system communicates with the database for all operations involving retrieval and storage.

Zero Level DFD (Level 1 DFD)

This level breaks PC Forge into its major functional subsystems.



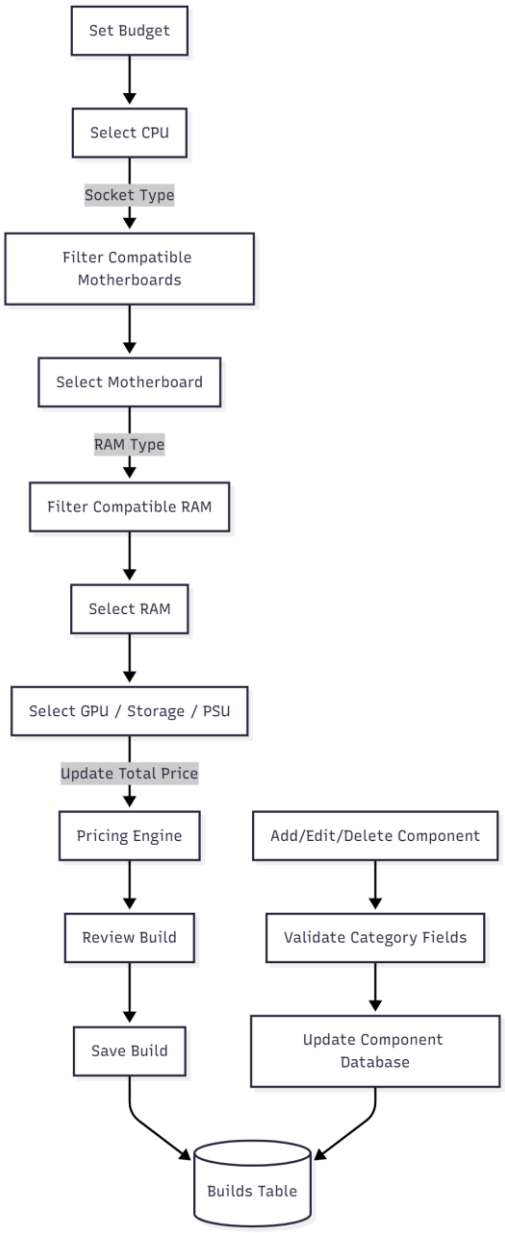
Explanation of Processes

The system is divided into:

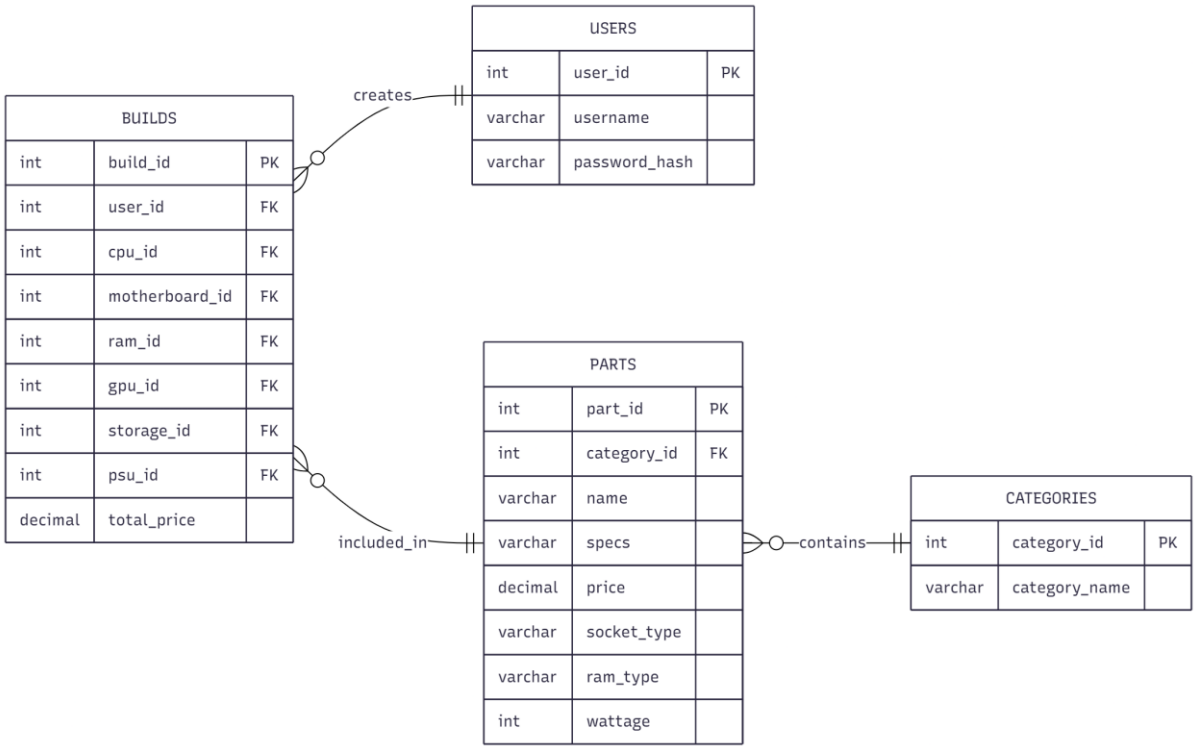
- **User actions** such as selecting parts and saving builds.
- **Admin actions** such as modifying inventory.
- **Core internal processes** like compatibility filtering, pricing, authentication, and inventory management.

First Level DFD (Level 2 DFD)

This level explains detailed internal operations, especially within the **Builder Process** and **Admin Inventory Process**.



ER-Diagram



Explanation

- A **User** can have multiple **Builds**.
- Each **Build** stores references to selected parts (CPU, RAM, GPU, etc.).
- **Categories** organize components into types like CPU, Motherboard, RAM.
- **Parts** belong to a single category and contain attributes used for compatibility filtering.

Data Dictionary

The Data Dictionary provides a detailed description of the data elements, database tables, fields, and their relationships within the PC Forge system. It serves as a technical reference for developers, ensuring clarity, consistency, and correctness in database operations.

PC Forge uses a **MySQL relational database**, structured to support compatibility filtering, build management, user authentication, and admin inventory functions.

Database Overview

PC Forge contains the following primary tables:
1. **users** – Stores user authentication data.
2. **builds** – Stores saved PC builds for each user.
3. **categories** – Stores hardware categories such as CPU, RAM, GPU, etc.
4. **parts** – Stores detailed hardware component information.

This modular schema enables efficient querying, compatibility filtering, and data integrity.

Table-Level Data Dictionary

Below is the formal description of each table and field used in the PC Forge system.

Table 1: users

Field Name	Data Type	Description
user_id (PK)	INT (Auto Inc.)	Unique identifier for each user.
username	VARCHAR(255)	Unique username for login.
password_hash	VARCHAR(255)	Secure hashed password generated using password_hash().

Table 2: builds

Field Name	Data Type	Description
build_id (PK)	INT (Auto Inc.)	Unique identifier for each saved build.
user_id (FK)	INT	Links build to its owner (users table).
cpu_id	INT (FK)	Selected CPU component.
motherboard_id	INT (FK)	Selected Motherboard component.
ram_id	INT (FK)	Selected RAM component.
gpu_id	INT (FK)	Selected GPU component.

Field Name	Data Type	Description
storage_id	INT (FK)	Selected Storage component.
psu_id	INT (FK)	Selected Power Supply Unit.
total_price	DECIMAL(10,2)	Final calculated price of the build.

Table 3: categories

Field Name	Data Type	Description
category_id (PK)	INT (Auto Inc.)	Unique identifier for category.
category_name	VARCHAR(255)	Category name (e.g., CPU, RAM, GPU).

Table 4: parts

Field Name	Data Type	Description
part_id (PK)	INT (Auto Inc.)	Unique identifier for hardware component.
category_id (FK)	INT	Links part to a category.
name	VARCHAR(255)	Component name (e.g., Intel i9-12900K).
specs	TEXT	Key specifications (clock speed, memory size, etc.).
price	DECIMAL(10,2)	Component price used for real-time updates.
socket_type	VARCHAR(255)	Used for CPU–Motherboard compatibility.
ram_type	VARCHAR(255)	Used for Motherboard–RAM compatibility.
wattage	INT	Used for PSU compatibility calculations.

System Design (Technical Overview)

System design describes how the overall architecture, UI, backend logic, and database interact.

Architecture Type:

- Presentation Layer: HTML5, CSS (Dark Theme), JavaScript.
- Logic Layer: PHP (Core), Compatibility Engine, Filtering Logic.
- Data Layer: MySQL database using PDO connection.

Design Goals:

- Modular structure for easy maintenance.
- Component-based filtering driven by database attributes.
- User-friendly UI with instant price updates.
- Secure authentication and data handling.

Scheduling

The project was completed in planned phases to ensure systematic development.

Phase	Description	Duration
Requirement Gathering	User needs, PC compatibility rules, workflow planning	Week 1
Database Design	Schema creation using SQL	Week 1
Backend Development	Authentication, CRUD, builder logic	Week 2
Frontend Development	UI creation, dark theme styling, JS logic	Week 3
Testing & Integration	Functional & system testing	Week 4
Documentation	Project report, diagrams, code summary	Final Phase

Estimate of Efforts

Task Category	Effort (%)	Description
Requirement Analysis	15%	Understanding flow, compatibility rules
Design	20%	UI/UX, architecture, diagrams
Development	40%	PHP backend, JS logic, CRUD operations
Testing	20%	Validation of builder, admin, filtering
Documentation	5%	Report preparation & formatting

Modules of PC Forge

1. Category Selection Module
Displays selectable component groups (CPU, RAM, GPU, etc.).

2. Part Listing Module
Shows filtered components based on compatibility and category.

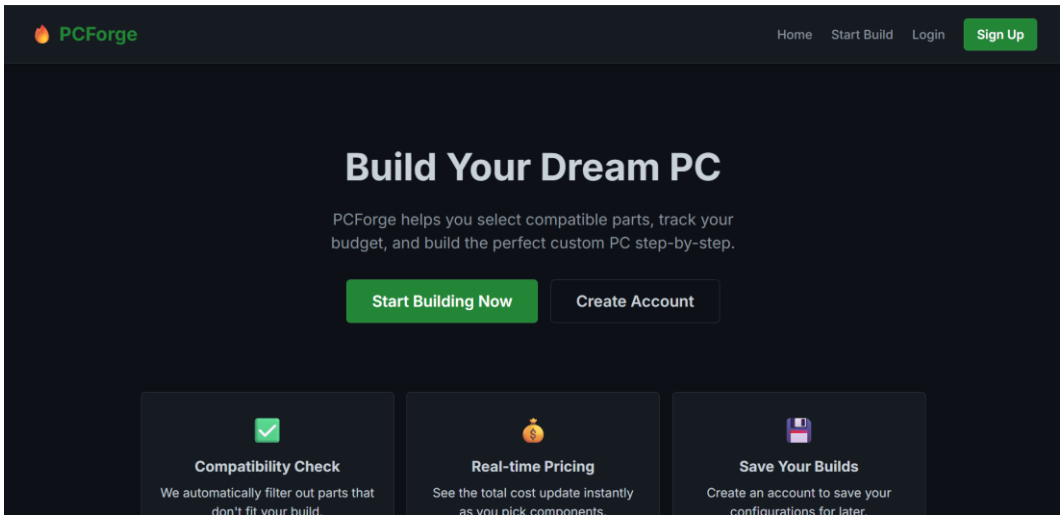
3. Compatibility Engine
Ensures only compatible parts (socket type, RAM type, wattage) are displayed.

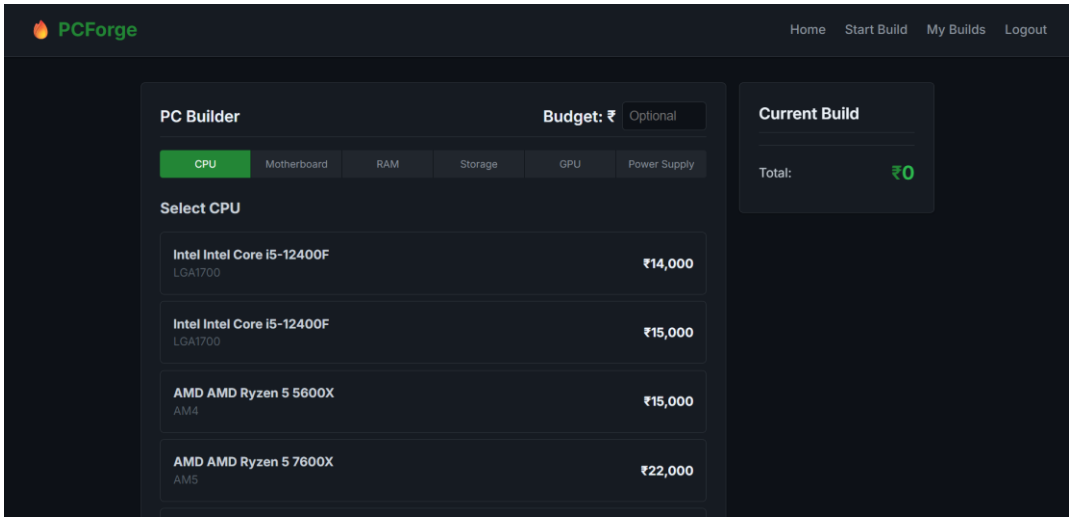
4. Summary & Pricing Module
Calculates total price dynamically as users add parts.

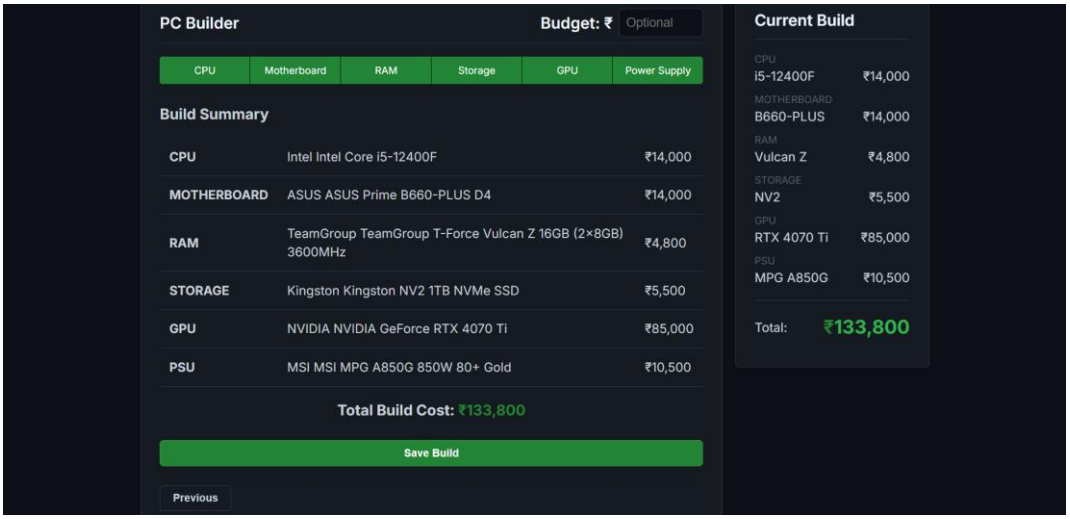
5. User Build Module
Allows saving, viewing, and exporting user builds.

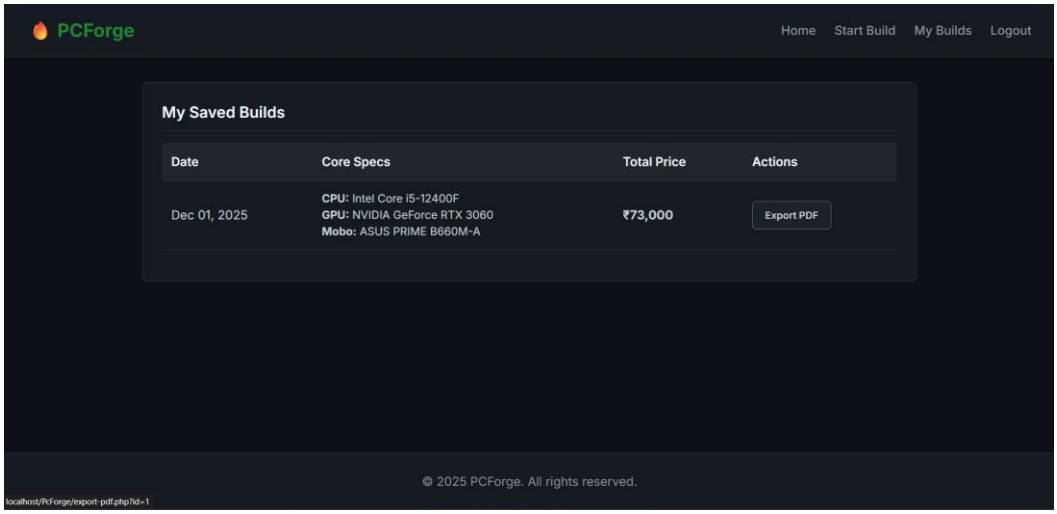
6. Admin Inventory Module
Allows managing parts, categories, and dashboard data.

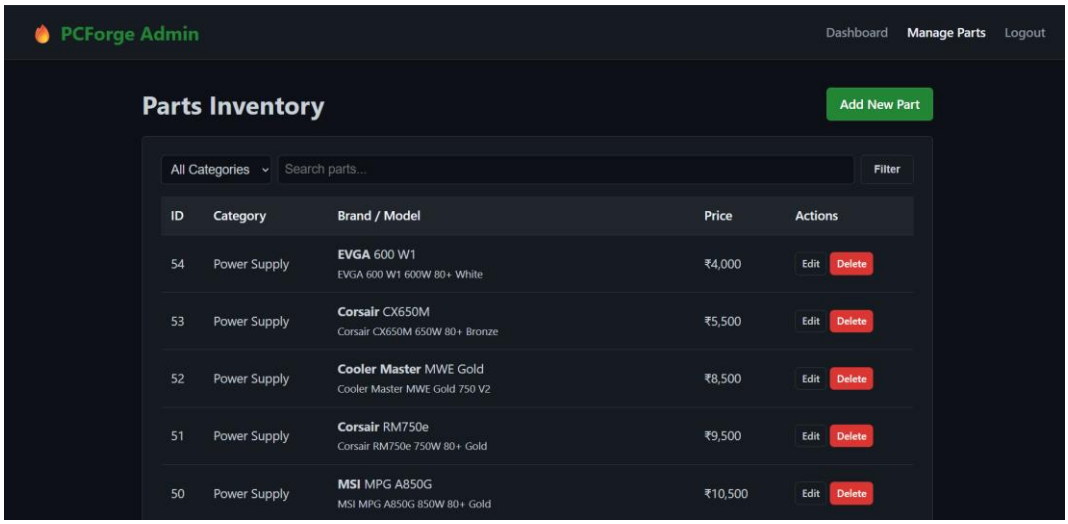
Screenshots (UI)

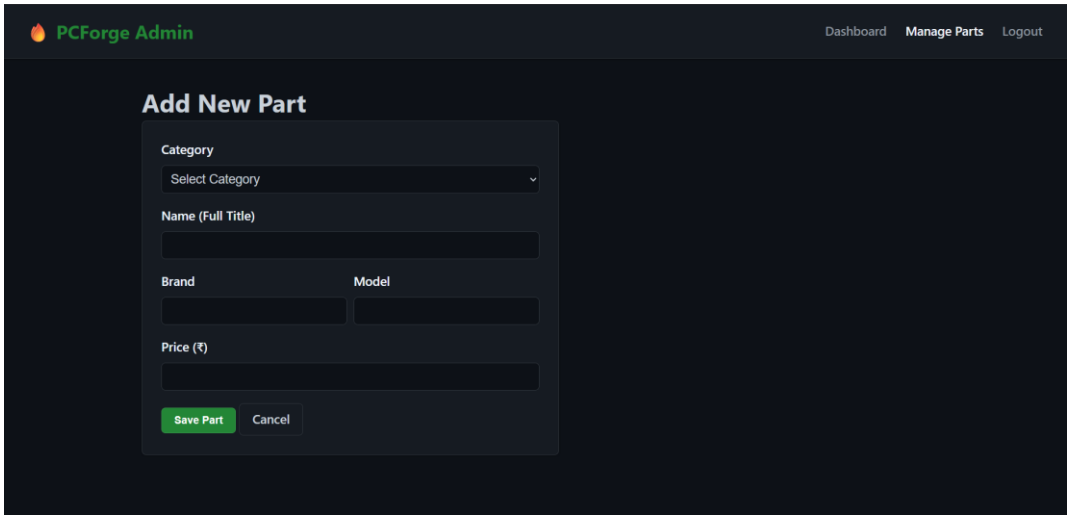







PCForge
Home
Start Build
My Builds
Logout
My Saved Builds
Date
Core Specs
Total Price
Actions
Dec 01, 2025
CPU: Intel Core i5-12400F
GPU: NVIDIA GeForce RTX 3060
Mobo: ASUS PRIME B660M-A
₹73,000
Export PDF
© 2025 PCForge. All rights reserved.
localhost/PcForge/export-pdf.php?id=1


PCForge Admin
Dashboard
Manage Parts
Logout
Parts Inventory
Add New Part
All Categories
Search parts...
Filter
ID
Category
Brand / Model
Price
Actions
54
Power Supply
EVGA 600 W1
EVGA 600 W1 600W 80+ White
₹4,000
Edit
Delete
53
Power Supply
Corsair CX650M
Corsair CX650M 650W 80+ Bronze
₹5,500
Edit
Delete
52
Power Supply
Cooler Master MWE Gold
Cooler Master MWE Gold 750 V2
₹8,500
Edit
Delete
51
Power Supply
Corsair RM750e
Corsair RM750e 750W 80+ Gold
₹9,500
Edit
Delete
50
Power Supply
MSI MPG A850G
MSI MPG A850G 850W 80+ Gold
₹10,500
Edit
Delete


PCForge Admin
Dashboard
Manage Parts
Logout
Add New Part
Category
Select Category
Name (Full Title)
Brand
Model
Price (₹)
Save Part
Cancel

Source Code

Builder.php :

```
<?php
require_once 'includes/db.php';
require_once 'includes/user-auth.php';

// Fetch all categories
$stmt = $pdo->query("SELECT * FROM categories ORDER BY id");
$categories = $stmt->fetchAll();

// Fetch all parts
$stmt = $pdo->query("SELECT * FROM parts ORDER BY price ASC");
$all_parts = $stmt->fetchAll();

// Group parts by category slug
$parts_by_category = [];
foreach ($categories as $cat) {
$parts_by_category[$cat['slug']] = [];
}

foreach ($all_parts as $part) {
// Find category slug for this part
foreach ($categories as $cat) {
if ($cat['id'] == $part['category_id']) {
$parts_by_category[$cat['slug']][] = $part;
break;
}
}
}

// Handle Save Build
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['save_build'])) {
requireLogin();

$budget = !empty($_POST['budget']) ? $_POST['budget'] : null;
$total_price = $_POST['total_price'];
$cpu_id = $_POST['cpu_id'] ?: null;
$mobo_id = $_POST['motherboard_id'] ?: null;
$ram_id = $_POST['ram_id'] ?: null;
$storage_id = $_POST['storage_id'] ?: null;
$gpu_id = $_POST['gpu_id'] ?: null;
$psu_id = $_POST['psu_id'] ?: null;

$stmt = $pdo->prepare("INSERT INTO builds (user_id, budget, total_price, cpu_id, motherboard_id, ram_id, storage_id, gpu_id, psu_id) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");
```

```
  if ($stmt->execute([$_SESSION['user_id'], $budget, $total_price, $cpu_id, $mobo_id, $ram_id, $storage_id, $gpu_id, $psu_id])) {
  header("Location: my-builds.php");
  exit();
  } else {
  $error = "Failed to save build.";
  }
  }

  include 'includes/header.php';
  ?>

  <script>
  // Pass PHP data to JS
  const categories = <?php echo json_encode($categories); ?>;
  const partsData = <?php echo json_encode($parts_by_category); ?>;
  </script>

  <div class="d-flex gap-2" style="align-items: flex-start;">
  <!-- Main Builder Area -->
  <div style="flex: 3;">
  <div class="card">
  <div class="card-header d-flex justify-between align-center">
  <span>PC Builder</span>
  <div class="d-flex align-center gap-1">
  <label for="budget-input">Budget: ₹</label>
  <input type="number" id="budget-input" class="form-control" style="width: 120px;" placeholder="Optional">
  </div>
  </div>

  <!-- Progress Bar -->
  <div style="display: flex; margin-bottom: 1.5rem; background: #21262d; border-radius: 4px; overflow: hidden;">
  <?php foreach ($categories as $index => $cat): ?>
  <div id="step-indicator-<?php echo $cat['slug']; ?>" class="step-indicator">
  <?php echo $cat['name']; ?>
  </div>
  <?php endforeach; ?>
  </div>

  <!-- Steps -->
  <div id="builder-steps">
  <!-- Steps will be generated by JS or we can hardcode containers -->
  <?php foreach ($categories as $cat): ?>
  <div id="step-<?php echo $cat['slug']; ?>" class="builder-step">
  <h3 class="mb-2">Select <?php echo $cat['name']; ?></h3>
  <div id="list-<?php echo $cat['slug']; ?>">
  <!-- Parts list will be injected here -->
  </div>
```

```
</div>
<?php endforeach; ?>

<!-- Summary Step -->
<div id="step-summary" class="builder-step">
<h3 class="mb-2">Build Summary</h3>
<div id="summary-content"></div>

<?php if (isLoggedIn()): ?>
<form method="POST" id="save-form" class="mt-3">
<input type="hidden" name="save_build" value="1">
<input type="hidden" name="budget" id="input-budget">
<input type="hidden" name="total_price" id="input-total">
<input type="hidden" name="cpu_id" id="input-cpu">
<input type="hidden" name="motherboard_id" id="input-motherboard">
<input type="hidden" name="ram_id" id="input-ram">
<input type="hidden" name="storage_id" id="input-storage">
<input type="hidden" name="gpu_id" id="input-gpu">
<input type="hidden" name="psu_id" id="input-psu">
<button type="submit" class="btn btn-primary w-100">Save Build</button>
</form>
<?php else: ?>
<div class="mt-3 text-center">
<a href="login.php" class="btn btn-primary">Login to Save Build</a>
</div>
<?php endif; ?>
</div>
</div>

<div class="mt-3 d-flex justify-between">
<button id="btn-prev" class="btn btn-outline" style="visibility: hidden;">Previous</button>
<button id="btn-next" class="btn btn-primary" disabled>Next</button> <!-- Disabled until selection made -->
</div>
</div>
</div>

<!-- Sidebar Summary -->
<div style="flex: 1;">
<div class="card" style="position: sticky; top: 1rem;">
<div class="card-header">Current Build</div>
<ul id="mini-summary" class="mb-2">
<!-- Selected parts list -->
</ul>
<div style="border-top: 1px solid var(--color-border); padding-top: 1rem;">
<div class="d-flex justify-between align-center mb-1">
<span>Total:</span>
<span class="summary-total">₹<span id="total-price">0</span></span>
</div>
```

```
  <div id="budget-status" class="budget-warning" style="display: none;">
  Over Budget!
  </div>
  </div>
  </div>
  </div>
  </div>

  <script src="assets/js/builder.js"></script>

  <?php include 'includes/footer.php'; ?>
```

admin/dashboard.php :

```
<?php
require_once '../includes/db.php';
require_once '../includes/admin-auth.php';

requireAdminLogin();

// Stats
$parts_count = $pdo->query("SELECT COUNT(*) FROM parts")->fetchColumn();
$users_count = $pdo->query("SELECT COUNT(*) FROM users")->fetchColumn();
$builds_count = $pdo->query("SELECT COUNT(*) FROM builds")->fetchColumn();

?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Admin Dashboard - PC Forge</title>
<link rel="stylesheet" href="../assets/css/style.css">
</head>
<body>
<header>
<div class="logo">
<a href="dashboard.php">🔥 <span>PC Forge Admin</span></a>
</div>
<nav>
<ul>
<li><a href="dashboard.php" class="active">Dashboard</a></li>
<li><a href="parts-list.php">Manage Parts</a></li>
<li><a href="logout.php">Logout</a></li>
</ul>
</nav>
</header>
<main>
<h1>Dashboard</h1>
```

```
<div class="d-flex gap-2 mt-3">
<div class="card w-100 text-center">
<h3>Total Parts</h3>
<div style="font-size: 2rem; font-weight: bold; color: var(--color-primary);"><?php
echo $parts_count; ?></div>
</div>
<div class="card w-100 text-center">
<h3>Registered Users</h3>
<div style="font-size: 2rem; font-weight: bold; color: var(--color-primary);"><?php
echo $users_count; ?></div>
</div>
<div class="card w-100 text-center">
<h3>Total Builds Saved</h3>
<div style="font-size: 2rem; font-weight: bold; color: var(--color-primary);"><?php
echo $builds_count; ?></div>
</div>
</div>

<div class="mt-3">
<h3>Quick Actions</h3>
<div class="d-flex gap-2 mt-1">
<a href="parts-add.php" class="btn btn-primary">Add New Part</a>
<a href="parts-list.php" class="btn btn-outline">View All Parts</a>
</div>
</div>
</main>
</body>
</html>
```

my-builds.php :

```
<?php
require_once 'includes/db.php';
require_once 'includes/user-auth.php';

requireLogin();

$stmt = $pdo->prepare("
    SELECT b.*,
           cpu.name as cpu_name,
           gpu.name as gpu_name,
           mobo.name as mobo_name
    FROM builds b
    LEFT JOIN parts cpu ON b.cpu_id = cpu.id
    LEFT JOIN parts gpu ON b.gpu_id = gpu.id
    LEFT JOIN parts mobo ON b.motherboard_id = mobo.id
    WHERE b.user_id = ?
    ORDER BY b.created_at DESC
");
```

```
$stmt->execute([$_SESSION['user_id']]);
$builds = $stmt->fetchAll();

include 'includes/header.php';
?>

<div class="card">
  <div class="card-header">My Saved Builds</div>

  <?php if (count($builds) === 0): ?>
        <p class="text-center p-3">You haven't saved any builds yet. <a
href="builder.php" style="color: var(--color-primary);">Start building!</a></p>
  <?php else: ?>
    <table class="table">
      <thead>
        <tr>
          <th>Date</th>
          <th>Core Specs</th>
          <th>Total Price</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        <?php foreach ($builds as $build): ?>
          <tr>
            <td><?php echo date('M d, Y', strtotime($build['created_at'])); ?></td>
            <td>
              <div style="font-size: 0.9rem;">
                                    <strong>CPU:</strong> <?php echo
htmlspecialchars($build['cpu_name'] ?? 'N/A'); ?><br>
                                    <strong>GPU:</strong> <?php echo
htmlspecialchars($build['gpu_name'] ?? 'N/A'); ?><br>
                                    <strong>Mobo:</strong> <?php echo
htmlspecialchars($build['mobo_name'] ?? 'N/A'); ?>
              </div>
            </td>
                          <td style="font-weight: bold;">₹<?php echo
number_format($build['total_price']); ?></td>
            <td>
                <a href="export-pdf.php?id=<?php echo $build['id']; ?>"
target="_blank" class="btn btn-outline" style="font-size: 0.8rem;">Export PDF</a>
            </td>
          </tr>
        <?php endforeach; ?>
      </tbody>
    </table>
  <?php endif; ?>
</div>

<?php include 'includes/footer.php'; ?>
```

Limitations and Future Enhancements

Limitations

PC Forge is a powerful platform that simplifies the process of custom PC building through compatibility filtering, real-time pricing, and build management. However, like any software system, it has certain constraints and areas that can be improved in future versions.

1. Manual Data Entry for Inventory
All hardware components must be manually added, updated, or removed by the admin. There is **no real-time integration with external retailers**, meaning prices and stock details must be manually updated.

2. Limited Compatibility Rules
Current compatibility checking covers basic relationships such as:

- CPU socket compatibility with motherboard
- RAM generation compatibility (DDR4 / DDR5)

More advanced compatibility parameters—such as **GPU clearance**, **cooler TDP support**, or **PSU wattage headroom**—are not included.

3. Single-Currency Price Support
PC Forge currently uses a single currency (₹) hardcoded in views.
It does not support multi-currency display or automated conversion.

4. Limited Build Analytics
The system helps users save and export builds but does not provide:

- Performance predictions
- Recommended alternatives
- Benchmark-based insights

This reduces the depth of assistance provided to advanced users.

5. Basic PDF Export Format
While PC Forge includes PDF export functionality, the output is simple and best suited for basic documentation purposes.
More visually rich export formats are not yet implemented.

6. No Mobile App Version
PC Forge is web-based only.
There is no Android or iOS application for mobile-first users.

7. No Community or Sharing Features
Users cannot:

- Share builds publicly
- Get feedback from others
- Browse trending builds

Community-driven engagement is not yet a part of the system.

Future Enhancements

To make PC Forge more feature-rich, scalable, and aligned with modern PC-building platforms, the following enhancements are recommended:

1. API Integration for Real-Time Pricing
Integrate APIs from:
- Amazon
- Newegg
- PC component retailers

This would automate:
- Fetching live prices
- Checking availability
- Updating inventory

2. Advanced Compatibility Engine
Future versions could include:
- PSU wattage recommendations
- GPU size vs. cabinet clearance
- CPU cooler support matrix
- M.2 slot compatibility rules

This would significantly improve precision and reduce user errors.

3. Multi-Currency & Regional Support
Enable:
- Dynamic currency switching
- Regional pricing models
- Localization for multiple languages

This is especially useful if PC Forge expands internationally.

4. Community Build Sharing
Allow users to:
- Make builds public
- Like/comment on builds
- Follow other builders

This transforms PC Forge into a collaborative platform rather than a standalone tool.

5. AI-Based Recommendations
Implement an AI engine to:
- Suggest the most cost-effective parts
- Recommend performance-balanced builds
- Optimize builds for gaming, editing, or workstation workloads

Such capabilities enhance usability for beginners.

6. Inventory Automation for Admin Panel
Add features such as:
- Import/export CSV for batch updates
- Notifications for outdated or missing component details
- Auto-refreshing database sync via cron jobs

This will reduce admin workload and improve accuracy.

7. Enhanced PDF Export Templates
Support premium-quality downloadable build sheets with:
- Images
- Performance charts
- Component compatibility summary

This would make PC Forge more appealing for presentation or client use.

8. Mobile App Development
Create Android and iOS applications with:
- Local build storage
- Push notifications
- Offline viewing of saved builds

This increases accessibility for mobile-centric users.

9. Browser Extension for Quick PC Part Lookup
A browser extension could allow users to:
- Highlight PC parts on e-commerce sites
- Check compatibility with their active build

This elevates PC Forge beyond static selection workflows.

REFERENCES

- **PHP Documentation**, Official PHP Manual
Available at: https://www.php.net/docs.php
(Referenced for backend development, password hashing, and PDO integration.)

- **MySQL Reference Manual**, Oracle MySQL
Available at: https://dev.mysql.com/doc/
(Referenced for designing database schema, SQL queries, and relational integrity.)

- **MDN Web Docs (HTML, CSS, JavaScript)**
Available at: https://developer.mozilla.org/
(Used for frontend development standards, UI design patterns, and JavaScript DOM handling.)

- **PCPartPicker (Feature Inspiration)**
(Reviewed for understanding existing PC builder tools, compatibility workflows, and user experience patterns.)