Web Development and PHP Application Project Report

Course Name: PHP Application Development

Course Code: 03CIT4054

Student Name: Au Pak Yeung, Ng Yu Ham Baldwin

Table of Contents:

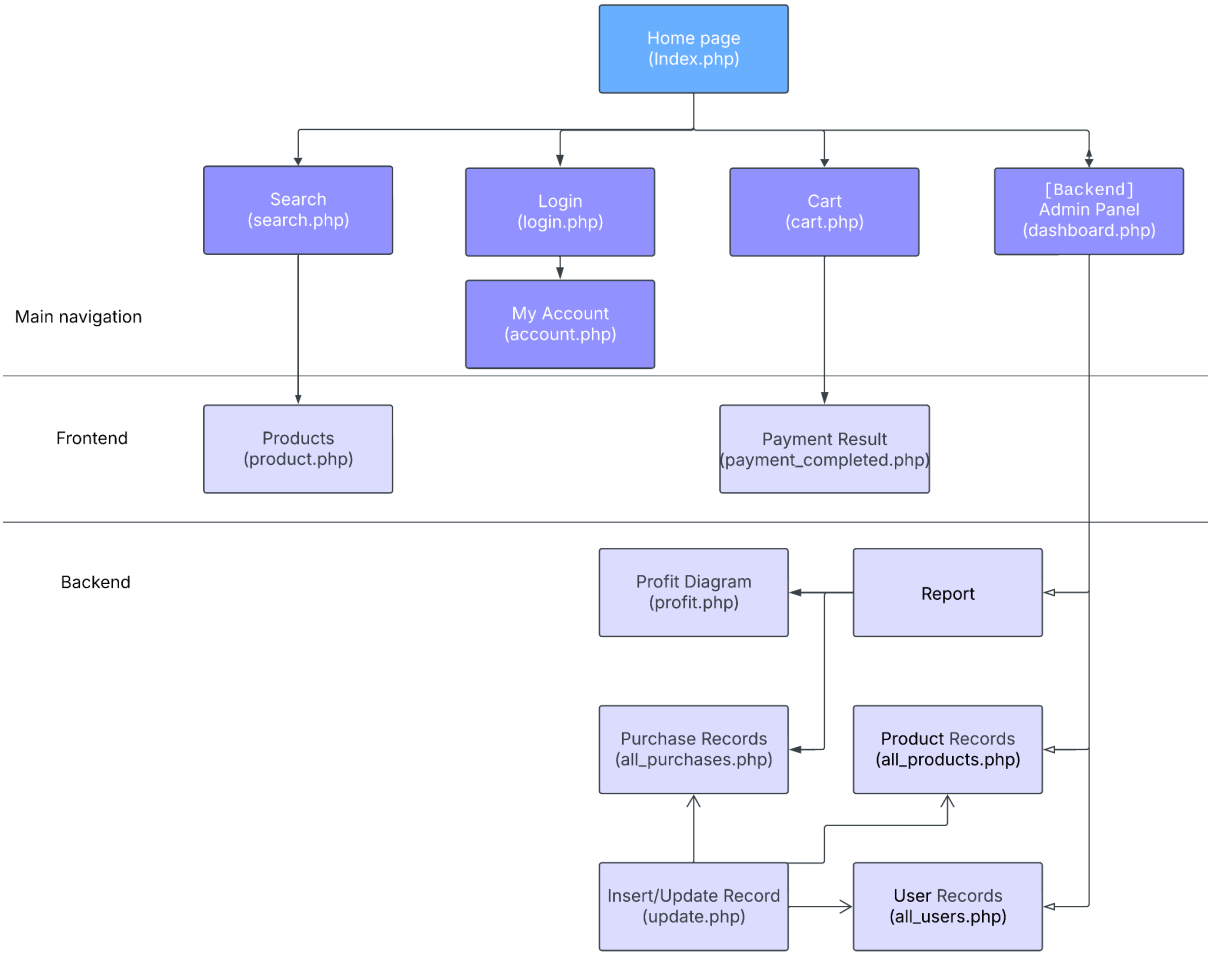
Introduction

The purpose of this PHP-based website project is to design and implement a full-stack e-commerce application that tries to mimic the structure and design philosophy of platforms such as Steam. This website is conceptualised as a digital distribution service for software products, where users can register, browse items, add them to their carts, and proceed to a secure payment process. The frontend is developed using HTML, CSS, JavaScript, and Bootstrap 5, while the backend architecture integrates both traditional PHP server-side scripting and modern cloud-based services, including Firebase and Node.js. A key motivation for this hybrid approach is to demonstrate the practical benefits of leveraging cloud services, APIs, and SDKs within a conventional server-side architecture.

The project not only allows for the management of user accounts and product listings but also supports secure Google OAuth authentication via Firebase. Administrative features such as user deletion, session management, and database synchronisation are implemented through the Firebase Admin SDK via Composer. Additional security is enforced by selectively integrating reCAPTCHA v2 based on real-time IP threat intelligence powered by ipdata.co, along with the SSL cURL protected by CA certificates.

The system incorporates modular design principles and extensively uses internal API routing, allowing for reusable code and separation of concerns across logical layers. All backend communications follow a RESTful structure, ensuring both human readability and developer extensibility. The project reflects a balance between academic learning and real-world applicability, with support for scalable components and future expansion, such as stock systems and data reporting.

Site Map


Home page
(Index.php)
Search
(search.php)
Login
(login.php)
Cart
(cart.php)
[Backend]
Admin Panel
(dashboard.php)
My Account
(account.php)
Main navigation
Frontend
Products
(product.php)
Payment Result
(payment_completed.php)
Backend
Profit Diagram
(profit.php)
Report
Purchase Records
(all_purchases.php)
Product Records
(all_products.php)
Insert/Update Record
(update.php)
User Records
(all_users.php)

Features

The core features of the project include:

- Google OAuth Authentication: Implemented via Firebase CDN and JavaScript SDK. New users can sign in with Google credentials, and authenticated sessions are tracked across the site.

- Product Display and Browsing: Products are retrieved from Firebase Firestore and rendered dynamically using JavaScript. Each product card includes images, titles, and prices.

- Shopping Cart System: A persistent cart powered by local storage and integrated with Stripe via Node.js to facilitate payments.

- Purchases Collection: Transactions are logged in Firestore for backend administrative review and dynamic frontend updates.

- User Account Management: Users can view and delete their account, with Firebase Admin SDK allowing administrators to permanently delete Firebase Authentication records.

- Admin Panel Features: Admin users can browse all registered users, edit or delete them, and manage product listings.

- Internal API Routing: PHP classes like `Class_users.php`, Class_products.php, and `Class_purchases.php` interact with Firestore via a trait in `firestore.php`. These classes expose endpoints such as `/user/{id}` or `/uid/{uid}`.

- reCAPTCHA Enforcement: A reCAPTCHA validation is enforced selectively for suspicious IPs, enhancing security against bots and attackers.

Workflow / Scenarios / Wireframe

Start ➔

→ Search Page (Show product listing)
- ↳ Click product ➔ Open Product Detail Modal
- ↳ Click 'Add to Cart' ➔ Save to Local Storage

→ Login Page (if not logged in)
- ↳ Google Sign-In Popup
- ↳ (If suspicious IP ➔ reCAPTCHA required)
- ↳ After login ➔ Redirect to Account

→ My Account
- ↳ View Account Info
- ↳ Change Password
- ↳ Delete Account
  - ↳ Confirm reauthentication ➔ Delete Firebase OAuth + Firestore user data

→ Cart Page
- ↳ Show cart items (LocalStorage)
- ↳ Checkout Button
  - ↳ Call Stripe Node.js API
  - ↳ Redirect to Stripe Payment Page
  - ↳ Payment success/failure handling by Stripe in the backend

→ Dashboard (Only for Admin User)
- ↳ Manage Users
  - ↳ Edit User Info (manually)
  - ↳ Delete User (Firebase Admin + Firestore)
- ↳ Manage Products
  - ↳ Add/Edit/Delete Products
- ↳ Manage Purchase Records
  - ↳ See who purchased what

→ Chatbot Widget
- ↳Available globally, ask questions anytime

→ Logout
- ↳ Destroy session
- ↳ Firebase Sign-Out
- ↳ Redirect to Login

End.

(*I suggest just watching the video demonstration*)
Example 1: User Registration and Login via Google OAuth

1. The user navigates to the login page and clicks the "Login with Google" button.

2. Firebase Auth SDK triggers a Google login pop-up.

3. On success, the UID is sent to a PHP endpoint to sync with Firestore and create the account if it doesn't already exist.

Example 2: Product Purchase

1. A product is selected, added to the cart, and the user proceeds to checkout.

2. A Stripe-hosted page is invoked using a Node.js endpoint that generates a secure session.

3. On success, the client is redirected to a confirmation page, and the purchase record is stored in Firestore.

Example 3: Account Deletion

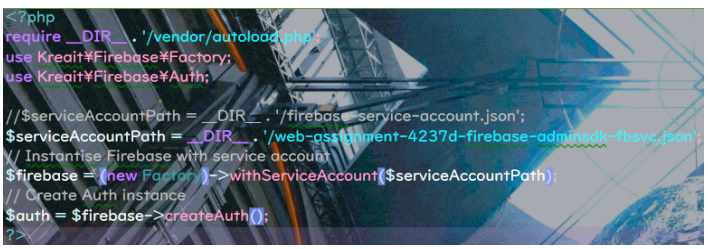
1. The user clicks the "Delete Account" button on the account page.

2. A reauthentication pop-up ensures secure deletion.

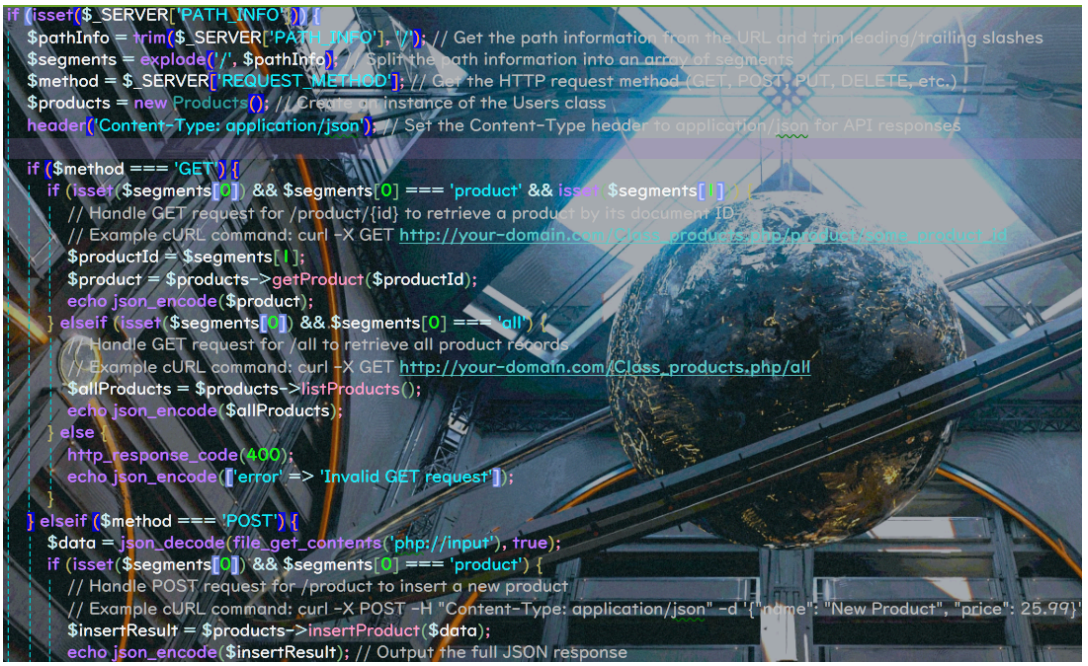
3. A DELETE request is sent to `Class_users.php/uid/{uid}`, and the record is removed from both Firebase Auth and Firestore.

Technical Specifications

- **Firebase Integration**: Firebase is initialised via CDN in `firebase_cdn.js`. Functions like `firebase.auth().signInWithPopup()` handles login, while Admin deletion uses the Kreait SDK in PHP (`firebase_admin.php`).



- **API Structure:** All backend requests are handled using RESTful internal APIs. Example: `GET /Class_products.php/all` returns all products. Each `Class_*.php` file routes based on the HTTP method and path segments.



- **Session and Security Handling:** PHP sessions are used in conjunction with Firebase Auth. `head.php` includes session validation, reCAPTCHA flags, and Sentry error tracking.

- **Data Formatting:** All timestamps from Firestore are converted using the trait defined in `firestore.php`.

- **Styling:** Bootstrap 5 is used extensively. Google Fonts, custom CSS, and Font Awesome icons contribute to a visually engaging interface.

- **Analytics & Monitoring:** Google Analytics via Firebase and Sentry error tracking are enabled site-wide.

Setup Manual

1. **Unzip** the project files into a web server directory such as `C:\wamp64\www\Web Assignment`.

2. **Install Software**:

    - PHP 8.4+ with Composer (required for Firebase Admin SDK and Sentry).

    - Node.js v23+ (required for Stripe and Gemini chatbot).

    - A local server stack such as WAMP or XAMPP.

3. **Run Composer:**
   composer install
   Then:
   composer require kreait/firebase-php sentry/sdk
4. **Firebase Setup**:

    - Replace the Firebase config inside `firebase_cdn.js` with your own.

    - Place the service account key in the root directory and update `firebase_admin.php`.

5. **Node.js Setup**:

    - Install dependencies and run your chatbot and Stripe handlers. This is included in "Gemini Chatbot via Nodejs Report.docx" and "Stripe Payment Integration via Nodejs Report.docx".

6. **reCAPTCHA Setup**:

    - Replace `site key` and `secret key` in `.env`(needs node module: dotenv).
    - Include `cacert.pem` for cURL HTTPS requests.
7. **Database & Firestore**:
    - Collections: `tb_users`, `tb_products`, `tb_purchases`
    - Access through Firebase Console or REST API for manual testing, or directly operate through the internal API in update.php.

Project Timeline

- Week 1: Reset project environment and backend structure.
- Week 2: Firebase OAuth login integration.
- Week 2: IP check, reCAPTCHA.
- Week 3: Implement product logic and Firestore interactions.
- Week 3: OAuth user management for admins and deletion logic.
- Week 4: Documentation, testing and bug fixing.

User Acceptance

Feature	Expected Outcome	Status
OAuth Login	User can log in using Google, GitHub and Email with Password OAuth.	✅ Passed
Add to Cart	Cart updates and persists in localStorage.	✅ Passed
Stripe Payment	Stripe session creates and redirects securely.	✅ Passed
Account Deletion	Firebase and Firestore records are deleted.	✅ Passed
Dashboard Admin Panel	Admin can delete or edit users/products.	✅ Passed
reCAPTCHA Enforcement	Only shown when IP is suspicious	✅ Passed
Firestore Data Handling	Properly formatted timestamps and validation.	✅ Passed
Chatbot Response	Gemini provides basic context-aware answers.	Skeptical
IP Threat Detection	ipdata API blocks access .	✅ Passed

Appendix

- Firebase Console:
  https://console.firebase.google.com/project/web-assignment-4237d/overview

- Gemini:
  "Integrating Google Gemini to Node.js Application" by Reetesh Kumar
  https://medium.com/@rajreetesh7/integrating-google-gemini-to-node-js-application-e45328613130

- reCAPTCHA:
  https://developers.google.com/recaptcha/docs/display

- Stripe:
  https://github.com/stripe/stripe-node

- Sentry:
  https://docs.sentry.io/platforms/php/

- Templates:
  - Firebase UI Auth:
    https://firebase.google.com/docs/auth/web/firebaseui
    Reference:
    https://gist.github.com/shadowlion/e85106cbcd3cd4542a66e3c8b42702b3
  - Bootstrap: https://getbootstrap.com
- And assistance from ChatGPT and Gemini generative AI.

AI Declaration

[ ] I declare that GenAI tools have not been used to prepare and produce the submitted assessment.NO AI. The assessment is completed entirely without AI assistance.

[X] I declare that GenAI tools have been used to prepare and produce the submitted assessment.

The involvement of GenAI is [2]

1. AI Assisted coding. AI is used to complete certain coding.

2. Full AI. AI is used to complete most of the areas of the project.