Computer Science Department
Web Application and Technologies (COMP 334)
First Semester 2024/2025

Project: PHP Script, CSS, and HTML

The Phase one due on 10/01/2026; Final Project submssion on 26/01/2026 see the submission instructions below.

Freelance Services Marketplace: Requirements Specification

Project Introduction

The Freelance Services Marketplace is a web-based platform that connects clients seeking professional services with freelancers offering their expertise across multiple categories including web development, graphic design, writing, digital marketing, and more. The primary objective of this project is to build a complete, database-driven web application using fundamental web technologies—HTML5, CSS, PHP, and MySQL—without using JavaScript frameworks or CSS libraries. You will implement essential e-commerce functionality including user authentication with role-based access control (clients and freelancers), service browsing with advanced filtering and search capabilities, a shopping cart system, a multi-step checkout process, and comprehensive order management. Through this project, you will apply core concepts of server-side web development, session management, form validation, file uploads, database design and querying, and user interface design using CSS Grid and Flexbox, culminating in a fully functional marketplace application that demonstrates your ability to integrate multiple web technologies into a cohesive, professional system.

Main Use Cases

1. **Use Case 1: User Registration**
2. **Use Case 2: User Login**
3. **Use Case 3: Profile Management**
4. **Use Case 4: Create Service Listing**
5. **Use Case 5: Edit Service Listing**
6. **Use Case 6: Browse and Search Services**
7. **Use Case 7: View Service Details**
8. **Use Case 8: Add Service to Cart**
9. **Use Case 9: View and Manage Shopping Cart**
10. **Use Case 10: Checkout and Place Order**
11. **Use Case 11: Order Management**

Project Documents

The project documentation is organized into three files:

1. **Requirements Specification**: Details use cases, functional descriptions, and main business rules for the Freelance Services Marketplace. *(this document)*
2. **Appendix A – CSS Styling Requirements**: Lists all CSS classes used in the application, serving as a comprehensive styling reference.
3. **Appendix B – Complete Database Schema**: Outlines the full database structure, including tables, fields, data types, constraints, and relationships.

Important Notes

- This is an individual assignment.
- Only use the following technologies: HTML5, CSS, PHP, and MySQL. Frameworks and JavaScript are strictly prohibited.
- Do not use any external web development tools; any attempt will result in a zero mark.
- All database connection details must be defined in a separate file named `db.php.inc`, which should create a **PDO** connection object using the required variables.
- For all SQL queries, use **prepared statements with named binding parameters**.
- **Use only external CSS files**—embedded or inline CSS is not allowed. Submit final CSS files containing only the classes actually used in your HTML pages; remove any unused or temporary classes. **Only use the CSS rules provided in the lecture**; frameworks such as Bootstrap are not permitted.
- The application is multi-page: each navigation action should load a new PHP page.

Submission

Phase 1 Submission

Deadline: Phase 1 deliverables must be submitted by **10/01/2026 at 22:00 to both ITC and CShost**. Upload all files (PHP, HTML, CSS, images, and SQL) to CShost, and also submit a compressed archive containing these files to ITC.

Phase 1 Requirements:

1. **Database Schema:**
    - Design and create the database schema, including all tables, their fields, and the relationships (primary and foreign keys) as described in Appendix B.
    - Import this schema to the CShost server.
    - Populate the database with some data for testing
2. **Navigation System:**

    - Implement the navigation system as described in Appendix A: Page Structure.
    - Submit all related HTML, PHP scripts, and CSS style files.
3. **Service Table Display:**
    - Load all services into a table, styled according to the specifications in Appendix A: My Services Table. Display the table in the main contect section of the page.

Final Submission Instructions

Deadline:
All submissions must be completed by **26/01/2026 at 22:00**.

- The CShost server will close at this time, so ensure your project is submitted before the deadline.
- In addition to submitting to CShost, you must also submit your project to ITC by the same deadline.
- Submission is only allowed through the domain created for you and through ITC, as specified in the instructions on the last page.

Responsibility for Testing:

- Since you are submitting to your domain at CShost, it is your responsibility to ensure that your scripts, database schema, and all related files work properly on the CShost server.
- Upload your scripts and other development files before the submission date so you have time to test and resolve any issues that may arise.

Submission to ITC:

- Compress your project files into a single archive.
- When extracted, the archive should create a folder named **stdID** (replace "ID" with your student ID).
- After extraction, verify that your project works locally by visiting:
  `http://localhost/webprojects/stdID`
  Ensure all links and functionality are working correctly.

Your submission folder must include:

- All PHP scripts, HTML, CSS files, images, and other related files.
- The database schema exported as an SQL file named:
  `dbschema_yourNumber.sql`
- An index.html file.

Index Page Requirements:

- Display your name and student ID.
- Provide a link to your main page.

Database Setup:

- The database on the server should be named:
  {prefix}_stdID.sql
  ("stdID" is your student ID; "prefix" will be provided by the server—refer to the illustration video).
- Define a database user named `proj{stID}` and a password `pass{stID}` for connecting from your PHP scripts.

Testing Requirements:

- Create two user types for testing: **Client** and **Freelance**.
- Display their usernames and passwords.
- Populate the database with sample data for testing purposes.

Use Cases

Refer to Appendix A for all styling requirements, including page layout, data display, form design, .., etc. Always follow the specifications in Appendix A when implementing any visual or structural aspect of your project.

Use Case 1: User Registration

Description: New users create account by providing personal information and selecting role (Client or Freelancer).

Preconditions: None (public access)

Flow of Events:

1. User navigates to registration page → Fills form → Selects account type → Submits
2. System validates inputs → Generates 10-digit User ID → Creates account → Redirects to login with success message

Input Requirements

Field	Required	Format/Constraints	Validation
Full Name	Yes	2-50 characters, letters and spaces only	No special characters/numbers
Email	Yes	Valid email format (with @ and domain)	Must be unique in system
Password	Yes	Min 8 chars: 1 upper, 1 lower, 1 number, 1 special char	Check complexity
Confirm Password	Yes	Must match Password	Compare fields
Phone Number	Yes	Exactly 10 digits, numbers only	Length and numeric
Account Type	Yes	Radio: "Client" or "Freelancer"	One selected
City	Yes	Dropdown from predefined list	Valid selection
Bio/About	Conditional	Optional for Client, Required for Freelancer, max 500 chars	Required if Freelancer
Age Verification	Yes	Checkbox: "I am 18+ years old"	Must be checked

System-Generated Data

- **User ID:** 10-digit unique numeric identifier
- **Registration Date:** Current timestamp
- **Account Status:** "Active"
- **Initial Rating:** 0.0 (freelancers only)

Validation & Error Handling

- All required fields must be completed
- Email must not exist in database
- Password must meet complexity requirements
- Display specific errors for each field with red border
- Preserve valid data (except passwords) on error
- Show all errors simultaneously

Success Handling

Message: "Account created successfully! Please login." → Redirect to login after 2 seconds

Page Specifications

- Layout: Standard page structure, form centered, max-width 600px
- Navigation: "Register" active
- Styling: See CSS Styling Requirements document, Sections: Global Page Layout, Form Standards
- Sections: "Create Your Account", "Personal Information", "Account Security", "Account Type"

Use Case 2: User Login

Description: Registered users authenticate to access role-based features.

Precondition: User has registered account with Active status.

Flow of Events:

1. User enters email and password → Submits
2. System validates credentials and account status
3. System creates session → Redirects to role-specific page (Clients: Browse Services, Freelancers: Dashboard)
4. System updates header with user profile card, role-specific navigation, cart icon (clients only)

Input Requirements

Field	Required	Validation	Notes
Email	Yes	Valid format, exists in database	-
Password	Yes	Matches hashed password	No format validation on login
Remember Me	No	-	Placeholder only (Phase 1)

Session State Requirements

After successful login, system must maintain:

- User authentication status across all pages
- User identity for database queries and authorization
- User role for access control (Client/Freelancer)
- User display information (name, profile photo for header)
- Shopping cart contents during browsing

Technical Constraints:

- Use PHP session management
- Session timeout: 24 hours of inactivity
- Sensitive data (passwords) must never be stored in session

Business Rules

- Maximum 5 failed attempts in 15 minutes → Account locked for 30 minutes
- Display remaining attempts after 3rd failure
- Account must have "Active" status

Error Handling

- **Invalid credentials:** "Invalid email or password" (generic for security)
- **Account locked:** "Account temporarily locked. Please try again in X minutes."
- **Include link:** "Don't have an account? Sign up"

Page Specifications

- Layout: Standard page structure, form centered, max-width 400px, heading: "Login to Your Account"
- Navigation: "Login" active
- Styling: See CSS Styling Requirements document, Sections: Form Standards, Button Classes
- Additional Links: "Forgot password?" (placeholder), "Sign up" link

Use Case 3: Profile Management

Description: Users view and update personal information, contact details, profile picture. Freelancers can update skills and rates.

1. Overview

- **Actor:** Logged-in user (Client or Freelancer)
- **Goal:** View and update profile
- **Precondition:** User is logged in
- **Postcondition:** Profile updated in database and session

2. Page Structure

- **URL:** /profile.php
- **Layout:** Two columns (Left: 30%, Right: 70%)
    - **Left Column:** Profile card, statistics (freelancers only)
    - **Right Column:** Edit profile form (account, personal, professional info)

3. User Flow

1. User navigates to the profile page.
2. Profile card and statistics (if freelancer) are displayed on the left.
3. Edit form is populated with current user data on the right.
4. User modifies desired fields.
5. If changing password, user enters current password.
6. If uploading a new photo, user selects a file.
7. User clicks "Save Changes."
8. Server validates all inputs.
9. If validation fails: specific error messages are shown, and data is preserved for correction.
10. If validation succeeds: user record, photo, and/or password are updated; session is refreshed; a success message is displayed; the page is refreshed to show updates.

4. Profile Card (Left Column)

- **Profile Photo:** 150x150px, circular, upload/change option, default placeholder if not set
- **User Information:** Full name, email address, role badge (Client: blue, Freelancer: green), member since date
- **Role Badge Styling:** Colored background, white text, rounded corners, positioned below name

5. Statistics Card (Freelancers Only)

- **Location:** Below profile card
- **Display:** Four statistics in a 2x2 grid:
    - Total Services (all owned)
    - Active Services (green accent)
    - Featured Services (gold/yellow if at limit, format X/3)
    - Total Orders (completed only)
- **Visuals:** Large numbers, small labels, white/light background, border, optional icons

6. Edit Profile Form (Right Column)

6.1 Account Information

- **Fields:**
  - Email (required, unique, valid format)
  - Current Password (required if changing password)
  - New Password (min 8 chars, complexity rules)
  - Confirm New Password (must match new password)
- **Rules:** Current password required for change; validation for all fields

6.2 Personal Information

- **Fields:**
  - First Name (required, 2–50 chars, alphabetic)
  - Last Name (required, 2–50 chars, alphabetic)
  - Phone Number (required, 10 digits)
  - Country (required, select from list)
  - City (required, 2–50 chars)
  - Profile Photo (optional, JPG/JPEG/PNG, max 2MB, min 300x300px)
- **Photo Storage:** /uploads/profiles/[user_id]/profile_photo.jpg; path stored in database

6.3 Professional Information (Freelancers Only)

- **Fields:**
  - Professional Title (required, 10–100 chars)
  - Bio/Description (required, 50–500 chars)
  - Skills (optional, comma-separated, max 200 chars)
  - Years of Experience (optional, 0–50, integer)
- **Note:** Clients do not see this section

7. Validation & Error Handling

- **All fields validated as per registration requirements**
- **Email:** Must be unique; error if already in use
- **Password:** Complexity, match, current password check; error if incorrect
- **Photo:** File type/size, upload success; errors for invalid type, size, or upload failure
- **Errors:** Specific messages, preserve entered data for correction

8. Database & Session Operations

- **Update:** User info, password, photo as needed
- **Security:** Use prepared statements for all queries
- **Session:** Refresh session variables after update

9. Layout & Styling

- See CSS Styling Requirements document, Section: Page-Specific Styling - Profile Page Layout
- Navigation: "My Profile" active

Use Case 4: Create Service Listing

Page: `/create-service.php`

Description: Freelancers create service offerings through 3-step process: basic info, upload images, review/confirm.

Precondition: User is logged in as Freelancer, Only freelancers can create services.

Flow of Events:

Step 1 - Basic Information:

1. System displays form → Freelancer enters service details → Submits
2. System validates → Stores in session → Redirects to Step 2

Step 2 - Upload Images:

1. System displays upload form → Freelancer uploads 1-3 images, selects main image → Submits
2. System validates files → Stores temporarily, paths in session → Redirects to Step 3

Step 3 - Review & Confirm:

1. System displays all data from session → Freelancer reviews → Confirms
2. System generates 10-digit Service ID → Saves to database → Moves images to permanent storage → Clears session → Redirects to "My Services" with success message

Step 1: Basic Information Input Requirements

Field	Required	Format/Constraints	Validation
Service Title	Yes	10-100 characters, descriptive	Length check
Category	Yes	Dropdown: Web Development, Graphic Design, Writing & Translation, Digital Marketing, Video & Animation, Music & Audio, Business Consulting, Tutoring & Education	"Category is required", Must be valid category from predefined list
Subcategory	Yes	Dropdown with `<optgroup>` grouping subcategories by category (see below)	Must select option (not optgroup label). "Subcategory is required", Must be valid subcategory for selected category
Description	Yes	100-2000 characters, detailed explanation	Length check

Delivery Time	Yes	1-90 days, integer	"Delivery time is required", "Delivery time must be 1-90 days"
Revisions Included	Yes	0-999, integer (only)	"Number of revisions is required", "Revisions must be 0-999"
Price	Yes	Numeric, min $5, max $10,000	"Price is required", "Price must be between $5 and $10,000"

Subcategories (Examples):

Design & Creative:	**Writing & Translation**	**Digital Marketing**
Logo Design Brand Identity Web Design Graphic Design Illustration UI/UX Design	Article Writing Copywriting Proofreading Translation Technical Writing	SEO Social Media Marketing Email Marketing Content Marketing PPC Advertising
Programming & Tech	**Business & Consulting**	**Video & Animation:**
Web Development Mobile App Development WordPress E-commerce Development Bug Fixes	Business Planning Financial Consulting Legal Consulting HR Consulting	Video Editing Animation Whiteboard Animation Video Production

Subcategory Structure (HTML `<optgroup>`):

```
<select name="subcategory">
  <optgroup label="Web Development">
    <option>Frontend Development</option>
    <option>Backend Development</option>
    <option>Full Stack Development</option>
    <option>WordPress Development</option>
  </optgroup>
  <optgroup label="Graphic Design">
    <option>Logo Design</option>
    <option>Brand Identity</option>
    <option>Print Design</option>
    <option>Illustration</option>
  </optgroup>
  <!-- Similar structure for remaining categories -->
</select>
```

Step 2: Image Upload Requirements

Requirement	Specification
Minimum Images	1 required
Maximum Images	3 per service
File Formats	JPG, JPEG, PNG only
Max File Size	5 MB per file
Min Dimensions	800x600 pixels
Main Image	User selects via radio buttons (first uploaded is default)

Service Images
Upload Interface:

- Multiple file input or three separate file inputs
- Clear labels: "Service Image 1 (required)", "Service Image 2 (optional)", "Service Image 3 (optional)"

System-Generated Data

- **Service ID:** 10-digit unique numeric identifier, verify uniqueness before insert
- **Freelancer ID:** Retrieved from session
- **Creation Date:** Current timestamp
- **Status:** "Active"
- **Featured Status:** "No"
- **Image Paths:** `/uploads/services/[service_id]/image_01.jpg`, etc.

Validation Rules

- All Step 1 fields required before proceeding to Step 2
- Title unique per freelancer (can duplicate other freelancers)
- Minimum 1 image required in Step 2
- One image must be selected as main
- Service title must be unique for this freelancer
- Maximum 50 active services per freelancer

Error Handling

- Step-specific errors displayed
- Preserve entered data
- If image upload fails, allow retry without re-entering text data
- If user navigates away: "You have unsaved changes. Are you sure?"
- If session expires during creation, redirect to login
- Auto-delete temporary images after 2 hours if not saved

Session Management During Creation:

- Store in session
- Clear all service creation data after successful save or user cancellation

Success Handling

Message: "Service created successfully!" → Show Service ID → Provide links: "View Service", "Create Another Service"

Page Styling:

- See CSS Styling Requirements document, Sections:
    - Progress Breadcrumb Navigation (for step indicators)
    - Form Standards (for all form fields)
    - Page-Specific Styling - Create/Edit Service Page (for image uploads, thumbnails, subcategories)
- Navigation: "Create New Service" active
- Breadcrumb: Home > My Services > Create New Service

Use Case 5: Edit Service Listing

Description: Enable freelancers to view, edit, and manage their service listings on the platform. All changes must be reflected immediately in the system.

Precondition: User is logged in as Freelancer, owns the service

Postconditions: Service updated in database, changes reflected immediately

Flow of Events:

Functional Requirements

4.1 My Services Page

- Display all services owned by the logged-in freelancer in a table format.
- Show a statistics card summarizing key service metrics, see Use Case 3 for details).
- Table columns: Image, Service Title, Category, Price, Status, Featured, Created Date, Actions: Edit, Deactivate, or Activate buttons as appropriate, based on service status.
- **Table Styling**:
    - See CSS Styling Requirements document, Sections: Table Styling Requirements, Page-Specific Styling - My Services Table

4.2 Service Status Management

- Each service can be Active or Inactive.
- Freelancer can change the service status via radio buttons control, with "Active" and "Inactive" options, the “current status” pre-selected, and labeled as "Service Status".
- Active: Visible in Browse Services, purchasable, searchable, can be featured.

- Inactive: Hidden from Browse Services, not purchasable, not searchable, editable only by freelancer.
- Deactivate: Sets status to Inactive, removes Featured, refreshes page with success message.
- Activate: Sets status to Active, refreshes page with success message.

4.3 Featured Status Management

- Freelancer can mark a service as Featured using a checkbox or toggle.
- Only Active services can be featured.
- Maximum of 3 featured services per freelancer.
- If exceeding limit, block action and show error.
- Featured Status Display:
  - Gold star icon displayed using CSS ::before pseudo-element
  - See CSS Styling Requirements document, Section: Page-Specific Styling - My Services Table - Featured Star Icon

4.4 Edit Service Page & Image Management

- Only the owner can access the edit page.
- Form pre-populated with current service data, form structure: same as Create Service form (Use Case 4).
- Editable fields: Basic Info, Pricing & Delivery, Images, Status, Featured.
- When replacing an image, delete the old image from the server and update database.
- At least one image must remain.
- New images must meet validation rules (max 5MB, JPG/JPEG/PNG).

5. Business Rules

- - Maximum of 3 featured services per freelancer.
- If service is Inactive, Featured status is removed.
- Only Active services can be featured.

6. Validation

- Validate all fields as per Create Service requirements.
- Block featuring more than 3 services.
- Remove Featured status when setting Inactive.
- Image uploads: max 5MB, JPG/JPEG/PNG, at least one image required.

7. Database Operations

- Update service record with new data.
- If status is Inactive, set Featured to No.
- Verify featured count before setting Featured to Yes.

- Delete old image file when replaced and update database path.

8. User Flow

8.1 My Services Page

1. Freelancer navigates to My Services.
2. Page displays statistics card at top and services table below.
3. Freelancer can:
    - View all their services.
    - See service statistics.
    - Click "Edit" to modify a service.
    - Click "Deactivate/Activate" to change status quickly. Success Handling: display success message: Service updated successfully" → Redirect to "My Services"
    - Click service title to view public service page.

8.2 Edit Service

Page: `/edit-service.php?id=[service_id]`

1. Freelancer clicks "Edit" button for a service.
2. Edit Service form loads with current data.
3. Freelancer modifies desired fields:
    - Update text information.
    - Change pricing/delivery.
    - Replace images.
    - Change status (Active/Inactive).
    - Toggle featured status.
4. Freelancer clicks "Update Service."
5. Server validates all inputs.
    - If validation fails: Display error messages, preserve entered changes, user corrects and resubmits.
    - If validation succeeds: Update service record in database, update images if replaced, apply status changes, display success message, redirect to My Services page

Use Case 6: Browse and Search Services

6.1 Overview

Actor: All Users (Guests, Clients, Freelancers)
Goal: Browse available services, search and filter results
Preconditions: None (accessible to all)
Postconditions: User views service listings, may navigate to service detail

6.2 Browse Services Page

Page: `/browse-services.php` or `/index.php`

Access: Available to all users (no login required)

6.2.1 Page Layout

Components:
Search and Filter Bar (top)
Services Grid (main content)
Pagination (bottom, if needed)
6.2.2 Search and Filter Bar
6.2.2 Search and Filter Bar (Simplified)

- **Search:**
  Enter keywords to find services by title or description.
  Search is case-insensitive and supports partial matches.
- **Category Filter:**
  Select a category from a dropdown to narrow results.
  Option to show all categories.
- **Sort**:
  Sort services by newest, oldest, price low-to-high, or high-to-low.
- **Combined Filters:**
  All filters work together, (search + category + sort).
  Results update automatically and are reflected in the page URL.

6.2.3 Services Grid

Display: Grid layout of service cards

- **Grid and Card Styling:**
  - See CSS Styling Requirements document, Section: Page-Specific Styling - Service Browse Page Layout

Only Active Services Displayed:

- Only services with status = 'Active' shown
- Inactive services not visible to any user (except owner in My Services)

6.2.3 Service Card Component

Each Service Card Displays:

- Service image (clickable, links to service detail page)
- Title (clickable, links to service detail page)
- Freelancer name and small profile photo (30x30px, circular)
- Category
- Price ("Starting at $XX")
- Featured badge (only if service is featured)

Grid and Card Styling:

- See CSS Styling Requirements document, Section: Page-Specific Styling - Service Browse Page Layout

6.2.4 Featured Services Section

Location: Above main services grid
Display:

- Heading: "Featured Services"
- Separate row of featured services
- Featured services also appear in main grid
- This section highlights them separately

6.3 Search Functionality

- **Keyword Search:**
  Enter keywords to find services by title or description.
  Search is case-insensitive and supports partial matches.
  Empty search shows all services.
  Display search term above results: e.g "Search results for 'logo design'" and result count.
  If no results, show a clear message and link to clear filters: "Show All Services".
- **Category Filter:**

Select a category to narrow results.
Display category name above results and resultcount.
Option to remove filter, (link: "Show All Categories")

Pagination

If Many Services:

- Show 12, 15, or 20 services per page.
- Pagination controls at the bottom: "Previous," "Next," and page numbers.
- URL parameters reflect current page and filters.

6.6 User Flow

1. User goes to the Browse Services page.
2. All active services are shown by default (newest first).
3. User can search, filter by category, or sort.
4. Results are updated based on user selection
5. User clicks a service card or title to view details.

Use Case 7: View Service Details

7.1 Overview

Actor: All Users (Guests, Clients, Freelancers)
Goal: View complete details of a specific service
Preconditions: Service exists and is Active
Postconditions: User views service information, may add to cart (clients) or contact freelancer

7.2 Service Detail Page

Page: `/service-detail.php?id=[service_id]`

Access: Available to all users

URL Parameter: Service ID passed via GET

Layout & Styling:

- See CSS Styling Requirements document, Section: Page-Specific Styling - Service Detail Page Layout

7.2.1 Page Layout

Layout: Two-column layout

Two-column layout:

- **Left (65% width):** Service images (Gallery) , title, category, description, freelancer info.
- **Right (35% width):** Booking card (sticky), price, delivery time, revisions, Add to Cart/Order Now (clients), Contact Freelancer.

7.2.2 Service Images Gallery

- Main image displayed large and high quality; thumbnails displayed horizontally below main image.
- Clicking a thumbnail switches the main image. Images: image_1, image_2, image_3 (if available)
- Use simple HTML/CSS for image switching (anchor links and :target selector); JavaScript not required.

7.2.3 Service Information (Left Column)

- **Service Title:** Large heading (H1), pulled from database.
- **Category Badge:** Shows category and subcategory, styled as breadcrumb Category > Subcategory.

- **Freelancer Info:** Profile photo, name, member since, link to profile link. Freelancer Info displayed as a card with light background and padding, profile photo on left, info on right.
- **Service Description:** Up to 1000 characters, multi-paragraph, readable font, heading: "About This Service".

7.2.4 Booking Card (Right Column - Sticky)

Sticky card in right sidebar, stays visible while scrolling.

Displays:

- Price (large, “Starting at” heading)
- Delivery time
- Revisions included

Buttons:

- Add to Cart (clients only), see use case 8
- Order Now (clients only), Add to cart and redirect to checkout
- Login to Order (guests, and fee viewing other services),
- Edit Service (Freelancers viewing own service)

  Styling:
  - **Card:** White background, light border, box shadow, rounded corners, padding.
  - **Buttons:** Full width of card

7.3 Recently Viewed Services

Purpose: Track user's recently viewed services using cookies

Cookie Implementation:

Cookie Format: Comma-separated string of service IDs

- Example: "1234567890,9876543210,5555555555,1111111111"

Maximum: 4 service IDs (most recent)

Cookie Lifespan: 30 days

Cookie Operations (Server-Side):

When User Views Service (This Page):

1. Read the cookie and split it into an array. Use PHP explode() function.
2. Remove the current service ID if it exists.
3. Add the current service ID to the end.
4. If more than 4 items, remove the oldest.
5. Join the array and update the cookie with. Use PHP implode() function for join.

7.4 Access Control and Visibility

- Only display services with status = 'Active'.
- If service is inactive or doesn't exist: show error ("Service not found" or "Service no longer available").
- Exception: If the logged-in user is the service owner (freelancer), allow viewing even if inactive, and show a notice: "This service is currently inactive and not visible to clients."

7.5 User Flow

- User clicks a service card from Browse Services.
- Service detail page loads with the selected service.
- Page displays service images, information, booking card, and reviews.
- Recently viewed services cookie is updated.
- View counter is incremented.
- User can:
    - View all service details
    - Read reviews
    - Add to cart or order (if client)
    - Login to order (if guest)
    - Explore other services

7.6 Button Behavior & Actions

Guests (Not Logged In):

- Hide "Add to Cart" and "Order Now" buttons.
- Show "Login to Order" button (redirects to login page).

Clients (Logged In):

- Show "Add to Cart" and "Order Now" buttons.
- "Add to Cart": Adds service to cart, stays on page, shows success message, updates cart badge (icon in header).
- "Order Now": Adds service to cart and redirects to checkout.

Freelancers Viewing Own Service:

- Hide cart buttons.
- Show "Edit Service"button (redirects to Edit Service page).

Freelancers Viewing Other Services:

- Show "Add to Cart" and "Order Now" buttons (freelancers can purchase services too).

Use Case 8: Add Service to Cart

8.1 Overview

Actor: Client
Goal: Add a service to shopping cart for later checkout
Preconditions: User logged in as Client, service is Active
Postconditions: Service added to cart session, cart badge updated

8.2 Object-Oriented Programming Requirement

Service Class Required:

You must create a `Service` class to represent services in the cart.

8.2.1 Service Class UML Diagram

Class Name: Service

Implemntaion Notes:

- `getFormattedPrice()` : string (returns "$XX" format)
- `getFormattedDelivery()` : string (returns "X days")

- `calculateServiceFee()` : float (returns 5% of price)
- `getTotalWithFee()` : float (returns price + service fee)

8.2.2 Cart Structure

Session Variable: `$_SESSION['cart']`

Structure: Array of Service objects

Each Service object contains:

- All service data at time of adding to cart
- Price locked (see Section 8.3)
- Added timestamp

8.3 Price Handling

- **Price Lock:** When a service is added to the cart, its price is captured from the database and stored in the Service object.
- **No Price Changes:** The price in the cart remains the same for the entire session, even if the freelancer updates the price later.
- **No Alerts:** The system does not check for or notify about price changes after adding to the cart. The client always pays the price shown at the time of adding the service.

8.4 Add to Cart Process

Trigger: Client clicks "Add to Cart" on Service Detail page (Use Case 7).

Validation:

- Must be logged in as (Client or freelancer who is not the service owner), if not logged in display an error message and: redirect to login page
- Service must be Active, if service is inactive dispplay an error message.
- Check if the service already exists in the cart, then an error message should display: "Service already in cart" (no duplicates).

Process:

- Validate user and service status.
- Retrieve service data from database.
- Create Service object with current data and timestamp.
- Add Service object to cart session array.
- Update cart badge count in header.
- Show success message ("Service added to cart successfully!"), then return (redirect) the user to current page (Service Detail page).

Notes:

- Only one instance of each service per order.
- Cart is session-based and temporary.

8.5 Cart Badge Update

- **Badge Location:** Shopping cart icon in header (clients only)
- **Display:** Circular red badge, white text shows count of services in cart, **See CSS Styling Requirements document for cart icon styling details**
- **Updates:** When services are added (this use case) or removed (use case 8); shows 0 or hides badge if cart is empty

Use Case 9: View and Manage Shopping Cart

9.1 Overview

Actor: Client
Goal: View all services in cart, update cart, proceed to checkout
Preconditions: User logged in as Client
Postconditions: User views cart contents, may remove services, may proceed to checkout

9.2 Shopping Cart Page

Page: `/cart.php`

Access: Clients only

If Not Client:

- Redirect to login page with error: "Please login as client to view cart"

9.2.1 Page Layout

Layout & Styling:
- See CSS Styling Requirements document, Section: Page-Specific Styling - Shopping Cart Page

Two States:

1. Cart with items
2. Empty cart

9.2.2 Cart with Items

Main Area:
Layout: Table of services.
- Cart items are dispplayed as table, the table should be stylled with sticky header for column titles.
- Each service displayed as row, that shows:
    - Thumbnail (100x75px, links to service detail)

- Title (prominent, links to detail)
- Freelancer (links to freelancer's services)
- Category (small text)
- Delivery time ("X days")
- Revisions ("X" or "Unlimited")
- Price ("$XX")
- Remove button ("Remove" or "×" icon, removes item)

Summary Sidebar (Right):

Order summary: subtotal, service fee (5%), total.

Checkout button (prominent, full width, primary color), action: navigate to checkout page (Use Case 10).

Example

```
ORDER SUMMARY
─────────────────────────────
Services Subtotal: $600.00
Service Fee (5%):   $30.00
─────────────────────────────
Total:             $630.00

[Proceed to Checkout]
```

9.2.3 Remove from Cart

Trigger: User clicks "Remove" on a cart item.

Process:

1. Identify service to remove.
2. Remove service from cart.
3. Show message: "Service removed from cart."
4. Update cart badge count.
5. Redirect to cart page.

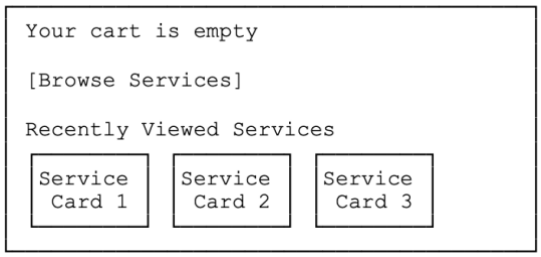
9.2.3 Empty Cart State

Display: see example below

- Message: "Your cart is empty"
- Empty cart or shopping bag icon
- "Browse Services" button (links to Browse Services page)

Recently Viewed Services:

- If available (from cookie), show up to 4 recently viewed active services as cards
- Each card links to service detail
- If none, skip this section



9.3 Cart Validation Before Checkout

When User Clicks "Proceed to Checkout":

Checkout Validation:

- Ensure cart is not empty before checkout; show error and if emty redirect to browse services.
- Verify all services in cart are still active; remove inactive ones and display warning. Service '[title]' is no longer available and has been removed
- Proceed to checkout only if all services are valid.

9.4 Cart Session Management

Session Management:

- Cart persists while user is logged in and across page navigations.
- Cart is cleared when user logs out, session expires, order is placed (use case 10), or all items are manually removed.
- Cart is not cleared when user closes browser (if session persists) or navigates to other pages.

9.5 User Flow

1. Client clicks cart icon in header.
2. Shopping cart page loads.
    - If cart is empty: show empty cart message, "Browse Services" button, and recently viewed services (if available).
    - If cart has items: display services, order summary, and "Proceed to Checkout" button.
3. User can remove services, view details, proceed to checkout, or continue shopping.
4. When a service is removed, the cart updates and badge count refreshes.
5. On "Proceed to Checkout," validation checks are performed and user is redirected to checkout page (Use Case 10).

Use Case 10: Checkout and Place Order

10.1 Overview

Actor: Client
Goal: Complete purchase of services in cart
Preconditions: User logged in as Client, cart contains at least one service
Postconditions: Orders created in database, cart cleared, user redirected to success page

10.2 Important Concepts

10.2.1 Order Structure

One Service = One Order:

The checkout process creates **one order per service** in the cart.

Example:

- Cart contains 5 services
- Checkout creates 5 separate order records
- Each order has unique order_id
- Each order links to one service and one freelancer

10.2.2 Service Requirements

- Only service-specific requirements are collected during checkout.
- Each service in the cart has its own requirements and files.
- Freelancers see only the requirements for their assigned service.
- Each service's requirements are independent; no overlap between services.

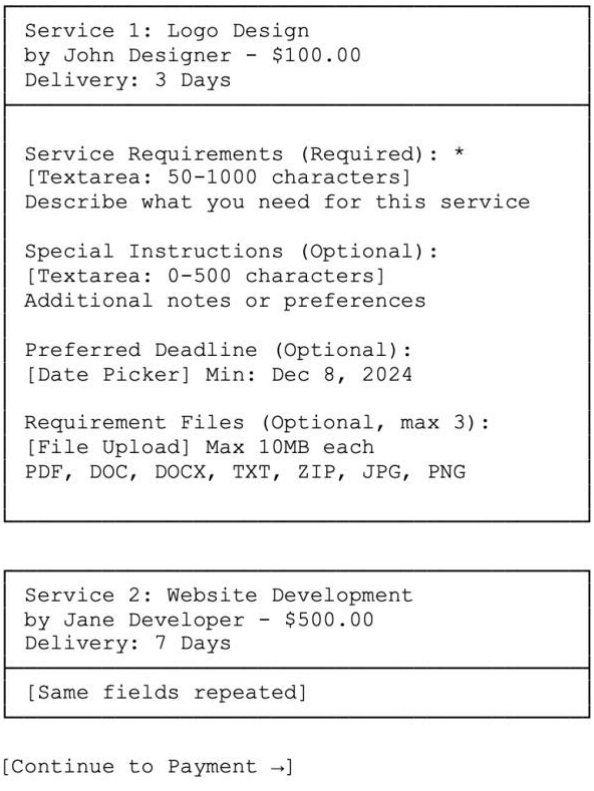
10.3 Checkout Page Structure

- **Page:** /checkout.php
- **Access:** Clients only; cart must have items
- **Process:** 3 steps, all on one page (progressive form)
    1. Service Requirements
    2. Payment Information
    3. Review and Confirmation

- Progress Breadcrumb Navigation (Styling & Layout)
    - See CSS Styling Requirements document, Section: Checkout Progress Breadcrumb Navigation
- Forms:
    - See CSS Styling Requirements document, Section: Form Standards

10.4 Step 1: Service Requirements

- **Purpose:** Collect requirements for each service in the cart.
- **For Each Service:**
  - Show service info: title, freelancer, price, delivery time.
  - **Required:** Service Requirements (textarea, 50–1000 chars).
  - **Optional:** Special Instructions (textarea, up to 500 chars), Preferred Deadline (date picker, must be at least service delivery time from today), Requirement Files (up to 3 files, 10MB each, PDF/DOC/DOCX/TXT/ZIP/JPG/PNG).
- **Validation:**
  - Service Requirements: Required, 50–1000 chars.
  - Special Instructions: Optional, up to 500 chars.
  - Deadline: Optional, must be a valid future date (≥ delivery time).
  - Files: Optional, max 3 per service, 10MB each, allowed types only.
  - All services must have requirements filled to proceed.
- **Actions:**
  - **Continue to Payment:** Validates all fields, If validation fails: Display errors, preserve data. If validation passes: Store in session, show Step 2.
  - **Edit Cart:** Returns to cart, preserves entered data.
- **Layout:** Each service appears as a card with the above fields, styled for clarity and usability.



10.5 Step 2: Payment Information

- **Purpose:** Collect payment details (simulation only; no real payment processing).
- **Payment Method:** Required. Options: Credit Card (default), PayPal, Bank Transfer.
- **Credit Card Fields (if selected):**
    - Card Number: 16 digits, required.
    - Cardholder Name: 2–100 alphabetic chars, required.
    - Expiration Date: MM/YY, must be future date, required.
    - CVV: 3 digits, required.
- **Billing Address:** Required fields—Address Line 1, City, State/Province, Postal Code, Country. Address Line 2 is optional.
- **Validation:** All fields validated as specified; errors shown and data preserved on failure.
- **Security:** No card details stored; payment is simulated for educational purposes.
- **Actions:** Continue to Review (validates and stores non-sensitive data), Edit Payment (return to this step to make changes). Validate payment information. If validation fails: Display errors, preserve data. If validation passes: Store in session (non-sensitive data only), show Step 3

10.6 Step 3: Review and Confirmation

- **Purpose:** Review all entered information before placing the order.
- **Layout:**
    - **Two-column design:**
        - **Order Summary Sidebar (Right, 35%, Sticky):**
            - Order Count: Text: "You will place X orders" (where X = number of services); clarify that each service creates a separate order.
            - Lists all services as cards, see display card example below, each showing:
                - Service title (bold)
                - Freelancer name (clickable link)
                - Price, service fee, total (prominent)
                - Green checkmark icon
            - Cards stack vertically, full width of sidebar, consistent spacing (Flexbox column layout)
            - Summary totals below cards: subtotal, service fee (5%), grand total (bold, large, primary color), see suammary table example below.
            - Place Order button: full width, large, green, disabled until terms checkbox is checked
        - **Main Content (Left, 65%):**
            - For each service, expandable/collapsible section with:
                - Service header (title, freelancer, price)
                - Service requirements, special instructions, deadline preference, uploaded files

        - Files shown in horizontal containers with icon, file info, size, date, clickable name, see file display styling requirements below.
        - Empty state: "No files uploaded" message, styled for clarity
    - Payment information: selected method, billing address, masked card info
    - Terms agreement checkbox (required to place order), links to Terms of Service and Privacy Policy
    - Action buttons: Place Order (large, green), Edit Payment Information, Edit Service Requirements

Each Service Display as card, example	Summary Totals Cards, example
✓ Service Title by Freelancer Name Price: $100.00 Service Fee: $5.00 Total: $105.00	Services Subtotal: $650.00 Total Service Fee (5%): $32.50 GRAND TOTAL: $682.50 [Place Order]

File Display Styling Requirements:

- See CSS Styling Requirements document, Section: File Display Styling Requirements

10.7 Place Order Process

- **Trigger:** User clicks "Place Order" button.
- **Validation:** Terms agreement checked, all services have requirements, payment information complete, cart not empty.
- **Order Creation Flow:**
    1. Generate Transaction ID ("TXN" + timestamp + random).
    2. For each service in cart: generate unique 10-digit order ID, calculate fees, retrieve requirements, instructions, deadline, uploaded files, calculate expected delivery date, insert order record into database.
    3. **Upload and store requirement files in directory:** `/uploads/orders/[order_id]/requirements/`
        - For each file, store in database: `file_id (auto-increment), order_id, file_path (full path), original_filename, file_size, mime_type, file_type ('requirement'), upload_timestamp (current).`
    4. Store order ID for success page.
    5. Clear cart and checkout session.

6. Redirect to success page, display all created orders.

10.8 Order Success Page)

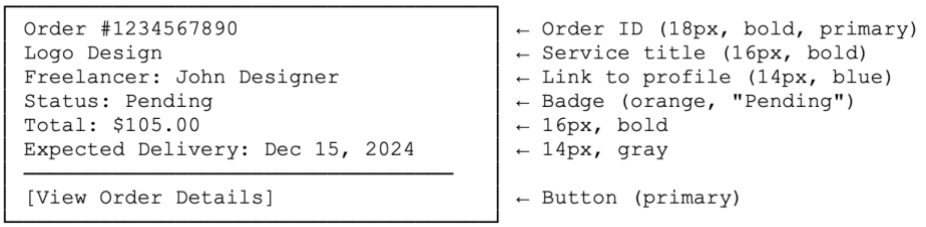
- **Page:** /order-success.php
- **Display:** Confirmation of successful order placement

Layout Requirements:

- See CSS Styling Requirements document, Section: Page-Specific Styling - Order Success Page

- **Order Cards:**
  - Each order shown as a card: `Order ID (bold, primary color), Service title (bold), Freelancer (link to profile), Status badge ("Pending", orange), Total amount, Expected delivery date, "View Order Details" button.` See example below.
  - Card styling: white background, light gray border, rounded corners, padding, margin-bottom, shadow with hover effect
- **Action Buttons:**
  - "View All Orders" (primary, large), "Browse More Services" (secondary, large), spaced and center-aligned
  - **Actions:**
    - View All Orders → Navigate to My Orders page (Use Case 11)
    - Browse More Services → Navigate to Browse Services page

Order card Example:



Use Case 11: Order Management

11.1 Overview

Actor: Clients and Freelancers
Goal: View orders, track progress, upload deliveries, request revisions, complete orders, leave reviews
Preconditions: User is logged in
Postconditions: Orders managed according to user actions

11.2 My Orders Page

- **Page:** /my-orders.php
- **Access:** Clients & Freelancers
- **Purpose:** Show all orders for the logged-in user.
- **Display:**
  - **Clients:** See orders they placed.
  - **Freelancers:** See orders assigned to them.
- **Table Columns:**
  - Order ID (clickable, links to details)
  - Service (title)
  - Freelancer/Client (shows the other party)
  - Price (total, formatted as $XXX.XX)
  - Status (color-coded badge)
  - Order Date (MMM DD, YYYY)
  - Expected Delivery
  - Actions (View Details button)
- **Status Badge Colors:**
  - Styling Requirements
  - Status badges: See CSS Styling Requirements document, Section: Status and Badge Classes
- **Filter by Status:**
  - Dropdown for All, Pending, In Progress, Delivered, Completed, Cancelled.
  - Actions**:** Apply filter

11.3 Order Details Page

- **Page:** `/order-details.php?id=[order_id]`
- **Access:** Only the order owner (client or assigned freelancer)
- **If not authorized:** Show error ("Order not found" or "Access denied")
- **Order Information:**
  - Order ID (prominent)
  - Service title & category
  - Order date & status (badge)

    - Expected and actual delivery dates
    - Pricing: service price, fee, total
- **Service Details:** Delivery time, revisions included
- **Client/Freelancer Info:** Name and profile photo of the other party
- **Service Requirements:** Requirements text, special instructions, deadline preference, and uploaded files (with icon, name, size, date, download), *File Display Requirements: detailed file display styling in section 10.6*
- **Order History:** View delivery files and revision history
- **If cancelled:** Show cancellation date and reason
- **Available Actions:** Vary by status and user role:
    - **Order Status: Pending**
        - **Client:**
            - Cancel Order (if status is Pending)
        - **Freelancer:**
            - Start Working
    - **Order Status: In Progress**
        - **Client:**
            - No actions (wait for delivery)
        - **Freelancer:**
            - Upload Delivery
    - **Order Status: Delivered**
        - **Client:**
            - Mark as Completed
            - Request Revision (if revisions available)
        - **Freelancer:**
            - No actions (wait for client approval or revision request)
    - **Order Status: Revision Requested**
        - **Client:**
            - View revision request status
            - No actions (wait for freelancer response)
        - **Freelancer:**
            - Accept Revision & Upload Revised Work
            - Reject Revision (with reason)
    - **Order Status: Completed**
        - **Client:**
            - View delivery files
        - **Freelancer:**
            - View delivery files and order history
    - **Order Status: Cancelled**
        - **Both:**
            - View only (no actions); see cancellation date and reason
    - **All Statuses:**
        - **Both:**

        - View delivery files
        - View order history

11.4 Cancel Order (Clients Only)

Cancellation Flow

- **Who can cancel:** Only the client who placed the order, and only if the order status is "Pending".
- **How to cancel:**
    1. Client clicks "Cancel Order" button.
    2. Redirect the client to the Order Cancellation page. This page displays a warning message, order details, and an optional field for the cancellation reason. The client must check a confirmation box and submit the cancellation request from this page.
    3. Client must check a confirmation box and submit.
    4. System validates the cancellation request: Checks that the order status is "Pending", Verifies the user is the order client, Confirms the checkbox is checked If valid, order status is updated to "Cancelled", refund is processed (simulated), and freelancer is notified via status update.
    5. Client is redirected to the orders page with a success message.
- **Restrictions:**
    - Cancellation is not allowed if the order is already "In Progress", "Delivered", "Revision Requested", "Completed", or "Cancelled".
    - Errors are shown if the action is not permitted or confirmation is missing.
- **After Cancellation:**
    - Freelancer sees the cancelled order in their list, with the cancellation date and reason.
    - Both parties can only view the order; no further actions are available.

11.5 Upload Delivery (Freelancers Only)

- **Who can upload:** Only the freelancer assigned to the order, when status is "In Progress" or "Revision Requested".
- **How to upload:**
    - Freelancer clicks "Upload Delivery".
    - Fills out a form with:
        - Delivery message (required, 50–500 characters)
        - 1–5 delivery files (max 50MB each, any format)
        - Optional notes
    - System validates inputs and uploads files.
        - Create directory: `/uploads/orders/[order_id]/deliverables/`

        - Move uploaded files to directory
        - For each file: Insert into file_attachments table: `order_id, file_path, original_filename, file_size, mime_type, file_type ('deliverable'), upload_timestamp`
    - Order status updates to "Delivered"; actual delivery date is recorded.
    - Client is notified and can review the delivery.
- **Restrictions:**
    - At least one file required, max five files.
    - Delivery message required.
    - Only available for eligible orders and assigned freelancers.

11.6 Mark as Completed (Clients Only)

- **Who can mark as completed:** Only the client who placed the order, when the order status is "Delivered".
- **How to mark as completed:**
    - Client reviews the delivered work.
    - Clicks "Mark as Completed".
    - Client is redirected to an **Order Completion Confirmation page**.
        - This page displays a message: "Are you sure you want to mark this order as completed? This action cannot be undone."
        - The client must confirm to proceed.
    - System updates the order status to "Completed".
    - Success message is displayed, and the client is redirected to the order details page.
    - "Leave Review" option becomes available.
- **Restrictions:**
    - Only available if the order status is "Delivered" and the user is the client.

11.7 Request Revision (Clients Only)

- **Who can request:** Only the client, when order status is "Delivered" and revisions are still available.
- **Revision count & validation:**
    - Each order has a set number of allowed revisions (e.g., 3), stored in the database.
    - Both accepted and rejected requests count toward this limit. If revisions_included = 999, always allow – unlimited.
    - Before allowing a new request, the system checks if the number of used revisions is less than the allowed maximum.
    - If the limit is reached, the option to request a revision is disabled and a message is shown: "You have used all X revision requests for this order."
- **How to request:**
    - Client clicks "Request Revision".

- A form appears showing revision limits and a required description field (50–500 characters). See Form Example below
- Client must check a confirmation box acknowledging the request counts toward their revision limit.
- System validates the request (description length, confirmation box, and available revisions).
- If valid, Insert Revision Request with Revision request status = "New", and update order status to "Revision Requested" and freelancer is notified.

- **Restrictions:**
  - Not available if all revisions are used.
  - Rejected requests also count toward the revision limit.
- **After request:**
  - Client is redirected to the order details page.
  - Freelancer sees the new revision request and can respond.

Request Revision Form Example

```
REQUEST REVISION

⚠ IMPORTANT NOTICE

This service includes 3 revision
requests.

• ALL requests count (accepted +
  rejected)
• Freelancer may reject if request is
  outside original scope
• Rejected requests still count toward
  your limit
• Be clear and specific

Revisions Used: 1/3
Revisions Remaining: 2

_____________________________________


Revision Description (required):
[Text area, 50-500 characters]

Be specific about what needs change:
• What element to modify
• How to modify it
• Why it needs changing

☐ I understand this request will count
  toward my revision limit

[Submit Request] [Cancel]
```

Example of Freelancer Order Details Page Shows

```
🔔 NEW REVISION REQUEST

Client Feedback:
"Please change logo colors to blue
 and increase font size by 2pt"

Requested: Dec 5, 2024 10:30 AM

Status: NEW - Awaiting Your Response

Client's Revisions: 2/3 used
(This is their 3rd and final request)

[Accept & Upload Revision]
[Reject Request]
```

Reject Revision Form Example

```
REJECT REVISION REQUEST

Reason for rejection (required):
[Text area, 50-500 characters]

Examples:
• Request outside original scope
• Work matches agreed specifications
• Additional work requires new order

[Submit Rejection] [Cancel]
```

11.8 Respond to Revision Request (Freelancers Only)

- **Who can respond:** Only the freelancer assigned to the order, when status is "Revision Requested" and the revision request status is "New".
- **How to respond:**
  - Freelancer views the revision request and client feedback.
  - Two options:
    - **Accept Revision & Upload Revised Work:**
      - Freelancer clicks "Accept & Upload Revision".
      - Upload revised files (same process as delivery).
      - Order status returns to "Delivered"; client is notified.
    - **Reject Revision:**
      - Freelancer clicks "Reject Request".
      - Completes a required rejection reason (50–500 characters), see Reject Revision Form Example above.
      - Order status returns to "Delivered"; client is notified, and rejection reason is displayed.
  - All responses (accepted or rejected) count toward the client's revision limit.
- **Status meanings:**
  - **New:** Just submitted, freelancer hasn't responded (does not count toward limit).
  - **Accepted:** Freelancer accepted, will provide revised work (counts toward limit).
  - **Rejected:** Freelancer rejected as unreasonable/out of scope (counts toward limit).
- **After response:**
  - Client can review revised work, mark as completed, or request another revision (if allowed).
  - Rejected requests also count toward the revision limit.

11.10 **Revision History Display.**

Where shown: On the *Order Details Page*

Revision History should display as a card with two components: a revision summary card and a history table. Summary card: shows total requests, how many were accepted, rejected, and pending and revisions remaining out of the allowed limit. The history table displays: Request number, Request date, Description, Status (New, Accepted, Rejected), Response (if any), Date of response.

Styling Requirements:

- Revision history: See CSS Styling Requirements document, Section: Page-Specific Styling - Order Details Page - Revision History Card